SIEMENS

AWL-Editor/ Batch-Compiler

STL Editor/ Batch-Compiler

Editeur LIST/ Compilateur Batch

Beschreibung
Description
Description

C79000-G8563-C677-02

SIEMENS

AWL-Editor/ Batch-Compiler

AWL-Editor/Batch-Compiler Beschreibung
Einführung 1
Beschreibung 2
Bedienung am Programmiergerät 3
Kommandozeilen-Version 4
Anhang 5

STL Editor/ Batch-Compiler

STL-Editor/Batch-Compiler Description
Introduction 1
Description 2
Operation of the Programming Unit 3
Command Line Version 4
Appendix 5

Editeur LIST/ Compilateur Batch

Beschreibung
Description
Description

Editeur LIST/Compilateur Batch Description
Introduction 1
Description 2
Commande à la console de programmation 3
Version lignes de commande 4
Annexe 5

C79000-G8563-C677-02

Haftungsausschluß

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so daß wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden jedoch regelmäßig überprüft und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen bleiben vorbehalten.

Copyright

Copyright © Siemens AG 1992 All Rights Reserved

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadensersatz. Alle Rechte vorbehalten, insbesondere für den Fall der Patenterteilung oder GM-Eintragung.

Haftungsausschluß

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so daß wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden jedoch regelmäßig überprüft und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen bleiben vorbehalten.

SIEMENS

AWL-Editor/ Batch-Compiler

Beschreibung

C79000-G8500-C677

Ausgabe 02

SIEMENS

AWL-Editor/ Batch-Compiler

Beschreibung

Einführung 1

Beschreibung 2

Bedienung am Programmiergerät 3

Kommandozeilen-Version 4

Anhang 5

C79000-G8500-C677

Ausgabe 02

Siemens Aktiengesellschaft

C79000-G8500-C677-01
EWK Elektronikwerk Karlsruhe

Printed in the Federal Republic of Germany

Siemens Aktiengesellschaft

C79000-G8500-C677-01
EWK Elektronikwerk Karlsruhe

Printed in the Federal Republic of Germany

Inhaltsverzeichnis

2.7 Eingabe von STEP 5-Anweisungen mit anderen Editoren ..... 2 - 24
2.7.1 AWL-Quelldatei als Schnittstelle ..... 2 - 24
2.7.2 Format der sequentiellen Quelldatei des Editors ..... 2 - 24
2.7.3 Steuerzeichen #TAB zur Verarbeitung von Fremddateien ..... 2 - 25

3 Bedienung am Programmiergerät ..... 3 - 1
3.1 Installieren der Software ..... 3 - 2
3.2 Bedienschritte bis zur Funktionsanwahl ..... 3 - 3
3.2.1 Allgemeines zur Bedienung ..... 3 - 3
3.2.2 Einstieg in das Paket AWL-Editor/Batch-Compiler ..... 3 - 3
3.3 Editieren ..... 3 - 8
3.3.1 Beschreibung des Editors ..... 3 - 9
3.3.2 Die Steuerzeichen des AWL-Editors/Batch-Compilers und ihre Schreibkonventionen ..... 3 - 14
3.3.3 Die STEP 5-Operationen im AWL-Editor/Batch-Compiler und ihre Schreibkonventionen ..... 3 - 20
3.3.4 Eingabe von Programmbausteinen ..... 3 - 25
3.3.5 Benutzen der Softkey-Leiste EDITIEREN ..... 3 - 30
3.3.6 Eingabe von Funktionsbausteinen ..... 3 - 41
3.3.7 Eingabe von Datenbausteinen ..... 3 - 48
3.3.8 Ändern einer AWL-Quelldatei ..... 3 - 51
3.4 Übersetzen mit der Funktion COMPILER ..... 3 - 52
3.4.1 Bedienfolge: Übersetzen in die Programmdatei ..... 3 - 52
3.4.2 Bedienfolge: Rückübersetzen der Programmdatei ..... 3 - 54
3.5 Fehlerliste ..... 3 - 55
3.6 Drucken ..... 3 - 56

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einführung ..... 1 - 1

2 Beschreibung ..... 2 - 1

2.1 Arbeitsweise des AWL-Editors/Batch-Compilers ..... 2 - 2

2.2 Erstellen von STEP 5-Bausteinen ..... 2 - 7

2.2.1 Allgemeines ..... 2 - 7

2.2.2 Editierfunktionen ..... 2 - 9

2.2.3 Steuerzeichen ..... 2 - 11

2.2.4 Übersetzen ..... 2 - 13

2.2.5 Drucken ..... 2 - 15

2.3 Die Zwischendatei A1.SEQ ..... 2 - 16

2.3.1 Zusammenhänge zwischen AWL-Quelldatei und Zwischendatei ..... 2 - 16

2.3.2 Sonderfunktionen ..... 2 - 18

2.3.3 Standardprogramme ..... 2 - 19

2.3.4 Fremdsprachige Versionen eines STEP 5-Programms ..... 2 - 20

2.4 Ändern und Ergänzen von STEP 5-Bausteinen ..... 2 - 21

2.4.1 Bausteine erstellt mit dem AWL-Editor ..... 2 - 21

2.4.2 Bausteine erstellt mit dem Paket KOP, FUP, AWL ..... 2 - 21

2.5 Prüflauf ..... 2 - 22

2.5.1 Prüfen einer Programmdatei ..... 2 - 22

2.5.2 Prüfen von Sonderbausteinen ..... 2 - 22

2.6 Fehlerliste ..... 2 - 23

2.6.1 Fehlerdatei ..... 2 - 23

Inhaltsverzeichnis

3.7 SONDERfunktionen zur Bearbeitung von Zwischen- und Quelldateien ... 3 - 57
3.7.1 KOPIEREN ... 3 - 57
3.7.2 SEQ>ZWI ... 3 - 58
3.7.3 ZWI>SEQ ... 3 - 58
3.7.4 SEQLOESCH und ZWILOESCH ... 3 - 59
3.7.5 PRUEFLAUF ... 3 - 59
3.7.6 SYM-GEN ... 3 - 60

4 Kommandozeilen-Version ... 4 - 1
4.1 Aufruf ... 4 - 3
4.1.1 Aufruf ohne Parameterangabe: ... 4 - 3
4.1.2 Aufruf mit einer Input-Datei ... 4 - 4
4.2 Format der Input-Datei ... 4 - 5
4.2.1 Quell- und Zieldateiangabe ... 4 - 6
4.2.2 Parameter $BAUST ... 4 - 6
4.2.3 Parameter $SYMB ... 4 - 6
4.2.4 Parameter $OPT ... 4 - 7
4.2.5 Parameter $AGTYP ... 4 - 7
4.2.6 Parameter $DRU ... 4 - 7
4.3 Default-Einstellungen der Parameter ... 4 - 8

5 Anhang ... 5 - 1
Fehlermeldungen ... 5 - 2

C79000-B8500-C677-02 0 - 5

Inhaltsverzeichnis

0 - 6

C79000-B8500-C677-02

Einführung 1

Einführung 1

C79000-B8500-C677-02 1 - 1

Funktionsumfang

Das Optionspaket AWL-Editor/Batch-Compiler bietet Ihnen einen eigenständigen Editor für Programme in der Darstellungsart AWL und einen eigenständigen Compiler für die Übersetzung solcher Anweisungslisten in ein ablauffähiges STEP 5-Programm.

Bisher konnten Sie ein STEP 5-Programm nur innerhalb des Basispakets im Paket KOP, FUP, AWL erstellen. Der Editierkomfort dieses Pakets ist aber eingeschränkt, und es wird vorausgesetzt, daß Sie die Verdrahtung Ihrer Anlage bereits kennen, um die Ein-/Ausgänge richtig zu schalten.

Der AWL-Editor bietet Ihnen nun die Möglichkeit, mit hohem Bedienkomfort Ihr Programm zu erstellen, z.B. können Sie hier Netzwerke kopieren. Ebenso erlaubt dieses Paket, das Programm ausschließlich symbolisch zu schreiben. Sie brauchen aber nicht vorher eine definitive Zuordnung von Symbol und Ein-/Ausgang zu treffen, sondern können die Zuordnungen zu einem späteren Zeitpunkt nachliefern, z.B. wenn Ihre Anlage fertiggestellt ist. Diese Zuordnungen werden dann mittels des Batch-Compilers mit der im Editor erstellten Anweisungsliste verbunden und in ein STEP 5-Programm übersetzt, das auf Ihrer Anlage abläuft. Dieses Programm können Sie im Automatisierungsgerät testen und korrigieren.

Mit dem Batch-Compiler werden auch Rückübersetzungen aus einem STEP 5-Programm durchgeführt, so daß z.B. die Änderungen des getesteten Programms in Ihre Quelle eingetragen werden können und Ihre Anweisungsliste aktualisiert ist. In der gleichen Weise kann ein STEP 5-Programm behandelt werden, das im Paket KOP, FUP, AWL erstellt worden ist.

Das Optionspaket AWL-Editor/Batch-Compiler bietet Ihnen also viele Vorteile:

1. *Sie können mit Symbolen ein* **aufgaben***spezifisches Programm erstellen, das mit einer individuellen Zuordnungsliste zu einem* **anlagen***spezifischen STEP 5-Programm wird. D.h., Sie können eine Bibliothek standardisierter Programme aufbauen.*
2. *Mit einem Include-Befehl können standardisierte Programme und Module immer wieder neu kombiniert und zu einem individuellen STEP 5-Programm verbunden werden, ohne daß Sie in den Programmen selbst umfangreiche Änderungen durchführen müssen.*

Einführung

Funktionsumfang

Das Optionspaket AWL-Editor/Batch-Compiler bietet Ihnen einen eigenständigen Editor für Programme in der Darstellungsart AWL und einen eigenständigen Compiler für die Übersetzung solcher Anweisungslisten in ein ablauffähiges STEP 5-Programm.

Bisher konnten Sie ein STEP 5-Programm nur innerhalb des Basispakets im Paket KOP, FUP, AWL erstellen. Der Editierkomfort dieses Pakets ist aber eingeschränkt, und es wird vorausgesetzt, daß Sie die Verdrahtung Ihrer Anlage bereits kennen, um die Ein-/Ausgänge richtig zu schalten.

Der AWL-Editor bietet Ihnen nun die Möglichkeit, mit hohem Bedienkomfort Ihr Programm zu erstellen, z.B. können Sie hier Netzwerke kopieren. Ebenso erlaubt dieses Paket, das Programm ausschließlich symbolisch zu schreiben. Sie brauchen aber nicht vorher eine definitive Zuordnung von Symbol und Ein-/Ausgang zu treffen, sondern können die Zuordnungen zu einem späteren Zeitpunkt nachliefern, z.B. wenn Ihre Anlage fertiggestellt ist. Diese Zuordnungen werden dann mittels des Batch-Compilers mit der im Editor erstellten Anweisungsliste verbunden und in ein STEP 5-Programm übersetzt, das auf Ihrer Anlage abläuft. Dieses Programm können Sie im Automatisierungsgerät testen und korrigieren.

Mit dem Batch-Compiler werden auch Rückübersetzungen aus einem STEP 5-Programm durchgeführt, so daß z.B. die Änderungen des getesteten Programms in Ihre Quelle eingetragen werden können und Ihre Anweisungsliste aktualisiert ist. In der gleichen Weise kann ein STEP 5-Programm behandelt werden, das im Paket KOP, FUP, AWL erstellt worden ist.

Das Optionspaket AWL-Editor/Batch-Compiler bietet Ihnen also viele Vorteile:

1. *Sie können mit Symbolen ein* **aufgaben***spezifisches Programm erstellen, das mit einer individuellen Zuordnungsliste zu einem* **anlagen***spezifischen STEP 5-Programm wird. D.h., Sie können eine Bibliothek standardisierter Programme aufbauen.*
2. *Mit einem Include-Befehl können standardisierte Programme und Module immer wieder neu kombiniert und zu einem individuellen STEP 5-Programm verbunden werden, ohne daß Sie in den Programmen selbst umfangreiche Änderungen durchführen müssen.*

1 - 2

C79000-B8500-C677-02

3. *Im AWL-Editor haben Sie einen wesentlich höheren Bedienkomfort als im Paket KOP, FUP, AWL. Sie können hier Netzwerke, Programmteile und Bausteine kopieren und verschieben. In der Handhabung am Programmiergerät gleicht der AWL-Editor dem Symbolik-Editor.*
4. *Häufig wiederkehrende Programmteile können auf Dateien ausgelagert werden und lassen sich mit Hilfe von Kopier- oder Include-Befehlen an jede Stelle Ihrer aufgeschlagenen oder einer anderen Datei außerhalb eines Bausteins einbinden.*
5. *Sie können Ihre Anweisungsliste auch mit einem anderen Texteditor erstellen, denn der Batch-Compiler kann sequentielle Dateien (ASCII-Dateien) in ein STEP 5-Programm übersetzen. So sind Sie offen zu anderen Systemen.*
6. *Weil die Übersetzung über eine Zwischendatei erfolgt, kann Ihre Anweisungsliste auch mit einem fremdsprachigen AWL-Editor bearbeitet werden.*
7. *Bereits bestehende STEP 5-Programme, erstellt im Paket KOP, FUP, AWL, werden mit dem Batch-Compiler rückübersetzt und stehen Ihnen somit ebenfalls als Grundlage für Standardprogramme und fremdsprachige Versionen zur Verfügung.*

Als weitere Funktionen bietet dieses Optionspaket einen AG-spezifischen Prüflauf für das übersetzte STEP 5-Programm und eine Fehlerliste.

In den folgenden Kapiteln finden Sie eine detaillierte Beschreibung des AWL-Editors und des Batch-Compilers, die Ihnen die Arbeitsweise und die verschiedenen Funktionen erklärt. Eine Bedienungsanleitung führt Sie an einem praktischen Beispiel in die Handhabung am Programmiergerät ein. Die Bedienungsanleitung enthält Übersichtstabellen zu Steuerzeichen, STEP 5-Operationen und den Schreibkonventionen.

Zur besseren Lesbarkeit markieren wir Wörter

- **fett**, wenn von Tasten des Geräts die Rede ist,
- *kursiv*, wenn reine Funktionen der Software genannt sind (z.B. Übernahme) und wenn Sie selbst Text eingeben sollen (im Beispiel).

Einführung

1

3. *Im AWL-Editor haben Sie einen wesentlich höheren Bedienkomfort als im Paket KOP, FUP, AWL. Sie können hier Netzwerke, Programmteile und Bausteine kopieren und verschieben. In der Handhabung am Programmiergerät gleicht der AWL-Editor dem Symbolik-Editor.*
4. *Häufig wiederkehrende Programmteile können auf Dateien ausgelagert werden und lassen sich mit Hilfe von Kopier- oder Include-Befehlen an jede Stelle Ihrer aufgeschlagenen oder einer anderen Datei außerhalb eines Bausteins einbinden.*
5. *Sie können Ihre Anweisungsliste auch mit einem anderen Texteditor erstellen, denn der Batch-Compiler kann sequentielle Dateien (ASCII-Dateien) in ein STEP 5-Programm übersetzen. So sind Sie offen zu anderen Systemen.*
6. *Weil die Übersetzung über eine Zwischendatei erfolgt, kann Ihre Anweisungsliste auch mit einem fremdsprachigen AWL-Editor bearbeitet werden.*
7. *Bereits bestehende STEP 5-Programme, erstellt im Paket KOP, FUP, AWL, werden mit dem Batch-Compiler rückübersetzt und stehen Ihnen somit ebenfalls als Grundlage für Standardprogramme und fremdsprachige Versionen zur Verfügung.*

Als weitere Funktionen bietet dieses Optionspaket einen AG-spezifischen Prüflauf für das übersetzte STEP 5-Programm und eine Fehlerliste.

In den folgenden Kapiteln finden Sie eine detaillierte Beschreibung des AWL-Editors und des Batch-Compilers, die Ihnen die Arbeitsweise und die verschiedenen Funktionen erklärt. Eine Bedienungsanleitung führt Sie an einem praktischen Beispiel in die Handhabung am Programmiergerät ein. Die Bedienungsanleitung enthält Übersichtstabellen zu Steuerzeichen, STEP 5-Operationen und den Schreibkonventionen.

Zur besseren Lesbarkeit markieren wir Wörter

- **fett**, wenn von Tasten des Geräts die Rede ist,
- *kursiv*, wenn reine Funktionen der Software genannt sind (z.B. Übernahme) und wenn Sie selbst Text eingeben sollen (im Beispiel).

C79000-B8500-C677-02

1 - 3

Einführung

1 - 4

C79000-B8500-C677-02

Beschreibung 2

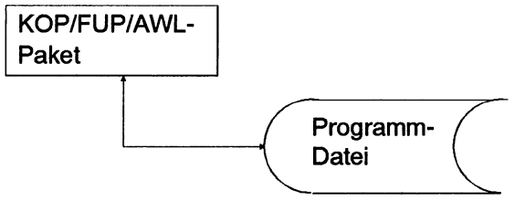
Beschreibung 2

C79000-B8500-C677-02 2 - 1

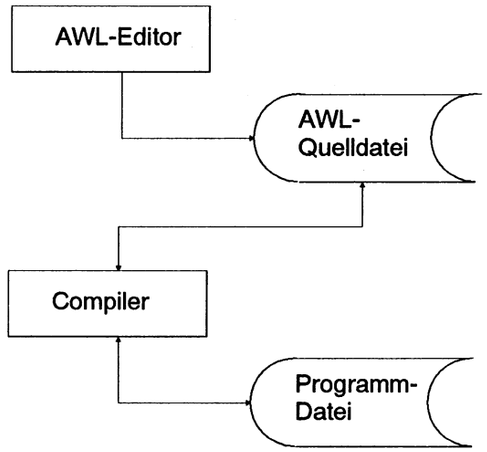
2.1 Arbeitsweise des AWL-Editors/Batch-Compilers

Die Erstellung eines STEP 5-Programms im AWL-Editor/Batch-Compiler unterscheidet sich vom Paket KOP, FUP, AWL in folgendem Punkt:

Im Paket KOP, FUP, AWL wird die Anweisungsliste gleich in der Programmdatei editiert und sofort in den Maschinencode übersetzt.



Im Paket AWL-Editor/Batch-Compiler sind *Editieren* und *Übersetzen* zeitlich getrennte Vorgänge.

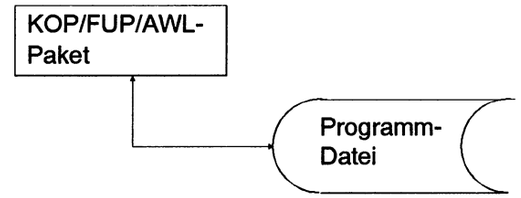


Beschreibung

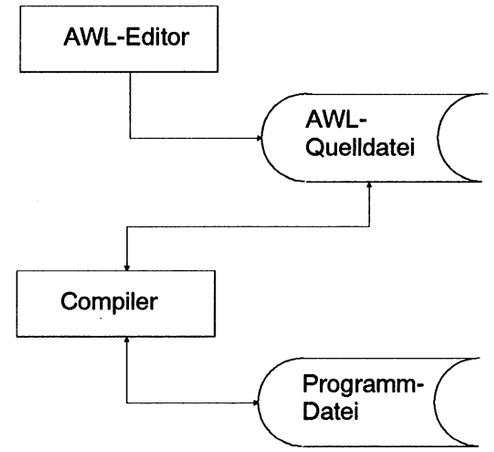
2.1 Arbeitsweise des AWL-Editors/Batch-Compilers

Die Erstellung eines STEP 5-Programms im AWL-Editor/Batch-Compiler unterscheidet sich vom Paket KOP, FUP, AWL in folgendem Punkt:

Im Paket KOP, FUP, AWL wird die Anweisungsliste gleich in der Programmdatei editiert und sofort in den Maschinencode übersetzt.



Im Paket AWL-Editor/Batch-Compiler sind *Editieren* und *Übersetzen* zeitlich getrennte Vorgänge.

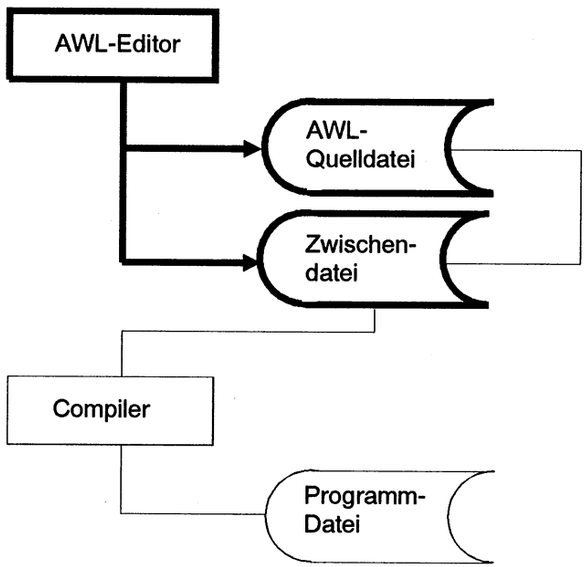


Editieren

Beim ersten Schritt, dem *Editieren*, schreiben Sie mit dem AWL-Editor eine sequentielle Textdatei, die AWL-Quelldatei. Sie kann eine Anweisungsliste enthalten, die ausschließlichlich mit Symbolen erstellt ist.

Übernehmen

Beim Abspeichern mit der Funktion *Übernahme* oder der **Übernahmetaste** legt das Paket automatisch neben der AWL-Quelldatei eine Zwischendatei an. Diese enthält einen Code, der unabhängig von Nationalsprachen (sprachunabhängig), aber noch kein Maschinencode ist. Bei dieser ersten *Übersetzung* wird Ihre Anweisungsliste auf Syntax und Format geprüft.



Beschreibung

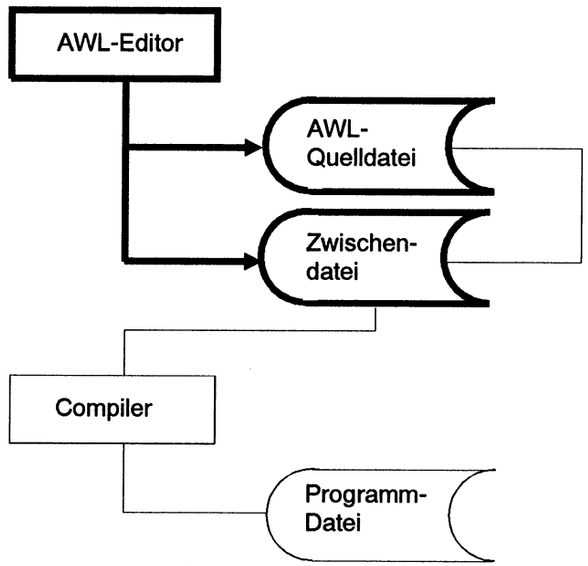
Editieren

Beim ersten Schritt, dem *Editieren*, schreiben Sie mit dem AWL-Editor eine sequentielle Textdatei, die AWL-Quelldatei. Sie kann eine Anweisungsliste enthalten, die ausschließlichlich mit Symbolen erstellt ist.

Übernehmen

Beim Abspeichern mit der Funktion *Übernahme* oder der **Übernahmetaste** legt das Paket automatisch neben der AWL-Quelldatei eine Zwischendatei an. Diese enthält einen Code, der unabhängig von Nationalsprachen (sprachunabhängig), aber noch kein Maschinencode ist. Bei dieser ersten *Übersetzung* wird Ihre Anweisungsliste auf Syntax und Format geprüft.

2

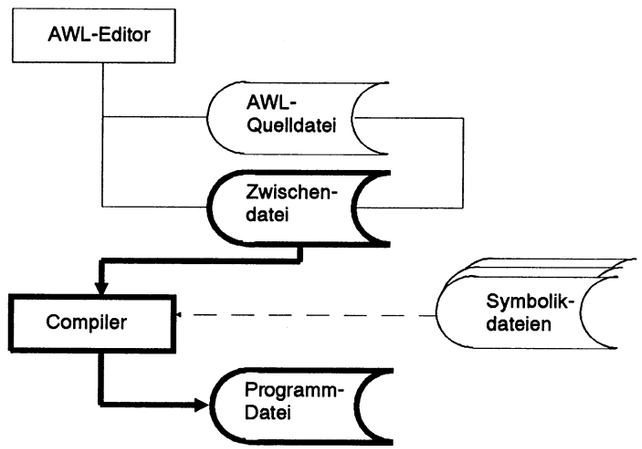


Übersetzung

Der zweite Schritt, die *Übersetzung*, wird von Ihnen selbst über eine Funktionstaste angestoßen. Dabei führt der Batch-Compiler die Zwischendatei über in eine STEP 5-Programmdatei. Haben Sie Ihre Anweisungsliste symbolisch programmiert, braucht der Batch-Compiler an dieser Stelle eine Symbolikdatei mit anlagenspezifischen Zuordnungen.

Prüfung

Bei der *Übersetzung* in die Programmdatei erfolgt eine *Prüfung* der Zuordnungen. Haben Sie einen bestimmten AG-Typ angegeben, wird auch geprüft, ob die verwendeten Operationen für Ihr Ziel-AG zulässig sind (*AG-spezifische Prüfung*). Eine mit dem AWL-Editor/Batch-Compiler erstellte Programmdatei ist identisch mit einer im Paket KOP, FUP, AWL erstellten Programmdatei.



Bemerkung: Die Übersetzung der AWL-Quelldatei kann auch in einem Schritt ausgeführt werden, z.B. wenn die Quelle nur gesichert, aber keine Zwischendatei erzeugt wurde, oder aber wenn die Quelle mit einem Fremdeditor erstellt wurde.

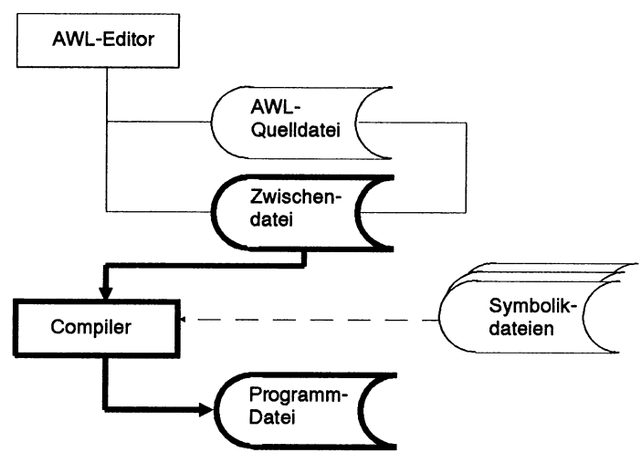
Beschreibung

Übersetzung

Der zweite Schritt, die *Übersetzung*, wird von Ihnen selbst über eine Funktionstaste angestoßen. Dabei führt der Batch-Compiler die Zwischendatei über in eine STEP 5-Programmdatei. Haben Sie Ihre Anweisungsliste symbolisch programmiert, braucht der Batch-Compiler an dieser Stelle eine Symbolikdatei mit anlagenspezifischen Zuordnungen.

Prüfung

Bei der *Übersetzung* in die Programmdatei erfolgt eine *Prüfung* der Zuordnungen. Haben Sie einen bestimmten AG-Typ angegeben, wird auch geprüft, ob die verwendeten Operationen für Ihr Ziel-AG zulässig sind (*AG-spezifische Prüfung*). Eine mit dem AWL-Editor/Batch-Compiler erstellte Programmdatei ist identisch mit einer im Paket KOP, FUP, AWL erstellten Programmdatei.

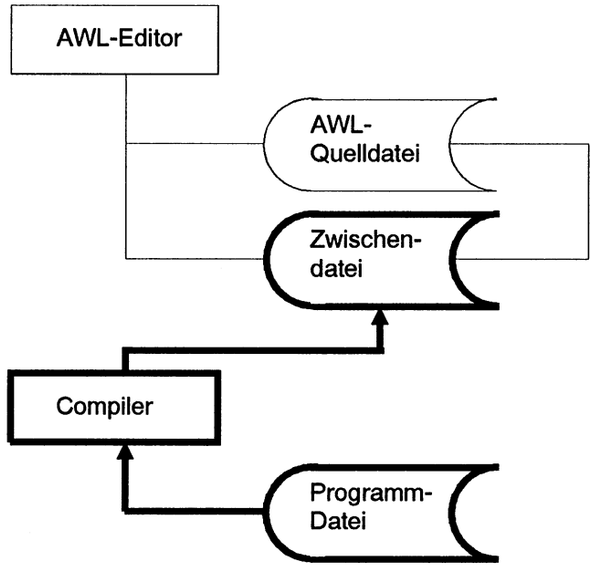


Bemerkung: Die Übersetzung der AWL-Quelldatei kann auch in einem Schritt ausgeführt werden, z.B. wenn die Quelle nur gesichert, aber keine Zwischendatei erzeugt wurde, oder aber wenn die Quelle mit einem Fremdeditor erstellt wurde.

Aus einer Programmdatei können Sie mit dem AWL-Editor/Batch-Compiler auch eine Quelldatei erstellen, z.B. kann dies nötig sein, nachdem ein STEP 5-Programm im AG getestet und korrigiert worden ist. Dabei spielt es keine Rolle, ob das Programm im Paket KOP, FUP, AWL oder im Paket AWL-Editor/Batch-Compiler editiert worden ist.

Rückübersetzen

Bei einer solchen *Rückübersetzung* wird vom Batch-Compiler zunächst aus der Programmdatei eine Zwischendatei generiert. Aus dieser Zwischendatei wird dann die AWL-Quelldatei zur Programmdatei gebildet.



Bemerkung: Die Erzeugung einer AWL-Quelldatei aus einer Programmdatei kann auch direkt in einem Schritt erfolgen.

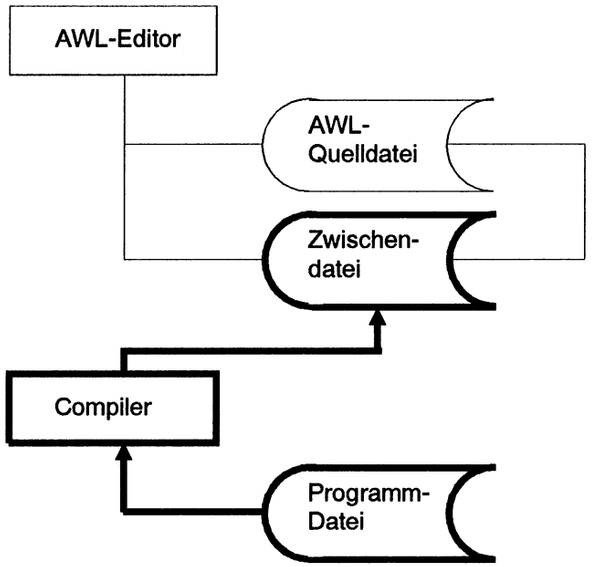
Beschreibung

Aus einer Programmdatei können Sie mit dem AWL-Editor/Batch-Compiler auch eine Quelldatei erstellen, z.B. kann dies nötig sein, nachdem ein STEP 5-Programm im AG getestet und korrigiert worden ist. Dabei spielt es keine Rolle, ob das Programm im Paket KOP, FUP, AWL oder im Paket AWL-Editor/Batch-Compiler editiert worden ist.

Rückübersetzen

Bei einer solchen *Rückübersetzung* wird vom Batch-Compiler zunächst aus der Programmdatei eine Zwischendatei generiert. Aus dieser Zwischendatei wird dann die AWL-Quelldatei zur Programmdatei gebildet.

2

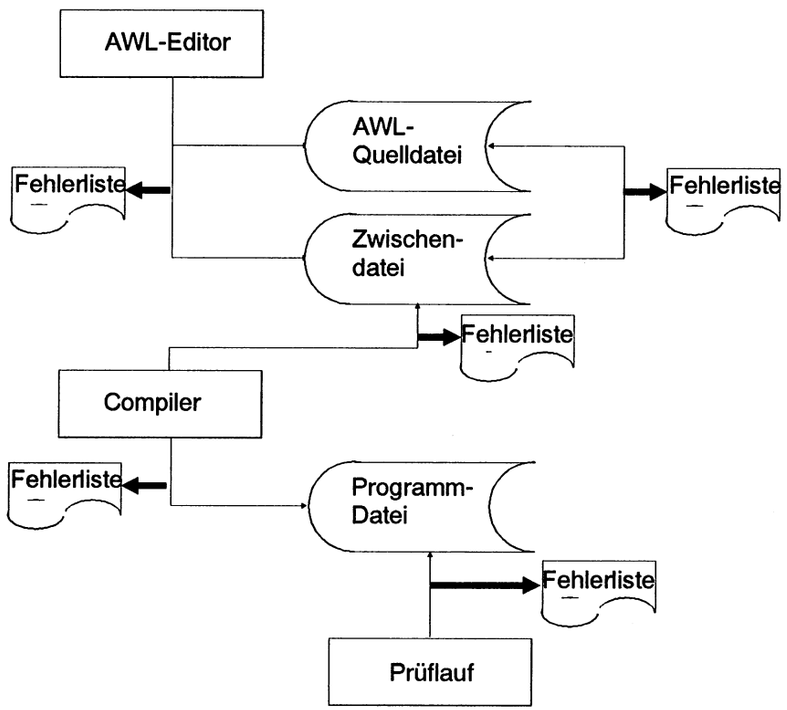


Bemerkung: Die Erzeugung einer AWL-Quelldatei aus einer Programmdatei kann auch direkt in einem Schritt erfolgen.

Prüflauf

Wie bereits erwähnt, werden beim *Übersetzen Prüfungen* durchgeführt. Zudem gibt es nach der *Übersetzung* einen *Prüflauf für die Bausteine der Programmdatei*, währenddessen z.B. überprüft wird, ob Formaloperanden und Aktualoperanden bei Funktionsbausteinen richtig zugeordnet sind und übereinstimmen. Alle auftretenden Fehler werden in einer Fehlerliste gesammelt und können auf den Drucker ausgegeben werden.

In der Fehlerliste sind jedoch immer nur die Fehler des letzten Arbeitsschritts enthalten, sie wird bei jeder neuen *Übersetzung* bzw. *Prüfung* überschrieben. Geben Sie deshalb Ihre Fehlerliste immer auf den Drucker aus! Ist ein Arbeitsschritt fehlerfrei abgelaufen wird keine Fehlerliste angelegt bzw. eine eventuell existierende gelöscht!

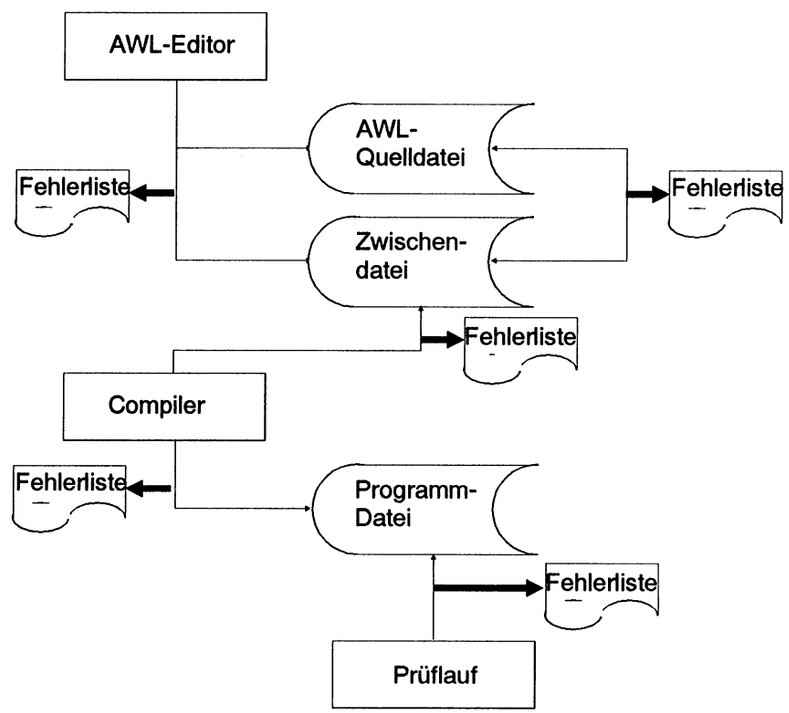


Beschreibung

Prüflauf

Wie bereits erwähnt, werden beim *Übersetzen Prüfungen* durchgeführt. Zudem gibt es nach der *Übersetzung* einen *Prüflauf für die Bausteine der Programmdatei*, währenddessen z.B. überprüft wird, ob Formaloperanden und Aktualoperanden bei Funktionsbausteinen richtig zugeordnet sind und übereinstimmen. Alle auftretenden Fehler werden in einer Fehlerliste gesammelt und können auf den Drucker ausgegeben werden.

In der Fehlerliste sind jedoch immer nur die Fehler des letzten Arbeitsschritts enthalten, sie wird bei jeder neuen *Übersetzung* bzw. *Prüfung* überschrieben. Geben Sie deshalb Ihre Fehlerliste immer auf den Drucker aus! Ist ein Arbeitsschritt fehlerfrei abgelaufen wird keine Fehlerliste angelegt bzw. eine eventuell existierende gelöscht!


AWL-Editor
AWL-
Quelldatei
Fehlerliste
Fehlerliste
Zwischen-
datei
Fehlerliste
Compiler
Fehlerliste
Programm-
Datei
Fehlerliste
Prüflauf

2.2 Erstellen von STEP 5-Bausteinen

2.2.1 Allgemeines

Im AWL-Editor erstellen Sie mit dem Komfort eines Texteditors Ihr Steuerungsprogramm als Anweisungsliste. Sie schreiben Ihre Anweisungsliste mit dem gleichen Befehlsvorrat und in der gleichen Syntax wie im Paket KOP, FUP, AWL. Einziger Unterschied ist, daß Sie gewisse Konventionen einhalten müssen, wie z.B. Steuerzeichen für Bausteinanfänge und Kommentare.

2

Alle Bausteinarten, die im Paket KOP, FUP, AWL möglich sind, können auch mit dem Paket AWL-Editor/Batch-Compiler erstellt werden. Ebenso können alle Kommentare, die in einem Baustein möglich sind, geschrieben werden, jedoch keine Anlagenkommentare (DOK-Datei). Zusätzlich dazu erlaubt der AWL-Editor sogenannte Zusatzkommentare an jeder Stelle der Anweisungsliste. Diese Kommentare werden allerdings nicht in die Programmdatei übertragen und gehen bei einer Rückübersetzung in die gleiche AWL-Quelle verloren.

Nicht möglich sind ein DB0 (reseviert im AG für die Bausteinadressliste), ein DB1 (für die Peripheriezuteilung des AG 135U und AG 155U), ein DX0 (für die Systemparametrierung, CPU 928, R-Prozessor, AG 155U), ein DB2 (für die Reglerliste des R64), GRAPH 5- und Assembler-Bausteine.

Wie Sie beim Editieren genau vorgehen, und welche Steuerzeichen und Schreibkonventionen dabei beachtet werden müssen, finden Sie in der Bedienungsanleitung, Kapitel 3, erläutert.

Ihre Anweisungen werden nicht sofort in die AWL-Quelldatei auf Festplatte oder Diskette geschrieben, sondern in einen Editierpuffer im Arbeitsspeicher Ihres Programmiergerätes. Der Inhalt des Editierpuffers, also Ihre Anweisungsliste, wird erst mit den Speicherfunktionen *Sichern* oder *Übernehmen* in die AWL-Quelldatei abgespeichert. Beachte Sie dabei folgende Unterschiede:

- Mit *Sichern* speichern Sie nur Ihre AWL-Quelldatei.
- Mit *Übernehmen* speichern Sie Ihre AWL-Quelldatei *und* übersetzen sie in die Zwischendatei.
- Mit *Abbruch* (Abbruchtaste) verwerfen Sie die gesamte im Arbeitsspeicher editierte Datei und verlassen den Editor.

Beschreibung

2.2 Erstellen von STEP 5-Bausteinen

2.2.1 Allgemeines

Im AWL-Editor erstellen Sie mit dem Komfort eines Texteditors Ihr Steuerungsprogramm als Anweisungsliste. Sie schreiben Ihre Anweisungsliste mit dem gleichen Befehlsvorrat und in der gleichen Syntax wie im Paket KOP, FUP, AWL. Einziger Unterschied ist, daß Sie gewisse Konventionen einhalten müssen, wie z.B. Steuerzeichen für Bausteinanfänge und Kommentare.

2

Alle Bausteinarten, die im Paket KOP, FUP, AWL möglich sind, können auch mit dem Paket AWL-Editor/Batch-Compiler erstellt werden. Ebenso können alle Kommentare, die in einem Baustein möglich sind, geschrieben werden, jedoch keine Anlagenkommentare (DOK-Datei). Zusätzlich dazu erlaubt der AWL-Editor sogenannte Zusatzkommentare an jeder Stelle der Anweisungsliste. Diese Kommentare werden allerdings nicht in die Programmdatei übertragen und gehen bei einer Rückübersetzung in die gleiche AWL-Quelle verloren.

Nicht möglich sind ein DB0 (reseviert im AG für die Bausteinadressliste), ein DB1 (für die Peripheriezuteilung des AG 135U und AG 155U), ein DX0 (für die Systemparametrierung, CPU 928, R-Prozessor, AG 155U), ein DB2 (für die Reglerliste des R64), GRAPH 5- und Assembler-Bausteine.

Wie Sie beim Editieren genau vorgehen, und welche Steuerzeichen und Schreibkonventionen dabei beachtet werden müssen, finden Sie in der Bedienungsanleitung, Kapitel 3, erläutert.

Ihre Anweisungen werden nicht sofort in die AWL-Quelldatei auf Festplatte oder Diskette geschrieben, sondern in einen Editierpuffer im Arbeitsspeicher Ihres Programmiergerätes. Der Inhalt des Editierpuffers, also Ihre Anweisungsliste, wird erst mit den Speicherfunktionen *Sichern* oder *Übernehmen* in die AWL-Quelldatei abgespeichert. Beachte Sie dabei folgende Unterschiede:

- Mit *Sichern* speichern Sie nur Ihre AWL-Quelldatei.
- Mit *Übernehmen* speichern Sie Ihre AWL-Quelldatei *und* übersetzen sie in die Zwischendatei.
- Mit *Abbruch* (Abbruchtaste) verwerfen Sie die gesamte im Arbeitsspeicher editierte Datei und verlassen den Editor.

C79000-B8500-C677-02 2 - 7

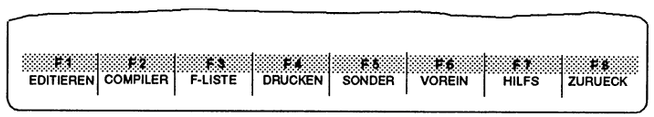
Anmerkung: Obwohl der Editierpuffer nur eine bestimmte, endliche Größe hat (diese Größe wird zu Ihrer Information in der Editiermaske eingeblendet), können Sie weitaus größere Anweisungslisten bearbeiten. Zu diesem Zweck legt der AWL-Editor temporäre Dateien (A0.TM0, A0.TM1, A0.TM2) an. Für diese Dateien wird entsprechend Platz auf dem externen Datenträger (im allgemeinen Festplatte) benötigt. Ist dieser Platz nicht ausreichend verfügbar, erfolgt eine Fehlermeldung. Nach *Sichern* bzw. *Übernehmen* werden diese temporären Dateien gelöscht. In Ausnahmefällen (z.B. Netzausfall während des Editierens) bleiben diese Dateien bestehen; sie dürfen jederzeit von Ihnen gelöscht werden.

In der Voreinstellung werden vier Dateien festgelegt:

- die AWL-Quelldatei, die Sie editieren wollen (A0.SEQ);
- die Zwischendatei, die beim Speichern mit der Übernahmetaste entsteht und die in den Zwischencode übersetzte Anweisungsliste enthält (A1.SEQ);
- die Symbolikdatei, in der eine Zuordnungsliste steht (Z0.INI), und
- die Programmdatei, in die nach dem Übersetzen das STEP 5-Programm geschrieben werden soll (ST.S5D).

Diese vier Dateien werden automatisch mit gleichem Namen eingetragen und können gegebenfalls geändert werden. AWL-Quelldatei und Zwischendatei haben aber immer den gleichen Namen.

In der Funktionsanwahl bietet Ihnen das Paket AWL-Editor/Batch-Compiler als Funktionen an:



EDITIEREN zum Erstellen und Bearbeiten der AWL-Quelldatei; (Bedienung siehe Kap. 3.3)

COMPILER zum Übersetzen und Rückübersetzen; (Bedienung siehe Kap. 3.4)

Beschreibung

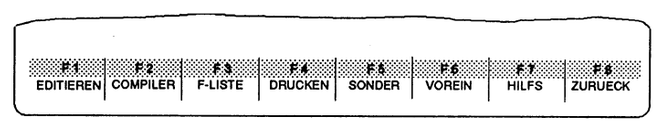
Anmerkung: Obwohl der Editierpuffer nur eine bestimmte, endliche Größe hat (diese Größe wird zu Ihrer Information in der Editiermaske eingeblendet), können Sie weitaus größere Anweisungslisten bearbeiten. Zu diesem Zweck legt der AWL-Editor temporäre Dateien (A0.TM0, A0.TM1, A0.TM2) an. Für diese Dateien wird entsprechend Platz auf dem externen Datenträger (im allgemeinen Festplatte) benötigt. Ist dieser Platz nicht ausreichend verfügbar, erfolgt eine Fehlermeldung. Nach *Sichern* bzw. *Übernehmen* werden diese temporären Dateien gelöscht. In Ausnahmefällen (z.B. Netzausfall während des Editierens) bleiben diese Dateien bestehen; sie dürfen jederzeit von Ihnen gelöscht werden.

In der Voreinstellung werden vier Dateien festgelegt:

- die AWL-Quelldatei, die Sie editieren wollen (A0.SEQ);
- die Zwischendatei, die beim Speichern mit der Übernahmetaste entsteht und die in den Zwischencode übersetzte Anweisungsliste enthält (A1.SEQ);
- die Symbolikdatei, in der eine Zuordnungsliste steht (Z0.INI), und
- die Programmdatei, in die nach dem Übersetzen das STEP 5-Programm geschrieben werden soll (ST.S5D).

Diese vier Dateien werden automatisch mit gleichem Namen eingetragen und können gegebenfalls geändert werden. AWL-Quelldatei und Zwischendatei haben aber immer den gleichen Namen.

In der Funktionsanwahl bietet Ihnen das Paket AWL-Editor/Batch-Compiler als Funktionen an:



EDITIEREN zum Erstellen und Bearbeiten der AWL-Quelldatei; (Bedienung siehe Kap. 3.3)

COMPILER zum Übersetzen und Rückübersetzen; (Bedienung siehe Kap. 3.4)

F-LISTE die Fehlerliste der Prüfläufe; (Bedienung siehe Kap. 3.5)

DRUCKEN zum Drucken der AWL-Quelldatei; (Bedienung siehe Kap. 3.6)

SONDER funktionen, mit denen Zwischendateien und AWL-Quelldatei generiert werden können; (Bedienung siehe Kap. 3.7)

VOREIN zum Ändern der Voreinstellung;

HILFS funktionen, zur Verwaltung von Bausteinen in der voreingestellten Programmdatei und

ZURUECK zum Verlassen des Pakets AWL-Editor/Batch-Compiler.

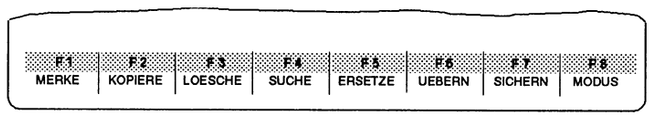
In den folgenden Abschnitten werden diese Funktionen im Überblick kurz beschrieben.

2.2.2 Editierfunktionen

Der AWL-Editor gibt eine Editiermaske auf dem Bildschirm aus, die für eine Anweisungsliste vorbereitet ist. Die Editiermaske besteht aus:

- einer Kopfzeile mit dem Namen der AWL-Quelldatei,
- die in Spalten angeordneten Eingabefelder für ADRESSE, ANWEISUNG, OPERANDENSYMBOL und ANWEISUNGS-KOMMENTAR,
- das Menü mit den Editierfunktionen.

Er bietet Ihnen eine Reihe von Funktionen, mit denen Sie Ihr Programm komfortabel editieren können. Sie sind mit denen des Symbolik-Editors vergleichbar:



Beschreibung

F-LISTE die Fehlerliste der Prüfläufe; (Bedienung siehe Kap. 3.5)

DRUCKEN zum Drucken der AWL-Quelldatei; (Bedienung siehe Kap. 3.6)

SONDER funktionen, mit denen Zwischendateien und AWL-Quelldatei generiert werden können; (Bedienung siehe Kap. 3.7)

VOREIN zum Ändern der Voreinstellung;

HILFS funktionen, zur Verwaltung von Bausteinen in der voreingestellten Programmdatei und

ZURUECK zum Verlassen des Pakets AWL-Editor/Batch-Compiler.

2

In den folgenden Abschnitten werden diese Funktionen im Überblick kurz beschrieben.

2.2.2 Editierfunktionen

Der AWL-Editor gibt eine Editiermaske auf dem Bildschirm aus, die für eine Anweisungsliste vorbereitet ist. Die Editiermaske besteht aus:

- einer Kopfzeile mit dem Namen der AWL-Quelldatei,
- die in Spalten angeordneten Eingabefelder für ADRESSE, ANWEISUNG, OPERANDENSYMBOL und ANWEISUNGS-KOMMENTAR,
- das Menü mit den Editierfunktionen.

Er bietet Ihnen eine Reihe von Funktionen, mit denen Sie Ihr Programm komfortabel editieren können. Sie sind mit denen des Symbolik-Editors vergleichbar:

F1 MERKE | F2 KOPIERE | F3 LOESCHE | F4 SUCHE | F5 ERSETZE | F6 UEBERN | F7 SICHERN | F8 MODUS

C79000-B8500-C677-02

2 - 9

Die **MERKE**- und **KOPIERE**-Funktion ermöglichen, beliebige Zeichenfolgen und Textblöcke in einen Puffer oder auf eine sequentielle Datei zu schreiben ("auslagern") und sie an eine beliebige Stelle zu kopieren. So können Sie z.B. Netzwerke verschieben oder immer wieder neu ins Programm einbinden.

Zusätzlich können Sie mit der Kopierfunktion auch andere AWL-Quelldateien oder einzelne Bausteine daraus einlesen. Merk-, Kopier- und Löschfunktion können mit einem Wiederholungsfaktor kombiniert werden.

Die Funktionen **SUCHE** und **ERSETZE** erleichtern die Korrekturen Ihres Programms:

Sie können sich schnell und gezielt in Ihrer Datei bewegen. Sie können einzelne Zeichenfolgen, wie z.B. Symbole oder Operanden, mit einer Funktion in Ihrer gesamten Anweisungsliste ändern. Außerdem werden Ihnen Einfüge- und Überschreib**modus** wahlweise angeboten.

Mit **SICHERN** haben Sie, wie bereits erwähnt, die Möglichkeit, Ihre Datei zu speichern, ohne den Editor zu verlassen. Bei jeder kurzzeitigen Unterbrechung einer Editiersitzung ist ein solches Sichern Ihrer AWL-Quelldatei sinnvoll. Mit **UEBER**Nahme speichern Sie Ihre Datei, übersetzen sie zugleich in die Zwischendatei und verlassen den Editor.

Wie Sie diese Editierfunktionen praktisch einsetzen, zeigen wir Ihnen im Kapitel 3.3.5.

Beschreibung

Die **MERKE**- und **KOPIERE**-Funktion ermöglichen, beliebige Zeichenfolgen und Textblöcke in einen Puffer oder auf eine sequentielle Datei zu schreiben ("auslagern") und sie an eine beliebige Stelle zu kopieren. So können Sie z.B. Netzwerke verschieben oder immer wieder neu ins Programm einbinden.

Zusätzlich können Sie mit der Kopierfunktion auch andere AWL-Quelldateien oder einzelne Bausteine daraus einlesen. Merk-, Kopier- und Löschfunktion können mit einem Wiederholungsfaktor kombiniert werden.

Die Funktionen **SUCHE** und **ERSETZE** erleichtern die Korrekturen Ihres Programms:

Sie können sich schnell und gezielt in Ihrer Datei bewegen. Sie können einzelne Zeichenfolgen, wie z.B. Symbole oder Operanden, mit einer Funktion in Ihrer gesamten Anweisungsliste ändern. Außerdem werden Ihnen Einfüge- und Überschreib**modus** wahlweise angeboten.

Mit **SICHERN** haben Sie, wie bereits erwähnt, die Möglichkeit, Ihre Datei zu speichern, ohne den Editor zu verlassen. Bei jeder kurzzeitigen Unterbrechung einer Editiersitzung ist ein solches Sichern Ihrer AWL-Quelldatei sinnvoll. Mit **UEBER**Nahme speichern Sie Ihre Datei, übersetzen sie zugleich in die Zwischendatei und verlassen den Editor.

Wie Sie diese Editierfunktionen praktisch einsetzen, zeigen wir Ihnen im Kapitel 3.3.5.

2 - 10

C79000-B8500-C677-02

2.2.3 Steuerzeichen

Um die Übersetzung der AWL-Quelldatei in eine STEP 5-Programmdatei zu ermöglichen, müssen beim Editieren bestimmte Steuerzeichen und Schreibkonventionen beachtet werden. Eine vollständige Zusammenstellung aller Editiervorschriften finden Sie in den Kapiteln 3.3.2 und 3.3.3. Hier seien nur einige wenige Steuerzeichen beschrieben, die gegenüber dem Paket KOP, FUP, AWL Neuerungen bieten.

#TY markiert den AG-Typ. Nach diesem Steuerzeichen können Sie hier das AG angeben, in dem das Programm ablaufen soll. Diese Angabe muß mit dem Eintrag im Sprachraumfeld der Voreinstellung übereinstimmen. Der Batch-Compiler *prüft* beim *Übersetzen* in der Programmdatei, ob die editierten Operationen im Operationsvorrat des eingetragenen AGs zulässig sind. Der AG-Typ darf in der AWL-Quelldatei am Dateianfang und an Bausteingrenzen stehen.

Beschreibung

2.2.3 Steuerzeichen

Um die Übersetzung der AWL-Quelldatei in eine STEP 5-Programmdatei zu ermöglichen, müssen beim Editieren bestimmte Steuerzeichen und Schreibkonventionen beachtet werden. Eine vollständige Zusammenstellung aller Editiervorschriften finden Sie in den Kapiteln 3.3.2 und 3.3.3. Hier seien nur einige wenige Steuerzeichen beschrieben, die gegenüber dem Paket KOP, FUP, AWL Neuerungen bieten.

#TY markiert den AG-Typ. Nach diesem Steuerzeichen können Sie hier das AG angeben, in dem das Programm ablaufen soll. Diese Angabe muß mit dem Eintrag im Sprachraumfeld der Voreinstellung übereinstimmen. Der Batch-Compiler *prüft* beim *Übersetzen* in der Programmdatei, ob die editierten Operationen im Operationsvorrat des eingetragenen AGs zulässig sind. Der AG-Typ darf in der AWL-Quelldatei am Dateianfang und an Bausteingrenzen stehen.

2

C79000-B8500-C677-02

2 - 11

Nachfolgend aufgelistete Bezeichnungen für den AG-Sprachraum sind zulässig:

AG-Typ	Prozessor	Sprachraumbezeichnung
AG 100 U	CPU 100 CPU 102 CPU 103	CPU100 CPU102 CPU103
AG 101 U		AG101U
AG 110 S		AG110S
AG 115 U	CPU 941 CPU 942, 943, 944	CPU941 CPU942
AG 130 WB		AG130W
AG 135 U	CPU 921 CPU 922 CPU 928 CPU 928B	CPU921 CPU922 CPU928 CPU928B
AG 135 W		AG135W
AG 135 WB		AG135B
AG 150 A/K		AG150A
AG 150 S/U		AG150S
AG 155 U	CPU 946/947	AG155U
E/A-Prozessor	IP 257	IP257

Die *Übersetzung* in die Programmdatei wird nur dann ausgeführt, wenn die Bezeichnung des AGs in der Voreinstellung (Feld "SPRACHRAUM") mit den Angaben in den #TY-Zeilen der AWL-Quelldatei übereinstimmt. Bei Ungleichheit wird die *Übersetzung* in der #TY-Zeile abgebrochen. Tragen Sie in der Voreinstellung für den Sprachraum "NEIN" ein, wird die *Übersetzung* ohne *AG-spezifischen Prüflauf* durchgeführt.

Beschreibung

Nachfolgend aufgelistete Bezeichnungen für den AG-Sprachraum sind zulässig:

AG-Typ	Prozessor	Sprachraumbezeichnung
AG 100 U	CPU 100 CPU 102 CPU 103	CPU100 CPU102 CPU103
AG 101 U		AG101U
AG 110 S		AG110S
AG 115 U	CPU 941 CPU 942, 943, 944	CPU941 CPU942
AG 130 WB		AG130W
AG 135 U	CPU 921 CPU 922 CPU 928 CPU 928B	CPU921 CPU922 CPU928 CPU928B
AG 135 W		AG135W
AG 135 WB		AG135B
AG 150 A/K		AG150A
AG 150 S/U		AG150S
AG 155 U	CPU 946/947	AG155U
E/A-Prozessor	IP 257	IP257

Die *Übersetzung* in die Programmdatei wird nur dann ausgeführt, wenn die Bezeichnung des AGs in der Voreinstellung (Feld "SPRACHRAUM") mit den Angaben in den #TY-Zeilen der AWL-Quelldatei übereinstimmt. Bei Ungleichheit wird die *Übersetzung* in der #TY-Zeile abgebrochen. Tragen Sie in der Voreinstellung für den Sprachraum "NEIN" ein, wird die *Übersetzung* ohne *AG-spezifischen Prüflauf* durchgeführt.

2 - 12

C79000-B8500-C677-02

Include-Befehl

Der Include-Befehl #I ermöglicht das Einbinden einer beliebigen Datei und darf in der AWL-Quelldatei am Dateianfang und an Bausteingrenzen, d.h. nach BE, stehen. Nach #I folgt die Dateibezeichnung. Wichtig dabei ist, daß das Laufwerk mit angegeben wird (Beispiel: #I B:TEST).

Der Dateieinschluß erfolgt auf Zwischencodeebene, d. h. die einzufügende Datei muß als Zwischendatei vorhanden sein. Fehlt die Laufwerksangabe, wird auf das Laufwerk zugegriffen, das in der Voreinstellung für die Zwischendatei festgelegt ist.

Kommen in beiden Dateien Bausteine mit identischer Bezeichnung vor, müssen Sie diese vor der *Übersetzung* umbenennen. Diesem Problem können Sie aber ausweichen, wenn Sie in Ihren AWL-Quelldateien die Bausteine symbolisch benennen.

Aber hierfür ist unbedingt nötig, daß die passende Symbolikdatei vorhanden ist, denn zum Übersetzen ist der Bausteintyp und die Nummer unabdingbar.

Der Include-Befehl ist besonders für anwenderspezifische Bibliotheken geeignet: Standardisierte Programme können mit Include-Dateien aufgabenspezifisch verändert werden. Bei einer Änderung brauchen z.B. nur die Include-Dateien getauscht werden. Nach der Änderung wird bei allen Programmen der neue Stand zur Erzeugung der Programmdatei benutzt.

2.2.4 Übersetzen

Erzeugen einer Programmdatei

Mit dem Batch-Compiler können Sie nun alle Bausteine, eine Bausteingruppe, oder einen einzelnen Baustein aus der Zwischendatei oder der AWL-Quelldatei in die Programmdatei übersetzen. Wenn Sie jede Änderung Ihrer Anweisungsliste in der Quelldatei mit *Übernahme* abgespeichert haben, genügt die Übersetzung der Zwischendatei. Andernfalls müssen Sie die Übersetzung der AWL-Quelldatei anstoßen, die automatisch eine aktuelle Zwischendatei erzeugt.

Haben Sie Ihre AWL-Quelldatei symbolisch programmiert, wird beim *Übersetzen* in die Programmdatei die voreingestellte Symbolikdatei mit der Zwischendatei verbunden. Eine Symbolikdatei wird nicht vom AWL-Editor angelegt, sondern muß mit dem Symbolik-

Beschreibung

Include-Befehl

Der Include-Befehl #I ermöglicht das Einbinden einer beliebigen Datei und darf in der AWL-Quelldatei am Dateianfang und an Bausteingrenzen, d.h. nach BE, stehen. Nach #I folgt die Dateibezeichnung. Wichtig dabei ist, daß das Laufwerk mit angegeben wird (Beispiel: #I B:TEST).

Der Dateieinschluß erfolgt auf Zwischencodeebene, d. h. die einzufügende Datei muß als Zwischendatei vorhanden sein. Fehlt die Laufwerksangabe, wird auf das Laufwerk zugegriffen, das in der Voreinstellung für die Zwischendatei festgelegt ist.

Kommen in beiden Dateien Bausteine mit identischer Bezeichnung vor, müssen Sie diese vor der *Übersetzung* umbenennen. Diesem Problem können Sie aber ausweichen, wenn Sie in Ihren AWL-Quelldateien die Bausteine symbolisch benennen.

Aber hierfür ist unbedingt nötig, daß die passende Symbolikdatei vorhanden ist, denn zum Übersetzen ist der Bausteintyp und die Nummer unabdingbar.

Der Include-Befehl ist besonders für anwenderspezifische Bibliotheken geeignet: Standardisierte Programme können mit Include-Dateien aufgabenspezifisch verändert werden. Bei einer Änderung brauchen z.B. nur die Include-Dateien getauscht werden. Nach der Änderung wird bei allen Programmen der neue Stand zur Erzeugung der Programmdatei benutzt.

2

2.2.4 Übersetzen

Erzeugen einer Programmdatei

Mit dem Batch-Compiler können Sie nun alle Bausteine, eine Bausteingruppe, oder einen einzelnen Baustein aus der Zwischendatei oder der AWL-Quelldatei in die Programmdatei übersetzen. Wenn Sie jede Änderung Ihrer Anweisungsliste in der Quelldatei mit *Übernahme* abgespeichert haben, genügt die Übersetzung der Zwischendatei. Andernfalls müssen Sie die Übersetzung der AWL-Quelldatei anstoßen, die automatisch eine aktuelle Zwischendatei erzeugt.

Haben Sie Ihre AWL-Quelldatei symbolisch programmiert, wird beim *Übersetzen* in die Programmdatei die voreingestellte Symbolikdatei mit der Zwischendatei verbunden. Eine Symbolikdatei wird nicht vom AWL-Editor angelegt, sondern muß mit dem Symbolik-

C79000-B8500-C677-02 2 - 13

Editor erstellt werden. Binden Sie mit dem Include-Befehl #I eine andere Datei ein, so müssen Sie darauf achten, daß die Symbole für diese Datei in der voreingestellten Symbolikdatei enthalten sind.

In den Kommandozeilen des **Compilers** können Sie angeben, ob Maschinencode generiert oder nur ein Test auf Fehlerfreiheit durchgeführt werden soll, und, ob Sie beim Überschreiben von Bausteinen zur Kontrolle gefragt werden wollen. Ebenso können Sie gleichzeitig eine Ausgabe des übersetzten Programms auf den Drucker vereinbaren.

Rückübersetzen aus einer Programmdatei

Für Bausteine, die mit dem Paket KOP, FUP, AWL erstellt worden sind, bestehen weder AWL-Quelldateien noch Zwischendateien. Der AWL-Editor/Batch-Compiler kann diese Dateien aus einer Programmdatei erstellen.

Beim Rückübersetzen eines Bausteines, einer Bausteingruppe, oder aller Bausteine aus der Programmdatei können Sie zunächst die Zwischendatei erzeugen oder direkt die sequentielle AWL-Quelldatei, die Sie ändern und ergänzen können.

Bei der *Rückübersetzung* wählen Sie, wie Ihre "neue" AWL-Quelldatei aussehen soll: Die Anweisungen werden entweder nur mit Symbolen wiedergegeben, oder nur mit Absolutparametern, oder mit beiden. Außerdem wird das Steuerzeichen für die Sprachraumkennung in der Zwischendatei eingetragen, wenn in der Voreinstellung eine Sprachraumkennung (AG-Typ) eingetragen ist.

Der AWL-Editor kann Dateien mit max. 65535 Zeilen verarbeiten. Die Zeilenzahl der AWL-Quelldatei ist aber nicht nur von der Anzahl der STEP 5-Anweisungen, sondern auch von Sonderanweisungen, Kommentarzeilen usw. abhängig. Ist die Programmdatei, die Sie *rückübersetzen* wollen, größer, müssen die Bausteine auf mehrere Zwischendateien verteilt werden.

Standard-Funktionsbausteine, sowie GRAPH 5- und Assemblerbausteine werden nicht rückübersetzt.

Prüfungen beim Übersetzen

Der Zwischencode wird während der *Übersetzung/Rückübersetzung* auf Zulässigkeit der entstehenden Anweisung überprüft. Ebenso geprüft wird die Zulässigkeit einer Anweisung in Bezug auf den Bausteintyp. Der Sprachraum wird geprüft, falls Sie in der Vor-

Beschreibung

Editor erstellt werden. Binden Sie mit dem Include-Befehl #I eine andere Datei ein, so müssen Sie darauf achten, daß die Symbole für diese Datei in der voreingestellten Symbolikdatei enthalten sind.

In den Kommandozeilen des **Compilers** können Sie angeben, ob Maschinencode generiert oder nur ein Test auf Fehlerfreiheit durchgeführt werden soll, und, ob Sie beim Überschreiben von Bausteinen zur Kontrolle gefragt werden wollen. Ebenso können Sie gleichzeitig eine Ausgabe des übersetzten Programms auf den Drucker vereinbaren.

Rückübersetzen aus einer Programmdatei

Für Bausteine, die mit dem Paket KOP, FUP, AWL erstellt worden sind, bestehen weder AWL-Quelldateien noch Zwischendateien. Der AWL-Editor/Batch-Compiler kann diese Dateien aus einer Programmdatei erstellen.

Beim Rückübersetzen eines Bausteines, einer Bausteingruppe, oder aller Bausteine aus der Programmdatei können Sie zunächst die Zwischendatei erzeugen oder direkt die sequentielle AWL-Quelldatei, die Sie ändern und ergänzen können.

Bei der *Rückübersetzung* wählen Sie, wie Ihre "neue" AWL-Quelldatei aussehen soll: Die Anweisungen werden entweder nur mit Symbolen wiedergegeben, oder nur mit Absolutparametern, oder mit beiden. Außerdem wird das Steuerzeichen für die Sprachraumkennung in der Zwischendatei eingetragen, wenn in der Voreinstellung eine Sprachraumkennung (AG-Typ) eingetragen ist.

Der AWL-Editor kann Dateien mit max. 65535 Zeilen verarbeiten. Die Zeilenzahl der AWL-Quelldatei ist aber nicht nur von der Anzahl der STEP 5-Anweisungen, sondern auch von Sonderanweisungen, Kommentarzeilen usw. abhängig. Ist die Programmdatei, die Sie *rückübersetzen* wollen, größer, müssen die Bausteine auf mehrere Zwischendateien verteilt werden.

Standard-Funktionsbausteine, sowie GRAPH 5- und Assemblerbausteine werden nicht rückübersetzt.

Prüfungen beim Übersetzen

Der Zwischencode wird während der *Übersetzung/Rückübersetzung* auf Zulässigkeit der entstehenden Anweisung überprüft. Ebenso geprüft wird die Zulässigkeit einer Anweisung in Bezug auf den Bausteintyp. Der Sprachraum wird geprüft, falls Sie in der Vor-

2 - 14

C79000-B8500-C677-02

einstellung einen AG-Typ angegeben haben. Bei symbolischer Programmierung werden die Zuordnungen in Verbindung mit den Operanden geprüft.

Haben Sie in der AWL-Quelldatei sowohl einen absoluten als auch einen symbolischen Operanden angegeben, dann wird die Übereinstimmung mit der Symbolikdatei überprüft. Stimmen die Parameter nicht überein, wird der dem Symbol zugeordnete Absolutparameter aus der Symbolikdatei verwendet und in der Fehlerliste eine Warnmeldung abgelegt. Bei absoluter Programmierung erfolgt kein Zugriff auf die Symbolikdatei. Fehler, die bei diesen Prüfungen festgestellt werden, werden in der Fehlerliste ausgewiesen.

2.2.5 Drucken

Ein Listing der AWL-Quelldatei können Sie über die Funktion **Drucken** (in der Funktionsanwahl) erzeugen. Diese Funktion gibt aber nur die voreingestellte AWL-Quelldatei auf Drucker aus.

In den Kommandozeilen der Funktion **Compiler** wird Ihnen beim *Übersetzen* eine Ausgabe auf Drucker angeboten. Damit können Sie das Ergebnis jedes Übersetzungslaufs, auch des Testlaufs, festhalten.

Für das Layout Ihrer Druckerausgabe bietet Ihnen der AWL-Editor/Batch-Compiler die im Basispaket STEP 5 üblichen Druckformate. Sie wählen zwischen Standarddruckausgabe, Normalschrift, Schmalschrift und Superschmalschrift. Zu Ausgaben im A3-Format muß der Schriftfuß 132 Zeichen breit sein (Datei F2.INI), in A4-Format 80 Zeichen (Datei F1.INI). Bei Ausgabe in Schmalschrift erscheint zusätzlich der Operandenkommentar. Bei Ausgabe in Superschmalschrift erscheint zusätzlich der Symbolkommentar.

Beschreibung

einstellung einen AG-Typ angegeben haben. Bei symbolischer Programmierung werden die Zuordnungen in Verbindung mit den Operanden geprüft.

Haben Sie in der AWL-Quelldatei sowohl einen absoluten als auch einen symbolischen Operanden angegeben, dann wird die Übereinstimmung mit der Symbolikdatei überprüft. Stimmen die Parameter nicht überein, wird der dem Symbol zugeordnete Absolutparameter aus der Symbolikdatei verwendet und in der Fehlerliste eine Warnmeldung abgelegt. Bei absoluter Programmierung erfolgt kein Zugriff auf die Symbolikdatei. Fehler, die bei diesen Prüfungen festgestellt werden, werden in der Fehlerliste ausgewiesen.

2

2.2.5 Drucken

Ein Listing der AWL-Quelldatei können Sie über die Funktion **Drucken** (in der Funktionsanwahl) erzeugen. Diese Funktion gibt aber nur die voreingestellte AWL-Quelldatei auf Drucker aus.

In den Kommandozeilen der Funktion **Compiler** wird Ihnen beim *Übersetzen* eine Ausgabe auf Drucker angeboten. Damit können Sie das Ergebnis jedes Übersetzungslaufs, auch des Testlaufs, festhalten.

Für das Layout Ihrer Druckerausgabe bietet Ihnen der AWL-Editor/Batch-Compiler die im Basispaket STEP 5 üblichen Druckformate. Sie wählen zwischen Standarddruckausgabe, Normalschrift, Schmalschrift und Superschmalschrift. Zu Ausgaben im A3-Format muß der Schriftfuß 132 Zeichen breit sein (Datei F2.INI), in A4-Format 80 Zeichen (Datei F1.INI). Bei Ausgabe in Schmalschrift erscheint zusätzlich der Operandenkommentar. Bei Ausgabe in Superschmalschrift erscheint zusätzlich der Symbolkommentar.

C79000-B8500-C677-02

2 - 15

2.3 Die Zwischendatei A1.SEQ

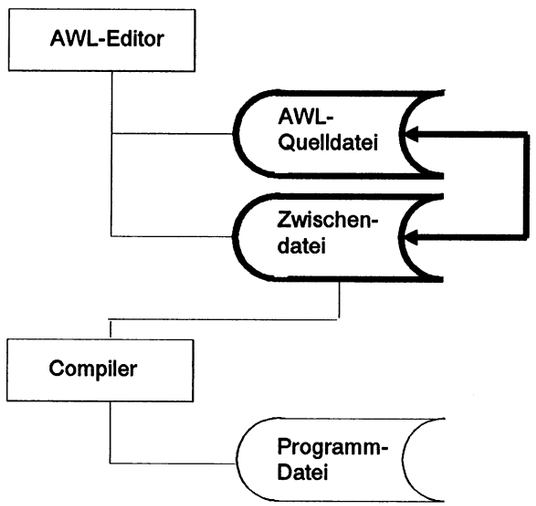
Die Zwischendatei bildet die zentrale Datei im Paket AWL-Editor/Batch-Compiler. Sprachunabhängig und noch nicht Maschinencode MC5, ist sie die Basis für alle Übersetzungsläufe. Aus ihr können jederzeit

- STEP 5-Programmdateien,
- AWL-Quelldateien,
- anlagenspezifische Versionen eines Programms,
- fremdsprachige Versionen eines Programms

erstellt werden.

Deshalb ist es unbedingt nötig, immer die Zwischendatei zu sichern, und es ist empfehlenswert, eine AWL-Quelldatei immer mit *Übernehmen* zu verlassen, damit die Zwischendatei aktuell ist.

2.3.1 Zusammenhänge zwischen AWL-Quelldatei und Zwischendatei



Beschreibung

2.3 Die Zwischendatei A1.SEQ

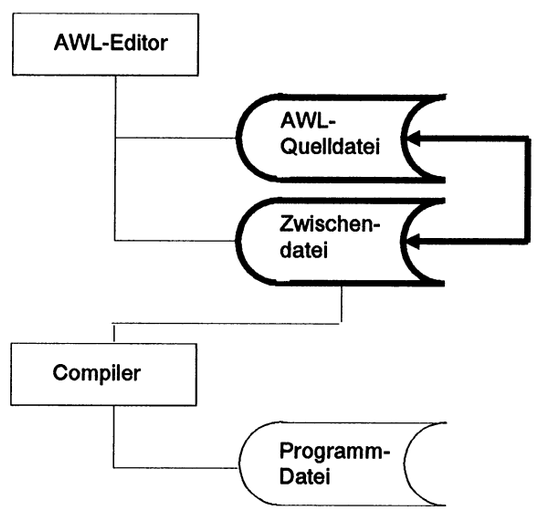
Die Zwischendatei bildet die zentrale Datei im Paket AWL-Editor/Batch-Compiler. Sprachunabhängig und noch nicht Maschinencode MC5, ist sie die Basis für alle Übersetzungsläufe. Aus ihr können jederzeit

- STEP 5-Programmdateien,
- AWL-Quelldateien,
- anlagenspezifische Versionen eines Programms,
- fremdsprachige Versionen eines Programms

erstellt werden.

Deshalb ist es unbedingt nötig, immer die Zwischendatei zu sichern, und es ist empfehlenswert, eine AWL-Quelldatei immer mit *Übernehmen* zu verlassen, damit die Zwischendatei aktuell ist.

2.3.1 Zusammenhänge zwischen AWL-Quelldatei und Zwischendatei



AWL-Quelldatei und Zwischendatei sind eng miteinander verbunden: Sie haben den gleichen Namen und ihre Dateikennung unterscheidet sich nur in einem Zeichen (A0.SEQ, A1.SEQ). Der Name einer Zwischendatei kann nie unabhängig von der AWL-Quelldatei geändert werden. Beide Dateien können jedoch auf unterschiedlichen Laufwerken liegen. Über die Namensgleichheit ist gewährleistet, daß Ihre editierten Programme bei *Übernahme* in die zugehörige Zwischendatei übersetzt werden.

Rückübersetzung

Auch bei einer *Rückübersetzung* wird in die namentlich zugehörige AWL-Quelldatei gespeichert. Dies ist zu beachten, wenn AWL-Quelldatei und Zwischendatei nicht auf gleichem Stand sind, oder die "alte" Anweisungsliste nicht überschrieben werden soll, z.B. wenn Ihre erste AWL-Quelldatei Zusatzkommentare enthält. Diese werden nicht in die Programmdatei mitübertragen und sind nach dem Rückübersetzen verloren.

Folgende Bedingungen ergeben sich deshalb für das Generieren einer AWL-Quelldatei aus einer Zwischendatei:

- Ist keine AWL-Quelldatei vorhanden, wird sie automatisch erstellt, wenn Sie *Editieren* aufrufen. Sie erhält den Namen, der vorab in der Voreinstellung festgelegt worden ist.
- Ist eine AWL-Quelldatei gleichen Namens vorhanden, muß die Zwischendatei explizit in die AWL-Quelldatei überführt werden. Dies geschieht mit der Sonderfunktion **ZWI>SEQ** (s.u.). Dabei wird die alte AWL-Quelldatei überschrieben.

Soll eine "alte" AWL-Quelldatei erhalten bleiben, tragen Sie vor der *Rückübersetzung* einer Programmdatei in der Voreinstellung einen Namen für die neue AWL-Quelldatei ein. Unter diesem Namen wird dann in die Zwischendatei rückübersetzt und in die Quelldatei abgespeichert.

Deshalb ist es wichtig, vor jedem *Übersetzungslauf*, insbesondere aber bei einer *Rückübersetzung*, zu prüfen, ob in der Voreinstellung die richtigen Dateien eingetragen sind.

Hinweis
Die Funktionen SEQ>MC5 und MC5>SEQ erzeugen automatisch eine aktuelle Zwischendatei (siehe Abschnitt 2.3.2)

Beschreibung

AWL-Quelldatei und Zwischendatei sind eng miteinander verbunden: Sie haben den gleichen Namen und ihre Dateikennung unterscheidet sich nur in einem Zeichen (A0.SEQ, A1.SEQ). Der Name einer Zwischendatei kann nie unabhängig von der AWL-Quelldatei geändert werden. Beide Dateien können jedoch auf unterschiedlichen Laufwerken liegen. Über die Namensgleichheit ist gewährleistet, daß Ihre editierten Programme bei *Übernahme* in die zugehörige Zwischendatei übersetzt werden.

Rückübersetzung

Auch bei einer *Rückübersetzung* wird in die namentlich zugehörige AWL-Quelldatei gespeichert. Dies ist zu beachten, wenn AWL-Quelldatei und Zwischendatei nicht auf gleichem Stand sind, oder die "alte" Anweisungsliste nicht überschrieben werden soll, z.B. wenn Ihre erste AWL-Quelldatei Zusatzkommentare enthält. Diese werden nicht in die Programmdatei mitübertragen und sind nach dem Rückübersetzen verloren.

Folgende Bedingungen ergeben sich deshalb für das Generieren einer AWL-Quelldatei aus einer Zwischendatei:

- Ist keine AWL-Quelldatei vorhanden, wird sie automatisch erstellt, wenn Sie *Editieren* aufrufen. Sie erhält den Namen, der vorab in der Voreinstellung festgelegt worden ist.
- Ist eine AWL-Quelldatei gleichen Namens vorhanden, muß die Zwischendatei explizit in die AWL-Quelldatei überführt werden. Dies geschieht mit der Sonderfunktion **ZWI>SEQ** (s.u.). Dabei wird die alte AWL-Quelldatei überschrieben.

Soll eine "alte" AWL-Quelldatei erhalten bleiben, tragen Sie vor der *Rückübersetzung* einer Programmdatei in der Voreinstellung einen Namen für die neue AWL-Quelldatei ein. Unter diesem Namen wird dann in die Zwischendatei rückübersetzt und in die Quelldatei abgespeichert.

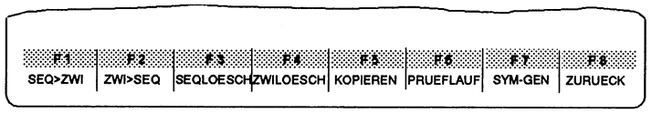
Deshalb ist es wichtig, vor jedem *Übersetzungslauf*, insbesondere aber bei einer *Rückübersetzung*, zu prüfen, ob in der Voreinstellung die richtigen Dateien eingetragen sind.

Hinweis
Die Funktionen SEQ>MC5 und MC5>SEQ erzeugen automatisch eine aktuelle Zwischendatei (siehe Abschnitt 2.3.2)

C79000-B8500-C677-02

2 - 17

2.3.2
Sonderfunktionen



Die Sonderfunktionen bieten verschiedene Umwandlungen, um AWL-Quelldateien und Zwischendateien zu erzeugen. Dies kann nötig sein, weil AWL-Quelldatei und Zwischendatei immer den gleichen Namen haben, aber manchmal verschiedene Ausgabestände (s.o., 2.3.1).

Wie Sie diese Funktionen praktisch benutzen, zeigen wir Ihnen im Kapitel 3.7.

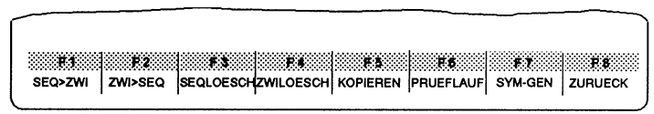
SEQ>ZWI wandelt eine sequentielle Datei in eine Zwischendatei. Diese Funktion setzen Sie dann ein, wenn z.B. Ihre AWL-Quelldatei mit einem fremden Texteditor geschrieben wurde und zu einer STEP 5-Programmdatei übersetzt werden soll, oder wenn die Zwischendatei nicht mehr vorhanden ist.

ZWI>SEQ setzt eine Zwischendatei in eine sequentielle Datei um, z.B. haben Sie eine Programmdatei rückübersetzt und wollen diese mit dem AWL-Editor bearbeiten. Dazu muß die Zwischendatei in eine sequentielle Datei umgesetzt werden. Diese Funktion ist besonders wichtig, wenn ein alter Ausgabestand der Quelldatei existiert. Wollen Sie die Quelldatei mit einem fremdsprachigen AWL-Editor ausgeben (siehe unter Abschnitt 2.3.4), ist diese Funktion ebenfalls sehr hilfreich.

SEQLOESCH damit können Sie sequentielle Dateien löschen, wenn Sie diese z.B. bei Übersetzungsläufen neu generieren wollen. *Editieren* Sie dann die übersetzten Dateien, werden die sequentiellen Dateien automatisch vom Editor erstellt.

Beschreibung

2.3.2
Sonderfunktionen



Die Sonderfunktionen bieten verschiedene Umwandlungen, um AWL-Quelldateien und Zwischendateien zu erzeugen. Dies kann nötig sein, weil AWL-Quelldatei und Zwischendatei immer den gleichen Namen haben, aber manchmal verschiedene Ausgabestände (s.o., 2.3.1).

Wie Sie diese Funktionen praktisch benutzen, zeigen wir Ihnen im Kapitel 3.7.

SEQ>ZWI wandelt eine sequentielle Datei in eine Zwischendatei. Diese Funktion setzen Sie dann ein, wenn z.B. Ihre AWL-Quelldatei mit einem fremden Texteditor geschrieben wurde und zu einer STEP 5-Programmdatei übersetzt werden soll, oder wenn die Zwischendatei nicht mehr vorhanden ist.

ZWI>SEQ setzt eine Zwischendatei in eine sequentielle Datei um, z.B. haben Sie eine Programmdatei rückübersetzt und wollen diese mit dem AWL-Editor bearbeiten. Dazu muß die Zwischendatei in eine sequentielle Datei umgesetzt werden. Diese Funktion ist besonders wichtig, wenn ein alter Ausgabestand der Quelldatei existiert. Wollen Sie die Quelldatei mit einem fremdsprachigen AWL-Editor ausgeben (siehe unter Abschnitt 2.3.4), ist diese Funktion ebenfalls sehr hilfreich.

SEQLOESCH damit können Sie sequentielle Dateien löschen, wenn Sie diese z.B. bei Übersetzungsläufen neu generieren wollen. *Editieren* Sie dann die übersetzten Dateien, werden die sequentiellen Dateien automatisch vom Editor erstellt.

ZWILOESCH löscht Zwischendateien (z.B. alte Ausgabestände). Sie werden entweder mit der Funktion **SEQ>ZWI** aus einer aktualisierten Quelldatei wieder hergestellt, oder beim *Übernehmen* einer editierten Quelle.

KOPIERE -Funktion benutzen Sie, wenn Sie Zwischendatei und AWL-Quelldatei zur Sicherung auf andere Laufwerke kopieren wollen. Die Dateien können aber nur im Dienstprogramm D-HILFS umbenannt werden.

PRUEFLAUF bietet die Möglichkeit, die übersetzten Bausteine einer Programmdatei, aber auch Bausteine aus dem Paket KOP, FUP, AWL nachträglich auf die Zulässigkeit der Befehle für den eingestellten AG-Typ zu prüfen.

SYM-GEN erzeugt aus der AWL-Quelldatei eine Symbolik-Quelldatei, die alle verwendeten Symbole und Absolutparameter enthält. Diese Symbolik-Quelle können Sie mit dem Symbolikeditor um Zuordnungen und Kommentare erweitern, ohne die bereits in der AWL-Quelle enthaltenen Symbole und Absolutparameter erneut eingeben zu müssen.

2.3.3 Standardprogramme

Die Möglichkeit, eine Anweisungsliste ausschließlich mit Symbolen zu erstellen, und die Tatsache, daß die Zwischendatei sprachunabhängig ist, erlaubt Ihnen, Standardprogramme anzulegen.

Ausgetestete Bausteine und Module können in Bibliotheken abgelegt und dann mit dem Include-Befehl anlagenspezifisch zu individuellen Programmen zusammengebunden werden. So brauchen Sie Ihre neu kombinierten Programme nur noch mit einer speziellen Zuordnungsliste zu verbinden und erhalten anlagenspezifische STEP 5-Programme für individuelle Steuerungsprobleme.

Beschreibung

ZWILOESCH löscht Zwischendateien (z.B. alte Ausgabestände). Sie werden entweder mit der Funktion **SEQ>ZWI** aus einer aktualisierten Quelldatei wieder hergestellt, oder beim *Übernehmen* einer editierten Quelle.

KOPIERE -Funktion benutzen Sie, wenn Sie Zwischendatei und AWL-Quelldatei zur Sicherung auf andere Laufwerke kopieren wollen. Die Dateien können aber nur im Dienstprogramm D-HILFS umbenannt werden.

PRUEFLAUF bietet die Möglichkeit, die übersetzten Bausteine einer Programmdatei, aber auch Bausteine aus dem Paket KOP, FUP, AWL nachträglich auf die Zulässigkeit der Befehle für den eingestellten AG-Typ zu prüfen.

SYM-GEN erzeugt aus der AWL-Quelldatei eine Symbolik-Quelldatei, die alle verwendeten Symbole und Absolutparameter enthält. Diese Symbolik-Quelle können Sie mit dem Symbolikeditor um Zuordnungen und Kommentare erweitern, ohne die bereits in der AWL-Quelle enthaltenen Symbole und Absolutparameter erneut eingeben zu müssen.

2

2.3.3 Standardprogramme

Die Möglichkeit, eine Anweisungsliste ausschließlich mit Symbolen zu erstellen, und die Tatsache, daß die Zwischendatei sprachunabhängig ist, erlaubt Ihnen, Standardprogramme anzulegen.

Ausgetestete Bausteine und Module können in Bibliotheken abgelegt und dann mit dem Include-Befehl anlagenspezifisch zu individuellen Programmen zusammengebunden werden. So brauchen Sie Ihre neu kombinierten Programme nur noch mit einer speziellen Zuordnungsliste zu verbinden und erhalten anlagenspezifische STEP 5-Programme für individuelle Steuerungsprobleme.

C79000-B8500-C677-02 2 - 19

2.3.4 Fremdsprachige Versionen eines STEP 5-Programms

Mit dem Batch-Compiler können Sie auch fremdsprachige Versionen eines STEP 5-Programms erzeugen, wenn Sie Ihr Programm absolut programmiert haben oder wenn Sie über eine rückübersetzte Zwischendatei mit Absolutparametern verfügen: Mit der englischen und französischen Software-Version dieses Pakets erstellen Sie englische und französische AWL-Quellen.

Dazu gehen Sie ins Dienstprogramm **D-HILFS, D-UEBER** und kopieren Ihre deutsche AWL-Quelldatei und Zwischendatei mit einem neuen Namen. Diesen neuen Namen tragen Sie in der Voreinstellung der englischen/französischen Software-Version ein. Mit der Sonderfunktion **ZWI> SEQ** (s.o.) überführen Sie die sprachunabhängige Zwischendatei in die sequentielle Quelldatei. Diese wird dann beim **EDITIEREN** mit englischen/französischen STEP 5-Befehlen ausgegeben.

Eine andere Möglichkeit besteht darin, im englischen/französischen Paket die deutsche sequentielle Quelldatei zu löschen (Sonderfunktion **SEQLOESCH**). Eine neue wird automatisch generiert, wenn Sie **EDITIEREN** aufrufen.

Symbole und Kommentare werden nicht in den Fremdsprachen ausgegeben.

Beschreibung

2.3.4 Fremdsprachige Versionen eines STEP 5-Programms

Mit dem Batch-Compiler können Sie auch fremdsprachige Versionen eines STEP 5-Programms erzeugen, wenn Sie Ihr Programm absolut programmiert haben oder wenn Sie über eine rückübersetzte Zwischendatei mit Absolutparametern verfügen: Mit der englischen und französischen Software-Version dieses Pakets erstellen Sie englische und französische AWL-Quellen.

Dazu gehen Sie ins Dienstprogramm **D-HILFS, D-UEBER** und kopieren Ihre deutsche AWL-Quelldatei und Zwischendatei mit einem neuen Namen. Diesen neuen Namen tragen Sie in der Voreinstellung der englischen/französischen Software-Version ein. Mit der Sonderfunktion **ZWI> SEQ** (s.o.) überführen Sie die sprachunabhängige Zwischendatei in die sequentielle Quelldatei. Diese wird dann beim **EDITIEREN** mit englischen/französischen STEP 5-Befehlen ausgegeben.

Eine andere Möglichkeit besteht darin, im englischen/französischen Paket die deutsche sequentielle Quelldatei zu löschen (Sonderfunktion **SEQLOESCH**). Eine neue wird automatisch generiert, wenn Sie **EDITIEREN** aufrufen.

Symbole und Kommentare werden nicht in den Fremdsprachen ausgegeben.

2 - 20 C79000-B8500-C677-02

2.4 Ändern und Ergänzen von STEP 5-Bausteinen

2.4.1 Bausteine erstellt mit dem AWL-Editor

In der Voreinstellung geben Sie den Namen der AWL-Quelldatei an und eventuell noch das Laufwerk für die Zwischendatei. Nach Aufruf des AWL-Editors erscheint auf dem Bildschirm die gewünschte Anweisungsliste und das Editiermenü. Die Anweisungsliste können Sie nun mit Hilfe der Editierfunktionen ändern oder ergänzen. Speichern Sie Ihre geänderte Quelldatei immer mit *Übernahme* ab, damit die Zwischendatei aktualisiert wird und Ihre "neue" Programmdatei nicht mit einem "alten" Programm erstellt wird.

Wenn Sie den AWL-Editor mit der **Abbruchtaste** verlassen und den Abbruch bestätigen, werden die Änderungen oder Ergänzungen nicht in die AWL-Quelldatei übernommen.

2.4.2 Bausteine erstellt mit dem Paket KOP, FUP, AWL

Bausteine aus einer Programmdatei müssen rückübersetzt werden, damit sie im AWL-Editor bearbeitet werden können. Sind sie dann in einer AWL-Quelldatei vorhanden, werden sie wie oben, Abschnitt 2.4.1, beschrieben im AWL-Editor bearbeitet.

Beschreibung

2.4 Ändern und Ergänzen von STEP 5-Bausteinen

2.4.1 Bausteine erstellt mit dem AWL-Editor

In der Voreinstellung geben Sie den Namen der AWL-Quelldatei an und eventuell noch das Laufwerk für die Zwischendatei. Nach Aufruf des AWL-Editors erscheint auf dem Bildschirm die gewünschte Anweisungsliste und das Editiermenü. Die Anweisungsliste können Sie nun mit Hilfe der Editierfunktionen ändern oder ergänzen. Speichern Sie Ihre geänderte Quelldatei immer mit *Übernahme* ab, damit die Zwischendatei aktualisiert wird und Ihre "neue" Programmdatei nicht mit einem "alten" Programm erstellt wird.

Wenn Sie den AWL-Editor mit der **Abbruchtaste** verlassen und den Abbruch bestätigen, werden die Änderungen oder Ergänzungen nicht in die AWL-Quelldatei übernommen.

2.4.2 Bausteine erstellt mit dem Paket KOP, FUP, AWL

Bausteine aus einer Programmdatei müssen rückübersetzt werden, damit sie im AWL-Editor bearbeitet werden können. Sind sie dann in einer AWL-Quelldatei vorhanden, werden sie wie oben, Abschnitt 2.4.1, beschrieben im AWL-Editor bearbeitet.

C79000-B8500-C677-02

2 - 21

2.5 Prüflauf

2.5.1 Prüfen einer Programmdatei

Der Prüflauf ist der *Übersetzung* nachgeschaltet und prüft die Bausteine der Programmdatei. Er übernimmt z.B. die Prüfung der Parameterübergabe bei Funktionsbausteinen und das Vorhandensein der aufgerufenen Bausteine. Sie können einen Prüflauf über einen Baustein, eine Bausteingruppe oder alle Bausteine einer Programmdatei vereinbaren. Wenn in der Voreinstellung eine Sprachraumkennung eingetragen ist, wird zusätzlich die Zulässigkeit der Anweisungen zum AG-Typ geprüft. Nicht zulässige Anweisung werden in der Fehlerliste protokolliert.

2.5.2 Prüfen von Sonderbausteinen

Standard-Funktionsbausteine, GRAPH 5- und Assembler-Bausteine können mit dem AWL-Editor/Batch-Compiler zwar nicht erstellt und rückübersetzt, aber sie können mit dem nachgeschalteten *Prüflauf* geprüft werden! Für sie gilt, daß sowohl Existenz und Übergabe der Parameter als auch die Zulässigkeit der AWL-Anweisungen zum voreingestellten AG-Typ geprüft werden.

Beschreibung

2.5 Prüflauf

2.5.1 Prüfen einer Programmdatei

Der Prüflauf ist der *Übersetzung* nachgeschaltet und prüft die Bausteine der Programmdatei. Er übernimmt z.B. die Prüfung der Parameterübergabe bei Funktionsbausteinen und das Vorhandensein der aufgerufenen Bausteine. Sie können einen Prüflauf über einen Baustein, eine Bausteingruppe oder alle Bausteine einer Programmdatei vereinbaren. Wenn in der Voreinstellung eine Sprachraumkennung eingetragen ist, wird zusätzlich die Zulässigkeit der Anweisungen zum AG-Typ geprüft. Nicht zulässige Anweisung werden in der Fehlerliste protokolliert.

2.5.2 Prüfen von Sonderbausteinen

Standard-Funktionsbausteine, GRAPH 5- und Assembler-Bausteine können mit dem AWL-Editor/Batch-Compiler zwar nicht erstellt und rückübersetzt, aber sie können mit dem nachgeschalteten *Prüflauf* geprüft werden! Für sie gilt, daß sowohl Existenz und Übergabe der Parameter als auch die Zulässigkeit der AWL-Anweisungen zum voreingestellten AG-Typ geprüft werden.

2 - 22 C79000-B8500-C677-02

2.6 Fehlerliste

2.6.1 Fehlerdatei

Fehlermeldungen entstehen bei den folgenden Arbeitsschritten:

- Übersetzen der AWL-Quelldatei in die Zwischendatei.
- Übersetzen der Zwischendatei in die Programmdatei.
- Rückübersetzen der Programmdatei in die Zwischendatei.
- Rückübersetzen der Zwischendatei in die AWL-Quelldatei.
- Prüfen der Programmdatei (Prüflauf).

Die Fehlermeldungen legt das Programmiergerät in einer Fehlerliste in der Fehlerdatei <name>AF.SEQ ab.

In der Fehlerdatei ist immer nur die Fehlerliste des zuletzt durchgeführten Arbeitsschrittes enthalten. Die Fehlerdatei können Sie am Bildschirm oder am Drucker in dem von Ihnen gewünschten Druckausgabeformat ausgeben lassen. Die Fehlerdatei ist nicht vorhanden, wenn der letzte Arbeitsschritt fehlerfrei abgeschlossen wurde!

Beschreibung

2.6 Fehlerliste

2.6.1 Fehlerdatei

Fehlermeldungen entstehen bei den folgenden Arbeitsschritten:

- Übersetzen der AWL-Quelldatei in die Zwischendatei.
- Übersetzen der Zwischendatei in die Programmdatei.
- Rückübersetzen der Programmdatei in die Zwischendatei.
- Rückübersetzen der Zwischendatei in die AWL-Quelldatei.
- Prüfen der Programmdatei (Prüflauf).

2

Die Fehlermeldungen legt das Programmiergerät in einer Fehlerliste in der Fehlerdatei <name>AF.SEQ ab.

In der Fehlerdatei ist immer nur die Fehlerliste des zuletzt durchgeführten Arbeitsschrittes enthalten. Die Fehlerdatei können Sie am Bildschirm oder am Drucker in dem von Ihnen gewünschten Druckausgabeformat ausgeben lassen. Die Fehlerdatei ist nicht vorhanden, wenn der letzte Arbeitsschritt fehlerfrei abgeschlossen wurde!

C79000-B8500-C677-02

2 - 23

2.7 Eingabe von STEP 5-Anweisungen mit anderen Editoren

2.7.1 AWL-Quelldatei als Schnittstelle

Die AWL-Quelldatei können Sie auch mit anderen Editoren erstellen. Voraussetzung dafür ist allerdings, daß diese Editoren "echte" Tabulatoren (d.h., den Hexacode 09H) verarbeiten können. Andernfalls muß die erste Zeile der AWL-Quelldatei über das Steuerzeichen #TAB die Anfangsspalten der einzelnen Teilfelder festlegen (siehe Abschnitt 2.7.3).

Die ersten sechs Zeichen des Dateinamens sind frei wählbar, es müssen aber sechs Zeichen sein. Für die zwei letzten Namenszeichen und die Erweiterung muß A0.SEQ eingetragen werden. Eine Weiterbearbeitung dieser Datei mit den Werkzeugen des Paketes AWL-Editor/Batch-Compiler ist nur dann problemlos möglich, wenn Sie das unten beschriebene Format der sequentiellen Quelldatei einhalten. Der AWL-Editor/Batch-Compiler unterstützt Sie dann mit der Sonderfunktion **SEQ>ZWI**, und mit der anschließenden *Übersetzung* in die Programmdatei, bzw. mit der direkten Übersetzung mit der Funktion **SEQ>MC5**.

2.7.2 Format der sequentiellen Quelldatei des Editors

Je Anweisungszeile geben Sie einen Datensatz ein. Ein Datensatz beginnt mit dem Tabulatorzeichen (09H) und besteht aus 4 Datenfeldern, die ebenfalls durch Tabulatoren voneinander getrennt sind. Die Markierung des Datensatzendes durch "Carriage Return, CR" (=0DH) und "Line feed, LF" (=0AH) wird beim Zeilenabschluß mittels Returntaste vom Editor automatisch angefügt. Die maximale Zeichenanzahl beträgt für die Felder

TAB	TAB	TAB	TAB	CR, LF
Adresse	**Anweisung**	**Operandensymbol**	**Anweisungskommentar**	
4 Zeichen	13 Zeichen	24 Zeichen	32 Zeichen	

Der Datensatz einer Leerzeile besteht somit aus vier Tabulatorzeichen gefolgt von den "CR"- und "LF"-Zeichen.

Beschreibung

2.7 Eingabe von STEP 5-Anweisungen mit anderen Editoren

2.7.1 AWL-Quelldatei als Schnittstelle

Die AWL-Quelldatei können Sie auch mit anderen Editoren erstellen. Voraussetzung dafür ist allerdings, daß diese Editoren "echte" Tabulatoren (d.h., den Hexacode 09H) verarbeiten können. Andernfalls muß die erste Zeile der AWL-Quelldatei über das Steuerzeichen #TAB die Anfangsspalten der einzelnen Teilfelder festlegen (siehe Abschnitt 2.7.3).

Die ersten sechs Zeichen des Dateinamens sind frei wählbar, es müssen aber sechs Zeichen sein. Für die zwei letzten Namenszeichen und die Erweiterung muß A0.SEQ eingetragen werden. Eine Weiterbearbeitung dieser Datei mit den Werkzeugen des Paketes AWL-Editor/Batch-Compiler ist nur dann problemlos möglich, wenn Sie das unten beschriebene Format der sequentiellen Quelldatei einhalten. Der AWL-Editor/Batch-Compiler unterstützt Sie dann mit der Sonderfunktion SEQ>ZWI, und mit der anschließenden Übersetzung in die Programmdatei, bzw. mit der direkten Übersetzung mit der Funktion SEQ>MC5.

2.7.2 Format der sequentiellen Quelldatei des Editors

Je Anweisungszeile geben Sie einen Datensatz ein. Ein Datensatz beginnt mit dem Tabulatorzeichen (09H) und besteht aus 4 Datenfeldern, die ebenfalls durch Tabulatoren voneinander getrennt sind. Die Markierung des Datensatzendes durch "Carriage Return, CR" (=0DH) und "Line feed, LF" (=0AH) wird beim Zeilenabschluß mittels Returntaste vom Editor automatisch angefügt. Die maximale Zeichenanzahl beträgt für die Felder

TAB TAB TAB TAB CR, LF

Adresse 4 Zeichen | Anweisung 13 Zeichen | Operandensymbol 24 Zeichen | Anweisungskommentar 32 Zeichen

Der Datensatz einer Leerzeile besteht somit aus vier Tabulatorzeichen gefolgt von den "CR"- und "LF"-Zeichen.

2 - 24 C79000-B8500-C677-02

Der Datensatz für Kommentarzeilen fängt mit dem Tabulatorzeichen (=09H) an, unmittelbar gefolgt von den Steuerzeichen ✱ und ; für Netzwerk- und Zusatzkommentare. Danach maximal 79 Zeichen für den Kommentar und den Zeilenabschluß mit den "CR"- (=0DH) und "LF"-Zeichen (=0AH).

Die Datensätze dürfen Groß- und Kleinschreibung enthalten. Beim Einlesen wandelt der Editor in den Feldern Adresse und Anweisung automatisch alle Klein- in Großbuchstaben um. In den Feldern Operandensymbol und Anweisungskommentar bleibt die Groß-/Kleinschreibung erhalten. Umlaute dürfen nicht verwendet werden.

2.7.3 Steuerzeichen #TAB zur Verarbeitung von Fremddateien

Das Steuerzeichen #TAB ermöglicht die Übersetzung von Dateien ohne echte Tabulatorzeichen, wie sie viele Textprogramme wie z. B. 1st Wordplus erzeugen. Der AWL-Editor kann diese Dateien jedoch nicht bearbeiten, sondern gibt die Fehlermeldung "Dateiformat falsch" aus.

#TAB muß unmittelbar am Anfang der Quelldatei stehen. Davor sind lediglich Leerzeichen zugelassen. Danach müssen, durch Kommata getrennt, 4 Zahlen stehen, die die Anfangsspalten der Teilfelder angeben. Weitere Angaben in der ersten Zeile sind nicht zulässig!

Beispiel: Falls zwischen den Teilfeldern jeweils 1 Leerzeichen als Trennung stehen soll, lautet die erste Zeile der AWL-Quelldatei:

#TAB 1,6,21,46 RETURN (CR LF)

Die Spaltenangaben sind also immer vom Zeilenanfang aus gerechnet. Die Differenz der aufeinanderfolgenden Angaben muß mindestens so groß wie die entsprechende Teilfeldlänge (siehe Abschnitt 2.7.2) sein.

Beschreibung

Der Datensatz für Kommentarzeilen fängt mit dem Tabulatorzeichen (=09H) an, unmittelbar gefolgt von den Steuerzeichen ✱ und ; für Netzwerk- und Zusatzkommentare. Danach maximal 79 Zeichen für den Kommentar und den Zeilenabschluß mit den "CR"- (=0DH) und "LF"-Zeichen (=0AH).

Die Datensätze dürfen Groß- und Kleinschreibung enthalten. Beim Einlesen wandelt der Editor in den Feldern Adresse und Anweisung automatisch alle Klein- in Großbuchstaben um. In den Feldern Operandensymbol und Anweisungskommentar bleibt die Groß-/Kleinschreibung erhalten. Umlaute dürfen nicht verwendet werden.

2

2.7.3 Steuerzeichen #TAB zur Verarbeitung von Fremddateien

Das Steuerzeichen #TAB ermöglicht die Übersetzung von Dateien ohne echte Tabulatorzeichen, wie sie viele Textprogramme wie z. B. 1st Wordplus erzeugen. Der AWL-Editor kann diese Dateien jedoch nicht bearbeiten, sondern gibt die Fehlermeldung "Dateiformat falsch" aus.

#TAB muß unmittelbar am Anfang der Quelldatei stehen. Davor sind lediglich Leerzeichen zugelassen. Danach müssen, durch Kommata getrennt, 4 Zahlen stehen, die die Anfangsspalten der Teilfelder angeben. Weitere Angaben in der ersten Zeile sind nicht zulässig!

Beispiel: Falls zwischen den Teilfeldern jeweils 1 Leerzeichen als Trennung stehen soll, lautet die erste Zeile der AWL-Quelldatei:

#TAB 1,6,21,46 RETURN (CR LF)

Die Spaltenangaben sind also immer vom Zeilenanfang aus gerechnet. Die Differenz der aufeinanderfolgenden Angaben muß mindestens so groß wie die entsprechende Teilfeldlänge (siehe Abschnitt 2.7.2) sein.

C79000-B8500-C677-02 2 - 25

Beschreibung

2 - 26

C79000-B8500-C677-02

Bedienung am Programmiergerät 3

Bedienung am Programmiergerät 3

C79000-B8500-C677-02 3 - 1

3.1 Installieren der Software

Kopieren Sie die Dateien des AWL-Editors/Batch-Compilers mit dem PCP/M-Kommando PIP auf die Festplatte Ihres Programmiergeräts. Ziehen Sie dazu das PCP/M-Tabellenheft zu Rate, das Ihrem PG-Handbuch beiliegt.

Beispiel: PIP b:[G0] = a:*.*

In diesem Beispiel ist b: das Ziellaufwerk, [G0] der User und a: das Quellaufwerk.

Bedienung am Programmiergerät

3.1 Installieren der Software

Kopieren Sie die Dateien des AWL-Editors/Batch-Compilers mit dem PCP/M-Kommando PIP auf die Festplatte Ihres Programmiergeräts. Ziehen Sie dazu das PCP/M-Tabellenheft zu Rate, das Ihrem PG-Handbuch beiliegt.

Beispiel: PIP b:[G0] = a:*.*

In diesem Beispiel ist b: das Ziellaufwerk, [G0] der User und a: das Quellaufwerk.

3.2 Bedienschritte bis zur Funktionsanwahl

3.2.1 Allgemeines zur Bedienung

Die Bedienung des Programmiergeräts entspricht beim AWL-Editor/Batch-Compiler den Konventionen aller anderen S5-Pakete, d.h. Sie füllen eine Voreinstellung aus und finden in der Funktionsanwahl die Softkey-Leiste mit den möglichen Editier- und Bearbeitungsfunktionen. Die **Cursor-Tasten**, die **Help-, Abbruch-** und **Übernahmetaste** sind in ihren Funktionen identisch.

Im AWL-Editor arbeiten Sie genauso wie im Symbolik-Editor. Dagegen bestehen einige Unterschiede zu den **EINGABE/AUSGABE**-Funktionen im Paket KOP, FUP, AWL. So ist es z.B. nötig, im AWL-Editor einige Anweisungen mit Steuerzeichen zu versehen, die Symbole dagegen brauchen keine Bindestriche. Wir werden Sie an den jeweiligen Stellen besonders darauf aufmerksam machen.

Zu Masken, Softkeys und Eingabefeldern erhalten Sie über die **Help-Taste** Erklärungen und Informationen. Innerhalb dieser Help-Texte antworten Sie auf die Frage "Weiter?" mit der **Abbruchtaste** (= nein) oder mit der **Übernahmetaste** (= ja).

Ebenso gelten in diesem Paket alle Begriffsdefinitionen von STEP 5. Wollen Sie sich allgemein über STEP 5 informieren, lesen Sie bitte die Einführung in den Gerätehandbüchern Ihres Programmiergeräts.

3

3.2.2 Einstieg in das Paket AWL-Editor/Batch-Compiler

☞ Das folgende Beispiel ist auf einem PG 750 mit einem Diskettenlaufwerk erstellt worden. Mit ➤ sind alle Tätigkeiten gekennzeichnet, die Sie am Gerät durchführen. Die Zeichen, die Sie eingeben, sind *kursiv*, die zu benutzenden Funktionen bzw. Funktionstasten **fett** gedruckt; versal sind alle die Begriffe geschrieben, die am Bildschirm erscheinen und auf die bei der Bedienung Bezug genommen wird.

Bedienung am Programmiergerät

3.2 Bedienschritte bis zur Funktionsanwahl

3.2.1 Allgemeines zur Bedienung

Die Bedienung des Programmiergeräts entspricht beim AWL-Editor/Batch-Compiler den Konventionen aller anderen S5-Pakete, d.h. Sie füllen eine Voreinstellung aus und finden in der Funktionsanwahl die Softkey-Leiste mit den möglichen Editier- und Bearbeitungsfunktionen. Die **Cursor-Tasten**, die **Help-, Abbruch-** und **Übernahmetaste** sind in ihren Funktionen identisch.

Im AWL-Editor arbeiten Sie genauso wie im Symbolik-Editor. Dagegen bestehen einige Unterschiede zu den **EINGABE/AUSGABE**-Funktionen im Paket KOP, FUP, AWL. So ist es z.B. nötig, im AWL-Editor einige Anweisungen mit Steuerzeichen zu versehen, die Symbole dagegen brauchen keine Bindestriche. Wir werden Sie an den jeweiligen Stellen besonders darauf aufmerksam machen.

Zu Masken, Softkeys und Eingabefeldern erhalten Sie über die **Help-Taste** Erklärungen und Informationen. Innerhalb dieser Help-Texte antworten Sie auf die Frage "Weiter?" mit der **Abbruchtaste** (= nein) oder mit der **Übernahmetaste** (= ja).

Ebenso gelten in diesem Paket alle Begriffsdefinitionen von STEP 5. Wollen Sie sich allgemein über STEP 5 informieren, lesen Sie bitte die Einführung in den Gerätehandbüchern Ihres Programmiergeräts.

3

3.2.2 Einstieg in das Paket AWL-Editor/Batch-Compiler

☞ Das folgende Beispiel ist auf einem PG 750 mit einem Diskettenlaufwerk erstellt worden. Mit ➤ sind alle Tätigkeiten gekennzeichnet, die Sie am Gerät durchführen. Die Zeichen, die Sie eingeben, sind *kursiv*, die zu benutzenden Funktionen bzw. Funktionstasten **fett** gedruckt; versal sind alle die Begriffe geschrieben, die am Bildschirm erscheinen und auf die bei der Bedienung Bezug genommen wird.

C79000-B8500-C677-02 3 - 3

Voraussetzung:

Das Optionspaket AWL-Editor/Batch-Compiler ist auf der Festplatte Ihres Programmiergeräts geladen. Das Gerät ist betriebsbereit und das Betriebssystem PCP/M-86 ist aktiv. Der Bildschirm gibt die Bereitschaftsmeldung B> aus.

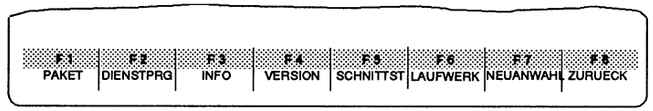
Starten des S5-Komi

Starten Sie den S5-Kommandointerpreter (S5-KOMI) aus PCP/M heraus mit dem Kommando

➤ *S5*

➤ **Return-Taste** drücken.

Auf dem Bildschirm werden dann in der PAKETANWAHL die STEP 5-Pakete aufgelistet.



Die Bedeutung der einzelnen Softkey-Tasten erläutert Ihnen die **Help-Taste**. Detaillierte Informationen zu diesen Funktionen finden Sie in der STEP 5-Beschreibung, dem Handbuch Band 2 Ihres Programmiergeräts, Kapitel S5-Kommandointerpreter.

Paket AWL-Editor/Batch-Compiler laden

➤ Cursor vor dem Paket AWL-Editor/Batch-Compiler positionieren,

➤ **PAKET** (F1) aufrufen

oder

➤ **Übernahmetaste** drücken.

Daraufhin erscheint die VOREINSTELLUNG.

Bedienung am Programmiergerät

Voraussetzung:

Das Optionspaket AWL-Editor/Batch-Compiler ist auf der Festplatte Ihres Programmiergeräts geladen. Das Gerät ist betriebsbereit und das Betriebssystem PCP/M-86 ist aktiv. Der Bildschirm gibt die Bereitschaftsmeldung B> aus.

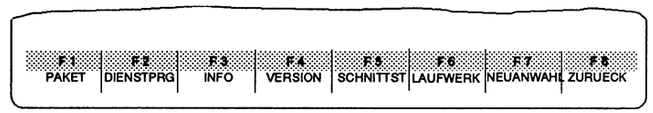
Starten des S5-Komi

Starten Sie den S5-Kommandointerpreter (S5-KOMI) aus PCP/M heraus mit dem Kommando

➤ *S5*

➤ **Return-Taste** drücken.

Auf dem Bildschirm werden dann in der PAKETANWAHL die STEP 5-Pakete aufgelistet.



Die Bedeutung der einzelnen Softkey-Tasten erläutert Ihnen die **Help-Taste**. Detaillierte Informationen zu diesen Funktionen finden Sie in der STEP 5-Beschreibung, dem Handbuch Band 2 Ihres Programmiergeräts, Kapitel S5-Kommandointerpreter.

Paket AWL-Editor/Batch-Compiler laden

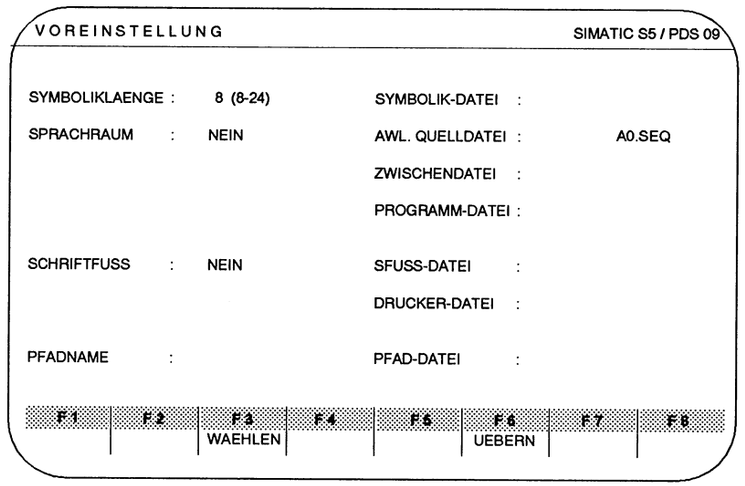
➤ Cursor vor dem Paket AWL-Editor/Batch-Compiler positionieren,

➤ **PAKET** (F1) aufrufen

oder

➤ **Übernahmetaste** drücken.

Daraufhin erscheint die VOREINSTELLUNG.



VOREINSTELLUNG

Der Cursor blinkt in der Zeile der AWL-QUELLDATEI (A0.SEQ). In dieser Datei wird Ihre Anweisungsliste gespeichert. Sie ist als sequentielle Datei, d.h., als ASCII-Datei, gekennzeichnet und ist die Quelle für den Übersetzungslauf.

Geben Sie an dieser Stelle den Namen Ihrer Datei ein. In unserem Beispiel soll die Datei "Test" heißen.

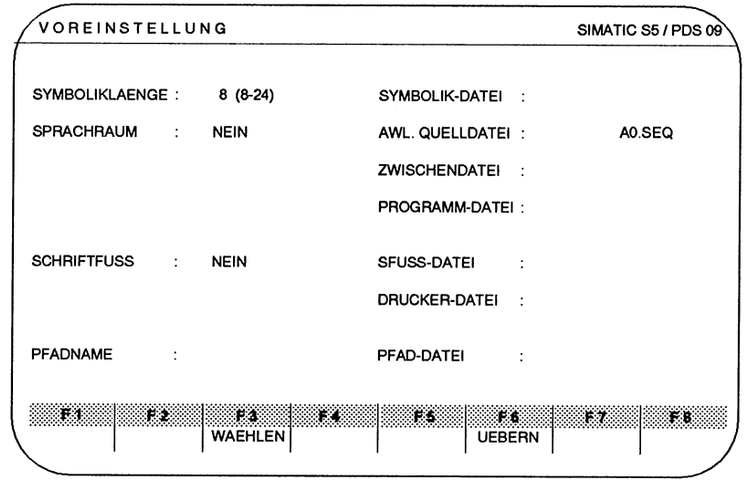
➤ Zeichenfolge *Test* eingeben,

➤ **Return-Taste** drücken.

Jetzt wird vom Programmiergerät die Festplatte als Laufwerk eingetragen und der Name mit @ aufgefüllt. Weiterhin werden automatisch die ZWISCHENDATEI (A1.SEQ), die SYMBOLIK-DATEI (Z0.INI) und die PROGRAMM-DATEI (ST.S5D) mit dem gleichen Namen aufgefüllt. Dadurch werden sie als zusammengehörig gekennzeichnet.

Ist keine SYMBOLIK-DATEI, also keine Zuordnungsliste mit dem aufgefüllten Namen vorhanden, werden Sie an drei Stellen darauf hingewiesen: Hinter dem Dateinamen steht (GESP), das Programmiergerät meldet "Datei B:TEST@@Z0.INI: Datei nicht vorhanden", und im Feld SYMBOLIKLAENGE wird aus der voreingestellten 8 eine 0.

Bedienung am Programmiergerät



3

VOREINSTELLUNG

Der Cursor blinkt in der Zeile der AWL-QUELLDATEI (A0.SEQ). In dieser Datei wird Ihre Anweisungsliste gespeichert. Sie ist als sequentielle Datei, d.h., als ASCII-Datei, gekennzeichnet und ist die Quelle für den Übersetzungslauf.

Geben Sie an dieser Stelle den Namen Ihrer Datei ein. In unserem Beispiel soll die Datei "Test" heißen.

➤ Zeichenfolge *Test* eingeben,

➤ **Return-Taste** drücken.

Jetzt wird vom Programmiergerät die Festplatte als Laufwerk eingetragen und der Name mit @ aufgefüllt. Weiterhin werden automatisch die ZWISCHENDATEI (A1.SEQ), die SYMBOLIK-DATEI (Z0.INI) und die PROGRAMM-DATEI (ST.S5D) mit dem gleichen Namen aufgefüllt. Dadurch werden sie als zusammengehörig gekennzeichnet.

Ist keine SYMBOLIK-DATEI, also keine Zuordnungsliste mit dem aufgefüllten Namen vorhanden, werden Sie an drei Stellen darauf hingewiesen: Hinter dem Dateinamen steht (GESP), das Programmiergerät meldet "Datei B:TEST@@Z0.INI: Datei nicht vorhanden", und im Feld SYMBOLIKLAENGE wird aus der voreingestellten 8 eine 0.

Namen überschreiben

Soll die Quelldatei beim Übersetzen mit einer anders benannten Symbolik-Datei verbunden und/oder in eine anders benannte Programm-Datei übersetzt werden, so können Sie jetzt die einzelnen Namen überschreiben. Nach dem Überschreiben schließen Sie jeweils mit der Return-Taste ab. Diese Dateien sind auch in anderen STEP 5-Paketen aktiv und werden beim Laden in deren Voreinstellungen eingetragen. Die Dateien für Schriftfuß, Drucker und Pfade werden automatisch namentlich an die Programm-Datei angepaßt.

➤ Cursor in der jeweiligen Zeile positionieren,

➤ **Pfeiltaste rechts** drücken,

➤ das Eingabefeld ist ausgefüllt.

Das PG prüft, ob die angegebenen Dateien vorhanden sind. Sollen andere Dateien verwendet werden, tragen Sie deren Namen ein. Sind Drucker- und Pfaddatei nicht vorhanden, werden ihre Namen bei der nächsten Cursorbewegung nach oben oder unten wieder gelöscht.

Die Zeilen PFADNAME und SCHRIFTFUSS werden wie im Paket KOP, FUP, AWL behandelt: Beim Pfadnamen geben Sie den Namen ein, für den Schriftfuß wählen Sie die Breite.

Sprachraum

Im Teilfeld SPRACHRAUM sollten Sie unbedingt die **Help-Taste** benutzen (stellen Sie dazu den Cursor auf einen Buchstaben von NEIN): Die **Help-Taste** zeigt Ihnen Automatisierungsgeräte (AG) und Zentralprozessoren (CPU) an, für die der Batch-Compiler AG-spezifisch übersetzt und prüft.

Tragen Sie also gegebenenfalls für den Sprachraum das Gerät aus der Liste ein, auf dem Ihr Programm laufen soll. Als weitere Hilfestellung siehe die Tabelle im Kapitel 2.2.3. Der Batch-Compiler überprüft dann beim Übersetzen in die Programmdatei, ob Ihre Anweisungsliste mit dem Sprachraum des AGs übereinstimmt.

Das Teilfeld SYMBOLIKLAENGE zeigt Ihnen die Symboliklänge der eingetragenen Symbolikdatei an. Sie können dieses reine Anzeigefeld nicht verändern.

Zum Abspeichern dieser VOREINSTELLUNG drücken Sie

➤ **UEBERN**ehmen (F6)

oder

➤ die **Übernahmetaste**.

Bedienung am Programmiergerät

Namen überschreiben

Soll die Quelldatei beim Übersetzen mit einer anders benannten Symbolik-Datei verbunden und/oder in eine anders benannte Programm-Datei übersetzt werden, so können Sie jetzt die einzelnen Namen überschreiben. Nach dem Überschreiben schließen Sie jeweils mit der Return-Taste ab. Diese Dateien sind auch in anderen STEP 5-Paketen aktiv und werden beim Laden in deren Voreinstellungen eingetragen. Die Dateien für Schriftfuß, Drucker und Pfade werden automatisch namentlich an die Programm-Datei angepaßt.

➤ Cursor in der jeweiligen Zeile positionieren,

➤ **Pfeiltaste rechts** drücken,

➤ das Eingabefeld ist ausgefüllt.

Das PG prüft, ob die angegebenen Dateien vorhanden sind. Sollen andere Dateien verwendet werden, tragen Sie deren Namen ein. Sind Drucker- und Pfaddatei nicht vorhanden, werden ihre Namen bei der nächsten Cursorbewegung nach oben oder unten wieder gelöscht.

Die Zeilen PFADNAME und SCHRIFTFUSS werden wie im Paket KOP, FUP, AWL behandelt: Beim Pfadnamen geben Sie den Namen ein, für den Schriftfuß wählen Sie die Breite.

Sprachraum

Im Teilfeld SPRACHRAUM sollten Sie unbedingt die **Help-Taste** benutzen (stellen Sie dazu den Cursor auf einen Buchstaben von NEIN): Die **Help-Taste** zeigt Ihnen Automatisierungsgeräte (AG) und Zentralprozessoren (CPU) an, für die der Batch-Compiler AG-spezifisch übersetzt und prüft.

Tragen Sie also gegebenenfalls für den Sprachraum das Gerät aus der Liste ein, auf dem Ihr Programm laufen soll. Als weitere Hilfestellung siehe die Tabelle im Kapitel 2.2.3. Der Batch-Compiler überprüft dann beim Übersetzen in die Programmdatei, ob Ihre Anweisungsliste mit dem Sprachraum des AGs übereinstimmt.

Das Teilfeld SYMBOLIKLAENGE zeigt Ihnen die Symboliklänge der eingetragenen Symbolikdatei an. Sie können dieses reine Anzeigefeld nicht verändern.

Zum Abspeichern dieser VOREINSTELLUNG drücken Sie

➤ **UEBERN**ehmen (F6)

oder

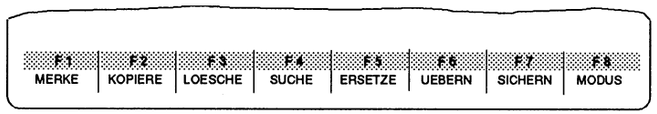
➤ die **Übernahmetaste**.

3 - 6

C79000-B8500-C677-02

Sie erhalten nun die FUNKTIONSANWAHL

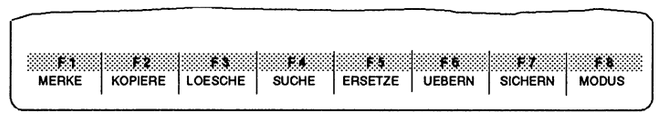
FUNKTIONSANWAHL Die FUNKTIONSANWAHL bietet Ihnen in der Softkey-Leiste die folgenden Editier- und Bearbeitungsfunktionen. Ihre Bedienung wird in den folgenden Kapiteln erläutert.



Bedienung am Programmiergerät

Sie erhalten nun die FUNKTIONSANWAHL

FUNKTIONSANWAHL Die FUNKTIONSANWAHL bietet Ihnen in der Softkey-Leiste die folgenden Editier- und Bearbeitungsfunktionen. Ihre Bedienung wird in den folgenden Kapiteln erläutert.

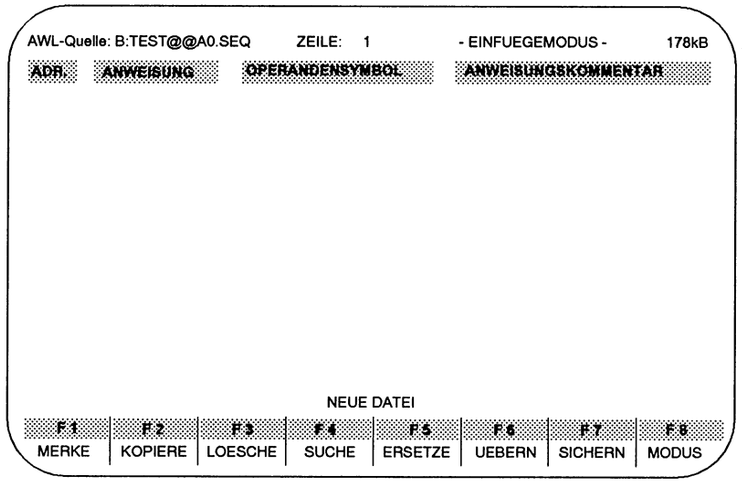


3

3.3 Editieren

Editiermodus aufrufen

➤ **EDITIEREN** (F1) aufrufen.



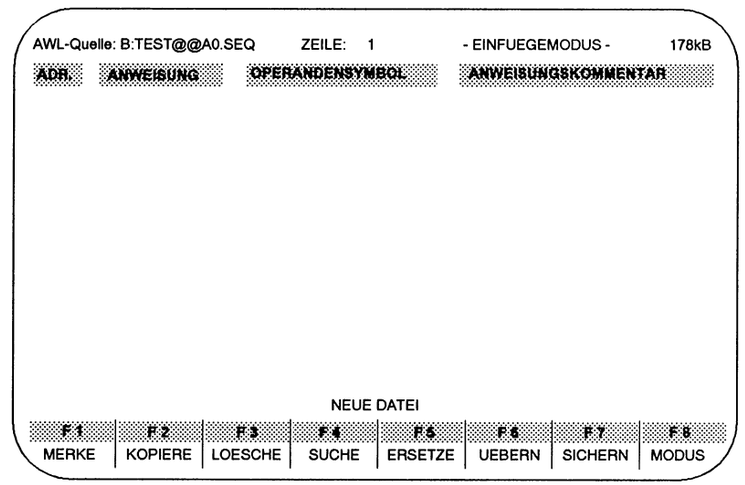
Dieser Bildschirm ist vorbereitet, eine Anweisungsliste zu editieren, d.h., neu einzugeben oder eine bereits vorhandene zur Bearbeitung (Korrekturen, Änderungen) auszugeben.

Bedienung am Programmiergerät

3.3 Editieren

Editiermodus aufrufen

➤ **EDITIEREN** (F1) aufrufen.



Dieser Bildschirm ist vorbereitet, eine Anweisungsliste zu editieren, d.h., neu einzugeben oder eine bereits vorhandene zur Bearbeitung (Korrekturen, Änderungen) auszugeben.

3.3.1 Beschreibung des Editors

Kopfzeile

Hier finden Sie:

- den Namen Ihrer voreingestellten AWL-Quelldatei und das zugehörige Laufwerk,
- die Zeilenangabe für die Cursor-Position,
- der Editiermodus Einfügen oder Überschreiben und
- die Puffergrößen des Speichers. Diese Angabe ist interessant für die Verarbeitungsgeschwindigkeit.

Editierfeld

Das Editierfeld ist in vier Spalten eingeteilt, deren Breite nicht verändert werden kann. Werte und vorgesehener Inhalt der Spalten sind hier kurz zusammengestellt:

ADR 4 Zeichen	ANWEISUNG 13 Zeichen	OPERANDENSYMBOL 24 Zeichen (max. Symbollänge)	ANWEISUNGSKOMMENTAR 32 Zeichen
Adressen, Sprungmarken	Operationen, absolute Operanden, Konstanten	Symbole deren Wert	Kommentare

Fußzeile

In dieser Zeile stehen alle Meldungen des Geräts, z.B. "Neue Datei", wenn Sie eine neue Anweisungsliste anlegen.

Bedienung am Programmiergerät

3.3.1 Beschreibung des Editors

Kopfzeile

Hier finden Sie:

- den Namen Ihrer voreingestellten AWL-Quelldatei und das zugehörige Laufwerk,
- die Zeilenangabe für die Cursor-Position,
- der Editiermodus Einfügen oder Überschreiben und
- die Puffergrößen des Speichers. Diese Angabe ist interessant für die Verarbeitungsgeschwindigkeit.

Editierfeld

Das Editierfeld ist in vier Spalten eingeteilt, deren Breite nicht verändert werden kann. Werte und vorgesehener Inhalt der Spalten sind hier kurz zusammengestellt:

ADR 4 Zeichen	ANWEISUNG 13 Zeichen	OPERANDENSYMBOL 24 Zeichen (max. Symbollänge)	ANWEISUNGSKOMMENTAR 32 Zeichen
Adressen, Sprungmarken	Operationen, absolute Operanden, Konstanten	Symbole deren Wert	Kommentare

Fußzeile

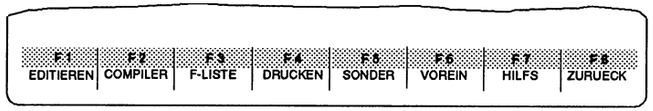
In dieser Zeile stehen alle Meldungen des Geräts, z.B. "Neue Datei", wenn Sie eine neue Anweisungsliste anlegen.

C79000-B8500-C677-02

3 - 9

Softkey-Leiste

Diese Editierfunktionen sind mit denen des SYMBOLIK-EDITORS identisch. Sie dienen der Erstellung und Bearbeitung einer Anweisungsliste.



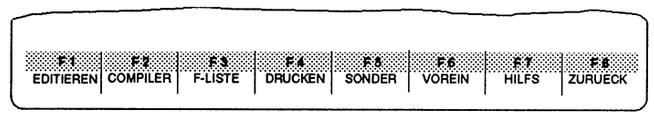
Funktionen der Softkey-Leiste EDITIEREN

Die nachfolgende Graphik gibt einen Überblick über die "Werkzeuge", die in den einzelnen Editierfunktionen zur Verfügung stehen. Drücken Sie eine der Tasten des Editiermenüs, wird das entsprechende Menü (am Ende des Pfeils) ausgegeben.

Bedienung am Programmiergerät

Softkey-Leiste

Diese Editierfunktionen sind mit denen des SYMBOLIK-EDITORS identisch. Sie dienen der Erstellung und Bearbeitung einer Anweisungsliste.

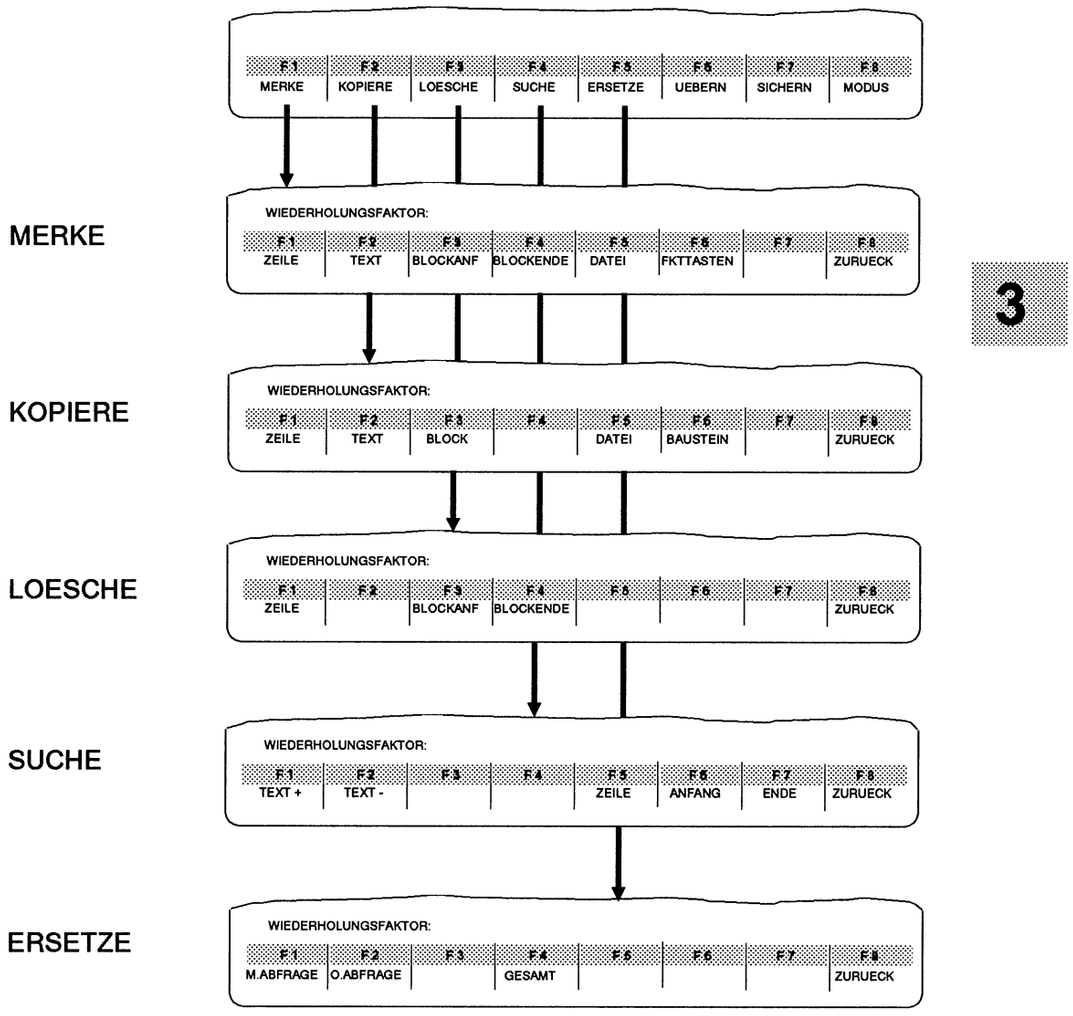


Funktionen der Softkey-Leiste EDITIEREN

Die nachfolgende Graphik gibt einen Überblick über die "Werkzeuge", die in den einzelnen Editierfunktionen zur Verfügung stehen. Drücken Sie eine der Tasten des Editiermenüs, wird das entsprechende Menü (am Ende des Pfeils) ausgegeben.

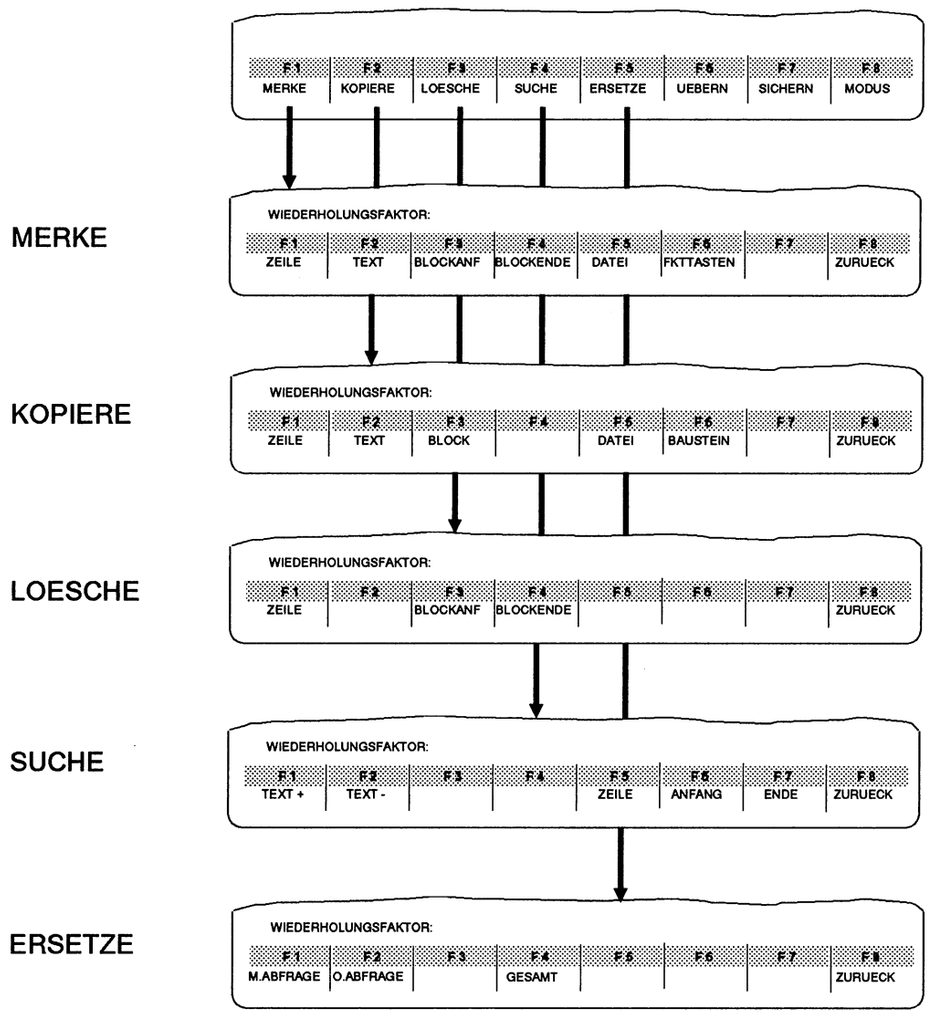
3 - 10

C79000-B8500-C677-02



3

Bedienung am Programmiergerät



3

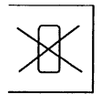
Sondertasten

Neben diesen Funktionen haben Sie mit den Sondertasten weitere "Werkzeuge", um Ihre Datei zu bearbeiten

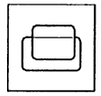
Hardkeys

Zur Cursorbewegung stehen Ihnen die Cursor-Tasten in vollem Umfang zur Verfügung.

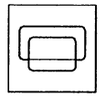
Das Zeichen auf dem Sie stehen, löschen Sie mit (bzw. Taste DEL)



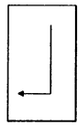
In Ihrer Datei blättern Sie vorwärts mit (Text nach oben schieben)



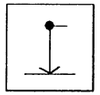
und rückwärts mit (Text nach unten schieben)



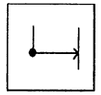
Im EINFUEGEMODUS erhalten Sie eine neue Zeile unterhalb der Cursorposition mit



eine neue Zeile oberhalb der Cursorposition mit "vertikal spreizen",



ein Leerzeichen auch mit "horizontal spreizen" (neben der Leerzeichentaste).



Bedienung am Programmiergerät

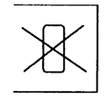
Sondertasten

Neben diesen Funktionen haben Sie mit den Sondertasten weitere "Werkzeuge", um Ihre Datei zu bearbeiten

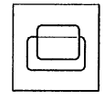
Hardkeys

Zur Cursorbewegung stehen Ihnen die Cursor-Tasten in vollem Umfang zur Verfügung.

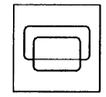
Das Zeichen auf dem Sie stehen, löschen Sie mit (bzw. Taste DEL)



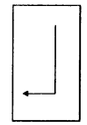
In Ihrer Datei blättern Sie vorwärts mit (Text nach oben schieben)



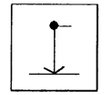
und rückwärts mit (Text nach unten schieben)



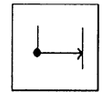
Im EINFUEGEMODUS erhalten Sie eine neue Zeile unterhalb der Cursorposition mit



eine neue Zeile oberhalb der Cursorposition mit "vertikal spreizen",



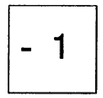
ein Leerzeichen auch mit "horizontal spreizen" (neben der Leerzeichentaste).



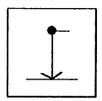
3 - 12

C79000-B8500-C677-02

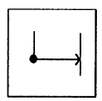
Sie löschen das Zeichen
links vom Cursor mit
(oder Taste "-")



Im UEBERSCHREIBMODUS
fügen Sie eine neue Zeile nur ein mit



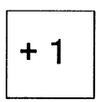
ein weiteres Leerzeichen nur mit



3

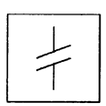
Wollen Sie innerhalb eines Kommandos in der Fußzeile den gesamtenText zwischen den beiden Doppelpunkten löschen, benutzen Sie dazu:

(oder Taste "+")

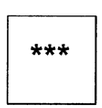


Mit der Abbruchtaste beenden Sie jede aktivierte Funktion, verlieren aber dabei möglicherweise Daten. Haben Sie z.B. eine Datei korrigiert und drücken Sie dann die Abbruchtaste, werden alle Veränderungen verworfen.

(oder Taste ESC)



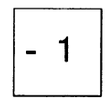
Die Netzwerkendetaste



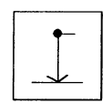
erzeugt "***", d. h. einen Netzwerkende-Befehl, falls der Cursor im Feld "Anweisung" steht. Außerhalb dieses Feldes sowie bei Zusatz- und Netzwerkkommentaren ist die Taste gesperrt.

Bedienung am Programmiergerät

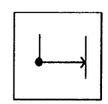
Sie löschen das Zeichen
links vom Cursor mit
(oder Taste "-")



Im UEBERSCHREIBMODUS
fügen Sie eine neue Zeile nur ein mit



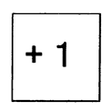
ein weiteres Leerzeichen nur mit



3

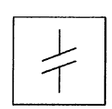
Wollen Sie innerhalb eines Kommandos in der Fußzeile den gesamtenText zwischen den beiden Doppelpunkten löschen, benutzen Sie dazu:

(oder Taste "+")

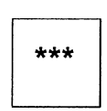


Mit der Abbruchtaste beenden Sie jede aktivierte Funktion, verlieren aber dabei möglicherweise Daten. Haben Sie z.B. eine Datei korrigiert und drücken Sie dann die Abbruchtaste, werden alle Veränderungen verworfen.

(oder Taste ESC)



Die Netzwerkendetaste



erzeugt "***", d. h. einen Netzwerkende-Befehl, falls der Cursor im Feld "Anweisung" steht. Außerhalb dieses Feldes sowie bei Zusatz- und Netzwerkkommentaren ist die Taste gesperrt.

3.3.2 Die Steuerzeichen des AWL-Editors/Batch-Compilers und ihre Schreibkonventionen

Der AWL-EDITOR verlangt für bestimmte Eingaben eine Reihe von Steuerzeichen, damit die Übersetzung der Anweisungsliste in eine STEP 5-Programmdatei möglich ist. Zum Beispiel müssen Netzwerküberschriften und -kommentare, Aktualoperanden und Bausteinkennungen als solche gekennzeichnet werden.

Spalte ANWEISUNGEN Steuerzeichen	Kennzeichen für	Schreibkonventionen mit Beispielen
#TAB	Quelldatei ohne echte Tabulatoren	#TAB 1,6,21,46
#TY	AG-Typ	mit Leerzeichen #TY_AG155U #TY_CPU928
#PBn #OBn #FBn, #FXn #DBn, #DXn (#SBn, kein GRAPH 5-Baustein)	Programmbausteinanfang Organisationsbausteinanf. Funktionsbausteinanfang Datenbausteinanfang Schrittbausteinanfang	ohne Leerzeichen #PB11 #OB1 #FB25, #FX12 #DB5, #DX33 #SB3
#BI	Bibliotheksnummer	mit Leerzeichen #BI_12345 nicht höher als 65535
#N	Name eines Funktionsbausteins	mit Leerzeichen #N_GARAGE max. 6 Zeichen

Bedienung am Programmiergerät

3.3.2 Die Steuerzeichen des AWL-Editors/Batch-Compilers und ihre Schreibkonventionen

Der AWL-EDITOR verlangt für bestimmte Eingaben eine Reihe von Steuerzeichen, damit die Übersetzung der Anweisungsliste in eine STEP 5-Programmdatei möglich ist. Zum Beispiel müssen Netzwerküberschriften und -kommentare, Aktualoperanden und Bausteinkennungen als solche gekennzeichnet werden.

Spalte ANWEISUNGEN Steuerzeichen	Kennzeichen für	Schreibkonventionen mit Beispielen
#TAB	Quelldatei ohne echte Tabulatoren	#TAB 1,6,21,46
#TY	AG-Typ	mit Leerzeichen #TY_AG155U #TY_CPU928
#PBn #OBn #FBn, #FXn #DBn, #DXn (#SBn, kein GRAPH 5-Baustein)	Programmbausteinanfang Organisationsbausteinanf. Funktionsbausteinanfang Datenbausteinanfang Schrittbausteinanfang	ohne Leerzeichen #PB11 #OB1 #FB25, #FX12 #DB5, #DX33 #SB3
#BI	Bibliotheksnummer	mit Leerzeichen #BI_12345 nicht höher als 65535
#N	Name eines Funktionsbausteins	mit Leerzeichen #N_GARAGE max. 6 Zeichen

3 - 14 C79000-B8500-C677-02

Die Übersichtstabelle listet diese Steuerzeichen auf. Sie zeigt die Reihenfolge, die für eine problemlose Übersetzung in die Zwischen- und Programmdatei festgelegt ist. Weiterhin informiert sie über die Schreibkonventionen (_ repräsentiert ein Leerzeichen) und die Position der Steuerzeichen innerhalb der Anweisungsliste und sie gibt weitere Erläuterungen.

Position innerhalb der Anweisungsliste	Erläuterungen
Immer die erste Zeile der Datei	Ermöglicht die Übersetzung von Dateien, die mit einem Fremdeditor, z. B. 1st Wordplus, erstellt wurden Gilt nur für den Compiler, nicht für den AWL-Editor!
immer die erste Anweisung einer Datei	Eventuelle Kommentare stehen nur in der AWL-Quelldatei, sie werden nicht übersetzt und gehen bei der Rückübersetzung verloren.
Beginn eines Bausteins; nach einem BE (Bausteinende s.u., Operationen)	Wertebereich: n= 0 - 255, je nach AG-Typ Wollen Sie nach einem Bausteinende weitere Anweisungen eingeben, muß diesen ein neuer Bausteinanfang vorausgehen, weil sonst diese Anweisungen beim Übersetzen im Programmiergerät verloren gehen. DB0, DB1, DB2 sind nicht zulässig.
nach dem Bausteinanfang oder nach dem Baustein-Namen (s.u., #N)	Für Ihre eigenen Bibliotheksnummern; die Nummern von Standard-Funktionsbausteinen können und brauchen Sie nicht eingeben. Eventuelle Kommentare stehen nur in der AWL-Quelldatei, sie werden nicht übersetzt und gehen bei der Rückübersetzung verloren.
vor oder nach der Bibliotheksnummer, aber zu Beginn des Bausteins	

3

Bedienung am Programmiergerät

Die Übersichtstabelle listet diese Steuerzeichen auf. Sie zeigt die Reihenfolge, die für eine problemlose Übersetzung in die Zwischen- und Programmdatei festgelegt ist. Weiterhin informiert sie über die Schreibkonventionen (_ repräsentiert ein Leerzeichen) und die Position der Steuerzeichen innerhalb der Anweisungsliste und sie gibt weitere Erläuterungen.

Position innerhalb der Anweisungsliste	Erläuterungen
Immer die erste Zeile der Datei	Ermöglicht die Übersetzung von Dateien, die mit einem Fremdeditor, z. B. 1st Wordplus, erstellt wurden Gilt nur für den Compiler, nicht für den AWL-Editor!
immer die erste Anweisung einer Datei	Eventuelle Kommentare stehen nur in der AWL-Quelldatei, sie werden nicht übersetzt und gehen bei der Rückübersetzung verloren.
Beginn eines Bausteins; nach einem BE (Bausteinende s.u., Operationen)	Wertebereich: n= 0 - 255, je nach AG-Typ Wollen Sie nach einem Bausteinende weitere Anweisungen eingeben, muß diesen ein neuer Bausteinanfang vorausgehen, weil sonst diese Anweisungen beim Übersetzen im Programmiergerät verloren gehen. DB0, DB1, DB2 sind nicht zulässig.
nach dem Bausteinanfang oder nach dem Baustein-Namen (s.u., #N)	Für Ihre eigenen Bibliotheksnummern; die Nummern von Standard-Funktionsbausteinen können und brauchen Sie nicht eingeben. Eventuelle Kommentare stehen nur in der AWL-Quelldatei, sie werden nicht übersetzt und gehen bei der Rückübersetzung verloren.
vor oder nach der Bibliotheksnummer, aber zu Beginn des Bausteins	

3

C79000-B8500-C677-02

3 - 15

Spalte ANWEISUNGEN Steuerzeichen	Kennzeichen für	Schreibkonventionen mit Beispielen
#UB	Netzwerküberschrift	Das Steuerzeichen steht in der Spalte ANWEISUNG, der Überschrifttext in der Spalte ANWEISUNGS-KOMMENTAR.
( )	Formalparametertyp	Der Formelparametertyp muß in Klammern stehen. (D) (E)
,	Aktualoperanden zum Parametrieren eines Funktionsbausteins	erstes Zeichen in der Spalte; der Parameter folgt unmittelbar ,E1.0
#I	Include-Datei	mit Leerzeichen, Laufwerks-angabe und den ersten sechs Zeichen des Dateinamens #I_A:UEBUNG
#	Symbolischer Bausteinname	

Bedienung am Programmiergerät

Spalte ANWEISUNGEN Steuerzeichen	Kennzeichen für	Schreibkonventionen mit Beispielen
#UB	Netzwerküberschrift	Das Steuerzeichen steht in der Spalte ANWEISUNG, der Überschrifttext in der Spalte ANWEISUNGS-KOMMENTAR.
( )	Formalparametertyp	Der Formelparametertyp muß in Klammern stehen. (D) (E)
,	Aktualoperanden zum Parametrieren eines Funktionsbausteins	erstes Zeichen in der Spalte; der Parameter folgt unmittelbar ,E1.0
#I	Include-Datei	mit Leerzeichen, Laufwerks-angabe und den ersten sechs Zeichen des Dateinamens #I_A:UEBUNG
#	Symbolischer Bausteinname	

3 - 16 C79000-B8500-C677-02

Position innerhalb der Anweisungsliste	Erläuterungen
nur am Anfang eines Netzwerks	Diese Kommentartexte werden mit in die Programmmdatei übernommen. Wollen Sie nähere Informationen zur Kommentierung eines STEP 5-Programms, schlagen Sie bitten in der STEP 5-Beschreibung nach, im Handbuch Band 2 Ihres Programmiergeräts.
Direkt unter dem Bausteinnamen	
Innerhalb eines Bausteins	
nur an Bausteingrenzen: vor dem ersten Baustein oder zwischen BE und #PBn	Das Steuerzeichen ermöglicht, andere Dateien einzubinden (Include). Diese Dateien müssen aber als Zwischendateien vorhanden sein, d.h. entweder im AWL-Editor mit der Übernahmetaste abgeschlossen oder über die Rückübersetzung erstellt worden sein. Achten Sie darauf, daß keine gleichen Bausteinnamen in den Dateien vorkommen, die verbunden werden sollen. Denn beim Erzeugen der Programm-Datei überschreibt der letzte Baustein gleichen Namens den vorhergehenden. Die Include-Datei wird beim Übersetzen mit der voreingestellten Symbolik-Datei verbunden. Diese muß deshalb auch die Include-Datei mit Zuordnungen versorgen können.

Bedienung am Programmiergerät

Position innerhalb der Anweisungsliste | Erläuterungen
nur am Anfang eines Netzwerks | Diese Kommentartexte werden mit in die Programmmdatei übernommen. Wollen Sie nähere Informationen zur Kommentierung eines STEP 5-Programms, schlagen Sie bitten in der STEP 5-Beschreibung nach, im Handbuch Band 2 Ihres Programmiergeräts.
Direkt unter dem Bausteinnamen |
Innerhalb eines Bausteins |
nur an Bausteingrenzen: vor dem ersten Baustein oder zwischen BE und #PBn | Das Steuerzeichen ermöglicht, andere Dateien einzubinden (Include). Diese Dateien müssen aber als Zwischendateien vorhanden sein, d.h. entweder im AWL-Editor mit der Übernahmetaste abgeschlossen oder über die Rückübersetzung erstellt worden sein. Achten Sie darauf, daß keine gleichen Bausteinnamen in den Dateien vorkommen, die verbunden werden sollen. Denn beim Erzeugen der Programm-Datei überschreibt der letzte Baustein gleichen Namens den vorhergehenden. Die Include-Datei wird beim Übersetzen mit der voreingestellten Symbolik-Datei verbunden. Diese muß deshalb auch die Include-Datei mit Zuordnungen versorgen können.

3

C79000-B8500-C677-02

3 - 17

Spalte ADR. Steuerzeichen	Kennzeichen für	Schreibkonventionen mit Beispielen
*	Netzwerkkommentar	Das Steuerzeichen steht nur am Anfang eines Netzwerks; besteht eine Netzwerküberschrift, muß sie unmittelbar vorausgehen.
;	Zusatzkommentar	Das Steuerzeichen steht in der Spalte ADR., für den Text steht die gesamte Bildschirmbreite, unabhängig von Spalten, zur Verfügung.

Bedienung am Programmiergerät

Spalte ADR. Steuerzeichen	Kennzeichen für	Schreibkonventionen mit Beispielen
*	Netzwerkkommentar	Das Steuerzeichen steht nur am Anfang eines Netzwerks; besteht eine Netzwerküberschrift, muß sie unmittelbar vorausgehen.
;	Zusatzkommentar	Das Steuerzeichen steht in der Spalte ADR., für den Text steht die gesamte Bildschirmbreite, unabhängig von Spalten, zur Verfügung.

3 - 18

C79000-B8500-C677-02

Position innerhalb der Anweisungsliste	Erläuterungen
an beliebiger Stelle innerhalb eines Bausteins	Diese Zusatzkommentare stehen nur in der AWL-Quelldatei. Beim Übersetzen bleiben sie unberücksichtigt. Rückübersetzen Sie in die gleiche AWL-Quelldatei, sind diese Kommentare dort verloren.

Bedienung am Programmiergerät

Position innerhalb der Anweisungsliste | Erläuterungen

an beliebiger Stelle innerhalb eines Bausteins | Diese Zusatzkommentare stehen nur in der AWL-Quelldatei. Beim Übersetzen bleiben sie unberücksichtigt. Rückübersetzen Sie in die gleiche AWL-Quelldatei, sind diese Kommentare dort verloren.

3

C79000-B8500-C677-02

3 - 19

3.3.3 Die STEP 5-Operationen im AWL-Editor/ Batch-Compiler und ihre Schreibkonventionen

Im AWL-Editor/Batch-Compiler ist der volle Umfang der STEP 5-Operationen möglich. Nur der Sprachraum des Automatisierungsgeräts bzw. der CPU bilden die Grenzen. Ziehen Sie deshalb beim Programmieren die Operationsliste Ihres Geräts zu Rate.

	ADRESSE	ANWEISUNG
Operation mit absolutem Operand		Operation und absoluter Operand *U_E1.2* formatfreie Eingabe
mit symbolischem Operand		Operation *U*
Operation mit Daten		Operation und Datenformat *L_KT* formatfreie Eingabe
Daten	Adresse *11*	Datenformat *KH* *KF* *KC* oder *C* *KG* *KT* *KZ* *KY* oder *A* *KM*

Bedienung am Programmiergerät

3.3.3 Die STEP 5-Operationen im AWL-Editor/ Batch-Compiler und ihre Schreibkonventionen

Im AWL-Editor/Batch-Compiler ist der volle Umfang der STEP 5-Operationen möglich. Nur der Sprachraum des Automatisierungsgeräts bzw. der CPU bilden die Grenzen. Ziehen Sie deshalb beim Programmieren die Operationsliste Ihres Geräts zu Rate.

	ADRESSE	ANWEISUNG
Operation mit absolutem Operand		Operation und absoluter Operand *U_E1.2* formatfreie Eingabe
mit symbolischem Operand		Operation *U*
Operation mit Daten		Operation und Datenformat *L_KT* formatfreie Eingabe
Daten	Adresse *11*	Datenformat *KH* *KF* *KC* oder *C* *KG* *KT* *KZ* *KY* oder *A* *KM*

3 - 20

C79000-B8500-C677-02

Die folgende Tabelle, die sich an den Spalten des Bildschirms orientiert, stellt die Schreibkonventionen für die absolute und symbolische Programmierung zusammen.

OPERANDENSYMBOL	ANWEISUNGSKOMMENTAR
	Taste "aussen auf"
Symbol *T-AUF A* ohne Bindestrich	
Wert des Datums *005.2*	
Wert, 1 Datenwort pro Zeile *6248* *+ 13512* 'Anzeige' nur einfaches Anführungszeichen, bis zu 11 Datenworte pro Zeile -1169368-38 max. 1 Datendoppelwort pro Zeile 123.1 735 125,018 00011100 11101111	

Bedienung am Programmiergerät

Die folgende Tabelle, die sich an den Spalten des Bildschirms orientiert, stellt die Schreibkonventionen für die absolute und symbolische Programmierung zusammen.

OPERANDENSYMBOL	ANWEISUNGSKOMMENTAR
	Taste "aussen auf"
Symbol *T-AUF A* ohne Bindestrich	
Wert des Datums *005.2*	
Wert, 1 Datenwort pro Zeile *6248* *+ 13512* 'Anzeige' nur einfaches Anführungszeichen, bis zu 11 Datenworte pro Zeile -1169368-38 max. 1 Datendoppelwort pro Zeile 123.1 735 125,018 00011100 11101111	

3

C79000-B8500-C677-02

3 - 21

	ADRESSE	ANWEISUNG
Formaloperanden: Definition	Name *ZEIT* *TA-A* *MAB* max. 4 Zeichen	Typ *(D)* *(E)* *(A)* in Klammern
Operation mit Formaloperanden	Operation und Formaloperand *U_=TA-A* *=_=MAB* formatfreie Eingabe, dem Formaloperanden muß unmittelbar ein Gleichheitszeichen vorausgehen.	
Aktualoperanden(=Parameter für FBs), absolut	Steuerzeichen mit Operand *,E1.2* *,DW1* ohne Leerzeichen	
symbolisch	Steuerzeichen ,	
Daten	Steuerzeichen mit Datentyp *,KT*	
Sprungmarken	Marke *EIN* *M003*	
relative Adressen, Datenwortadressen	*17*	
Bausteinende		

Bedienung am Programmiergerät

	ADRESSE	ANWEISUNG
Formaloperanden: Definition	Name *ZEIT* *TA-A* *MAB* max. 4 Zeichen	Typ *(D)* *(E)* *(A)* in Klammern
Operation mit Formaloperanden	Operation und Formaloperand *U_=TA-A* *=_=MAB* formatfreie Eingabe, dem Formaloperanden muß unmittelbar ein Gleichheitszeichen vorausgehen.	
Aktualoperanden(=Parameter für FBs), absolut	Steuerzeichen mit Operand *,E1.2* *,DW1* ohne Leerzeichen	
symbolisch	Steuerzeichen ,	
Daten	Steuerzeichen mit Datentyp *,KT*	
Sprungmarken	Marke *EIN* *M003*	
relative Adressen, Datenwortadressen	*17*	
Bausteinende		

3 - 22

C79000-B8500-C677-02

OPERANDENSYMBOL	ANWEISUNGSKOMMENTAR
Symbol *MAB*	
Wert *005.2*	
BE	

Bedienung am Programmiergerät

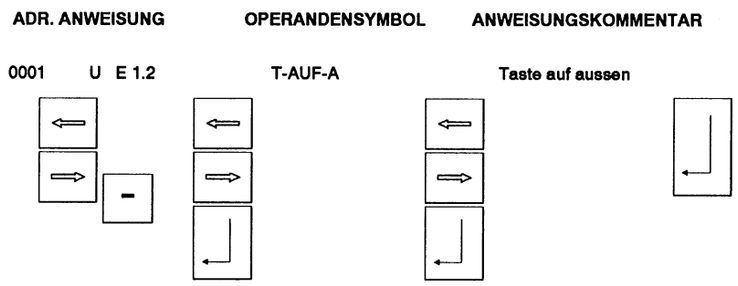
OPERANDENSYMBOL | ANWEISUNGSKOMMENTAR
Symbol MAB
Wert 005.2
BE

3

C79000-B8500-C677-02

3 - 23

Zwischen den Spalten des Bildschirms bewegen Sie sich mit Doppelpfeiltasten und Return. Die Return-Taste setzt den Cursor immer auf das 1. Zeichen der ANWEISUNGsspalte.



Symbole

Programmieren Sie symbolisch, beachten Sie bitte, daß Sie im Gegensatz zum Paket KOP, FUP, AWL keinen Bindestrich vor das Symbol setzen dürfen. Den Bausteinanfang können Sie nur dann als Symbol eingeben, wenn dafür eine Zuordnung von Bausteintyp und Nummer zu einem Symbol existiert. Ist dies nicht der Fall, programmieren Sie den Bausteinanfang absolut, z.B. #PB3, denn der Batch-Compiler braucht zum Erstellen der Zwischendatei den genauen Bausteintyp und seine Nummer.

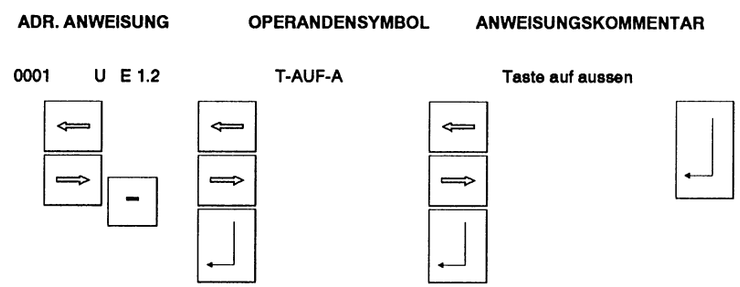
Die Symbole, die Sie im AWL-Editor verwenden, müssen mit denen der Symbolikdatei absolut übereinstimmen. Dies gilt auch für Leerzeichen:

_NOTAUS ≠ NOTAUS

Weitere Unterschiede zum Paket KOP, FUP, AWL sind
- Steuerzeichen,
- selbst einzugebende Leerzeichen bei Operationen,
- Datenkonstanten und Wert stehen in verschiedenen Spalten.

Bedienung am Programmiergerät

Zwischen den Spalten des Bildschirms bewegen Sie sich mit Doppelpfeiltasten und Return. Die Return-Taste setzt den Cursor immer auf das 1. Zeichen der ANWEISUNGsspalte.



Symbole

Programmieren Sie symbolisch, beachten Sie bitte, daß Sie im Gegensatz zum Paket KOP, FUP, AWL keinen Bindestrich vor das Symbol setzen dürfen. Den Bausteinanfang können Sie nur dann als Symbol eingeben, wenn dafür eine Zuordnung von Bausteintyp und Nummer zu einem Symbol existiert. Ist dies nicht der Fall, programmieren Sie den Bausteinanfang absolut, z.B. #PB3, denn der Batch-Compiler braucht zum Erstellen der Zwischendatei den genauen Bausteintyp und seine Nummer.

Die Symbole, die Sie im AWL-Editor verwenden, müssen mit denen der Symbolikdatei absolut übereinstimmen. Dies gilt auch für Leerzeichen:

_NOTAUS ≠ NOTAUS

Weitere Unterschiede zum Paket KOP, FUP, AWL sind
- Steuerzeichen,
- selbst einzugebende Leerzeichen bei Operationen,
- Datenkonstanten und Wert stehen in verschiedenen Spalten.

3.3.4 Eingabe von Programmbausteinen

Programmierbeispiel

Die auf der nachfolgenden Seite abgedruckte Datei soll als Arbeitsbeispiel dienen. Anhand dieses Beispiels erklären wir Ihnen die Bedienung des AWL-Editors/Batch-Compilers und der Funktionen dieses Pakets.

Das Programm steuert ein Garagentor. Von außen wird es mit einem Schlüssel und jeweils einer Taste geöffnet bzw. geschlossen, von innen genügt das Drücken der Tasten "auf" und "zu". Das Tor wird mit einer Verzögerung von 5 Sekunden geschlossen.

Bedienung am Programmiergerät

3.3.4 Eingabe von Programmbausteinen

Programmierbeispiel

Die auf der nachfolgenden Seite abgedruckte Datei soll als Arbeitsbeispiel dienen. Anhand dieses Beispiels erklären wir Ihnen die Bedienung des AWL-Editors/Batch-Compilers und der Funktionen dieses Pakets.

Das Programm steuert ein Garagentor. Von außen wird es mit einem Schlüssel und jeweils einer Taste geöffnet bzw. geschlossen, von innen genügt das Drücken der Tasten "auf" und "zu". Das Tor wird mit einer Verzögerung von 5 Sekunden geschlossen.

3

C79000-B8500-C677-02

3 - 25

```
AWL-Quelle: B:TEST@@A0.SEQ

ADR.     ANWEISUNG        OPERANDENSYMBOL          ANWEISUNGSKOMMENTAR
         #PB1
         #UB                                       OEFFNEN VON AUSSEN UND INNEN
*DIE TASTE "AUF AUSSEN" UND DER SCHLUESSELSCHALTER ODER DIE TASTE "AUF INNEN"
*SETZEN DEN MOTOR AUFWAERTS IN BEWEGUNG. DER MOTOR LAEUFT SOLANGE, BIS DER
*OBERE ENDSCHALTER ERREICHT IST, ODER DIE SICHERHEITSTASTE STOP GEDRUECKT
*WIRD.
         U(
         U       E  1.2                            TASTE AUF AUSSEN
         U       E  1.4                            SCHLUESSELSCHALTER
         O       E  1.5                            TASTE AUF INNEN
         )
         UN      E  1.0                            ENDSCHALTER OBEN
         S       A  1.0                            MOTOR AUFWAERTS
         ***
         #UB                                       OEFFNEN VON AUSSEN UND INNEN
*RUECKSETZEN DES AUSGANGS MOTOR AUFWAERTS.
         O       E 1.0                             ENDSCHALTER OBEN
         O       E 1.7                             SICHERHEITSSCHALTER STOP
         R       A 1.0                             MOTOR AUFWAERTS
         ***
         #UB                                       SCHLIESSEN VON AUSSEN ODER INNEN
*DIE TASTE "ZU AUSSEN" UND DER SCHLUESSELSCHALTER ODER DIE TASTE "ZU INNEN"
*SETZEN DEN MOTOR ABWAERTS MIT EINER EINSCHALTVERZOEGERUNG VON 5 SEK. IN
*BEWEGUNG. DER MOTOR ABWAERTS LAEUFT SOLANGE, BIS DER ENDSCHALTER UNTEN
*ERREICHT IST, ODER DIE SICHERHEITSTASTE STOP GEDRUECKT WIRD.
         U(
         U               T-ZU A
         U               SCHLUESS
         O               T-ZU I
         )
         UN              END-UNT
         L   KT          005.2
         SS              EIN-VZOE
         O               END-UNT
         O               STOP
         R               EIN-VZOE
         L               EIN-VZOE
         T   MW 100
         LC              EIN-VZOE
         T   MW 102
         U               EIN-VZOE
         =               MOT-AB
         BE
```


Bedienung am Programmiergerät

AWL-Quelle: B:TEST@@A0.SEQ

ADR.   ANWEISUNG   OPERANDENSYMBOL   ANWEISUNGSKOMMENTAR
         #PB1
         #UB   OEFFNEN VON AUSSEN UND INNEN
*DIE TASTE "AUF AUSSEN" UND DER SCHLUESSELSCHALTER ODER DIE TASTE "AUF INNEN"
*SETZEN DEN MOTOR AUFWAERTS IN BEWEGUNG. DER MOTOR LAEUFT SOLANGE, BIS DER
*OBERE ENDSCHALTER ERREICHT IST, ODER DIE SICHERHEITSTASTE STOP GEDRUECKT
*WIRD.
         U(
         U   E 1.2   TASTE AUF AUSSEN
         U   E 1.4   SCHLUESSELSCHALTER
         O   E 1.5   TASTE AUF INNEN
)
         UN   E 1.0   ENDSCHALTER OBEN
         S   A 1.0   MOTOR AUFWAERTS
         ***
         #UB   OEFFNEN VON AUSSEN UND INNEN
*RUECKSETZEN DES AUSGANGS MOTOR AUFWAERTS.
         O   E 1.0   ENDSCHALTER OBEN
         O   E 1.7   SICHERHEITSSCHALTER STOP
         R   A 1.0   MOTOR AUFWAERTS
         ***
         #UB   SCHLIESSEN VON AUSSEN ODER INNEN
*DIE TASTE "ZU AUSSEN" UND DER SCHLUESSELSCHALTER ODER DIE TASTE "ZU INNEN"
*SETZEN DEN MOTOR ABWAERTS MIT EINER EINSCHALTVERZOEGERUNG VON 5 SEK. IN
*BEWEGUNG. DER MOTOR ABWAERTS LAEUFT SOLANGE, BIS DER ENDSCHALTER UNTEN
*ERREICHT IST, ODER DIE SICHERHEITSTASTE STOP GEDRUECKT WIRD.
         U(
         U   T-ZU A
         U   SCHLUESS
         O   T-ZU I
)
         UN   END-UNT
         L   KT   005.2
         SS   EIN-VZOE
         O   END-UNT
         O   STOP
         R   EIN-VZOE
         L   EIN-VZOE
         T   MW 100
         LC   EIN-VZOE
         T   MW 102
         U   EIN-VZOE
         =   MOT-AB
         BE

3 - 26

C79000-B8500-C677-02

Eingabe

Voraussetzungen: Das Paket AWL-Editor/Batch-Compiler ist geladen, die Voreinstellung ausgefüllt und Sie haben die Editierfunktion aufgerufen.

MODUS festlegen (F8)

Sie können mit dieser Funktion zwischen zwei Editiermodi wählen: Einfügen oder Überschreiben. Welchen Sie gewählt haben, zeigt Ihnen das PG in der Kopfzeile des Bildschirms.

➤ **MODUS** (F8) drücken, bis der gewünschte Modus aktiviert ist.

Bausteinanfang Gehen Sie folgendermaßen vor (Die Zeichenfolgen, die Sie eingeben, sind *kursiv*, die zu benutzenden Funktionen **fett** gedruckt.):

➤ *#PB1* als Bausteinanfang eingeben,

➤ **Return-Taste** zweimal drücken, die Leerzeile dient der optischen Gliederung bei der Eingabe,

➤ *#UB* für die Überschrift des ersten Netzwerks,

➤ **Doppelpfeiltaste rechts** zweimal drücken, um in die Spalte ANWEISUNGSKOMMENTAR zu kommen,

➤ *Oeffnen von aussen oder innen*

➤ **Return-Taste** drücken,

➤ **Doppelpfeiltaste links** einmal drücken, um in die ADRessenspalte zu kommen,

➤ ✱ als Steuerzeichen für den Netzwerkkommentar eingeben.

Jetzt können Sie den ersten Text des Beispiels einfügen. Dafür haben Sie die ganze Breite des Bildschirms zur Verfügung. Schließen Sie jede Zeile mit **Return** ab. Eine neue Textzeile beginnen Sie wie beschrieben mit der **Doppelpfeiltaste links** und ✱, denn der Cursor springt automatisch nur in die Spalte ANWEISUNG.

Schreiben Sie im Einfügemodus, so achten Sie auf das Zeilenende! Weil Sie nur innerhalb einer Zeile einfügen können, kann Text über das Zeilenende hinausrutschen und geht damit verloren.

Zur Bearbeitung Ihres Textes stehen Ihnen die Cursor- und Sondertasten zur Verfügung, die oben (Kapitel 3.3.1) beschrieben worden sind. Das Steuerzeichen ✱ läßt sich allerdings nicht über "Zeichen löschen" entfernen, sondern nur über die Funktionen **LOESCHE** und **ZEILE** (s.u. Kapitel 3.3.5, Abschnitt Bearbeitung von Zeilen).

Bedienung am Programmiergerät

Eingabe

Voraussetzungen: Das Paket AWL-Editor/Batch-Compiler ist geladen, die Voreinstellung ausgefüllt und Sie haben die Editierfunktion aufgerufen.

MODUS festlegen (F8)

Sie können mit dieser Funktion zwischen zwei Editiermodi wählen: Einfügen oder Überschreiben. Welchen Sie gewählt haben, zeigt Ihnen das PG in der Kopfzeile des Bildschirms.

➤ **MODUS** (F8) drücken, bis der gewünschte Modus aktiviert ist.

Bausteinanfang Gehen Sie folgendermaßen vor (Die Zeichenfolgen, die Sie eingeben, sind *kursiv*, die zu benutzenden Funktionen **fett** gedruckt.):

3

➤ *#PB1* als Bausteinanfang eingeben,

➤ **Return-Taste** zweimal drücken, die Leerzeile dient der optischen Gliederung bei der Eingabe,

➤ *#UB* für die Überschrift des ersten Netzwerks,

➤ **Doppelpfeiltaste rechts** zweimal drücken, um in die Spalte ANWEISUNGSKOMMENTAR zu kommen,

➤ *Oeffnen von aussen oder innen*

➤ **Return-Taste** drücken,

➤ **Doppelpfeiltaste links** einmal drücken, um in die ADRessenspalte zu kommen,

➤ ✱ als Steuerzeichen für den Netzwerkkommentar eingeben.

Jetzt können Sie den ersten Text des Beispiels einfügen. Dafür haben Sie die ganze Breite des Bildschirms zur Verfügung. Schließen Sie jede Zeile mit **Return** ab. Eine neue Textzeile beginnen Sie wie beschrieben mit der **Doppelpfeiltaste links** und ✱, denn der Cursor springt automatisch nur in die Spalte ANWEISUNG.

Schreiben Sie im Einfügemodus, so achten Sie auf das Zeilenende! Weil Sie nur innerhalb einer Zeile einfügen können, kann Text über das Zeilenende hinausrutschen und geht damit verloren.

Zur Bearbeitung Ihres Textes stehen Ihnen die Cursor- und Sondertasten zur Verfügung, die oben (Kapitel 3.3.1) beschrieben worden sind. Das Steuerzeichen ✱ läßt sich allerdings nicht über "Zeichen löschen" entfernen, sondern nur über die Funktionen **LOESCHE** und **ZEILE** (s.u. Kapitel 3.3.5, Abschnitt Bearbeitung von Zeilen).

C79000-B8500-C677-02

3 - 27

Absolute Operanden Nach dem Netzwerkkommentar folgen die einzelnen Anweisungen für das Öffnen des Garagentors, zunächst in absoluter Form mit Anweisungskommentar. Achten Sie darauf, daß Sie die Leerzeichen selbst eingeben müssen.

- ➤ *U(*
- ➤ **Return-Taste** drücken,
- ➤ *U*
- ➤ Leerzeichen
- ➤ *E 1.2*
- ➤ **Doppelpfeiltaste rechts** zweimal drücken, um den Kommentar
- ➤ *Taste auf aussen* einzugeben.
- ➤ **Return-Taste** drücken,
- ➤ *U*
- ➤ Leerzeichen
- ➤ *E 1.4*
- ➤ **Doppelpfeiltaste rechts** zweimal drücken, um den Kommentar
- ➤ *Schluesselschalter* einzugeben.
- ➤ **Return-Taste** drücken,

und so weiter.

Für das Netzwerkende tippen Sie drei Sternchen.

Sichern Sie die bisherige Eingabe mit der Funktion

- ➤ **SICHERN** (F7).

SICHERN (F7) Mit dieser Funktion sichern Sie Ihre AWL-Quelldatei, ohne den Editor zu verlassen. Sie können also problemlos zwischenspeichern oder Ihre Editierung kurzzeitig unterbrechen.

Im Unterschied zum Paket KOP, FUP, AWL, wo Sie beim Speichern die Eingabe immer verlassen und zum Weiterarbeiten in die Ausgabe gehen müssen.

Bedienung am Programmiergerät

Absolute Operanden Nach dem Netzwerkkommentar folgen die einzelnen Anweisungen für das Öffnen des Garagentors, zunächst in absoluter Form mit Anweisungskommentar. Achten Sie darauf, daß Sie die Leerzeichen selbst eingeben müssen.

- ➤ *U(*
- ➤ **Return-Taste** drücken,
- ➤ *U*
- ➤ Leerzeichen
- ➤ *E 1.2*
- ➤ **Doppelpfeiltaste rechts** zweimal drücken, um den Kommentar
- ➤ *Taste auf aussen* einzugeben.
- ➤ **Return-Taste** drücken,
- ➤ *U*
- ➤ Leerzeichen
- ➤ *E 1.4*
- ➤ **Doppelpfeiltaste rechts** zweimal drücken, um den Kommentar
- ➤ *Schluesselschalter* einzugeben.
- ➤ **Return-Taste** drücken,

und so weiter.

Für das Netzwerkende tippen Sie drei Sternchen.

Sichern Sie die bisherige Eingabe mit der Funktion

- ➤ **SICHERN** (F7).

SICHERN (F7) Mit dieser Funktion sichern Sie Ihre AWL-Quelldatei, ohne den Editor zu verlassen. Sie können also problemlos zwischenspeichern oder Ihre Editierung kurzzeitig unterbrechen.

Im Unterschied zum Paket KOP, FUP, AWL, wo Sie beim Speichern die Eingabe immer verlassen und zum Weiterarbeiten in die Ausgabe gehen müssen.

Symbolische Operanden

Jetzt folgen die Anweisungen für das Schließen des Tores im Netzwerk 3, diesmal symbolisch programmiert.

Überschrift und Kommentar geben Sie wieder so ein, wie oben beschrieben. Bei den symbolischen Operanden gehen Sie so vor:

➤ *U*

➤ **Doppelpfeiltaste rechts** einmal drücken, der Cursor steht jetzt in der Spalte Operandensymbol.

➤ *T-ZU A* (ohne einleitenden Bindestrich!)

➤ **Return-Taste** drücken

➤ *U*

➤ **Doppelpfeiltaste rechts** einmal drücken, um das Symbol

➤ *Schluess* einzugeben,

➤ **Return-Taste** drücken.

Sind Sie am Bausteinende BE angekommen, speichern Sie Ihre Daten mit

➤ **SICHERN** (F7).

Im folgenden Abschnitt zeigen wir Ihnen, wie Sie mit den einzelnen Funktionen umgehen, um eine Anweisungsliste zu bearbeiten. Die Bearbeitung bzw. Korrektur einer Datei erfolgt wie im Symbolik-Editor ebenfalls in der Editierfunktion.

Bedienung am Programmiergerät

Symbolische Operanden

Jetzt folgen die Anweisungen für das Schließen des Tores im Netzwerk 3, diesmal symbolisch programmiert.

Überschrift und Kommentar geben Sie wieder so ein, wie oben beschrieben. Bei den symbolischen Operanden gehen Sie so vor:

➤ *U*

➤ **Doppelpfeiltaste rechts** einmal drücken, der Cursor steht jetzt in der Spalte Operandensymbol.

➤ *T-ZU A* (ohne einleitenden Bindestrich!)

➤ **Return-Taste** drücken

➤ *U*

➤ **Doppelpfeiltaste rechts** einmal drücken, um das Symbol

➤ *Schluess* einzugeben,

➤ **Return-Taste** drücken.

Sind Sie am Bausteinende BE angekommen, speichern Sie Ihre Daten mit

➤ **SICHERN** (F7).

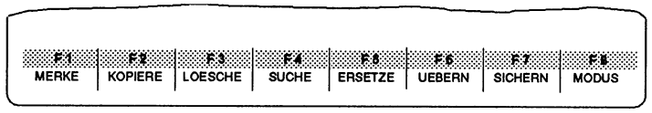
Im folgenden Abschnitt zeigen wir Ihnen, wie Sie mit den einzelnen Funktionen umgehen, um eine Anweisungsliste zu bearbeiten. Die Bearbeitung bzw. Korrektur einer Datei erfolgt wie im Symbolik-Editor ebenfalls in der Editierfunktion.

3

C79000-B8500-C677-02

3 - 29

3.3.5 Benutzen der Softkey-Leiste EDITIEREN



Bei den Erläuterungen zur Bedienung der Editierfunktion wollen wir insbesondere das Zusammenspiel von **MERKE, KOPIERE** und **LOESCHE** aufzeigen und vorführen, wie Sie Teile Ihres Programms zur Weiter- und Wiederverarbeitung auf Dateien abspeichern und Funktionstasten mit Zeichenfolgen belegen können.

Ein hilfreiches Werkzeug ist der Wiederholungsfaktor: Nach Aufrufen einer Funktion geben Sie über die Schreibmaschinentastatur Ihres Programmiergeräts eine Zahl ein. Mit diesem Faktor wird die nachfolgende Funktion ausgeführt, z.B. eine Zeile 7 mal kopieren.

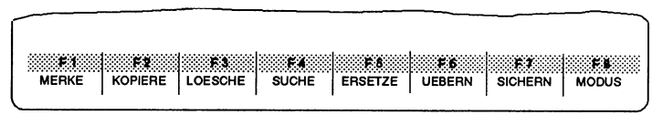
Welche Funktion Sie aktiviert haben, steht in der Kopfzeile über dem ANWEISUNGSKOMMENTAR. Eine Funktion muß immer mit **ZURUECK** (F8) verlassen werden, bevor Sie im Text weiterarbeiten können.

Wollen Sie innerhalb einer Funktion einen Vorgang abbrechen, benutzen Sie dazu die **Abbruchtaste** und/oder die Taste F8 **ZURUECK**.

> **Vorsicht**
> Arbeiten Sie mit der Abbruchtaste, verlieren Sie möglicherweise Daten! Haben Sie z.B. eine Datei korrigiert und drücken Sie dann die Abbruchtaste, sind alle Veränderungen verworfen.

Bedienung am Programmiergerät

3.3.5 Benutzen der Softkey-Leiste EDITIEREN



Bei den Erläuterungen zur Bedienung der Editierfunktion wollen wir insbesondere das Zusammenspiel von **MERKE, KOPIERE** und **LOESCHE** aufzeigen und vorführen, wie Sie Teile Ihres Programms zur Weiter- und Wiederverarbeitung auf Dateien abspeichern und Funktionstasten mit Zeichenfolgen belegen können.

Ein hilfreiches Werkzeug ist der Wiederholungsfaktor: Nach Aufrufen einer Funktion geben Sie über die Schreibmaschinentastatur Ihres Programmiergeräts eine Zahl ein. Mit diesem Faktor wird die nachfolgende Funktion ausgeführt, z.B. eine Zeile 7 mal kopieren.

Welche Funktion Sie aktiviert haben, steht in der Kopfzeile über dem ANWEISUNGSKOMMENTAR. Eine Funktion muß immer mit **ZURUECK** (F8) verlassen werden, bevor Sie im Text weiterarbeiten können.

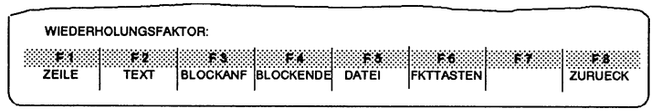
Wollen Sie innerhalb einer Funktion einen Vorgang abbrechen, benutzen Sie dazu die **Abbruchtaste** und/oder die Taste F8 **ZURUECK**.

> **Vorsicht**
> Arbeiten Sie mit der Abbruchtaste, verlieren Sie möglicherweise Daten! Haben Sie z.B. eine Datei korrigiert und drücken Sie dann die Abbruchtaste, sind alle Veränderungen verworfen.

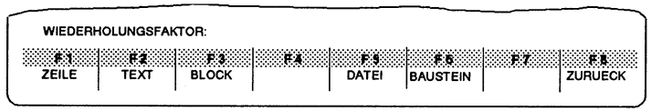
Die Funktionen MERKE, KOPIERE und LOESCHE

MERKE Über diese Funktion können Textteile gemerkt werden. Beliebige Zeichenfolgen (bis maximal 40 Zeichen), Einzelzeilen und Zeichenblöcke (maximal 500 Zeilen) werden jeweils in einen eigenen Puffer geschrieben und können an beliebigen Stellen kopiert werden. Der im Puffer gemerkte Text kann auch in eine sequentielle Datei abgespeichert werden.

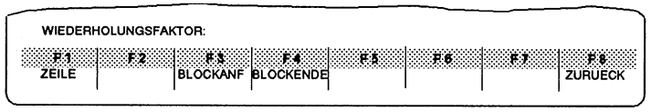
Funktionstasten werden ebenfalls über die Merkfunktion mit Zeichenfolgen belegt (maximal 40 Zeichen).



KOPIERE Gemerkter Text aus dem Puffer oder aus sequentiellen Dateien (Merk-, Quelldatei) wird mit dieser Funktion an der Cursorposition eingefügt. Sie können auch eine gesamte AWL-Quelldatei oder einen Baustein daraus an die Cursorposition kopieren.



LOESCHE Das Löschmenü bietet Ihnen an, Einzelzeilen und markierte Textblöcke zu löschen. Der gelöschte Text wird sicherheitshalber in den Puffer geschrieben. Damit geht vorher "gemerkter" Text verloren.

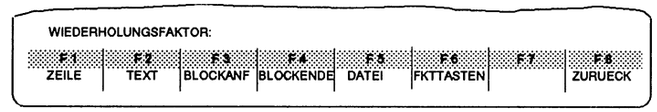


Bedienung am Programmiergerät

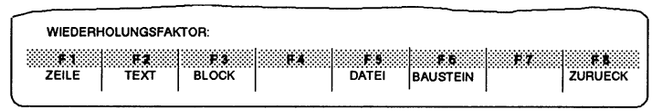
Die Funktionen MERKE, KOPIERE und LOESCHE

MERKE Über diese Funktion können Textteile gemerkt werden. Beliebige Zeichenfolgen (bis maximal 40 Zeichen), Einzelzeilen und Zeichenblöcke (maximal 500 Zeilen) werden jeweils in einen eigenen Puffer geschrieben und können an beliebigen Stellen kopiert werden. Der im Puffer gemerkte Text kann auch in eine sequentielle Datei abgespeichert werden.

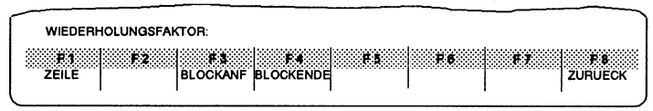
Funktionstasten werden ebenfalls über die Merkfunktion mit Zeichenfolgen belegt (maximal 40 Zeichen).



KOPIERE Gemerkter Text aus dem Puffer oder aus sequentiellen Dateien (Merk-, Quelldatei) wird mit dieser Funktion an der Cursorposition eingefügt. Sie können auch eine gesamte AWL-Quelldatei oder einen Baustein daraus an die Cursorposition kopieren.



LOESCHE Das Löschmenü bietet Ihnen an, Einzelzeilen und markierte Textblöcke zu löschen. Der gelöschte Text wird sicherheitshalber in den Puffer geschrieben. Damit geht vorher "gemerkter" Text verloren.



Bearbeiten einer Zeile

z.B. soll die Überschrift des ersten Netzwerks mit der des zweiten identisch sein.

1. Merken

- ➤ Cursor auf diese Zeile stellen,
- ➤ **MERKE** (F1) aufrufen,
- ➤ **ZEILE** (F1) drücken, diese Zeile steht jetzt im Puffer,
- ➤ mit **ZURUECK** (F8) die Merkfunktion verlassen.

2. Kopieren

- ➤ Den Cursor beim zweiten Netzwerk auf dessen alte Überschrift stellen,
- ➤ **KOPIERE** (F2) aufrufen (Schreibmodus beachten! s.u. Hinweis),
- ➤ **ZEILE** (F1) drücken, die neue Überschrift wird eingefügt,
- ➤ mit **ZURUECK** (F8) die Kopierfunktion verlassen.

3. Löschen

- ➤ Den Cursor auf die alte Überschrift stellen,
- ➤ **LOESCHE** (F3) aufrufen,
- ➤ **ZEILE** (F1) drücken, die alte Überschrift ist gelöscht,
- ➤ mit **ZURUECK** (F8) die Löschfunktion verlassen.

4. Verschieben

- ➤ Den Cursor auf die Zeile stellen, die verschoben werden soll,
- ➤ **LOESCHE** (F3) aufrufen,
- ➤ **ZEILE** (F1) drücken, der Text dieser Zeile steht nun im Puffer.
- ➤ Den Cursor an die Stelle positionieren, an die geschoben werden soll,
- ➤ **KOPIERE** (F2) aufrufen,
- ➤ **ZEILE** (F1) drücken, die Zeile wird eingefügt.
- ➤ Mit **ZURUECK** (F8) die Kopierfunktion beenden.

Bedienung am Programmiergerät

Bearbeiten einer Zeile

z.B. soll die Überschrift des ersten Netzwerks mit der des zweiten identisch sein.

1. Merken

- ➤ Cursor auf diese Zeile stellen,
- ➤ **MERKE** (F1) aufrufen,
- ➤ **ZEILE** (F1) drücken, diese Zeile steht jetzt im Puffer,
- ➤ mit **ZURUECK** (F8) die Merkfunktion verlassen.

2. Kopieren

- ➤ Den Cursor beim zweiten Netzwerk auf dessen alte Überschrift stellen,
- ➤ **KOPIERE** (F2) aufrufen (Schreibmodus beachten! s.u. Hinweis),
- ➤ **ZEILE** (F1) drücken, die neue Überschrift wird eingefügt,
- ➤ mit **ZURUECK** (F8) die Kopierfunktion verlassen.

3. Löschen

- ➤ Den Cursor auf die alte Überschrift stellen,
- ➤ **LOESCHE** (F3) aufrufen,
- ➤ **ZEILE** (F1) drücken, die alte Überschrift ist gelöscht,
- ➤ mit **ZURUECK** (F8) die Löschfunktion verlassen.

4. Verschieben

- ➤ Den Cursor auf die Zeile stellen, die verschoben werden soll,
- ➤ **LOESCHE** (F3) aufrufen,
- ➤ **ZEILE** (F1) drücken, der Text dieser Zeile steht nun im Puffer.
- ➤ Den Cursor an die Stelle positionieren, an die geschoben werden soll,
- ➤ **KOPIERE** (F2) aufrufen,
- ➤ **ZEILE** (F1) drücken, die Zeile wird eingefügt.
- ➤ Mit **ZURUECK** (F8) die Kopierfunktion beenden.

3 - 32

C79000-B8500-C677-02

Hinweis
Mit MERKE werden eine Zeile, ein Text und ein Block in jeweils einem eigenen Puffer abgelegt . Im Puffer wird die gemerkte Zeile nur von einer neu gemerkten Zeile überschrieben, ein Text nur von einem neu gemerkten Text und ein Block nur von einem neu gemerkten Block. So haben Sie die Möglichkeit, drei Zeichenfolgen parallel zu merken.
Beim Kopieren den Cursor vor dem Aufruf der **KOPIERE**-Funktion positionieren, weil die Funktionen sofort an der Cursorposition durchgeführt werden. Beachten Sie auch den Schreibmodus: im Einfügemodus wird gemerkter Text eingefügt, im Überschreibmodus wird der Text vom Cursor ab mit dem gemerkten Text überschrieben. Schaffen Sie also Platz, wenn Sie im Überschreibmodus einfügen wollen.

Bearbeiten eines Blocks (max. 500 Zeilen)

1. Merken und kopieren z.B. ein Netzwerkkommentar

- ➤ **MERKE** (F1) aufrufen,
- ➤ den Cursor an den Anfang eines Netzwerkkommentars stellen,
- ➤ mit **BLOCKANF**ang (F3) markieren,
- ➤ den Cursor ans Ende des Kommentars stellen,
- ➤ dieses mit **BLOCKENDE** (F4) markieren,
- ➤ mit **ZURUECK** (F8) die Merkfunktion verlassen.
- ➤ **KOPIERE** (F2) aufrufen,
- ➤ den Cursor z.B. ans Ende der Datei positionieren,
- ➤ **BLOCK** (F3) drücken, der Kommentar wird hier eingefügt,
- ➤ mit **ZURUECK** (F8) die Kopierfunktion verlassen.

2. Merken auf Datei z.B. die Einschaltverzögerung in die Datei TIMER@ speichern.

- ➤ **MERKE** (F1) aufrufen,
- ➤ Cursor auf die Zeile L KT 005.2 stellen,
- ➤ **BLOCKANF**ang (F3) drücken,

Bedienung am Programmiergerät

Hinweis
Mit MERKE werden eine Zeile, ein Text und ein Block in jeweils einem eigenen Puffer abgelegt . Im Puffer wird die gemerkte Zeile nur von einer neu gemerkten Zeile überschrieben, ein Text nur von einem neu gemerkten Text und ein Block nur von einem neu gemerkten Block. So haben Sie die Möglichkeit, drei Zeichenfolgen parallel zu merken.
Beim Kopieren den Cursor vor dem Aufruf der **KOPIERE**-Funktion positionieren, weil die Funktionen sofort an der Cursorposition durchgeführt werden. Beachten Sie auch den Schreibmodus: im Einfügemodus wird gemerkter Text eingefügt, im Überschreibmodus wird der Text vom Cursor ab mit dem gemerkten Text überschrieben. Schaffen Sie also Platz, wenn Sie im Überschreibmodus einfügen wollen.

3

Bearbeiten eines Blocks (max. 500 Zeilen)

1. Merken und kopieren z.B. ein Netzwerkkommentar

- ➤ **MERKE** (F1) aufrufen,
- ➤ den Cursor an den Anfang eines Netzwerkkommentars stellen,
- ➤ mit **BLOCKANF**ang (F3) markieren,
- ➤ den Cursor ans Ende des Kommentars stellen,
- ➤ dieses mit **BLOCKENDE** (F4) markieren,
- ➤ mit **ZURUECK** (F8) die Merkfunktion verlassen.
- ➤ **KOPIERE** (F2) aufrufen,
- ➤ den Cursor z.B. ans Ende der Datei positionieren,
- ➤ **BLOCK** (F3) drücken, der Kommentar wird hier eingefügt,
- ➤ mit **ZURUECK** (F8) die Kopierfunktion verlassen.

2. Merken auf Datei z.B. die Einschaltverzögerung in die Datei TIMER@ speichern.

- ➤ **MERKE** (F1) aufrufen,
- ➤ Cursor auf die Zeile L KT 005.2 stellen,
- ➤ **BLOCKANF**ang (F3) drücken,

C79000-B8500-C677-02 3 - 33

➤ Cursor nach unten führen bis zum Ende des Timers, Zeile EIN-VZOE,

➤ **BLOCKENDE** (F4) drücken, dieser Block ist jetzt im Puffer gemerkt.

➤ **DATEI** (F5) aufrufen und

➤ mit *TIMER@* ausfüllen,

➤ **Return-Taste** drücken und

➤ **UEBERN**ehmen (F6). Die Einschaltverzögerung liegt jetzt in der Datei TIMER@A0.SEQ

➤ Mit **ZURUECK** (F8) die Merkfunktion verlassen.

➤ Den Cursor hinter das Bausteinende (BE) stellen,

➤ **KOPIERE** (F2) aufrufen,

➤ **DATEI** (F5) aufrufen und die Datei eingeben, die kopiert werden soll. Wir übernehmen die eingeblendete Datei B:TIMER@A0.SEQ,

➤ mit **UEBERN** (F6) oder mit der **Übernahmetaste** den Kopiervorgang auslösen.

Die Einschaltverzögerung wird eingefügt.

➤ Mit **ZUERUECK** (F8) die Kopierfunktion verlassen.

3. Löschen Die eingefügten Teile sollen gelöscht werden.

➤ **LOESCHE** (F3) aufrufen. Die Einschaltverzögerung

➤ mit **BLOCKANF**ang (F3) und

➤ **BLOCKENDE** (F4) markieren, der Text ist gelöscht.

➤ **ZURUECK** gehen (F8).

Den Netzwerkkommentar können Sie z.B. auch zeilenweise mit Wiederholungsfaktor löschen.

➤ Den Cursor positionieren,

➤ **LOESCHE** (F3) aufrufen,

➤ über die Schreibmaschinentastatur einen Faktor (z.B. 4) eingeben

➤ **ZEILE** (F1) drücken,

Die (vier) Zeilen sind gelöscht.

➤ **ZURUECK** gehen (F8).

Bedienung am Programmiergerät

➤ Cursor nach unten führen bis zum Ende des Timers, Zeile EIN-VZOE,

➤ **BLOCKENDE** (F4) drücken, dieser Block ist jetzt im Puffer gemerkt.

➤ **DATEI** (F5) aufrufen und

➤ mit *TIMER@* ausfüllen,

➤ **Return-Taste** drücken und

➤ **UEBERN**ehmen (F6). Die Einschaltverzögerung liegt jetzt in der Datei TIMER@A0.SEQ

➤ Mit **ZURUECK** (F8) die Merkfunktion verlassen.

➤ Den Cursor hinter das Bausteinende (BE) stellen,

➤ **KOPIERE** (F2) aufrufen,

➤ **DATEI** (F5) aufrufen und die Datei eingeben, die kopiert werden soll. Wir übernehmen die eingeblendete Datei B:TIMER@A0.SEQ,

➤ mit **UEBERN** (F6) oder mit der **Übernahmetaste** den Kopiervorgang auslösen.

Die Einschaltverzögerung wird eingefügt.

➤ Mit **ZUERUECK** (F8) die Kopierfunktion verlassen.

3. Löschen Die eingefügten Teile sollen gelöscht werden.

➤ **LOESCHE** (F3) aufrufen. Die Einschaltverzögerung

➤ mit **BLOCKANF**ang (F3) und

➤ **BLOCKENDE** (F4) markieren, der Text ist gelöscht.

➤ **ZURUECK** gehen (F8).

Den Netzwerkkommentar können Sie z.B. auch zeilenweise mit Wiederholungsfaktor löschen.

➤ Den Cursor positionieren,

➤ **LOESCHE** (F3) aufrufen,

➤ über die Schreibmaschinentastatur einen Faktor (z.B. 4) eingeben

➤ **ZEILE** (F1) drücken,

Die (vier) Zeilen sind gelöscht.

➤ **ZURUECK** gehen (F8).

3 - 34

C79000-B8500-C677-02

4. Verschieben Den Block, der verschoben werden soll, mit

- ➤ **LOESCHE** (F3),
- ➤ **BLOCKANF**ang (F3) und
- ➤ **BLOCKENDE** (F4) löschen und in den Puffer schreiben.
- ➤ **ZURUECK** (F8) gehen.
- ➤ Den Cursor auf die neue Position stellen,
- ➤ **KOPIERE** (F2) aufrufen und
- ➤ **BLOCK** (F3) drücken. Der Text ist verschoben.
- ➤ **ZURUECK** (F8) gehen.

Bearbeiten einer Zeichenfolge oder eines Textes (max. 40 Zeichen)

z.B. wollen Sie (nachträglich) an mehreren Stellen einen Text oder eine Anweisung einfügen oder ändern, oder Sie haben einen ständig wiederkehrenden Anweisungskommentar, den Sie nicht immer neu schreiben wollen. Solche Texte können Sie im Puffer ablegen.

1. Merken

- ➤ **MERKE** (F1) aufrufen,
- ➤ **TEXT** (F2) drücken,
- ➤ zwischen den Doppelpunkten Text eingeben, z.B. *U E 2.0.*
- ➤ **UEBERN** (F6) drücken. Der eingegebene Text befindet sich nun im Puffer.
- ➤ Mit **ZURUECK** die Merkfunktion verlassen.

Text zwischen Doppelpunkten können Sie mit der Taste **+1** löschen.

2. Kopieren

- ➤ Den Cursor positionieren,
- ➤ **KOPIERE** (F2) aufrufen (Schreibmodus beachten! s.o. Hinweis),
- ➤ Cursor an die Stelle rücken, die bearbeitet werden soll,
- ➤ **TEXT** (F2) drücken und der gemerkte Text wird eingefügt.
- ➤ Mit **ZURUECK** (F8) verlassen Sie die Kopierfunktion.

Bedienung am Programmiergerät

4. Verschieben Den Block, der verschoben werden soll, mit

- ➤ **LOESCHE** (F3),
- ➤ **BLOCKANF**ang (F3) und
- ➤ **BLOCKENDE** (F4) löschen und in den Puffer schreiben.
- ➤ **ZURUECK** (F8) gehen.
- ➤ Den Cursor auf die neue Position stellen,
- ➤ **KOPIERE** (F2) aufrufen und
- ➤ **BLOCK** (F3) drücken. Der Text ist verschoben.
- ➤ **ZURUECK** (F8) gehen.

Bearbeiten einer Zeichenfolge oder eines Textes (max. 40 Zeichen)

z.B. wollen Sie (nachträglich) an mehreren Stellen einen Text oder eine Anweisung einfügen oder ändern, oder Sie haben einen ständig wiederkehrenden Anweisungskommentar, den Sie nicht immer neu schreiben wollen. Solche Texte können Sie im Puffer ablegen.

3

1. Merken

- ➤ **MERKE** (F1) aufrufen,
- ➤ **TEXT** (F2) drücken,
- ➤ zwischen den Doppelpunkten Text eingeben, z.B. *U E 2.0.*
- ➤ **UEBERN** (F6) drücken. Der eingegebene Text befindet sich nun im Puffer.
- ➤ Mit **ZURUECK** die Merkfunktion verlassen.

Text zwischen Doppelpunkten können Sie mit der Taste **+1** löschen.

2. Kopieren

- ➤ Den Cursor positionieren,
- ➤ **KOPIERE** (F2) aufrufen (Schreibmodus beachten! s.o. Hinweis),
- ➤ Cursor an die Stelle rücken, die bearbeitet werden soll,
- ➤ **TEXT** (F2) drücken und der gemerkte Text wird eingefügt.
- ➤ Mit **ZURUECK** (F8) verlassen Sie die Kopierfunktion.

C79000-B8500-C677-02 3 - 35

Belegung von Funktionstasten (mit max. 40 Zeichen)

z.B. soll das Netzwerkende als häufig auftretende Zeichenfolge auf eine Funktionstaste gelegt werden.

➤ **MERKE** (F1) aufrufen,

➤ mit **FKTTASTEN** (F6) die Liste der Funktionstasten auf den Bildschirm ausgeben. Der Cursor blinkt in der Zeile f1.

➤ Zwischen den Doppelpunkten *** eingeben,

➤ mit F6 **UEBERN**ehmen.

Diese Zeichenfolge erhalten Sie an jeder Stelle in Ihrer Datei, wenn Sie bei den Geräten PG 695, PG 695II und PG 750 die Tasten Shift und F1 drücken, beim PG 685 die Taste "Oeffner".

Fügen Sie z.B. nach einem Netzwerkende ein weiteres ein:

➤ Den Cursor auf *** stellen,

➤ die Tasten **SHIFT** und **F1** drücken.

Verwerfen Sie nun alle diese Änderungen:

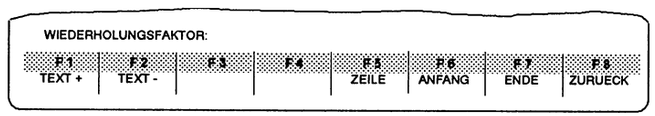
➤ **Abbruchtaste** drücken und

➤ mit der **Übernahmetaste** bestätigen. In der ausgegebenen FUNKTIONSANWAHL

➤ erneut **EDITIEREN** (F1) aufrufen.

Die Funktion SUCHE

Über **SUCHE** können Sie an den Anfang und ans Ende Ihrer Datei und zu Einzelzeilen springen. Die Zeile 0 ist nicht zulässig. Weiterhin können Sie beliebige Zeichenfolgen mit maximal 20 Zeichen (Wörter und Zahlen) suchen, die innerhalb eines Feldes stehen. Text + sucht solche Zeichenfolgen ab der Cursorposition vorwärts, Textsucht rückwärts. Mit **SUCHE** bewegen Sie sich also gezielt und bequem innerhalb Ihrer Datei.



Bedienung am Programmiergerät

Belegung von Funktionstasten (mit max. 40 Zeichen)

z.B. soll das Netzwerkende als häufig auftretende Zeichenfolge auf eine Funktionstaste gelegt werden.

➤ **MERKE** (F1) aufrufen,

➤ mit **FKTTASTEN** (F6) die Liste der Funktionstasten auf den Bildschirm ausgeben. Der Cursor blinkt in der Zeile f1.

➤ Zwischen den Doppelpunkten *** eingeben,

➤ mit F6 **UEBERN**ehmen.

Diese Zeichenfolge erhalten Sie an jeder Stelle in Ihrer Datei, wenn Sie bei den Geräten PG 695, PG 695II und PG 750 die Tasten Shift und F1 drücken, beim PG 685 die Taste "Oeffner".

Fügen Sie z.B. nach einem Netzwerkende ein weiteres ein:

➤ Den Cursor auf *** stellen,

➤ die Tasten **SHIFT** und **F1** drücken.

Verwerfen Sie nun alle diese Änderungen:

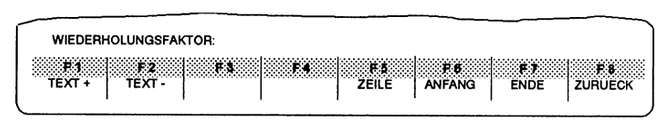
➤ **Abbruchtaste** drücken und

➤ mit der **Übernahmetaste** bestätigen. In der ausgegebenen FUNKTIONSANWAHL

➤ erneut **EDITIEREN** (F1) aufrufen.

Die Funktion SUCHE

Über **SUCHE** können Sie an den Anfang und ans Ende Ihrer Datei und zu Einzelzeilen springen. Die Zeile 0 ist nicht zulässig. Weiterhin können Sie beliebige Zeichenfolgen mit maximal 20 Zeichen (Wörter und Zahlen) suchen, die innerhalb eines Feldes stehen. Text + sucht solche Zeichenfolgen ab der Cursorposition vorwärts, Textsucht rückwärts. Mit **SUCHE** bewegen Sie sich also gezielt und bequem innerhalb Ihrer Datei.



3 - 36

C79000-B8500-C677-02

Zu beachten ist, daß der zu suchende Text exakt die Groß-/Kleinschreibung der gesuchten Zeichenfolge aufweisen muß. Texte, die in den Feldern "ADR" und "Anweisung" stehen, müssen Sie daher in Großbuchstaben eingeben!

z.B. springen Sie ans Ende mit

➤ **ENDE** (F7), der Cursor steht unterhalb des BE;

an den Anfang mit

➤ **ANFANG** (F6), der Cursor steht auf der AG-Typangabe (soweit vorhanden).

in eine neue Zeile mit

➤ **ZEILE** (F5),

➤ Sie gegen die Nummer ein , z.B. *12*,

➤ und **UEBERN**ehmen. Der Cursor blinkt in Zeile 12.

Einen bestimmten Begriff, eine Adresse, einen Operanden suchen Sie von Ihrer Cursorposition aus vorwärts mit

➤ **TEXT+** (F1)

➤ Sie geben die Zeichenfolge ein, z.B. *UN E1.0,* und

➤ **UEBERN**ehmen (F6). Der Cursor springt ans Ende der gesuchten Zeichenfolge, die nach der aktuellen Cursorposition im Text als nächste gefunden wird.

von Ihrer Position aus rückwärts mit

➤ **TEXT-** (F2)

➤ Sie geben auch hier die Zeichenfolge ein, z.B. *TASTE AUF*

➤ und **UEBERN**ehmen (F6).

Der Cursor springt nun zum letzten Zeichen der gesuchten Zeichenfolge, die vor der aktuellen Cursorposition im Text als nächste gefunden wird.

Bedienung am Programmiergerät

Zu beachten ist, daß der zu suchende Text exakt die Groß-/Kleinschreibung der gesuchten Zeichenfolge aufweisen muß. Texte, die in den Feldern "ADR" und "Anweisung" stehen, müssen Sie daher in Großbuchstaben eingeben!

z.B. springen Sie ans Ende mit

➤ **ENDE** (F7), der Cursor steht unterhalb des BE;

an den Anfang mit

➤ **ANFANG** (F6), der Cursor steht auf der AG-Typangabe (soweit vorhanden).

in eine neue Zeile mit

➤ **ZEILE** (F5),

➤ Sie gegen die Nummer ein , z.B. *12*,

➤ und **UEBERN**ehmen. Der Cursor blinkt in Zeile 12.

Einen bestimmten Begriff, eine Adresse, einen Operanden suchen Sie von Ihrer Cursorposition aus vorwärts mit

➤ **TEXT+** (F1)

➤ Sie geben die Zeichenfolge ein, z.B. *UN E1.0,* und

➤ **UEBERN**ehmen (F6). Der Cursor springt ans Ende der gesuchten Zeichenfolge, die nach der aktuellen Cursorposition im Text als nächste gefunden wird.

von Ihrer Position aus rückwärts mit

➤ **TEXT-** (F2)

➤ Sie geben auch hier die Zeichenfolge ein, z.B. *TASTE AUF*

➤ und **UEBERN**ehmen (F6).

Der Cursor springt nun zum letzten Zeichen der gesuchten Zeichenfolge, die vor der aktuellen Cursorposition im Text als nächste gefunden wird.

3

C79000-B8500-C677-02

3 - 37

Wollen Sie das Suchen nach dem eingegebenen Begriff fortsetzen, reaktivieren Sie die gewählte Funktion:

- ➤ z.B. nochmals **TEXT-** aufrufen und
- ➤ **UEBERN**ehmen, der Cursor springt wiederum zu der ihm am nächsten stehenden Zeichenfolge des eingegebenen Begriffs in der gewünschten Richtung.

Am Ende des Suchvorgangs schließen Sie mit

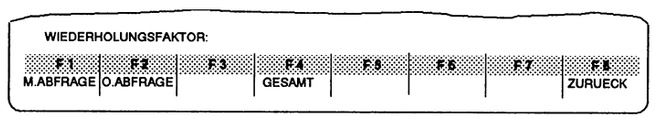
- ➤ **ZURUECK** (F8) ab.

Die Funktion ERSETZE

Beliebige, maximal 20 Zeichen lange Zeichenfolgen (Wörter und Zahlen) in den Spalten ADR, OPERANDENSYMBOL und ANWEISUNGSKOMMENTAR können durch andere ersetzt werden. Sie wählen dabei zwischen Einzelaustausch mit oder ohne kontrollierender Abfrage und einem gesamten Ersetzen. Einzelnes Ersetzen erfolgt nur von der Cursorposition abwärts, positionieren Sie also Ihren Cursor für diese Funktion um mindestens eine Zeile höher.

Groß-/Kleinschreibung

Zu beachten ist, daß der zu suchende Text exakt die Groß-/Kleinschreibung der gesuchten Zeichenfolge aufweisen muß. Texte, die in den Feldern "ADR" und "Anweisung" stehen, müssen Sie daher in Großbuchstaben eingeben!



ERSETZE bietet Ihnen die Möglichkeit, Ihre Datei schnell zu korrigieren z.B. soll in der gesamten Datei eine Anweisung oder ein Symbol ausgetauscht werden. Achten Sie dabei auf die absolute Identität der gesuchten Zeichenfolge. Auch Leerzeichen sind wichtig.

- ➤ **ERSETZE** (F5) aufrufen,
- ➤ **GESAMT** (F4) drücken,
- ➤ die auszutauschende, alte Zeichenfolge eingeben, z.B. *EINVZOE*
- ➤ **UEBERN**ehmen (F6) drücken,

Bedienung am Programmiergerät

Wollen Sie das Suchen nach dem eingegebenen Begriff fortsetzen, reaktivieren Sie die gewählte Funktion:

- ➤ z.B. nochmals **TEXT-** aufrufen und
- ➤ **UEBERN**ehmen, der Cursor springt wiederum zu der ihm am nächsten stehenden Zeichenfolge des eingegebenen Begriffs in der gewünschten Richtung.

Am Ende des Suchvorgangs schließen Sie mit

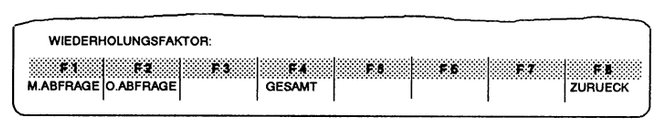
- ➤ **ZURUECK** (F8) ab.

Die Funktion ERSETZE

Beliebige, maximal 20 Zeichen lange Zeichenfolgen (Wörter und Zahlen) in den Spalten ADR, OPERANDENSYMBOL und ANWEISUNGSKOMMENTAR können durch andere ersetzt werden. Sie wählen dabei zwischen Einzelaustausch mit oder ohne kontrollierender Abfrage und einem gesamten Ersetzen. Einzelnes Ersetzen erfolgt nur von der Cursorposition abwärts, positionieren Sie also Ihren Cursor für diese Funktion um mindestens eine Zeile höher.

Groß-/Kleinschreibung

Zu beachten ist, daß der zu suchende Text exakt die Groß-/Kleinschreibung der gesuchten Zeichenfolge aufweisen muß. Texte, die in den Feldern "ADR" und "Anweisung" stehen, müssen Sie daher in Großbuchstaben eingeben!



ERSETZE bietet Ihnen die Möglichkeit, Ihre Datei schnell zu korrigieren z.B. soll in der gesamten Datei eine Anweisung oder ein Symbol ausgetauscht werden. Achten Sie dabei auf die absolute Identität der gesuchten Zeichenfolge. Auch Leerzeichen sind wichtig.

- ➤ **ERSETZE** (F5) aufrufen,
- ➤ **GESAMT** (F4) drücken,
- ➤ die auszutauschende, alte Zeichenfolge eingeben, z.B. *EINVZOE*
- ➤ **UEBERN**ehmen (F6) drücken,

➤ die neue Zeichenfolge eingeben, z.B. *SS-TIMER*

➤ und **UEBERN**ehmen (F6). Jetzt werden alle Symbole EIN-VZOE gegen SS-TIMER ausgetauscht.

Zeichenfolge tauschen

Soll eine Zeichenfolge nur einmal getauscht werden, haben Sie eine Kontrollmöglichkeit, wenn Sie **M**(it).**ABFRAGE** wählen; **O**(hne).**ABFRAGE** führt diesen Tausch sofort durch. Der Cursor muß hier oberhalb der zu tauschenden Zeichenfolge stehen, weil beim Einzeltausch immer vom Cursor abwärts ersetzt wird.

➤ **ERSETZE** (F5) aufrufen,

➤ **M.ABFRAGE** (F1) drücken,

➤ die auszutauschende, alte Zeichenfolge eingeben, z.B. *AUSSEN*

➤ **UEBERN**ehmen (F6) drücken,

➤ die neue Zeichenfolge eingeben, z.B. *ZZZZZZ*,

➤ und **UEBERN**ehmen. Sie haben jetzt die Kontrollmöglichkeit **JA** (F1) oder **NEIN** (F3),

➤ drücken Sie **JA** (F1). Die dem Cursor nächstfolgende Zeichenfolge "AUSSEN" wird durch "ZZZZZZ" ersetzt.

Eventuell erhalten Sie in der Meldezeile die Angabe "Nicht gefunden". Dann war im Text nach der aktuellen Cursorposition kein "AUSSEN" mehr zu finden. Gehen Sie mit dem Cursor an den Anfang der Datei und wiederholen Sie den Vorgang mit **ERSETZE** (F5), **M.ABFRAGE** (F1), zweimal **Übernahmetaste** und **JA** (F1). **O.ABFRAGE** funktioniert analog dazu, mit dem Unterschied, daß Sie keine Kontrolle haben.

Wiederholungsfaktor

Sie können den Einzeltausch mit dem Wiederholungsfaktor kombinieren. Grundsätzlich werden auch die Zeichenfolgen in Kommentaren mit angesprungen. Bei Angabe eines Wiederholungsfaktors werden sie vom Programm mitgezählt! Sie werden jedoch grundsätzlich nicht ersetzt, auch dann nicht, wenn Sie **M.ABFRAGE, JA** aufrufen. Mit der **Abbruchtaste** beenden Sie den Austausch vorzeitig, die bereits ausgetauschten Zeichenfolgen bleiben erhalten.

Bedienung am Programmiergerät

➤ die neue Zeichenfolge eingeben, z.B. *SS-TIMER*

➤ und **UEBERN**ehmen (F6). Jetzt werden alle Symbole EIN-VZOE gegen SS-TIMER ausgetauscht.

Zeichenfolge tauschen

Soll eine Zeichenfolge nur einmal getauscht werden, haben Sie eine Kontrollmöglichkeit, wenn Sie **M**(it).**ABFRAGE** wählen; **O**(hne).**ABFRAGE** führt diesen Tausch sofort durch. Der Cursor muß hier oberhalb der zu tauschenden Zeichenfolge stehen, weil beim Einzeltausch immer vom Cursor abwärts ersetzt wird.

➤ **ERSETZE** (F5) aufrufen,

➤ **M.ABFRAGE** (F1) drücken,

➤ die auszutauschende, alte Zeichenfolge eingeben, z.B. *AUSSEN*

➤ **UEBERN**ehmen (F6) drücken,

➤ die neue Zeichenfolge eingeben, z.B. *ZZZZZZ*,

➤ und **UEBERN**ehmen. Sie haben jetzt die Kontrollmöglichkeit **JA** (F1) oder **NEIN** (F3),

➤ drücken Sie **JA** (F1). Die dem Cursor nächstfolgende Zeichenfolge "AUSSEN" wird durch "ZZZZZZ" ersetzt.

Eventuell erhalten Sie in der Meldezeile die Angabe "Nicht gefunden". Dann war im Text nach der aktuellen Cursorposition kein "AUSSEN" mehr zu finden. Gehen Sie mit dem Cursor an den Anfang der Datei und wiederholen Sie den Vorgang mit **ERSETZE** (F5), **M.ABFRAGE** (F1), zweimal **Übernahmetaste** und **JA** (F1). **O.ABFRAGE** funktioniert analog dazu, mit dem Unterschied, daß Sie keine Kontrolle haben.

Wiederholungsfaktor

Sie können den Einzeltausch mit dem Wiederholungsfaktor kombinieren. Grundsätzlich werden auch die Zeichenfolgen in Kommentaren mit angesprungen. Bei Angabe eines Wiederholungsfaktors werden sie vom Programm mitgezählt! Sie werden jedoch grundsätzlich nicht ersetzt, auch dann nicht, wenn Sie **M.ABFRAGE, JA** aufrufen. Mit der **Abbruchtaste** beenden Sie den Austausch vorzeitig, die bereits ausgetauschten Zeichenfolgen bleiben erhalten.

C79000-B8500-C677-02 3 - 39

➤ Verlassen Sie mit **ZURUECK** (F8) die **ERSETZE**-Funktion und verwerfen Sie alle Veränderungen mit der

➤ **Abbruchtaste,** quittieren Sie mit der

➤ **Übernahmetaste**. Sie sind wieder in der FUNKTIONSANWAHL.

Geben Sie nun die ursprüngliche Datei wieder aus:

➤ **EDITIEREN** (F1)

Die Speicherfunktionen SICHERN und UEBERN

Die Funktion **SICHERN** ist Ihnen schon bekannt. Mit ihr speichern Sie und können Ihre Sitzung unterbrechen und wiederaufnehmen, ohne die Datei neu auszugeben.

Die Funktion **UEBERN** (F6) oder die **Übernahmetaste** speichert die Datei ab, erzeugt automatisch die zugehörige Zwischendatei und beendet die Editiersitzung. Während der Übersetzung wird die erstellte Anweisungsliste geprüft. Bei mehr als einem Fehler wird eine Fehlerliste erstellt. Bei nur einem Fehler wird die fehlerhafte Stelle am Bildschirm angezeigt und Sie können anhand der Fehlermeldung in der Fußzeile korrigieren.

Zur Fehlerliste lesen Sie bitte im Kapitel 3.4.

➤ Mit **UEBERN** (F6) die Datei abspeichern: Sie wird übersetzt, das Editieren beendet. Die Maske FUNKTIONSANWAHL wird ausgegeben.

Bedienung am Programmiergerät

➤ Verlassen Sie mit **ZURUECK** (F8) die **ERSETZE**-Funktion und verwerfen Sie alle Veränderungen mit der

➤ **Abbruchtaste,** quittieren Sie mit der

➤ **Übernahmetaste**. Sie sind wieder in der FUNKTIONSANWAHL.

Geben Sie nun die ursprüngliche Datei wieder aus:

➤ **EDITIEREN** (F1)

Die Speicherfunktionen SICHERN und UEBERN

Die Funktion **SICHERN** ist Ihnen schon bekannt. Mit ihr speichern Sie und können Ihre Sitzung unterbrechen und wiederaufnehmen, ohne die Datei neu auszugeben.

Die Funktion **UEBERN** (F6) oder die **Übernahmetaste** speichert die Datei ab, erzeugt automatisch die zugehörige Zwischendatei und beendet die Editiersitzung. Während der Übersetzung wird die erstellte Anweisungsliste geprüft. Bei mehr als einem Fehler wird eine Fehlerliste erstellt. Bei nur einem Fehler wird die fehlerhafte Stelle am Bildschirm angezeigt und Sie können anhand der Fehlermeldung in der Fußzeile korrigieren.

Zur Fehlerliste lesen Sie bitte im Kapitel 3.4.

➤ Mit **UEBERN** (F6) die Datei abspeichern: Sie wird übersetzt, das Editieren beendet. Die Maske FUNKTIONSANWAHL wird ausgegeben.

3.3.6 Eingabe von Funktionsbausteinen

Beispiel Die Datei B:FBTESTA0.SEQ, abgedruckt auf der folgenden Seite, dient als Arbeitsbeispiel. Wieder ist es die Steuerung des Garagentors, aber diesmal als Funktionsbaustein programmiert, um Ihnen das unterschiedliche Editieren dieser Bausteintypen vorzuführen.

Hier soll der Bausteinaufruf symbolisch programmiert werden. Erstellen Sie bitte deshalb die folgende Zuordnungsliste in der Symbolikdatei TEST@@Z0.INI, damit

die Übersetzung funktioniert. Wie Sie mit dem Symbolik-Editor arbeiten, finden Sie in der STEP 5-Beschreibung, Handbuch Band 2, im Kapitel Symbolik-Editor.


```
SEQ. DATEI: TEST@@Z0.INI

E1.0            END-OBEN            ENDSCHALTER OBEN
E1.1            END-UNT             ENDSCHALTER UNTEN
E1.2            T-AUF A             TASTE AUF AUSSEN
E1.3            T-ZU A              TASTE ZU AUSSEN
E1.4            SCHLUESS            SCHLUESSELSCHALTER AUSSEN
E1.5            T-AUF I             TASTE AUF INNEN
E1.6            T-ZU I              TASTE ZU INNEN
E1.7            STOP                SICHERHEITSSCHALTER STOP
A1.0            MOT-AUF             MOTOR AUFWAERTS
A1.1            MOT-AB              MOTOR ABWAERTS
T1              EIN-VZOE            EINSCHALTVERZOEGERUNG, 5 SEK.
FB1             GARAGE              FB ZUR STEUERUNG EINES GARAGENTORS
```

Wollen Sie sich allgemein über Funktionsbausteine informieren, lesen Sie bitte in der STEP 5-Beschreibung, Handbuch Band 2, Ihres Programmiergeräts die Einleitung und den Abschnitt Funktionsbausteine nach.

Bedienung am Programmiergerät

3.3.6 Eingabe von Funktionsbausteinen

Beispiel Die Datei B:FBTESTA0.SEQ, abgedruckt auf der folgenden Seite, dient als Arbeitsbeispiel. Wieder ist es die Steuerung des Garagentors, aber diesmal als Funktionsbaustein programmiert, um Ihnen das unterschiedliche Editieren dieser Bausteintypen vorzuführen.

Hier soll der Bausteinaufruf symbolisch programmiert werden. Erstellen Sie bitte deshalb die folgende Zuordnungsliste in der Symbolikdatei TEST@@Z0.INI, damit

die Übersetzung funktioniert. Wie Sie mit dem Symbolik-Editor arbeiten, finden Sie in der STEP 5-Beschreibung, Handbuch Band 2, im Kapitel Symbolik-Editor.

3

SEQ. DATEI: TEST@@Z0.INI

E1.0 END-OBEN ENDSCHALTER OBEN
E1.1 END-UNT ENDSCHALTER UNTEN
E1.2 T-AUF A TASTE AUF AUSSEN
E1.3 T-ZU A TASTE ZU AUSSEN
E1.4 SCHLUESS SCHLUESSELSCHALTER AUSSEN
E1.5 T-AUF I TASTE AUF INNEN
E1.6 T-ZU I TASTE ZU INNEN
E1.7 STOP SICHERHEITSSCHALTER STOP
A1.0 MOT-AUF MOTOR AUFWAERTS
A1.1 MOT-AB MOTOR ABWAERTS
T1 EIN-VZOE EINSCHALTVERZOEGERUNG, 5 SEK.
FB1 GARAGE FB ZUR STEUERUNG EINES GARAGENTORS

Wollen Sie sich allgemein über Funktionsbausteine informieren, lesen Sie bitte in der STEP 5-Beschreibung, Handbuch Band 2, Ihres Programmiergeräts die Einleitung und den Abschnitt Funktionsbausteine nach.

C79000-B8500-C677-02

3 - 41

```
AWL-Quelle: B: FBTESTA0.SEQ

ADR  ANWEISUNG        OPERANDENSYMBOL          ANWEISUNGSKOMMENTAR
     #                GARAGE                   FB 1 FUER EIN GARAGENTOR
     #N  GARAGE
ENDO (E)                                       ENDSCHALTER OBEN
ENDU (E)                                       ENDSCHALTER UNTEN
TA-I (E)                                       TASTE AUF INNEN
TA-A (E)                                       TASTE AUF AUSSEN
TZ-I (E)                                       TASTE ZU INNEN
TZ-A (E)                                       TASTE ZU AUSSEN
SCHL (E)                                       SCHLUESSELSCHALTER
STOP (E)                                       SICHERHEITSSCHALTER STOP
MAUF (A)                                       MOTOR AUFWAERTS
MAB  (A)                                       MOTOR ABWAERTS
     #UB                                       OEFFNEN VON AUSSEN ODER INNEN
     UN       =STOP
     U(
     U        =TA-A
     U        =SCHL
     O        =TA-I
     )
     UN       =ENDO
     S        = MAUF
     ***
     #UB                                       SCHLIESSEN VON AUSSEN ODER INNEN
*RUECKSETZEN DES AUSGANGS MOTOR AUFWAERTS
     O        =ENDO
     O        =STOP
     RB       =MAUF
     ***
     #UB                                       SCHLIESSEN VON AUSSEN ODER INNEN
*HIER SCHLIESST SICH DAS TOR SOFORT
     U(
     U        =TZ-A
     U        =SCHL
     O        =TZ-I
     )
     UN       =ENDU
     S        =MAB
     ***
     #UB                                       SCHLIESSEN VON AUSSEN ODER INNEN
*RUECKSETZEN DES AUSGANGS MOTOR ABWAERTS
     O        =ENDU
     O        =STOP
     RB       =MAB
     BE
```


Bedienung am Programmiergerät


```
AWL-Quelle: B: FBTESTA0.SEQ

ADR  ANWEISUNG        OPERANDENSYMBOL          ANWEISUNGSKOMMENTAR
     #                GARAGE                   FB 1 FUER EIN GARAGENTOR
     #N  GARAGE
ENDO (E)                                       ENDSCHALTER OBEN
ENDU (E)                                       ENDSCHALTER UNTEN
TA-I (E)                                       TASTE AUF INNEN
TA-A (E)                                       TASTE AUF AUSSEN
TZ-I (E)                                       TASTE ZU INNEN
TZ-A (E)                                       TASTE ZU AUSSEN
SCHL (E)                                       SCHLUESSELSCHALTER
STOP (E)                                       SICHERHEITSSCHALTER STOP
MAUF (A)                                       MOTOR AUFWAERTS
MAB  (A)                                       MOTOR ABWAERTS
     #UB                                       OEFFNEN VON AUSSEN ODER INNEN
     UN       =STOP
     U(
     U        =TA-A
     U        =SCHL
     O        =TA-I
     )
     UN       =ENDO
     S        = MAUF
     ***
     #UB                                       SCHLIESSEN VON AUSSEN ODER INNEN
*RUECKSETZEN DES AUSGANGS MOTOR AUFWAERTS
     O        =ENDO
     O        =STOP
     RB       =MAUF
     ***
     #UB                                       SCHLIESSEN VON AUSSEN ODER INNEN
*HIER SCHLIESST SICH DAS TOR SOFORT
     U(
     U        =TZ-A
     U        =SCHL
     O        =TZ-I
     )
     UN       =ENDU
     S        =MAB
     ***
     #UB                                       SCHLIESSEN VON AUSSEN ODER INNEN
*RUECKSETZEN DES AUSGANGS MOTOR ABWAERTS
     O        =ENDU
     O        =STOP
     RB       =MAB
     BE
```

Eingabe

Voraussetzung: Das Paket AWL-Editor/Batch-Compiler ist geladen.

Steigen Sie neu in das Beispiel ein, füllen Sie die VOREINSTELLUNG mit den Dateinamen FBTEST für die AWL-Quelldatei und Zwischendatei, und TEST@@ für Programm- und Symbolik-Datei aus. *Übernehmen* Sie und rufen Sie die Editierfunktion auf.

Haben Sie das Paket AWL-Editor/Batch-Compiler nicht verlassen, dann wird die Maske FUNKTIONSANWAHL des Pakets ausgegeben.

Gehen Sie in die VOREINSTELLUNG und ändern Sie den Namen der AWL-Quelldatei in FBTEST.

VOREINSTELLUNG ändern

- ➤ VOREINstellung (F6) aufrufen; der Cursor blinkt im Feld AWL-QUELLDATEI.
- ➤ Den Cursor mit der **Einfachpfeiltaste rechts** auf den Dateinamen stellen und mit *FBTEST* überschreiben.
- ➤ Return-Taste drücken, auch die Zwischendatei hat jetzt den Namen FBTEST. Die übrigen Dateien bleiben unverändert.
- ➤ **UEBERN**ehmen (F6).
- ➤ Rufen Sie mit **EDITIEREN** (F1) die Editierfunktion auf.

Jetzt können Sie beginnen, Ihren Funktionsbaustein "Garage" zu programmieren.

Erläuterungen zum Editieren von Funktionsbausteinen: Für die Schreibkonventionen der Steuerzeichen und Operationen schlagen Sie bitte in den Abschnitten 3.3.2 und 3.3.3 nach. Beachten Sie beim Eingeben die Besonderheiten für Funktionsbausteine:

- Formaloperanden werden in der Spalte ADR und ANWEISUNG definiert. Diese Definition entspricht der Bezeichnerliste in Funktionsbausteinen, die im Paket KOP, FUP, AWL erstellt werden.
- Die Formaloperanden haben maximal 4 Zeichen.
- Die Definition als Eingang, Ausgang usw. steht in Klammern.
- Beim Programmieren muß dem Formaloperanden ein Gleichheitszeichen vorausgehen (wie in Paket KOP, FUP, AWL).

Bedienung am Programmiergerät

Eingabe

Voraussetzung: Das Paket AWL-Editor/Batch-Compiler ist geladen.

Steigen Sie neu in das Beispiel ein, füllen Sie die VOREINSTELLUNG mit den Dateinamen FBTEST für die AWL-Quelldatei und Zwischendatei, und TEST@@ für Programm- und Symbolik-Datei aus. *Übernehmen* Sie und rufen Sie die Editierfunktion auf.

Haben Sie das Paket AWL-Editor/Batch-Compiler nicht verlassen, dann wird die Maske FUNKTIONSANWAHL des Pakets ausgegeben.

Gehen Sie in die VOREINSTELLUNG und ändern Sie den Namen der AWL-Quelldatei in FBTEST.

3

VOREINSTELLUNG ändern

- ➤ VOREINstellung (F6) aufrufen; der Cursor blinkt im Feld AWL-QUELLDATEI.
- ➤ Den Cursor mit der **Einfachpfeiltaste rechts** auf den Dateinamen stellen und mit *FBTEST* überschreiben.
- ➤ Return-Taste drücken, auch die Zwischendatei hat jetzt den Namen FBTEST. Die übrigen Dateien bleiben unverändert.
- ➤ **UEBERN**ehmen (F6).
- ➤ Rufen Sie mit **EDITIEREN** (F1) die Editierfunktion auf.

Jetzt können Sie beginnen, Ihren Funktionsbaustein "Garage" zu programmieren.

Erläuterungen zum Editieren von Funktionsbausteinen: Für die Schreibkonventionen der Steuerzeichen und Operationen schlagen Sie bitte in den Abschnitten 3.3.2 und 3.3.3 nach. Beachten Sie beim Eingeben die Besonderheiten für Funktionsbausteine:

- Formaloperanden werden in der Spalte ADR und ANWEISUNG definiert. Diese Definition entspricht der Bezeichnerliste in Funktionsbausteinen, die im Paket KOP, FUP, AWL erstellt werden.
- Die Formaloperanden haben maximal 4 Zeichen.
- Die Definition als Eingang, Ausgang usw. steht in Klammern.
- Beim Programmieren muß dem Formaloperanden ein Gleichheitszeichen vorausgehen (wie in Paket KOP, FUP, AWL).

C79000-B8500-C677-02 3 - 43

Der Funktionsbaustein FB1 namens Garage ist in der Symbolikdatei dem Symbol "Garage" zugeordnet. Deshalb kann der Bausteinanfang symbolisch programmiert werden.

Kommentare und Netzwerküberschriften werden wie beim Programmbaustein behandelt. Das Netzwerkende geben Sie über die von Ihnen belegte Funktionstaste ein (s.o. 3.3.5) und speichern Sie mit **SICHERN** (F7).

Bedienfolge (einzugebender Text ist *kursiv*, zu benutzende Funktionen sind **fett**)

- ➤ #
- ➤ **Doppelpfeiltaste rechts** drücken, in der Spalte OPERANDEN-SYMBOL *Garage* eingeben,
- ➤ **Doppelpfeiltaste rechts** drücken für den Kommentar
- ➤ *FB1 für ein Garagentor*
- ➤ **Return-Taste** drücken,
- ➤ *#N*
- ➤ Leerzeichen
- ➤ *Garage*
- ➤ **Return-Taste** drücken,
- ➤ **Doppelpfeiltaste links** drücken, der Cursor steht in der Spalte ADRessen.
- ➤ *ENDO* schreiben, der Cursor springt nach dem vierten Zeichen in die Spalte ANWEISUNG
- ➤ *(E)*
- ➤ **Doppelpfeiltaste rechts** zweimal drücken. Als Anweisungskommentar schreiben Sie
- ➤ *Endschalter oben*
- ➤ **Return-Taste** drücken;

und so weiter.

Bedienung am Programmiergerät

Der Funktionsbaustein FB1 namens Garage ist in der Symbolikdatei dem Symbol "Garage" zugeordnet. Deshalb kann der Bausteinanfang symbolisch programmiert werden.

Kommentare und Netzwerküberschriften werden wie beim Programmbaustein behandelt. Das Netzwerkende geben Sie über die von Ihnen belegte Funktionstaste ein (s.o. 3.3.5) und speichern Sie mit **SICHERN** (F7).

Bedienfolge (einzugebender Text ist *kursiv*, zu benutzende Funktionen sind **fett**)

- ➤ #
- ➤ **Doppelpfeiltaste rechts** drücken, in der Spalte OPERANDEN-SYMBOL *Garage* eingeben,
- ➤ **Doppelpfeiltaste rechts** drücken für den Kommentar
- ➤ *FB1 für ein Garagentor*
- ➤ **Return-Taste** drücken,
- ➤ *#N*
- ➤ Leerzeichen
- ➤ *Garage*
- ➤ **Return-Taste** drücken,
- ➤ **Doppelpfeiltaste links** drücken, der Cursor steht in der Spalte ADRessen.
- ➤ *ENDO* schreiben, der Cursor springt nach dem vierten Zeichen in die Spalte ANWEISUNG
- ➤ *(E)*
- ➤ **Doppelpfeiltaste rechts** zweimal drücken. Als Anweisungskommentar schreiben Sie
- ➤ *Endschalter oben*
- ➤ **Return-Taste** drücken;

und so weiter.

Die Anweisungen entsprechen dem Schema des Programmbausteins:

- *U(*
- **Return-Taste** drücken,
- *U*
- Leerzeichen
- *=TA-A*
- **Return-Taste** drücken;

und so weiter.

Bedienung am Programmiergerät

Die Anweisungen entsprechen dem Schema des Programmbausteins:

- *U(*
- **Return-Taste** drücken,
- *U*
- Leerzeichen
- *=TA-A*
- **Return-Taste** drücken;

und so weiter.

3

C79000-B8500-C677-02

3 - 45

Parametrieren eines Funktionsbausteins

Um den Funktionsbaustein zu parametrieren, d.h., mit aktuellen Operanden zu versehen, schreiben Sie einen Programmbaustein:

```
AWL-Quelle:  B: FBTESTA0.SEQ

ADRANWEISUNG  OPERANDENSYMBOL          ANWEISUNGSKOMMENTAR
     #PB2
     SPA              GARAGE                  Parametrieren des FB1
     ,E 1.0
     ,E 1.1
     ,                T-AUF I
     ,                T-AUF A
     ,                T-ZU I
     ,                T-ZU A
     ,                SCHLUESS
     ,                STOP
     ,                MOT-AUF
     ,                MOT-AB
     BE
```

Die Aktualoperanden können Sie absolut oder symbolisch eingeben. Beachten Sie dabei bitte,

- daß jedem Aktualoperanden ein Komma als Steuerzeichen vorausgehen muß, und
- daß die Parameter in ihrer Reihenfolge mit der Bezeichnerliste der Formaloperanden im Funktionsbaustein übereinstimmen.

Bedienung am Programmiergerät

Parametrieren eines Funktionsbausteins

Um den Funktionsbaustein zu parametrieren, d.h., mit aktuellen Operanden zu versehen, schreiben Sie einen Programmbaustein:

```
AWL-Quelle:  B: FBTESTA0.SEQ

ADRANWEISUNG  OPERANDENSYMBOL          ANWEISUNGSKOMMENTAR
     #PB2
     SPA              GARAGE                  Parametrieren des FB1
     ,E 1.0
     ,E 1.1
     ,                T-AUF I
     ,                T-AUF A
     ,                T-ZU I
     ,                T-ZU A
     ,                SCHLUESS
     ,                STOP
     ,                MOT-AUF
     ,                MOT-AB
     BE
```

Die Aktualoperanden können Sie absolut oder symbolisch eingeben. Beachten Sie dabei bitte,

- daß jedem Aktualoperanden ein Komma als Steuerzeichen vorausgehen muß, und
- daß die Parameter in ihrer Reihenfolge mit der Bezeichnerliste der Formaloperanden im Funktionsbaustein übereinstimmen.

3 - 46

C79000-B8500-C677-02

Bedienfolge:
- ➤ *#PB2*
- ➤ **Return-Taste** drücken,
- ➤ *SPA*
- ➤ **Doppelpfeiltaste rechts** einmal drücken,
- ➤ *GARAGE*
- ➤ **Doppelpfeiltaste rechts** einmal drücken,
- ➤ *Parametrieren des FB1*
- ➤ **Return-Taste** drücken,
- ➤ *,E 1.0*
- ➤ **Return-Taste** drücken,

oder symbolisch:

- ➤ ,
- ➤ **Doppelpfeiltaste rechts** einmal drücken,
- ➤ *T-AUF A*
- ➤ **Return-Taste** drücken.

Speichern Sie zwischen mit **SICHERN** (F7), denn der in Abschnitt 3.3.7 folgende Datenbaustein wird ebenfalls in diese AWL-Quelldatei geschrieben.

Bedienung am Programmiergerät

Bedienfolge:
- ➤ *#PB2*
- ➤ **Return-Taste** drücken,
- ➤ *SPA*
- ➤ **Doppelpfeiltaste rechts** einmal drücken,
- ➤ *GARAGE*
- ➤ **Doppelpfeiltaste rechts** einmal drücken,
- ➤ *Parametrieren des FB1*
- ➤ **Return-Taste** drücken,
- ➤ *,E 1.0*
- ➤ **Return-Taste** drücken,

oder symbolisch:

- ➤ ,
- ➤ **Doppelpfeiltaste rechts** einmal drücken,
- ➤ *T-AUF A*
- ➤ **Return-Taste** drücken.

Speichern Sie zwischen mit **SICHERN** (F7), denn der in Abschnitt 3.3.7 folgende Datenbaustein wird ebenfalls in diese AWL-Quelldatei geschrieben.

3

C79000-B8500-C677-02

3 - 47

3.3.7 Eingabe von Datenbausteinen

Beispiel Der folgende Datenbaustein soll die Arbeitsgrundlage dieses Abschnitts sein.

Wollen Sie sich über Datenbausteine informieren, lesen Sie bitte in der STEP 5-Beschreibung Ihres PG-Handbuchs, Band 2, in der Einführung und im Kapitel Datenbausteine nach.

```
AWL-Quelle:  B:FBTESTA0.SEQ

ADR   ANWEISUNG        OPERANDENSYMBOL   ANWEISUNGSKOMMENTAR

      #DB 12
0     KH                FFFF
1     KM                11111111 11000000 STUECKZAHL
2     KH                0013
3     KF                -32768
4     KF                +32767
5     KG                -2740000+22
7     KG                -1234000+05
100   KY                022,033
111   KY                022,033
      KY                022,033
      KY                022,033
      KH                ADAC
      KH                4538
      KF                +32767
      KF                +32767
      KF                +32767
      KF                +32767
      KF                +32767
      KY                022,033
      KY                022,033
```


Bedienung am Programmiergerät

3.3.7 Eingabe von Datenbausteinen

Beispiel Der folgende Datenbaustein soll die Arbeitsgrundlage dieses Abschnitts sein.

Wollen Sie sich über Datenbausteine informieren, lesen Sie bitte in der STEP 5-Beschreibung Ihres PG-Handbuchs, Band 2, in der Einführung und im Kapitel Datenbausteine nach.

AWL-Quelle: B:FBTESTA0.SEQ

ADR   ANWEISUNG   OPERANDENSYMBOL   ANWEISUNGSKOMMENTAR

      #DB 12
0   KH   FFFF
1   KM   11111111 11000000 STUECKZAHL
2   KH   0013
3   KF   -32768
4   KF   +32767
5   KG   -2740000+22
7   KG   -1234000+05
100   KY   022,033
111   KY   022,033
      KY   022,033
      KY   022,033
      KH   ADAC
      KH   4538
      KF   +32767
      KF   +32767
      KF   +32767
      KF   +32767
      KF   +32767
      KY   022,033
      KY   022,033

Eingabe

Voraussetzung: Das Paket AWL-Editor/Batch-Compiler ist geladen.

Haben Sie gerade das Beispiel für Funktionsbausteine durchgeführt, befinden Sie sich in der Editierfunktion und die Datei FBTEST ist ausgegeben.

Steigen Sie neu in dieses Beispiel ein, füllen Sie die VOREINSTELLUNG mit den Dateinamen FBTEST für die AWL-Quelldatei und Zwischendatei, und TEST@@ für die Programm- und Symbolik-Datei aus. *Übernehmen* Sie und rufen Sie die Editierfunktion auf.

Erläuterungen zum Editieren von Datenbausteinen

Für die Schreibkonventionen der Steuerzeichen und Datenwörter schlagen Sie bitte in den Abschnitten 3.3.2 und 3.3.3 nach.

Beachten Sie beim Eingeben die Besonderheiten für Datenbausteine:
- Das Datenformat steht in der Spalte ANWEISUNGEN
- Der Wert in der Spalte OPERANDENSYMBOL.
- Die Datenwortadresse, relativ zum Bausteinanfang, steht in der Spalte ADR.
- Sie brauchen Ihre Datenworte weder mit Gleichheitszeichen zu markieren, noch mit Strichpunkt abzuschließen.

TIP: Geben Sie eine Adresse ein, die nicht mit der tatsächlichen Adresse in DB übereinstimmt, dann wird beim Übersetzen die Lücke mit KH 0000 aufgefüllt (im Beispiel die Adressen 9 bis 99). Dadurch schaffen Sie Platz für Daten aus dem Prozess.

Im Unterschied zum Paket KOP, FUP, AWL kann der Wiederholungsfaktor nicht direkt benutzt werden, sondern nur über die Kombination mit der Funktion **KOPIERE**.

Bedienfolge:
- ➤ *#DB12*
- ➤ **Return-Taste** drücken
- ➤ **Doppelpfeiltaste links** einmal drücken und die Adresse
- ➤ *0* eingeben.

Bedienung am Programmiergerät

Eingabe

Voraussetzung: Das Paket AWL-Editor/Batch-Compiler ist geladen.

Haben Sie gerade das Beispiel für Funktionsbausteine durchgeführt, befinden Sie sich in der Editierfunktion und die Datei FBTEST ist ausgegeben.

Steigen Sie neu in dieses Beispiel ein, füllen Sie die VOREINSTELLUNG mit den Dateinamen FBTEST für die AWL-Quelldatei und Zwischendatei, und TEST@@ für die Programm- und Symbolik-Datei aus. *Übernehmen* Sie und rufen Sie die Editierfunktion auf.

Erläuterungen zum Editieren von Datenbausteinen

Für die Schreibkonventionen der Steuerzeichen und Datenwörter schlagen Sie bitte in den Abschnitten 3.3.2 und 3.3.3 nach.

3

Beachten Sie beim Eingeben die Besonderheiten für Datenbausteine:
- Das Datenformat steht in der Spalte ANWEISUNGEN
- Der Wert in der Spalte OPERANDENSYMBOL.
- Die Datenwortadresse, relativ zum Bausteinanfang, steht in der Spalte ADR.
- Sie brauchen Ihre Datenworte weder mit Gleichheitszeichen zu markieren, noch mit Strichpunkt abzuschließen.

TIP: Geben Sie eine Adresse ein, die nicht mit der tatsächlichen Adresse in DB übereinstimmt, dann wird beim Übersetzen die Lücke mit KH 0000 aufgefüllt (im Beispiel die Adressen 9 bis 99). Dadurch schaffen Sie Platz für Daten aus dem Prozess.

Im Unterschied zum Paket KOP, FUP, AWL kann der Wiederholungsfaktor nicht direkt benutzt werden, sondern nur über die Kombination mit der Funktion **KOPIERE**.

Bedienfolge:
- ➤ *#DB12*
- ➤ **Return-Taste** drücken
- ➤ **Doppelpfeiltaste links** einmal drücken und die Adresse
- ➤ *0* eingeben.

C79000-B8500-C677-02

3 - 49

➤ **Doppelpfeiltaste rechts** einmal drücken. Geben Sie in der ANWEISUNGsspalte

➤ *KH* ein.

➤ **Doppelpfeiltaste rechts** einmal drücken. Geben Sie in die OPERANDENSYMBOLspalte den Wert

➤ *FFFF* ein,

➤ **Return-Taste** drücken.

➤ **Doppelpfeiltaste links** einmal drücken und die Adresse

➤ *1* eingeben.

➤ **Doppelpfeiltaste rechts** drücken,

➤ *KM* eingeben,

➤ **Doppelpfeiltaste rechts** drücken,

➤ *11111111 11000000* eingeben,

➤ **Doppelpfeiltaste rechts** drücken,

➤ *STUECKZAHL* als Kommentar schreiben,

➤ **Return-Taste** drücken

und so weiter.

Bei KY 22,23

➤ schreiben Sie die Zeile einmal,

➤ merken sie mit **MERKE** (F1),

➤ **ZEILE** (F1), **ZURUECK** (F8), **Return**-Taste, und

➤ kopieren sie mit **KOPIERE** (F2),

➤ Wiederholungsfaktor 3 und

➤ **ZEILE** (F1).

Sie können zum Vervielfälltigen auch die Funktion KOPIERE (F2) BLOCK (F4) verwenden.

Schließen Sie nun Ihre Übungsbeispiele mit **UEBERN**ehmen (F6) ab. Die AWL-Quelldatei FBTEST wird damit gespeichert und in die Zwischendatei übersetzt. Das *Editieren* ist ebenfalls beendet.

Bedienung am Programmiergerät

➤ **Doppelpfeiltaste rechts** einmal drücken. Geben Sie in der ANWEISUNGsspalte

➤ *KH* ein.

➤ **Doppelpfeiltaste rechts** einmal drücken. Geben Sie in die OPERANDENSYMBOLspalte den Wert

➤ *FFFF* ein,

➤ **Return-Taste** drücken.

➤ **Doppelpfeiltaste links** einmal drücken und die Adresse

➤ *1* eingeben.

➤ **Doppelpfeiltaste rechts** drücken,

➤ *KM* eingeben,

➤ **Doppelpfeiltaste rechts** drücken,

➤ *11111111 11000000* eingeben,

➤ **Doppelpfeiltaste rechts** drücken,

➤ *STUECKZAHL* als Kommentar schreiben,

➤ **Return-Taste** drücken

und so weiter.

Bei KY 22,23

➤ schreiben Sie die Zeile einmal,

➤ merken sie mit **MERKE** (F1),

➤ **ZEILE** (F1), **ZURUECK** (F8), **Return**-Taste, und

➤ kopieren sie mit **KOPIERE** (F2),

➤ Wiederholungsfaktor 3 und

➤ **ZEILE** (F1).

Sie können zum Vervielfälltigen auch die Funktion KOPIERE (F2) BLOCK (F4) verwenden.

Schließen Sie nun Ihre Übungsbeispiele mit **UEBERN**ehmen (F6) ab. Die AWL-Quelldatei FBTEST wird damit gespeichert und in die Zwischendatei übersetzt. Das *Editieren* ist ebenfalls beendet.

3 - 50

C79000-B8500-C677-02

3.3.8 Ändern einer AWL-Quelldatei

Wollen Sie eine AWL-Quelldatei innerhalb des AWL-Editors/Batch-Compilers ändern, geben Sie sie mit *Editieren* auf den Bildschirm aus und bearbeiten sie mit den Editierfunktionen.

In unserem Beispiel soll die Datei FBTEST über den Include-Befehl in die AWL-Quelldatei TEST@@ eingebunden werden. Dazu muß FBTESTA0.SEQ als Zwischendatei vorhanden sein. Diese Voraussetzung ist in unserem Fall bereits erfüllt (s.o.)

Voraussetzung:

In der Voreinstellung TEST@@ als AWL-Quelldatei eintragen.

➤ **EDITIEREN** (F1) der Datei TEST@@.

Springen Sie ans Ende der Datei mit

➤ **SUCHE** (F4),

➤ **ENDE** (F7), und

➤ gehen Sie dann wieder **ZURUECK** (F8) in den Editiermodus.

Der Einfügemodus ist voreingestellt.

➤ Stellen Sie den Cursor vor den ersten Baustein, zwischen BE und #PBn oder an das Dateiende nach dem letzten Bausteinende BE;

➤ spreizen Sie vertikal; jetzt haben Sie Platz für den Include-Befehl.

➤ *#I*

➤ Leerzeichen

➤ *B:FBTEST*

➤ **UEBERN** (F6) drücken, um zu speichern und zu übersetzen. Damit ist Ihre Zwischendatei aktualisiert.

Wenn Sie jetzt die AWL-Quelldatei TEST@@A0.SEQ in die STEP 5-Programmdatei TEST@@ST.S5D übersetzen, wird auch FBTESTA1.SEQ übersetzt und in die Programmdatei übertragen. Dort sind dann alle während dieser Übungssitzung editierten Bausteine vorhanden.

Bedienung am Programmiergerät

3.3.8 Ändern einer AWL-Quelldatei

Wollen Sie eine AWL-Quelldatei innerhalb des AWL-Editors/Batch-Compilers ändern, geben Sie sie mit *Editieren* auf den Bildschirm aus und bearbeiten sie mit den Editierfunktionen.

In unserem Beispiel soll die Datei FBTEST über den Include-Befehl in die AWL-Quelldatei TEST@@ eingebunden werden. Dazu muß FBTESTA0.SEQ als Zwischendatei vorhanden sein. Diese Voraussetzung ist in unserem Fall bereits erfüllt (s.o.)

Voraussetzung:

In der Voreinstellung TEST@@ als AWL-Quelldatei eintragen.

➤ **EDITIEREN** (F1) der Datei TEST@@.

Springen Sie ans Ende der Datei mit

➤ **SUCHE** (F4),

➤ **ENDE** (F7), und

➤ gehen Sie dann wieder **ZURUECK** (F8) in den Editiermodus.

Der Einfügemodus ist voreingestellt.

➤ Stellen Sie den Cursor vor den ersten Baustein, zwischen BE und #PBn oder an das Dateiende nach dem letzten Bausteinende BE;

➤ spreizen Sie vertikal; jetzt haben Sie Platz für den Include-Befehl.

➤ *#I*

➤ Leerzeichen

➤ *B:FBTEST*

➤ **UEBERN** (F6) drücken, um zu speichern und zu übersetzen. Damit ist Ihre Zwischendatei aktualisiert.

Wenn Sie jetzt die AWL-Quelldatei TEST@@A0.SEQ in die STEP 5-Programmdatei TEST@@ST.S5D übersetzen, wird auch FBTESTA1.SEQ übersetzt und in die Programmdatei übertragen. Dort sind dann alle während dieser Übungssitzung editierten Bausteine vorhanden.

C79000-B8500-C677-02

3 - 51

3.4 Übersetzen mit der Funktion COMPILER

Ihre mit *Übernahme* abgespeicherte AWL-Quelldatei liegt danach als Zwischendatei (ZWI) vor. Um diese nun in eine STEP 5-Programmdatei zu übersetzen, rufen Sie die **COMPILER**-Funktion auf. Dort können Sie Ihre Anweisungsliste in die Programmdatei übersetzen, die in der Voreinstellung genannt ist. Mit **ZWI>MC5** wird die Zwischendatei in den Maschinencode MC5 gewandelt, mit **SEQ>MC5** die AWL-Quelldatei unter automatischer Erzeugung der Zwischendatei.

Ebenso können Sie rückübersetzen: Aus einer Programmdatei in MC5 wird mit **MC5>ZWI** eine Zwischendatei (Eine solche wird mit den **SONDER**funktionen zu einer AWL-Quelldatei weiterbearbeitet.), oder mit **MC5>SEQ** direkt eine AWL-Quelldatei mit der entsprechenden Zwischendatei.

Die Funktion SEQ>MC5 führt zunächst die Übersetzung SEQ>ZWI aus. Treten dabei Fehler auf, wird die Übersetzung ZWI>MC5 nicht gestartet, sondern die Funktion beendet. Damit stehen in der Fehlerliste die beim Erzeugen der Zwischendatei aufgetretenen Fehlermeldungen. Analog startet die Funktion MC5>SEQ zuerst die Übersetzung MC5>ZWI und nur bei fehlerfreier Erzeugung der Zwischendatei die Übersezung ZWI>SEQ.

3.4.5 Bedienfolge: Übersetzen in die Programmdatei

Voraussetzung: In der Voreinstellung steht die AWL-Quelldatei FBTESTA0.SEQ.

Übersetzen

- ➤ Die **COMPILER**-Funktion mit F2 aufrufen,
- ➤ **ZWI>MC5** (F2) oder **SEQ>MC5** (F1) drücken.
- ➤ Die folgende Kommandozeile ausfüllen:

Uebersetzen der Bausteine:	OPT:	DRU:

Die **Help-Taste** gibt Ihnen zu jedem Eingabefeld Auskunft über die möglichen Eingaben.

Bedienung am Programmiergerät

3.4 Übersetzen mit der Funktion COMPILER

Ihre mit *Übernahme* abgespeicherte AWL-Quelldatei liegt danach als Zwischendatei (ZWI) vor. Um diese nun in eine STEP 5-Programmdatei zu übersetzen, rufen Sie die **COMPILER**-Funktion auf. Dort können Sie Ihre Anweisungsliste in die Programmdatei übersetzen, die in der Voreinstellung genannt ist. Mit **ZWI>MC5** wird die Zwischendatei in den Maschinencode MC5 gewandelt, mit **SEQ>MC5** die AWL-Quelldatei unter automatischer Erzeugung der Zwischendatei.

Ebenso können Sie rückübersetzen: Aus einer Programmdatei in MC5 wird mit **MC5>ZWI** eine Zwischendatei (Eine solche wird mit den **SONDER**funktionen zu einer AWL-Quelldatei weiterbearbeitet.), oder mit **MC5>SEQ** direkt eine AWL-Quelldatei mit der entsprechenden Zwischendatei.

Die Funktion SEQ>MC5 führt zunächst die Übersetzung SEQ>ZWI aus. Treten dabei Fehler auf, wird die Übersetzung ZWI>MC5 nicht gestartet, sondern die Funktion beendet. Damit stehen in der Fehlerliste die beim Erzeugen der Zwischendatei aufgetretenen Fehlermeldungen. Analog startet die Funktion MC5>SEQ zuerst die Übersetzung MC5>ZWI und nur bei fehlerfreier Erzeugung der Zwischendatei die Übersezung ZWI>SEQ.

3.4.5 Bedienfolge: Übersetzen in die Programmdatei

Voraussetzung: In der Voreinstellung steht die AWL-Quelldatei FBTESTA0.SEQ.

Übersetzen

- ➤ Die **COMPILER**-Funktion mit F2 aufrufen,
- ➤ **ZWI>MC5** (F2) oder **SEQ>MC5** (F1) drücken.
- ➤ Die folgende Kommandozeile ausfüllen:

Uebersetzen der Bausteine:	OPT:	DRU:

Die **Help-Taste** gibt Ihnen zu jedem Eingabefeld Auskunft über die möglichen Eingaben.

3 - 52 C79000-B8500-C677-02

➤ Im Feld BAUSTEINE die **Help-Taste** drücken:

Zusätzlich zu den im STEP 5-Basispaket üblichen Eingaben können Sie Bausteinbereiche angeben, die bearbeitet werden sollen, z.B. PB12 - PB21.

➤ Schreiben Sie *B* in dieses Feld und schließen Sie mit

➤ **Return-Taste** ab.

➤ **Help-Taste** im Feld OPTion drücken.

Mit "2" können Sie einen Übersetzungstest anstoßen: Ihre Zwischendatei wird übersetzt und dabei auf Fehler geprüft, sie wird aber nicht in die Programmdatei abgespeichert. Die aufgetretenen Fehler sind über die Fehlerliste zugänglich.

➤ Geben Sie *2* ein und

➤ drücken Sie die **Return-Taste**.

Enthält Ihre Programmdatei bereits namensgleiche Bausteine und ist das OPTionsfeld leer, fordert Sie das Programmiergerät dazu auf, jeden Kopiervorgang zu bestätigen; in diesem Fall wird der bisherige Baustein jeweils mit dem neuen, namensgleichen Baustein überschrieben.
Verwenden Sie die Option "1", werden die Bausteine in der Programmdatei ohne Ihre Bestätigung überschrieben.

➤ Im Feld DRUcker die **Help-Taste** drücken.

Die Formate für Druckerausgaben sind die im STEP 5-Basispaket üblichen: Normalschrift, Schmalschrift und Superschmalschrift. Die Blattgröße DIN A3 oder A4 hängt vom angeschlossenen Drucker ab.

In unserem Beispiel bleibt dieses Feld leer.

Uebersetzen der Bausteine: **B**	OPT: **2**	DRU:

➤ **Übernahmetaste** drücken.

Das Programmiergerät führt nun die Übersetzung und Prüfung durch. Es meldet, welche Bausteine bearbeitet werden und wieviele Fehler aufgetreten sind, bzw. daß fehlerfrei übersetzt worden ist. Daraufhin springt das Programmiergerät in die Funktionsanwahl zurück.

Bedienung am Programmiergerät

➤ Im Feld BAUSTEINE die **Help-Taste** drücken:

Zusätzlich zu den im STEP 5-Basispaket üblichen Eingaben können Sie Bausteinbereiche angeben, die bearbeitet werden sollen, z.B. PB12 - PB21.

➤ Schreiben Sie *B* in dieses Feld und schließen Sie mit

➤ **Return-Taste** ab.

➤ **Help-Taste** im Feld OPTion drücken.

Mit "2" können Sie einen Übersetzungstest anstoßen: Ihre Zwischendatei wird übersetzt und dabei auf Fehler geprüft, sie wird aber nicht in die Programmdatei abgespeichert. Die aufgetretenen Fehler sind über die Fehlerliste zugänglich.

3

➤ Geben Sie *2* ein und

➤ drücken Sie die **Return-Taste**.

Enthält Ihre Programmdatei bereits namensgleiche Bausteine und ist das OPTionsfeld leer, fordert Sie das Programmiergerät dazu auf, jeden Kopiervorgang zu bestätigen; in diesem Fall wird der bisherige Baustein jeweils mit dem neuen, namensgleichen Baustein überschrieben.
Verwenden Sie die Option "1", werden die Bausteine in der Programmdatei ohne Ihre Bestätigung überschrieben.

➤ Im Feld DRUcker die **Help-Taste** drücken.

Die Formate für Druckerausgaben sind die im STEP 5-Basispaket üblichen: Normalschrift, Schmalschrift und Superschmalschrift. Die Blattgröße DIN A3 oder A4 hängt vom angeschlossenen Drucker ab.

In unserem Beispiel bleibt dieses Feld leer.

Uebersetzen der Bausteine: **B**	OPT: **2**	DRU:

➤ **Übernahmetaste** drücken.

Das Programmiergerät führt nun die Übersetzung und Prüfung durch. Es meldet, welche Bausteine bearbeitet werden und wieviele Fehler aufgetreten sind, bzw. daß fehlerfrei übersetzt worden ist. Daraufhin springt das Programmiergerät in die Funktionsanwahl zurück.

C79000-B8500-C677-02

3 - 53

Ist die Übersetzung fehlerfrei, wiederholen Sie den Übersetzungsvorgang. Lassen Sie das Optionsfeld leer, damit die Programmdatei erstellt wird, und geben Sie im DRUckfeld ✱ ein, um einen Aus

druck zu erhalten.

Uebersetzen der Bausteine: **B** OPT: DRU: ✱

➤ **Übernahmetaste** drücken.

Jetzt werden die Bausteine der AWL-Quelldatei FBTESTA0.SEQ als Maschinencode in die Programmdatei TEST@@ST.S5D übertragen und abgespeichert. Sie können jetzt die Bausteine FB1 und PB2 im Paket KOP, FUP, AWL weiter verarbeiten (z.B. im AG testen).

3.4.6 Bedienfolge: Rückübersetzen der Programmdatei

Die Bedienung ist analog zum Übersetzen. Sie benutzen nur die Funktion **MC5>ZWI** (F4) bzw. **MC5>SEQ** (F5). Wichtig für das *Rückübersetzen* ist, daß die entsprechenden Dateien in der Voreinstellung stehen. Für die Kommandozeile gelten die gleichen Konventionen wie beim *Übersetzen*. Die Help-Taste im OPTionsfeld bietet Ihnen folgende Varianten der Übertragung an:

➤ **COMPILER** (2) aufrufen,

➤ **MC5>ZWI** (F4) wählen,

➤ die Kommandozeile ausfüllen,

➤ **Return-Taste** drücken.

Die Zwischendatei wird nun neu angelegt. Existiert bereits eine Zwischendatei gleichen Namens, wird sie (nach Rückfrage) überschrieben. Die Funktion **MC5>SEQ** erzeugt automatisch die AWL-Quelldatei, die Sie mit dem AWL-Editor bearbeiten können (siehe 3.4). Bei der Funktion **MC5>ZWI** müssen Sie selbst mit der SONDERfunktion ZWI>SEQ aus der Zwischendatei eine sequentielle Quelldatei erstellen (s.u., Abschnitt 3.7).

Bedienung am Programmiergerät

Ist die Übersetzung fehlerfrei, wiederholen Sie den Übersetzungsvorgang. Lassen Sie das Optionsfeld leer, damit die Programmdatei erstellt wird, und geben Sie im DRUckfeld ✱ ein, um einen Aus

druck zu erhalten.

Uebersetzen der Bausteine: **B** OPT: DRU: ✱

➤ **Übernahmetaste** drücken.

Jetzt werden die Bausteine der AWL-Quelldatei FBTESTA0.SEQ als Maschinencode in die Programmdatei TEST@@ST.S5D übertragen und abgespeichert. Sie können jetzt die Bausteine FB1 und PB2 im Paket KOP, FUP, AWL weiter verarbeiten (z.B. im AG testen).

3.4.6 Bedienfolge: Rückübersetzen der Programmdatei

Die Bedienung ist analog zum Übersetzen. Sie benutzen nur die Funktion **MC5>ZWI** (F4) bzw. **MC5>SEQ** (F5). Wichtig für das *Rückübersetzen* ist, daß die entsprechenden Dateien in der Voreinstellung stehen. Für die Kommandozeile gelten die gleichen Konventionen wie beim *Übersetzen*. Die Help-Taste im OPTionsfeld bietet Ihnen folgende Varianten der Übertragung an:

➤ **COMPILER** (2) aufrufen,

➤ **MC5>ZWI** (F4) wählen,

➤ die Kommandozeile ausfüllen,

➤ **Return-Taste** drücken.

Die Zwischendatei wird nun neu angelegt. Existiert bereits eine Zwischendatei gleichen Namens, wird sie (nach Rückfrage) überschrieben. Die Funktion **MC5>SEQ** erzeugt automatisch die AWL-Quelldatei, die Sie mit dem AWL-Editor bearbeiten können (siehe 3.4). Bei der Funktion **MC5>ZWI** müssen Sie selbst mit der SONDERfunktion ZWI>SEQ aus der Zwischendatei eine sequentielle Quelldatei erstellen (s.u., Abschnitt 3.7).

3.5 Fehlerliste

In der Fehlerliste stehen nicht nur die Fehler, die beim *Übersetzen* aufgetreten sind, sondern sie bietet ein vollständiges Protokoll der Übersetzung: Sie listet zusätzlich die fehlerfrei übersetzten Bausteine auf und im Falle eines Abbruchs gibt sie die entsprechende Stelle an.

Um ein echtes Beispiel zu haben, bauen Sie in Ihren FB1 (programmiert im Abschnitt 3.3.5) einen Fehler ein: **EDITIEREN** Sie den FB1 und schreiben Sie z.B. die Rücksetzebefehle nur mit *R*. Bereits beim Abspeichern mit **UEBERN**ehmen werden Sie darauf hingewiesen, daß Fehler aufgetreten sind.

- ➤ Rufen Sie nun die **F-LISTE** (F3) auf.
- ➤ Füllen Sie in der Kommandozeile das Eingabefeld DRUcker aus, damit Sie mit diesem Ausdruck die AWL-Quelle bequem korrigieren können. Die Help-Taste zeigt Ihnen die Parameter für das DRUckerfeld an.
- ➤ **Übernahmetaste** drücken: Jede fehlerhafte Anweisung wird mit Bausteinkennung und Zeilenzahl angezeigt und erläutert, ebenso werden die richtig übersetzten Bausteine aufgelistet.

```
Datei B:FBTESTAF.SEQ
UEBERSETZUNG AWL-QUELLE B:FBTESTA0.SEQ  ⇒ ZWISCHENDATEI B:FBTESTA1.SEQ
       R =MAUF
*** FEHLER IN ZEILE    28: OPERAND UNZULAESSIG   ***
       R =MAB
*** FEHLER IN ZEILE    45: OPERAND UNZULAESSIG   ***
*** FB 1  UEBERSETZT,     2 FEHLER GEFUNDEN   ***
*** PB 1  UEBERSETZT, BAUSTEIN FEHLERFREI   ***
*** UEBERSETZUNG BEENDET,     2  FEHLER,  KEINE WARNUNG(EN)   ***
```

Ausgeben der Fehlerliste

Auf den Bildschirm wird die Fehlerliste ausgegeben, wenn Sie das DRUckerfeld in der Kommandozeile nicht ausfüllen. Bei längeren Fehlerlisten wird die Bildschirmausgabe nach jeweils 20 Zeilen angehalten und Sie können entweder mit der Abbruchtaste die Ausgabe abbrechen oder sich mit der Übernahmetaste die nächste Bildschirmseite anzeigen lassen.

Bedienung am Programmiergerät

3.5 Fehlerliste

In der Fehlerliste stehen nicht nur die Fehler, die beim *Übersetzen* aufgetreten sind, sondern sie bietet ein vollständiges Protokoll der Übersetzung: Sie listet zusätzlich die fehlerfrei übersetzten Bausteine auf und im Falle eines Abbruchs gibt sie die entsprechende Stelle an.

Um ein echtes Beispiel zu haben, bauen Sie in Ihren FB1 (programmiert im Abschnitt 3.3.5) einen Fehler ein: **EDITIEREN** Sie den FB1 und schreiben Sie z.B. die Rücksetzebefehle nur mit *R*. Bereits beim Abspeichern mit **UEBERN**ehmen werden Sie darauf hingewiesen, daß Fehler aufgetreten sind.

3

- ➤ Rufen Sie nun die **F-LISTE** (F3) auf.
- ➤ Füllen Sie in der Kommandozeile das Eingabefeld DRUcker aus, damit Sie mit diesem Ausdruck die AWL-Quelle bequem korrigieren können. Die Help-Taste zeigt Ihnen die Parameter für das DRUckerfeld an.
- ➤ **Übernahmetaste** drücken: Jede fehlerhafte Anweisung wird mit Bausteinkennung und Zeilenzahl angezeigt und erläutert, ebenso werden die richtig übersetzten Bausteine aufgelistet.

```
Datei B:FBTESTAF.SEQ
UEBERSETZUNG AWL-QUELLE B:FBTESTA0.SEQ  ⇒ ZWISCHENDATEI B:FBTESTA1.SEQ
       R =MAUF
*** FEHLER IN ZEILE    28: OPERAND UNZULAESSIG   ***
       R =MAB
*** FEHLER IN ZEILE    45: OPERAND UNZULAESSIG   ***
*** FB 1  UEBERSETZT,     2 FEHLER GEFUNDEN   ***
*** PB 1  UEBERSETZT, BAUSTEIN FEHLERFREI   ***
*** UEBERSETZUNG BEENDET,     2  FEHLER,  KEINE WARNUNG(EN)   ***
```

Ausgeben der Fehlerliste

Auf den Bildschirm wird die Fehlerliste ausgegeben, wenn Sie das DRUckerfeld in der Kommandozeile nicht ausfüllen. Bei längeren Fehlerlisten wird die Bildschirmausgabe nach jeweils 20 Zeilen angehalten und Sie können entweder mit der Abbruchtaste die Ausgabe abbrechen oder sich mit der Übernahmetaste die nächste Bildschirmseite anzeigen lassen.

C79000-B8500-C677-02 3 - 55

3.6 Drucken

Mit dieser Funktion drucken Sie nur die voreingestellte AWL-Quelldatei. Sie brauchen deshalb in der Kommandozeile nur das Layout Ihrer Druckausgabe bestimmen. (Details zum Druckbild schlagen Sie bitte in der STEP 5-Beschreibung, Handbuch Band 2, im Kapitel zur Ein-/Ausgabe von AWL-Bausteinen nach). Übersetzte Dateien können nur über die Kommandozeilen der Funktion COMPILER auf Drucker ausgegeben werden.

Voraussetzung: Der Drucker ist angeschlossen und betriebsbereit; haben Sie einen Fremddrucker, muß dieser im DIENSTPROGramm DRUCKER parametriert werden. Voreingestellt ist der Drucker PT 88. In der Voreinstellung des AWL-Editors/Batch-Compilers steht der Name der zu druckenden Datei, z.B. FBTEST. Die Funktionsanwahl ist am Bildschirm ausgegeben.

- ➤ **DRUCKEN** (F4) aufrufen
- ➤ DRUckerfeld ausfüllen, voreingestellt ist die Standardausgabe in Normalschrift.
- ➤ **Übernahmetaste** drücken.

Die AWL-Quelldatei FBTESTAQ.SEQ wird gedruckt. Das Programmiergerät springt in die Funktionsanwahl zurück.

Sie können wie im Paket KOP, FUP, AWL die Druckausgabe über eine Datei leiten. Diese Datei definieren Sie im DIENSTPROGramm DRUCKER und tragen dann den Namen der Drucker-Datei in der VOREINSTELLUNG ein. Für Ihren aktuellen Druckauftrag legen Sie wie gewohnt in der Kommandozeile ein Layout fest. Dieses Layout wird mit in die Druckdatei übertragen.

Bedienung am Programmiergerät

3.6 Drucken

Mit dieser Funktion drucken Sie nur die voreingestellte AWL-Quelldatei. Sie brauchen deshalb in der Kommandozeile nur das Layout Ihrer Druckausgabe bestimmen. (Details zum Druckbild schlagen Sie bitte in der STEP 5-Beschreibung, Handbuch Band 2, im Kapitel zur Ein-/Ausgabe von AWL-Bausteinen nach). Übersetzte Dateien können nur über die Kommandozeilen der Funktion COMPILER auf Drucker ausgegeben werden.

Voraussetzung: Der Drucker ist angeschlossen und betriebsbereit; haben Sie einen Fremddrucker, muß dieser im DIENSTPROGramm DRUCKER parametriert werden. Voreingestellt ist der Drucker PT 88. In der Voreinstellung des AWL-Editors/Batch-Compilers steht der Name der zu druckenden Datei, z.B. FBTEST. Die Funktionsanwahl ist am Bildschirm ausgegeben.

- ➤ **DRUCKEN** (F4) aufrufen
- ➤ DRUckerfeld ausfüllen, voreingestellt ist die Standardausgabe in Normalschrift.
- ➤ **Übernahmetaste** drücken.

Die AWL-Quelldatei FBTESTAQ.SEQ wird gedruckt. Das Programmiergerät springt in die Funktionsanwahl zurück.

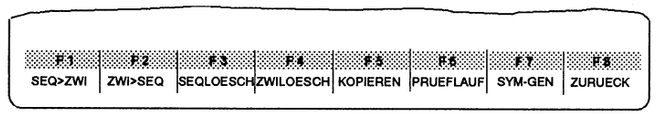
Sie können wie im Paket KOP, FUP, AWL die Druckausgabe über eine Datei leiten. Diese Datei definieren Sie im DIENSTPROGramm DRUCKER und tragen dann den Namen der Drucker-Datei in der VOREINSTELLUNG ein. Für Ihren aktuellen Druckauftrag legen Sie wie gewohnt in der Kommandozeile ein Layout fest. Dieses Layout wird mit in die Druckdatei übertragen.

3 - 56 C79000-B8500-C677-02

3.7 SONDERfunktionen zur Bearbeitung von Zwischen- und Quelldateien

Die **Sonderfunktionen** dienen der Bearbeitung und Umwandlung von sequentiellen Dateien und Zwischendateien und bieten Ihnen einen nachgeschalteten Prüflauf für die übersetzte Programmdatei. Alle Vorgänge beziehen sich auf die Dateien, die in der Voreinstellung festgelegt werden. Achten Sie deshalb darauf, daß dort die für Ihr Vorhaben richtigen Dateien eingetragen sind. In unserem Beispiel lassen wir die Voreinstellung unverändert.

Jeder Vorgang kann mit der Abbruchtaste gestoppt werden.



3

3.7.5 KOPIEREN

Um Sicherungskopien anzulegen benutzen Sie die Funktion KOPIEREN. Sie ermöglicht zunächst die Kopie der Zwischendatei, dann aber auch eine Kopie der AWL-Quelldatei auf ein anderes Laufwerk. Der Kopiervorgang wird vom Programmiergerät kommentiert, z.B. mit "Hardware-Fehler", wenn das Diskettenlaufwerk nicht geschlossen ist.

Um die Beispieldateien gefahrlos zu bearbeiten, kopieren Sie sie auf eine Diskette.

- ➤ In der VOREINSTELLUNG die AWL-Quelldatei eintragen,
- ➤ **SONDERfunktionen** (F5) aufrufen,
- ➤ **KOPIEREN** (F5) aufrufen,
- ➤ Laufwerk eingeben: *A*
- ➤ **Übernahmetaste** drücken. Die Zwischendatei ist jetzt auf Diskette gesichert. Auf die Frage "SEQ. Quelldatei ebenfalls kopieren?"
- ➤ **Übernahmetaste** drücken (ja).

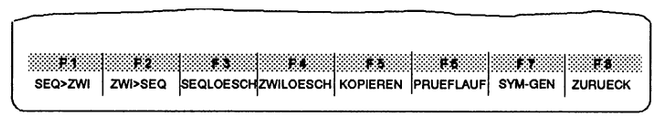
Zwischen- und AWL-Quelldatei sind jetzt auf Diskette kopiert. Das Programmiergerät springt in die Funktionsanwahl zurück.

Bedienung am Programmiergerät

3.7 SONDERfunktionen zur Bearbeitung von Zwischen- und Quelldateien

Die **Sonderfunktionen** dienen der Bearbeitung und Umwandlung von sequentiellen Dateien und Zwischendateien und bieten Ihnen einen nachgeschalteten Prüflauf für die übersetzte Programmdatei. Alle Vorgänge beziehen sich auf die Dateien, die in der Voreinstellung festgelegt werden. Achten Sie deshalb darauf, daß dort die für Ihr Vorhaben richtigen Dateien eingetragen sind. In unserem Beispiel lassen wir die Voreinstellung unverändert.

Jeder Vorgang kann mit der Abbruchtaste gestoppt werden.



3

3.7.5 KOPIEREN

Um Sicherungskopien anzulegen benutzen Sie die Funktion KOPIEREN. Sie ermöglicht zunächst die Kopie der Zwischendatei, dann aber auch eine Kopie der AWL-Quelldatei auf ein anderes Laufwerk. Der Kopiervorgang wird vom Programmiergerät kommentiert, z.B. mit "Hardware-Fehler", wenn das Diskettenlaufwerk nicht geschlossen ist.

Um die Beispieldateien gefahrlos zu bearbeiten, kopieren Sie sie auf eine Diskette.

- ➤ In der VOREINSTELLUNG die AWL-Quelldatei eintragen,
- ➤ **SONDERfunktionen** (F5) aufrufen,
- ➤ **KOPIEREN** (F5) aufrufen,
- ➤ Laufwerk eingeben: *A*
- ➤ **Übernahmetaste** drücken. Die Zwischendatei ist jetzt auf Diskette gesichert. Auf die Frage "SEQ. Quelldatei ebenfalls kopieren?"
- ➤ **Übernahmetaste** drücken (ja).

Zwischen- und AWL-Quelldatei sind jetzt auf Diskette kopiert. Das Programmiergerät springt in die Funktionsanwahl zurück.

3.7.6 SEQ>ZWI

SEQ>ZWI (F1) rufen Sie auf, wenn Sie z.B. eine AWL-Quelldatei übersetzen möchten, die mit einem anderen Texteditor erstellt worden ist. Mit dieser Funktion wird die Textdatei in eine Zwischendatei umgewandelt - als Voraussetzung für eine Übersetzung in die Programmdatei.

- ➤ In der VOREINSTELLUNG die Textdatei eintragen,
- ➤ **SONDERfunktionen** (F5) aktivieren,
- ➤ **SEQ>ZWI** (F1) aufrufen, das Gerät meldet: "Übersetzen der AWL-Quelldatei in die Zwischendatei?"
- ➤ **Übernahmetaste** drücken (ja).

Jetzt liegt eine Zwischendatei zur Weiterverarbeitung vor. Das Programmiergerät springt in die Funktionsanwahl zurück.

3.7.7 ZWI>SEQ

ZWI>SEQ (F2) rufen Sie auf, wenn Sie z.B. eine Programmdatei rückübersetzt haben (mit COMPILER, MC5>ZWI) und Sie diese im Editor verändern wollen. Dafür müssen Sie die Zwischendatei in eine sequentielle Datei umsetzen. Zusatzkommentare der ehemaligen AWL-Quelldatei gehen dabei verloren.

- ➤ In der VOREINSTELLUNG die betroffenen Dateien festlegen.
- ➤ **SONDERfunktion** (F5) aktivieren,
- ➤ **ZWI>SEQ** (F2) aufrufen, das Gerät fragt: "Übersetzen der Zwischendatei in die AWL-Quelldatei?"
- ➤ **Übernahmetaste** drücken (ja).

Damit ist eine AWL-Quelldatei neu generiert. Das Programmiergerät springt in die Funktionsanwahl zurück.

Bedienung am Programmiergerät

3.7.6 SEQ>ZWI

SEQ>ZWI (F1) rufen Sie auf, wenn Sie z.B. eine AWL-Quelldatei übersetzen möchten, die mit einem anderen Texteditor erstellt worden ist. Mit dieser Funktion wird die Textdatei in eine Zwischendatei umgewandelt - als Voraussetzung für eine Übersetzung in die Programmdatei.

- ➤ In der VOREINSTELLUNG die Textdatei eintragen,
- ➤ SONDERfunktionen (F5) aktivieren,
- ➤ SEQ>ZWI (F1) aufrufen, das Gerät meldet: "Übersetzen der AWL-Quelldatei in die Zwischendatei?"
- ➤ Übernahmetaste drücken (ja).

Jetzt liegt eine Zwischendatei zur Weiterverarbeitung vor. Das Programmiergerät springt in die Funktionsanwahl zurück.

3.7.7 ZWI>SEQ

ZWI>SEQ (F2) rufen Sie auf, wenn Sie z.B. eine Programmdatei rückübersetzt haben (mit COMPILER, MC5>ZWI) und Sie diese im Editor verändern wollen. Dafür müssen Sie die Zwischendatei in eine sequentielle Datei umsetzen. Zusatzkommentare der ehemaligen AWL-Quelldatei gehen dabei verloren.

- ➤ In der VOREINSTELLUNG die betroffenen Dateien festlegen.
- ➤ SONDERfunktion (F5) aktivieren,
- ➤ ZWI>SEQ (F2) aufrufen, das Gerät fragt: "Übersetzen der Zwischendatei in die AWL-Quelldatei?"
- ➤ Übernahmetaste drücken (ja).

Damit ist eine AWL-Quelldatei neu generiert. Das Programmiergerät springt in die Funktionsanwahl zurück.

3 - 58 C79000-B8500-C677-02

3.7.8 SEQLOESCH und ZWILOESCH

SEQLOESCH und **ZWILOESCH** löschen die voreingestelle AWL-Quell- und Zwischendatei.

- ➤ In der VOREINSTELLUNG die betroffenen Dateien festlegen.
- ➤ **SONDERfunktionen** (F5) aufrufen,
- ➤ **SEQLOESCH** (F3) aufrufen, das Gerät fragt: "Loeschen der AWL-Quelldatei?"
- ➤ **Übernahmetaste** drücken (ja).

Das Programmiergerät springt in die Funktionsanwahl zurück.

- ➤ **SONDERfunktionen** (F5) aufrufen,
- ➤ **ZWILOESCH** (F4) aufrufen, das Gerät fragt: "Loeschen der Zwischendatei?"
- ➤ **Übernahmetaste** drücken (ja) oder abbrechen (nein).

Das Programmiergerät springt in die Funktionsanwahl zurück.

3.7.9 PRUEFLAUF

Der **PRUEFLAUF** ist eine nachträgliche Prüfung von Bausteinen in der voreingestellten Programmdatei. Hier wird auch geprüft, ob Standard-Funktionsbausteine mit richtigen Parametern versorgt werden. Falls Fehler auftreten, sind diese über die Fehlerliste zugänglich.

In der Kommandozeile dieser Funktion können Sie Einzelbausteine, Bausteingruppen, Bausteintypen oder alle Bausteine einer Programmdatei eintragen; benutzen Sie die Help-Taste für diese Informationen.

Bedienung am Programmiergerät

3.7.8 SEQLOESCH und ZWILOESCH

SEQLOESCH und ZWILOESCH löschen die voreingestelle AWL-Quell- und Zwischendatei.

➤ In der VOREINSTELLUNG die betroffenen Dateien festlegen.
➤ SONDERfunktionen (F5) aufrufen,
➤ SEQLOESCH (F3) aufrufen, das Gerät fragt: "Loeschen der AWL-Quelldatei?"
➤ Übernahmetaste drücken (ja).

Das Programmiergerät springt in die Funktionsanwahl zurück.

➤ SONDERfunktionen (F5) aufrufen,
➤ ZWILOESCH (F4) aufrufen, das Gerät fragt: "Loeschen der Zwischendatei?"
➤ Übernahmetaste drücken (ja) oder abbrechen (nein).

Das Programmiergerät springt in die Funktionsanwahl zurück.

3.7.9 PRUEFLAUF

Der PRUEFLAUF ist eine nachträgliche Prüfung von Bausteinen in der voreingestellten Programmdatei. Hier wird auch geprüft, ob Standard-Funktionsbausteine mit richtigen Parametern versorgt werden. Falls Fehler auftreten, sind diese über die Fehlerliste zugänglich.

In der Kommandozeile dieser Funktion können Sie Einzelbausteine, Bausteingruppen, Bausteintypen oder alle Bausteine einer Programmdatei eintragen; benutzen Sie die Help-Taste für diese Informationen.

C79000-B8500-C677-02

3 - 59

➤ In der VOREINSTELLUNG die zu prüfende Programmdatei und eventuell den gewünschten AG-Typ festlegen,

➤ **SONDERfunktionen** (F5) aufrufen,

➤ **PRUEFLAUF** (F6) aufrufen,

➤ Kommandozeile ausfüllen: z.B. ✱ eingeben,

➤ **Übernahmetaste** drücken.

➤ Bausteinliste ausfüllen mit *PB1*.

➤ **Return-Taste** drücken,

➤ *FB1, DB12* analog eingeben,

➤ **Übernahmetaste** drücken.

Das Programmiergerät kommentiert die Prüfung. Werden Fehler festgestellt, lassen Sie sich die Fehlerliste ausgeben.

3.7.10 SYM-GEN

SYM-GEN erzeugt aus einer AWL-Quelldatei eine Symbolik-Quelldatei, die alle verwendeten Absolutparameter und Symbole enthält. Die Symbolik-Quelldatei können Sie mit Hilfe des Symbolikeditors weiter bearbeiten, um z. B. die Zuordnungen zu ergänzen und Kommentare einzutragen. Die Symbole und Absolutparameter erscheinen so oft in der Symbolikquelle, wie sie in der AWL-Quelldatei benutzt wurden. Um die Mehrfachangaben zu eliminieren, sollten Sie wie folgt vorgehen:

➤ Erzeugen der Symbolikquelle mit SYM-GEN

Beispiel:

Absolut	Symbol	Kommentar
	NOT-AUS	
	NOT-AUS	
	NOT-AUS	

➤ Wechsel in den Symbolikeditor (Paket 8)

➤ Vervollständigen der Zuordnung beim ersten Auftreten des Symbols bzw. des Absolutparameters

Bedienung am Programmiergerät

➤ In der VOREINSTELLUNG die zu prüfende Programmdatei und eventuell den gewünschten AG-Typ festlegen,

➤ **SONDERfunktionen** (F5) aufrufen,

➤ **PRUEFLAUF** (F6) aufrufen,

➤ Kommandozeile ausfüllen: z.B. ✱ eingeben,

➤ **Übernahmetaste** drücken.

➤ Bausteinliste ausfüllen mit *PB1*.

➤ **Return-Taste** drücken,

➤ *FB1, DB12* analog eingeben,

➤ **Übernahmetaste** drücken.

Das Programmiergerät kommentiert die Prüfung. Werden Fehler festgestellt, lassen Sie sich die Fehlerliste ausgeben.

3.7.10 SYM-GEN

SYM-GEN erzeugt aus einer AWL-Quelldatei eine Symbolik-Quelldatei, die alle verwendeten Absolutparameter und Symbole enthält. Die Symbolik-Quelldatei können Sie mit Hilfe des Symbolikeditors weiter bearbeiten, um z. B. die Zuordnungen zu ergänzen und Kommentare einzutragen. Die Symbole und Absolutparameter erscheinen so oft in der Symbolikquelle, wie sie in der AWL-Quelldatei benutzt wurden. Um die Mehrfachangaben zu eliminieren, sollten Sie wie folgt vorgehen:

➤ Erzeugen der Symbolikquelle mit SYM-GEN

Beispiel:

Absolut	Symbol	Kommentar
	NOT-AUS	
	NOT-AUS	
	NOT-AUS	

➤ Wechsel in den Symbolikeditor (Paket 8)

➤ Vervollständigen der Zuordnung beim ersten Auftreten des Symbols bzw. des Absolutparameters

Beispiel:

Absolut	Symbol	Kommentar
E 1.0	NOT-AUS	Not-Aus
	NOT-AUS	
	NOT-AUS	

➤ Übersetzen in die Symbolikdatei. Die Fehlermeldungen "Symbol bereits vorhanden" können sie hier ignorieren.

➤ Rückübersetzen der Symbolikdatei in die Symbolikquelle (INT > SEQ).

Beispiel:

Absolut	Symbol	Kommentar
E 1.0	NOT-AUS	Not-Aus

Die Symbolik-Quelldatei enthält jetzt nur noch eine Zuordnung, die das Symbol NOT-AUS enthält.

3

Bedienung am Programmiergerät

Beispiel:

Absolut	Symbol	Kommentar
E 1.0	NOT-AUS	Not-Aus
	NOT-AUS	
	NOT-AUS	

➤ Übersetzen in die Symbolikdatei. Die Fehlermeldungen "Symbol bereits vorhanden" können sie hier ignorieren.

➤ Rückübersetzen der Symbolikdatei in die Symbolikquelle (INT > SEQ).

Beispiel:

Absolut	Symbol	Kommentar
E 1.0	NOT-AUS	Not-Aus

Die Symbolik-Quelldatei enthält jetzt nur noch eine Zuordnung, die das Symbol NOT-AUS enthält.

3

C79000-B8500-C677-02

3 - 61

Bedienung am Programmiergerät

3 - 62

C79000-B8500-C677-02

Kommandozeilen-Version 4

Kommandozeilen-Version 4

C79000-B8500-C677-02 4 - 1

Programm COMPILE.CMD

Das Programm **COMPILE.CMD** ist ein normales CP/M-Programm, das vom Betriebssystem aus aufgerufen werden muß, wie z. B. **S5KONVER**. Sie können es daher auch in SUBMIT- Dateien verwenden, um Befehlsfolgen automatisch ablaufen zu lassen. **COMPILE.CMD** führt die Übersetzungen, die im Paket AWL-Batch-Compiler anwählbar sind, ohne weitere Benutzereingaben aus. Dazu lädt **COMPILE.CMD** die erforderlichen S5-Treiber und entfernt sie anschließend wieder. Die S5WX*.*-Dateien müssen also auf dem PG vorhanden sein, sonst bricht **COMPILE** mit einer Fehlermeldung ab.

Input-Datei

Sie können die zu bearbeitenden Dateien beim Aufruf mit angeben oder in eine sogenannte Input-Datei schreiben. Die Input-Datei kann auch die Angaben für mehrere Übersetzungsvorgänge enthalten. In den Dateinamen sind keine Jokerzeichen (?, *) erlaubt.

Die Ausführung der Übersetzungsfunktionen ist identisch mit dem Programmablauf des AWL-Batch-Compilers (Paket 9) bei Anwahl der entsprechenden Übersetzungsläufe. Die Optionen, die Sie im AWL-Batch-Compiler jeweils in der Kommandozeile eingeben können, sind nur über die Inputdatei einstellbar. Die von COMPILE.CMD verwendeten Einstellungen sind in Abschnitt 4.3 erläutert.

Kommandozeilen-Version

Programm COMPILE.CMD

Das Programm **COMPILE.CMD** ist ein normales CP/M-Programm, das vom Betriebssystem aus aufgerufen werden muß, wie z. B. **S5KONVER**. Sie können es daher auch in SUBMIT- Dateien verwenden, um Befehlsfolgen automatisch ablaufen zu lassen. **COMPILE.CMD** führt die Übersetzungen, die im Paket AWL-Batch-Compiler anwählbar sind, ohne weitere Benutzereingaben aus. Dazu lädt **COMPILE.CMD** die erforderlichen S5-Treiber und entfernt sie anschließend wieder. Die S5WX*.*-Dateien müssen also auf dem PG vorhanden sein, sonst bricht **COMPILE** mit einer Fehlermeldung ab.

Input-Datei

Sie können die zu bearbeitenden Dateien beim Aufruf mit angeben oder in eine sogenannte Input-Datei schreiben. Die Input-Datei kann auch die Angaben für mehrere Übersetzungsvorgänge enthalten. In den Dateinamen sind keine Jokerzeichen (?, *) erlaubt.

Die Ausführung der Übersetzungsfunktionen ist identisch mit dem Programmablauf des AWL-Batch-Compilers (Paket 9) bei Anwahl der entsprechenden Übersetzungsläufe. Die Optionen, die Sie im AWL-Batch-Compiler jeweils in der Kommandozeile eingeben können, sind nur über die Inputdatei einstellbar. Die von COMPILE.CMD verwendeten Einstellungen sind in Abschnitt 4.3 erläutert.

4 - 2 C79000-B8500-C677-02

4.1 Aufruf

4.1.1 Aufruf ohne Parameterangabe:

Syntax: COMPILE <QUELLE> <ZIEL> <SPRACHKENNUNG>

Beispiel: COMPILE @@@@@@A0.SEQ @@@@@@ST.S5D D

Folgende sechs Kombinationen für Quelle und Ziel sind möglich:

AWL-Quelldatei -->	Programmdatei:	nnnnnnA0.SEQ	nnnnnnST.S5D
AWL-Quelldatei -->	Zwischendatei:	nnnnnnA0.SEQ	nnnnnnA1.SEQ
Zwischendatei -->	Programmdatei:	nnnnnnA1.SEQ	nnnnnnST.S5D
Zwischendatei -->	AWL-Quelldatei:	nnnnnnA1.SEQ	nnnnnnA0.SEQ
Programmdatei -->	AWL-Quelldatei:	nnnnnnST.S5D	nnnnnnA0.SEQ
Programmdatei -->	Zwischendatei:	nnnnnnST.S5D	nnnnnnA1.SEQ

4

Die Dateinamen müssen den S5-Konventionen entsprechen, d.h., nach den sechs wahlfreien Zeichen des Namens folgen die zugelassenen Dateikennungen (A0.SEQ, A1.SEQ, ST.S5D). Es wird nicht zwischen Klein- und Großschreibung unterschieden. Wenn Sie kein Laufwerk angeben, arbeitet **COMPILE** mit dem momentanen Defaultlaufwerk.

Beim COMPILE-Aufruf können Sie als Option die Sprachkennung angeben. Möglich sind D (deutsch), E (englisch) und F (französisch). Die Defaulteinstellung ist D (deutsch).

Als Trennzeichen zwischen den Argumenten sind Tabulator- und Leerzeichen sowie Kommata zugelassen.

Kommandozeilen-Version

4.1 Aufruf

4.1.1 Aufruf ohne Parameterangabe:

Syntax: COMPILE <QUELLE> <ZIEL> <SPRACHKENNUNG>

Beispiel: COMPILE @@@@@@A0.SEQ @@@@@@ST.S5D D

Folgende sechs Kombinationen für Quelle und Ziel sind möglich:

AWL-Quelldatei --> Programmdatei: nnnnnnA0.SEQ nnnnnnST.S5D
AWL-Quelldatei --> Zwischendatei: nnnnnnA0.SEQ nnnnnnA1.SEQ
Zwischendatei --> Programmdatei: nnnnnnA1.SEQ nnnnnnST.S5D
Zwischendatei --> AWL-Quelldatei: nnnnnnA1.SEQ nnnnnnA0.SEQ
Programmdatei --> AWL-Quelldatei: nnnnnnST.S5D nnnnnnA0.SEQ
Programmdatei --> Zwischendatei: nnnnnnST.S5D nnnnnnA1.SEQ

4

Die Dateinamen müssen den S5-Konventionen entsprechen, d.h., nach den sechs wahlfreien Zeichen des Namens folgen die zugelassenen Dateikennungen (A0.SEQ, A1.SEQ, ST.S5D). Es wird nicht zwischen Klein- und Großschreibung unterschieden. Wenn Sie kein Laufwerk angeben, arbeitet **COMPILE** mit dem momentanen Defaultlaufwerk.

Beim COMPILE-Aufruf können Sie als Option die Sprachkennung angeben. Möglich sind D (deutsch), E (englisch) und F (französisch). Die Defaulteinstellung ist D (deutsch).

Als Trennzeichen zwischen den Argumenten sind Tabulator- und Leerzeichen sowie Kommata zugelassen.

C79000-B8500-C677-02 4 - 3

4.1.2 Aufruf mit einer Input-Datei

Syntax:	COMPILE	#<INPUT-DATEI NAME>	<SPRACHKENNUNG>
Beispiel:	COMPILE	#A:INPTEST1.INP	D

Der Name der Inputdatei muß den CPM-Konventionen entsprechen, d.h. der Dateiname kann aus bis zu acht und der Dateityp aus max. drei Zeichen (Buchstaben oder Ziffern) bestehen. Geben Sie kein Laufwerk oder keine Sprachkennung an, wird das Defaultlaufwerk bzw. D (deutsch) als Sprachkennung verwendet.

Um eine Datei als Inputdatei zu kennzeichnen, ist das #-Zeichen vor dem Dateinamen zwingend erforderlich. Andernfalls bricht **COMPILE** mit einer Fehlermeldung ab.

Kommandozeilen-Version

4.1.2 Aufruf mit einer Input-Datei

Syntax: COMPILE #<INPUT-DATEI NAME> <SPRACHKENNUNG>
Beispiel: COMPILE #A:INPTEST1.INP D

Der Name der Inputdatei muß den CPM-Konventionen entsprechen, d.h. der Dateiname kann aus bis zu acht und der Dateityp aus max. drei Zeichen (Buchstaben oder Ziffern) bestehen. Geben Sie kein Laufwerk oder keine Sprachkennung an, wird das Defaultlaufwerk bzw. D (deutsch) als Sprachkennung verwendet.

Um eine Datei als Inputdatei zu kennzeichnen, ist das #-Zeichen vor dem Dateinamen zwingend erforderlich. Andernfalls bricht **COMPILE** mit einer Fehlermeldung ab.

4 - 4 C79000-B8500-C677-02

4.2 Format der Input-Datei

Die Inputdatei enthält eine Liste der zu bearbeitenden Dateien und der gewünschten Einstellungen. Alle Einstellungen sind identisch mit den Parametern, die in der Kommandozeile der einzelnen Übersetzungsfunktionen des Pakets AWL-Batch-Compiler eingegeben werden können. Einstellungen, die nach der Angabe von Quell- und Zieldatei folgen, bleiben solange erhalten, bis sie explizit geändert werden. Für nicht angegebene Parameter wird die Defaulteinstellung übernommen (siehe Abschnitt4.3). Nach jedem Parameterblock können Sie wieder eine neue Quell- und Zieldatei mit neuen Parametern angeben.

Syntax:

<QUELLDATEI 1> <ZIELDATEI 1>
$BAUST: <BAUSTEINANGABE>
$SYMB: <SYMBOLIK-DATEINAME>
$OPT: <OPTIONEN>
$AGTYP: <SPRACHRAUMBEZEICHNUNG>
$DRU: <OPTION> <DRUCKERDATEINAME>

<QUELLDATEI 2> <ZIELDATEI 2>

$..
$..

Beispiel:

```
TEST1@A0.SEQ TEST1@ST.S5D
$BAUST:          FB3-10
$SYMB:       SYMB23Z0.INI
$OPT:                  1
$AGTYP:          AG139W
$DRU:        *_B:DRUCK1DR.INI

PRUEF2ST.S5D TEST11A1.SEQ
$BAUST:                B
$OPT:                  2
```


Kommandozeilen-Version

4.2 Format der Input-Datei

Die Inputdatei enthält eine Liste der zu bearbeitenden Dateien und der gewünschten Einstellungen. Alle Einstellungen sind identisch mit den Parametern, die in der Kommandozeile der einzelnen Übersetzungsfunktionen des Pakets AWL-Batch-Compiler eingegeben werden können. Einstellungen, die nach der Angabe von Quell- und Zieldatei folgen, bleiben solange erhalten, bis sie explizit geändert werden. Für nicht angegebene Parameter wird die Defaulteinstellung übernommen (siehe Abschnitt4.3). Nach jedem Parameterblock können Sie wieder eine neue Quell- und Zieldatei mit neuen Parametern angeben.

Syntax:

<QUELLDATEI 1> <ZIELDATEI 1>
$BAUST: <BAUSTEINANGABE>
$SYMB: <SYMBOLIK-DATEINAME>
$OPT: <OPTIONEN>
$AGTYP: <SPRACHRAUMBEZEICHNUNG>
$DRU: <OPTION> <DRUCKERDATEINAME>

<QUELLDATEI 2> <ZIELDATEI 2>

$..
$..

Beispiel:

TEST1@A0.SEQ TEST1@ST.S5D
$BAUST: FB3-10
$SYMB: SYMB23Z0.INI
$OPT: 1
$AGTYP: AG139W
$DRU: *_B:DRUCK1DR.INI

PRUEF2ST.S5D TEST11A1.SEQ
$BAUST: B
$OPT: 2

4.2.1
Quell- und Zieldateiangabe

In der ersten Zeile der Inputdatei müssen Sie die Quell- und Zieldatei eintragen. Format und Kombinationsmöglichkeiten sind identisch mit dem **COMPILE**-Aufruf ohne Parameter (siehe Abschnitt 4.1). Allerdings dürfen Sie hier keine Sprachkennung angeben, da diese gegebenenfalls bereits in der Aufrufzeile hinter dem Namen der Inputdatei steht.

4.2.2
Parameter $BAUST

Bei allen Übersetzungen außer AWL-Quelldatei in Zwischendatei und umgekehrt können Sie folgende Bausteinabgaben parametrieren:

OBn, FBn, PBn:	Übersetzen von Einzelbausteinen
*:	Übersetzen mehrerer Einzelbausteine
OBn-m, FBn-m:	Übersetzen von Bausteinbereichen
OB, FB, PB:	Übersetzen von Bausteinarten
B:	Übersetzen aller Bausteine (Voreinstellung)

(siehe Abschnitt 3.4.1)

4.2.3
Parameter $SYMB

Mit dieser Option können Sie der Übersetzung den Namen der Symbolikdatei mitteilen. Dies entspricht im Paket AWL-Batch-Compiler dem Eintrag der Symbolikdatei in der Einstellungsmaske. Der Dateiname darf maximal 6 Zeichen und die Laufwerksangabe umfassen, da die Erweiterung Z0.SEQ automatisch angehängt wird. Voreingestellt ist der Name der AWL-Quell- bzw. Zwischendatei.

Kommandozeilen-Version

4.2.1
Quell- und Zieldateiangabe

In der ersten Zeile der Inputdatei müssen Sie die Quell- und Zieldatei eintragen. Format und Kombinationsmöglichkeiten sind identisch mit dem **COMPILE**-Aufruf ohne Parameter (siehe Abschnitt 4.1). Allerdings dürfen Sie hier keine Sprachkennung angeben, da diese gegebenenfalls bereits in der Aufrufzeile hinter dem Namen der Inputdatei steht.

4.2.2
Parameter $BAUST

Bei allen Übersetzungen außer AWL-Quelldatei in Zwischendatei und umgekehrt können Sie folgende Bausteinabgaben parametrieren:

OBn, FBn, PBn: Übersetzen von Einzelbausteinen
*: Übersetzen mehrerer Einzelbausteine
OBn-m, FBn-m: Übersetzen von Bausteinbereichen
OB, FB, PB: Übersetzen von Bausteinarten
B: Übersetzen aller Bausteine (Voreinstellung)

(siehe Abschnitt 3.4.1)

4.2.3
Parameter $SYMB

Mit dieser Option können Sie der Übersetzung den Namen der Symbolikdatei mitteilen. Dies entspricht im Paket AWL-Batch-Compiler dem Eintrag der Symbolikdatei in der Einstellungsmaske. Der Dateiname darf maximal 6 Zeichen und die Laufwerksangabe umfassen, da die Erweiterung Z0.SEQ automatisch angehängt wird. Voreingestellt ist der Name der AWL-Quell- bzw. Zwischendatei.

4.2.4
Parameter $OPT

Bei der Übersetzung von der AWL-Quelldatei in die Programmdatei bzw. von der Zwischendatei in die Programmdatei sind folgende Optionen möglich:

LEER: MC5-Code erzeugen und vor Überschreiben fragen

1: MC5-Code erzeugen und Überschreiben ohne zu fragen

2: Keinen MC5-Code erzeugen, nur Testlauf

(siehe Abschnitt 3.4.1)

Bei der Übersetzung von der Programmdatei in die AWL-Quelldatei bzw. von der Programmdatei in die Zwischendatei sind folgende Optionen möglich:

LEER: Nur Symbole in die AWL-Quelldatei eintragen

1: Symbole und Absolutparameter in die AWL-Quelldatei eintragen

2: Nur Absolutparameter in die AWL-Quelldatei eintragen

(siehe Abschnitt 3.4.2)

4.2.5
Parameter $AGTYP

Dieser Parameter legt bei der Übersetzung den Sprachraum, d.h. den Befehlsvorrat für den entsprechenden AG-Typ, fest. Die Voreinstellung ist NEIN.

(siehe Abschnitt 2.2.3)

4.2.6
Parameter $DRU

Die Angabe einer Druckerdatei ist nur gültig für die Übersetzung von der AWL-Quelldatei oder der Zwischendatei in die Programmdatei. Der Parameter besteht aus zwei Teilen, und zwar erstens aus der Option und zweitens aus dem Namen der Druckerdatei. Die Option muß immer vor dem Dateinamen stehen und bietet folgende drei Möglichkeiten:

*: Standard-Druckausgabe

1: Druckausgabe in Normalschrift

2: Druckausgabe in Schmalschrift (mit Heftrand)

Die Voreinstellung ist LEER (kein Listing);

(siehe Abschnitt 3.4.1)

Kommandozeilen-Version

4.2.4
Parameter $OPT

Bei der Übersetzung von der AWL-Quelldatei in die Programmdatei bzw. von der Zwischendatei in die Programmdatei sind folgende Optionen möglich:

LEER: MC5-Code erzeugen und vor Überschreiben fragen

1: MC5-Code erzeugen und Überschreiben ohne zu fragen

2: Keinen MC5-Code erzeugen, nur Testlauf

(siehe Abschnitt 3.4.1)

Bei der Übersetzung von der Programmdatei in die AWL-Quelldatei bzw. von der Programmdatei in die Zwischendatei sind folgende Optionen möglich:

LEER: Nur Symbole in die AWL-Quelldatei eintragen

1: Symbole und Absolutparameter in die AWL-Quelldatei eintragen

2: Nur Absolutparameter in die AWL-Quelldatei eintragen

(siehe Abschnitt 3.4.2)

4.2.5
Parameter $AGTYP

Dieser Parameter legt bei der Übersetzung den Sprachraum, d.h. den Befehlsvorrat für den entsprechenden AG-Typ, fest. Die Voreinstellung ist NEIN.

(siehe Abschnitt 2.2.3)

4.2.6
Parameter $DRU

Die Angabe einer Druckerdatei ist nur gültig für die Übersetzung von der AWL-Quelldatei oder der Zwischendatei in die Programmdatei. Der Parameter besteht aus zwei Teilen, und zwar erstens aus der Option und zweitens aus dem Namen der Druckerdatei. Die Option muß immer vor dem Dateinamen stehen und bietet folgende drei Möglichkeiten:

*: Standard-Druckausgabe

1: Druckausgabe in Normalschrift

2: Druckausgabe in Schmalschrift (mit Heftrand)

Die Voreinstellung ist LEER (kein Listing);

(siehe Abschnitt 3.4.1)

C79000-B8500-C677-02

4 - 7

4.3 Default-Einstellungen der Parameter

COMPILE.CMD verwendet folgende Default-Einstellungen:

$BAUST:	B	alle Bausteine
$OPT:		
SEQ>MC5 and ZWI>MC5:	1	
		Code erzeugen und Datei überschreiben ohne fragen
MC5>SEQ and MC5>ZWI:	LEER	
		nur Symbole eintragen
$DRU:	LEER	kein Listing
$AGTYP:	NEIN	Sprachraum NEIN

Kommandozeilen-Version

4.3 Default-Einstellungen der Parameter

COMPILE.CMD verwendet folgende Default-Einstellungen:

$BAUST:	B	alle Bausteine
$OPT:		
SEQ>MC5 and ZWI>MC5:	1	
		Code erzeugen und Datei überschreiben ohne fragen
MC5>SEQ and MC5>ZWI:	LEER	
		nur Symbole eintragen
$DRU:	LEER	kein Listing
$AGTYP:	NEIN	Sprachraum NEIN

Anhang 5

Anhang 5

C79000-B8500-C677-02 5 - 1

Fehlermeldungen

* AG-Typ unzulässig:
Angabe eines ungültigen AG-Typs.

* Absolutparameter zu lang:
Systemfehler! Format der AWL-Quelldatei fehlerhaft.

* Aktualparameter nicht erlaubt:
Die Angabe von Aktualparametern ist nur nach einem FB-Aufruf erlaubt.

* BIB-Nr. schon vorhanden:
Das Steuerzeichen #BI ist mehrmals angegeben.

* BIB-Nr. ungültig:
Bibliotheksnummer zu lang, oder unzulässige Zeichen enthalten (max. 5 Ziffern).

* Baustein in Zwischendatei nicht fehlerfrei:
Zwischendatei (A1.SEQ) ist defekt (Formatfehler).
Zwischendatei mit der Funktion **SEQ>ZWI** nochmals aus der sequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Baustein ohne BE:
Befehl BE (Bausteinendekennzeichen) fehlt.

* Baustein zu lang:
Programm aufteilen (max. 8 kByte).

* Bausteinanfang fehlt:
Kein #-Zeichen mit absoluter- und/oder symbolischer Bausteinbezeichnung vorhanden.

Fehlermeldungen

Fehlermeldungen

* AG-Typ unzulässig:
Angabe eines ungültigen AG-Typs.

* Absolutparameter zu lang:
Systemfehler! Format der AWL-Quelldatei fehlerhaft.

* Aktualparameter nicht erlaubt:
Die Angabe von Aktualparametern ist nur nach einem FB-Aufruf erlaubt.

* BIB-Nr. schon vorhanden:
Das Steuerzeichen #BI ist mehrmals angegeben.

* BIB-Nr. ungültig:
Bibliotheksnummer zu lang, oder unzulässige Zeichen enthalten (max. 5 Ziffern).

* Baustein in Zwischendatei nicht fehlerfrei:
Zwischendatei (A1.SEQ) ist defekt (Formatfehler).
Zwischendatei mit der Funktion **SEQ>ZWI** nochmals aus der sequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Baustein ohne BE:
Befehl BE (Bausteinendekennzeichen) fehlt.

* Baustein zu lang:
Programm aufteilen (max. 8 kByte).

* Bausteinanfang fehlt:
Kein #-Zeichen mit absoluter- und/oder symbolischer Bausteinbezeichnung vorhanden.

5 - 2 C79000-B8500-C677-02

* Bausteinart unbestimmt (Symbol nicht gefunden):
Bei rein symbolischer Programmierung fehlt die symbolische Bausteinbezeichnung.

* Bausteinname schon vorhanden:
Das Steuerzeichen #N ist mehrmals angegeben.

* Befehl für AG-Typ nicht zulässig:
Befehl für angegebenen AG-Typ nicht erlaubt.

* Befehl im Baustein nicht erlaubt:
Befehle aus dem "ergänzenden Operationsvorrat" sind nur in FBs erlaubt.

* Befehl nicht definiert:
Kein zulässiger MC5-Befehl. Programmdatei (ST.S5D) defekt.

* Befehl nicht erlaubt:
Kein zulässiger STEP 5-Befehl.

* DB-Adresse ungültig:
DB-Adresse zu lang, oder unzulässige Zeichen enthalten (max. 5 Zeichen).

* Datenposition falsch:
Bei Konstanten ist der Wert im Feld "OPERANDEN-SYMBOL" einzugeben.

* Dok-Baustein zu lang:
Programmdokumentation aufteilen oder kürzen (max. 16 kByte).

Fehlermeldungen

* Bausteinart unbestimmt (Symbol nicht gefunden):
Bei rein symbolischer Programmierung fehlt die symbolische Bausteinbezeichnung.

* Bausteinname schon vorhanden:
Das Steuerzeichen #N ist mehrmals angegeben.

* Befehl für AG-Typ nicht zulässig:
Befehl für angegebenen AG-Typ nicht erlaubt.

* Befehl im Baustein nicht erlaubt:
Befehle aus dem "ergänzenden Operationsvorrat" sind nur in FBs erlaubt.

* Befehl nicht definiert:
Kein zulässiger MC5-Befehl. Programmdatei (ST.S5D) defekt.

* Befehl nicht erlaubt:
Kein zulässiger STEP 5-Befehl.

* DB-Adresse ungültig:
DB-Adresse zu lang, oder unzulässige Zeichen enthalten (max. 5 Zeichen).

* Datenposition falsch:
Bei Konstanten ist der Wert im Feld "OPERANDEN-SYMBOL" einzugeben.

* Dok-Baustein zu lang:
Programmdokumentation aufteilen oder kürzen (max. 16 kByte).

C79000-B8500-C677-02

5 - 3

* DVS-Datei nicht eröffnet

* Falsche Funktionsnummer:
Systemfehler!

* Falsche Klammerungstiefe:
Klammerungs-Abschluß nicht ausgeglichen (Klammerebenen beachten).

* Falscher Datensatz in SEQ-Datei:
Sequentielle Arbeitsdatei AWL-Quelldatei oder Zwischendatei defekt (Formatfehler).

* Falsches Format:
Fehlerbehaftetes Format

* Fehler beim Umsetzen:
Systemfehler!

* Fehler beim Umwandeln:
Zahlenbereich wurde überschritten.

* Formalparameter nicht definiert:
Parametername und Parameterart sind im FB nicht definiert.

* Formalparameter schon vorhanden:
Der Parametername ist mehrfach vergeben.

* Formalparameter ungültig:
Im Parameternamen sind ungültige Zeichen enthalten, oder die Parameterart ist nicht zulässig.

Fehlermeldungen

* DVS-Datei nicht eröffnet

* Falsche Funktionsnummer:
Systemfehler!

* Falsche Klammerungstiefe:
Klammerungs-Abschluß nicht ausgeglichen (Klammerebenen beachten).

* Falscher Datensatz in SEQ-Datei:
Sequentielle Arbeitsdatei AWL-Quelldatei oder Zwischendatei defekt (Formatfehler).

* Falsches Format:
Fehlerbehaftetes Format

* Fehler beim Umsetzen:
Systemfehler!

* Fehler beim Umwandeln:
Zahlenbereich wurde überschritten.

* Formalparameter nicht definiert:
Parametername und Parameterart sind im FB nicht definiert.

* Formalparameter schon vorhanden:
Der Parametername ist mehrfach vergeben.

* Formalparameter ungültig:
Im Parameternamen sind ungültige Zeichen enthalten, oder die Parameterart ist nicht zulässig.

5 - 4

C79000-B8500-C677-02

* Kein Aktualparameter angegeben:
Nach dem FB-Aufruf fehlt im Feld "ANWEISUNG" die benötigte Aktualparameter-Angabe.

* Kein Bausteinname angegeben:
Im Funktionsbaustein fehlt Steuerzeichen #N und der Bausteinname.

* Kein Datum angegeben:
Nach Angabe eines Konstanten-Typ im Feld "ANWEISUNG" fehlt die Konstanten-Wert-Angabe im Feld "OPERANDENSYMBOL".

* Kein Formaloperand angegeben:
Zum angegebenen Aktualparameter nach dem FB-Aufruf fehlt die zugeordnete Formalparameterdeklaration im Feld "ADR" des FB.

* Kein Operandenkennzeichen angegeben:
Operandenkennzeichen fehlt.

* Kein Parameter angegeben:
Parameterangabe fehlt (bei rein absoluter Programmierung).

* Kein Symbol angegeben:
Operandensymbol fehlt (bei rein symbolischer Programmierung).

* Keine BIB-Nummer angegeben:
Nach #BI fehlt die Angabe der Bibliotheksnummer.

* Kommentar zu lang:
Systemfehler! Format der AWL-Quelldatei fehlerhaft.
Anweisungskommentar zu lang (max. 32 Zeichen).

Fehlermeldungen

* Kein Aktualparameter angegeben:
Nach dem FB-Aufruf fehlt im Feld "ANWEISUNG" die benötigte Aktualparameter-Angabe.

* Kein Bausteinname angegeben:
Im Funktionsbaustein fehlt Steuerzeichen #N und der Bausteinname.

* Kein Datum angegeben:
Nach Angabe eines Konstanten-Typ im Feld "ANWEISUNG" fehlt die Konstanten-Wert-Angabe im Feld "OPERANDENSYMBOL".

* Kein Formaloperand angegeben:
Zum angegebenen Aktualparameter nach dem FB-Aufruf fehlt die zugeordnete Formalparameterdeklaration im Feld "ADR" des FB.

* Kein Operandenkennzeichen angegeben:
Operandenkennzeichen fehlt.

* Kein Parameter angegeben:
Parameterangabe fehlt (bei rein absoluter Programmierung).

* Kein Symbol angegeben:
Operandensymbol fehlt (bei rein symbolischer Programmierung).

* Keine BIB-Nummer angegeben:
Nach #BI fehlt die Angabe der Bibliotheksnummer.

* Kommentar zu lang:
Systemfehler! Format der AWL-Quelldatei fehlerhaft.
Anweisungskommentar zu lang (max. 32 Zeichen).

* Kommentarbaustein zu lang:
Programm aufteilen oder kürzen (max. 16 kByte).

* Lesefehler:
Diskettenfehler, Datei defekt.

* Marke nicht definiert:
Sprungmarke zum angegebenen Sprungziel (Symboladresse) im Feld "ADR" nicht eingetragen.

* Marke ungültig:
Sprungmarke mit nichtzulässigen Zeichen

* Marke unzulässig:
Sprungmarke an unzulässiger Position

* Marke zu lang:
Im Feld "ANWEISUNG" eingetragenes Sprungziel (Symboladresse) zu lang (max. 4 Zeichen).

* Marken sind gleich:
Sprungmarke mehrfach vorhanden.

* Mehr Kommentar als Anweisungen

* Netzwerkende fehlt oder Netzwerk ist zu lang:
Netzwerkende-Zeichen *** oder Bildaufbaubefehl für Segmentende (BLD 255) fehlt, oder Netzwerk ist zu lang (max. 255 Zeilen).

* Nur bei AWL-Bausteinen erlaubt:
Datenbaustein

Fehlermeldungen

* Kommentarbaustein zu lang:
Programm aufteilen oder kürzen (max. 16 kByte).

* Lesefehler:
Diskettenfehler, Datei defekt.

* Marke nicht definiert:
Sprungmarke zum angegebenen Sprungziel (Symboladresse) im Feld "ADR" nicht eingetragen.

* Marke ungültig:
Sprungmarke mit nichtzulässigen Zeichen

* Marke unzulässig:
Sprungmarke an unzulässiger Position

* Marke zu lang:
Im Feld "ANWEISUNG" eingetragenes Sprungziel (Symboladresse) zu lang (max. 4 Zeichen).

* Marken sind gleich:
Sprungmarke mehrfach vorhanden.

* Mehr Kommentar als Anweisungen

* Netzwerkende fehlt oder Netzwerk ist zu lang:
Netzwerkende-Zeichen *** oder Bildaufbaubefehl für Segmentende (BLD 255) fehlt, oder Netzwerk ist zu lang (max. 255 Zeilen).

* Nur bei AWL-Bausteinen erlaubt:
Datenbaustein

5 - 6 C79000-B8500-C677-02

* Nur bei Funktionsbausteinen:
Erweiterter Befehlssatz nicht erlaubt

* Nur eine Überschrift pro Netzwerk:
Das Steuerzeichen #UB ist am Netzwerkanfang mehrmals angegeben.

* Nur nach FB-Aufruf:
Aktualparameter sind nur unmittelbar nach einem FB-Aufruf zulässig.

* Operand unzulässig:
Kein Operand erlaubt.

* Operand zu lang:
Operandenkennzeichen zu lang (max. 2 Zeichen).

* Operandenkennzeichen nicht definiert:
Operandenkennzeichen in STEP 5 nicht definiert.

* Operandenkennzeichen unzulässig:
Operandenkennzeichen paßt nicht zum Operator.

* Operator nicht angegeben:
Fehlende Operatorangabe bei symbolischer Programmierung.

* Operator ungültig:
Operator in STEP 5 nicht definiert.

* Operator zu lang:
(max. 3 Zeichen)

Fehlermeldungen

* Nur bei Funktionsbausteinen:
Erweiterter Befehlssatz nicht erlaubt

* Nur eine Überschrift pro Netzwerk:
Das Steuerzeichen #UB ist am Netzwerkanfang mehrmals angegeben.

* Nur nach FB-Aufruf:
Aktualparameter sind nur unmittelbar nach einem FB-Aufruf zulässig.

* Operand unzulässig:
Kein Operand erlaubt.

* Operand zu lang:
Operandenkennzeichen zu lang (max. 2 Zeichen).

* Operandenkennzeichen nicht definiert:
Operandenkennzeichen in STEP 5 nicht definiert.

* Operandenkennzeichen unzulässig:
Operandenkennzeichen paßt nicht zum Operator.

* Operator nicht angegeben:
Fehlende Operatorangabe bei symbolischer Programmierung.

* Operator ungültig:
Operator in STEP 5 nicht definiert.

* Operator zu lang:
(max. 3 Zeichen)

C79000-B8500-C677-02

5 - 7

* Parameter fehlerhaft:
Ungültiger Parameter.

* Parameter unzulässig:
Kein Parameter erlaubt.

* Parameter zu lang (max. 4 Zeichen):
Im Feld "ANWEISUNG" eingetragener Formalparameter zu lang (max. 4 Zeichen).

* Parameterzahl falsch:
Deklarierte Formalparameter-Anzahl in FB ist ungleich Anzahl angegebener Aktualparamter nach FB-Aufruf. (Prüflauf)

* Parameterbereich des AG-Typs überschritten:
Beim angegebenen AG-Typ ist dieser Parameterwert nicht erlaubt.

* Parametertyp falsch:
Im FB im Feld "ANWEISUNG" angegebene Formal-parameterart ist ungleich der zugeordneten Aktual-parameterart nach FB-Aufruf. (Prüflauf)

* SYS-Befehl nicht erlaubt:
entfällt

* Steuerzeichen ungültig:
Auf das #Zeichen folgt ein nicht erlaubtes Steuerzeichen.

* Symbol nicht erlaubt:
Befehl erlaubt keine Operandenangabe.

Fehlermeldungen

* Parameter fehlerhaft:
Ungültiger Parameter.

* Parameter unzulässig:
Kein Parameter erlaubt.

* Parameter zu lang (max. 4 Zeichen):
Im Feld "ANWEISUNG" eingetragener Formalparameter zu lang (max. 4 Zeichen).

* Parameterzahl falsch:
Deklarierte Formalparameter-Anzahl in FB ist ungleich Anzahl angegebener Aktualparamter nach FB-Aufruf. (Prüflauf)

* Parameterbereich des AG-Typs überschritten:
Beim angegebenen AG-Typ ist dieser Parameterwert nicht erlaubt.

* Parametertyp falsch:
Im FB im Feld "ANWEISUNG" angegebene Formal-parameterart ist ungleich der zugeordneten Aktual-parameterart nach FB-Aufruf. (Prüflauf)

* SYS-Befehl nicht erlaubt:
entfällt

* Steuerzeichen ungültig:
Auf das #Zeichen folgt ein nicht erlaubtes Steuerzeichen.

* Symbol nicht erlaubt:
Befehl erlaubt keine Operandenangabe.

5 - 8 C79000-B8500-C677-02

* Symbol paßt nicht zu Absolutparameter:
Absolutoperand und Symboloperand sind in AWL-Quelldatei und Symbolik-Datei unterschiedlich zugeordnet.

* Symbol zu lang:
Systemfehler! Format der AWL-Quelldatei fehlerhaft.
(max. 24 Zeichen)

* Symbolikdatei nicht vorhanden:
Symbolikdatei fehlt bei rein symbolischer Programmierung.

* Systembefehle nicht erlaubt:
entfällt

* Zeichen ungültig:
ungültiges Zeichen vorhanden

* Zeile nicht erlaubt:
Reihenfolge (Steuerzeichen) bei der Bausteineingabe beachten.

* Zeile ungültig:
Reihenfolge (Steuerzeichen) bei der Bausteineingabe beachten.

* Zeile wurde nicht bearbeitet:
Bausteinart ist unbestimmt.

* Zu viele Aktualparameter:
(max. 40)

* Zu viele Formalparameter:
(max. 40)

Fehlermeldungen

* Symbol paßt nicht zu Absolutparameter:
Absolutoperand und Symboloperand sind in AWL-Quelldatei und Symbolik-Datei unterschiedlich zugeordnet.

* Symbol zu lang:
Systemfehler! Format der AWL-Quelldatei fehlerhaft.
(max. 24 Zeichen)

* Symbolikdatei nicht vorhanden:
Symbolikdatei fehlt bei rein symbolischer Programmierung.

* Systembefehle nicht erlaubt:
entfällt

* Zeichen ungültig:
ungültiges Zeichen vorhanden

* Zeile nicht erlaubt:
Reihenfolge (Steuerzeichen) bei der Bausteineingabe beachten.

* Zeile ungültig:
Reihenfolge (Steuerzeichen) bei der Bausteineingabe beachten.

* Zeile wurde nicht bearbeitet:
Bausteinart ist unbestimmt.

* Zu viele Aktualparameter:
(max. 40)

* Zu viele Formalparameter:
(max. 40)

* Zwischendatei existiert bereits, löschen?:
Eine Zwischendatei mit identischem Dateinamen ist bereits vorhanden.

* Zwischendatei nicht fehlerfrei:
Zwischendatei (A1.SEQ) ist defekt (Formatfehler).
Zwischendatei mit der Funktion **SEQ>ZWI** nochmals aus der sequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Zwischendateiformat ungültig:
Zwischencode-Datei (A1.SEQ) ist defekt (Formatfehler).
Zwischencode-Datei mit der Funktion **SEQ>ZWI** nochmals aus dersequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Zwischendateikennung falsch:
Datei wurde mit Werkzeugen eines anderen Ausgabestandeserzeugt. Zwischendatei mit der Funktion **SEQ>ZWI** aus der sequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Geschachtelte Includeanweisung nicht erlaubt
Eine Zwischendatei, die mit #INCLUDE eingeschlossen wurde, enthält wiederum eine #INCLUDE-Anweisung.

* Symbolik-Quelldatei vorhanden, ueberschreiben?
Eine Symbolik-Quelldatei mit identischem Dateinamen ist bereits vorhanden (Funktion SYM-GEN).

Fehlermeldungen

* Zwischendatei existiert bereits, löschen?:
Eine Zwischendatei mit identischem Dateinamen ist bereits vorhanden.

* Zwischendatei nicht fehlerfrei:
Zwischendatei (A1.SEQ) ist defekt (Formatfehler).
Zwischendatei mit der Funktion **SEQ>ZWI** nochmals aus der sequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Zwischendateiformat ungültig:
Zwischencode-Datei (A1.SEQ) ist defekt (Formatfehler).
Zwischencode-Datei mit der Funktion **SEQ>ZWI** nochmals aus dersequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Zwischendateikennung falsch:
Datei wurde mit Werkzeugen eines anderen Ausgabestandeserzeugt. Zwischendatei mit der Funktion **SEQ>ZWI** aus der sequentiellen Arbeitsdatei (A0.SEQ) neu generieren.

* Geschachtelte Includeanweisung nicht erlaubt
Eine Zwischendatei, die mit #INCLUDE eingeschlossen wurde, enthält wiederum eine #INCLUDE-Anweisung.

* Symbolik-Quelldatei vorhanden, ueberschreiben?
Eine Symbolik-Quelldatei mit identischem Dateinamen ist bereits vorhanden (Funktion SYM-GEN).

Trennblatt

C79000-D8300-C018-01

1

Trennblatt

2

C79000-D8300-C018-01

SIEMENS

STL Editor/ Batch-Compiler

Description

C79000-G8576-C677

Release 02

SIEMENS

STL Editor/ Batch-Compiler

Description

Introduction 1

Description 2

Operation of the Programming Unit 3

Command Line Version 4

Appendix 5

C79000-G8576-C677

Release 02

Siemens Aktiengesellschaft

C79000-G8576-C677-01
EWK Elektronikwerk Karlsruhe

Printed in the Federal Republic of Germany

Siemens Aktiengesellschaft

C79000-G8576-C677-01
EWK Elektronikwerk Karlsruhe

Printed in the Federal Republic of Germany

Contents

Contents

Contents

1 Introduction 1-1

2 Description 2-1

2.1 Functioning of the STL Editor/Batch Compiler 2-2

2.2 Generating STEP 5 Blocks 2-7

2.2.1 In General 2-7
2.2.2 Editing Functions 2-9
2.2.3 Control Characters 2-10
2.2.4 Compilation 2-13
2.2.5 Printing 2-15

2.3 The A1.SEQ Intermediate File 2-17

2.3.1 Relations between the STL Source File and the Intermediate File ... 2-17
2.3.2 Special Functions 2-19
2.3.3 Standard Programs 2-20
2.3.4 Foreign Language STEP 5 Program Versions 2-21

2.4 Editing and Supplementing STEP 5 Blocks 2-22

2.4.1 Blocks Created Using the STL Editor 2-22
2.4.2 Blocks Created Using the LAD, CSF, STL Package 2-22

2.5 Test Run 2-23

2.5.1 Program File Testing 2-23
2.5.2 Testing of Special Blocks 2-23

C79000-B8576-C677-02

0 - 3

Contents

2.6 Error List ... 2-24

2.6.1 Error File ... 2-24

2.7 Entering STEP 5 Statements with other Editors ... 2-25

2.7.1 STL Source File as an Interface ... 2-25
2.7.2 Sequential Source File Format ... 2-25
2.7.3 #TAB Control Characters for Processing of External Files ... 2-26

3 Operation of theProgramming Unit ... 3-1

3.1 Software Installation ... 3-2

3.2 Operational Steps up to the Function Selection ... 3-3

3.2.1 The Operation in General ... 3-3
3.2.2 Entry into the STL Editor/Batch Compiler Package ... 3-3

3.3 Edit ... 3-8

3.3.1 Editor Description ... 3-9
3.3.2 The STL Editor/Batch Compiler Control Characters and their Writing Conventions ... 3-14
3.3.3 STEP 5 Operations in the STL Editor/Batch Compiler and their Writing Conventions ... 3-20
3.3.4 Entry of Program Blocks ... 3-25
3.3.5 Using the EDIT softkeys ... 3-30
3.3.6 Input of Function Blocks ... 3-41
3.3.7 Data Block Input ... 3-48
3.3.8 Editing an STL Source File ... 3-51

Contents

3.4 Compiling with the COMPILER Function 3-52

3.4.1 Operating Sequence: Compiling into the Program File 3-52
3.4.2 Operating Sequence: Recompiling the Program File 3-54

3.5 Error List 3-55

3.6 Printing 3-56

3.7 SPECIAL Functions for Editing Intermediate Files and Source Files 3-57

3.7.1 COPYing 3-57
3.7.2 SEQ>INT 3-58
3.7.3 INT>SEQ 3-58
3.7.4 SEQ DEL and INT DEL 3-59
3.7.5 TEST RUN 3-59
3.7.6 SYM-GEN 3-60

Contents

4 Command Line Version 4-1

4.1 Calling 4-3

4.1.1 Calling without Parameter Entry 4-3
4.1.2 Calling with an Input File 4-4

4.2 Input File Format 4-5

4.2.1 Entry of Source and Target File 4-6
4.2.2 $BLOCK Parameter 4-6
4.2.3 $SYMB Parameter 4-6
4.2.4 $OPT: Parameter 4-7
4.2.5 $PCTYPE: Parameter 4-7
4.2.6 $PRT Parameter 4-7

4.3 Parameter Default Values 4-8

5 Appendix 5-1

5.1 **Error Messages 5-2**

0 - 6

C79000-B8576-C677-02

Introduction 1

Introduction 1

The STL editor/batch compiler option package has an independent editor for Programing in the STL representation mode and an independent compiler for the compilation of such statement lists into a runnable STEP 5 program.

Up to now a STEP 5 program could only be created within the LAD, CSF, STL package. However, the ease of editing with this package is restricted and the wiring of the system must be known ahead of time in order to properly set up the inputs/outputs.

By means of the STL editor your program can now be created very easily, e.g. copying segments. This package can also be used to create a program symbolically. That means, you do not initially have to have a definite assignment of symbols and inputs/outputs, but you can supply these assignments later, e.g. when your system is finished. These assignments are then connected with the statement list created in the editor by the batch compiler and are compiled into a STEP 5 program, which runs on your system. You can then test and correct this program in the programmable controller.

The batch compiler also carries out recompilations from a STEP 5 program so that e.g. the alterations of the tested program can be entered into your source and the statement list updated. A STEP 5 program, which has been created in the LAD, CSF, STL package, can be handled in a same manner.

That means, the STL editor/batch compiler option package provides many advantages:

1. *You can create a function specific program with symbols which becomes a system specific STEP 5 program (due to individual assignements); i.e. you can create a library with standardized programs.*
2. *Standardized programs and modules can be re-combined again and again using the Include command and can be connected to an individual STEP 5 program, without needing to carry out extensive alterations in the programs.*

Introduction

The STL editor/batch compiler option package has an independent editor for Programing in the STL representation mode and an independent compiler for the compilation of such statement lists into a runnable STEP 5 program.

Up to now a STEP 5 program could only be created within the LAD, CSF, STL package. However, the ease of editing with this package is restricted and the wiring of the system must be known ahead of time in order to properly set up the inputs/outputs.

By means of the STL editor your program can now be created very easily, e.g. copying segments. This package can also be used to create a program symbolically. That means, you do not initially have to have a definite assignment of symbols and inputs/outputs, but you can supply these assignments later, e.g. when your system is finished. These assignments are then connected with the statement list created in the editor by the batch compiler and are compiled into a STEP 5 program, which runs on your system. You can then test and correct this program in the programmable controller.

The batch compiler also carries out recompilations from a STEP 5 program so that e.g. the alterations of the tested program can be entered into your source and the statement list updated. A STEP 5 program, which has been created in the LAD, CSF, STL package, can be handled in a same manner.

That means, the STL editor/batch compiler option package provides many advantages:

1. *You can create a function specific program with symbols which becomes a system specific STEP 5 program (due to individual assignements); i.e. you can create a library with standardized programs.*
2. *Standardized programs and modules can be re-combined again and again using the Include command and can be connected to an individual STEP 5 program, without needing to carry out extensive alterations in the programs.*

1 - 2

C79000-B8576-C677-02

3. *In the STL editor operation is much easier than in the LAD, CSF, STL package. Here you can copy and move segments, program parts and entire blocks. The handling of the STL editor on the programming unit is similar to that of the symbol editor.*
4. *Program parts which occur frequently can be stored in files and can then be included at any point beyond of entire blocks of your open file or any other file using the Copy or Include command.*
5. *You can also create your statement list with another text editor because the batch compiler can also compile sequential files (ASCII files) into a STEP 5 program. Thus, other systems are also possible.*
6. *During the compilation via an intermediate file, your statement list can also be edited with a foreign-language STL editor.*
7. *Already existing STEP 5 programs, which were created in the LAD, CSF, STL package, are recompiled by the batch compiler and are therefore also available as a basis for standard programs and foreign-language versions.*

Further functions provided by this option package are a PC-specific test run for the compiled STEP 5 program and an error list.

In the following chapters you will find detailed descriptions of the STL editor and the batch compiler, in which the operation and the various functions are explained. The manual uses a pratical example to introduce you to the operation of the programming unit. The manual contains tables on control characters, STEP 5 operations and writing conventions.

In order to make reading easier words are written in

- **bold** letters, when discussing device keys and in
- *italics*, when discussing pure software functions (e.g. transfer) and when text is to be entered by the user (in the example).

Introduction

1

3. *In the STL editor operation is much easier than in the LAD, CSF, STL package. Here you can copy and move segments, program parts and entire blocks. The handling of the STL editor on the programming unit is similar to that of the symbol editor.*
4. *Program parts which occur frequently can be stored in files and can then be included at any point beyond of entire blocks of your open file or any other file using the Copy or Include command.*
5. *You can also create your statement list with another text editor because the batch compiler can also compile sequential files (ASCII files) into a STEP 5 program. Thus, other systems are also possible.*
6. *During the compilation via an intermediate file, your statement list can also be edited with a foreign-language STL editor.*
7. *Already existing STEP 5 programs, which were created in the LAD, CSF, STL package, are recompiled by the batch compiler and are therefore also available as a basis for standard programs and foreign-language versions.*

Further functions provided by this option package are a PC-specific test run for the compiled STEP 5 program and an error list.

In the following chapters you will find detailed descriptions of the STL editor and the batch compiler, in which the operation and the various functions are explained. The manual uses a pratical example to introduce you to the operation of the programming unit. The manual contains tables on control characters, STEP 5 operations and writing conventions.

In order to make reading easier words are written in

- **bold** letters, when discussing device keys and in
- *italics*, when discussing pure software functions (e.g. transfer) and when text is to be entered by the user (in the example).

Introduction

1 - 4

C79000-B8576-C677-02

Description 2

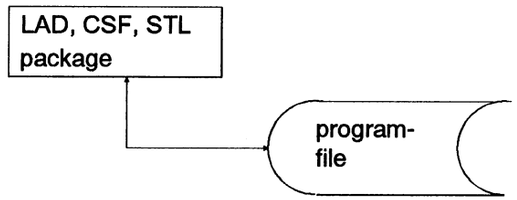
Description 2

C79000-B8576-C677-02 2 - 1

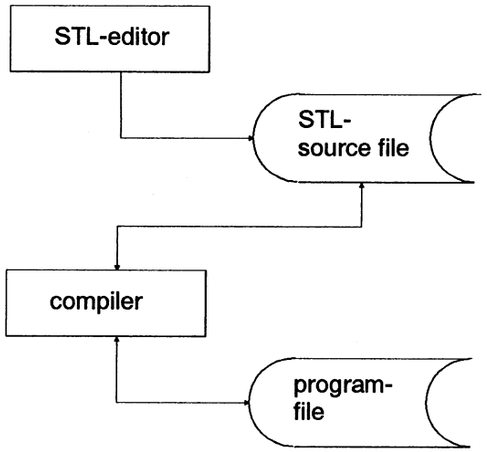
2.1 Functioning of the STL Editor/Batch Compiler

The creation of a STEP 5 program in the STL editor/batch compiler differs in the following respect from the LAD, CSF, STL package:

In the LAD, CSF, STL package the statement list is directly edited in the program file and immediately compiled into machine code.



In the STL editor/batch compiler package *editing* and *compilation* are separate processes, which are not carried out simultaneously.

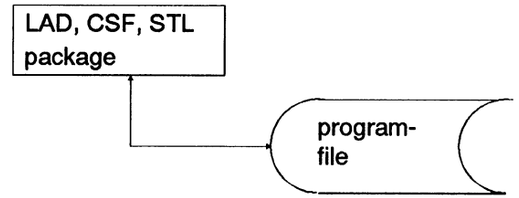


Description

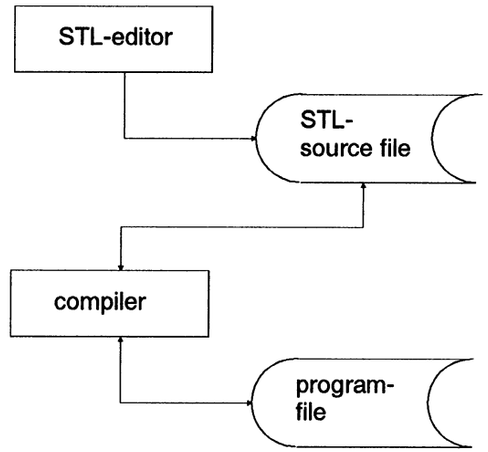
2.1 Functioning of the STL Editor/Batch Compiler

The creation of a STEP 5 program in the STL editor/batch compiler differs in the following respect from the LAD, CSF, STL package:

In the LAD, CSF, STL package the statement list is directly edited in the program file and immediately compiled into machine code.



In the STL editor/batch compiler package *editing* and *compilation* are separate processes, which are not carried out simultaneously.



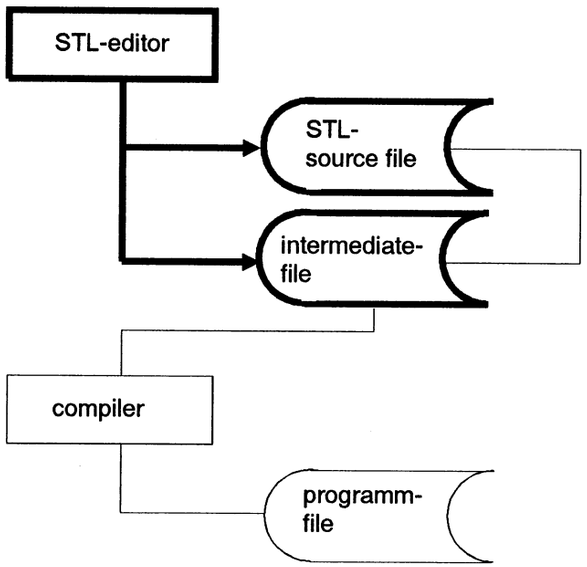
2 - 2
C79000-B8576-C677-02

editing

During the first step *(editing)*, write a sequential text file (STL source file) with the STL editor. It can contain a statement list, which has been created exclusively with symbols.

transfer

When data is saved via the *transfer* function or the **ENTER key**, the package automatically creates an intermediate file in addition to the STL source file. This intermediate file contains a code which is independent from national languages (language-independent), but is not yet a machine code. During this first *compilation* your statement list is checked on syntax and format.



Description

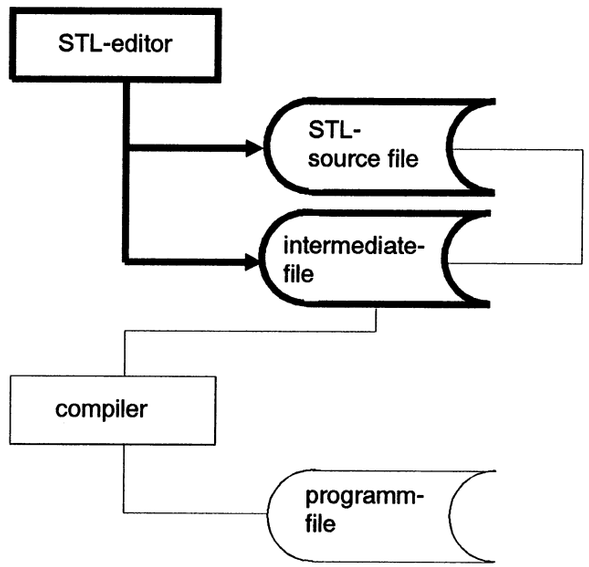
editing

During the first step *(editing)*, write a sequential text file (STL source file) with the STL editor. It can contain a statement list, which has been created exclusively with symbols.

transfer

When data is saved via the *transfer* function or the **ENTER key**, the package automatically creates an intermediate file in addition to the STL source file. This intermediate file contains a code which is independent from national languages (language-independent), but is not yet a machine code. During this first *compilation* your statement list is checked on syntax and format.

2

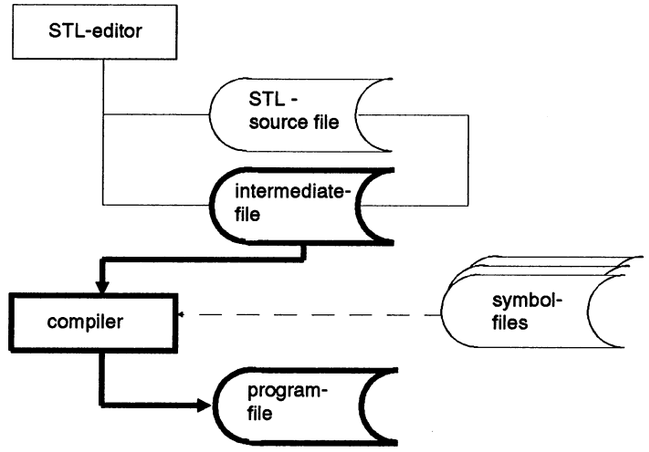


compilation

During the second step initiate the *compilation* via a function key. Here the batch compiler converts the intermediate file into a STEP 5 program file. If your statement list has been programmed symbolically, the batch compiler requires a symbol file with system specific assignments.

test

During the *compilation* of the program file, the assignments are *tested*. If you have entered a certain PC type, it is also checked whether the used operations are admissable for your destination PC (*PC specific test*). A program file created with the STL editor/batch compiler is identical to a program file created in an LAD, CSF, STL package.



Note: The STL source file can also be compiled in one step, e.g. if the source has only been saved but no intermediate file has been created, or if the source has been created by means of an external editor.

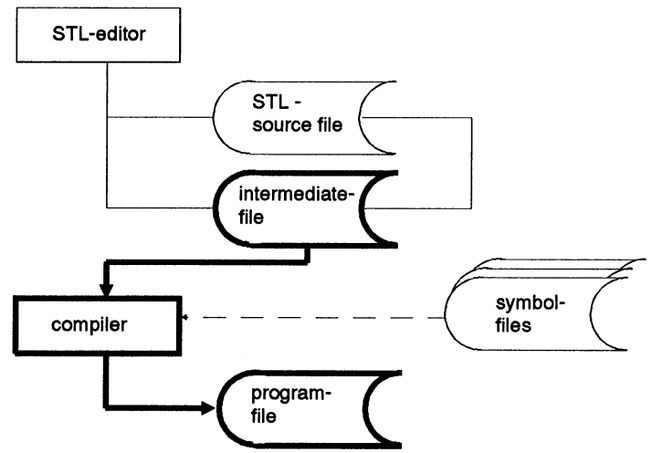
Description

compilation

During the second step initiate the *compilation* via a function key. Here the batch compiler converts the intermediate file into a STEP 5 program file. If your statement list has been programmed symbolically, the batch compiler requires a symbol file with system specific assignments.

test

During the *compilation* of the program file, the assignments are *tested*. If you have entered a certain PC type, it is also checked whether the used operations are admissable for your destination PC (*PC specific test*). A program file created with the STL editor/batch compiler is identical to a program file created in an LAD, CSF, STL package.

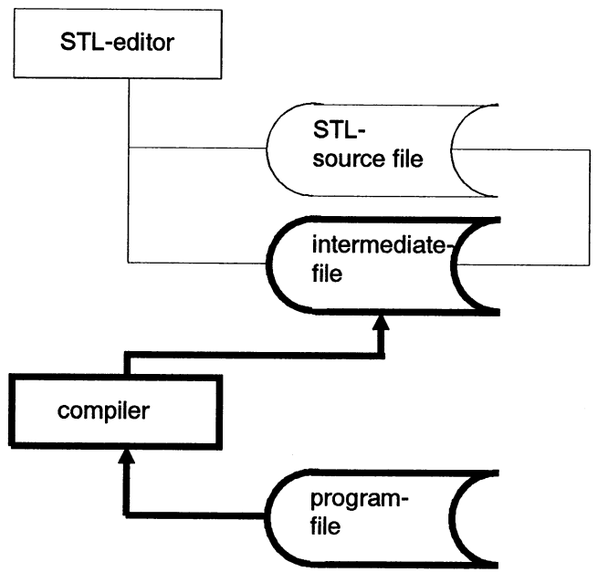


Note: The STL source file can also be compiled in one step, e.g. if the source has only been saved but no intermediate file has been created, or if the source has been created by means of an external editor.

You can also create a source file from a program file with the STL editor/batch compiler. This can be necessary e.g. after the STEP 5 program has been tested in the PC and corrected. Here it does not matter whether the program was edited in the LAD, CSF, STL package or in the STL editor/batch compiler package.

recompilation

During such a *recompilation* the batch compiler at first generates an intermediate file from the program file. From this intermediate file the STL source file is then created.



Note: The STL source file can also be created directly in one step from a program file.

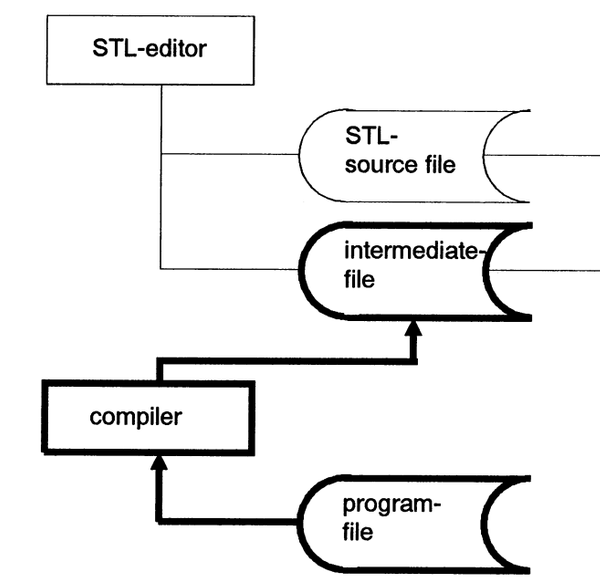
Description

You can also create a source file from a program file with the STL editor/batch compiler. This can be necessary e.g. after the STEP 5 program has been tested in the PC and corrected. Here it does not matter whether the program was edited in the LAD, CSF, STL package or in the STL editor/batch compiler package.

recompilation

During such a *recompilation* the batch compiler at first generates an intermediate file from the program file. From this intermediate file the STL source file is then created.

2

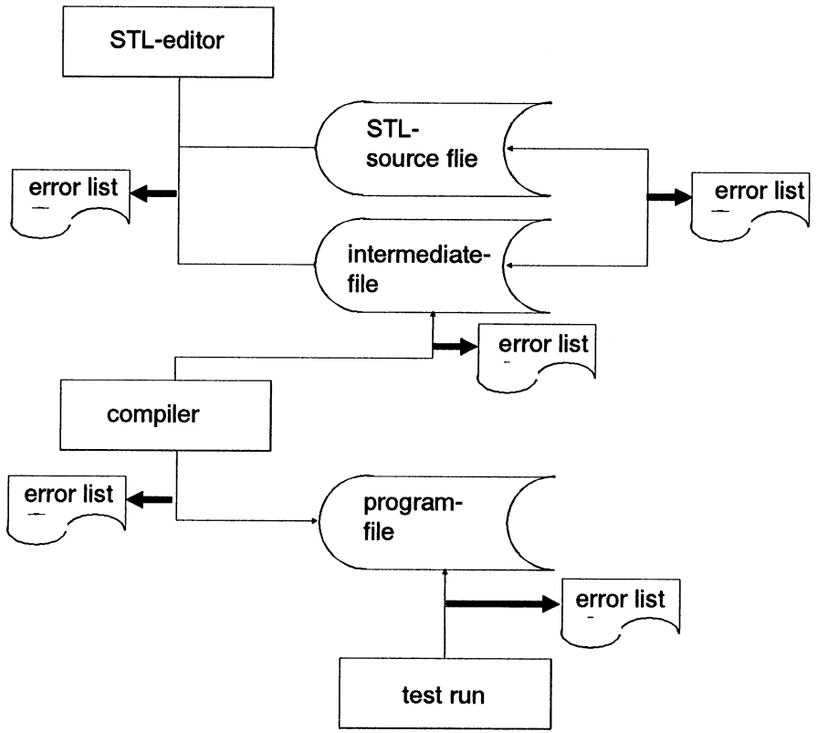


Note: The STL source file can also be created directly in one step from a program file.

test run

As mentioned above, *tests* are carried out during the *compilation*. Furthermore, a *test run for the program file blocks* is carried out. During that test run a check is made, whether formal operands and actual operands in function blocks have been correctly assigned and are equivalent. All occurring errors are listed in an error list and can be output on the printer.

The error list, however, only contains the errors of the last working step and is overwritten after every new *compilation* or *test*, resp. Therefore always output your error list on the printer! If a working step is carried out without any errors, no error list is created and existing ones are deleted!

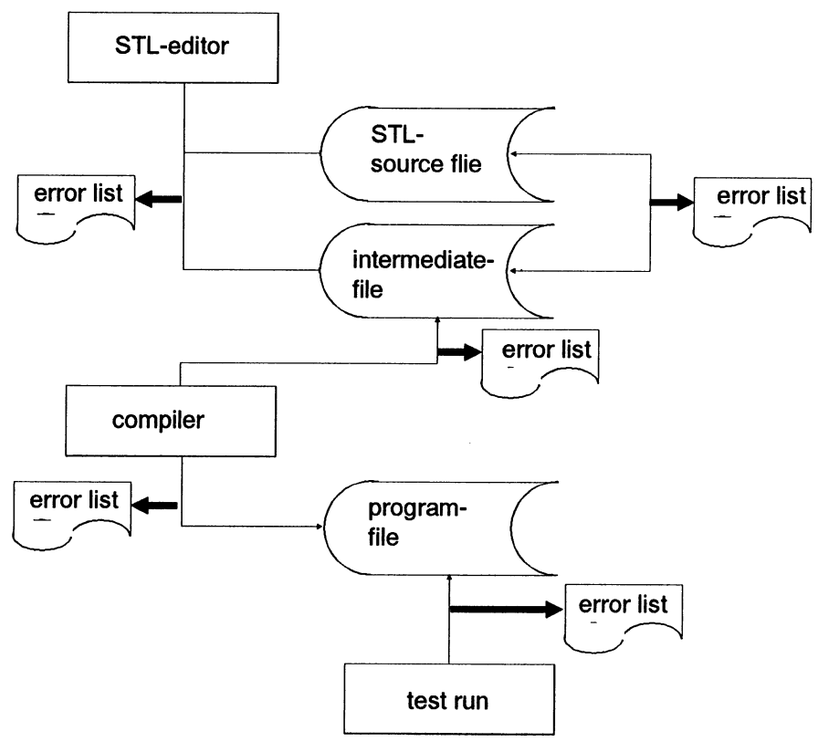


Description

test run

As mentioned above, *tests* are carried out during the *compilation*. Furthermore, a *test run for the program file blocks* is carried out. During that test run a check is made, whether formal operands and actual operands in function blocks have been correctly assigned and are equivalent. All occurring errors are listed in an error list and can be output on the printer.

The error list, however, only contains the errors of the last working step and is overwritten after every new *compilation* or *test*, resp. Therefore always output your error list on the printer! If a working step is carried out without any errors, no error list is created and existing ones are deleted!



2.2 Generating STEP 5 Blocks

2.2.1 In General

In the STL editor you can create your control program as a statement list using the advantages of a text editor. You write your statement list using the same commands and the same syntax as in the LAD, CSF, STL package. The only difference is that you have to adhere to certain conventions such as e.g. control characters for beginnings of blocks and comments. All block types which can be created in the LAD, CSF, STL package can also be created with the STL editor/batch compiler package.

All comments, which can be written in a block, can also be written. However, system comments (DOC file) cannot be written. In addition, the STL editor permits so-called additional comments that can be placed anywhere in the statement list. However, these comments are transferred into the program file and are lost during a re-compilation into the same STL source.

DB0 (reserved in the AG (=PC) for the block address list), DB1 (for the periphery assignment of the AG 135U and AG 155U), DX0 (for the system parametrization, CPU 928, R-processor, AG 155U), DB2 (for the controller list of R64), GRAPH 5 and Assembler blocks are not possible.

How to proceed during the editing and which control characters and writing conventions to observe is explained in the operating instructions, chapter 3.

Your statements are not written immediately into the STL source file on the hard or floppy disk but into an editing buffer in the main memory of your programming unit. The contents of the editing buffer, i.e. your statement list, is not stored in the STL source file before the *save* or *transfer* storage functions are used. Please observe the following differences:

- With *save* you only save your STL source file.
- With *transfer* you save your STL source file and compile it into the intermediate file.
- With *abort* (abort key) you delete the whole file edited in the main memory and leave the editor.

Description

2.2 Generating STEP 5 Blocks

2.2.1 In General

In the STL editor you can create your control program as a statement list using the advantages of a text editor. You write your statement list using the same commands and the same syntax as in the LAD, CSF, STL package. The only difference is that you have to adhere to certain conventions such as e.g. control characters for beginnings of blocks and comments. All block types which can be created in the LAD, CSF, STL package can also be created with the STL editor/batch compiler package.

2

All comments, which can be written in a block, can also be written. However, system comments (DOC file) cannot be written. In addition, the STL editor permits so-called additional comments that can be placed anywhere in the statement list. However, these comments are transferred into the program file and are lost during a re-compilation into the same STL source.

DB0 (reserved in the AG (=PC) for the block address list), DB1 (for the periphery assignment of the AG 135U and AG 155U), DX0 (for the system parametrization, CPU 928, R-processor, AG 155U), DB2 (for the controller list of R64), GRAPH 5 and Assembler blocks are not possible.

How to proceed during the editing and which control characters and writing conventions to observe is explained in the operating instructions, chapter 3.

Your statements are not written immediately into the STL source file on the hard or floppy disk but into an editing buffer in the main memory of your programming unit. The contents of the editing buffer, i.e. your statement list, is not stored in the STL source file before the *save* or *transfer* storage functions are used. Please observe the following differences:

- With *save* you only save your STL source file.
- With *transfer* you save your STL source file and compile it into the intermediate file.
- With *abort* (abort key) you delete the whole file edited in the main memory and leave the editor.

C79000-B8576-C677-02

2 - 7

Note:

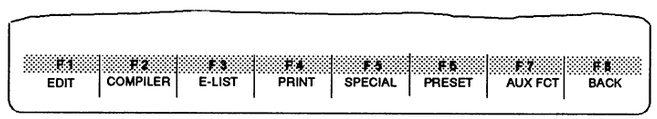
Although the editing buffer only has a certain finite size (this size is displayed in the editing mask for your information), you can edit much longer statement lists. For this purpose the STL editor sets up temporary files (A0.TM0, A0.TM1, A0.TM2). For these files adequate memory space is required on an external data carrier (usually hard disk). If there is not enough memory space available, an error message occurs. After *save* or *ENTER* , resp., these temporary files are deleted. Even in exceptional situations (e.g. power failure during editing) this data is not lost; however, they can be deleted by you at any time.

During the presetting four files are determined:

- the STL source file you would like to edit (A0.SEQ);
- the intermediate file which is created when storing via the transfer key and contains the statement list compiled into the intermediate code (A1.SEQ);
- the symbol file, which contains an assignment list (Z0.INI), and
- the program file in which the STEP 5 program is to be written after the compilation (ST.S5D).

These four files are automatically entered using the same name and can be altered, if necessary. The STL source file and the intermediate file always have the same name, however.

In the function selection, the STL editor/batch compiler package provides the following functions:



EDIT for creating and editing the STL source file (operation see chapter 3.3)

COMPILER for compiling and recompiling (operation see chapter 3.4)

E-LIST for the error list of the test runs; (operation see chapter 3.5)

Description

Note:

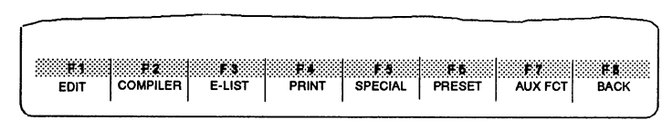
Although the editing buffer only has a certain finite size (this size is displayed in the editing mask for your information), you can edit much longer statement lists. For this purpose the STL editor sets up temporary files (A0.TM0, A0.TM1, A0.TM2). For these files adequate memory space is required on an external data carrier (usually hard disk). If there is not enough memory space available, an error message occurs. After *save* or *ENTER* , resp., these temporary files are deleted. Even in exceptional situations (e.g. power failure during editing) this data is not lost; however, they can be deleted by you at any time.

During the presetting four files are determined:

- the STL source file you would like to edit (A0.SEQ);
- the intermediate file which is created when storing via the transfer key and contains the statement list compiled into the intermediate code (A1.SEQ);
- the symbol file, which contains an assignment list (Z0.INI), and
- the program file in which the STEP 5 program is to be written after the compilation (ST.S5D).

These four files are automatically entered using the same name and can be altered, if necessary. The STL source file and the intermediate file always have the same name, however.

In the function selection, the STL editor/batch compiler package provides the following functions:



EDIT for creating and editing the STL source file (operation see chapter 3.3)

COMPILER for compiling and recompiling (operation see chapter 3.4)

E-LIST for the error list of the test runs; (operation see chapter 3.5)

PRINT for printing the STL source file (operation see chapter 3.6)

SPECIAL functions, for generating intermediate files and the STL source file (operation see chapter 3.7)

PRESET for changing the presetting;

AUX FCT functions, for administrating blocks in the preset program file and

BACK for exiting the STL editor/batch compiler.

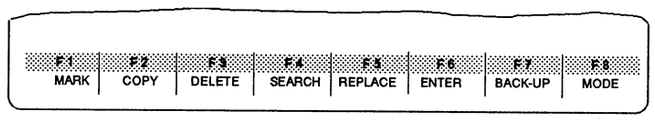
The following sections briefly outline these functions.

2.2.2 Editing Functions

The STL editor displays an editing mask on the screen, which is ready for a statement list. The editing mask consists of:

- a header containing the name of the STL source file,
- the input fields for ADDRESS, STATEMENT, OPERAND SYMBOL and STATEMENT COMMENT, arranged in columns,
- the menu with the editing functions.

It provides several functions so that your program can be edited more easily. They are comparable to the symbol editor functions:



By means of the **BUFFER** and **COPY** functions any character sequence and text block can be written into a buffer or to a sequential file ("evacuate") and copied to any place. That means, you can move or integrate segments again and again.

Additionally, you can read in other STL source files or single blocks from them with the copy function. The buffer, copy and delete functions can be combined with a repetition factor.

Description

PRINT for printing the STL source file (operation see chapter 3.6)

SPECIAL functions, for generating intermediate files and the STL source file (operation see chapter 3.7)

PRESET for changing the presetting;

AUX FCT functions, for administrating blocks in the preset program file and

BACK for exiting the STL editor/batch compiler.

The following sections briefly outline these functions.

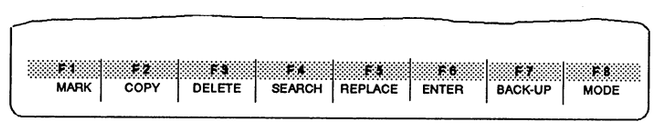
2

2.2.2 Editing Functions

The STL editor displays an editing mask on the screen, which is ready for a statement list. The editing mask consists of:

- a header containing the name of the STL source file,
- the input fields for ADDRESS, STATEMENT, OPERAND SYMBOL and STATEMENT COMMENT, arranged in columns,
- the menu with the editing functions.

It provides several functions so that your program can be edited more easily. They are comparable to the symbol editor functions:



By means of the BUFFER and COPY functions any character sequence and text block can be written into a buffer or to a sequential file ("evacuate") and copied to any place. That means, you can move or integrate segments again and again.

Additionally, you can read in other STL source files or single blocks from them with the copy function. The buffer, copy and delete functions can be combined with a repetition factor.

The **SEARCH** and **REPLACE** functions make it easier to correct your program:

you can move quickly and to desired locations in your file. You can alter individual character sequences, such as e.g. symbols or operands, in your entire statement list using only one function. Furthermore, you can select either the insert or overwrite **mode**.

As already mentioned, with **SAVE** you can save your file without exiting the editor. Saving your STL source file is reasonable for every short interruption of an editing session. **ENTER** saves your file and at the same time compiles it into the intermediate file and then exits the editor.

How to apply these editing functions in practice is described in chapter 3.3.5.

2.2.3 Control Characters

In order to make the compilation of the STL source file into a STEP 5 file possible, certain control characters and writing conventions have to be observed during the editing. You will find a complete survey of all editing conventions in chapters 3.3.2 and 3.3.3. Here only a few control characters are described, which are amendments compared to the LAD, CSF, STL package.

#TY marks the PC type. After this control character you can enter the PC, where the program should run. The entry has to be equivalent to the one in the language category field of the presetting. The batch compiler *checks* during the *compilation* in the program file, whether the edited operations are allowed in the available operations set of the given PC. The PC type can be placed in the STL source file at the beginning of the file and at block limits.

Description

The **SEARCH** and **REPLACE** functions make it easier to correct your program:

you can move quickly and to desired locations in your file. You can alter individual character sequences, such as e.g. symbols or operands, in your entire statement list using only one function. Furthermore, you can select either the insert or overwrite **mode**.

As already mentioned, with **SAVE** you can save your file without exiting the editor. Saving your STL source file is reasonable for every short interruption of an editing session. **ENTER** saves your file and at the same time compiles it into the intermediate file and then exits the editor.

How to apply these editing functions in practice is described in chapter 3.3.5.

2.2.3 Control Characters

In order to make the compilation of the STL source file into a STEP 5 file possible, certain control characters and writing conventions have to be observed during the editing. You will find a complete survey of all editing conventions in chapters 3.3.2 and 3.3.3. Here only a few control characters are described, which are amendments compared to the LAD, CSF, STL package.

#TY marks the PC type. After this control character you can enter the PC, where the program should run. The entry has to be equivalent to the one in the language category field of the presetting. The batch compiler *checks* during the *compilation* in the program file, whether the edited operations are allowed in the available operations set of the given PC. The PC type can be placed in the STL source file at the beginning of the file and at block limits.

2 - 10

C79000-B8576-C677-02

The following names for the PC language category are allowed:

PC type	Processor	Language category names
AG 100 U	CPU 100 CPU 102 CPU 103	CPU100 CPU102 CPU103
AG 101 U		AG101U
AG 110 S		AG110S
AG 115 U	CPU 941 CPU 942, 943 CPU 944	CPU941 CPU942 CPU944
AG 130 WB		AG130W
AG 135 U	CPU 921 CPU 922 CPU 928 CPU 928B	CPU921 CPU922 CPU928 CPU928B
AG 135 W		AG135W
AG 135 WB		AG135B
AG 90		AG90
AG 95		AG95
AG 150 A/K		AG150A
AG 150 S/U		AG150U
AG 155 U	CPU 946/947	AG155U
I/O processor	IP 257	IP257

The *compilation* into the program file is only carried out, if the name of the PC in the presetting ("LANGUAGE CATEGORY" field) is equivalent to the entry in the #TY lines of the STL source file. If they are not equivalent, the *compilation* in the #TY-line is aborted. If you enter "NO" in the presetting for the language category, the *compilation* is carried out without a *PC specific test run*..

Description

The following names for the PC language category are allowed:

PC type	Processor	Language category names
AG 100 U	CPU 100 CPU 102 CPU 103	CPU100 CPU102 CPU103
AG 101 U		AG101U
AG 110 S		AG110S
AG 115 U	CPU 941 CPU 942, 943 CPU 944	CPU941 CPU942 CPU944
AG 130 WB		AG130W
AG 135 U	CPU 921 CPU 922 CPU 928 CPU 928B	CPU921 CPU922 CPU928 CPU928B
AG 135 W		AG135W
AG 135 WB		AG135B
AG 90		AG90
AG 95		AG95
AG 150 A/K		AG150A
AG 150 S/U		AG150U
AG 155 U	CPU 946/947	AG155U
I/O processor	IP 257	IP257

2

The *compilation* into the program file is only carried out, if the name of the PC in the presetting ("LANGUAGE CATEGORY" field) is equivalent to the entry in the #TY lines of the STL source file. If they are not equivalent, the *compilation* in the #TY-line is aborted. If you enter "NO" in the presetting for the language category, the *compilation* is carried out without a *PC specific test run*..

include command

The #I Include command enables the integration of any file and can be placed in the STL source file at the beginning of the file or at block limits, i.e. after BE. #I is followed by the file name. Here, it is important that the disk drive is also entered (for example: #I B:TEST).

The file is included via an intermediate code, i.e. the file to be included has to exist as an intermediate file. If the disk drive is not entered, the drive is accessed which was determined in the presetting for the intermediate file. If blocks with identical names are found in both files, then they have to be renamed before the *compilation*. You can avoid this problem, however, by giving symbolic names to the blocks in your STL source files.

For this purpose it is absolutely necessary that the corresponding symbol file exists, as the block type and number are indispensible for the compilation.

The Include command is especially well suited for user specific libraries: standardised programs can be edited with Include files according to the required task. During modifications only the Include files have to be exchanged. After the modification the latest edition is used for all programs for the creation of the program file.

2.2.4 Compilation

Creating a Program File

The batch compiler can now compile all blocks, a block group or an individual block from the intermediate file or the STL source file into the program file. If you have saved all the modifications in your statement list in the source file with *ENTER*, only the intermediate file needs to be compiled. If not, you must initiate the compilation of the AWL source file, which automatically creates an updated intermediate file.

If you programmed your STL source file symbolically the preset symbol file is linked to the intermediate file during the

compilation into the program file. A symbol file is not created by the STL editor, but has to be created with the symbol editor. If another file is included with the #I Include command, you have to observe that the preset symbol file contains the symbols for this file.

Description

include command

The #I Include command enables the integration of any file and can be placed in the STL source file at the beginning of the file or at block limits, i.e. after BE. #I is followed by the file name. Here, it is important that the disk drive is also entered (for example: #I B:TEST).

The file is included via an intermediate code, i.e. the file to be included has to exist as an intermediate file. If the disk drive is not entered, the drive is accessed which was determined in the presetting for the intermediate file. If blocks with identical names are found in both files, then they have to be renamed before the *compilation*. You can avoid this problem, however, by giving symbolic names to the blocks in your STL source files.

For this purpose it is absolutely necessary that the corresponding symbol file exists, as the block type and number are indispensible for the compilation.

The Include command is especially well suited for user specific libraries: standardised programs can be edited with Include files according to the required task. During modifications only the Include files have to be exchanged. After the modification the latest edition is used for all programs for the creation of the program file.

2.2.4 Compilation

Creating a Program File

The batch compiler can now compile all blocks, a block group or an individual block from the intermediate file or the STL source file into the program file. If you have saved all the modifications in your statement list in the source file with *ENTER*, only the intermediate file needs to be compiled. If not, you must initiate the compilation of the AWL source file, which automatically creates an updated intermediate file.

If you programmed your STL source file symbolically the preset symbol file is linked to the intermediate file during the

compilation into the program file. A symbol file is not created by the STL editor, but has to be created with the symbol editor. If another file is included with the #I Include command, you have to observe that the preset symbol file contains the symbols for this file.

2 - 12

C79000-B8576-C677-02

In the command lines of the **compiler** you can enter whether a machine code should be generated or whether only a test on errors should be carried out, and whether a back-up confirmation should occur when overwriting blocks. An output of the compiled program on the printer can also be specified.

Recompilation from a Program File

Neither STL source files nor intermediate files exist for blocks which were established with the LAD, CSF, STL package. The STL editor/batch compiler can create these files from a program file. The intermediate file is created after *recompiling* a block, a block group or all blocks from a program file. When a block, a block group or all blocks are recompiled from the program file, you can primarily create the intermediate file or directly the STL source file, which can be modified or complemented.

2

During the *recompilation* you select what your "new" STL source file is going to look like. The statements are either only displayed with symbols or absolute parameters, or with both. Furthermore, the control character for the language category identification is entered in the intermediate file, if a language category identification (PC type) had been entered in the presetting.

The STL editor can process files having up to 65535 lines. The number of lines of the STL source file however does not only depend on the number of STEP 5 instructions, but also on special statements, comment lines etc. If the program file you would like to *recompile* is longer, the blocks have to be distributed to several intermediate files.

Standard function blocks as well as Graph 5 and Assembler blocks are not recompiled.

compilation checks

During the *compilation/recompilation* the intermediate code is checked as to whether the statement being created is admissible. It is also checked as to whether the statement is admissible concerning the block type. The language category is checked for the PC type entered in the presetting.

Description

In the command lines of the **compiler** you can enter whether a machine code should be generated or whether only a test on errors should be carried out, and whether a back-up confirmation should occur when overwriting blocks. An output of the compiled program on the printer can also be specified.

Recompilation from a Program File

Neither STL source files nor intermediate files exist for blocks which were established with the LAD, CSF, STL package. The STL editor/batch compiler can create these files from a program file. The intermediate file is created after *recompiling* a block, a block group or all blocks from a program file. When a block, a block group or all blocks are recompiled from the program file, you can primarily create the intermediate file or directly the STL source file, which can be modified or complemented.

2

During the *recompilation* you select what your "new" STL source file is going to look like. The statements are either only displayed with symbols or absolute parameters, or with both. Furthermore, the control character for the language category identification is entered in the intermediate file, if a language category identification (PC type) had been entered in the presetting.

The STL editor can process files having up to 65535 lines. The number of lines of the STL source file however does not only depend on the number of STEP 5 instructions, but also on special statements, comment lines etc. If the program file you would like to *recompile* is longer, the blocks have to be distributed to several intermediate files.

Standard function blocks as well as Graph 5 and Assembler blocks are not recompiled.

compilation checks

During the *compilation/recompilation* the intermediate code is checked as to whether the statement being created is admissible. It is also checked as to whether the statement is admissible concerning the block type. The language category is checked for the PC type entered in the presetting.

In the case of symbolic programming the assignments in connection with the operands are checked. If an absolute as well as a symbol operand have been entered in the STL source file, a check is carried out whether the symbol file corresponds to it. If the parameters are not equivalent, the absolute parameter assigned to the symbol from the symbol file is used and an alarm message is stored in the error list. In the case of absolute programming no access to the symbol file occurs. Errors found during these checks are outputted in the error list.

2.2.5 Printing

You can create a listing of the STL source file via the **printing** function (in the function selection). However, this function only outputs the preset STL source file on the printer.

In the command lines of the **compiler** function a printer output is offered during the *compilation*, so that you can record the result of every compilation run including the test run.

The STL editor/batch compiler provides the printing formats, which are normally used in the STEP 5 basic package for the layout of your printer output. A selection can be made between a standard output, normal print, small print and super small print. The title block has to be 132 characters wide in the A3 format (F2.INI file) and 80 characters in the A4 format (F1.INI file). The symbol comment is also displayed when super small print is output.

Description

In the case of symbolic programming the assignments in connection with the operands are checked. If an absolute as well as a symbol operand have been entered in the STL source file, a check is carried out whether the symbol file corresponds to it. If the parameters are not equivalent, the absolute parameter assigned to the symbol from the symbol file is used and an alarm message is stored in the error list. In the case of absolute programming no access to the symbol file occurs. Errors found during these checks are outputted in the error list.

2.2.5 Printing

You can create a listing of the STL source file via the **printing** function (in the function selection). However, this function only outputs the preset STL source file on the printer.

In the command lines of the **compiler** function a printer output is offered during the *compilation*, so that you can record the result of every compilation run including the test run.

The STL editor/batch compiler provides the printing formats, which are normally used in the STEP 5 basic package for the layout of your printer output. A selection can be made between a standard output, normal print, small print and super small print. The title block has to be 132 characters wide in the A3 format (F2.INI file) and 80 characters in the A4 format (F1.INI file). The symbol comment is also displayed when super small print is output.

2 - 14 C79000-B8576-C677-02

2.3 The A1.SEQ Intermediate File

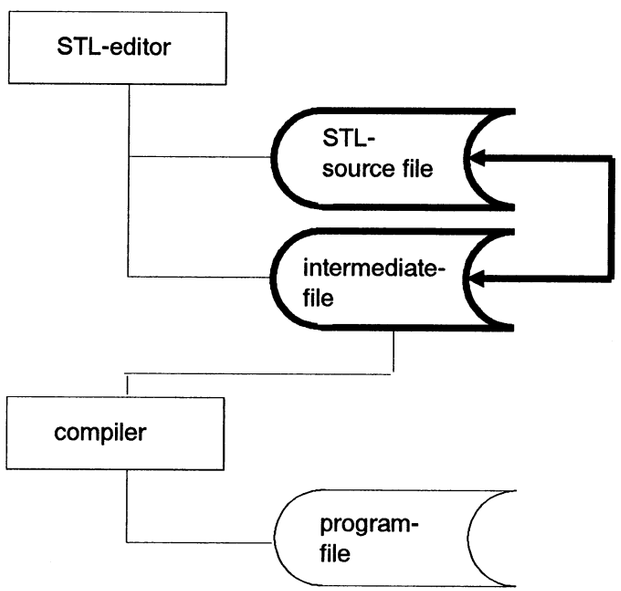
The intermediate file is the central file in the STL editor/batch compiler package. It is the basis for all compilation as it is language independent and not yet in MC5 machine code. At any time

- STEP 5 program files,
- STL source files,
- system specific program versions,
- foreign language program versions
- can be created from the intermediate file.

Therefore, it is absolutely necessary to always save the intermediate file and it is recommended to always exit an STL source file with *ENTER* so that the intermediate file is up-to-date.

2

2.3.1 Relations between the STL Source File and the Intermediate File



Description

2.3 The A1.SEQ Intermediate File

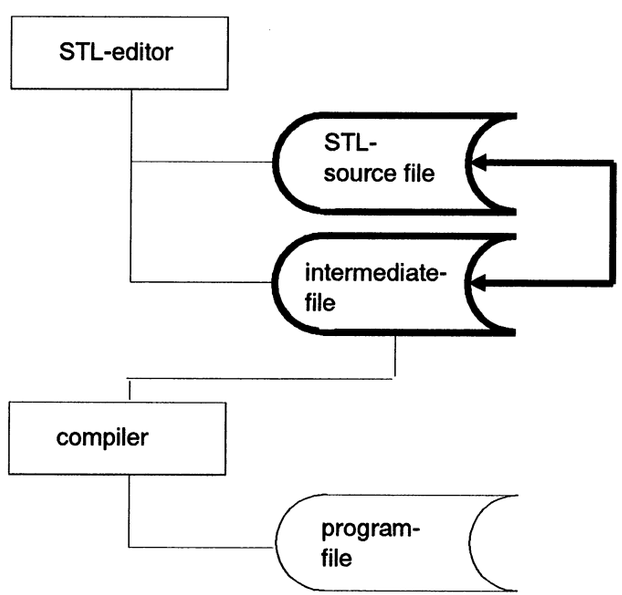
The intermediate file is the central file in the STL editor/batch compiler package. It is the basis for all compilation as it is language independent and not yet in MC5 machine code. At any time

- STEP 5 program files,
- STL source files,
- system specific program versions,
- foreign language program versions
- can be created from the intermediate file.

Therefore, it is absolutely necessary to always save the intermediate file and it is recommended to always exit an STL source file with *ENTER* so that the intermediate file is up-to-date.

2

2.3.1 Relations between the STL Source File and the Intermediate File



The STL source file and the intermediate file are closely connected: they have the same name and their file identification differs only in one character (A0.SEQ, A1.SEQ). The name of an intermediate file can never be altered independently from the STL source file. However, both files can be on different disk drives. By using identical names it is guaranteed that your edited programs are compiled in the pertinent intermediate file in the case of an *ENTER*.

recompilation

During a *recompilation* it is also stored in the STL source file with the same name. It should be noted that if the STL source file and the intermediate file are not in the same Version or if the "old" statement list should not be overwritten,e.g. if your first STL source file contains additional comments; comments are not transferred into the program file and are lost after the recompilation.

Therefore, the following conditions are required for the generation of an STL source file from an intermediate file:

- If no STL source file is available it is created automatically, if *edit* is called. It is named as pre-determined in the presetting.
- If an STL source file with the same name exists, the intermediate file has to be transferred explicitly into the STL source file. This occurs using the **INT>SEQ** special function (as below). The STL source file is overwritten.

If an "old" STL source file should not be deleted, please enter a name for the new STL source file before the *recompilation* of a program file. Under this name the recompilation into the intermediate file and saving into the STL source file then occurs.

That is why it is important to check before each *compilation run*, however especially before a *recompilation*, whether the correct files have been entered in the presetting.

Note:
The SEQ>MC5 and MC5>SEQ functions automatically create an updated intermediate file (see section 2.3.2)

Description

The STL source file and the intermediate file are closely connected: they have the same name and their file identification differs only in one character (A0.SEQ, A1.SEQ). The name of an intermediate file can never be altered independently from the STL source file. However, both files can be on different disk drives. By using identical names it is guaranteed that your edited programs are compiled in the pertinent intermediate file in the case of an *ENTER*.

recompilation

During a *recompilation* it is also stored in the STL source file with the same name. It should be noted that if the STL source file and the intermediate file are not in the same Version or if the "old" statement list should not be overwritten,e.g. if your first STL source file contains additional comments; comments are not transferred into the program file and are lost after the recompilation.

Therefore, the following conditions are required for the generation of an STL source file from an intermediate file:

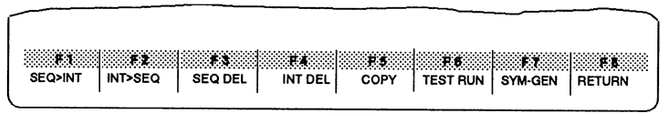
- If no STL source file is available it is created automatically, if *edit* is called. It is named as pre-determined in the presetting.
- If an STL source file with the same name exists, the intermediate file has to be transferred explicitly into the STL source file. This occurs using the **INT>SEQ** special function (as below). The STL source file is overwritten.

If an "old" STL source file should not be deleted, please enter a name for the new STL source file before the *recompilation* of a program file. Under this name the recompilation into the intermediate file and saving into the STL source file then occurs.

That is why it is important to check before each *compilation run*, however especially before a *recompilation*, whether the correct files have been entered in the presetting.

Note:
The SEQ>MC5 and MC5>SEQ functions automatically create an updated intermediate file (see section 2.3.2)

2.3.2
Special Functions



The special functions offer various conversions for creating STL source files and intermediate files. This can be necessary because the STL source file and the intermediate file always bear the same name, but sometimes are dated differently (as above, 2.3.1).

In chapter 3.7 we will show you how to use these functions in practice.

SEQ>INT converts a sequential file into an intermediate file. You will use this function if e.g. your STL source file was written with another text editor and should now be compiled into a STEP 5 program file, or if the intermediate file no longer exists.

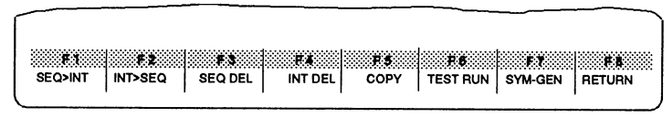
INT>SEQ converts an intermediate file into a sequential file. E.g you have recompiled from a program file and intend to edit it using the STL editor. The intermediate file has to be converted into a sequential file for this purpose. This function is especially important if an old version of the source file exists. This function is also very helpful, when editing the source file with a foreign language STL editor (as below, 2.3.4),.

SEQDELETE With the **SEQDELETE** you can delete sequential files if you want to e.g. newly generate them during compilation runs. If the compiled files are *edited* then the sequential files are automatically generated by the editor.

Intermediate files are deleted using **INTDELETE** (e.g. old versions). They are either reestablished using the **SEQ>INT** function from an updated source file or when *entering* an edited source.

Description

2.3.2
Special Functions



2

The special functions offer various conversions for creating STL source files and intermediate files. This can be necessary because the STL source file and the intermediate file always bear the same name, but sometimes are dated differently (as above, 2.3.1).

In chapter 3.7 we will show you how to use these functions in practice.

SEQ>INT converts a sequential file into an intermediate file. You will use this function if e.g. your STL source file was written with another text editor and should now be compiled into a STEP 5 program file, or if the intermediate file no longer exists.

INT>SEQ converts an intermediate file into a sequential file. E.g you have recompiled from a program file and intend to edit it using the STL editor. The intermediate file has to be converted into a sequential file for this purpose. This function is especially important if an old version of the source file exists. This function is also very helpful, when editing the source file with a foreign language STL editor (as below, 2.3.4),.

SEQDELETE With the **SEQDELETE** you can delete sequential files if you want to e.g. newly generate them during compilation runs. If the compiled files are *edited* then the sequential files are automatically generated by the editor.

Intermediate files are deleted using **INTDELETE** (e.g. old versions). They are either reestablished using the **SEQ>INT** function from an updated source file or when *entering* an edited source.

COPY If you want to copy the intermediate file and the STL source file to other drives as back-up, use the **COPY** function. However, the files can only be renamed in the the F-AUX utility program.

TEST RUN The **TEST RUN** can be used to subsequently test the compiled blocks of a program file, but also blocks from the LAD, CSF, STL package on the admissibility of the commands for the set PC type.

SYM-GEN The **SYM-GEN** function creates a symbolic source file from the STL source file, which contains all used symbols and absolute parameters. This symbolic source can be extended by assignments and comments by means of the symbolic editor. The symbols and absolute parameters which already exist in the STL source need not be entered again.

2.3.3 Standard Programs

Standard programs can be established due to the fact that a statement list can be created using only symbols and that the intermediate file is language independent. Tested blocks and modules can be stored in libraries and then be interconnected with the Include command to individual programs for the specific systems. That means, you only need to connect your newly combined programs to a special assignment list and system specific STEP 5 programs will be given for individual control problems.

Description

COPY If you want to copy the intermediate file and the STL source file to other drives as back-up, use the COPY function. However, the files can only be renamed in the the F-AUX utility program.

TEST RUN The TEST RUN can be used to subsequently test the compiled blocks of a program file, but also blocks from the LAD, CSF, STL package on the admissibility of the commands for the set PC type.

SYM-GEN The SYM-GEN function creates a symbolic source file from the STL source file, which contains all used symbols and absolute parameters. This symbolic source can be extended by assignments and comments by means of the symbolic editor. The symbols and absolute parameters which already exist in the STL source need not be entered again.

2.3.3 Standard Programs

Standard programs can be established due to the fact that a statement list can be created using only symbols and that the intermediate file is language independent. Tested blocks and modules can be stored in libraries and then be interconnected with the Include command to individual programs for the specific systems. That means, you only need to connect your newly combined programs to a special assignment list and system specific STEP 5 programs will be given for individual control problems.

2 - 18 C79000-B8576-C677-02

2.3.4 Foreign Language STEP 5 Program Versions

You can also create foreign-language STEP 5 program versions with the batch compiler if you have programmed your program absolutely or if you have a recompiled intermediate file with absolute parameters: with the English and French software version of this package you create English and French STL sources.

For this purpose enter the **F-AUX, F-TRANSFER** utility program and copy your German STL source file and intermediate file with a new name. You must enter this new name in the presetting of the English/French software version. The **INT>SEQ** (as above) special function transfers the language-independent intermediate file into the sequential source file. This file is then outputted during the **EDITING** with English/French STEP 5 commands.

Another possibility is to delete the German sequential source file (**SEQDELETE** special function) in the English/French package. A new one is automatically generated if **EDIT** is called.

Symbols and comments are not outputted in foreign languages.

Description

2.3.4 Foreign Language STEP 5 Program Versions

You can also create foreign-language STEP 5 program versions with the batch compiler if you have programmed your program absolutely or if you have a recompiled intermediate file with absolute parameters: with the English and French software version of this package you create English and French STL sources.

For this purpose enter the **F-AUX, F-TRANSFER** utility program and copy your German STL source file and intermediate file with a new name. You must enter this new name in the presetting of the English/French software version. The **INT>SEQ** (as above) special function transfers the language-independent intermediate file into the sequential source file. This file is then outputted during the **EDITING** with English/French STEP 5 commands.

Another possibility is to delete the German sequential source file (**SEQDELETE** special function) in the English/French package. A new one is automatically generated if **EDIT** is called.

Symbols and comments are not outputted in foreign languages.

2

C79000-B8576-C677-02

2 - 19

2.4 Editing and Supplementing STEP 5 Blocks

2.4.1 Blocks Created Using the STL Editor

The name of the STL source file and possibly the drive for the intermediate file is entered in the presetting. After calling the STL editor the desired statement list and the editing menu are displayed on the screen. Now the statement list can be edited or supplemented using the editing functions. Always save your edited source file with *enter*, so that the intermediate file is updated and your "new" program file is not created with an "old" program.

If you exit the STL editor with the **abort key** and confirm the abort, the editions or supplements are not transferred into the STL source file.

2.4.2 Blocks Created Using the LAD, CSF, STL Package

Program file blocks have to be recompiled so that they can be edited in the STL editor. If they then exist in an STL source file, they are edited in the STL editor as described above, in 2.4.1.

Description

2.4 Editing and Supplementing STEP 5 Blocks

2.4.1 Blocks Created Using the STL Editor

The name of the STL source file and possibly the drive for the intermediate file is entered in the presetting. After calling the STL editor the desired statement list and the editing menu are displayed on the screen. Now the statement list can be edited or supplemented using the editing functions. Always save your edited source file with *enter*, so that the intermediate file is updated and your "new" program file is not created with an "old" program.

If you exit the STL editor with the **abort key** and confirm the abort, the editions or supplements are not transferred into the STL source file.

2.4.2 Blocks Created Using the LAD, CSF, STL Package

Program file blocks have to be recompiled so that they can be edited in the STL editor. If they then exist in an STL source file, they are edited in the STL editor as described above, in 2.4.1.

2.5 Test Run

2.5.1 Program File Testing

The test run occurs after the *compilation*. During the test run the program file blocks are tested. E.g. the parameter transfer of function blocks and the existence of the called blocks are checked. You can determine a test run for a block, a block group or all blocks of a program file. If a language category code has been entered in the presetting, additionally the admissibilty of the statements to the PC type is checked. Non-admissible statements are logged in the error list.

2.5.2 Testing of Special Blocks

Standard function blocks, Graph 5 and Assembler blocks cannot be created and recompiled with the STL editor/batch compiler but they can be tested using the subsequent *test run* ! The existence and transfer of parameters as well as the admissibility of the STL statements to the preset PC type are checked.

Description

2.5 Test Run

2.5.1 Program File Testing

The test run occurs after the *compilation*. During the test run the program file blocks are tested. E.g. the parameter transfer of function blocks and the existence of the called blocks are checked. You can determine a test run for a block, a block group or all blocks of a program file. If a language category code has been entered in the presetting, additionally the admissibilty of the statements to the PC type is checked. Non-admissible statements are logged in the error list.

2.5.2 Testing of Special Blocks

Standard function blocks, Graph 5 and Assembler blocks cannot be created and recompiled with the STL editor/batch compiler but they can be tested using the subsequent *test run* ! The existence and transfer of parameters as well as the admissibility of the STL statements to the preset PC type are checked.

2

C79000-B8576-C677-02

2 - 21

2.6 Error List

2.6.1 Error File

During the following processes error messages can occur:

- Compilation of the STL source file into the intermediate file.
- Compilation of the intermediate file into the program file.
- Recompilation of the program file into the intermediate file.
- Recompilation of the intermediate file into the STL source file.
- Program file test (test run).

The error messages are stored by the programming unit in an error list in a <name>AF.SEQ error file.

Only the error list of the last process carried out is always continued in the error file. The error file can be outputted on the screen or printer in the desired printed output format. This error file does not exist if the last working step was terminated without any errors!

Description

2.6 Error List

2.6.1 Error File

During the following processes error messages can occur:

- Compilation of the STL source file into the intermediate file.
- Compilation of the intermediate file into the program file.
- Recompilation of the program file into the intermediate file.
- Recompilation of the intermediate file into the STL source file.
- Program file test (test run).

The error messages are stored by the programming unit in an error list in a <name>AF.SEQ error file.

Only the error list of the last process carried out is always continued in the error file. The error file can be outputted on the screen or printer in the desired printed output format. This error file does not exist if the last working step was terminated without any errors!

2 - 22

C79000-B8576-C677-02

2.7 Entering STEP 5 Statements with other Editors

2.7.1 STL Source File as an Interface

The STL source file can also be created with other editors. However, a necessary condition is that these editors can process "real" tabs (i.e. the 09H hexcode). If not, the initial columns of the individual part fields must be determined in the first line of the STL source file by means of the #TAB control characters (see Section 2.7.3).

The first six characters of the file name can be selected as desired. The name has to consist of six characters. A0.SEQ has to be the last two characters of the name and the extension. This file can only be further processed with the tools of the STL editor/batch compiler package without problems, if the format of the sequential source file described below is adhered to. The STL editor/batch compiler then supports you with the **SEQ>INT**, and with the subsequent *compilation* into the program file, or with the direct compilation by means of the **SEQMC5** function, resp..

2.7.2 Sequential Source File Format

One data block has to be entered per statement line. A data block begins with the tab character (09H) and consists of four data fields which are also separated by tabs. The end of a block is marked with "carriage return, CR" (=0DH) and "line feed, LF". This is automa tically added by the editor at the end of a line via the return key. The maximum number of characters for the following fields are:

TAB	TAB	TAB	TAB	CR, LF
Address 4 characters	**Statement** 14 characters	**Operand symbol** 23 characters	**Statement comment** 32 characters	

That means, the data block of a blank line consists of four tab characters followed by the "CR" and "LF" characters.

Description

2.7 Entering STEP 5 Statements with other Editors

2.7.1 STL Source File as an Interface

The STL source file can also be created with other editors. However, a necessary condition is that these editors can process "real" tabs (i.e. the 09H hexcode). If not, the initial columns of the individual part fields must be determined in the first line of the STL source file by means of the #TAB control characters (see Section 2.7.3).

The first six characters of the file name can be selected as desired. The name has to consist of six characters. A0.SEQ has to be the last two characters of the name and the extension. This file can only be further processed with the tools of the STL editor/batch compiler package without problems, if the format of the sequential source file described below is adhered to. The STL editor/batch compiler then supports you with the SEQ>INT, and with the subsequent compilation into the program file, or with the direct compilation by means of the SEQMC5 function, resp..

2.7.2 Sequential Source File Format

One data block has to be entered per statement line. A data block begins with the tab character (09H) and consists of four data fields which are also separated by tabs. The end of a block is marked with "carriage return, CR" (=0DH) and "line feed, LF". This is automa tically added by the editor at the end of a line via the return key. The maximum number of characters for the following fields are:

TAB TAB TAB TAB CR, LF

Address 4 characters | Statement 14 characters | Operand symbol 23 characters | Statement comment 32 characters

That means, the data block of a blank line consists of four tab characters followed by the "CR" and "LF" characters.

C79000-B8576-C677-02

2 - 23

The data block for comment lines starts with the tab character (=09H), directly followed by the control characters ✱ and ; for segment and additional comments. A comment of up to 79 characters may follow along with and the end of the line including the "CR" (=0DH) and "LF" characters (=0AH).

Lower and upper case letters are allowed in the data blocks as lower case letters are automatically transferred into upper case letters by the editor when they are read in. Accented vowels cannot be used (e.g. ö,é...etc.).

2.7.3 #TAB Control Characters for Processing of External Files

Thanks to the #TAB control character even files without real tab characters, which are created by many text programs such as e.g. 1st Wordplus, can be compiled. However, the STL editor cannot edit these files and a "wrong file format" error message is output.

#TAB must be placed directly at the beginning of the source file. Only blanks are allowed before it. It must be followed by 4 numbers, separated by commas, which determine the initial columns of the part fields. Any further entries are not allowed in the first line!

Example: If 1 blank each is required between the part fields as a separation, the first line of the STL source file is as follows:

#TAB 1,6,21,46 RETURN (CR LF)

The entries for the columns always refer to the beginning of the line. The difference of the entries which follow each other must be at least as big as the corresponding lengths of the part fields (see 2.7.2).

Description

The data block for comment lines starts with the tab character (=09H), directly followed by the control characters ✱ and ; for segment and additional comments. A comment of up to 79 characters may follow along with and the end of the line including the "CR" (=0DH) and "LF" characters (=0AH).

Lower and upper case letters are allowed in the data blocks as lower case letters are automatically transferred into upper case letters by the editor when they are read in. Accented vowels cannot be used (e.g. ö,é...etc.).

2.7.3 #TAB Control Characters for Processing of External Files

Thanks to the #TAB control character even files without real tab characters, which are created by many text programs such as e.g. 1st Wordplus, can be compiled. However, the STL editor cannot edit these files and a "wrong file format" error message is output.

#TAB must be placed directly at the beginning of the source file. Only blanks are allowed before it. It must be followed by 4 numbers, separated by commas, which determine the initial columns of the part fields. Any further entries are not allowed in the first line!

Example: If 1 blank each is required between the part fields as a separation, the first line of the STL source file is as follows:

#TAB 1,6,21,46 RETURN (CR LF)

The entries for the columns always refer to the beginning of the line. The difference of the entries which follow each other must be at least as big as the corresponding lengths of the part fields (see 2.7.2).

Operation of the Programming Unit

3

Operation of the Programming Unit

3

C79000-B8576-C677-02

3 - 1

3.1 Software Installation

Copy the STL editor/batch compiler files to the hard disk of your programming unit using the PCP/M PIP command. Consult the PCP/M table booklet which is enclosed in your PG manual.

Example: PIP b:[G0]=a:*.*

In this example b: represents the target drive, [G0] the user and a: the source drive,

Operation

3.1 Software Installation

Copy the STL editor/batch compiler files to the hard disk of your programming unit using the PCP/M PIP command. Consult the PCP/M table booklet which is enclosed in your PG manual.

Example: PIP b:[G0]=a:*.*

In this example b: represents the target drive, [G0] the user and a: the source drive,

3 - 2

C79000-B8576-C677-02

3.2 Operational Steps up to the Function Selection

3.2.1 The Operation in General

The operation of the programming unit corresponds for the STL editor/batch compiler to the conventions of all other S5 packages. i.e. when filling in a presetting the softkeys with the possible editing and operating functions are displayed in the function selection. The **cursor keys**, the **help, abort** und **ENTER key** are identical regarding their functions.

In the STL editor they work exactly as they do in the symbol editor. However, there are several differences to the **INPUT/OUTPUT** functions in the LAD, CSF, STL package. That means it is e.g. necessary to provide some statements in the STL editor with control characters. The symbols, however, do not require hyphens. When especially applicable we will especially draw your attention to it.

The **Help key** gives explanations and information concerning masks, softkeys and input fields. Within these help texts the question "Continue?" is answered with the **abort key** (= no) or with the **ENTER key** (= yes).

In this package all the STEP 5 term definitions are valid. If you require general information about STEP 5, please read the introduction in the manuals of your programming unit.

3

3.2.2 Entry into the STL Editor/Batch Compiler Package

☞ The following example has been created on a PG 750 with one disk drive. All actions to be carried out are marked with ➤ . The characters which you enter are written in *italics*; the functions to be used or the function keys, resp., in **bold** letters. All the terms which are displayed on the screen and which are referred to during the operation are written in uppercase letters.

Operation

3.2 Operational Steps up to the Function Selection

3.2.1 The Operation in General

The operation of the programming unit corresponds for the STL editor/batch compiler to the conventions of all other S5 packages. i.e. when filling in a presetting the softkeys with the possible editing and operating functions are displayed in the function selection. The **cursor keys**, the **help, abort** und **ENTER key** are identical regarding their functions.

In the STL editor they work exactly as they do in the symbol editor. However, there are several differences to the **INPUT/OUTPUT** functions in the LAD, CSF, STL package. That means it is e.g. necessary to provide some statements in the STL editor with control characters. The symbols, however, do not require hyphens. When especially applicable we will especially draw your attention to it.

The **Help key** gives explanations and information concerning masks, softkeys and input fields. Within these help texts the question "Continue?" is answered with the **abort key** (= no) or with the **ENTER key** (= yes).

In this package all the STEP 5 term definitions are valid. If you require general information about STEP 5, please read the introduction in the manuals of your programming unit.

3

3.2.2 Entry into the STL Editor/Batch Compiler Package

☞ The following example has been created on a PG 750 with one disk drive. All actions to be carried out are marked with ➤ . The characters which you enter are written in *italics*; the functions to be used or the function keys, resp., in **bold** letters. All the terms which are displayed on the screen and which are referred to during the operation are written in uppercase letters.

C79000-B8576-C677-02

3 - 3

Conditions:

The STL editor/batch compiler option package has been loaded on the hard disk of your programming unit. The programming unit is ready for operation and the PCP/M-86 operating system is active. On the screen the B> ready-for-operation-message is displayed.

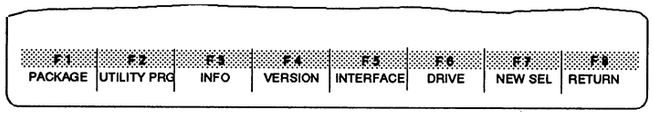
Starting the S5-Komi

The S5 command interpreter (S5-KOMI) is started from the starting mask with the commands

➤ F1 (start S5-DOS)

➤ or on the operating system level with the command: S5

Then the STEP 5 packages are listed in the PACKAGE SELECTION on the screen.



The softkeys are explained when you press the **Help key.** Detailed information to these functions can be found in the STEP 5 description, the 2nd volume of your programming unit manual, S5 Command Interpreter chapter.

Loading the STL Editor/Batch Compiler Package

➤ Position cursor before the STL editor/batch compiler package,

➤ call **PACKAGE** (F1)

or

➤ press **ENTER key.**

Then the PRESETTING appears.

Operation

Conditions:

The STL editor/batch compiler option package has been loaded on the hard disk of your programming unit. The programming unit is ready for operation and the PCP/M-86 operating system is active. On the screen the B> ready-for-operation-message is displayed.

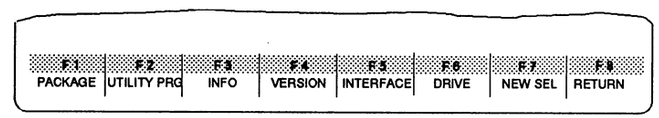
Starting the S5-Komi

The S5 command interpreter (S5-KOMI) is started from the starting mask with the commands

➤ F1 (start S5-DOS)

➤ or on the operating system level with the command: S5

Then the STEP 5 packages are listed in the PACKAGE SELECTION on the screen.



The softkeys are explained when you press the **Help key.** Detailed information to these functions can be found in the STEP 5 description, the 2nd volume of your programming unit manual, S5 Command Interpreter chapter.

Loading the STL Editor/Batch Compiler Package

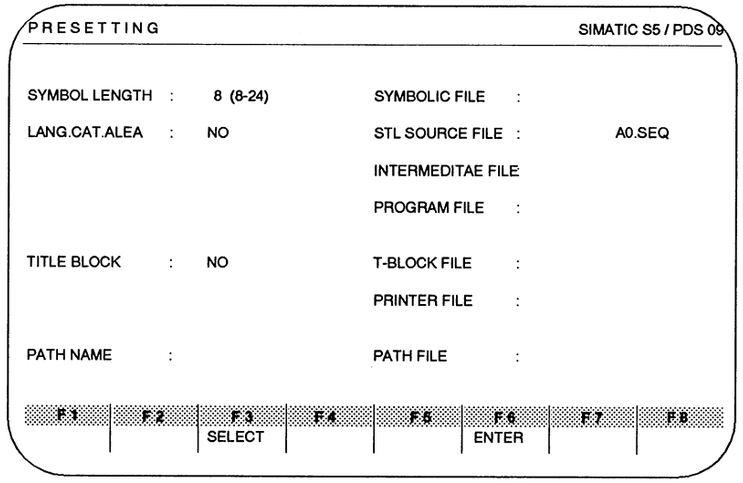
➤ Position cursor before the STL editor/batch compiler package,

➤ call **PACKAGE** (F1)

or

➤ press **ENTER key.**

Then the PRESETTING appears.



PRESETTING

The cursor is flashing in the line of the STL SOURCE FILE (A0.SEQ). In this file your statement list is stored. It is marked as a sequential file, i.e. as an ASCII file and it is the source for the compilation run.

Here please enter the name of your file. In our example the file is named "test".

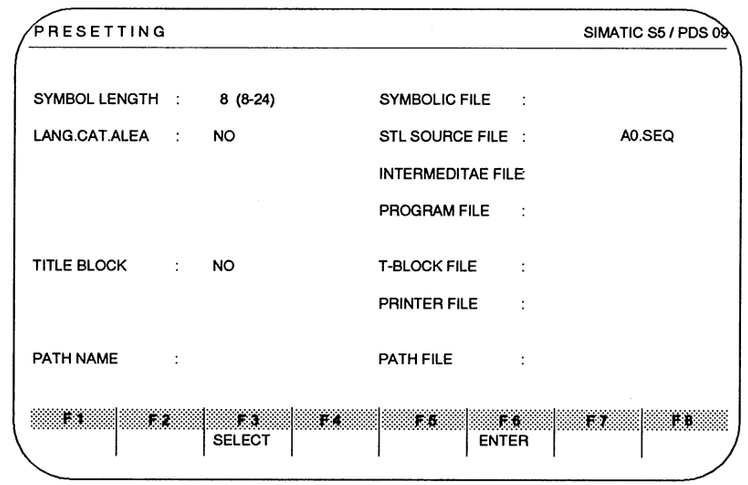
➤ enter character sequence *Test* ,

➤ press **return key**.

Now the hard disk is entered as the drive by the programming unit and the name is filled in with @. Furthermore, the INTERMEDIATE FILE (A1.SEQ), the SYMBOL FILE (Z0.INI) and the PROGRAM FILE (ST.s5D) are filled in with the same name. Now they are marked in a way which shows that they belong together.

If no SYMBOL FILE, i.e. no assignment list with a filled-in name is available, then this is indicated three times: (GESP) is displayed after the file name, the programming unit outputs the message "File B:TEST@@Z0.INI: file not existing", and in the SYMBOL LENGTH field the preset 8 is converted into a 0.

Operation



3

PRESETTING

The cursor is flashing in the line of the STL SOURCE FILE (A0.SEQ). In this file your statement list is stored. It is marked as a sequential file, i.e. as an ASCII file and it is the source for the compilation run.

Here please enter the name of your file. In our example the file is named "test".

➤ enter character sequence *Test* ,

➤ press **return key**.

Now the hard disk is entered as the drive by the programming unit and the name is filled in with @. Furthermore, the INTERMEDIATE FILE (A1.SEQ), the SYMBOL FILE (Z0.INI) and the PROGRAM FILE (ST.s5D) are filled in with the same name. Now they are marked in a way which shows that they belong together.

If no SYMBOL FILE, i.e. no assignment list with a filled-in name is available, then this is indicated three times: (GESP) is displayed after the file name, the programming unit outputs the message "File B:TEST@@Z0.INI: file not existing", and in the SYMBOL LENGTH field the preset 8 is converted into a 0.

overwrite names

If the source file is to be linked to a symbol file with another name during the compilation and/or is to be compiled into a program file with another name, then the individual names can now be overwritten. After overwriting terminate using the return key in each case. These files are also active in other STEP 5 packages and are entered in their presettings during the loading. The names of the files for title block, printer and paths are automatically adapted to the program file.

➤ position cursor in the required line,

➤ press **arrow key right,**

➤ the input field is filled in.

The PG checks whether the entered files are available. If other files are to be used, please enter their names. If the printer and path file are not available, their names are deleted when the cursor is moved upwards or downwards the next time.

The PATH NAME and TITLE BLOCK lines are treated as in the LAD, CSF, STL package: The name is entered at the path name and the width is selected at the title block.

language category

In the LANGUAGE CATEGORY field you should by all means use the **Help key** (position the cursor on one letter of the word NO): via the **Help key** the programmable controllers (AG) and central processing units (CPU) are displayed, for which the batch compiler compiles and tests.

If required you enter at language category a that device from the list, on which your program is supposed to run. For further information please see the table in chapter 2.2.3. Then the batch compiler checks during the compilation in the program file, whether your statement list corresponds to the PC language category.

In the SYMBOL LENGTH field the symbol length of the entered symbol file is displayed. You cannot modify this field, which is merely for display.

For saving this PRESETTING please press

➤ ENTER (F6)

or

➤ the **ENTER key.**

Operation

overwrite names

If the source file is to be linked to a symbol file with another name during the compilation and/or is to be compiled into a program file with another name, then the individual names can now be overwritten. After overwriting terminate using the return key in each case. These files are also active in other STEP 5 packages and are entered in their presettings during the loading. The names of the files for title block, printer and paths are automatically adapted to the program file.

➤ position cursor in the required line,

➤ press **arrow key right,**

➤ the input field is filled in.

The PG checks whether the entered files are available. If other files are to be used, please enter their names. If the printer and path file are not available, their names are deleted when the cursor is moved upwards or downwards the next time.

The PATH NAME and TITLE BLOCK lines are treated as in the LAD, CSF, STL package: The name is entered at the path name and the width is selected at the title block.

language category

In the LANGUAGE CATEGORY field you should by all means use the **Help key** (position the cursor on one letter of the word NO): via the **Help key** the programmable controllers (AG) and central processing units (CPU) are displayed, for which the batch compiler compiles and tests.

If required you enter at language category a that device from the list, on which your program is supposed to run. For further information please see the table in chapter 2.2.3. Then the batch compiler checks during the compilation in the program file, whether your statement list corresponds to the PC language category.

In the SYMBOL LENGTH field the symbol length of the entered symbol file is displayed. You cannot modify this field, which is merely for display.

For saving this PRESETTING please press

➤ ENTER (F6)

or

➤ the **ENTER key.**

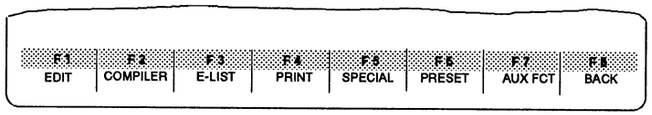
3 - 6

C79000-B8576-C677-02

Now the FUNCTION SELECTION is displayed.

FUNCTION SELECTION

The FUNCTION SELECTION offers the softkeys for the following editing and processing functions. The following chapters explain how to use them.



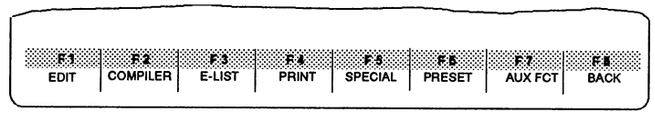
3

Operation

Now the FUNCTION SELECTION is displayed.

FUNCTION SELECTION

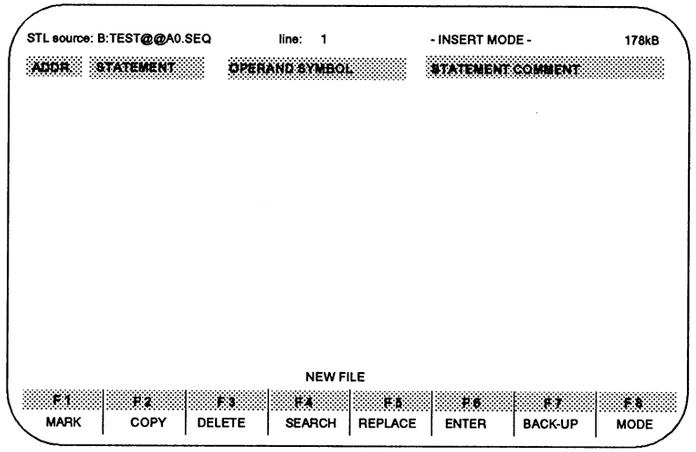
The FUNCTION SELECTION offers the softkeys for the following editing and processing functions. The following chapters explain how to use them.



3

3.3 Edit

Call editing mode Call **EDIT** (F1).

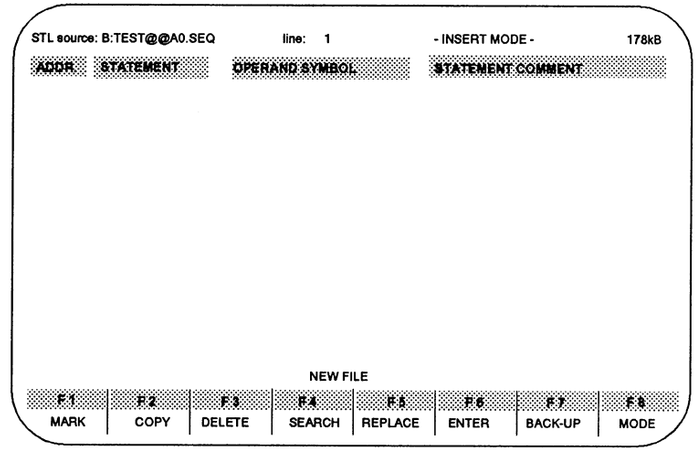


This screen is ready to edit a statement list, i.e. either to enter a new one or to output one already available for editing (corrections, modifications).

Operation

3.3 Edit

Call editing mode Call **EDIT** (F1).



This screen is ready to edit a statement list, i.e. either to enter a new one or to output one already available for editing (corrections, modifications).

3.3.1
Editor Description

Screen

heading Here you will find

the name of your preset STL source file and the corresponding disk drive,

the line entry for the cursor position,

the insert or overwrite editing mode and

the buffer sizes of the memory. This information is important for the processing speed.

editing field The editing field is divided into four columns for which the widths cannot be modified. The widths and the intended contents of the columns are as follows:

ADDR 4 characters	STATEMENT 14 characters	OPERAND SYMBOL 23 characters	STATEMENT COMMENT 32 characters (max. symbol length)
Addresses, Labels	Operations, Absolute operands, Constants	Operations,Symbols Values of the Constants	Comments

footer All device messages are displayed in this line, e.g. "new file", if a new statement list is being created.

Operation

3.3.1
Editor Description

Screen

heading Here you will find

the name of your preset STL source file and the corresponding disk drive,

the line entry for the cursor position,

the insert or overwrite editing mode and

the buffer sizes of the memory. This information is important for the processing speed.

editing field The editing field is divided into four columns for which the widths cannot be modified. The widths and the intended contents of the columns are as follows:

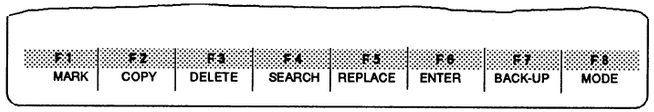
ADDR 4 characters	STATEMENT 14 characters	OPERAND SYMBOL 23 characters	STATEMENT COMMENT 32 characters (max. symbol length)
Addresses, Labels	Operations, Absolute operands, Constants	Operations,Symbols Values of the Constants	Comments

footer All device messages are displayed in this line, e.g. "new file", if a new statement list is being created.

C79000-B8576-C677-02

3 - 9

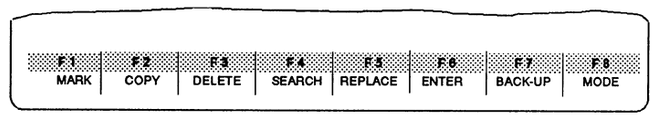
softkeys These editing functions are identical to those of the SYMBOL EDITOR. They are used to create and process a statement list.



EDIT softkey functions The following diagram is a survey of the tools, which are available for the individual editing functions. If one of the editing menu keys is pressed, then the corresponding menu (at the end of the arrow) is output.

Operation

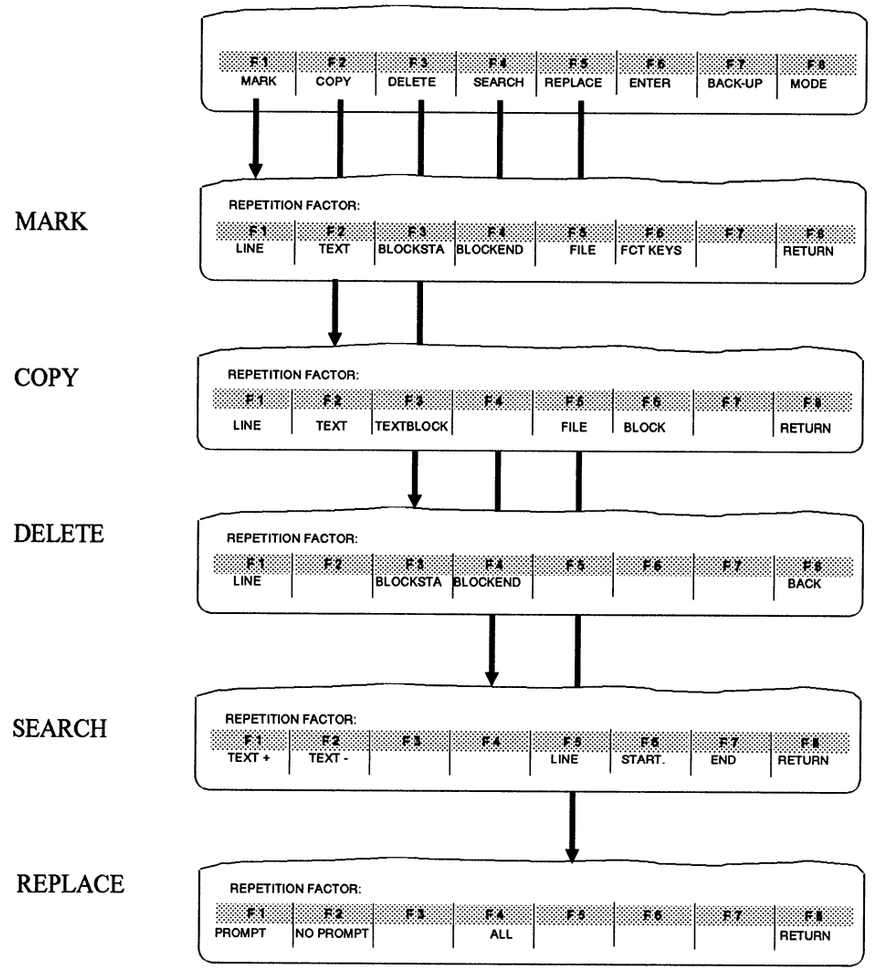
softkeys These editing functions are identical to those of the SYMBOL EDITOR. They are used to create and process a statement list.



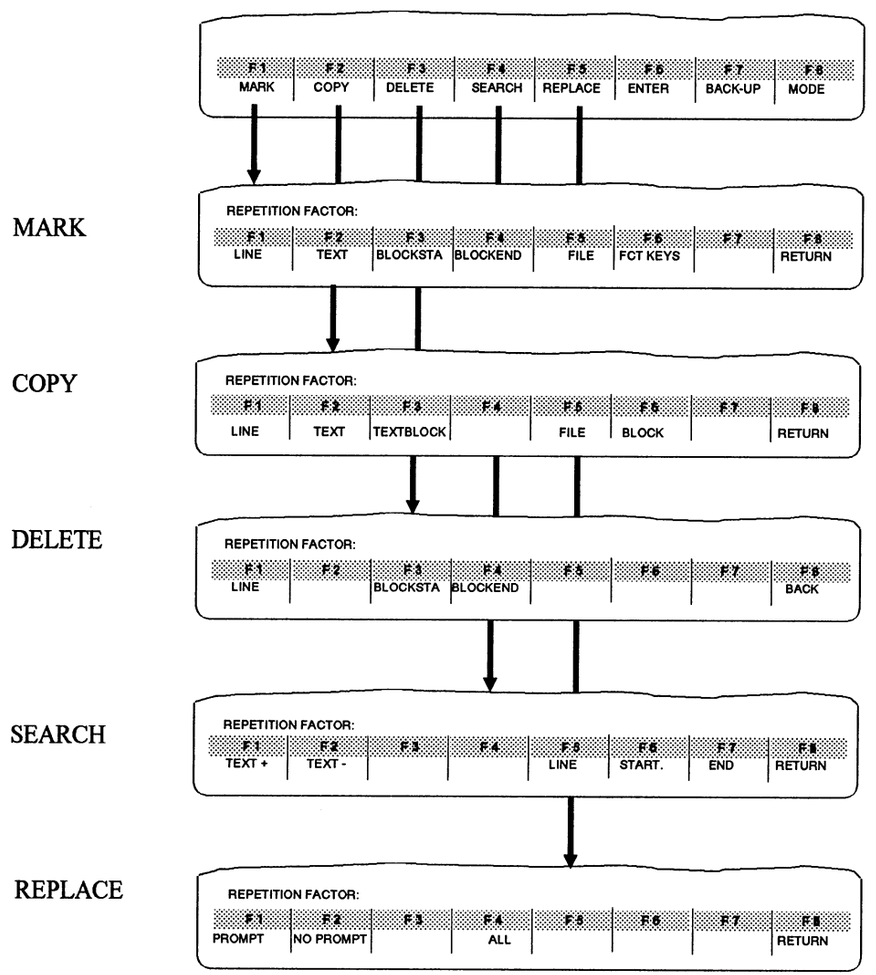
EDIT softkey functions The following diagram is a survey of the tools, which are available for the individual editing functions. If one of the editing menu keys is pressed, then the corresponding menu (at the end of the arrow) is output.

3 - 10

C79000-B8576-C677-02


F1 MARK
F2 COPY
F3 DELETE
F4 SEARCH
F5 REPLACE
F6 ENTER
F7 BACK-UP
F8 MODE
MARK
REPETITION FACTOR:
F1 LINE
F2 TEXT
F3 BLOCKSTA
F4 BLOCKEND
F5 FILE
F6 FCT KEYS
F8 RETURN
COPY
REPETITION FACTOR:
F1 LINE
F2 TEXT
F3 TEXTBLOCK
F5 FILE
F6 BLOCK
F8 RETURN
DELETE
REPETITION FACTOR:
F1 LINE
F3 BLOCKSTA
F4 BLOCKEND
F8 BACK
SEARCH
REPETITION FACTOR:
F1 TEXT +
F2 TEXT -
F5 LINE
F6 START.
F7 END
F8 RETURN
REPLACE
REPETITION FACTOR:
F1 PROMPT
F2 NO PROMPT
F4 ALL
F8 RETURN

3


F1 MARK
F2 COPY
F3 DELETE
F4 SEARCH
F5 REPLACE
F6 ENTER
F7 BACK-UP
F8 MODE
MARK
REPETITION FACTOR:
F1 LINE
F2 TEXT
F3 BLOCKSTA
F4 BLOCKEND
F5 FILE
F6 FCT KEYS
F8 RETURN
COPY
REPETITION FACTOR:
F1 LINE
F2 TEXT
F3 TEXTBLOCK
F5 FILE
F6 BLOCK
F8 RETURN
DELETE
REPETITION FACTOR:
F1 LINE
F3 BLOCKSTA
F4 BLOCKEND
F8 BACK
SEARCH
REPETITION FACTOR:
F1 TEXT +
F2 TEXT -
F5 LINE
F6 START.
F7 END
F8 RETURN
REPLACE
REPETITION FACTOR:
F1 PROMPT
F2 NO PROMPT
F4 ALL
F8 RETURN

3

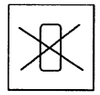
special keys

In addition to these functions the special keys are available as further "tools" for editing your file.

hardkeys

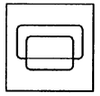
All cursor keys are available for moving the cursor.

The character, where the cursor
is positioned is deleted with
(or DEL key resp.)

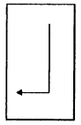


In your file you can page down with
(moves text up on the screen)

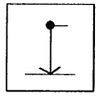
and page up with
(moves text down on the screen)



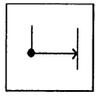
In the INSERT MODE
a new line is inserted
below the cursor with



and a new line is inserted
above the cursor with the
"vertical space",



and a blank with the
"horizontal space"
(beside the blank key).



Operation

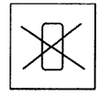
special keys

In addition to these functions the special keys are available as further "tools" for editing your file.

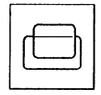
hardkeys

All cursor keys are available for moving the cursor.

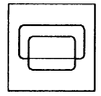
The character, where the cursor
is positioned is deleted with
(or DEL key resp.)



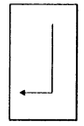
In your file you can page down with
(moves text up on the screen)



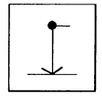
and page up with
(moves text down on the screen)



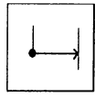
In the INSERT MODE
a new line is inserted
below the cursor with



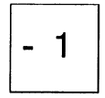
and a new line is inserted
above the cursor with the
"vertical space",



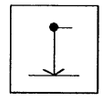
and a blank with the
"horizontal space"
(beside the blank key).



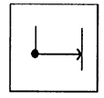
The character to the left of the cursor is deleted with (or "-" key resp.)



In the OVERWRITE MODE you can only insert a new line with

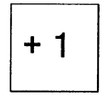


and another blank only with



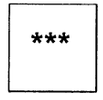
3

If you want to delete the whole text between the two colons within a command in the footer, use:
(or "+" key resp.)



By pressing the abort key you will end any activated function, but data may be therefore lost. E.g. if a file has been corrected and then the abort key is pressed, all modifications will be lost.
(or ESC key resp.)

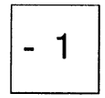
The end of network key



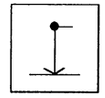
creates "***", i.e. an end of network command, if the cursor is placed in the "statement" field. The key is locked outside this field and in the case of additional and network comments.

Operation

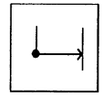
The character to the left of the cursor is deleted with (or "-" key resp.)



In the OVERWRITE MODE you can only insert a new line with

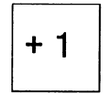


and another blank only with

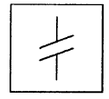


3

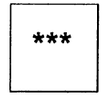
If you want to delete the whole text between the two colons within a command in the footer, use:
(or "+" key resp.)



By pressing the abort key you will end any activated function, but data may be therefore lost. E.g. if a file has been corrected and then the abort key is pressed, all modifications will be lost.
(or ESC key resp.)



The end of network key



creates "***", i.e. an end of network command, if the cursor is placed in the "statement" field. The key is locked outside this field and in the case of additional and network comments.

3.3.2 The STL Editor/Batch Compiler Control Characters and their Writing Conventions

For certain entries the STL EDITOR requires several control characters so that it is possible to compile the statement list into a STEP 5 program file. E.g. segment headlines and comments, actual operands and block identifications have to be marked correspondingly.

Column STATEMENTS Control characters	Identification for	Writing conventions with examples
#TAB	source file without "real" tabs	#TAB 1,6,21,46
#TY	PC type	with blanks #TY_AG155U #TY_CPU928
#PBn #OBn #FBn, #FXn #DBn, #DXn (#SBn, no GRAPH 5 block)	Program block beginning Organisation block begin. Function block beginning Data block beginning Step block beginning	without blanks #PB11 #OB1 #FB25, #FX12 #DB5, #DX33 #SB3
#BI	Library number	with blanks #BI_12345 not higher than 65535
#N	Function block name	with blanks #N_GARAGE 6 characters max.

Operation

3.3.2 The STL Editor/Batch Compiler Control Characters and their Writing Conventions

For certain entries the STL EDITOR requires several control characters so that it is possible to compile the statement list into a STEP 5 program file. E.g. segment headlines and comments, actual operands and block identifications have to be marked correspondingly.

Column STATEMENTS Control characters	Identification for	Writing conventions with examples
#TAB	source file without "real" tabs	#TAB 1,6,21,46
#TY	PC type	with blanks #TY_AG155U #TY_CPU928
#PBn #OBn #FBn, #FXn #DBn, #DXn (#SBn, no GRAPH 5 block)	Program block beginning Organisation block begin. Function block beginning Data block beginning Step block beginning	without blanks #PB11 #OB1 #FB25, #FX12 #DB5, #DX33 #SB3
#BI	Library number	with blanks #BI_12345 not higher than 65535
#N	Function block name	with blanks #N_GARAGE 6 characters max.

3 - 14

C79000-B8576-C677-02

These control characters are listed in the table. It shows the order determined for a trouble-free compilation into the intermediate and program file. Furthermore, you are informed about the writing conventions (_ represents a blank) and the position of the control characters within the statement list and further explanations will also be found.

Position within the statement list	Explanations
always first line in a file	Allowes the translation of files, which are created with another editor, for example 1st word plus. Relevant only for the compiler, not for the STL editor.
always first statement in a file	Comments that occur only exist in the STL source file. They are not compiled and are lost during a recompilation.
Beginning of a block; after a BE (block end, as below, operations)	Value range: n= 0 - 255, according to PC type If further statements need to be entered after the ending of a block then they must be entered after the beginning of a new block. If not, these statements are lost during the compilation in the programming unit. DB0, DB1, DB2 are not allowed.
after the beginning of the block or after the block name (as below, #N)	For your own library numbers; the standard function block numbers cannot and do not need to be entered. Comments can only occur in the STL source file. They are not compiled and are lost during the recompilation.
before or after the library number, but at the block beginning.	

Operation

These control characters are listed in the table. It shows the order determined for a trouble-free compilation into the intermediate and program file. Furthermore, you are informed about the writing conventions (_ represents a blank) and the position of the control characters within the statement list and further explanations will also be found.

Position within the statement list	Explanations
always first line in a file	Allowes the translation of files, which are created with another editor, for example 1st word plus. Relevant only for the compiler, not for the STL editor.
always first statement in a file	Comments that occur only exist in the STL source file. They are not compiled and are lost during a recompilation.
Beginning of a block; after a BE (block end, as below, operations)	Value range: n= 0 - 255, according to PC type If further statements need to be entered after the ending of a block then they must be entered after the beginning of a new block. If not, these statements are lost during the compilation in the programming unit. DB0, DB1, DB2 are not allowed.
after the beginning of the block or after the block name (as below, #N)	For your own library numbers; the standard function block numbers cannot and do not need to be entered. Comments can only occur in the STL source file. They are not compiled and are lost during the recompilation.
before or after the library number, but at the block beginning.	

3

C79000-B8576-C677-02

3 - 15

Column STATEMENTS Control characters	Identification for	**Writing conventions with examples**
#UB	Segment header	The control character is found in the STATEMENT column, the text for the heading in the STATEMENT COMMENT column
( )	Formal parameter type	The formal parameter type has to be written in brackets. (D) (I)
, actual operands for the parametrization of a function block	first character in the column; directly followed by the parameter ,I 1.0	
#I	Include file	with blanks, enter disk drive and the first six characters of the file name #I_A:EXERCI
#	Symbolic block name	

Operation

Column STATEMENTS Control characters	Identification for	**Writing conventions with examples**
#UB	Segment header	The control character is found in the STATEMENT column, the text for the heading in the STATEMENT COMMENT column
( )	Formal parameter type	The formal parameter type has to be written in brackets. (D) (I)
, actual operands for the parametrization of a function block	first character in the column; directly followed by the parameter ,I 1.0	
#I	Include file	with blanks, enter disk drive and the first six characters of the file name #I_A:EXERCI
#	Symbolic block name	

Position within the statement list	Explanations
only at the beginning of a segment.	These comment texts are also transferred into the program file. If you require further information about commenting a STEP 5 program, please refer to the STEP 5 description, in the 2nd volume of the manual of your programming unit.
directly below the block name	
within a block	
only at block limits: before the first block or between BE and #PBn	With this control character other files can be included. These files have to be available as intermediate files, however, i.e. either terminated in the STL editor using the ENTER key or created by a recompilation. It has to be observed that no identical file names occur in the files to be linked, as the last block overwrites the previous one with the same name when a program file is generated. During the compilation the Include file is linked to the preset symbol file. Therefore the symbol file has to supply the Include file with assignments.

Operation

Position within the statement list	Explanations
only at the beginning of a segment.	These comment texts are also transferred into the program file. If you require further information about commenting a STEP 5 program, please refer to the STEP 5 description, in the 2nd volume of the manual of your programming unit.
directly below the block name	
within a block	
only at block limits: before the first block or between BE and #PBn	With this control character other files can be included. These files have to be available as intermediate files, however, i.e. either terminated in the STL editor using the ENTER key or created by a recompilation. It has to be observed that no identical file names occur in the files to be linked, as the last block overwrites the previous one with the same name when a program file is generated. During the compilation the Include file is linked to the preset symbol file. Therefore the symbol file has to supply the Include file with assignments.

3

C79000-B8576-C677-02

3 - 17

Column ADR Control characters	**Identification for**	Writing conventions with examples
Q	segment comment	This control character is only placed at the beginning of a segment; if a segment heading exists it has to be positioned directly before it.
;	additional comment	The control character is placed in the ADDR column and the total screen witch is available for the text, independent from columns.

Operation

Column ADR Control characters	**Identification for**	Writing conventions with examples
Q	segment comment	This control character is only placed at the beginning of a segment; if a segment heading exists it has to be positioned directly before it.
;	additional comment	The control character is placed in the ADDR column and the total screen witch is available for the text, independent from columns.

3 - 18

C79000-B8576-C677-02

Position within the statement list	Explanations
at any place within a block	These additional comments only exist in the STL source file. They are ignored during the compilation. If you recompile into the same STL source file, these comments are lost.

Operation

Position within the statement list | Explanations

at any place within a block | These additional comments only exist in the STL source file. They are ignored during the compilation. If you recompile into the same STL source file, these comments are lost.

3

C79000-B8576-C677-02

3 - 19

3.3.3 STEP 5 Operations in the STL Editor/Batch Compiler and their Writing Conventions

All STEP 5 operations are possible in the STL editor/batch compiler. Only the language category of the programmable controller or the CPU is limited. Therefore, refer to the operation list of your device when programming.

	ADDRESS	STATEMENT
Operation with an absolute operand		Operation and absolute operand *A_I 1.2* entry without format
with symbolic operand		operation *A*
operation with data		operation and data format *L_KT* entry without format
data	address *11*	data format KH *KF* *KS* or *S* *KG* *KT* *KC* *KY* or *B* *KM*

Operation

3.3.3 STEP 5 Operations in the STL Editor/Batch Compiler and their Writing Conventions

All STEP 5 operations are possible in the STL editor/batch compiler. Only the language category of the programmable controller or the CPU is limited. Therefore, refer to the operation list of your device when programming.

	ADDRESS	STATEMENT
Operation with an absolute operand		Operation and absolute operand *A_I 1.2* entry without format
with symbolic operand		operation *A*
operation with data		operation and data format *L_KT* entry without format
data	address *11*	data format KH *KF* *KS* or *S* *KG* *KT* *KC* *KY* or *B* *KM*

3 - 20

C79000-B8576-C677-02

The following table, which corresponds to the screen columns, lists the writing conventions for absolute and symbolic programming.

OPERAND SYMBOL	STATEMENT COMMENT
	"open outside" push button
symbol *PB-OPEN O* without hyphen	
data value *005.2*	
value, 1 data word per line *6248* *+ 13512* 'display' only single quotation marks, up to 11 data words per line -1169368-38 1 double data word max. per line 123.1 735 125,018 00011100 11101111	

Operation

The following table, which corresponds to the screen columns, lists the writing conventions for absolute and symbolic programming.

OPERAND SYMBOL	STATEMENT COMMENT
	"open outside" push button
symbol *PB-OPEN O* without hyphen	
data value *005.2*	
value, 1 data word per line *6248* *+ 13512* 'display' only single quotation marks, up to 11 data words per line -1169368-38 1 double data word max. per line 123.1 735 125,018 00011100 11101111	

3

	ADDRESS	STATEMENT
Formal operands: definition	name *TIME* *PBO-O* *MAB* 4 characters max.	**type** *(D)* *(I)* *(Q)* in brackets
operation with formal operands	operation and formal operand *A_=PBO-O* *=_=MAB* entry without format; an equal sign has to be placed directly before the formal operand.	
actual operands (=parameters for FBs), absolute	control character with operand *,I1.2* *,DW1* without blanks	
symbolic	control character ,	
data	control character with data type *,KT*	
label	mark *ON* *M003*	
relative addresses, data word addresses	*17*	
block end		

Operation

	ADDRESS	STATEMENT
Formal operands: definition	name *TIME* *PBO-O* *MAB* 4 characters max.	**type** *(D)* *(I)* *(Q)* in brackets
operation with formal operands	operation and formal operand *A_=PBO-O* *=_=MAB* entry without format; an equal sign has to be placed directly before the formal operand.	
actual operands (=parameters for FBs), absolute	control character with operand *,I1.2* *,DW1* without blanks	
symbolic	control character ,	
data	control character with data type *,KT*	
label	mark *ON* *M003*	
relative addresses, data word addresses	*17*	
block end		

3 - 22

C79000-B8576-C677-02

OPERAND SYMBOL	STATEMENT COMMENT
symbol *MAB*	
value *005.2*	
BE	

Operation

OPERAND SYMBOL | STATEMENT COMMENT

symbol
MAB

value
005.2

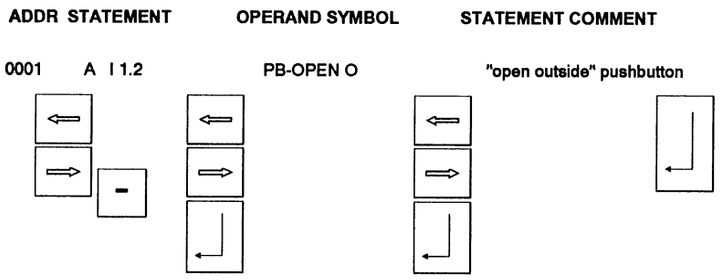
BE

3

C79000-B8576-C677-02

3 - 23

Moving from one column on the screen to the other is done by using the double arrow keys and return. If the return key is pressed the cursor is always moved to the 1st character of the STATEMENT column.



symbols

Please observe for symbolic programming that, in contrast to the CSF, LAD, STL package, a hyphen must not be placed before the symbol. A block beginning can only be entered as a symbol, if an assignment of the block type and number to a symbol exists. If not, the block beginning is programmed absolutely, e.g. #PB3, as the batch compiler requires the exact block type and its number for creating the intermediate file.

The symbols used in the STL editor have to be identical to those in the symbol file. This is also valid for blanks:

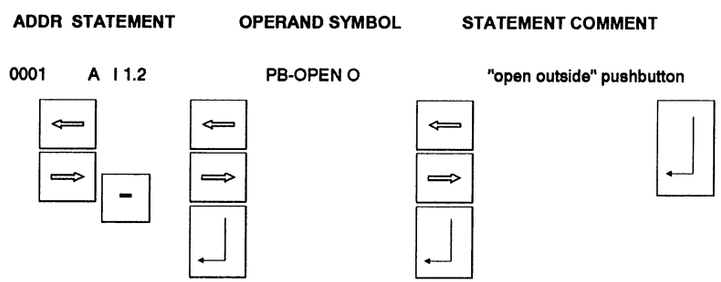
_EMEOFF ≠ EMEOFF

Further differences to the CSF, LAD, STL package are

- control characters,
- blanks for operations to be entered by the user,
- data constants and values are in different columns.

Operation

Moving from one column on the screen to the other is done by using the double arrow keys and return. If the return key is pressed the cursor is always moved to the 1st character of the STATEMENT column.



symbols

Please observe for symbolic programming that, in contrast to the CSF, LAD, STL package, a hyphen must not be placed before the symbol. A block beginning can only be entered as a symbol, if an assignment of the block type and number to a symbol exists. If not, the block beginning is programmed absolutely, e.g. #PB3, as the batch compiler requires the exact block type and its number for creating the intermediate file.

The symbols used in the STL editor have to be identical to those in the symbol file. This is also valid for blanks:

_EMEOFF ≠ EMEOFF

Further differences to the CSF, LAD, STL package are

- control characters,
- blanks for operations to be entered by the user,
- data constants and values are in different columns.

3.3.4 Entry of Program Blocks

Programming Example

The file on the next page is a programming example. By means of this example we will explain the operation of the STL editor/batch compiler and the functions of this package.

A garage door is being controlled by this program. It opens or closes from the outside with a key along with push-buttons and from the inside only "open" and "close" push-buttons have to be pressed. The door is closed after an interval of 5 seconds.

Operation

3.3.4 Entry of Program Blocks

Programming Example

The file on the next page is a programming example. By means of this example we will explain the operation of the STL editor/batch compiler and the functions of this package.

A garage door is being controlled by this program. It opens or closes from the outside with a key along with push-buttons and from the inside only "open" and "close" push-buttons have to be pressed. The door is closed after an interval of 5 seconds.

3

C79000-B8576-C677-02

3 - 25

```
STL source: B:TEST@@A0.SEQ

ADDR   STATEMENT                OPERAND SYMBOL          STATEMENT COMMENT
       #PB1
       #UB                                              OPEN FROM THE OUTSIDE AND INSIDE
*THE MOTOR IS STARTED UPWARDS WITH THE "OPEN OUTSIDE" PUSH BUTTON AND THE KEY SWITCH OR THE
"OPEN INSIDE"
*PUSH BUTTON. THE MOTOR KEEPS RUNNING  UNTIL THE UPPER END SWITCH IS
*REACHED OR THE EMERGENCY STOP SWITCH IS PRESSED.
*
       A(
       A       I  1.2                                   PUSH BUTTON OPEN OUTSIDE
       A       I  1.4                                   KEY SWITCH
       O       I  1.5                                   PUSH BUTTON OPEN INSIDE
       )
       AN      I  1.0                                   UPPER END SWITCH
       S       Q  1.0                                   MOTOR UPWARDS
               QQQ
       #UB                                              OPEN FROM THE OUTSIDE AND INSIDE
*RESETTING OF THE MOTOR UPWARDS OUTPUT.
       O       I 1.0                                    UPPER END SWITCH
       O       I 1.7                                    EMERGENCY STOP SWITCH
       R       Q 1.0                                    MOTOR UPWARDS
       QQQ
       #UB                                              CLOSE FROM THE OUTSIDE OR INSIDE
*THE "CLOSE OUTSIDE" PUSH BUTTON AND THE KEY SWITCH OR THE "CLOSE INSIDE" PUSH
*BUTTON START THE MOTOR DOWNWARDS WITH A START DELAY OF 5 SEC. THE
*MOTOR DOWNWARDS KEEPS RUNNING UNTIL THE LOWER END SWITCH IS
*REACHED OR THE EMERGENCY STOP KEY IS PRESSED.
       A(
       A                        PB-CLOSE O
       A                        KEY
       O                        PB-CLOSE I
       )
       AN                       END LOW
       L       KT               005.2
       SS                       START DELAY
       O                        END LOW
       O                        STOP
       R                        START DELAY
       L                        START DELAY
       T   MW  100
       LC                       START DELAY
       T   MW  102
       A                        START DELAY
       =                        MOT DOWN
       BE
```


Operation


```
STL source: B:TEST@@A0.SEQ

ADDR   STATEMENT                OPERAND SYMBOL          STATEMENT COMMENT
       #PB1
       #UB                                              OPEN FROM THE OUTSIDE AND INSIDE
*THE MOTOR IS STARTED UPWARDS WITH THE "OPEN OUTSIDE" PUSH BUTTON AND THE KEY SWITCH OR THE
"OPEN INSIDE"
*PUSH BUTTON. THE MOTOR KEEPS RUNNING  UNTIL THE UPPER END SWITCH IS
*REACHED OR THE EMERGENCY STOP SWITCH IS PRESSED.
*
       A(
       A       I  1.2                                   PUSH BUTTON OPEN OUTSIDE
       A       I  1.4                                   KEY SWITCH
       O       I  1.5                                   PUSH BUTTON OPEN INSIDE
       )
       AN      I  1.0                                   UPPER END SWITCH
       S       Q  1.0                                   MOTOR UPWARDS
               QQQ
       #UB                                              OPEN FROM THE OUTSIDE AND INSIDE
*RESETTING OF THE MOTOR UPWARDS OUTPUT.
       O       I 1.0                                    UPPER END SWITCH
       O       I 1.7                                    EMERGENCY STOP SWITCH
       R       Q 1.0                                    MOTOR UPWARDS
       QQQ
       #UB                                              CLOSE FROM THE OUTSIDE OR INSIDE
*THE "CLOSE OUTSIDE" PUSH BUTTON AND THE KEY SWITCH OR THE "CLOSE INSIDE" PUSH
*BUTTON START THE MOTOR DOWNWARDS WITH A START DELAY OF 5 SEC. THE
*MOTOR DOWNWARDS KEEPS RUNNING UNTIL THE LOWER END SWITCH IS
*REACHED OR THE EMERGENCY STOP KEY IS PRESSED.
       A(
       A                        PB-CLOSE O
       A                        KEY
       O                        PB-CLOSE I
       )
       AN                       END LOW
       L       KT               005.2
       SS                       START DELAY
       O                        END LOW
       O                        STOP
       R                        START DELAY
       L                        START DELAY
       T   MW  100
       LC                       START DELAY
       T   MW  102
       A                        START DELAY
       =                        MOT DOWN
       BE
```

Entry

conditions: The STL editor/batch compiler has been loaded, the presetting has been filled in and the editing function has been called.

Determine **MODE** (F8)

This function can select between two editing modes: insert or overwrite. In the screen header the PG displays which function is selected.

➤ Press **MODE** (F8), until the desired mode is activated.

block beginning Proceed as follows (The character sequences you enter are written in *italics*, the functions to be used in **bold** letters.):

➤ Enter *#PB1* as the block beginning,

➤ Press the **return key** twice; by inserting this blank line the program will be optically structured during the entry,

➤ *#UB* for the heading of the first segment,

➤ press the **double arrow key right** twice to get into the STATEMENT COMMENT column,

➤ *Open from outside or inside*

➤ press **return key,**

➤ press the **double arrow key left** once to get into the ADDR column,

➤ enter ✱ as the control character for the segment comment.

Now you can enter the first text of the example. The whole screen width is available for this entry. Terminate each line with **return**. To begin a new text line you begin as described above with the **double arrow key left** and ✱, as the cursor only jumps automatically into the STATEMENT column.

If you write in the insert mode, observe the end of the line! You can only insert within one line, otherwise the text can slip past the end of the line and thus be lost.

The cursor and special keys which are described above (chapter 3.3.1.3) are available for editing your text. The ✱ control character, however, cannot be removed by "delete character" but only by means of the **DELETE** and **LINE** functions (as below, chapter 3.3.5.1, section Line Editing).

Operation

Entry

conditions: The STL editor/batch compiler has been loaded, the presetting has been filled in and the editing function has been called.

Determine **MODE** (F8)

This function can select between two editing modes: insert or overwrite. In the screen header the PG displays which function is selected.

➤ Press **MODE** (F8), until the desired mode is activated.

block beginning Proceed as follows (The character sequences you enter are written in *italics*, the functions to be used in **bold** letters.):

➤ Enter *#PB1* as the block beginning,

➤ Press the **return key** twice; by inserting this blank line the program will be optically structured during the entry,

➤ *#UB* for the heading of the first segment,

➤ press the **double arrow key right** twice to get into the STATEMENT COMMENT column,

➤ *Open from outside or inside*

➤ press **return key,**

➤ press the **double arrow key left** once to get into the ADDR column,

➤ enter ✱ as the control character for the segment comment.

Now you can enter the first text of the example. The whole screen width is available for this entry. Terminate each line with **return**. To begin a new text line you begin as described above with the **double arrow key left** and ✱, as the cursor only jumps automatically into the STATEMENT column.

If you write in the insert mode, observe the end of the line! You can only insert within one line, otherwise the text can slip past the end of the line and thus be lost.

The cursor and special keys which are described above (chapter 3.3.1.3) are available for editing your text. The ✱ control character, however, cannot be removed by "delete character" but only by means of the **DELETE** and **LINE** functions (as below, chapter 3.3.5.1, section Line Editing).

absolute operands

The segment comment is followed by the individual statements for opening the garage door, at first in the absolute way with statement comments. Please observe, that you have to enter the blanks yourself.

➤ *A(*
➤ press the **return key,**
➤ *A*
➤ blank
➤ *I 1.2*
➤ press the **double arrow key right** twice in order to enter the
➤ *push-button open outside* comment.
➤ press the **return key,**
➤ *A*
➤ blank
➤ *I 1.4*
➤ press the **double arrow key right** twice in order to enter the
➤ *key switch* comment.
➤ press the **return key,**

and so on.

Enter three asterisks for the end of the segment.

Save what you have entered up to now using the

➤ **BACK-UP** (F7) function.

BACK-UP (F7)

With this function you can save your STL source file without leaving the editor. That means, you can save intermittantly or temporarily stop editing without problems.

In this respect the CSF, LAD, STL packages differ, in which you always have to leave the editor when saving and you have to select output in order to continue your work.

Operation

absolute operands

The segment comment is followed by the individual statements for opening the garage door, at first in the absolute way with statement comments. Please observe, that you have to enter the blanks yourself.

➤ *A(*
➤ press the **return key,**
➤ *A*
➤ blank
➤ *I 1.2*
➤ press the **double arrow key right** twice in order to enter the
➤ *push-button open outside* comment.
➤ press the **return key,**
➤ *A*
➤ blank
➤ *I 1.4*
➤ press the **double arrow key right** twice in order to enter the
➤ *key switch* comment.
➤ press the **return key,**

and so on.

Enter three asterisks for the end of the segment.

Save what you have entered up to now using the

➤ **BACK-UP** (F7) function.

BACK-UP (F7)

With this function you can save your STL source file without leaving the editor. That means, you can save intermittantly or temporarily stop editing without problems.

In this respect the CSF, LAD, STL packages differ, in which you always have to leave the editor when saving and you have to select output in order to continue your work.

3 - 28 C79000-B8576-C677-02

symbolic operands

Now the statements for closing the door follow in segment 3, this time programmed symbolically.

Enter the heading and the comment again as described above. With the symbolic operands proceed as follows:

- *A*
- press the **double arrow key right** or **'-' (hyphen)** once, the cursor is now in the operand symbol column.
- *PB-CLOSE 0* (without introductory hyphen!)
- press the **return key**
- *A*
- press the **double arrow key right** once in order to enter the
- *key* symbol,
- press the **return key**.

If you have reached the block end BE, save your data with

- **BACK-UP** (F7).

In the following section we will show you how to handle the individual functions in order to edit a statement list. A file in the symbol editor is also edited or corrected in the editing function.

Operation

symbolic operands

Now the statements for closing the door follow in segment 3, this time programmed symbolically.

Enter the heading and the comment again as described above. With the symbolic operands proceed as follows:

- *A*
- press the **double arrow key right** or **'-' (hyphen)** once, the cursor is now in the operand symbol column.
- *PB-CLOSE 0* (without introductory hyphen!)
- press the **return key**
- *A*
- press the **double arrow key right** once in order to enter the
- *key* symbol,
- press the **return key**.

If you have reached the block end BE, save your data with

- **BACK-UP** (F7).

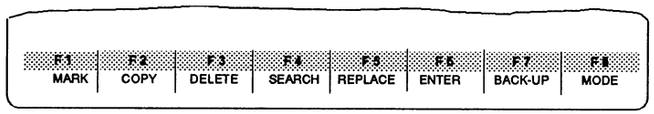
In the following section we will show you how to handle the individual functions in order to edit a statement list. A file in the symbol editor is also edited or corrected in the editing function.

3

C79000-B8576-C677-02

3 - 29

3.3.5 Using the EDIT softkeys



In explaining how the editing functions operate we will present you with the interplay of **BUFFER, COPY** and **DELETE** and show, how you can save parts of your program for further processing and reprocessing in files and how you can assign character sequences to function keys.

The repetition factor is a useful tool: after calling a function a number is entered via the typewriter keyboard of your programmable unit. The subsequent function is carried out with this factor, e.g. copy a line 7 times.

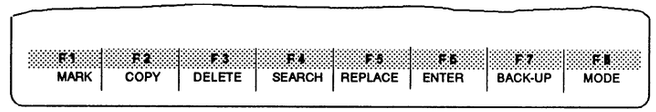
The function that has been activated is displayed in the heading above the STATEMENT COMMENT. A function must always be exited with **RETURN** (F8), before work can be continued in the text.

A process is aborted within a function by using the **abort key** and/or the F8 **RETURN** key.

ATTENTION
By using the abort key you might lose data! E.g. if you correct a file and then press the abort key all modifications will be lost.

Operation

3.3.5 Using the EDIT softkeys



In explaining how the editing functions operate we will present you with the interplay of **BUFFER, COPY** and **DELETE** and show, how you can save parts of your program for further processing and reprocessing in files and how you can assign character sequences to function keys.

The repetition factor is a useful tool: after calling a function a number is entered via the typewriter keyboard of your programmable unit. The subsequent function is carried out with this factor, e.g. copy a line 7 times.

The function that has been activated is displayed in the heading above the STATEMENT COMMENT. A function must always be exited with **RETURN** (F8), before work can be continued in the text.

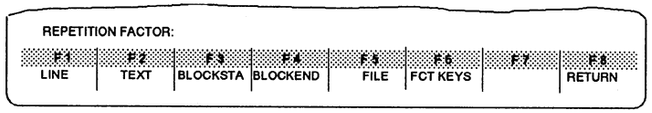
A process is aborted within a function by using the **abort key** and/or the F8 **RETURN** key.

ATTENTION
By using the abort key you might lose data! E.g. if you correct a file and then press the abort key all modifications will be lost.

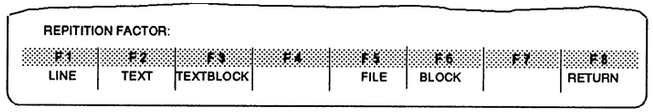
The BUFFER, COPY and DELETE Functions

BUFFER With this function parts of text can be buffered. Character sequences (40 characters max.), single lines and character blocks (500 lines max.) are each written into a separate buffer and can be copied to any place. The text stored in the buffer can also be saved in a sequential file.

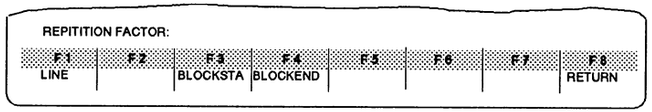
Via the buffer function character sequences are also assigned to function keys (40 characters max.).



COPY Texts stored in the buffer or in sequential files (buffer file, source file) are inserted at the cursor position, when this function is applied. You can also copy a complete STL source file at the cursor position.



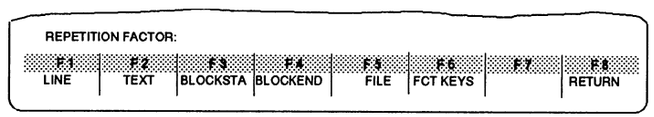
DELETE With the delete menu you can delete single lines and marked text blocks. For safety reasons the deleted text is written into the buffer. However, previously buffered text is lost.



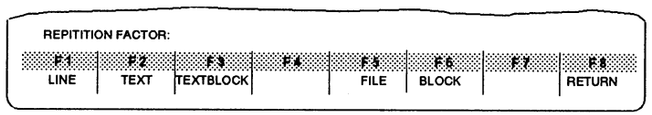
The BUFFER, COPY and DELETE Functions

BUFFER With this function parts of text can be buffered. Character sequences (40 characters max.), single lines and character blocks (500 lines max.) are each written into a separate buffer and can be copied to any place. The text stored in the buffer can also be saved in a sequential file.

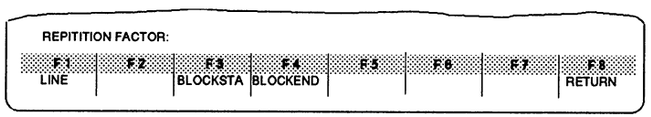
Via the buffer function character sequences are also assigned to function keys (40 characters max.).



COPY Texts stored in the buffer or in sequential files (buffer file, source file) are inserted at the cursor position, when this function is applied. You can also copy a complete STL source file at the cursor position.



DELETE With the delete menu you can delete single lines and marked text blocks. For safety reasons the deleted text is written into the buffer. However, previously buffered text is lost.



Line editing

E.g. the heading of the first segment should be identical to the heading of the second one.

1. Buffer
- ➤ Position cursor on this line,
- ➤ call **BUFFER** (F1),
- ➤ press **LINE** (F1); this line is now in the buffer,
- ➤ leave the buffer function with **RETURN** (F8).

2. Copy
- ➤ Position the cursor on the second segment at its old heading,
- ➤ call **COPY** (F2) (observe writing mode! as below, note),
- ➤ press **LINE** (F1); the new heading is inserted,
- ➤ leave the copy function with **RETURN** (F8).

3. Delete
- ➤ Position cursor on the old heading,
- ➤ call **DELETE** (F3),
- ➤ press **LINE** (F1); the old heading is deleted,
- ➤ leave the delete function with **RETURN** (F8).

4. Move
- ➤ Position the cursor on the line to be moved,
- ➤ call **DELETE** (F3),
- ➤ press **LINE** (F1); the text of this line is now in the buffer.
- ➤ Position the cursor on the position, to which the line is to be moved,
- ➤ call **COPY** (F2),
- ➤ press **LINE** (F1); the line is inserted.
- ➤ terminate the copy function with **RETURN** (F8).

Operation

Line editing

E.g. the heading of the first segment should be identical to the heading of the second one.

1. Buffer
- ➤ Position cursor on this line,
- ➤ call **BUFFER** (F1),
- ➤ press **LINE** (F1); this line is now in the buffer,
- ➤ leave the buffer function with **RETURN** (F8).

2. Copy
- ➤ Position the cursor on the second segment at its old heading,
- ➤ call **COPY** (F2) (observe writing mode! as below, note),
- ➤ press **LINE** (F1); the new heading is inserted,
- ➤ leave the copy function with **RETURN** (F8).

3. Delete
- ➤ Position cursor on the old heading,
- ➤ call **DELETE** (F3),
- ➤ press **LINE** (F1); the old heading is deleted,
- ➤ leave the delete function with **RETURN** (F8).

4. Move
- ➤ Position the cursor on the line to be moved,
- ➤ call **DELETE** (F3),
- ➤ press **LINE** (F1); the text of this line is now in the buffer.
- ➤ Position the cursor on the position, to which the line is to be moved,
- ➤ call **COPY** (F2),
- ➤ press **LINE** (F1); the line is inserted.
- ➤ terminate the copy function with **RETURN** (F8).

3 - 32

C79000-B8576-C677-02

Note
A line, text and block are each stored in a separate buffer when the **BUFFER** function is used. Therefore, a line, text or block in their respective buffers is only overwritten by an incoming line, text or block that is to be buffered. In the buffer a buffered line is only overwritten by a newly buffered line, a text only by a newly buffered text and a block only by a newly buffered block. That means, you can buffer three character sequences in parallel.
When copying, position the cursor before the **COPY** function is called. Please also observe the writing mode: in the insert mode buffered text is inserted, in the overwrite mode the text from the cursor onwards is overwritten with the buffered text. Therefore, you have to provide for space before you insert in the overwrite mode.
A line, text and block are each stored in a separate buffer when the **BUFFER** function is used. Therefore, a line, text or block in their respective buffers is only overwritten by an incoming line, text or block that is to be buffered. In the buffer a buffered line is only overwritten by a newly buffered line, a text only by a newly buffered text and a block only by a newly buffered block. That means, you can buffer three character sequences in parallel.
Control characters can only be deleted with **DELETE LINE**.

Block editing (500 lines max.)

1. Buffer and copy e.g. a segment comment

- ➤ call **BUFFER** (F1),
- ➤ position the cursor at the beginning of a segment comment,
- ➤ mark with **BLOCK STArt** (F3),
- ➤ position the cursor at the end of the comment,
- ➤ mark it with **BLOCK END** (F4),
- ➤ leave the buffer function with **RETURN** (F8).
- ➤ call **COPY** (F2),

Operation

Note
A line, text and block are each stored in a separate buffer when the **BUFFER** function is used. Therefore, a line, text or block in their respective buffers is only overwritten by an incoming line, text or block that is to be buffered. In the buffer a buffered line is only overwritten by a newly buffered line, a text only by a newly buffered text and a block only by a newly buffered block. That means, you can buffer three character sequences in parallel.
When copying, position the cursor before the **COPY** function is called. Please also observe the writing mode: in the insert mode buffered text is inserted, in the overwrite mode the text from the cursor onwards is overwritten with the buffered text. Therefore, you have to provide for space before you insert in the overwrite mode.
A line, text and block are each stored in a separate buffer when the **BUFFER** function is used. Therefore, a line, text or block in their respective buffers is only overwritten by an incoming line, text or block that is to be buffered. In the buffer a buffered line is only overwritten by a newly buffered line, a text only by a newly buffered text and a block only by a newly buffered block. That means, you can buffer three character sequences in parallel.
Control characters can only be deleted with **DELETE LINE**.

3

Block editing (500 lines max.)

1. Buffer and copy e.g. a segment comment

- ➤ call **BUFFER** (F1),
- ➤ position the cursor at the beginning of a segment comment,
- ➤ mark with **BLOCK STArt** (F3),
- ➤ position the cursor at the end of the comment,
- ➤ mark it with **BLOCK END** (F4),
- ➤ leave the buffer function with **RETURN** (F8).
- ➤ call **COPY** (F2),

C79000-B8576-C677-02 3 - 33

➤ position the cursor e.g. at the end of the file,

➤ press **BLOCK** (F3); the comment is inserted here,

➤ leave the copy function with **RETURN** (F8).

2. Buffer in a file e.g. store the TURN ON delay in the TIMER@ file.

➤ Call **BUFFER** (F1),

➤ position cursor on line L KT 005.2,

➤ press **BLOCK STArt** (F3),

➤ move cursor down to the end of the timer, line T1 START DELAY,

➤ press **BLOCK END** (F4); this block is now buffered.

➤ call **FILE** (F5) and

➤ fill in with *TIMER@*,

➤ press **return key** and

➤ **ENTER** (F6). The TURN ON delay is now in the TIMER@A0.SEQ file

➤ leave the buffer function with **RETURN** (F8).

➤ Position the cursor behind the block end (BE),

➤ call **COPY** (F2),

➤ call **FILE** (F5) and enter the file to be copied. We transfer the displayed B:TIMER@A0.SEQ file,

➤ start the copy process with **ENTER** (F6) or with **the ENTER key.**

The TURN ON delay is inserted.

➤ Leave the copy function with **RETURN** (F8).

3. Delete The inserted parts are to be deleted.

➤ Call **DELETE** (F3). Mark the TURN ON delay

➤ with **BLOCK STArt** (F3) and

➤ **BLOCK END** (F4); the text is deleted.

➤ Go **RETURN** (F8).

You can e.g. also delete the segment comment line per line using the repetition factor.

Operation

➤ position the cursor e.g. at the end of the file,

➤ press **BLOCK** (F3); the comment is inserted here,

➤ leave the copy function with **RETURN** (F8).

2. Buffer in a file e.g. store the TURN ON delay in the TIMER@ file.

➤ Call **BUFFER** (F1),

➤ position cursor on line L KT 005.2,

➤ press **BLOCK STArt** (F3),

➤ move cursor down to the end of the timer, line T1 START DELAY,

➤ press **BLOCK END** (F4); this block is now buffered.

➤ call **FILE** (F5) and

➤ fill in with *TIMER@*,

➤ press **return key** and

➤ **ENTER** (F6). The TURN ON delay is now in the TIMER@A0.SEQ file

➤ leave the buffer function with **RETURN** (F8).

➤ Position the cursor behind the block end (BE),

➤ call **COPY** (F2),

➤ call **FILE** (F5) and enter the file to be copied. We transfer the displayed B:TIMER@A0.SEQ file,

➤ start the copy process with **ENTER** (F6) or with **the ENTER key.**

The TURN ON delay is inserted.

➤ Leave the copy function with **RETURN** (F8).

3. Delete The inserted parts are to be deleted.

➤ Call **DELETE** (F3). Mark the TURN ON delay

➤ with **BLOCK STArt** (F3) and

➤ **BLOCK END** (F4); the text is deleted.

➤ Go **RETURN** (F8).

You can e.g. also delete the segment comment line per line using the repetition factor.

➤ Position the cursor,
➤ call **DELETE** (F3),
➤ enter a factor (e.g. 4) via the typewriter keyboard
➤ press **LINE** (F1).

The (four) lines are deleted.

➤ Go **RETURN** (F8).

4. Move Delete the block to be moved using

➤ **DELETE** (F3),
➤ **BLOCK STArt** (F3) and
➤ **BLOCK END** (F4) and write it into the buffer.
➤ Go **RETURN** (F8).
➤ Place the cursor on the new position,
➤ call **COPY** (F2) and
➤ press **BLOCK** (F3). The text has been moved.
➤ Go **RETURN** (F8).

Editing a character sequence or a text (40 characters max.)

Example: if you want to insert or edit a text or a statement (subsequently) at several places, if a statement comment needs to be repeated many times, which you do not want to rewrite again and again, then can these texts be stored in the buffer.

1. Buffer
➤ Call **BUFFER** (F1),
➤ press **TEXT** (F2),
➤ enter text between the colons, e.g. *A I 2.0.*
➤ Press **ENTER** (F6).This text is now stored in the buffer.
➤ leave the buffer function with **RETURN**.

Text between colons can be deleted using the +1 key.

2. Copy
➤ Position the cursor,
➤ call **COPY** (F2) (observe writing mode, as above, note),
➤ move cursor to the place to be edited,
➤ press **TEXT** (F2) and the buffered text is inserted.
➤ Leave the copy function with **RETURN** (F8).

Operation

➤ Position the cursor,
➤ call **DELETE** (F3),
➤ enter a factor (e.g. 4) via the typewriter keyboard
➤ press **LINE** (F1).

The (four) lines are deleted.

➤ Go **RETURN** (F8).

4. Move Delete the block to be moved using

➤ **DELETE** (F3),
➤ **BLOCK STArt** (F3) and
➤ **BLOCK END** (F4) and write it into the buffer.
➤ Go **RETURN** (F8).
➤ Place the cursor on the new position,
➤ call **COPY** (F2) and
➤ press **BLOCK** (F3). The text has been moved.
➤ Go **RETURN** (F8).

Editing a character sequence or a text (40 characters max.)

Example: if you want to insert or edit a text or a statement (subsequently) at several places, if a statement comment needs to be repeated many times, which you do not want to rewrite again and again, then can these texts be stored in the buffer.

1. Buffer
➤ Call **BUFFER** (F1),
➤ press **TEXT** (F2),
➤ enter text between the colons, e.g. *A I 2.0.*
➤ Press **ENTER** (F6).This text is now stored in the buffer.
➤ leave the buffer function with **RETURN**.

Text between colons can be deleted using the +1 key.

2. Copy
➤ Position the cursor,
➤ call **COPY** (F2) (observe writing mode, as above, note),
➤ move cursor to the place to be edited,
➤ press **TEXT** (F2) and the buffered text is inserted.
➤ Leave the copy function with **RETURN** (F8).

Function Key Assignment (with 40 characters max.)

Example: end of segment, being a frequently occuring character sequence, is to be assigned to a function key.

- ➤ Call **BUFFER** (F1),
- ➤ output the function key list with **FCT KEYS** (F6). The cursor flashes in line f1.
- ➤ Enter ******* between the colons,
- ➤ **ENTER** with F6.

This character sequence can be called at any place in your file if you press the Shift and F1 keys for the PG 695, PG 695II and PG 750 and the "opener" key for the PG 685.

Enter e.g. after an end of segment another one:

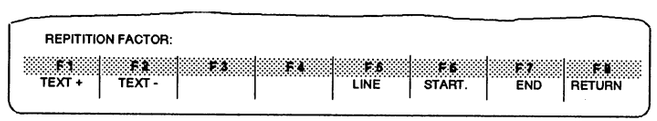
- ➤ position the cursor on *******,
- ➤ press the **SHIFT** and **F1** keys.

Now delete all these alterations:

- ➤ press the **abort key** and
- ➤ confirm with the **ENTER key**. In the output FUNCTION SELECTION
- ➤ call **EDIT** (F1) again.

The SEARCH Function

With the **SEARCH** function you can jump to the beginning and end of your file and to certain lines. Line 0 is not allowed. Furthermore, you can search for any character sequenceup to 20 characters (words and numbers), which is written in a field. Furthermore, you can search for any character sequence (words and figures), which is written in the ADR, OPERAND SYMBOL and STATEMENT COMMENT columns. With TEXT + these charcter sequences are searched from the cursor position on and with TEXT - backwards. Using **SEARCH** you can move easily and quickly within your file.



Operation

Function Key Assignment (with 40 characters max.)

Example: end of segment, being a frequently occuring character sequence, is to be assigned to a function key.

- ➤ Call **BUFFER** (F1),
- ➤ output the function key list with **FCT KEYS** (F6). The cursor flashes in line f1.
- ➤ Enter ******* between the colons,
- ➤ **ENTER** with F6.

This character sequence can be called at any place in your file if you press the Shift and F1 keys for the PG 695, PG 695II and PG 750 and the "opener" key for the PG 685.

Enter e.g. after an end of segment another one:

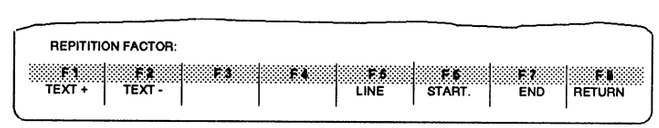
- ➤ position the cursor on *******,
- ➤ press the **SHIFT** and **F1** keys.

Now delete all these alterations:

- ➤ press the **abort key** and
- ➤ confirm with the **ENTER key**. In the output FUNCTION SELECTION
- ➤ call **EDIT** (F1) again.

The SEARCH Function

With the **SEARCH** function you can jump to the beginning and end of your file and to certain lines. Line 0 is not allowed. Furthermore, you can search for any character sequenceup to 20 characters (words and numbers), which is written in a field. Furthermore, you can search for any character sequence (words and figures), which is written in the ADR, OPERAND SYMBOL and STATEMENT COMMENT columns. With TEXT + these charcter sequences are searched from the cursor position on and with TEXT - backwards. Using **SEARCH** you can move easily and quickly within your file.



Please note, that the text to be searched for must be equivalent to the entered character sequence regarding upper/lower case letters. That means, you must enter the texts in the "ADDR" and "statement" fields in capital letters!

Example: you can jump to the end with

➤ **END** (F7); the cursor is positioned below the BE;

to the beginning with

➤ **START** (F6); the cursor is positioned on the PC type (if existing).

in a new line with

➤ **LINE** (F5),

➤ You enter the number e.g. *12*,

➤ and **ENTER**. The cursor flashes in line 12.

A certain term, an address or an operand from the cursor position on is searched with

➤ **TEXT+** (F1)

➤ You enter the character sequence, e.g. *I1.0,*

➤ **ENTER** (F6.)Now the cursor will jump to the end position of the next character sequence searched for that will be found after the actual cursor position.

and from the cursor position backwards with

➤ **TEXT-** (F2)

➤ Here you also enter the character sequence, e.g. *BUTTON OPEN*

➤ and **ENTER** (F6).

Now the cursor jumps to the end position of the next character sequence searched for that will be found before the actual cursor position.

Operation

Please note, that the text to be searched for must be equivalent to the entered character sequence regarding upper/lower case letters. That means, you must enter the texts in the "ADDR" and "statement" fields in capital letters!

Example: you can jump to the end with

➤ **END** (F7); the cursor is positioned below the BE;

to the beginning with

➤ **START** (F6); the cursor is positioned on the PC type (if existing).

in a new line with

➤ **LINE** (F5),

➤ You enter the number e.g. *12*,

➤ and **ENTER**. The cursor flashes in line 12.

A certain term, an address or an operand from the cursor position on is searched with

➤ **TEXT+** (F1)

➤ You enter the character sequence, e.g. *I1.0,*

➤ **ENTER** (F6.)Now the cursor will jump to the end position of the next character sequence searched for that will be found after the actual cursor position.

and from the cursor position backwards with

➤ **TEXT-** (F2)

➤ Here you also enter the character sequence, e.g. *BUTTON OPEN*

➤ and **ENTER** (F6).

Now the cursor jumps to the end position of the next character sequence searched for that will be found before the actual cursor position.

3

If you want to continue the search for the entered term, reactivate the selected function:

➤ e.g. call **TEXT-** and

➤ **ENTER**; the cursor again jumps to the next character sequence consisting of the entered term in the desired direction.

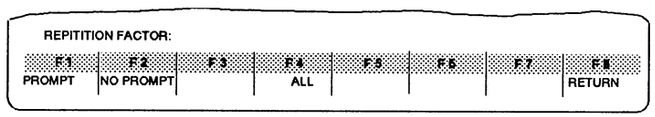
At the end of the search process terminate with

➤ **RETURN** (F8).

The REPLACE Function

Any character sequence up to 20 characters (words or figures) in the ADDR, OPERAND SYMBOL and STATEMENT COMMENT columns can be replaced by other ones. You can select between single replacement with or without confirmation and total replacement. Single replacement can only be done from the cursor position downwards. Therefore, position your cursor for this function at least one line above.

Please note, that the text to be searched for must be equivalent to the entered character sequence regarding upper/lower case letters. That means, you must enter the texts in the "ADDR" and "statement" fields in capital letters!



Using **REPLACE** you can quickly correct your file. For example: if a statement or a symbol is to be exchanged in the whole file. Please observe that the character sequence which is searched for has to be spelled correctly, including the blanks.

➤ call **REPLACE** (F5),

➤ press **ALL** (F4),

➤ enter the former character sequence to be exchanged, e.g. *ON DEL*

➤ press **ENTER** (F6)

Operation

If you want to continue the search for the entered term, reactivate the selected function:

➤ e.g. call **TEXT-** and

➤ **ENTER**; the cursor again jumps to the next character sequence consisting of the entered term in the desired direction.

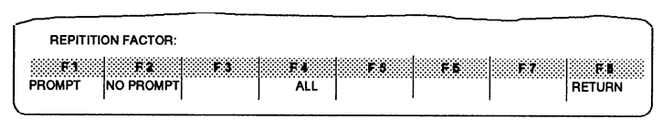
At the end of the search process terminate with

➤ **RETURN** (F8).

The REPLACE Function

Any character sequence up to 20 characters (words or figures) in the ADDR, OPERAND SYMBOL and STATEMENT COMMENT columns can be replaced by other ones. You can select between single replacement with or without confirmation and total replacement. Single replacement can only be done from the cursor position downwards. Therefore, position your cursor for this function at least one line above.

Please note, that the text to be searched for must be equivalent to the entered character sequence regarding upper/lower case letters. That means, you must enter the texts in the "ADDR" and "statement" fields in capital letters!



Using **REPLACE** you can quickly correct your file. For example: if a statement or a symbol is to be exchanged in the whole file. Please observe that the character sequence which is searched for has to be spelled correctly, including the blanks.

➤ call **REPLACE** (F5),

➤ press **ALL** (F4),

➤ enter the former character sequence to be exchanged, e.g. *ON DEL*

➤ press **ENTER** (F6)

➤ enter the new character sequence, e.g. *SS-TIMER*

➤ and **ENTER** (F6). Now all the ON DEL symbols are replaced by SS-TIMER.

exchange a charactere sequence

If a character sequence is to be exchanged only once you can check the exchange by selecting **PROMPT**; if you select **NO PROMPT** this exchange is immediately carried out. Here the cursor has to be positioned above the character sequence to be replaced because replacements occur for single exchange from the cursor downwards.

➤ Call **REPLACE** (F5),

➤ press **PROMPT** (F1),

➤ enter the former character sequence to be exchanged, e.g. *OUTSIDE*

➤ press **ENTER** (F6),

➤ enter the new character sequence, e.g. *ZZZZZZ*,

➤ and **ENTER**. Now you can confirm with **YES** (F1) or **NO** (F3),

➤ press **YES** (F1). The "outside" character sequence next to the cursor is exchanged.

If "outside" cannot be found below the cursor position, then "not found" is displayed in the message line. Move the cursor to the beginning of the file and repeat the process with **REPLACE** (F5), **PROMPT** (F1), **ENTER key** twice and **YES** (F1). **NO PROMPT** is carried out correspondingly, only without confirmation.

repetition factor

You can combine the single exchange with the repetition factor. During this multiple exchange also the character sequences of the comments are also addressed. They are counted but not replaced, not even if you call **PROMPT, YES**. Using the **abort key** you can end the exchange before it is finished. The character sequences which have already been exchanged remain as they are.

3

Operation

➤ enter the new character sequence, e.g. *SS-TIMER*

➤ and **ENTER** (F6). Now all the ON DEL symbols are replaced by SS-TIMER.

exchange a charactere sequence

If a character sequence is to be exchanged only once you can check the exchange by selecting **PROMPT**; if you select **NO PROMPT** this exchange is immediately carried out. Here the cursor has to be positioned above the character sequence to be replaced because replacements occur for single exchange from the cursor downwards.

➤ Call **REPLACE** (F5),

➤ press **PROMPT** (F1),

➤ enter the former character sequence to be exchanged, e.g. *OUTSIDE*

➤ press **ENTER** (F6),

➤ enter the new character sequence, e.g. *ZZZZZZ*,

➤ and **ENTER**. Now you can confirm with **YES** (F1) or **NO** (F3),

➤ press **YES** (F1). The "outside" character sequence next to the cursor is exchanged.

If "outside" cannot be found below the cursor position, then "not found" is displayed in the message line. Move the cursor to the beginning of the file and repeat the process with **REPLACE** (F5), **PROMPT** (F1), **ENTER key** twice and **YES** (F1). **NO PROMPT** is carried out correspondingly, only without confirmation.

repetition factor

You can combine the single exchange with the repetition factor. During this multiple exchange also the character sequences of the comments are also addressed. They are counted but not replaced, not even if you call **PROMPT, YES**. Using the **abort key** you can end the exchange before it is finished. The character sequences which have already been exchanged remain as they are.

3

C79000-B8576-C677-02

3 - 39

➤ Leave the **REPLACE** function with **RETURN** (F8) and delete all alterations using the

➤ **abort key**, acknowledge with the

➤ **ENTER key**. Now you are back in the FUNCTION SELECTION.

Now output your original file again:

➤ **EDIT** (F1)

BACK-UP and ENTER Storage Functions

the functions BACK-UP and ENTER

You are already familiar with the **BACK-UP** function. It can be used for saving and for interrupting a session and starting it again, without having to output the file again.

With the **ENTER** function (f6) or the **ENTER key** the file is stored, the corresponding intermediate file is automatically generated and the editing session is terminated. During the compilation the created statement list is checked. If more than one error occurs, an error list is created. If only one error occurs, its position is shown on the screen and you can correct it according to the error message in the footer.

You will find further information on the error list in chapter 3.4.

➤ Save the file with **ENTER** (F6): it is compiled and the editing is terminated. The screen FUNCTION SELECTION is shown.

Operation

➤ Leave the **REPLACE** function with **RETURN** (F8) and delete all alterations using the

➤ **abort key**, acknowledge with the

➤ **ENTER key**. Now you are back in the FUNCTION SELECTION.

Now output your original file again:

➤ **EDIT** (F1)

BACK-UP and ENTER Storage Functions

the functions BACK-UP and ENTER

You are already familiar with the **BACK-UP** function. It can be used for saving and for interrupting a session and starting it again, without having to output the file again.

With the **ENTER** function (f6) or the **ENTER key** the file is stored, the corresponding intermediate file is automatically generated and the editing session is terminated. During the compilation the created statement list is checked. If more than one error occurs, an error list is created. If only one error occurs, its position is shown on the screen and you can correct it according to the error message in the footer.

You will find further information on the error list in chapter 3.4.

➤ Save the file with **ENTER** (F6): it is compiled and the editing is terminated. The screen FUNCTION SELECTION is shown.

3 - 40

C79000-B8576-C677-02

3.3.6 Input of Function Blocks

Example The B:FBTESTA0.SEQ file, printed out on the next page, is used as a practice example. Again it is the control of a garage door, but this time programmed as a function block, in order to show you the different ways these block types are edited.

Here the program call is to be programmed symbolically. Therefore, please create the following assignment list in the TEST@@Z0.INI symbol file so that the compilation will work. The handling of the symbol editor can be found in the Introduction and the Function Block chapter in the STEP 5 description, manual volume 2.


```
SEQ. DATEI: TEST@@Z0.INI

I1.0            END UPPER              UPPER END SWITCH
I1.1            END LOWER              LOWER END SWITCH
I1.2            PB-O O                 PUSH BUTTON OPEN OUTSIDE
I1.3            PB-C O                    PUSH BUTTON CLOSE OUTSIDE
I1.4            KEY                       KEY SWITCH OUTSIDE
I1.5            PB-OPEN I                 PUSH BUTTON OPEN INSIDE
I1.6            PB-CLOSE I             PUSH BUTTON CLOSE INSIDE
I1.7            STOP                      EMERGENCY STOP SWITCH
Q1.0            MOT UP                 MOTOR UPWARDS
Q1.1            MOT DOWN               MOTOR DOWNWARDS
T1              START DELAY            START DELAY; 5SEC.
FB1             GARAGE                 FB FOR CONTROLLING A GARAGE DOOR
```

If you require general information on function blocks please refer to the Introduction and the Function Block chapter in the STEP 5 description, manual volume 2 of your programming unit.

Operation

3.3.6 Input of Function Blocks

Example The B:FBTESTA0.SEQ file, printed out on the next page, is used as a practice example. Again it is the control of a garage door, but this time programmed as a function block, in order to show you the different ways these block types are edited.

Here the program call is to be programmed symbolically. Therefore, please create the following assignment list in the TEST@@Z0.INI symbol file so that the compilation will work. The handling of the symbol editor can be found in the Introduction and the Function Block chapter in the STEP 5 description, manual volume 2.

3

SEQ. DATEI: TEST@@Z0.INI

I1.0 END UPPER UPPER END SWITCH
I1.1 END LOWER LOWER END SWITCH
I1.2 PB-O O PUSH BUTTON OPEN OUTSIDE
I1.3 PB-C O PUSH BUTTON CLOSE OUTSIDE
I1.4 KEY KEY SWITCH OUTSIDE
I1.5 PB-OPEN I PUSH BUTTON OPEN INSIDE
I1.6 PB-CLOSE I PUSH BUTTON CLOSE INSIDE
I1.7 STOP EMERGENCY STOP SWITCH
Q1.0 MOT UP MOTOR UPWARDS
Q1.1 MOT DOWN MOTOR DOWNWARDS
T1 START DELAY START DELAY; 5SEC.
FB1 GARAGE FB FOR CONTROLLING A GARAGE DOOR

If you require general information on function blocks please refer to the Introduction and the Function Block chapter in the STEP 5 description, manual volume 2 of your programming unit.

C79000-B8576-C677-02

3 - 41

```
STL source: B: FBTESTA0.SEQ

ADDR            STATEMENT        OPERAND SYMBOL      STATEMENT COMMENT
                #                GARAGE              FB 1 FOR A GARAGE DOOR
                #N  GARAGE
ENDU            (I)                                  UPPER END SWITCH
ENDL            (I)                                  LOWER END SWITCH
PBO-I           (I)                                  PUSH BUTTON OPEN INSIDE
PBO-O           (I)                                  PUSH BUTTON OPEN OUTSIDE
PBC-I           (I)                                  PUSH BUTTON CLOSE INSIDE
PBC-O           (I)                                  PUSH BUTTON CLOSE OUTSIDE
KEY             (I)                                  KEY SWITCH
STOP            (I)                                  EMERGENCY STOP SWITCH
MUP             (Q)                                  MOTOR UPWARDS
MDOWN  (Q)                                           MOTOR DOWNWARDS
                #UB                                  OPEN FROM THE INSIDE OR OUTSIDE
                AN       =STOP
                A
                A        =PBO-O
                A        =KEY
                O        =PBO-I
                )
                AN       =ENDU
                S        = MUP
                ***
                #UB                                  CLOSE FROM THE OUTSIDE OR INSIDE
*RESETTING THE OUTPUT MOTOR UPWARDS
                O        =ENDU
                O        =STOP
                RB       =MUP
                ***
                #UB                                  CLOSE FROM THE OUTSIDE OR INSIDE
*HERE THE DOOR CLOSES IMMEDIATELY
                A(
                A        =PBC-O
                A        =KEY
                O        =PBC-I
                )
                AN       =ENDL
                S        =MDOWN
                ***
                #UB                                  CLOSE FROM THE OUTSIDE OR INSIDE
*RESETTING THE OUTPUT MOTOR DOWNWARDS
                O        =ENDL
                O        =STOP
                RB       =MDOWN
                BE
```


Operation

STL source: B: FBTESTA0.SEQ

ADDR STATEMENT OPERAND SYMBOL STATEMENT COMMENT
GARAGE FB 1 FOR A GARAGE DOOR
#N GARAGE
ENDU (I) UPPER END SWITCH
ENDL (I) LOWER END SWITCH
PBO-I (I) PUSH BUTTON OPEN INSIDE
PBO-O (I) PUSH BUTTON OPEN OUTSIDE
PBC-I (I) PUSH BUTTON CLOSE INSIDE
PBC-O (I) PUSH BUTTON CLOSE OUTSIDE
KEY (I) KEY SWITCH
STOP (I) EMERGENCY STOP SWITCH
MUP (Q) MOTOR UPWARDS
MDOWN (Q) MOTOR DOWNWARDS
#UB OPEN FROM THE INSIDE OR OUTSIDE
AN =STOP
A
A =PBO-O
A =KEY
O =PBO-I
)
AN =ENDU
S = MUP
***
#UB CLOSE FROM THE OUTSIDE OR INSIDE
*RESETTING THE OUTPUT MOTOR UPWARDS
O =ENDU
O =STOP
RB =MUP
***
#UB CLOSE FROM THE OUTSIDE OR INSIDE
*HERE THE DOOR CLOSES IMMEDIATELY
A(
A =PBC-O
A =KEY
O =PBC-I
)
AN =ENDL
S =MDOWN
***
#UB CLOSE FROM THE OUTSIDE OR INSIDE
*RESETTING THE OUTPUT MOTOR DOWNWARDS
O =ENDL
O =STOP
RB =MDOWN
BE

3 - 42

C79000-B8576-C677-02

input

Condition: The STL editor/batch compiler package has to be loaded.

Return to the example. Fill in the PRESETTING with the file name FBTEST for the STL source file and the intermediate file and TEST@@ for the program file and the symbol file. *ENTER* and call the editing function.

If you have not left the STL editor/batch compiler package, the screen FUNCTION SELECTION of the package is shown.

Retrun to the PRESETTING and change the name of the STL source file to FBTEST.

change PRESETTING

➤ Call PRESETting (F6); the cursor flashes in the STL SOURCE FILE field.

➤ Position the cursor with the **single arrow key right** on the file name and overwrite it with *FBTEST*.

➤ Press the return key. Now the intermediate file is also called FBTEST. The other files remain unchanged.

➤ **ENTER** (F6).

➤ Call the editing function using **EDIT** (F1).

Now you can start programming your "garage" function block.

explanations for function block editing:

Please look up the writing conventions for the control characters and operations in chapters 3.3.2 and 3.3.3. When entering, please observe the special conditions for function blocks:

- Formal operands are defined in the ADDR and STATEMENT columns. This definition corresponds to the identifier list in function blocks, which are created in the CSF, LAD, STL package.
- The formal operands consist of up to 4 characters.
- The definition as input, output etc. is written in brackets.
- During the programming an equality sign has to be written before the formal operand (as in the CSF, LAD, STL package).

Operation

input

Condition: The STL editor/batch compiler package has to be loaded.

Return to the example. Fill in the PRESETTING with the file name FBTEST for the STL source file and the intermediate file and TEST@@ for the program file and the symbol file. *ENTER* and call the editing function.

If you have not left the STL editor/batch compiler package, the screen FUNCTION SELECTION of the package is shown.

Retrun to the PRESETTING and change the name of the STL source file to FBTEST.

change PRESETTING

➤ Call PRESETting (F6); the cursor flashes in the STL SOURCE FILE field.

➤ Position the cursor with the **single arrow key right** on the file name and overwrite it with *FBTEST*.

➤ Press the return key. Now the intermediate file is also called FBTEST. The other files remain unchanged.

➤ **ENTER** (F6).

➤ Call the editing function using **EDIT** (F1).

Now you can start programming your "garage" function block.

explanations for function block editing:

Please look up the writing conventions for the control characters and operations in chapters 3.3.2 and 3.3.3. When entering, please observe the special conditions for function blocks:

- Formal operands are defined in the ADDR and STATEMENT columns. This definition corresponds to the identifier list in function blocks, which are created in the CSF, LAD, STL package.
- The formal operands consist of up to 4 characters.
- The definition as input, output etc. is written in brackets.
- During the programming an equality sign has to be written before the formal operand (as in the CSF, LAD, STL package).

The FB1 function block named garage is assigned to the "garage" symbol in the symbol file. Therefore, the block beginning can be programmed symbolically.

Comments and segment headings are handled as in the program block. You enter the segment end using the function key that you assigned (as above 3.3.5.2) and store with **BACK-UP** (F7).

Operating sequence (text to be entered is written in *italics*, functions to be used in **bold** letters)

- #
- press **double arrow key right**, enter *Garage* in the OPERAND SYMBOL column,
- press **double arrow key right** for the comment
- *FB1 for a garage door*
- press **return key**,
- *#N*
- blank
- *garage*
- press **return key**,
- press **double arrow key left**, the cursor is now in the ADDRess column.
- Write *ENDO*, the cursor jumps into the STATEMENT column after the fourth character
- *(E)*
- press **double arrow key right** twice. As a statement comment you write
- *upper end switch*
- press **return key**;

and so on.

Operation

The FB1 function block named garage is assigned to the "garage" symbol in the symbol file. Therefore, the block beginning can be programmed symbolically.

Comments and segment headings are handled as in the program block. You enter the segment end using the function key that you assigned (as above 3.3.5.2) and store with **BACK-UP** (F7).

Operating sequence (text to be entered is written in *italics*, functions to be used in **bold** letters)

- #
- press **double arrow key right**, enter *Garage* in the OPERAND SYMBOL column,
- press **double arrow key right** for the comment
- *FB1 for a garage door*
- press **return key**,
- *#N*
- blank
- *garage*
- press **return key**,
- press **double arrow key left**, the cursor is now in the ADDRess column.
- Write *ENDO*, the cursor jumps into the STATEMENT column after the fourth character
- *(E)*
- press **double arrow key right** twice. As a statement comment you write
- *upper end switch*
- press **return key**;

and so on.

3 - 44

C79000-B8576-C677-02

The statements correspond to the program block statements:

- *A(*
- press **return key,**
- *A*
- blank
- *=PBO-O*
- press **return key;**

and so on.

Operation

The statements correspond to the program block statements:

- *A(*
- press **return key,**
- *A*
- blank
- *=PBO-O*
- press **return key;**

and so on.

3

C79000-B8576-C677-02

3 - 45

Function Block Parametrization

In order to parameterize the function block, i.e. to provide it with actual operands, write a program block:

```
STL source:  B: FBTESTA0.SEQ

ADDR STATEMENT          OPERAND SYMBOL              STATEMENT COMMENT
     #PB2
     SPA                        GARAGE                        Parameterizing the FB1
     ,I 1.0
     ,I 1.1
     ,                          PB-OPEN I
     ,                          PB-OPEN O
     ,                          PB-CLOSE I
     ,                          PB-CLOSE O
     ,                          KEY
     ,                          STOP
     ,                          MOT UP
     ,                          MOT DOWN
     BE
```

You can enter the actual operands either absolutely or symbolically. Here please observe

that a comma has to be placed before every actual operand as a control character, and

that the parameter sequence is equivalent to the identifier list of the formal operands in the function blocks.

Operation

Function Block Parametrization

In order to parameterize the function block, i.e. to provide it with actual operands, write a program block:

STL source: B: FBTESTA0.SEQ

ADDR STATEMENT OPERAND SYMBOL STATEMENT COMMENT
#PB2
SPA GARAGE Parameterizing the FB1
,I 1.0
,I 1.1
, PB-OPEN I
, PB-OPEN O
, PB-CLOSE I
, PB-CLOSE O
, KEY
, STOP
, MOT UP
, MOT DOWN
BE

You can enter the actual operands either absolutely or symbolically. Here please observe

that a comma has to be placed before every actual operand as a control character, and

that the parameter sequence is equivalent to the identifier list of the formal operands in the function blocks.

3 - 46

C79000-B8576-C677-02

Operating sequence:

- ➤ *#PB1*
- ➤ press **return key,**
- ➤ *JU*
- ➤ pres the **double arrow key right** once,
- ➤ *GARAGE*
- ➤ press **double arrow key right** once,
- ➤ *Parameterizing of the FB1*
- ➤ press **return key,**
- ➤ *,I 1.0*
- ➤ press **return key,**

or symbolically:

- ➤ ,
- ➤ press **double arrow key right** once,
- ➤ *PB-OPEN O*
- ➤ press **return key**.

Store intermediately with **BACK-UP** (F7), as the data block following in chapter 3.3.7 is also written into this STL source file.

Operation

Operating sequence:

- ➤ *#PB1*
- ➤ press **return key,**
- ➤ *JU*
- ➤ pres the **double arrow key right** once,
- ➤ *GARAGE*
- ➤ press **double arrow key right** once,
- ➤ *Parameterizing of the FB1*
- ➤ press **return key,**
- ➤ *,I 1.0*
- ➤ press **return key,**

or symbolically:

- ➤ ,
- ➤ press **double arrow key right** once,
- ➤ *PB-OPEN O*
- ➤ press **return key**.

Store intermediately with **BACK-UP** (F7), as the data block following in chapter 3.3.7 is also written into this STL source file.

3

C79000-B8576-C677-02

3 - 47

3.3.7
Data Block Input

Example The following data block will be the basis for this chapter.

If you require information on data blocks, please refer to the Introduction and to the Data Block chapter in the STEP 5 description of your PG manual, volume 2.

```
STL source:  B:FBTESTA0.SEQ

ADDR   STATEMENT        OPERAND SYMBOL           STATEMENT COMMENT

       ‡DB 12
0      KH               FFFF
1      KM               11111111 11000000       QUANTITY
2      KH               0013
3      KF               -32768
4      KF               +32767
5      KG               -2740000+22
6      KG               -1234000+05
100    KY               022,033
111    KY               022,033
       KY               022,033
       KY               022,033
       KH               ADAC
       KH               4538
       KF               +32767
       KF               +32767
       KF               +32767
       KF               +32767
       KF               +32767
       KY               022,033
       KY               022,033
       BE
```


Operation

3.3.7
Data Block Input

Example The following data block will be the basis for this chapter.

If you require information on data blocks, please refer to the Introduction and to the Data Block chapter in the STEP 5 description of your PG manual, volume 2.

STL source: B:FBTESTA0.SEQ

ADDR STATEMENT OPERAND SYMBOL STATEMENT COMMENT

‡DB 12
0 KH FFFF
1 KM 11111111 11000000 QUANTITY
2 KH 0013
3 KF -32768
4 KF +32767
5 KG -2740000+22
6 KG -1234000+05
100 KY 022,033
111 KY 022,033
KY 022,033
KY 022,033
KH ADAC
KH 4538
KF +32767
KF +32767
KF +32767
KF +32767
KF +32767
KY 022,033
KY 022,033
BE

input

condition: The STL editor/batch compiler package has been loaded.

If you have just carried out the function block example, you are in the editing function and the FBTEST file is displayed.

Return to this example, fill in the PRESETTING with the file names FBTEST for the STL source file and intermediate file and TEST@@ for the program file and the symbol file. *Transfer* and call the editing function.

explanations for data block editing Please look up the writing conventions for the control characters and data words in chapters 3.3.2 and 3.3.2.

When entering please observe the special conditions for data blocks:
- The data format is in the STATEMENT column
- The value is in the OPERAND SYMBOL column.
- The relative data word address in reference to the block beginning is in the ADDR column.
- You do not need to mark your data words with an equality sign nor terminate them with a semicolon.

HINT: If you enter an address, which is not equivalent to the actual address in the DB, the space is filled up with KH 0000 during the compilation (in the example the addresses 9 to 99). By doing this space is provided for the data from the process.

The repetition factor cannot be directly used as in the CSF, LAD, STL package but only with the combination of the **COPY** function.

operating sequence:
- ➤ *#DB12*
- ➤ press **return key**
- ➤ press **double arrow key left** once and enter
- ➤ the address *0* .
- ➤ press **double arrow key right** once. Enter
- ➤ *KH* in the STATEMENT column.

Operation

input

condition: The STL editor/batch compiler package has been loaded.

If you have just carried out the function block example, you are in the editing function and the FBTEST file is displayed.

Return to this example, fill in the PRESETTING with the file names FBTEST for the STL source file and intermediate file and TEST@@ for the program file and the symbol file. *Transfer* and call the editing function.

explanations for data block editing Please look up the writing conventions for the control characters and data words in chapters 3.3.2 and 3.3.2.

When entering please observe the special conditions for data blocks:
- The data format is in the STATEMENT column
- The value is in the OPERAND SYMBOL column.
- The relative data word address in reference to the block beginning is in the ADDR column.
- You do not need to mark your data words with an equality sign nor terminate them with a semicolon.

HINT: If you enter an address, which is not equivalent to the actual address in the DB, the space is filled up with KH 0000 during the compilation (in the example the addresses 9 to 99). By doing this space is provided for the data from the process.

The repetition factor cannot be directly used as in the CSF, LAD, STL package but only with the combination of the **COPY** function.

operating sequence:
- ➤ *#DB12*
- ➤ press **return key**
- ➤ press **double arrow key left** once and enter
- ➤ the address *0* .
- ➤ press **double arrow key right** once. Enter
- ➤ *KH* in the STATEMENT column.

C79000-B8576-C677-02 3 - 49

➤ press **double arrow key right** once. Enter the value
➤ *FFFF* in the OPERAND SYMBOL column,
➤ press **return key.**
➤ Press **double arrow key left** once and enter the address
➤ *1* .
➤ Press **double arrow key right,**
➤ enter *KM* ,
➤ press **double arrow key right,**
➤ enter *11111111 11000000* ,
➤ press **double arrow key right,**
➤ write *QUANTITY* as a comment,
➤ press **return key**

and so on.

With KY 22,23

➤ you write the line once,
➤ buffer it with **BUFFER** (F1),
➤ **LINE** (F1), **RETURN** (F8) + Return key and
➤ copy it with **COPY** (F2),
➤ repetition factor 3 and
➤ **LINE** (F1).

You can also use the COPY (F2) BLOCK (F4) function.

Now terminate your exercises with **ENTER** (F6). The FBTEST STL source file will then be stored and compiled into the intermediate file. The *Editing* is also terminated.

Operation

➤ press **double arrow key right** once. Enter the value
➤ *FFFF* in the OPERAND SYMBOL column,
➤ press **return key.**
➤ Press **double arrow key left** once and enter the address
➤ *1* .
➤ Press **double arrow key right,**
➤ enter *KM* ,
➤ press **double arrow key right,**
➤ enter *11111111 11000000* ,
➤ press **double arrow key right,**
➤ write *QUANTITY* as a comment,
➤ press **return key**

and so on.

With KY 22,23

➤ you write the line once,
➤ buffer it with **BUFFER** (F1),
➤ **LINE** (F1), **RETURN** (F8) + Return key and
➤ copy it with **COPY** (F2),
➤ repetition factor 3 and
➤ **LINE** (F1).

You can also use the COPY (F2) BLOCK (F4) function.

Now terminate your exercises with **ENTER** (F6). The FBTEST STL source file will then be stored and compiled into the intermediate file. The *Editing* is also terminated.

3.3.8 Editing an STL Source File

If you want to edit an STL source file within the STL editor/batch compiler, it is displayed on the screen with *Edit* and you can edit it using the editing functions.

In our example the FBTEST file is to be included in the TEST@@ STL source file, using the Include command. FBTESTA0.SEQ therefore has to be available as an intermediate file. This requirement has already been met in our example (as above).

condition: In the presetting TEST@@ has been entered as an STL source file.

➤ **EDITING** (F1) of the TEST@@ file.

Jump to the end of the file with

➤ **SEARCH** (F4),

➤ **END** (F7), and

➤ then **RETURN** (F8) to the editing mode.

The insert mode has been preset.

➤ Move the cursor before the first block, between BE and #PBn or to the file end after the last block end BE;

➤ expand vertically; now there is enough space for the Include command.

➤ *#I*

➤ blank

➤ *B:FBTEST*

➤ press **ENTER** (F6) for saving and compiling, so that your intermediate file is updated.

If you now compile the TEST@@A0.SEQ STL source file into the TEST@@ST.S5D STEP 5 program file, the FBTESTA1.SEQ is also compiled and transferred into the program file. Then all the blocks edited during this training session are available there.

Operation

3.3.8 Editing an STL Source File

If you want to edit an STL source file within the STL editor/batch compiler, it is displayed on the screen with *Edit* and you can edit it using the editing functions.

In our example the FBTEST file is to be included in the TEST@@ STL source file, using the Include command. FBTESTA0.SEQ therefore has to be available as an intermediate file. This requirement has already been met in our example (as above).

condition: In the presetting TEST@@ has been entered as an STL source file.

➤ **EDITING** (F1) of the TEST@@ file.

Jump to the end of the file with

➤ **SEARCH** (F4),

➤ **END** (F7), and

➤ then **RETURN** (F8) to the editing mode.

The insert mode has been preset.

➤ Move the cursor before the first block, between BE and #PBn or to the file end after the last block end BE;

➤ expand vertically; now there is enough space for the Include command.

➤ *#I*

➤ blank

➤ *B:FBTEST*

➤ press **ENTER** (F6) for saving and compiling, so that your intermediate file is updated.

If you now compile the TEST@@A0.SEQ STL source file into the TEST@@ST.S5D STEP 5 program file, the FBTESTA1.SEQ is also compiled and transferred into the program file. Then all the blocks edited during this training session are available there.

3.4 Compiling with the COMPILER Function

Your STL source file stored with *ENTER* is thereafter available as an intermediate file (INT). In order to compile it into a STEP 5 program, call the **COMPILER** function. There you can compile your statement list into the program file, which has been named in the presetting. With **INT>MC5** the intermediate file is transformed into the MC5 machine code. With **SEQ>MC5** the STL source file is transformed into the MC5 machine code and the intermediate file is generated automatically.

You can recompile in the same way: an intermediate file is created from an MC5 program file with **MC5>INT** (such an intermediate file is further edited with the **SPECIAL** functions for a STL source file), or an STL source file with the corresponding intermediate file is directly created with **MC5>SEQ.**

The **SEQ>MC5** function at first carries out the **SEQ>INT** compilation. If here any errors occur the **INT>MC5** compilation is not started but the function is terminated. That means, the error messages which occurred during the generation of the intermediate file, are written in the error list. Accordingly, the **MC5>SEQ** function at first starts the **MC5>INT** compilation and the **INT>SEQ** compilation is only started if the intermediate file is generated without any errors.

3.4.5 Operating Sequence: Compiling into the Program File

condition: In the presetting the FBTESTA0.SEQ STL source file is present.

compilation

➤ Call the **COMPILER** function with F2,

➤ press **INT>MC5** (F2) or **SEQ>MC5** (F1).

➤ Fill in the following command line:

Compile blocks:	OPT:	IMP:

When you press the **help key** you are informed about all possible inputs for each input field.

Operation

3.4 Compiling with the COMPILER Function

Your STL source file stored with *ENTER* is thereafter available as an intermediate file (INT). In order to compile it into a STEP 5 program, call the **COMPILER** function. There you can compile your statement list into the program file, which has been named in the presetting. With **INT>MC5** the intermediate file is transformed into the MC5 machine code. With **SEQ>MC5** the STL source file is transformed into the MC5 machine code and the intermediate file is generated automatically.

You can recompile in the same way: an intermediate file is created from an MC5 program file with **MC5>INT** (such an intermediate file is further edited with the **SPECIAL** functions for a STL source file), or an STL source file with the corresponding intermediate file is directly created with **MC5>SEQ.**

The **SEQ>MC5** function at first carries out the **SEQ>INT** compilation. If here any errors occur the **INT>MC5** compilation is not started but the function is terminated. That means, the error messages which occurred during the generation of the intermediate file, are written in the error list. Accordingly, the **MC5>SEQ** function at first starts the **MC5>INT** compilation and the **INT>SEQ** compilation is only started if the intermediate file is generated without any errors.

3.4.5 Operating Sequence: Compiling into the Program File

condition: In the presetting the FBTESTA0.SEQ STL source file is present.

compilation

➤ Call the **COMPILER** function with F2,

➤ press **INT>MC5** (F2) or **SEQ>MC5** (F1).

➤ Fill in the following command line:

Compile blocks:	OPT:	IMP:

When you press the **help key** you are informed about all possible inputs for each input field.

3 - 52 C79000-B8576-C677-02

➤ Press the **help key** in the BLOCK field:

In addition to the usual inputs in the STEP 5 basic package you can enter block ranges to be edited, e.g. PB12 - PB21.

➤ Write *A* into this field and terminate with the

➤ **return key**.

➤ press the **help key** in the OPTion field.

With "2" you can initiate a compilation test: your intermediate file is compiled and checked for errors. It is not stored in the program file, however. The errors which have occured can be accessed via the error list.

➤ Enter *2* and

➤ press the **return key.**

If your program file already contains blocks with identical names and the OPTion field is empty, you are prompted by the programming unit to confirm each copying process; when confirmed, the old block will be overwritten by the new, identical named one.

If you use the option "1", the blocks are overwritten in the program file without your confirmation.

➤ Press the **help key** in the BLOCK field.

The printer output formats are those, which are usual for the STEP 5 basic package: standard print, small print and super small print. The paper size DIN A3 or A4 depends on the connected printer.

In our example this field remains empty.

➤ press **ENTER key.**

Compile blocks: **A**	OPT: **2**	IMP:

The programming unit now carries out the compiling and testing. A display is then made of the blocks which have been edited and how many errors have occurred, or if compilation has been carried out without errors. Then the function selection is displayed again.

Operation

➤ Press the **help key** in the BLOCK field:

In addition to the usual inputs in the STEP 5 basic package you can enter block ranges to be edited, e.g. PB12 - PB21.

➤ Write *A* into this field and terminate with the

➤ **return key**.

➤ press the **help key** in the OPTion field.

With "2" you can initiate a compilation test: your intermediate file is compiled and checked for errors. It is not stored in the program file, however. The errors which have occured can be accessed via the error list.

➤ Enter *2* and

➤ press the **return key.**

If your program file already contains blocks with identical names and the OPTion field is empty, you are prompted by the programming unit to confirm each copying process; when confirmed, the old block will be overwritten by the new, identical named one.

If you use the option "1", the blocks are overwritten in the program file without your confirmation.

➤ Press the **help key** in the BLOCK field.

The printer output formats are those, which are usual for the STEP 5 basic package: standard print, small print and super small print. The paper size DIN A3 or A4 depends on the connected printer.

In our example this field remains empty.

➤ press **ENTER key.**

Compile blocks: **A**	OPT: **2**	IMP:

The programming unit now carries out the compiling and testing. A display is then made of the blocks which have been edited and how many errors have occurred, or if compilation has been carried out without errors. Then the function selection is displayed again.

C79000-B8576-C677-02

3 - 53

If the compilation was carried out without errors, then repeat the compilation. Do not fill in the option field so that the program file can be created and enter ✱ in the PRINTing field, in order to get a printout.

Compile blocks: **A**	OPT:	IMP:Q

➤ Press **ENTER key**.

Then the blocks of the FBTESTA0.SEQ STL source file are transferred into the TEST@@ST.S5D program file in machine code and saved. Now you can further process the FB1 and PB2 blocks in the CSF, LAD, STL package (e.g. test in the PC).

3.4.6 Operating Sequence: Recompiling the Program File

The operation is the same was as for the compilation. You just use the **MC5>INT** (F4) or **MC5>SEQ** (F5) function. It is important for the *recompilation* that the required data is available in the presetting. For the command line the same conventions are valid as was for the *compilation*. With the help key in the OPTion field the following other possibilities of a transfer are provided:

➤ Call **COMPILER** (2),

➤ select **MC5>INT** (F4),

➤ fill in the command line,

➤ press the **return key**.

The intermediate file is now regenerated or one with an equivalent name is overwritten (after confirmation). The **MC5>SEQ** function automatically generates the STL source file, which you can edit with the STL editor (see 3.4). For the **MC5>INT** function you must create a sequential source file yourself using the INT>SEQ SPECIAL function (as below, chapter 3.7).

Operation

If the compilation was carried out without errors, then repeat the compilation. Do not fill in the option field so that the program file can be created and enter ✱ in the PRINTing field, in order to get a printout.

Compile blocks: **A**	OPT:	IMP:Q

➤ Press **ENTER key**.

Then the blocks of the FBTESTA0.SEQ STL source file are transferred into the TEST@@ST.S5D program file in machine code and saved. Now you can further process the FB1 and PB2 blocks in the CSF, LAD, STL package (e.g. test in the PC).

3.4.6 Operating Sequence: Recompiling the Program File

The operation is the same was as for the compilation. You just use the **MC5>INT** (F4) or **MC5>SEQ** (F5) function. It is important for the *recompilation* that the required data is available in the presetting. For the command line the same conventions are valid as was for the *compilation*. With the help key in the OPTion field the following other possibilities of a transfer are provided:

➤ Call **COMPILER** (2),

➤ select **MC5>INT** (F4),

➤ fill in the command line,

➤ press the **return key**.

The intermediate file is now regenerated or one with an equivalent name is overwritten (after confirmation). The **MC5>SEQ** function automatically generates the STL source file, which you can edit with the STL editor (see 3.4). For the **MC5>INT** function you must create a sequential source file yourself using the INT>SEQ SPECIAL function (as below, chapter 3.7).

3.5 Error List

The error list is not only a list of the errors which occurred during the compilation, but it is also a complete log of the compilation: it additionally lists the blocks compiled without errors and in case of an abort indicates the corresponding position.

In order to have a real example, explicitly create an error in your FB1 (programmed in chapter 3.3.5): **EDIT** the FB1 and write e.g. only *R* as reset commands. Already when you are saving with **ENTER** an indication will show that errors have occurred.

➤ Now call the **E-LIST** (F3).

➤ Fill in the PRINTer input field in the command line so that you can easily correct the STL source with this print-out. When you press the help key the parameters for the PRINTer field are displayed.

➤ press the **ENTER key**: Each wrong statement is indicated with block identification, a line number and an explanation. The blocks which have been compiled correctly are also listed.


```
File B:FBTESTAF.SEQ
COMPILATION STL SOURCE  B:FBTESTA0.SEQ  ⇒ INTERMEDIATE FILE B:FBTESTA1.SEQ
       R =MUP
*** ERROR IN LINE      28: OPERAND NOT ALLOWED   ***
       R =MDOWN
*** ERROR IN LINE      45: OPERAND NOT ALLOWED   ***
*** FB 1  COMPILED,    2 ERRORS FOUND      ***
*** PB 1  COMPILED, BLOCK WITHOUT ERRORS  ***
*** COMPILATION TERMINATED,   2  ERRORS, NO WARNING(S)   ***
```

output of error list

The error list is output on the screen if you do not fill in the PRINTer field in the command line. If long error lists are output on the screen, they are stopped after 20 lines each and you can either abort the output by pressing the abort key or view the next screen page using the enter key.

Operation

3.5 Error List

The error list is not only a list of the errors which occurred during the compilation, but it is also a complete log of the compilation: it additionally lists the blocks compiled without errors and in case of an abort indicates the corresponding position.

In order to have a real example, explicitly create an error in your FB1 (programmed in chapter 3.3.5): **EDIT** the FB1 and write e.g. only *R* as reset commands. Already when you are saving with **ENTER** an indication will show that errors have occurred.

➤ Now call the **E-LIST** (F3).

➤ Fill in the PRINTer input field in the command line so that you can easily correct the STL source with this print-out. When you press the help key the parameters for the PRINTer field are displayed.

➤ press the **ENTER key**: Each wrong statement is indicated with block identification, a line number and an explanation. The blocks which have been compiled correctly are also listed.

3

```
File B:FBTESTAF.SEQ
COMPILATION STL SOURCE  B:FBTESTA0.SEQ  ⇒ INTERMEDIATE FILE B:FBTESTA1.SEQ
       R =MUP
*** ERROR IN LINE      28: OPERAND NOT ALLOWED   ***
       R =MDOWN
*** ERROR IN LINE      45: OPERAND NOT ALLOWED   ***
*** FB 1  COMPILED,    2 ERRORS FOUND      ***
*** PB 1  COMPILED, BLOCK WITHOUT ERRORS  ***
*** COMPILATION TERMINATED,   2  ERRORS, NO WARNING(S)   ***
```

output of error list

The error list is output on the screen if you do not fill in the PRINTer field in the command line. If long error lists are output on the screen, they are stopped after 20 lines each and you can either abort the output by pressing the abort key or view the next screen page using the enter key.

3.6 Printing

With this function you only print out the preset STL source file. Therefore only the layout of your printed output in the command line has to be determined. (You will find detailed information on the printed layout in the STEP 5 description, manual volume 2, in the Input/Output of STL Blocks chapter.) Compiled files can only be printed out via the command lines of the COMPILER function.

Condition: The printer is connected and ready for operation; if you have an external printer, it has to be parameterized in the PRINTER UTILITY PROGram. The PT 88 printer is preset. In the presetting of the STL editor/batch compiler the name of the file to be printed is written, e.g. FBTEST. The function selection is displayed on the screen.

➤ Call **PRINT** (F4)

➤ Fill in the PRINTer field; preset is the standard output in standard print.

➤ Press **ENTER key**.

The FBTESTAQ.SEQ STL source file is printed. The function selection is redisplayed on the programming unit.

You can control the printed output via a file as in the CSF, LAD, STL package. You define this file in the PRINTER UTILITY PROGram and then enter the name of the printer file in the PRESETTING. For your current printing job a layout is determined as usual in the command line. This layout is then also transferred to the printing file.

Operation

3.6 Printing

With this function you only print out the preset STL source file. Therefore only the layout of your printed output in the command line has to be determined. (You will find detailed information on the printed layout in the STEP 5 description, manual volume 2, in the Input/Output of STL Blocks chapter.) Compiled files can only be printed out via the command lines of the COMPILER function.

Condition: The printer is connected and ready for operation; if you have an external printer, it has to be parameterized in the PRINTER UTILITY PROGram. The PT 88 printer is preset. In the presetting of the STL editor/batch compiler the name of the file to be printed is written, e.g. FBTEST. The function selection is displayed on the screen.

➤ Call **PRINT** (F4)

➤ Fill in the PRINTer field; preset is the standard output in standard print.

➤ Press **ENTER key**.

The FBTESTAQ.SEQ STL source file is printed. The function selection is redisplayed on the programming unit.

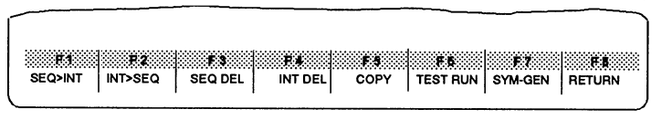
You can control the printed output via a file as in the CSF, LAD, STL package. You define this file in the PRINTER UTILITY PROGram and then enter the name of the printer file in the PRESETTING. For your current printing job a layout is determined as usual in the command line. This layout is then also transferred to the printing file.

3 - 56 C79000-B8576-C677-02

3.7 SPECIAL Functions for Editing Intermediate Files and Source Files

The **special functions** are used for editing and modifying sequential files and intermediate files and it provides for a subsequent test run for the compiled program file. All processes correspond to the files, which are determined in the presetting. Therefore, please check that the right data for your intentions has been entered there. In our example the presetting remains unchanged.

Any process can be stopped with the abort key.



3.7.5 COPYing

Use the COPY function for creating back-up copies. At first you can copy the intermediate file but then a copy can be made of the STL source file to another disk drive. The programming unit will display a message e.g. "hardware error" during the copying process if the disk drive is not closed.

In order to edit the example files without any risk, please copy them on to a floppy disk.

- ➤ Enter the STL source file in the PRESETTING,
- ➤ call **SPECIAL functions** (F5),
- ➤ call **COPY** (F5),
- ➤ enter disk drive: *A*
- ➤ press **ENTER key.** Now the intermediate file is also saved on a floppy disk. When the question appears "Also copy SEQ. source file?"
- ➤ press **ENTER key** (yes).

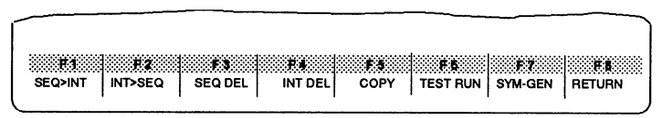
Now the intermediate file and the STL source file are copied to the disk. The function selection is displayed again on the programming unit.

Operation

3.7 SPECIAL Functions for Editing Intermediate Files and Source Files

The **special functions** are used for editing and modifying sequential files and intermediate files and it provides for a subsequent test run for the compiled program file. All processes correspond to the files, which are determined in the presetting. Therefore, please check that the right data for your intentions has been entered there. In our example the presetting remains unchanged.

Any process can be stopped with the abort key.



3

3.7.5 COPYing

Use the COPY function for creating back-up copies. At first you can copy the intermediate file but then a copy can be made of the STL source file to another disk drive. The programming unit will display a message e.g. "hardware error" during the copying process if the disk drive is not closed.

In order to edit the example files without any risk, please copy them on to a floppy disk.

- ➤ Enter the STL source file in the PRESETTING,
- ➤ call **SPECIAL functions** (F5),
- ➤ call **COPY** (F5),
- ➤ enter disk drive: *A*
- ➤ press **ENTER key.** Now the intermediate file is also saved on a floppy disk. When the question appears "Also copy SEQ. source file?"
- ➤ press **ENTER key** (yes).

Now the intermediate file and the STL source file are copied to the disk. The function selection is displayed again on the programming unit.

3.7.6 SEQ>INT

Call the **SEQ>INT** (F1), if you e.g. would like to compile the STL source file, which has been created with another text editor. With this function the text file is converted into an intermediate file, as a precondition to compile into the program file.

- ➤ Enter the text file in the PRESETTING,
- ➤ activate **SPECIAL functions** (F5),
- ➤ call **SEQ>INT** (F1). The device displays: "Compilation of the STL source file into the intermediate file?"
- ➤ Press **ENTER key** (yes)).

Now an intermediate file is available for further editing. The function selection is displayed again on the programming unit.

3.7.7 INT>SEQ

Call **INT>SEQ** (F2), if you e.g. recompiled a program file (with COMPILER, MC5>INT) and you want to edit it in the editor. For this purpose you have to convert the intermediate file into a sequential file. Additional comments of the former STL source are lost during this process.

- ➤ In the PRESETTING determine the files concerned.
- ➤ Activate **SPECIAL function** (F5),
- ➤ call **INT>SEQ** (F2). The device asks: "Compilation of the intermediate file into the STL source file?"
- ➤ press **ENTER key** (yes).

Now a new STL source file has been generated. The function selection is displayed again on the programming unit.

Operation

3.7.6 SEQ>INT

Call the **SEQ>INT** (F1), if you e.g. would like to compile the STL source file, which has been created with another text editor. With this function the text file is converted into an intermediate file, as a precondition to compile into the program file.

- ➤ Enter the text file in the PRESETTING,
- ➤ activate **SPECIAL functions** (F5),
- ➤ call **SEQ>INT** (F1). The device displays: "Compilation of the STL source file into the intermediate file?"
- ➤ Press **ENTER key** (yes)).

Now an intermediate file is available for further editing. The function selection is displayed again on the programming unit.

3.7.7 INT>SEQ

Call **INT>SEQ** (F2), if you e.g. recompiled a program file (with COMPILER, MC5>INT) and you want to edit it in the editor. For this purpose you have to convert the intermediate file into a sequential file. Additional comments of the former STL source are lost during this process.

- ➤ In the PRESETTING determine the files concerned.
- ➤ Activate **SPECIAL function** (F5),
- ➤ call **INT>SEQ** (F2). The device asks: "Compilation of the intermediate file into the STL source file?"
- ➤ press **ENTER key** (yes).

Now a new STL source file has been generated. The function selection is displayed again on the programming unit.

3.7.8 SEQ DEL and INT DEL

SEQ DEL and **INT DEL** delete the preset STL source file and the intermediate file.

➤ In the PRESETTING determine the files concerned.

➤ Call **SPECIAL functions** (F5),

➤ call **SEQ DEL** (F3). The device asks: "Delete the STL source file?"

➤ press **ENTER key** (yes).

The function selection is displayed again on the programming unit.

➤ Call **SPECIAL functions** (F5),

➤ call **INT DEL** (F4). The device asks: "Delete the intermediate file?"

➤ press **ENTER key** (yes) or abort (no).

The function selection is displayed again on the programming unit.

3.7.9 TEST RUN

The **TEST RUN** is an additional block check in the preset program file. Here a check is made, whether the standard function blocks are provided with the right parameters. If errors occur they can be accessed in the error list.

In the command line of this function you can enter single blocks, block groups, block types or all the blocks from a program file; use the help key if you require this information.

Operation

3.7.8 SEQ DEL and INT DEL

SEQ DEL and INT DEL delete the preset STL source file and the intermediate file.

➤ In the PRESETTING determine the files concerned.

➤ Call SPECIAL functions (F5),

➤ call SEQ DEL (F3). The device asks: "Delete the STL source file?"

➤ press ENTER key (yes).

The function selection is displayed again on the programming unit.

➤ Call SPECIAL functions (F5),

➤ call INT DEL (F4). The device asks: "Delete the intermediate file?"

➤ press ENTER key (yes) or abort (no).

The function selection is displayed again on the programming unit.

3.7.9 TEST RUN

The TEST RUN is an additional block check in the preset program file. Here a check is made, whether the standard function blocks are provided with the right parameters. If errors occur they can be accessed in the error list.

In the command line of this function you can enter single blocks, block groups, block types or all the blocks from a program file; use the help key if you require this information.

- ➤ In the PRESETTING determine the program file to be checked and possibly the desired PC type,
- ➤ call **SPECIAL functions** (F5),
- ➤ call **TEST RUN** (F6),
- ➤ fill in the command line: enter e.g. ✱,
- ➤ press **ENTER key**.
- ➤ Fill in the block list with *PB1*.
- ➤ Press **return key**,
- ➤ enter *FB1, DB12* accordingly,
- ➤ press **ENTER key.**

Test messages are displayed by the programming unit. If errors are found, have an error list printed out.

3.7.10 SYM-GEN

From an STL source file SYM-GEN generates a symbolic source file, which contains all the absolute parameters and symbols used. You can further edit the symbolic source file by means of the symbolic editor, in order to e.g. complement assignments and enter comments. The symbols and absolute parameters are displayed as often in the symbolic source, as they were used in the STL source file. In order to avoid this, you should carry out the following steps:

- ➤ Create the symbolic source with SYM-GEN

Example:

Absolute	Symbol	Comment
	EME-OFF	
	EME-OFF	
	EME-OFF	

- ➤ Change over to symbolic editor (package 8)
- ➤ Complete the assignment when the symbol or the absolute parameter, resp., occurs for the first time

Operation

- ➤ In the PRESETTING determine the program file to be checked and possibly the desired PC type,
- ➤ call **SPECIAL functions** (F5),
- ➤ call **TEST RUN** (F6),
- ➤ fill in the command line: enter e.g. ✱,
- ➤ press **ENTER key**.
- ➤ Fill in the block list with *PB1*.
- ➤ Press **return key**,
- ➤ enter *FB1, DB12* accordingly,
- ➤ press **ENTER key.**

Test messages are displayed by the programming unit. If errors are found, have an error list printed out.

3.7.10 SYM-GEN

From an STL source file SYM-GEN generates a symbolic source file, which contains all the absolute parameters and symbols used. You can further edit the symbolic source file by means of the symbolic editor, in order to e.g. complement assignments and enter comments. The symbols and absolute parameters are displayed as often in the symbolic source, as they were used in the STL source file. In order to avoid this, you should carry out the following steps:

- ➤ Create the symbolic source with SYM-GEN

Example:

Absolute	Symbol	Comment
	EME-OFF	
	EME-OFF	
	EME-OFF	

- ➤ Change over to symbolic editor (package 8)
- ➤ Complete the assignment when the symbol or the absolute parameter, resp., occurs for the first time

Example:

Absolute	Symbol	Comment
E 1.0	EME-OFF	emergency off
	EME-OFF	
	EME-OFF	

➤ Compile into symbolic file Here you may ignore the "symbol already existing" error messages.

➤ Recompile the symbolic file into the symbolic source (INT > SEQ)

Example:

Absolute	Symbol	Comment
E 1.0	EME-OFF	emergency off

Now the symbolic source file contains only one assignment with the EME-OFF symbol.

Operation

Example:

Absolute	Symbol	Comment
E 1.0	EME-OFF	emergency off
	EME-OFF	
	EME-OFF	

➤ Compile into symbolic file Here you may ignore the "symbol already existing" error messages.

➤ Recompile the symbolic file into the symbolic source (INT > SEQ)

Example:

Absolute	Symbol	Comment
E 1.0	EME-OFF	emergency off

Now the symbolic source file contains only one assignment with the EME-OFF symbol.

3

Operation

3 - 62

C79000-B8576-C677-02

Command Line Version 4

Command Line Version 4

COMPILE.CMD program

The **COMPILE.CMD** program is a normal CP/M program, which has to be called from the operating system, as e.g. **S5CONVER**. That means, you can also use it in SUBMIT files so that command sequences run automatically. **COMPILE.CMD** carries out compilations, which can be selected in the STL batch compiler package, without any further user entries. For this purpose **COMPILE.CMD** loads the required S5 drivers and deletes them afterwards. That means, the S5WX*.* files must exist in the PU. If not, **COMPILE** is aborted and an error message is output.

input file

You can enter the files to be edited when calling or you can write them in a so-called input file (see 4.2). The input file can also contain the data for several compilation processes. Joker characters (?, *) are not allowed in file names.

If the corresponding compilation processes are selected, the compilation functions are carried out in the same way as the program sequence of the STL batch compiler (package 9). The options which can be entered in the command line in the STL batch compiler, can only be set via the input file. The default settings used by **COMPILE.CMD** are described in section 4.3.

Command Line Version

COMPILE.CMD program

The **COMPILE.CMD** program is a normal CP/M program, which has to be called from the operating system, as e.g. **S5CONVER**. That means, you can also use it in SUBMIT files so that command sequences run automatically. **COMPILE.CMD** carries out compilations, which can be selected in the STL batch compiler package, without any further user entries. For this purpose **COMPILE.CMD** loads the required S5 drivers and deletes them afterwards. That means, the S5WX*.* files must exist in the PU. If not, **COMPILE** is aborted and an error message is output.

input file

You can enter the files to be edited when calling or you can write them in a so-called input file (see 4.2). The input file can also contain the data for several compilation processes. Joker characters (?, *) are not allowed in file names.

If the corresponding compilation processes are selected, the compilation functions are carried out in the same way as the program sequence of the STL batch compiler (package 9). The options which can be entered in the command line in the STL batch compiler, can only be set via the input file. The default settings used by **COMPILE.CMD** are described in section 4.3.

4.1 Calling

4.1.1 Calling without Parameter Entry

Syntax: COMPILE <SOURCE> <TARGET> <LANGUAGE IDENT.>
Example: COMPILE @@@@@@A0.SEQ @@@@@@ST.S5D E

The following six combinations are possible as source and target:

STL source file	-->	program file:	nnnnnnA0.SEQ	nnnnnnST.S5D
STL source file	-->	intermediate file:	nnnnnnA0.SEQ	nnnnnnA1.SEQ
intermediate file	-->	program file:	nnnnnnA1.SEQ	nnnnnnST.S5D
intermediate file	-->	STL source file:	nnnnnnA1.SEQ	nnnnnnA0.SEQ
program file	-->	STL source file:	nnnnnnST.S5D	nnnnnnA0.SEQ
program file	-->	intermediate file:	nnnnnnST.S5D	nnnnnnA1.SEQ

4

The file names must meet the S5 conventions, i.e. 6 freely selectable characters are followed by admissible file identifications (A0.SEQ, A1.SEQ, ST.S5D). There is no difference between lower and upper case letters. If you do not enter a drive, COMPILE uses the current default drive.

When you call **COMPILE** you can enter the language identification as an option. You can choose between D (German), E (English) and F (French).

Tabs, blanks and commas are allowed as separation characters between the arguments.

Command Line Version

4.1 Calling

4.1.1 Calling without Parameter Entry

Syntax: COMPILE <SOURCE> <TARGET> <LANGUAGE IDENT.>
Example: COMPILE @@@@@@A0.SEQ @@@@@@ST.S5D E

The following six combinations are possible as source and target:

STL source file	-->	program file:	nnnnnnA0.SEQ	nnnnnnST.S5D
STL source file	-->	intermediate file:	nnnnnnA0.SEQ	nnnnnnA1.SEQ
intermediate file	-->	program file:	nnnnnnA1.SEQ	nnnnnnST.S5D
intermediate file	-->	STL source file:	nnnnnnA1.SEQ	nnnnnnA0.SEQ
program file	-->	STL source file:	nnnnnnST.S5D	nnnnnnA0.SEQ
program file	-->	intermediate file:	nnnnnnST.S5D	nnnnnnA1.SEQ

4

The file names must meet the S5 conventions, i.e. 6 freely selectable characters are followed by admissible file identifications (A0.SEQ, A1.SEQ, ST.S5D). There is no difference between lower and upper case letters. If you do not enter a drive, COMPILE uses the current default drive.

When you call **COMPILE** you can enter the language identification as an option. You can choose between D (German), E (English) and F (French).

Tabs, blanks and commas are allowed as separation characters between the arguments.

4.1.2 Calling with an Input File

Syntax: COMPILE #<INPUT-FILE NAME> <LANGUAGE IDENT.>
Example: COMPILE #A:INPTEST1.INP E

The input file name must meet the CPM conventions, i.e. the file name may consist of up to eight characters (letters or numbers) and the file type of three characters max.. If you do not enter a drive or a language identification, the default drive is used and D (German) as language identification.

In order to mark a file as an input file, the # character before the file name is obligatory. Otherwise **COMPILE** is aborted and an error message is output.

Command Line Version

4.1.2 Calling with an Input File

Syntax: COMPILE #<INPUT-FILE NAME> <LANGUAGE IDENT.>
Example: COMPILE #A:INPTEST1.INP E

The input file name must meet the CPM conventions, i.e. the file name may consist of up to eight characters (letters or numbers) and the file type of three characters max.. If you do not enter a drive or a language identification, the default drive is used and D (German) as language identification.

In order to mark a file as an input file, the # character before the file name is obligatory. Otherwise **COMPILE** is aborted and an error message is output.

4 - 4 C79000-B8576-C677-02

4.2 Input File Format

The input file contains a list of the files to be edited and the desired settings. All settings are identical to those parameters, which can be entered in the command line of the individual compilation functions in the STL batch compiler package. The settings which follow the entries of the source and target file are preserved until they are edited explicitly. If any parameter is not entered, the default setting is taken (see 4.3). After each parameter block you can enter a new source and target file with new parameters.

Syntax:

```
<SOURCE FILE 1>  <TARGET FILE 1>
$BLOCK:          <BLOCK INPUT>
$SYMB:           <SYMBOLIC FILE NAME>
$OPT:            <OPTIONS>
$PCTYPE:         <LANGUAGE CATEGORY NAME>
$PRT:            <OPTION> <PRINTER FILE NAME>
<SOURCE FILE 2>  <TARGET FILE 2>
$..
$..
```

Example:

```
TEST1@A0.SEQ   TEST1@ST.S5D
$BLOCK:        FB3-10
$SYMB:         SYMB23Z0.INI
$OPT:          1
$PCTYPE:       PC135W
$PRT:          *_B:PRINT1PR.INI

CHECK2ST.S5D   CHECK2A1.SEQ
$BLOCK:        B
$OPT:          2
```

4

Command Line Version

4.2 Input File Format

The input file contains a list of the files to be edited and the desired settings. All settings are identical to those parameters, which can be entered in the command line of the individual compilation functions in the STL batch compiler package. The settings which follow the entries of the source and target file are preserved until they are edited explicitly. If any parameter is not entered, the default setting is taken (see 4.3). After each parameter block you can enter a new source and target file with new parameters.

Syntax:

<SOURCE FILE 1> <TARGET FILE 1>
$BLOCK: <BLOCK INPUT>
$SYMB: <SYMBOLIC FILE NAME>
$OPT: <OPTIONS>
$PCTYPE: <LANGUAGE CATEGORY NAME>
$PRT: <OPTION> <PRINTER FILE NAME>
<SOURCE FILE 2> <TARGET FILE 2>
$..
$..

Example:

TEST1@A0.SEQ TEST1@ST.S5D
$BLOCK: FB3-10
$SYMB: SYMB23Z0.INI
$OPT: 1
$PCTYPE: PC135W
$PRT: *_B:PRINT1PR.INI

CHECK2ST.S5D CHECK2A1.SEQ
$BLOCK: B
$OPT: 2

4

4.2.1
Entry of Source and Target File

In the first line of the input file you must enter the source and target file. The format and the combination possibilities are identical to the **COMPILE** calling without parameters (see 4.1). However, here you must not enter a language identification, as it is already written in the calling line after the input file name, if necessary.

4.2.2
$BLOCK Parameter

You can parameterize the following block data for all compilations apart from the STL source file into the intermediate file and vice versa:

OBn, FBn, PBn:	compilation of single blocks
*:	compilation of several single blocks
OBn-m, FBn-m:	compilation of block areas
OB, FB, PB:	compilation of block types
B:	compilation of all blocks (default)

(see Manual Chapter 3.4.1)

4.2.3
$SYMB Parameter

Using this option you can inform the compilation about the symbolic file name. In the STL batch compiler package it is equivalent to the symbolic file entry in the presetting mask. The file name must not consist of more than 6 characters and the drive, as the Z0.SEQ extension is added automatically. Default is the name of the STL source file or the intermediate file, resp.

Command Line Version

4.2.1
Entry of Source and Target File

In the first line of the input file you must enter the source and target file. The format and the combination possibilities are identical to the **COMPILE** calling without parameters (see 4.1). However, here you must not enter a language identification, as it is already written in the calling line after the input file name, if necessary.

4.2.2
$BLOCK Parameter

You can parameterize the following block data for all compilations apart from the STL source file into the intermediate file and vice versa:

OBn, FBn, PBn:	compilation of single blocks
*:	compilation of several single blocks
OBn-m, FBn-m:	compilation of block areas
OB, FB, PB:	compilation of block types
B:	compilation of all blocks (default)

(see Manual Chapter 3.4.1)

4.2.3
$SYMB Parameter

Using this option you can inform the compilation about the symbolic file name. In the STL batch compiler package it is equivalent to the symbolic file entry in the presetting mask. The file name must not consist of more than 6 characters and the drive, as the Z0.SEQ extension is added automatically. Default is the name of the STL source file or the intermediate file, resp.

4.2.4
$OPT: Parameter

The following options are possible for the compilation of the STL source file into the program file or of the intermediate file into the program file, resp.:

EMPTY: Create MC5 code and overwrite after confirmation

1: Create MC5 code and overwrite without confirmation

2: Do not create MC5 code, only test run

(see Manual, Chapter 3.4.1)

The following options are possible for the compilation of the program file into the STL source file or of the program file into the intermediate file, resp.:

EMPTY: Enter only symbols in the STL source file

1: Enter symbols and absolute parameters in the STL source file

2: Enter only absolute parameters in the STL source file

(see Manual, Chapter 3.4.2)

4.2.5
$PCTYPE: Parameter

This parameter determines the language category, i.e. the command supply for the corresponding PC type, during the compilation. Default value is NO.

(See Manual Chapter 2.2.3)

4.2.6
$PRT Parameter

The entry of a printer file is only valid for the compilation of the STL source file or the intermediate file into the program file. The parameter consists of two parts. The first part is the option and the second part the name of the printer file. The option always has to be placed before the file name and it offers the following three possibilities:

*: standard print

1: normal print

2: small print (with margin)

Default is EMPTY (no listing);

see Manual Chapter 3.4.1

Command Line Version

4.2.4
$OPT: Parameter

The following options are possible for the compilation of the STL source file into the program file or of the intermediate file into the program file, resp.:

EMPTY: Create MC5 code and overwrite after confirmation

1: Create MC5 code and overwrite without confirmation

2: Do not create MC5 code, only test run

(see Manual, Chapter 3.4.1)

The following options are possible for the compilation of the program file into the STL source file or of the program file into the intermediate file, resp.:

EMPTY: Enter only symbols in the STL source file

1: Enter symbols and absolute parameters in the STL source file

2: Enter only absolute parameters in the STL source file

(see Manual, Chapter 3.4.2)

4.2.5
$PCTYPE: Parameter

This parameter determines the language category, i.e. the command supply for the corresponding PC type, during the compilation. Default value is NO.

(See Manual Chapter 2.2.3)

4.2.6
$PRT Parameter

The entry of a printer file is only valid for the compilation of the STL source file or the intermediate file into the program file. The parameter consists of two parts. The first part is the option and the second part the name of the printer file. The option always has to be placed before the file name and it offers the following three possibilities:

*: standard print

1: normal print

2: small print (with margin)

Default is EMPTY (no listing);

see Manual Chapter 3.4.1

4.3 Parameter Default Values

COMPILE.CMD uses the following default values:

$BLOCK:	B	all blocks
$OPT:		
SEQ>MC5 and INT>MC5:	1	
		generate code and over-write file without asking back
MC5>SEQ and MC5>INT:	EMPTY	
		symbol entries only.
$PRT:	EMPTY	no listing.
$PCTYPE:	NO	language category NO

Command Line Version

4.3 Parameter Default Values

COMPILE.CMD uses the following default values:

$BLOCK:	B	all blocks
$OPT:		
SEQ>MC5 and INT>MC5:	1	
		generate code and over-write file without asking back
MC5>SEQ and MC5>INT:	EMPTY	
		symbol entries only.
$PRT:	EMPTY	no listing.
$PCTYPE:	NO	language category NO

4 - 8 C79000-B8576-C677-02

APPENDIX 5

APPENDIX 5

C79000-B8576-C677-02 5 - 1

ERROR MESSAGES

* Absolute parameter too long:
System error!

* Actual parameter not allowed:
Input of actual parameters only allowed after FB call

* BIB no. already exists:
The #BI control character entered more than once

* Block beginning missing:
No # character exists for absolute and/or sybolic block indentifiers

* Block in intermediate file not free from errors:
Intermediate code file (A1.SEQ) has a defect (format error).
Generate intermediate code file again using the "SEQ>INT" function from the sequential work file (A0.SEQ)

* Block name already exists:
The #N control character entered more than once

* Block too long:
Divide up program (max. 8k Bytes)

* Block type uncertain (symbol not found):
Symbolic block identifier missing for purely symbolic programming

* Block without BE:
BE command (block end identifier) missing

Error Messages

ERROR MESSAGES

* Absolute parameter too long:
System error!

* Actual parameter not allowed:
Input of actual parameters only allowed after FB call

* BIB no. already exists:
The #BI control character entered more than once

* Block beginning missing:
No # character exists for absolute and/or sybolic block indentifiers

* Block in intermediate file not free from errors:
Intermediate code file (A1.SEQ) has a defect (format error).
Generate intermediate code file again using the "SEQ>INT" function from the sequential work file (A0.SEQ)

* Block name already exists:
The #N control character entered more than once

* Block too long:
Divide up program (max. 8k Bytes)

* Block type uncertain (symbol not found):
Symbolic block identifier missing for purely symbolic programming

* Block without BE:
BE command (block end identifier) missing

* Command for PC type not allowed:
  Command for given PC type not allowed

* Command in block not allowed:
  Commands from the "supplementary operations supply" are only allowed in FB's

* Command not allowed:
  Not allowable STEP5 command

* Comment block too long:
  Divide or reduce (max. 16k Bytes) program

* Comment too long:
  System error!
  Statement comment too long (max. 40 characters)

* Compilation error:
  System error!

* Conversion error:
  Number range exeeded

* Docblock too long:
  Divide or reduce (max. 16k Bytes) program documentation

* Error in parameter:
  Invalid parameter

* Exceeded parameter range of PC type:
  For the given PC type this parameter valvue is not allowed

Error Messages

* Command for PC type not allowed:
  Command for given PC type not allowed

* Command in block not allowed:
  Commands from the "supplementary operations supply" are only allowed in FB's

* Command not allowed:
  Not allowable STEP5 command

* Comment block too long:
  Divide or reduce (max. 16k Bytes) program

* Comment too long:
  System error!
  Statement comment too long (max. 40 characters)

* Compilation error:
  System error!

* Conversion error:
  Number range exeeded

* Docblock too long:
  Divide or reduce (max. 16k Bytes) program documentation

* Error in parameter:
  Invalid parameter

* Exceeded parameter range of PC type:
  For the given PC type this parameter valvue is not allowed

C79000-B8576-C677-02

5 - 3

* Formal parameter already exists:
  Parameter name is given more than once

* Impermissible mark:
  Jump mark at an impermissible position

* Intermadiate file already exists, delete?:
  An intermediate file having identical name alredy exists

* Intermadiate file not error free:
  Defective intermediate code file (A1.SEQ) (format error).
  Regenerate the intermediate code file again from the sequential work file (A0.SEQ) using the "SEQ>INT" function

* Invalid BIB no.:
  Library no. too long, or containing unallowed characters (max. 5 digits)

* Invalid character:
  input character not allowed

* Invalid control character:
  An unpermittable control character follows the # character

* Invalid DB address:
  DB address too long, or containing unallowed characters (max. 5 digits)

* Invalid formal parameter:
  Invalid characters found in parameter name or parameter type not allowed

Error Messages

* Formal parameter already exists:
  Parameter name is given more than once

* Impermissible mark:
  Jump mark at an impermissible position

* Intermadiate file already exists, delete?:
  An intermediate file having identical name alredy exists

* Intermadiate file not error free:
  Defective intermediate code file (A1.SEQ) (format error).
  Regenerate the intermediate code file again from the sequential work file (A0.SEQ) using the "SEQ>INT" function

* Invalid BIB no.:
  Library no. too long, or containing unallowed characters (max. 5 digits)

* Invalid character:
  input character not allowed

* Invalid control character:
  An unpermittable control character follows the # character

* Invalid DB address:
  DB address too long, or containing unallowed characters (max. 5 digits)

* Invalid formal parameter:
  Invalid characters found in parameter name or parameter type not allowed

5 - 4

C79000-B8576-C677-02

* Invalid intermediate format:
  Defective intermediate code file (A1.SEQ) (format error).
  Regenerate the intermediate code file again from the sequential work file (A0.SEQ) using the "SEQ>INT" function

* Invalid line:
  Observe the order (control characters during the block input

* Invalid mark:
  Jump mark not allowable character

* Invalid operator:
  Undefined operator in STEP 5

* Line not allowed:
  Observe the order (control characters) during the block input

* Line was not processed:
  block type is uncertain

* Mark are equal:
  Jump marks exist more than once

* Mark too long:
  Jump destination (symbol address) enterd in the "STARTMENT" field too long (max. 4 characters)

* Nested include command (instruction) not allowed:
  An intermediate file, which has been included with #INCLUDE, contains another include command

* No actual parameter given:
  The required actual parameter input is missing in the "STARTMENT" FIEDL afte FB call

Error Messages

* Invalid intermediate format:
  Defective intermediate code file (A1.SEQ) (format error).
  Regenerate the intermediate code file again from the sequential work file (A0.SEQ) using the "SEQ>INT" function

* Invalid line:
  Observe the order (control characters during the block input

* Invalid mark:
  Jump mark not allowable character

* Invalid operator:
  Undefined operator in STEP 5

* Line not allowed:
  Observe the order (control characters) during the block input

* Line was not processed:
  block type is uncertain

* Mark are equal:
  Jump marks exist more than once

* Mark too long:
  Jump destination (symbol address) enterd in the "STARTMENT" field too long (max. 4 characters)

* Nested include command (instruction) not allowed:
  An intermediate file, which has been included with #INCLUDE, contains another include command

* No actual parameter given:
  The required actual parameter input is missing in the "STARTMENT" FIEDL afte FB call

5

C79000-B8576-C677-02

5 - 5

* No BIB number given:
  The input of the libaray number is missing sfter #BI

* No block name given:
  The #N control character is missing in the function block an its name

* No date given:
  The constant valvue in the "OPERAND SYMBOL" field is missing after entering the constant type in the "STARTMENT" field

* No formal operand given:
  After calling the FB the assigned formal parameter declaration is missing in the "ADR" filed of the FB, which goes along with the given actual parameter

* No operand identifier given:
  Operand identifier missing

* No parameter given:
  Parameter input missing (for purely absolute programming)

* No symbol given:
  Operand symbol missing (for purely symbolic programming)

* Only after FB call:
  Actual parameters are only allowed immediately after an FB call

* Only allowed with STL blocks:
  only with STL blocks

Error Messages

* No BIB number given:
  The input of the libaray number is missing sfter #BI

* No block name given:
  The #N control character is missing in the function block an its name

* No date given:
  The constant valvue in the "OPERAND SYMBOL" field is missing after entering the constant type in the "STARTMENT" field

* No formal operand given:
  After calling the FB the assigned formal parameter declaration is missing in the "ADR" filed of the FB, which goes along with the given actual parameter

* No operand identifier given:
  Operand identifier missing

* No parameter given:
  Parameter input missing (for purely absolute programming)

* No symbol given:
  Operand symbol missing (for purely symbolic programming)

* Only after FB call:
  Actual parameters are only allowed immediately after an FB call

* Only allowed with STL blocks:
  only with STL blocks

5 - 6

C79000-B8576-C677-02

* Only one headling per segment:
  The #UB control character is given more than once at the beginning of the segment

* Only with function blocks:
  extended command set allowed

* Operand identifier not allowed:
  Operand identifier does not match operator

* Operand not allowed:
  No operand allowed

* Operand too long:
  Operand identifier too long (max. 2 characters)

* Operator not given:
  Missing operator input for symbolic programming

* Operator too long:
  (max. 3 characters)

* Parameter not allowed:
  No parameter allowed

* Parameter too long (max. 4 characters):
  Formal parameter enterd in "STATEMENT" field too long (max. 4 lines)

* PC type not allowed:
  Invalid input of PC type

Error Messages

* Only one headling per segment:
  The #UB control character is given more than once at the beginning of the segment

* Only with function blocks:
  extended command set allowed

* Operand identifier not allowed:
  Operand identifier does not match operator

* Operand not allowed:
  No operand allowed

* Operand too long:
  Operand identifier too long (max. 2 characters)

* Operator not given:
  Missing operator input for symbolic programming

* Operator too long:
  (max. 3 characters)

* Parameter not allowed:
  No parameter allowed

* Parameter too long (max. 4 characters):
  Formal parameter enterd in "STATEMENT" field too long (max. 4 lines)

* PC type not allowed:
  Invalid input of PC type

C79000-B8576-C677-02

5 - 7

* Read error:
  Disk error, defective file
* Segment end missing or segment too long:
  Segment end character (***) or image structure command for segment end missing (BLD 255), or segment too long (max. 255 lines)
* Symbol does not match absolute parameter:
  The absolute and symbol operand in the STL source file are assigned differently
* Symbol not allowed:
  Command does not aalow operand input
* Symbol too long:
  System error! (max. 24 characters)
* Symbolic file does not exist:
  Symbolic file missing for purely symbolic programming
* Symbolic source file existing, overwrite?:
  A Symbolic source file with identical filename already exists (function SYM-GEN)
* SYS Comand not allowed:
  Not applicable
* System command not allowed:
  Not applicable
* Too many actual parameters:
  (max. 40)

Error Messages

* Read error:
  Disk error, defective file
* Segment end missing or segment too long:
  Segment end character (***) or image structure command for segment end missing (BLD 255), or segment too long (max. 255 lines)
* Symbol does not match absolute parameter:
  The absolute and symbol operand in the STL source file are assigned differently
* Symbol not allowed:
  Command does not aalow operand input
* Symbol too long:
  System error! (max. 24 characters)
* Symbolic file does not exist:
  Symbolic file missing for purely symbolic programming
* Symbolic source file existing, overwrite?:
  A Symbolic source file with identical filename already exists (function SYM-GEN)
* SYS Comand not allowed:
  Not applicable
* System command not allowed:
  Not applicable
* Too many actual parameters:
  (max. 40)

5 - 8

C79000-B8576-C677-02

* Too many formal parameters:
(max. 40)

* Undefinded operand identifier:
Operand identifier in STEP5 undefinded

* Undefined command:
Nonexistent MC5 command. Defective program file (ST.S5D)

* Undefined formal parameter:
Parameter name and type not definded in FB

* Undefined mark:
Jump mark for the given jump destination (symbol address) not enterd in the "ADR" field

* Wrong data block in SEQ file:
(A1.SEQ) defective sequential work file (A0.SEQ)(format error)

* Wrong data position:
The value is to be entered in the "OPERAND SYMBOL" field for constants

* Wrong format:
Input format not correct

* Wrong function number:
System error!

* Wrong intermediate file identifier:
File was created with tools from a different output state.
Regenerate the intermediate code file again from the sequential work file (A0.SEQ) using the "SEQ>INT" function

Error Messages

* Too many formal parameters:
(max. 40)

* Undefinded operand identifier:
Operand identifier in STEP5 undefinded

* Undefined command:
Nonexistent MC5 command. Defective program file (ST.S5D)

* Undefined formal parameter:
Parameter name and type not definded in FB

* Undefined mark:
Jump mark for the given jump destination (symbol address) not enterd in the "ADR" field

* Wrong data block in SEQ file:
(A1.SEQ) defective sequential work file (A0.SEQ)(format error)

* Wrong data position:
The value is to be entered in the "OPERAND SYMBOL" field for constants

* Wrong format:
Input format not correct

* Wrong function number:
System error!

* Wrong intermediate file identifier:
File was created with tools from a different output state.
Regenerate the intermediate code file again from the sequential work file (A0.SEQ) using the "SEQ>INT" function

C79000-B8576-C677-02

5 - 9

* Wrong nesting depth:
  Nesting end not equal (observe nesting level)

* Wrong number of parameters:
  Number of declared formal parameters in FB does not equal the number of given actual parameters after FB call (test run)

* Wrong parameter type:
  The formal parameter type given in the "STATEMENT" field of the FB does not equal the actual parameters assigned after FB call

Error Messages

* Wrong nesting depth:
  Nesting end not equal (observe nesting level)

* Wrong number of parameters:
  Number of declared formal parameters in FB does not equal the number of given actual parameters after FB call (test run)

* Wrong parameter type:
  The formal parameter type given in the "STATEMENT" field of the FB does not equal the actual parameters assigned after FB call

5 - 10

C79000-B8576-C677-02

Dividing page

C79000-D8376-C018-01

1

Dividing page

2

C79000-D8376-C018-01

SIEMENS

Editeur LIST/ Compilateur Batch

Description

C79000-G8577-C677

2ème édition

SIEMENS

Editeur LIST/ Compilateur Batch

Description

Introduction 1

Description 2

Commande à la console de programmation 3

Version lignes de commande 4

Annexe 5

C79000-G8577-C677

2ème édition

Siemens Aktiengesellschaft

C79000-G8577-C677-02
EWK Elektronikwerk Karlsruhe

Imprimé en République Fédérale d'Allemange

Siemens Aktiengesellschaft

C79000-G8577-C677-02
EWK Elektronikwerk Karlsruhe

Imprimé en République Fédérale d'Allemange

Sommaire

Sommaire

Sommaire

1 Introduction 1-1

2 Description 2-1

2.1 Mode opératoire de l'éditeur LIST/compilateur par lots 2-2

2.2 Création de blocs STEP 5 2-7

2.2.1 Généralités 2-7

2.2.2 Fonctions d'éditeur 2-9

2.2.3 Caractères de commande 2-11

2.2.4 Conversion 2-13

2.2.5 Imprimer 2-15

2.3 Le fichier intermédiaire A1.SEQ 2-16

2.3.1 Relations entre fichier source LIST et fichier intermédiaire 2-16

2.3.2 Fonctions spéciales 2-18

2.3.3 Programmes standard 2-19

2.3.4 Versions d'un programme STEP 5 en langue étrangère 2-20

2.4 Modifier et supplémenter des blocs STEP 5 2-21

2.4.1 Blocs créés à l'aide de l'éditeur LIST 2-21

2.4.2 Blocs créés à l'aide du paquet CONT-LOG-LIST 2-21

2.5 Contrôle 2-22

2.5.1 Contrôle d'un fichier programme 2-22

2.5.2 Contrôle des blocs spéciaux 2-22

2.6 Liste d'erreurs 2-23

2.6.1 Fichier d'erreurs 2-23

Sommaire

2.7 Introduction d'instructions STEP 5 avec d'autres éditeurs 2-24
2.7.1 Fichier source LIST comme interface 2-24
2.7.2 Format du fichier source séquentiel 2-24
2.7.3 Caractère de commande #TAB pour le traitement de fichiers étrangers 2-25

3 Commande à la console de programmation 3-1
3.1 Installer le logiciel 3-2
3.2 Phases de commande jusqu'à la sélection de fonction 3-3
3.2.1 Généralités de la commande 3-3
3.2.2 Introduction au paquet éditeur LIST/compilateur par lots 3-3
3.3 Editer 3-8
3.3.1 Description de l'éditeur 3-9
3.3.2 Les caractères de commande de l'éditeur LIST/compilateur par lots et leur conventions d'écriture 3-14
3.3.3 Les opérations STEP 5 dans l'éditeur LIST/compilateur par lots et leurs conventions d'écriture 3-20
3.3.4 Introduction de blocs programme 3-25
3.3.5 Utilisation de la barre des touches de fonction EDITEUR 3-30
3.3.6 Introduction de blocs fonctionnels 3-41
3.3.7 Introduction de blocs de données 3-48
3.3.8 Modification d'un fichier source LIST 3-53
3.4 Conversion avec la fonction COMPILER 3-52
3.4.1 Suite d'opérations: Conversion dans le fichier programme 3-52
3.4.2 Suite d'opérations: Reconvertir dans le fichier programme 3-54
3.5 Liste d'erreurs 3-55
3.6 Imprimer 3-56

0 - 4 C79000-B8577-C677-02

3.7 Fonctions SPECIALES pour le traitement de fichiers intermédiaires et source 3-57
3.7.1 COPIER 3-57
3.7.2 SEQ>INT 3-58
3.7.3 INT>SEQ 3-58
3.7.4 SEQEFF et INTEFF 3-59
3.7.5 CONTROLE 3-59
3.7.6 GEN-SYM 3-60

4 Version lignes de commande 4-1
4.1 Appel 4-3
4.1.1 Appel sans donnée de paramètre 4-3
4.1.2 Appel avec fichier input 4-4
4.2 Format du fichier input 4-5
4.2.1 Indication de fichier source et destination 4-6
4.2.2 Paramètre $BLOC: 4-6
4.2.3 Paramètre $SYMB: 4-6
4.2.4 Paramètre $OPT 4-7
4.2.5 Paramètre $ type d'AG 4-7
4.2.6 Paramètre $IMP 4-7
4.3 Valeurs par défaut (default) des paramètres 4-8

5 Annexe 5-1
5.1 Messages d'erreurs 5-2

Sommaire

0 - 6

C79000-B8577-C677-02

Introduction 1

Introduction 1

C79000-B8577-C677-02

1 - 1

Les fonctions

Le paquet optionnel éditeur LIST/compilateur par lots vous offre un éditeur indépendant pour les programmes en mode de réprésentation LIST et un compilateur indépendant pour la conversion de telles listes d'instructions dans un programme STEP 5 exécutable.

Jusqu'à présent, vous ne pouviez pas créer un programme STEP 5 qu'en le paquet de base avec le paquet CONT-LOG-LIST. Le confort d'édition de ce paquet est pourtant restricté, et vous devez connaître le filage de votre installation pour connecter les entrées et les sorties correctement.

L'éditeur LIST vous offre la possibilité de créer votre programme avec un grand confort de commande, vous pouvez par exemple copier des segments. Avec ce paquet, il est également possible d'écrire le programme exclusivement sous forme symbolique. Cependant, vous ne devez pas effectuer une assignation definitive du symbol et de l'entrée/la sortie d'avance mais vous pouvez le faire plus tard, par exemple quand votre installation sera complète. A l'aide du compilateur par lots, ces assignations seront ensuite liées à la liste d'instructions créée dans l'éditeur et après converties dans un programme STEP 5 qui est exécuté sur votre installation. Ce programme peut être testé et corrigé sur l'appareil d'automatisation.

A l'aide du compilateur par lots, il est aussi possible d'effectuer des reconversions d'un programme STEP 5 de sorte que p. ex. les modifications du programme testé peuvent être entrées dans votre source et votre liste d'instructions sera actualisée. Un programme STEP 5 créé dans le paquet CONT-LOG-LIST peut être traité de même façon.

Le paquet optionnel éditeur LIST/compilateur par lots vous offre beaucoup d'avantages:

1. *A l'aide de symboles, vous pouvez créer un programme spécifique de la tâche qui, avec une liste d'assignations individuelle, devient un programme STEP 5 spécifique de l'installation. C.-à.-d., vous pouvez établir une bibliothèque de programmes standardisés.*
2. *A l'aide d'une instruction Include, des programmes et modules standardisés peuvent être combinés toujours à nouveau et liés ensemble pour faire un programme STEP 5 individuel sans que vous deviez effectuer de vastes modifications dans les programmes mêmes.*

Introduction

Les fonctions

Le paquet optionnel éditeur LIST/compilateur par lots vous offre un éditeur indépendant pour les programmes en mode de réprésentation LIST et un compilateur indépendant pour la conversion de telles listes d'instructions dans un programme STEP 5 exécutable.

Jusqu'à présent, vous ne pouviez pas créer un programme STEP 5 qu'en le paquet de base avec le paquet CONT-LOG-LIST. Le confort d'édition de ce paquet est pourtant restricté, et vous devez connaître le filage de votre installation pour connecter les entrées et les sorties correctement.

L'éditeur LIST vous offre la possibilité de créer votre programme avec un grand confort de commande, vous pouvez par exemple copier des segments. Avec ce paquet, il est également possible d'écrire le programme exclusivement sous forme symbolique. Cependant, vous ne devez pas effectuer une assignation definitive du symbol et de l'entrée/la sortie d'avance mais vous pouvez le faire plus tard, par exemple quand votre installation sera complète. A l'aide du compilateur par lots, ces assignations seront ensuite liées à la liste d'instructions créée dans l'éditeur et après converties dans un programme STEP 5 qui est exécuté sur votre installation. Ce programme peut être testé et corrigé sur l'appareil d'automatisation.

A l'aide du compilateur par lots, il est aussi possible d'effectuer des reconversions d'un programme STEP 5 de sorte que p. ex. les modifications du programme testé peuvent être entrées dans votre source et votre liste d'instructions sera actualisée. Un programme STEP 5 créé dans le paquet CONT-LOG-LIST peut être traité de même façon.

Le paquet optionnel éditeur LIST/compilateur par lots vous offre beaucoup d'avantages:

1. *A l'aide de symboles, vous pouvez créer un programme spécifique de la tâche qui, avec une liste d'assignations individuelle, devient un programme STEP 5 spécifique de l'installation. C.-à.-d., vous pouvez établir une bibliothèque de programmes standardisés.*
2. *A l'aide d'une instruction Include, des programmes et modules standardisés peuvent être combinés toujours à nouveau et liés ensemble pour faire un programme STEP 5 individuel sans que vous deviez effectuer de vastes modifications dans les programmes mêmes.*

1 - 2 C79000-B8577-C677-02

3. *Dans l'éditeur LIST, vous avez un confort de commande beaucoup plus grand que dans le paquet CONT-LOG-LIST. Ici, vous pouvez copier et déplacer des segments, des parties de programmes et des blocs. Quant à la commande de la console de programmation, l'éditeur LIST ressemble à l'éditeur symbolique.*
4. *Des parties de programme qui reviennent fréquement peuvent être transférées à d'autres fichiers et peuvent être inclues à chaque point de votre fichier ouvert ou d'un autre fichier à l'aide d'instructions de copiage et Include.*
5. *Vous pouvez également créer votre liste d'instructions avec un autre éditeur car le compilateur par lots est capable de convertir des fichiers séquentiels (fichiers ASCII) dans un programme STEP 5. Ainsi, vous êtes ouvert à d'autres systèmes.*
6. *Comme la conversion est effectuée via un fichier intermédiaire, votre liste d'instructions peut être traitée également avec un éditeur LIST en langue étrangère.*
7. *Des programmes STEP 5 déjà existant qui sont écrits dans le paquet CONT-LOG-LIST seront reconvertis avec le compilateur par lots. Ils sont alors à votre disposition comme une base pour des programmes standard et des versions en langue étrangère.*

Comme fonctions additionnels, ce paquet d'option offre une contrôle spécifique de l'AG pour le programme STEP 5 converti et une liste d'erreurs.

Dans les chapitres suivants, vous trouvez une description détaillée de l'éditeur LIST et du compilateur par lots qui vous explique le mode opératoire et les fonctions diverses. Des instructions d'utilisation vous introduisent à la commande de la console de programmation à l'aide d'un exemple pratique. Les instructions d'utilisation contiennent des tableaux synoptiques qui concerne les caractères de commande, les opérations STEP 5 et les conventions d'écriture.

Pour une lisibilité plus grande, les mots sont marqués

- en **gros** s'il s'agit des touches de la console,
- en *italique*, si des fonctions du logiciel sont nommées (p. ex. validation) et si vous êtes demandé d'introduire du texte vous-même (dans l'exemple).

Introduction

3. *Dans l'éditeur LIST, vous avez un confort de commande beaucoup plus grand que dans le paquet CONT-LOG-LIST. Ici, vous pouvez copier et déplacer des segments, des parties de programmes et des blocs. Quant à la commande de la console de programmation, l'éditeur LIST ressemble à l'éditeur symbolique.*
4. *Des parties de programme qui reviennent fréquement peuvent être transférées à d'autres fichiers et peuvent être inclues à chaque point de votre fichier ouvert ou d'un autre fichier à l'aide d'instructions de copiage et Include.*
5. *Vous pouvez également créer votre liste d'instructions avec un autre éditeur car le compilateur par lots est capable de convertir des fichiers séquentiels (fichiers ASCII) dans un programme STEP 5. Ainsi, vous êtes ouvert à d'autres systèmes.*
6. *Comme la conversion est effectuée via un fichier intermédiaire, votre liste d'instructions peut être traitée également avec un éditeur LIST en langue étrangère.*
7. *Des programmes STEP 5 déjà existant qui sont écrits dans le paquet CONT-LOG-LIST seront reconvertis avec le compilateur par lots. Ils sont alors à votre disposition comme une base pour des programmes standard et des versions en langue étrangère.*

Comme fonctions additionnels, ce paquet d'option offre une contrôle spécifique de l'AG pour le programme STEP 5 converti et une liste d'erreurs.

Dans les chapitres suivants, vous trouvez une description détaillée de l'éditeur LIST et du compilateur par lots qui vous explique le mode opératoire et les fonctions diverses. Des instructions d'utilisation vous introduisent à la commande de la console de programmation à l'aide d'un exemple pratique. Les instructions d'utilisation contiennent des tableaux synoptiques qui concerne les caractères de commande, les opérations STEP 5 et les conventions d'écriture.

Pour une lisibilité plus grande, les mots sont marqués

- en **gros** s'il s'agit des touches de la console,
- en *italique*, si des fonctions du logiciel sont nommées (p. ex. validation) et si vous êtes demandé d'introduire du texte vous-même (dans l'exemple).

C79000-B8577-C677-02

1 - 3

Introduction

1 - 4

C79000-B8577-C677-02

Description 2

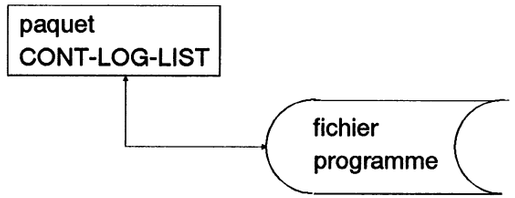
Description 2

C79000-B8577-C677-02 2 - 1

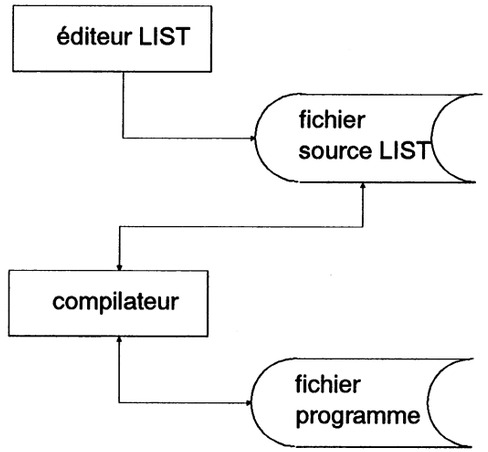
2.1 Mode opératoire de l'éditeur LIST/compilateur par lots

L'écriture d'un programme STEP 5 au sein de l'éditeur LIST/du compilateur par lots se distingue du paquet CONT-LOG-LIST par le point suivant:

Dans le paquet CONT-LOG-LIST, la liste d'instructions est éditée directement dans le fichier programme et est convertie immédiatement en code machine.



Dans le paquet CONT-LOG-LIST, *éditer* et *convertir* sont des processus séparés temporellement.

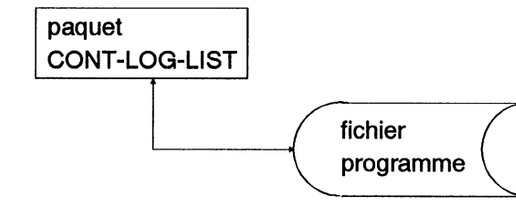


Description

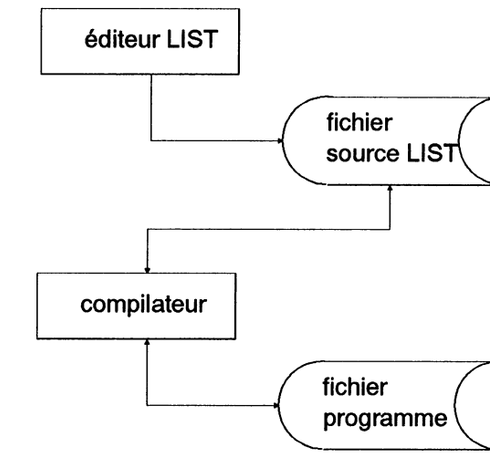
2.1 Mode opératoire de l'éditeur LIST/compilateur par lots

L'écriture d'un programme STEP 5 au sein de l'éditeur LIST/du compilateur par lots se distingue du paquet CONT-LOG-LIST par le point suivant:

Dans le paquet CONT-LOG-LIST, la liste d'instructions est éditée directement dans le fichier programme et est convertie immédiatement en code machine.



Dans le paquet CONT-LOG-LIST, *éditer* et *convertir* sont des processus séparés temporellement.

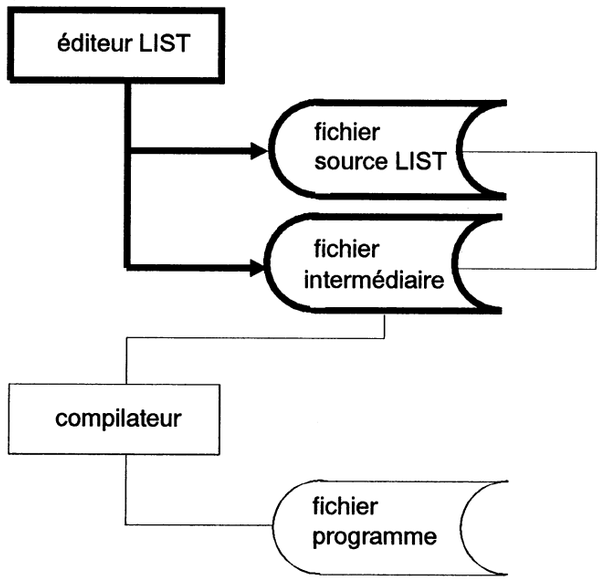


éditer

Lors de la première phase, qui est *éditer*, vous écrivez un fichier de texte séquentiel à l'aide de l'éditeur LIST, c'est le fichier source LIST. Il peut contenir une liste d'instructions qui est écrite uniquement via des symboles.

valider

Quand vous sauvegardez avec la fonction *valider* ou avec la touche de validation, automatiquement le paquet dresse un fichier intermédiaire en plus du fichier source LIST. Cette fichier intermédiaire contient un code qui est indépendant des langues nationales mais qui n'est pas encore un code machine. Lors de cette première*conversion* la syntaxe et le format de votre liste d'instructions sont contrôlés.



Description

éditer

Lors de la première phase, qui est éditer, vous écrivez un fichier de texte séquentiel à l'aide de l'éditeur LIST, c'est le fichier source LIST. Il peut contenir une liste d'instructions qui est écrite uniquement via des symboles.

valider

Quand vous sauvegardez avec la fonction valider ou avec la touche de validation, automatiquement le paquet dresse un fichier intermédiaire en plus du fichier source LIST. Cette fichier intermédiaire contient un code qui est indépendant des langues nationales mais qui n'est pas encore un code machine. Lors de cette premièreconversion la syntaxe et le format de votre liste d'instructions sont contrôlés.

2

éditeur LIST
fichier source LIST
fichier intermédiaire
compilateur
fichier programme

C79000-B8577-C677-02

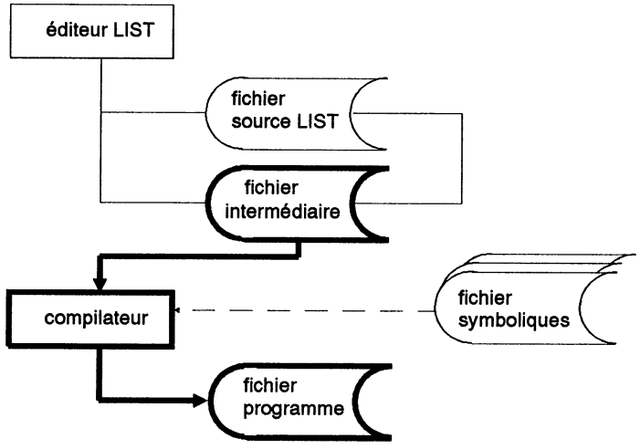
2 - 3

conversion

La deuxième phase, la *conversion*, est démarrée de vous même par une touche de fonction. Pendant celà, le compilateur par lots transforme le fichier intermédiaire dans un fichier programme STEP 5. Si vous avez programmé votre liste d'instructions sous forme symbolique, le compilateur par lots à ce point nécessite un fichier symbolique avec des assignations spécifiques de l'installation.

contrôle

Lors de la *conversion* dans le fichier programme un *contrôle* des assignations est effectuée. Si vous avez spécifié un type d'AG, il y a également un contrôle si les opérations utilisées sont admissibles pour votre AG source (*contrôle spécifique de l'AG*). Un fichier programme qui a été créé à l'aide de l'éditeur LIST/du compilateur par lots est identique avec un fichier programme créé au sein du paquet CONT-LOG-LIST.



Remarque: la conversion du fichier source LIST peut se faire aussi en une phase, p.ex. lorsque le fichier source a été uniquement sauvegardé et qu'aucun fichier intermédiaire n'a été créé, ou bien lorsque le fichier source a été créé avec un éditeur étranger.

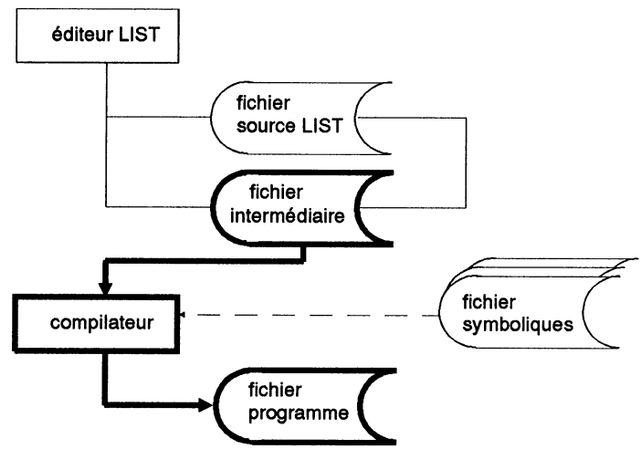
Description

conversion

La deuxième phase, la *conversion*, est démarrée de vous même par une touche de fonction. Pendant celà, le compilateur par lots transforme le fichier intermédiaire dans un fichier programme STEP 5. Si vous avez programmé votre liste d'instructions sous forme symbolique, le compilateur par lots à ce point nécessite un fichier symbolique avec des assignations spécifiques de l'installation.

contrôle

Lors de la *conversion* dans le fichier programme un *contrôle* des assignations est effectuée. Si vous avez spécifié un type d'AG, il y a également un contrôle si les opérations utilisées sont admissibles pour votre AG source (*contrôle spécifique de l'AG*). Un fichier programme qui a été créé à l'aide de l'éditeur LIST/du compilateur par lots est identique avec un fichier programme créé au sein du paquet CONT-LOG-LIST.

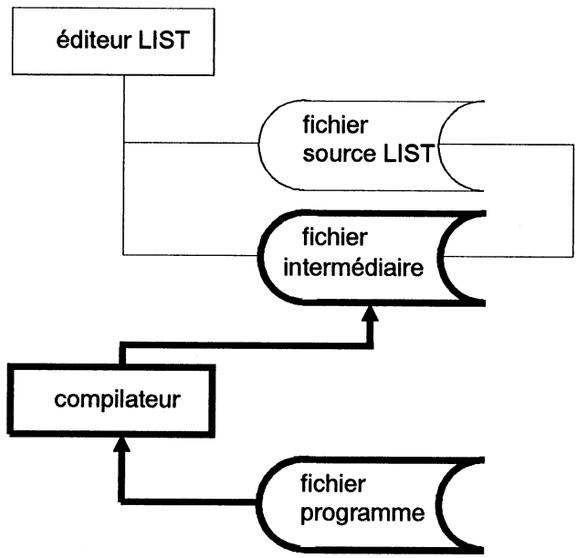


Remarque: la conversion du fichier source LIST peut se faire aussi en une phase, p.ex. lorsque le fichier source a été uniquement sauvegardé et qu'aucun fichier intermédiaire n'a été créé, ou bien lorsque le fichier source a été créé avec un éditeur étranger.

Avec l'éditeur LIST/le compilateur par lots, vous pouvez également créer un fichier source en utilisant un fichier programme. Ça peut être nécessaire p. ex. après qu'un programme STEP 5 a été testé et corrigé dans l'AG. Pour cela, il n'est pas important si le programme a été édité au sein du paquet CONT-LOG-LIST ou du paquet éditeur LIST/compilateur par lots.

reconversion

Au cours d'une telle *reconversion* le compilateur par lots génère d'abord un fichier intermédiaire en utilisant le fichier programme. Ensuite, cette fichier intermédiaire est employé pour produire le fichier source LIST du fichier programme.



Remarque: la création d'un fichier source LIST en utilisant un fichier programme peut se faire aussi en une seule phase

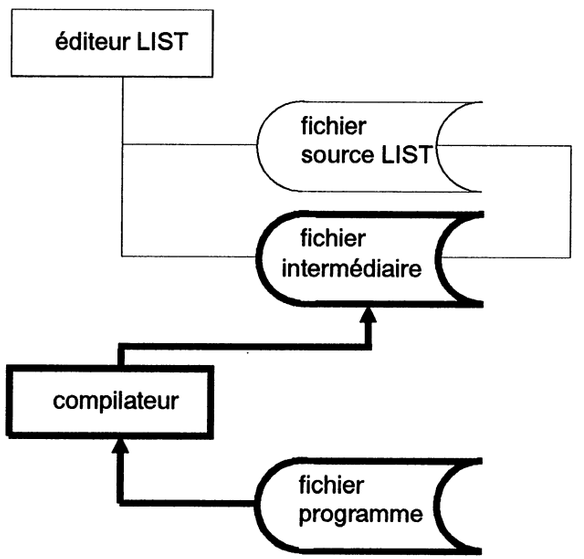
Description

Avec l'éditeur LIST/le compilateur par lots, vous pouvez également créer un fichier source en utilisant un fichier programme. Ça peut être nécessaire p. ex. après qu'un programme STEP 5 a été testé et corrigé dans l'AG. Pour cela, il n'est pas important si le programme a été édité au sein du paquet CONT-LOG-LIST ou du paquet éditeur LIST/compilateur par lots.

reconversion

Au cours d'une telle *reconversion* le compilateur par lots génère d'abord un fichier intermédiaire en utilisant le fichier programme. Ensuite, cette fichier intermédiaire est employé pour produire le fichier source LIST du fichier programme.

2

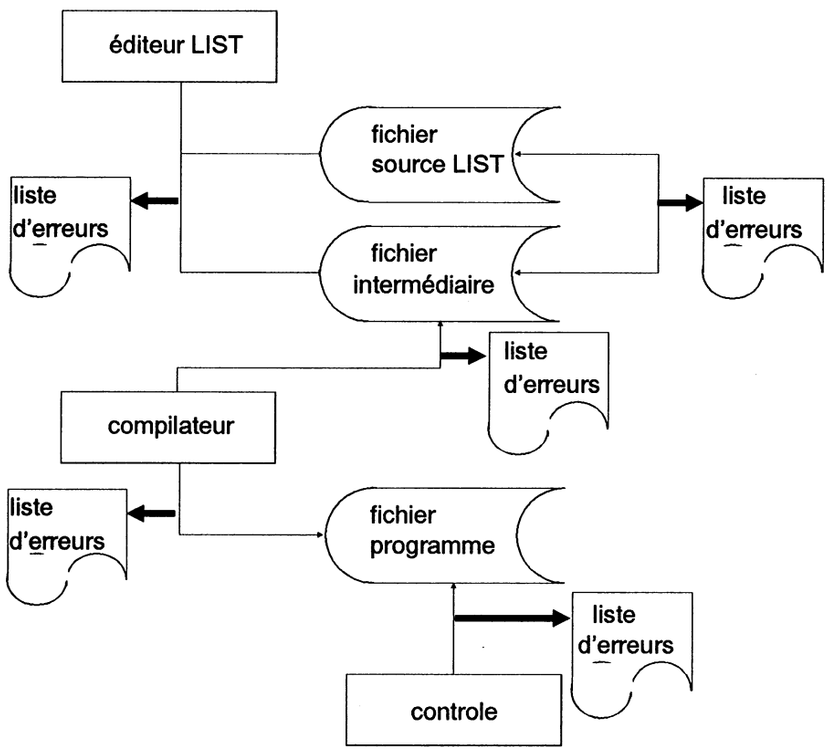


Remarque: la création d'un fichier source LIST en utilisant un fichier programme peut se faire aussi en une seule phase

contrôle

Comme déjà mentionné, des *contrôles* sont réalisés au cours de la *conversion*. En outre, il y a un *contrôle pour les blocs du fichier programme* après la *conversion*. Cette contrôle vérifie p. ex. si les opérandes formels et actuels des blocs de fonctions sont assignés correctement et s'ils concordent. Toutes les erreurs qui apparaissent sont inscrites dans une liste d'erreurs et peuvent être sorties sur l'imprimante.

Dans la liste d'erreurs, on ne trouve cependant que les erreurs de la dernière phase de travail; elle est écrasée avec chaque nouveau *conversion* ou *contrôle*. Par conséquent, sortez votre liste d'erreurs toujours sur l'imrimante! Lorsqu'une phase de travail est correcte, aucune liste d'erreur n'est dressée et, si celle-ci existe éventuellement, elle est effacée.

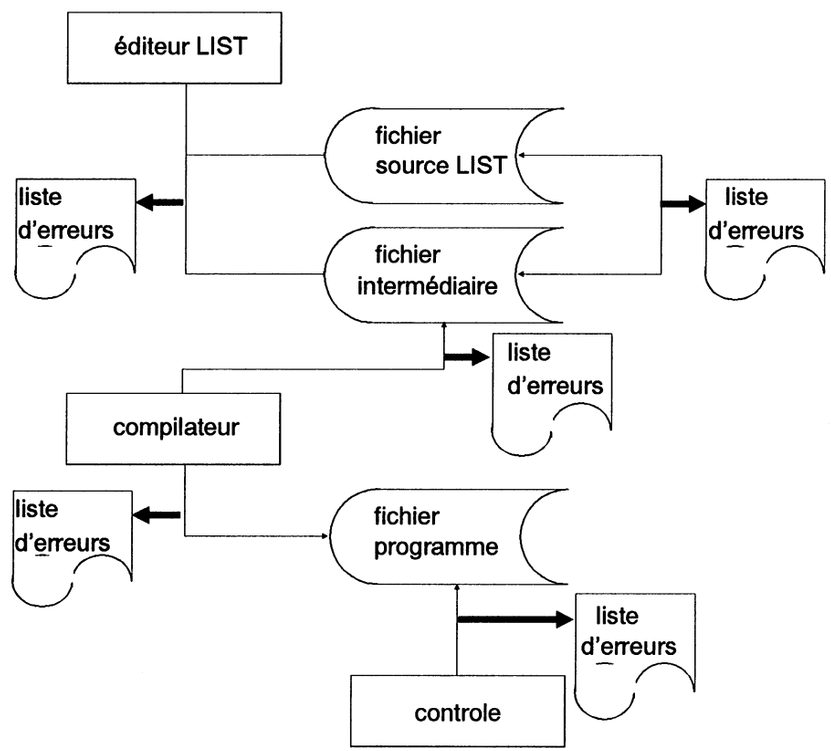


Description

contrôle

Comme déjà mentionné, des *contrôles* sont réalisés au cours de la *conversion*. En outre, il y a un *contrôle pour les blocs du fichier programme* après la *conversion*. Cette contrôle vérifie p. ex. si les opérandes formels et actuels des blocs de fonctions sont assignés correctement et s'ils concordent. Toutes les erreurs qui apparaissent sont inscrites dans une liste d'erreurs et peuvent être sorties sur l'imprimante.

Dans la liste d'erreurs, on ne trouve cependant que les erreurs de la dernière phase de travail; elle est écrasée avec chaque nouveau *conversion* ou *contrôle*. Par conséquent, sortez votre liste d'erreurs toujours sur l'imrimante! Lorsqu'une phase de travail est correcte, aucune liste d'erreur n'est dressée et, si celle-ci existe éventuellement, elle est effacée.



2 - 6
C79000-B8577-C677-02

2.2 Création de blocs STEP 5

2.2.1 Généralités

Au sein de l'éditeur LIST, vous créez votre programme de commande sous forme de liste d'instructions avec le confort d'un éditeur de texte. Vous écrivez votre liste d'instructions avec le même stock d'instructions et en utilisant le même syntaxe qu'avec le paquet CONT-LOG-LIST. La seule différence est que vous devez observer certaines conventions comme p. ex. des caractères de commande pour des débuts de blocs et des commentaires.

Toutes sortes de blocs qui sont possibles au sein du paquet CONT-LOG-LIST peuvent également être créées avec le paquet éditeur LIST/compilateur par lots. Tous commentaires qui sont possible dans un bloc peuvent également être écrits de cette façon sauf les commentaires d'installation (fichier DOC). De plus, l'éditeur LIST permet des "commentaires supplémentaires" à chaque point de la liste d'instructions. Cependant, ces commentaires ne sont pas transférés au fichier programme et se perdent lors d'une reconversion dans le même source LIST.

Un DB0 n'est pas possible (reservé dans l' AG pour la liste d'adressage de bloc), ainsi qu'un DB1 (pour l'assignation de la périphérie de l' AG 135U et AG 155U), un DX0 (pour la paramétrage du système, CPU 928, processeur R, AG 155U), un DB2 (pour la liste de réglage du R64), des blocs GRAPH 5 et blocs en assembleur.

Chap. 3 des instructions d'utilisation vous explique en détail comment il faut éditer et quels caractères de commande et quelles conventions d'écriture vous devez observer.

Vos instructions ne seront pas inscrites immédiatement dans le fichier source LIST sur disque dur ou disquette mais dans un tampon d'éditeur dans la mémoire de travail de votre console de programmation. Le contenu du tampon d'éditeur, c.-à-d. votre liste d'instructions, ne sera mémorisé dans le fichier source LIST qu'avec les fonctions de mémorisation *sauvegarder* ou *valider*. Notez les différences suivantes:

- Avec *sauvegarder*, vous ne mémorisez que votre fichier source LIST.
- Avec *valider*, vous mémorisez votre fichier source LIST et le conversez dans le fichier intermédiaire.

Description

2.2 Création de blocs STEP 5

2.2.1 Généralités

Au sein de l'éditeur LIST, vous créez votre programme de commande sous forme de liste d'instructions avec le confort d'un éditeur de texte. Vous écrivez votre liste d'instructions avec le même stock d'instructions et en utilisant le même syntaxe qu'avec le paquet CONT-LOG-LIST. La seule différence est que vous devez observer certaines conventions comme p. ex. des caractères de commande pour des débuts de blocs et des commentaires.

2

Toutes sortes de blocs qui sont possibles au sein du paquet CONT-LOG-LIST peuvent également être créées avec le paquet éditeur LIST/compilateur par lots. Tous commentaires qui sont possible dans un bloc peuvent également être écrits de cette façon sauf les commentaires d'installation (fichier DOC). De plus, l'éditeur LIST permet des "commentaires supplémentaires" à chaque point de la liste d'instructions. Cependant, ces commentaires ne sont pas transférés au fichier programme et se perdent lors d'une reconversion dans le même source LIST.

Un DB0 n'est pas possible (reservé dans l' AG pour la liste d'adressage de bloc), ainsi qu'un DB1 (pour l'assignation de la périphérie de l' AG 135U et AG 155U), un DX0 (pour la paramétrage du système, CPU 928, processeur R, AG 155U), un DB2 (pour la liste de réglage du R64), des blocs GRAPH 5 et blocs en assembleur.

Chap. 3 des instructions d'utilisation vous explique en détail comment il faut éditer et quels caractères de commande et quelles conventions d'écriture vous devez observer.

Vos instructions ne seront pas inscrites immédiatement dans le fichier source LIST sur disque dur ou disquette mais dans un tampon d'éditeur dans la mémoire de travail de votre console de programmation. Le contenu du tampon d'éditeur, c.-à-d. votre liste d'instructions, ne sera mémorisé dans le fichier source LIST qu'avec les fonctions de mémorisation *sauvegarder* ou *valider*. Notez les différences suivantes:

- Avec *sauvegarder*, vous ne mémorisez que votre fichier source LIST.
- Avec *valider*, vous mémorisez votre fichier source LIST et le conversez dans le fichier intermédiaire.

C79000-B8577-C677-02

2 - 7

- Avec *abandon* (touche d'abandon), vous rejetez le fichier entier édité dans la mémoire de travail et quittez l'éditeur.

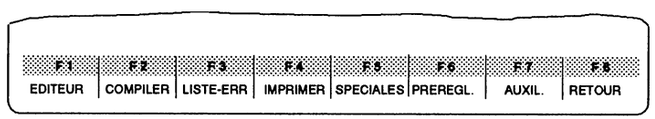
Remarque: Bien que le tampon d'éditeur n'ait qu'une grandeur spécifiée et finie (cette grandeur apparaît dans le masque d'éditeur pour votre information), vous pouvez traiter des listes d'instructions beaucoup plus grand. A cet effet, l'éditeur LIST dresse des fichiers temporaires (A0.TM0, A0.TM1, A0.TM2). Ces fichiers nécessitent un espace correspondant sur le porteur externe de données (disque dur en général). Si cet espace n'est pas de grandeur suffisante, il y aura un message d'erreur. Après *sauvegarder* ou *valider*, ces fichiers temporaires seront effacés. En des cas particuliers (p. ex. absence de secteur pendant l'édition), ces fichiers restent existants; ils peuvent être effacés en tout temps.

Quatre fichiers sont définis pendant le préréglage:

- le fichier source LIST que vous voulez éditer (A0.SEQ);
- le fichier intérmédiaire qui est généré lors de la mémorisation via touche de validation et qui contient la liste d'instructions convertie en code intérmédiaire (A1.SEQ);
- le fichier symbolique qui contient une liste d'assignations (Z0.INI) et
- le fichier programme dans lequel le programme STEP 5 sera écrit après la conversion (ST.S5D).

Ces quatre fichiers sont inscrits automatiquement avec les mêmes noms qu'on peut modifiés le cas échéant. Cependant, le fichier source LIST et le fichier intérmédiaire ont toujours le même nom.

Lors de la sélection de fonctions le paquet éditeur LIST/compilateur par lots vous offre comme fonctions:



EDITEUR pour créer et traiter le fichier source LIST;
(opérations v. chap. 3.3)

COMPILER pour convertir et reconvertir;
(opérations v. chap. 3.4)

Description

- Avec *abandon* (touche d'abandon), vous rejetez le fichier entier édité dans la mémoire de travail et quittez l'éditeur.

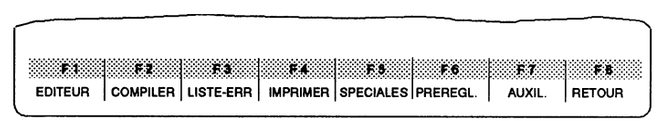
Remarque: Bien que le tampon d'éditeur n'ait qu'une grandeur spécifiée et finie (cette grandeur apparaît dans le masque d'éditeur pour votre information), vous pouvez traiter des listes d'instructions beaucoup plus grand. A cet effet, l'éditeur LIST dresse des fichiers temporaires (A0.TM0, A0.TM1, A0.TM2). Ces fichiers nécessitent un espace correspondant sur le porteur externe de données (disque dur en général). Si cet espace n'est pas de grandeur suffisante, il y aura un message d'erreur. Après *sauvegarder* ou *valider*, ces fichiers temporaires seront effacés. En des cas particuliers (p. ex. absence de secteur pendant l'édition), ces fichiers restent existants; ils peuvent être effacés en tout temps.

Quatre fichiers sont définis pendant le préréglage:

- le fichier source LIST que vous voulez éditer (A0.SEQ);
- le fichier intérmédiaire qui est généré lors de la mémorisation via touche de validation et qui contient la liste d'instructions convertie en code intérmédiaire (A1.SEQ);
- le fichier symbolique qui contient une liste d'assignations (Z0.INI) et
- le fichier programme dans lequel le programme STEP 5 sera écrit après la conversion (ST.S5D).

Ces quatre fichiers sont inscrits automatiquement avec les mêmes noms qu'on peut modifiés le cas échéant. Cependant, le fichier source LIST et le fichier intérmédiaire ont toujours le même nom.

Lors de la sélection de fonctions le paquet éditeur LIST/compilateur par lots vous offre comme fonctions:



EDITEUR pour créer et traiter le fichier source LIST;
(opérations v. chap. 3.3)

COMPILER pour convertir et reconvertir;
(opérations v. chap. 3.4)

LISTE ERR la liste d'erreurs des contrôles;
(opérations v. chap. 3.5)

IMPRIMER pour imprimer le fichier source LIST;
(opérations v. chap. 3.6)

SPECIALES des fonctions spéciales pour générer des fichiers intermédiaires et des fichiers source LIST;
(opérations v. chap. 3.7)

PREREGL. pour modifier le préréglage;

AUXIL. des fonctions auxiliaires pour la gestion de blocs dans le fichier programme préréglé et

RETOUR pour quitter le paquet éditeur LIST/compilateur par lots.

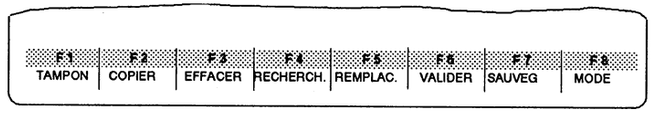
Ces fonctions seront brèvement décrites dans les paragraphes suivants.

2.2.2 Fonctions d'éditeur

L'éditeur LIST affiche un masque d'éditeur sur l'écran qui est préparé pour une liste d'instructions. Le masque d'éditeur consiste dans:

- une ligne d'en-tête avec le nom du fichier source LIST,
- les champs d'introduction en forme de colonnes pour ADRESSE, INSTRUCTION, MNEMONIQUE D'OPERANDE et COMMENTAIRE D'INSTRUCTION,
- le menu avec les fonctions d'éditeur.

Il vous offre plusieurs fonctions avec lesquelles vous pouvez éditer votre programme de façon confortable. Elles sont comparable à ceux de l'éditeur symbolique:



Description

LISTE ERR la liste d'erreurs des contrôles;
(opérations v. chap. 3.5)

IMPRIMER pour imprimer le fichier source LIST;
(opérations v. chap. 3.6)

SPECIALES des fonctions spéciales pour générer des fichiers intermédiaires et des fichiers source LIST;
(opérations v. chap. 3.7)

PREREGL. pour modifier le préréglage;

AUXIL. des fonctions auxiliaires pour la gestion de blocs dans le fichier programme préréglé et

RETOUR pour quitter le paquet éditeur LIST/compilateur par lots.

Ces fonctions seront brèvement décrites dans les paragraphes suivants.

2

2.2.2 Fonctions d'éditeur

L'éditeur LIST affiche un masque d'éditeur sur l'écran qui est préparé pour une liste d'instructions. Le masque d'éditeur consiste dans:

- une ligne d'en-tête avec le nom du fichier source LIST,
- les champs d'introduction en forme de colonnes pour ADRESSE, INSTRUCTION, MNEMONIQUE D'OPERANDE et COMMENTAIRE D'INSTRUCTION,
- le menu avec les fonctions d'éditeur.

Il vous offre plusieurs fonctions avec lesquelles vous pouvez éditer votre programme de façon confortable. Elles sont comparable à ceux de l'éditeur symbolique:

F1 TAMPON
F2 COPIER
F3 EFFACER
F4 RECHERCH.
F5 REMPLAC.
F6 VALIDER
F7 SAUVEG
F8 MODE

C79000-B8577-C677-02

2 - 9

Via les fonctions **TAMPON** et **COPIER**, il est possible d'écrire n'importe quelles chaînes de caractères et blocs de texte dans un tampon ou sur un fichier séquentiel ("transférer") et de les copier à tout endroit. Ainsi vous pouvez p. ex. déplacer des segments ou les inclure toujours à nouveau dans le programme. En outre, vous pouvez lire d'autres fichiers source LIST ou des blocs individuels avec la fonction copier. Les fonctions tampon, copier et effacer peuvent être combinées avec des facteurs de répétition.

Les fonctions **RECHERCHE** et **REMPLACER** facilitent les corrections de votre programme:

Vous pouvez trouver n'importe quoi dans votre fichier avec une très grande vitesse. Vous pouvez modifier des chaînes de caractères individuelles, comme p. ex. des mnémoniques ou des opérandes, à l'aide d'une seule fonction dans toute votre liste d'instructions. De plus, vous avez le choix entre le **mode** insérer et le **mode** écrasement.

SAUVEGARDER, comme déjà mentionné, vous donne la possibilité de mémoriser votre fichier sans quitter l'éditeur. Lors de chaque suspension passagère de la séance de travail une telle mémorisation est utile. Avec **VALIDER**, vous mémorisez votre fichier, le conversez dans le fichier intermédiaire et quittez l'éditeur.

Chapitre 3.3.5 vous montre l'lutilisation pratique de ces fonctions d'éditeur.

Description

Via les fonctions **TAMPON** et **COPIER**, il est possible d'écrire n'importe quelles chaînes de caractères et blocs de texte dans un tampon ou sur un fichier séquentiel ("transférer") et de les copier à tout endroit. Ainsi vous pouvez p. ex. déplacer des segments ou les inclure toujours à nouveau dans le programme. En outre, vous pouvez lire d'autres fichiers source LIST ou des blocs individuels avec la fonction copier. Les fonctions tampon, copier et effacer peuvent être combinées avec des facteurs de répétition.

Les fonctions **RECHERCHE** et **REMPLACER** facilitent les corrections de votre programme:

Vous pouvez trouver n'importe quoi dans votre fichier avec une très grande vitesse. Vous pouvez modifier des chaînes de caractères individuelles, comme p. ex. des mnémoniques ou des opérandes, à l'aide d'une seule fonction dans toute votre liste d'instructions. De plus, vous avez le choix entre le **mode** insérer et le **mode** écrasement.

SAUVEGARDER, comme déjà mentionné, vous donne la possibilité de mémoriser votre fichier sans quitter l'éditeur. Lors de chaque suspension passagère de la séance de travail une telle mémorisation est utile. Avec **VALIDER**, vous mémorisez votre fichier, le conversez dans le fichier intermédiaire et quittez l'éditeur.

Chapitre 3.3.5 vous montre l'lutilisation pratique de ces fonctions d'éditeur.

2.2.3 Caractères de commande

Pour la conversion du fichier source LIST dans un fichier programme STEP 5, certains caractères de commande et conventions d'écriture doivent être observés en éditant. Dans les chapitres 3.3.2 et 3.3.3, vous trouvez une liste complète de toutes les instructions d'édition. Ici, seulement ces caractères de commande seront décrits qui sont nouveau par rapport au paquet CONT-LOG-LIST.

#TY marque le type d'AG. Après cet caractère de commande vous pouvez ici nommer l'AG dans lequel vous voulez exécuter le programme. Cette indication doit concorder avec l'introduction dans le champ de la catégorie de langage de la préréglage. Le compilateur par lots *contrôle* lors de la *conversion* dans le fichier programme si les opérations éditées dans le stock d'opérations de l'AG inscrit sont autorisées. Dans le fichier source LIST, le type d'AG peut être situé au début du fichier et aux extrémités de blocs.

Description

2.2.3 Caractères de commande

Pour la conversion du fichier source LIST dans un fichier programme STEP 5, certains caractères de commande et conventions d'écriture doivent être observés en éditant. Dans les chapitres 3.3.2 et 3.3.3, vous trouvez une liste complète de toutes les instructions d'édition. Ici, seulement ces caractères de commande seront décrits qui sont nouveau par rapport au paquet CONT-LOG-LIST.

#TY marque le type d'AG. Après cet caractère de commande vous pouvez ici nommer l'AG dans lequel vous voulez exécuter le programme. Cette indication doit concorder avec l'introduction dans le champ de la catégorie de langage de la préréglage. Le compilateur par lots *contrôle* lors de la *conversion* dans le fichier programme si les opérations éditées dans le stock d'opérations de l'AG inscrit sont autorisées. Dans le fichier source LIST, le type d'AG peut être situé au début du fichier et aux extrémités de blocs.

2

C79000-B8577-C677-02

2 - 11

Les désignations suivantes pour la catégorie de langage sont autorisées:

Type d'AG	Processeur	Catégorie de langage
AG 100 U	CPU 100 CPU 102 CPU 103	CPU100 CPU 102 CPU 103
AG 101 U		AG101U
AG 110 S		AG110S
AG 115 U	CPU 941 CPU 942, 943 CPU944	CPU 941 CPU942 CPU944
AG 130 WB		AG130W
AG 135 U	CPU 921 CPU 922 CPU 928 CPU 928B	CPU 921 CPU 922 CPU 928 CPU 928B
AG 135 W		AG135W
AG 135 WB		AG135B
AG 90		AG 90
AG 95		AG 95
AG 150 A/K		AG150A
AG 150 S/U		AG150S
AG 155 U	CPU 946/947	AG155U
Processeur E/S	IP 257	IP257

La *conversion* dans le fichier programme n'est réalisée que si la désignation de l'AG dans le préréglage (champ "CATEGORIE LANGAGE") concorde avec les introductions dans les lignes #TY du fichier source LIST. En cas de disparité, la *conversion* sera abandonnée dans la ligne #TY. Si vous introduisez "NON" pour la catégorie de langage pendant le préréglage, la *conversion* sera réalisée sans *contrôle spécifique de l'AG.*

Description

Les désignations suivantes pour la catégorie de langage sont autorisées:

Type d'AG	Processeur	Catégorie de langage
AG 100 U	CPU 100 CPU 102 CPU 103	CPU100 CPU 102 CPU 103
AG 101 U		AG101U
AG 110 S		AG110S
AG 115 U	CPU 941 CPU 942, 943 CPU944	CPU 941 CPU942 CPU944
AG 130 WB		AG130W
AG 135 U	CPU 921 CPU 922 CPU 928 CPU 928B	CPU 921 CPU 922 CPU 928 CPU 928B
AG 135 W		AG135W
AG 135 WB		AG135B
AG 90		AG 90
AG 95		AG 95
AG 150 A/K		AG150A
AG 150 S/U		AG150S
AG 155 U	CPU 946/947	AG155U
Processeur E/S	IP 257	IP257

La *conversion* dans le fichier programme n'est réalisée que si la désignation de l'AG dans le préréglage (champ "CATEGORIE LANGAGE") concorde avec les introductions dans les lignes #TY du fichier source LIST. En cas de disparité, la *conversion* sera abandonnée dans la ligne #TY. Si vous introduisez "NON" pour la catégorie de langage pendant le préréglage, la *conversion* sera réalisée sans *contrôle spécifique de l'AG.*

l'instruction Include

L'instruction Include #I rend possible l'inclusion de n'importe quel fichier et peut être situé dans le fichier source LIST au début du fichier et aux extrémités de blocs, c.-à-d. apres BE. La désignation du fichier suit après #I. Il est important que le lecteur soit aussi nommé (exemple: #I B:TEST).

L'inclusion du fichier se fait au niveau de code intermédiaire, c.-à-d., le fichier à inclure doit être existant en forme de fichier intermédiaire. Si l'introduction du lecteur manque, le lecteur sera accédé qui est spécifié pour le fichier intermédiaire ou du fichier source LIST dans le préréglage. Si vous avez mémorisé chaque modification de votre liste d'instructions dans le fichier source avec valider, la conversion du fichier intermédiaire suffit. Autrement, vous devez déclencher la conversion du fichier source LIST qui crée automatiquement un fichier intermédiaire actuel.

En cas de blocs avec désignation identique dans les deux fichiers, vous devez les renommer avant la *conversion*. Vous pouvez éviter ce problème, si vous nommez les blocs dans vos fichiers source LIST sous forme symbolique.

Pour cela, il est absolument nécessaire que le fichier symbolique soit existant, car le type de bloc et le numéro sont indispensables pour la conversion.

L'instruction Include est particulièrement appropriée pour des bibliothèques spécifiques de l'utilisateur: A l'aide des fichiers Include, des programmes standard peuvent être modifiés de façon spécifique de la tâche. Lors d'une modification on ne change que les fichiers Include p. ex. Après la modification on employe la version actuelle de tous les programmes pour la génération du fichier programme.

2

2.2.4 Conversion

Génération d'un fichier programme

A l'aide du compilateur par lots, vous pouvez maintenant convertir tous les blocs, un groupe de blocs ou un bloc individuel du fichier intermédiaire dans le fichier programme. Pour cette raison, il est important d'avoir toujours un fichier intermédiaire actualisé. Donc, mémorisez chaque modification de votre liste d'instructions dans le fichier source avec *valider*.

Si vous avez programmé votre fichier source LIST sous forme symbolique, le fichier symbolique préréglé sera relié au fichier intermédiaire lors de la *conversion* dans le fichier programme. Un

Description

l'instruction Include

L'instruction Include #I rend possible l'inclusion de n'importe quel fichier et peut être situé dans le fichier source LIST au début du fichier et aux extrémités de blocs, c.-à-d. apres BE. La désignation du fichier suit après #I. Il est important que le lecteur soit aussi nommé (exemple: #I B:TEST).

L'inclusion du fichier se fait au niveau de code intermédiaire, c.-à-d., le fichier à inclure doit être existant en forme de fichier intermédiaire. Si l'introduction du lecteur manque, le lecteur sera accédé qui est spécifié pour le fichier intermédiaire ou du fichier source LIST dans le préréglage. Si vous avez mémorisé chaque modification de votre liste d'instructions dans le fichier source avec valider, la conversion du fichier intermédiaire suffit. Autrement, vous devez déclencher la conversion du fichier source LIST qui crée automatiquement un fichier intermédiaire actuel.

En cas de blocs avec désignation identique dans les deux fichiers, vous devez les renommer avant la *conversion*. Vous pouvez éviter ce problème, si vous nommez les blocs dans vos fichiers source LIST sous forme symbolique.

Pour cela, il est absolument nécessaire que le fichier symbolique soit existant, car le type de bloc et le numéro sont indispensables pour la conversion.

L'instruction Include est particulièrement appropriée pour des bibliothèques spécifiques de l'utilisateur: A l'aide des fichiers Include, des programmes standard peuvent être modifiés de façon spécifique de la tâche. Lors d'une modification on ne change que les fichiers Include p. ex. Après la modification on employe la version actuelle de tous les programmes pour la génération du fichier programme.

2

2.2.4 Conversion

Génération d'un fichier programme

A l'aide du compilateur par lots, vous pouvez maintenant convertir tous les blocs, un groupe de blocs ou un bloc individuel du fichier intermédiaire dans le fichier programme. Pour cette raison, il est important d'avoir toujours un fichier intermédiaire actualisé. Donc, mémorisez chaque modification de votre liste d'instructions dans le fichier source avec *valider*.

Si vous avez programmé votre fichier source LIST sous forme symbolique, le fichier symbolique préréglé sera relié au fichier intermédiaire lors de la *conversion* dans le fichier programme. Un

fichier symbolique ne sera pas dressé par l'éditeur LIST mais doit être créé avec l'éditeur symbolique. Si vous inclurez un autre fichier via l'instruction Include #I, prenez garde que les symboles pour ce fichier se trouvent dans le fichier symbolique préréglé.

Dans les lignes de commande du **compilateur**, vous pouvez introduire si un code de machine doit être généré ou si seulement un test d'erreurs doit être réalisé. Vous pouvez également indiquer, si vous voulez être demandé comme contrôle lors d'un écrasement de blocs. De même, vous pouvez sélectionner simultanément une impression du programme converti.

Reconversion d'un fichier programme

Il n'y a ni fichiers source LIST ni fichiers intermédiaires pour les blocs étant créés avec le paquet CONT-LOG-LIST. L'éditeur LIST/compilateur par lots peut dresser ces fichiers en utilisant un fichier programme.

En *reconvertissant* un bloc, un groupe de blocs ou tous les blocs du fichier programme, vous pouvez d'abord créer le fichier intermédiaire ou directement le fichier source LIST séquentiel, et pouvez le modifier et le compléter. Celui-ci sera ensuite transformé dans un fichier source LIST séquentiel que vous pouvez modifier et supplémenter.

Au cours de la *reconversion*, vous déterminez l'apparence de votre fichier source LIST "nouveau": Les instructions seront représentées ou seulement par symboles ou seulement par paramètres absolus ou par tous les deux. En outre, le caractère de commande pour l'identification de la catégorie de langage sera inscrit dans le fichier intermédiaire, si une identification de la catégorie de langage (type d'AG) est inscrite dans le préréglage.

L'éditeur LIST peut traiter des fichiers à 65535 lignes au max. Cependant, le nombre de lignes du fichier source LIST dépend non seulement du nombre d'instructions STEP 5 mais aussi des instructions spéciales, des lignes de commentaire etc. Si le fichier programme que vous voulez *reconvertir* est plus grand, il faut répartir les blocs sur plusieurs fichiers intermédiaires.

Les blocs fonctionnels (FB) standard, aussi bien que les blocs Graph 5 et les blocs en assembleur ne sont pas reconvertis.

Contrôles lors de la conversion

Lors de la *conversion/reconversion* du code intermédiaire, l'admissibilité de l'instruction qui se produit est vérifiée. De même,

Description

fichier symbolique ne sera pas dressé par l'éditeur LIST mais doit être créé avec l'éditeur symbolique. Si vous inclurez un autre fichier via l'instruction Include #I, prenez garde que les symboles pour ce fichier se trouvent dans le fichier symbolique préréglé.

Dans les lignes de commande du **compilateur**, vous pouvez introduire si un code de machine doit être généré ou si seulement un test d'erreurs doit être réalisé. Vous pouvez également indiquer, si vous voulez être demandé comme contrôle lors d'un écrasement de blocs. De même, vous pouvez sélectionner simultanément une impression du programme converti.

Reconversion d'un fichier programme

Il n'y a ni fichiers source LIST ni fichiers intermédiaires pour les blocs étant créés avec le paquet CONT-LOG-LIST. L'éditeur LIST/compilateur par lots peut dresser ces fichiers en utilisant un fichier programme.

En *reconvertissant* un bloc, un groupe de blocs ou tous les blocs du fichier programme, vous pouvez d'abord créer le fichier intermédiaire ou directement le fichier source LIST séquentiel, et pouvez le modifier et le compléter. Celui-ci sera ensuite transformé dans un fichier source LIST séquentiel que vous pouvez modifier et supplémenter.

Au cours de la *reconversion*, vous déterminez l'apparence de votre fichier source LIST "nouveau": Les instructions seront représentées ou seulement par symboles ou seulement par paramètres absolus ou par tous les deux. En outre, le caractère de commande pour l'identification de la catégorie de langage sera inscrit dans le fichier intermédiaire, si une identification de la catégorie de langage (type d'AG) est inscrite dans le préréglage.

L'éditeur LIST peut traiter des fichiers à 65535 lignes au max. Cependant, le nombre de lignes du fichier source LIST dépend non seulement du nombre d'instructions STEP 5 mais aussi des instructions spéciales, des lignes de commentaire etc. Si le fichier programme que vous voulez *reconvertir* est plus grand, il faut répartir les blocs sur plusieurs fichiers intermédiaires.

Les blocs fonctionnels (FB) standard, aussi bien que les blocs Graph 5 et les blocs en assembleur ne sont pas reconvertis.

Contrôles lors de la conversion

Lors de la *conversion/reconversion* du code intermédiaire, l'admissibilité de l'instruction qui se produit est vérifiée. De même,

2 - 14 C79000-B8577-C677-02

l'admissibilité d'une instruction est vérifiée quant au type de bloc. La catégorie de langage sera vérifiée, si vous avez indiqué un type d'AG dans le préréglage. En cas de programmation symbolique, les allocations seront vérifiées en combinaison avec les opérandes. Si vous avez indiqué non seulement un opérande absolu mais aussi un opérande symbolique, la concordence avec le fichier symbolique sera vérifiée. En cas de disparité des paramètres, le paramètre absolu du fichier symbolique sera utilisé qui est alloué au symbole, et un avertissement sera rangé dans la liste d'erreurs. En cas de programmation absolu, le fichier symbolique ne sera pas accédé. Les erreurs qui sont constatées au cours de ces contrôles seront indiquées dans la liste d'erreurs.

2.2.5 Imprimer

Vous pouvez produire un listing du fichier source LIST via la fonction **imprimer** (dans la sélection des fonctions). Cependant, cette fonction ne sortit sur l'imprimante que le fichier source LIST préréglé.

Dans les lignes de commande, la fonction **compiler** vous offre une impression sur l'imprimante lors de la *conversion*. Ainsi, vous pouvez garder le résultat de chaque conversion et aussi celui de la contrôle.

L'éditeur LIST/compilateur par lots vous offre les formats d'impression qui en général font partie du paquet de base STEP 5. Vous choisissez entre impression standard, écriture normale, écriture comprimée et écriture super-comprimée. La cartouche doit avoir une largeur de 132 caractères en format A3 (fichier F2.INI), 80 caractères en format A4 (fichier F1.INI). Lors de la sortie en écriture comprimée, le commentaire d'opérande apparaît en plus. Lors de la sortie en écriture super-comprimée, le commentaire symbolique apparaît en plus.

Description

l'admissibilité d'une instruction est vérifiée quant au type de bloc. La catégorie de langage sera vérifiée, si vous avez indiqué un type d'AG dans le préréglage. En cas de programmation symbolique, les allocations seront vérifiées en combinaison avec les opérandes. Si vous avez indiqué non seulement un opérande absolu mais aussi un opérande symbolique, la concordence avec le fichier symbolique sera vérifiée. En cas de disparité des paramètres, le paramètre absolu du fichier symbolique sera utilisé qui est alloué au symbole, et un avertissement sera rangé dans la liste d'erreurs. En cas de programmation absolu, le fichier symbolique ne sera pas accédé. Les erreurs qui sont constatées au cours de ces contrôles seront indiquées dans la liste d'erreurs.

2

2.2.5 Imprimer

Vous pouvez produire un listing du fichier source LIST via la fonction **imprimer** (dans la sélection des fonctions). Cependant, cette fonction ne sortit sur l'imprimante que le fichier source LIST préréglé.

Dans les lignes de commande, la fonction **compiler** vous offre une impression sur l'imprimante lors de la *conversion*. Ainsi, vous pouvez garder le résultat de chaque conversion et aussi celui de la contrôle.

L'éditeur LIST/compilateur par lots vous offre les formats d'impression qui en général font partie du paquet de base STEP 5. Vous choisissez entre impression standard, écriture normale, écriture comprimée et écriture super-comprimée. La cartouche doit avoir une largeur de 132 caractères en format A3 (fichier F2.INI), 80 caractères en format A4 (fichier F1.INI). Lors de la sortie en écriture comprimée, le commentaire d'opérande apparaît en plus. Lors de la sortie en écriture super-comprimée, le commentaire symbolique apparaît en plus.

C79000-B8577-C677-02

2 - 15

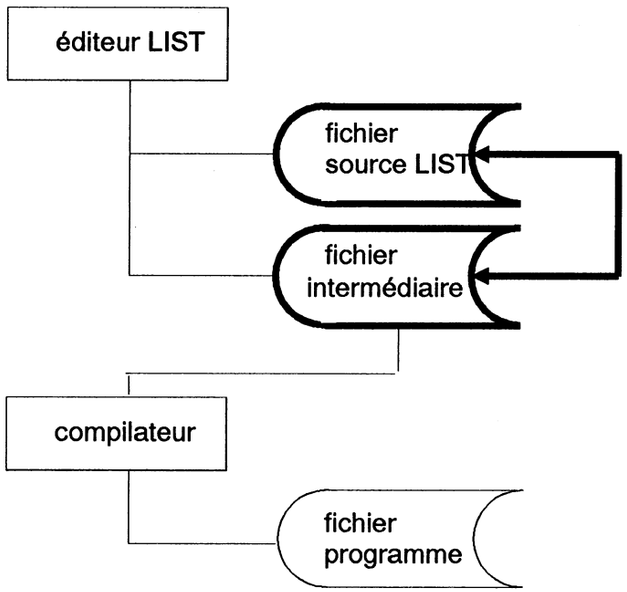
2.3 Le fichier intermédiaire A1.SEQ

Le fichier intermédiaire forme le fichier central dans le paquet éditeur LIST/compilateur par lots. Etant indépendant et ne pas encore du code machine MC5, il est la base pour toutes les conversions. En l'utilisant, on peut toujours créer

- des fichiers programme STEP 5,
- des fichiers source LIST,
- des versions d'un programme spécifiques de l'installation,
- des versions d'un programme en langue étrangère.

Pour cette raison, il est indispensable de sauvegarder toujours le fichier intermédiaire, et il est recommandable de quitter un fichier source LIST toujours avec *valider* pour que le fichier intermédiaire soit actuel.

2.3.1 Relations entre fichier source LIST et fichier intermédiaire



Description

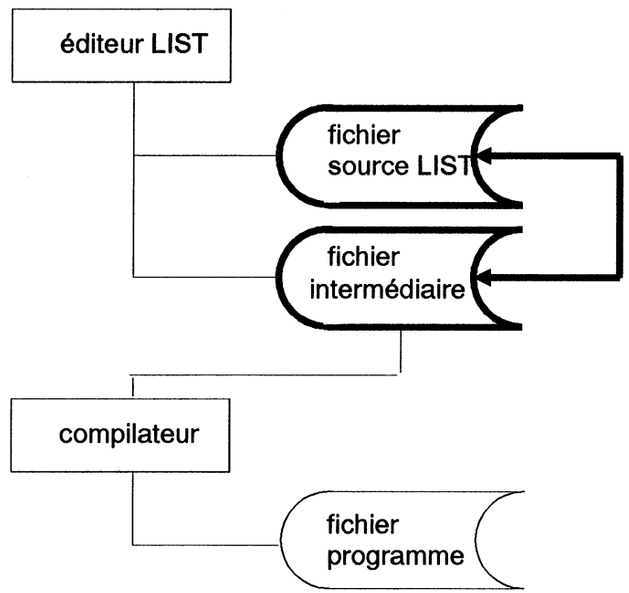
2.3 Le fichier intermédiaire A1.SEQ

Le fichier intermédiaire forme le fichier central dans le paquet éditeur LIST/compilateur par lots. Etant indépendant et ne pas encore du code machine MC5, il est la base pour toutes les conversions. En l'utilisant, on peut toujours créer

- des fichiers programme STEP 5,
- des fichiers source LIST,
- des versions d'un programme spécifiques de l'installation,
- des versions d'un programme en langue étrangère.

Pour cette raison, il est indispensable de sauvegarder toujours le fichier intermédiaire, et il est recommandable de quitter un fichier source LIST toujours avec *valider* pour que le fichier intermédiaire soit actuel.

2.3.1 Relations entre fichier source LIST et fichier intermédiaire



Le fichier source LIST et le fichier intermédiaire sont liés très étroitement: Ils ont le même nom et leurs identifications de fichier ne se distinguent que par un caractère (A0.SEQ, A1.SEQ). Le nom d'un fichier intermédiaire ne peut jamais être modifié sans que le fichier source LIST soit aussi concerné. Cependant, les deux fichiers peuvent se trouver sur des lecteurs différents. L'égalité des noms garantit que vos programmes édités seront convertis dans le fichier intermédiaire correspondant lors de *valider*.

reconversion

Dans le cas d'une *reconversion*, le fichier source LIST lié par son nom sera également mémorisé. Il faut considérer cela, si le fichier source LIST et le fichier intermédiaire n'ont pas le même niveau d'actualité ou si la "vieille" liste d'instructions ne doit pas être écrasée p. ex. si votre premier fichier source LIST contient des commentaires supplémentaires. Ceux-ci ne seront pas transférés dans le fichier programme et seront perdus après la reconversion.

A cause de cela, les conditions suivantes se produisent pour la génération d'un fichier source LIST en utilisant un fichier intermédiaire:

- Si aucun fichier source LIST n'existe, il sera créé automatiquement quand vous appelez *éditeur*. Il reçoit le nom qui a été déterminé en cours de préréglage.
- Si un fichier source LIST du même nom existe, le fichier intermédiaire doit être transféré au fichier source LIST de façon explicite. Ça se fait via la fonction spéciale **INT>SEQ** (voir plus bas). Au cours de ce procès, le fichier source LIST ancien sera écrasé.

Si un fichier source LIST "ancien" doit être gardé, introduisez, en cours de préréglage, un nom pour le fichier source LIST nouveau avant la *reconversion* d'un fichier programme. Sous ce nom, il sera reconverti ensuite et mémorisé dans le fichier source LIST.

Pour cette raison, il est important de vérifier avant chaque *conversion*, mais particulièrement lors d'une *reconversion*, si les données correctes ont été inscrit dans le préréglage.

Remarque:
les fonctions SEQMC5 et MC5SEQ génèrent automatiquement un fichier intermédiaire actuel (voir parag. 2.3.2)

Description

Le fichier source LIST et le fichier intermédiaire sont liés très étroitement: Ils ont le même nom et leurs identifications de fichier ne se distinguent que par un caractère (A0.SEQ, A1.SEQ). Le nom d'un fichier intermédiaire ne peut jamais être modifié sans que le fichier source LIST soit aussi concerné. Cependant, les deux fichiers peuvent se trouver sur des lecteurs différents. L'égalité des noms garantit que vos programmes édités seront convertis dans le fichier intermédiaire correspondant lors de *valider*.

reconversion

Dans le cas d'une *reconversion*, le fichier source LIST lié par son nom sera également mémorisé. Il faut considérer cela, si le fichier source LIST et le fichier intermédiaire n'ont pas le même niveau d'actualité ou si la "vieille" liste d'instructions ne doit pas être écrasée p. ex. si votre premier fichier source LIST contient des commentaires supplémentaires. Ceux-ci ne seront pas transférés dans le fichier programme et seront perdus après la reconversion.

A cause de cela, les conditions suivantes se produisent pour la génération d'un fichier source LIST en utilisant un fichier intermédiaire:

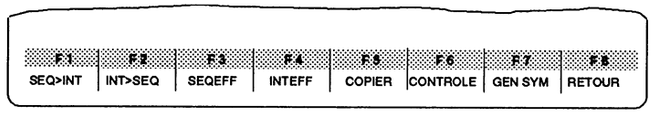
- Si aucun fichier source LIST n'existe, il sera créé automatiquement quand vous appelez *éditeur*. Il reçoit le nom qui a été déterminé en cours de préréglage.
- Si un fichier source LIST du même nom existe, le fichier intermédiaire doit être transféré au fichier source LIST de façon explicite. Ça se fait via la fonction spéciale **INT>SEQ** (voir plus bas). Au cours de ce procès, le fichier source LIST ancien sera écrasé.

Si un fichier source LIST "ancien" doit être gardé, introduisez, en cours de préréglage, un nom pour le fichier source LIST nouveau avant la *reconversion* d'un fichier programme. Sous ce nom, il sera reconverti ensuite et mémorisé dans le fichier source LIST.

Pour cette raison, il est important de vérifier avant chaque *conversion*, mais particulièrement lors d'une *reconversion*, si les données correctes ont été inscrit dans le préréglage.

Remarque:
les fonctions SEQMC5 et MC5SEQ génèrent automatiquement un fichier intermédiaire actuel (voir parag. 2.3.2)

2.3.2
Fonctions spéciales



Les fonctions spéciales offrent des transformations diverses pour générer des fichiers source LIST et des fichiers intermédiaires. Ceci peut être nécessaire car les fichiers source LIST et les fichiers intermédiaires ont toujours le même nom mais quelquefois des versions différentes (voir plus haut, 2.3.1).

Chapitre 3.7 vous montre comment il faut utiliser ces fonctions dans la pratique.

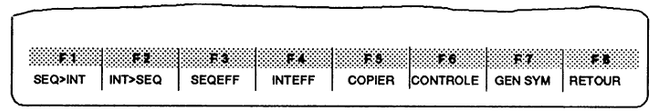
SEQ>INT transforme un fichier séquentiel en un fichier intermédiaire. Vous utilisez cette fonction, si, p. ex., votre fichier source LIST a été écrit avec un éditeur non Siemens et doit être converti dans un fichier programme STEP 5 ou si le fichier intermédiaire n'existe plus.

INT>SEQ transforme un fichier intermédiaire en un fichier séquentiel. Si vous avez, p. ex., reconverti un fichier programme et voulez le traiter avec l'éditeur LIST, le fichier intermédiaire doit être transformé en un fichier séquentiel. Cette fonction est particulièrement important, si une version ancienne du fichier source LIST existe. Si vous voulez sortir le fichier source LIST avec un éditeur LIST de langue étrangère (voir plus bas, 2.3.4), cette fonction est aussi très utile.

Via EFFSEQ, vous pouvez effacer des fichiers séquentiels, si vous voulez les générer à nouveau, p. ex., lors des conversions. Ensuite, quand vous *éditez* les fichiers convertis, les fichiers séquentiels seront dressés automatiquement de l'éditeur.

Description

2.3.2
Fonctions spéciales



Les fonctions spéciales offrent des transformations diverses pour générer des fichiers source LIST et des fichiers intermédiaires. Ceci peut être nécessaire car les fichiers source LIST et les fichiers intermédiaires ont toujours le même nom mais quelquefois des versions différentes (voir plus haut, 2.3.1).

Chapitre 3.7 vous montre comment il faut utiliser ces fonctions dans la pratique.

SEQ>INT transforme un fichier séquentiel en un fichier intermédiaire. Vous utilisez cette fonction, si, p. ex., votre fichier source LIST a été écrit avec un éditeur non Siemens et doit être converti dans un fichier programme STEP 5 ou si le fichier intermédiaire n'existe plus.

INT>SEQ transforme un fichier intermédiaire en un fichier séquentiel. Si vous avez, p. ex., reconverti un fichier programme et voulez le traiter avec l'éditeur LIST, le fichier intermédiaire doit être transformé en un fichier séquentiel. Cette fonction est particulièrement important, si une version ancienne du fichier source LIST existe. Si vous voulez sortir le fichier source LIST avec un éditeur LIST de langue étrangère (voir plus bas, 2.3.4), cette fonction est aussi très utile.

Via EFFSEQ, vous pouvez effacer des fichiers séquentiels, si vous voulez les générer à nouveau, p. ex., lors des conversions. Ensuite, quand vous *éditez* les fichiers convertis, les fichiers séquentiels seront dressés automatiquement de l'éditeur.

EFFINT efface les fichiers intermédiaires (p. ex. des versions anciennes). Ils seront réétablis ou avec la fonction **SEQ>INT** d'un fichier source actualisé ou lors de la *validation* d'une source éditée.

COPIER Si vous voulez copier le fichier source LIST et le fichier intermédiaire sur d'autres lecteurs pour avoir une sauvegarde, employez la fonction **COPIER**. Une renommation des fichiers n'est pourtant possible qu'au sein du programme utilitaire AUX. FICH.

La CONTROLE offre la possibilité de vérifier plus tard l'autorisation des instructions pour le type d'AG préréglé quant aux blocs convertis d'un fichier programme mais aussi aux blocs du paquet CONT-LOG-LIST.

SYM-GEN Cette fonction crée du fichier source LIST un fichier source symbolique qui contient tous les symboles et paramètres absolus utilisés. Vous pouvez élargir ce fichier en affectations et en commentaires via l'éditeur symbolique, sans devoir introduire de nouveau les symboles et paramètres absolus déjà contenus dans le fichier source LIST.

2.3.3 Programmes standard

La possibilité de créer une liste d'instructions uniquement avec des symboles et le fait que le fichier intermédiaire est indépendant du langage vous permettent de créer des programmes standard. Des blocs et modules testés peuvent être rangés dans des bibliothèques et après reliés en programmes individuels de façon spécifique de l'installation via l'instruction Include. Ainsi, vous ne devez que relier vos programmes combinés à nouveau avec une liste d'assignations spéciale, et vous avez des programmes STEP 5 spécifiques de l'installation pour les problèmes de commande individuels.

Description

EFFINT efface les fichiers intermédiaires (p. ex. des versions anciennes). Ils seront réétablis ou avec la fonction SEQ>INT d'un fichier source actualisé ou lors de la *validation* d'une source éditée.

COPIER Si vous voulez copier le fichier source LIST et le fichier intermédiaire sur d'autres lecteurs pour avoir une sauvegarde, employez la fonction COPIER. Une renommation des fichiers n'est pourtant possible qu'au sein du programme utilitaire AUX. FICH.

La CONTROLE offre la possibilité de vérifier plus tard l'autorisation des instructions pour le type d'AG préréglé quant aux blocs convertis d'un fichier programme mais aussi aux blocs du paquet CONT-LOG-LIST.

SYM-GEN Cette fonction crée du fichier source LIST un fichier source symbolique qui contient tous les symboles et paramètres absolus utilisés. Vous pouvez élargir ce fichier en affectations et en commentaires via l'éditeur symbolique, sans devoir introduire de nouveau les symboles et paramètres absolus déjà contenus dans le fichier source LIST.

2

2.3.3 Programmes standard

La possibilité de créer une liste d'instructions uniquement avec des symboles et le fait que le fichier intermédiaire est indépendant du langage vous permettent de créer des programmes standard. Des blocs et modules testés peuvent être rangés dans des bibliothèques et après reliés en programmes individuels de façon spécifique de l'installation via l'instruction Include. Ainsi, vous ne devez que relier vos programmes combinés à nouveau avec une liste d'assignations spéciale, et vous avez des programmes STEP 5 spécifiques de l'installation pour les problèmes de commande individuels.

C79000-B8577-C677-02

2 - 19

2.3.4 Versions d'un programme STEP 5 en langue étrangère

A l'aide du compilateur par lots, vous pouvez également générer des versions d'un programme STEP 5 en langue étrangère, si vous avez programmé votre programme sous forme absolue ou si vous disposez d'un fichier intermédiaire reconverti qui contient des paramètres absolus: Avec la version de logiciel anglaise et française de ce paquet, vous créez des sources LIST anglaises et françaises.

A cet effet, vous allez au programme utilitaire **AUX.-FICH, TRF.-FICH** et copiez votre fichier source LIST et fichier inter médiaire avec un nom nouveau. Vous inscrivez ce nom nouveau dans le préréglage de la version de logiciel anglaise/française. Vous transférez le fichier intermédiaire qui est indépendant du langage au fichier source séquentiel à l'aide de la fonction spéciale **INT>SEQ** (voir plus haut). Quand vous actionnez **EDITEUR,** le fichier source séquentiel est ensuite sorti avec des instructions STEP 5 anglaises/françaises.

Une autre possibilité consiste à effacer le fichier source séquentiel allemand dans le paquet anglais/français (fonction spéciale **EFFSEQ**). Un fichier nouveau sera généré automatiquement quand vous appelez **EDITER**.

Les symboles et les commentaires ne seront pas sortis en langues étrangères.

Description

2.3.4 Versions d'un programme STEP 5 en langue étrangère

A l'aide du compilateur par lots, vous pouvez également générer des versions d'un programme STEP 5 en langue étrangère, si vous avez programmé votre programme sous forme absolue ou si vous disposez d'un fichier intermédiaire reconverti qui contient des paramètres absolus: Avec la version de logiciel anglaise et française de ce paquet, vous créez des sources LIST anglaises et françaises.

A cet effet, vous allez au programme utilitaire **AUX.-FICH, TRF.-FICH** et copiez votre fichier source LIST et fichier inter médiaire avec un nom nouveau. Vous inscrivez ce nom nouveau dans le préréglage de la version de logiciel anglaise/française. Vous transférez le fichier intermédiaire qui est indépendant du langage au fichier source séquentiel à l'aide de la fonction spéciale **INT>SEQ** (voir plus haut). Quand vous actionnez **EDITEUR,** le fichier source séquentiel est ensuite sorti avec des instructions STEP 5 anglaises/françaises.

Une autre possibilité consiste à effacer le fichier source séquentiel allemand dans le paquet anglais/français (fonction spéciale **EFFSEQ**). Un fichier nouveau sera généré automatiquement quand vous appelez **EDITER**.

Les symboles et les commentaires ne seront pas sortis en langues étrangères.

2 - 20 C79000-B8577-C677-02

2.4 Modifier et supplémenter des blocs STEP 5

2.4.1 Blocs créés à l'aide de l'éditeur LIST

Dans le préréglage, vous indiquez le nom du fichier source LIST et, le cas échéant, aussi le lecteur pour le fichier intermédiaire. Après que l'éditeur LIST est appelé, la liste d'instructions choisie et le menu d'éditeur apparaît sur l'écran. Maintenant, vous pouvez modifier ou supplémenter la liste d'instructions à l'aide des fonctions d'éditeur. Mémorisez votre fichier source modifié toujours avec *valider* afin que le fichier intermédiaire soit actualisé et votre fichier programme "nouveau" ne soit pas créé avec un programme "ancien".

Si vous quittez l'éditeur LIST avec la **touche d'abandon** et confirmez l'abandon, les modifications ou supplémentations ne seront pas validées dans le fichier source LIST.

2.4.2 Blocs créés à l'aide du paquet CONT-LOG-LIST

Des blocs d'un fichier programme doivent être reconvertis afin qu'ils puissent être traités dans l'éditeur LIST. Quand ils existent ensuite dans un fichier source LIST, ils seront traités dans l'éditeur LIST comme décrit sous 2.4.1.

Description

2.4 Modifier et supplémenter des blocs STEP 5

2.4.1 Blocs créés à l'aide de l'éditeur LIST

Dans le préréglage, vous indiquez le nom du fichier source LIST et, le cas échéant, aussi le lecteur pour le fichier intermédiaire. Après que l'éditeur LIST est appelé, la liste d'instructions choisie et le menu d'éditeur apparaît sur l'écran. Maintenant, vous pouvez modifier ou supplémenter la liste d'instructions à l'aide des fonctions d'éditeur. Mémorisez votre fichier source modifié toujours avec *valider* afin que le fichier intermédiaire soit actualisé et votre fichier programme "nouveau" ne soit pas créé avec un programme "ancien".

Si vous quittez l'éditeur LIST avec la **touche d'abandon** et confirmez l'abandon, les modifications ou supplémentations ne seront pas validées dans le fichier source LIST.

2.4.2 Blocs créés à l'aide du paquet CONT-LOG-LIST

Des blocs d'un fichier programme doivent être reconvertis afin qu'ils puissent être traités dans l'éditeur LIST. Quand ils existent ensuite dans un fichier source LIST, ils seront traités dans l'éditeur LIST comme décrit sous 2.4.1.

2.5 Contrôle

2.5.1 Contrôle d'un fichier programme

La contrôle suit la *conversion* et vérifie les blocs du fichier programme. Elle réalise, p. ex. la vérification du transfert des paramètres dans des blocs fonctionnels et l'existence des blocs appelés. Vous pouvez sélectionner un contrôle d'un bloc, d'un groupe de blocs ou de tous les blocs d'un fichier programme. Si une identification de la catégorie de langage est entrée dans le préréglage, l'admissibilité des instructions quant au type d'AG sera également contrôlée. Des instructions non admissibles seront enregistrées dans la liste d'erreurs.

2.5.2 Contrôle des blocs spéciaux

L'éditeur LIST/compilateur par lots ne peut pas créer et reconvertir des blocs fonctionnels standard, blocs GRAPH 5 et blocs en assembleur mais ceux-ci peuvent être vérifiés avec le *contrôle* succédant! Quant à ces blocs, l'existence et le transfert des paramètres aussi bien que l'autorisation des instructions LIST pour le type d'AG préréglé seront vérifiés.

Description

2.5 Contrôle

2.5.1 Contrôle d'un fichier programme

La contrôle suit la *conversion* et vérifie les blocs du fichier programme. Elle réalise, p. ex. la vérification du transfert des paramètres dans des blocs fonctionnels et l'existence des blocs appelés. Vous pouvez sélectionner un contrôle d'un bloc, d'un groupe de blocs ou de tous les blocs d'un fichier programme. Si une identification de la catégorie de langage est entrée dans le préréglage, l'admissibilité des instructions quant au type d'AG sera également contrôlée. Des instructions non admissibles seront enregistrées dans la liste d'erreurs.

2.5.2 Contrôle des blocs spéciaux

L'éditeur LIST/compilateur par lots ne peut pas créer et reconvertir des blocs fonctionnels standard, blocs GRAPH 5 et blocs en assembleur mais ceux-ci peuvent être vérifiés avec le *contrôle* succédant! Quant à ces blocs, l'existence et le transfert des paramètres aussi bien que l'autorisation des instructions LIST pour le type d'AG préréglé seront vérifiés.

2 - 22

C79000-B8577-C677-02

2.6 Liste d'erreurs

2.6.1 Fichier d'erreurs

Des messages d'erreurs se produisent pendant les phases de travail suivantes:

- conversion du fichier source LIST dans le fichier intermédiaire
- conversion du fichier intermédiaire dans le fichier programme
- reconversion du fichier programme dans le fichier intermédiaire
- reconversion du fichier intermédiaire dans le fichier source LIST
- Vérification du fichier programme (contrôle).

Les messages d'erreurs sont déposés par la console de programmation dans une liste d'erreurs dans le fichier d'erreurs <nom>AF.SEQ. Le fichier d'erreurs ne contient que la liste d'erreurs n'est pas dressée si la dernière phase de travail est correcte ! Vous pouvez sortir le fichier d'erreurs sur l'écran ou sur l'imprimante en format d'impression que vous pouvez choisir.

Description

2.6 Liste d'erreurs

2.6.1 Fichier d'erreurs

Des messages d'erreurs se produisent pendant les phases de travail suivantes:

- conversion du fichier source LIST dans le fichier intermédiaire
- conversion du fichier intermédiaire dans le fichier programme
- reconversion du fichier programme dans le fichier intermédiaire
- reconversion du fichier intermédiaire dans le fichier source LIST
- Vérification du fichier programme (contrôle).

Les messages d'erreurs sont déposés par la console de programmation dans une liste d'erreurs dans le fichier d'erreurs <nom>AF.SEQ. Le fichier d'erreurs ne contient que la liste d'erreurs n'est pas dressée si la dernière phase de travail est correcte ! Vous pouvez sortir le fichier d'erreurs sur l'écran ou sur l'imprimante en format d'impression que vous pouvez choisir.

2

C79000-B8577-C677-02

2 - 23

2.7 Introduction d'instructions STEP 5 avec d'autres éditeurs

2.7.1 Fichier source LIST comme interface

Le fichier source LIST peut être également créé avec d'autres éditeurs. Le préalable pour cela est pourtant que ces éditeurs peuvent traiter de "véritables" tabulateurs (c.-à-d. le code hexadécimal 09H). Autrement la première ligne du fichier source LIST doit fixer, via le caractère de commande # TAB, les colonnes de début des sous-zones individuelles (voir chap. 2.7.3).

Le début du nom du fichier doit consister en six caractères que vous pouvez choisir librement. Il faut introduire A0.SEQ pour les deux derniers caractères du nom et pour l'extension. Un traitement ultérieur de ce fichier avec les outils du paquet éditeur LIST/compilateur par lots est uniquement possible sans problèmes, si vous observez le format du fichier séquentiel décrit ci-dessous. L'éditeur LIST/compilateur par lots vous assiste alors avec la fonction spéciale **SEQ>INT** et avec la *conversion* ultérieure dans le fichier programme, ou avec la conversion directe via la fonction SEQ>MC5.

2.7.2 Format du fichier source séquentiel

Vous introduisez un article dans chaque ligne d'instruction. Un article commence par le caractère de tabulation (09H) et se compose de 4 champs de données qui sont également séparés par des tabulateurs. Le marquage de la fin des articles par "Carriage Return, CR" (=0DH) et "Line feed, LF" (=0AH) est ajouté automatiquement par l'éditeur en fin de ligne via la touche Return. Le nombre maximal des caractères pour les champs suivants est indiqué ci-dessous:

TAB	TAB	TAB	TAB	CR, LF
Adresse 4 caractères	**Instruction** 14 caractères	**Mnémonique** 23 caractères	**Commentaire d'Instruction** 32 caractères	

L'article d'une ligne blanche se compose alors de 4 caractères de tabulation suivis par les caractères "CR" et "LF".

Description

2.7 Introduction d'instructions STEP 5 avec d'autres éditeurs

2.7.1 Fichier source LIST comme interface

Le fichier source LIST peut être également créé avec d'autres éditeurs. Le préalable pour cela est pourtant que ces éditeurs peuvent traiter de "véritables" tabulateurs (c.-à-d. le code hexadécimal 09H). Autrement la première ligne du fichier source LIST doit fixer, via le caractère de commande # TAB, les colonnes de début des sous-zones individuelles (voir chap. 2.7.3).

Le début du nom du fichier doit consister en six caractères que vous pouvez choisir librement. Il faut introduire A0.SEQ pour les deux derniers caractères du nom et pour l'extension. Un traitement ultérieur de ce fichier avec les outils du paquet éditeur LIST/compilateur par lots est uniquement possible sans problèmes, si vous observez le format du fichier séquentiel décrit ci-dessous. L'éditeur LIST/compilateur par lots vous assiste alors avec la fonction spéciale **SEQ>INT** et avec la *conversion* ultérieure dans le fichier programme, ou avec la conversion directe via la fonction SEQ>MC5.

2.7.2 Format du fichier source séquentiel

Vous introduisez un article dans chaque ligne d'instruction. Un article commence par le caractère de tabulation (09H) et se compose de 4 champs de données qui sont également séparés par des tabulateurs. Le marquage de la fin des articles par "Carriage Return, CR" (=0DH) et "Line feed, LF" (=0AH) est ajouté automatiquement par l'éditeur en fin de ligne via la touche Return. Le nombre maximal des caractères pour les champs suivants est indiqué ci-dessous:

TAB	TAB	TAB	TAB	CR, LF
Adresse 4 caractères	**Instruction** 14 caractères	**Mnémonique** 23 caractères	**Commentaire d'Instruction** 32 caractères	

L'article d'une ligne blanche se compose alors de 4 caractères de tabulation suivis par les caractères "CR" et "LF".

2 - 24 C79000-B8577-C677-02

L'article pour des lignes de commentaire commence par le caractère de tabulation (=09H), suivi immédiatement par les caractères de commande * et ; pour des commentaires de segment et des commentaires supplémentaires. Après, 79 caractères au maximum pour le commentaire et la fin de ligne avec les caractères "CR" (=0DH) et "LF" (=0AH).

Les articles peuvent se faire en écriture majuscule ou minuscule, car en lisant, l'éditeur transforme automatiquement toutes les lettres minuscules en lettres majuscules. L'emploi de trémas et d'accents n'est pas permis.

2.7.3 Caractère de commande #TAB pour le traitement de fichiers étrangers

Le caractère de commande #TAB permet la conversion de fichiers sans véritables caractères de tabulation, de même que les créent de nombreux programmes de textes, comme p.ex. 1st Wordplus. L'éditeur LIST ne peut pas traiter ces fichiers, et émet au contraire le message d'erreur "format fichier faux".

#TAB doit se trouver directement au début du fichier source. Devant cela, seuls des espaces sont permis. Puis, 4 nombres doivent suivre, séparés par une virgule, et qui indiquent les colonnes de début des zones individuelles. D'autres données sur la première ligne ne sont pas admises.

Exemple: dans le cas où un espace doit servir respectivement de séparation entre les zones individuelles, la première ligne du fichier source LIST est la suivante :

#TAB 1,6,21,46 RETURN (CR LF)

Les données de colonnes sont donc toujours calculées à partir du début de ligne. La différence entre les données successives doit être au-moins aussi longue que la taille de la zone correspondante (voir 2.7.2).

Description

L'article pour des lignes de commentaire commence par le caractère de tabulation (=09H), suivi immédiatement par les caractères de commande * et ; pour des commentaires de segment et des commentaires supplémentaires. Après, 79 caractères au maximum pour le commentaire et la fin de ligne avec les caractères "CR" (=0DH) et "LF" (=0AH).

Les articles peuvent se faire en écriture majuscule ou minuscule, car en lisant, l'éditeur transforme automatiquement toutes les lettres minuscules en lettres majuscules. L'emploi de trémas et d'accents n'est pas permis.

2

2.7.3 Caractère de commande #TAB pour le traitement de fichiers étrangers

Le caractère de commande #TAB permet la conversion de fichiers sans véritables caractères de tabulation, de même que les créent de nombreux programmes de textes, comme p.ex. 1st Wordplus. L'éditeur LIST ne peut pas traiter ces fichiers, et émet au contraire le message d'erreur "format fichier faux".

#TAB doit se trouver directement au début du fichier source. Devant cela, seuls des espaces sont permis. Puis, 4 nombres doivent suivre, séparés par une virgule, et qui indiquent les colonnes de début des zones individuelles. D'autres données sur la première ligne ne sont pas admises.

Exemple: dans le cas où un espace doit servir respectivement de séparation entre les zones individuelles, la première ligne du fichier source LIST est la suivante :

#TAB 1,6,21,46 RETURN (CR LF)

Les données de colonnes sont donc toujours calculées à partir du début de ligne. La différence entre les données successives doit être au-moins aussi longue que la taille de la zone correspondante (voir 2.7.2).

C79000-B8577-C677-02

2 - 25

Description

2 - 26

C79000-B8577-C677-02

Commande à la console de programmation

3

Commande à la console de programmation

3

C79000-B8577-C677-02

3 - 1

3.1 Installer le logiciel

Copiez les fichiers de l'éditeur LIST/complilateur par lots à l'aide de PIP, la commande PCP/M sur le disque dur de votre console de programmation. A cet effet, consultez le carnet de table PCP/M qui est joint à votre manuel PG.

Exemple: PIP b:[G0]=a:*.*

Dans cet exemple b: est le lecteur destination, [G0] l'utilisateur et a: le lecteur source.

Commande

3.1 Installer le logiciel

Copiez les fichiers de l'éditeur LIST/complilateur par lots à l'aide de PIP, la commande PCP/M sur le disque dur de votre console de programmation. A cet effet, consultez le carnet de table PCP/M qui est joint à votre manuel PG.

Exemple: PIP b:[G0]=a:*.*

Dans cet exemple b: est le lecteur destination, [G0] l'utilisateur et a: le lecteur source.

3 - 2

C79000-B8577-C677-02

3.2 Phases de commande jusqu'à la sélection de fonction

3.2.1 Généralités de la commande

Quant à l'éditeur LIST/complilateur par lots, la commande de la console de programmation correspond aux conventions de tous les autres paquets S5. Cela veut dire que vous remplissez un préréglage et trouvez la barre des touches de fonction (softkeys) avec les fonctions d'éditeur et de traitement dans la sélection des fonctions.

Les **touches du curseur**, les **touches Help**, les **touches d'abandon** et **de validation** sont identiques dans leurs fonctions. Elles travaillent dans l'éditeur LIST de même que dans l'éditeur symbolique. Il y a cependant quelques différences par rapport aux fonctions d'**ENTREE/SORTIE** du paquet CONT-LOG-LIST.

Avec l'éditeur LIST, il est nécessaire, p. ex., d'ajouter des caractères de commande à certaines instructions, les symboles, par contre, ne nécessitent pas de trait d'union. Nous allons y appeler votre attention aux endroits respectifs.

Vous recevez des explications et des informations quant aux masques, touches de fonction et champs d'introduction via la **touche Help**. Dans ces textes Help vous répondez sur la question "Continuer?" avec la **touche d'abandon** (= non) ou avec la **touche de validation** (= oui).

Les définitions de terme de STEP 5 sont également valables dans ce paquet. Si vous désirez des informations générales sur STEP 5, lisez l'introduction dans les manuels de votre console de programmation, s'il vous plaît.

3

3.2.2 Introduction au paquet éditeur LIST/compilateur par lots

☞ L'exemple suivant a été créé sur un PG 750 avec un lecteur disquette. Toutes vos opérations á la console sont marquées avec ➤ . Les caractères que vous introduisez sont imprimés en *italique*, les fonctions ou touches de fonction à utiliser en **gras**; les termes qui apparaissent sur l'écran et auxquels on se réfère pendant la commande sont imprimés en lettres majuscules.

Commande

3.2 Phases de commande jusqu'à la sélection de fonction

3.2.1 Généralités de la commande

Quant à l'éditeur LIST/complilateur par lots, la commande de la console de programmation correspond aux conventions de tous les autres paquets S5. Cela veut dire que vous remplissez un préréglage et trouvez la barre des touches de fonction (softkeys) avec les fonctions d'éditeur et de traitement dans la sélection des fonctions.

Les **touches du curseur**, les **touches Help**, les **touches d'abandon** et **de validation** sont identiques dans leurs fonctions. Elles travaillent dans l'éditeur LIST de même que dans l'éditeur symbolique. Il y a cependant quelques différences par rapport aux fonctions d'**ENTREE/SORTIE** du paquet CONT-LOG-LIST.

Avec l'éditeur LIST, il est nécessaire, p. ex., d'ajouter des caractères de commande à certaines instructions, les symboles, par contre, ne nécessitent pas de trait d'union. Nous allons y appeler votre attention aux endroits respectifs.

Vous recevez des explications et des informations quant aux masques, touches de fonction et champs d'introduction via la **touche Help**. Dans ces textes Help vous répondez sur la question "Continuer?" avec la **touche d'abandon** (= non) ou avec la **touche de validation** (= oui).

Les définitions de terme de STEP 5 sont également valables dans ce paquet. Si vous désirez des informations générales sur STEP 5, lisez l'introduction dans les manuels de votre console de programmation, s'il vous plaît.

3

3.2.2 Introduction au paquet éditeur LIST/compilateur par lots

☞ L'exemple suivant a été créé sur un PG 750 avec un lecteur disquette. Toutes vos opérations á la console sont marquées avec ➤ . Les caractères que vous introduisez sont imprimés en *italique*, les fonctions ou touches de fonction à utiliser en **gras**; les termes qui apparaissent sur l'écran et auxquels on se réfère pendant la commande sont imprimés en lettres majuscules.

C79000-B8577-C677-02

3 - 3

Condition: Le paquet optionnel éditeur LIST/complilateur par lots est chargé sur le lecteur du disque dur de votre console de programmation. La console est prête à être mise en service et le système d'exploitation PCP/M-86 est actif. L'écran sortit le message B> (prêt).

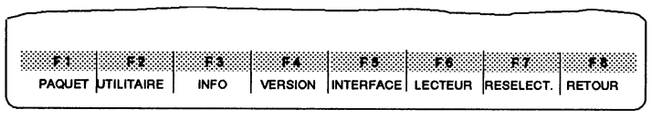
Démarrage du S5-Komi

Démarrez l'interpréteur de commande S5 (S5-KOMI) au sein du masque de démarrage avec la commande

➤ F1 (démarrer S5-DOS)

➤ ou au niveau du système d'exploitation avec la commande : S5

Les paquets STEP 5 seront alors édités sur l'écran en forme de liste pendant la SELECTION DE PAQUET.



La **touche Help** vous explique la signification des touches de fonction (softkeys) individuelles.

La description STEP 5, volume 2 du manuel de votre console de programmation, chapitre "Interpréteur de commande S5" vous donne des informations détaillés sur ces fonctions.

Chargement du paquet éditeur LIST/compilateur par lots

➤ Placer le curseur devant le paquet éditeur LIST/compilateur par lots,

➤ appeler **PAQUET** (F1)

ou

➤ actionner la **touche de validation**.

Le PREREGLAGE apparaît la-dessus.

Commande

Condition: Le paquet optionnel éditeur LIST/complilateur par lots est chargé sur le lecteur du disque dur de votre console de programmation. La console est prête à être mise en service et le système d'exploitation PCP/M-86 est actif. L'écran sortit le message B> (prêt).

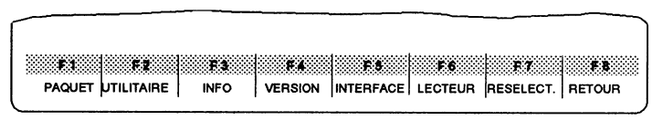
Démarrage du S5-Komi

Démarrez l'interpréteur de commande S5 (S5-KOMI) au sein du masque de démarrage avec la commande

➤ F1 (démarrer S5-DOS)

➤ ou au niveau du système d'exploitation avec la commande : S5

Les paquets STEP 5 seront alors édités sur l'écran en forme de liste pendant la SELECTION DE PAQUET.



La **touche Help** vous explique la signification des touches de fonction (softkeys) individuelles.

La description STEP 5, volume 2 du manuel de votre console de programmation, chapitre "Interpréteur de commande S5" vous donne des informations détaillés sur ces fonctions.

Chargement du paquet éditeur LIST/compilateur par lots

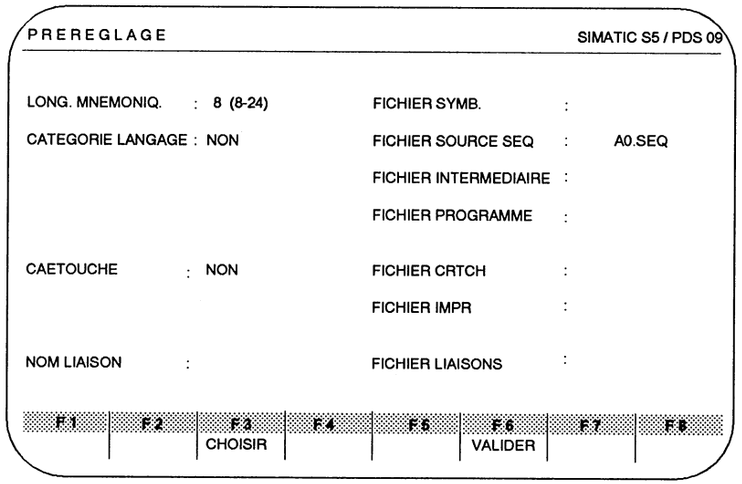
➤ Placer le curseur devant le paquet éditeur LIST/compilateur par lots,

➤ appeler **PAQUET** (F1)

ou

➤ actionner la **touche de validation**.

Le PREREGLAGE apparaît la-dessus.



PREREGLAGE

Le curseur clignote dans la ligne du fichier source LIST (A0.SEQ). Votre liste d'instruction sera mémorisée dans ce fichier. Elle est reconnaissable comme fichier séquentiel, c.-à-d. comme fichier ASCII et est la source pour la conversion.

A cet endroit, introduisez le nom de votre fichier. Dans notre exemple le fichier sera nommé "Test".

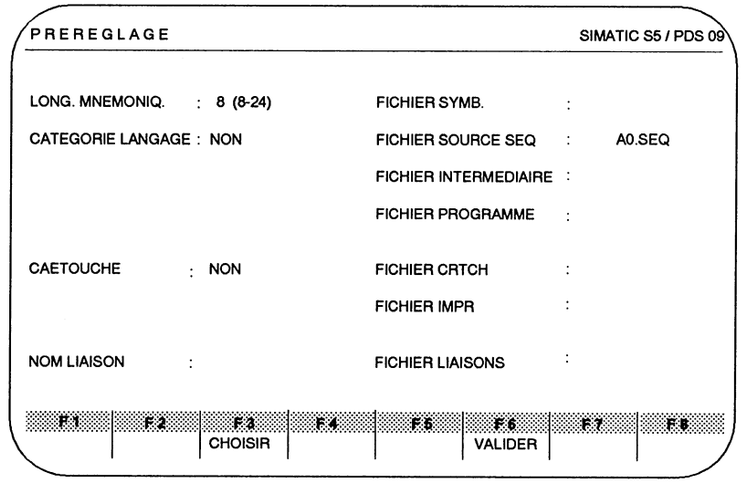
➤ Introduire chaîne de caractères *Test* ,

➤ actionner la **touche Return**.

Maintenant, la console de programmation inscrit le disque dur comme lecteur et remplit le nom par @. En plus, le FICHIER INTERMEDIAIRE (A1.SEQ), le FICHIER SYMBOLIQUE (Z0.INI) et le FICHIER PROGRAMME (ST.S5D) seront remplis automatiquement avec le même nom. Ainsi, il est reconnaissable qu'ils vont ensemble.

Si un FICHIER SYMBOLIQUE n'existe pas, c.-à-d. une liste d'assignations avec nom rempli, cela sera vous indiqué à trois endroits: Derrière le nom du fichier se trouve (GESP), la console de programmation sortit le message "Fichier B:TEST@@Z0.INI: Fichier non existant" et dans le champ LONGUEUR MNEMONIQUE, le 8 préréglé sera transformé en 0.

Commande



3

PREREGLAGE

Le curseur clignote dans la ligne du fichier source LIST (A0.SEQ). Votre liste d'instruction sera mémorisée dans ce fichier. Elle est reconnaissable comme fichier séquentiel, c.-à-d. comme fichier ASCII et est la source pour la conversion.

A cet endroit, introduisez le nom de votre fichier. Dans notre exemple le fichier sera nommé "Test".

➤ Introduire chaîne de caractères *Test* ,

➤ actionner la **touche Return**.

Maintenant, la console de programmation inscrit le disque dur comme lecteur et remplit le nom par @. En plus, le FICHIER INTERMEDIAIRE (A1.SEQ), le FICHIER SYMBOLIQUE (Z0.INI) et le FICHIER PROGRAMME (ST.S5D) seront remplis automatiquement avec le même nom. Ainsi, il est reconnaissable qu'ils vont ensemble.

Si un FICHIER SYMBOLIQUE n'existe pas, c.-à-d. une liste d'assignations avec nom rempli, cela sera vous indiqué à trois endroits: Derrière le nom du fichier se trouve (GESP), la console de programmation sortit le message "Fichier B:TEST@@Z0.INI: Fichier non existant" et dans le champ LONGUEUR MNEMONIQUE, le 8 préréglé sera transformé en 0.

écraser

Si, en conversant, le fichier source doit être relié à un fichier symbolique d'un autre nom et/ou être converti dans un fichier programme d'un autre nom, vous pouvez maintenant écraser les noms individuels. Vous terminez toujours avec la touche Return après l'écrasement. Ces fichiers sont également actifs dans d'autres paquets STEP 5 et seront inscrits dans leurs préréglages pendant le chargement. Les noms des fichiers pour cartouche, imprimante et liaisons seront adaptés automatiquement à celui du fichier programme.

➤ Placer le curseur à la ligne respective,

➤ appuyer sur la **touche flèche vers la droite**,

➤ le champ d'introduction est rempli.

Le PG vérifie, si les fichiers indiqués existent. Si vous voulez utiliser d'autres fichiers, introduisez leurs noms. Si les fichiers imprimante et liaison n'existent pas, leurs noms seront effacés avec le mouvement prochain du curseur vers le haut ou vers le bas.

Les lignes NOM LIAISON et CARTOUCHE seront traitées de même façon que dans le paquet CONT-LOG-LIST: Vous introduisez le nom au nom liaison, pour cartouche vous sélectez la largeur.

CATEGORIE LANGAGE

Au champ CATEGORIE LANGAGE, vous devriez absolument employer la **touche Help** (à cet effet, placez le curseur sur une lettre de NON): La **touche Help** vous indique les appareils d'automatisation (AG) et processeurs centraux (CPU), pour lesquels le compilateur par lots convertit et vérifie de façon spécifique de l'AG.

Le cas échéant, entrez alors au champ catégorie de langage l'appareil de la liste sur lequel vous voulez exécuter votre programme. Pour de l'information supplémentaire, consultez la table au chapitre 2.2.3. Le compilateur par lots vérifie en conversant dans le fichier programme, si votre liste d'instruction concorde avec la catégorie de langage de l'AG.

Le champ LONGUEUR MNEMONIQUE vous indique la longueur mnémonique du fichier symbolique inscrit. Vous ne pouvez pas modifier ce pur champ d'indication.

Pour mémoriser ce PREREGLAGE, actionnez

➤ VALIDER (F6)

ou

➤ la **touche de validation**.

Commande

écraser

Si, en conversant, le fichier source doit être relié à un fichier symbolique d'un autre nom et/ou être converti dans un fichier programme d'un autre nom, vous pouvez maintenant écraser les noms individuels. Vous terminez toujours avec la touche Return après l'écrasement. Ces fichiers sont également actifs dans d'autres paquets STEP 5 et seront inscrits dans leurs préréglages pendant le chargement. Les noms des fichiers pour cartouche, imprimante et liaisons seront adaptés automatiquement à celui du fichier programme.

➤ Placer le curseur à la ligne respective,

➤ appuyer sur la **touche flèche vers la droite**,

➤ le champ d'introduction est rempli.

Le PG vérifie, si les fichiers indiqués existent. Si vous voulez utiliser d'autres fichiers, introduisez leurs noms. Si les fichiers imprimante et liaison n'existent pas, leurs noms seront effacés avec le mouvement prochain du curseur vers le haut ou vers le bas.

Les lignes NOM LIAISON et CARTOUCHE seront traitées de même façon que dans le paquet CONT-LOG-LIST: Vous introduisez le nom au nom liaison, pour cartouche vous sélectez la largeur.

CATEGORIE LANGAGE

Au champ CATEGORIE LANGAGE, vous devriez absolument employer la **touche Help** (à cet effet, placez le curseur sur une lettre de NON): La **touche Help** vous indique les appareils d'automatisation (AG) et processeurs centraux (CPU), pour lesquels le compilateur par lots convertit et vérifie de façon spécifique de l'AG.

Le cas échéant, entrez alors au champ catégorie de langage l'appareil de la liste sur lequel vous voulez exécuter votre programme. Pour de l'information supplémentaire, consultez la table au chapitre 2.2.3. Le compilateur par lots vérifie en conversant dans le fichier programme, si votre liste d'instruction concorde avec la catégorie de langage de l'AG.

Le champ LONGUEUR MNEMONIQUE vous indique la longueur mnémonique du fichier symbolique inscrit. Vous ne pouvez pas modifier ce pur champ d'indication.

Pour mémoriser ce PREREGLAGE, actionnez

➤ VALIDER (F6)

ou

➤ la **touche de validation**.

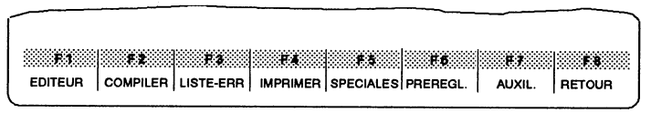
3 - 6

C79000-B8577-C677-02

SELECTION DES FONCTIONS

Vous recevez maintenant la SELECTION DES FONCTIONS

La SELECTION DES FONCTIONS vous offre sur la barre des touches de fonction les fonctions d'éditeur et de traitement suivantes. Leur commande sera expliquée aux chapitres suivants.

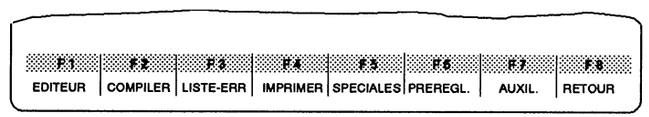


Commande

SELECTION DES FONCTIONS

Vous recevez maintenant la SELECTION DES FONCTIONS

La SELECTION DES FONCTIONS vous offre sur la barre des touches de fonction les fonctions d'éditeur et de traitement suivantes. Leur commande sera expliquée aux chapitres suivants.

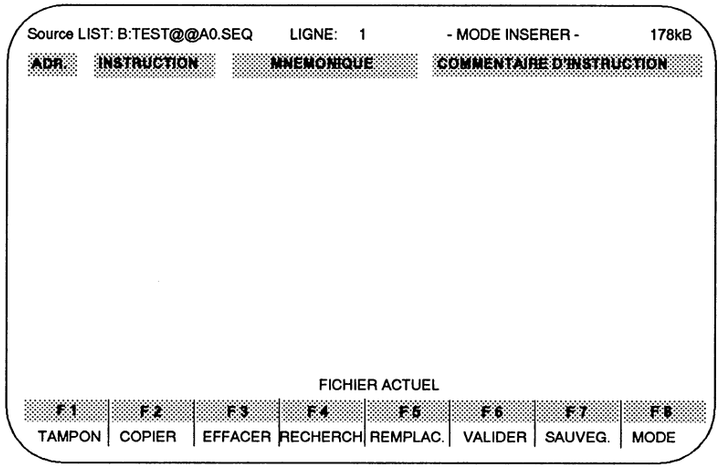


3

3.3 Editer

Appeler le mode d'éditeur

Appeler **EDITER** (F1).



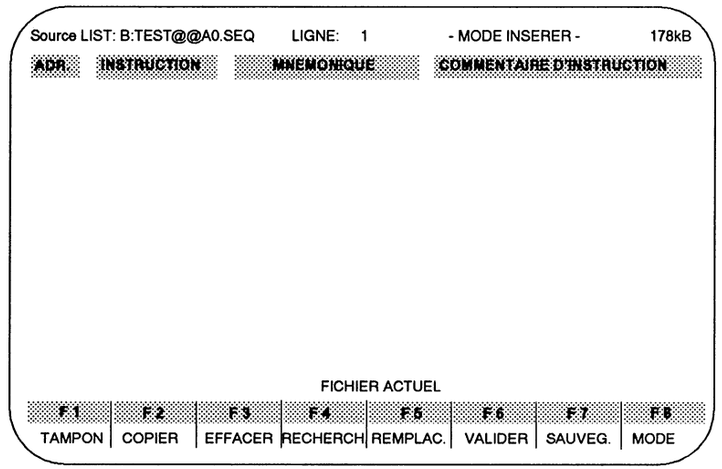
Cet écran est préparé d'éditer une liste d'instructions, c.-à-d., l'introduire à nouveau ou en sortir une pour le traitement (corrections, modifications).

Commande

3.3 Editer

Appeler le mode d'éditeur

Appeler **EDITER** (F1).



Cet écran est préparé d'éditer une liste d'instructions, c.-à-d., l'introduire à nouveau ou en sortir une pour le traitement (corrections, modifications).

3.3.1 Description de l'éditeur

Ligne d'en-tête

Ici, vous trouvez :

- le nom de votre fichier source LIST préréglé et le lecteur joint,
- l'indication de ligne pour la position du curseur,
- le mode d'éditeur insérer ou écraser et
- les grandeurs de tampon de la mémoire. Cette indication est intéressante pour la vitesse de traitement.

Champ d'éditeur

Le champ d'éditeur est partagé en quatre colonnes dont la largeur ne peut pas être modifiée. Un aperçu des valeurs et du contenu possible des colonnes est donné ci-dessous:

ADR 4 caractères	INSTRUCTION 14 caractères	MNEMONIQUE 23 caractères	COMMENT. D'INSTRUCT. 32 caractères (max. longueur de mnémon.)
Adresses, Marques de saut	Opérations, opérandes absolus, Constantes	Mnémoniques Valeur des constantes	Commentaires

Ligne au bas de la page

Ici se trouvent tous les messages de l'appareil, p. ex. "Fichier nouveau", lorsque vous créez une liste d'instructions nouvelle.

Commande

3.3.1 Description de l'éditeur

Ligne d'en-tête

Ici, vous trouvez :

- le nom de votre fichier source LIST préréglé et le lecteur joint,
- l'indication de ligne pour la position du curseur,
- le mode d'éditeur insérer ou écraser et
- les grandeurs de tampon de la mémoire. Cette indication est intéressante pour la vitesse de traitement.

Champ d'éditeur

Le champ d'éditeur est partagé en quatre colonnes dont la largeur ne peut pas être modifiée. Un aperçu des valeurs et du contenu possible des colonnes est donné ci-dessous:

3

ADR 4 caractères	INSTRUCTION 14 caractères	MNEMONIQUE 23 caractères	COMMENT. D'INSTRUCT. 32 caractères (max. longueur de mnémon.)
Adresses, Marques de saut	Opérations, opérandes absolus, Constantes	Mnémoniques Valeur des constantes	Commentaires

Ligne au bas de la page

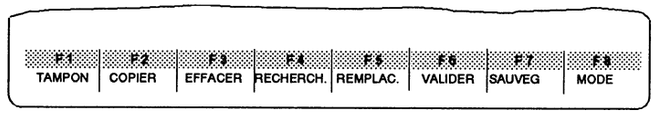
Ici se trouvent tous les messages de l'appareil, p. ex. "Fichier nouveau", lorsque vous créez une liste d'instructions nouvelle.

C79000-B8577-C677-02

3 - 9

Barre de touches de fonction (softkeys)

Ces fonctions d'éditeur sont identiques à ceux de l'EDITEUR SYMBOLIQUE. Elles servent à créer et traiter une liste d'instructions.



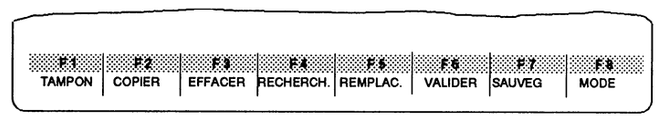
Fonctions de la barre des touches de fonction EDITER

La graphique suivante donne un aperçu des "outils" qui sont disponible dans les fonctions d'éditeur individuelles. Si vous appuyez sur une des touches du menu d'éditeur, le menu correspondant (au bout de la flèche) sera édité.

Commande

Barre de touches de fonction (softkeys)

Ces fonctions d'éditeur sont identiques à ceux de l'EDITEUR SYMBOLIQUE. Elles servent à créer et traiter une liste d'instructions.



Fonctions de la barre des touches de fonction EDITER

La graphique suivante donne un aperçu des "outils" qui sont disponible dans les fonctions d'éditeur individuelles. Si vous appuyez sur une des touches du menu d'éditeur, le menu correspondant (au bout de la flèche) sera édité.

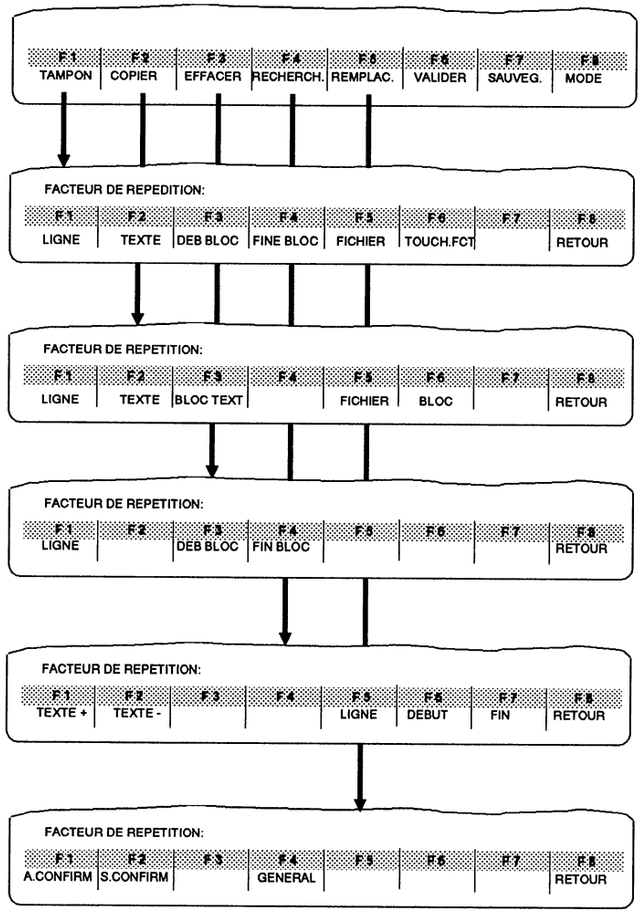
TAMPON

COPIER

EFFACER

RECHERCH

REMPLAC.



3

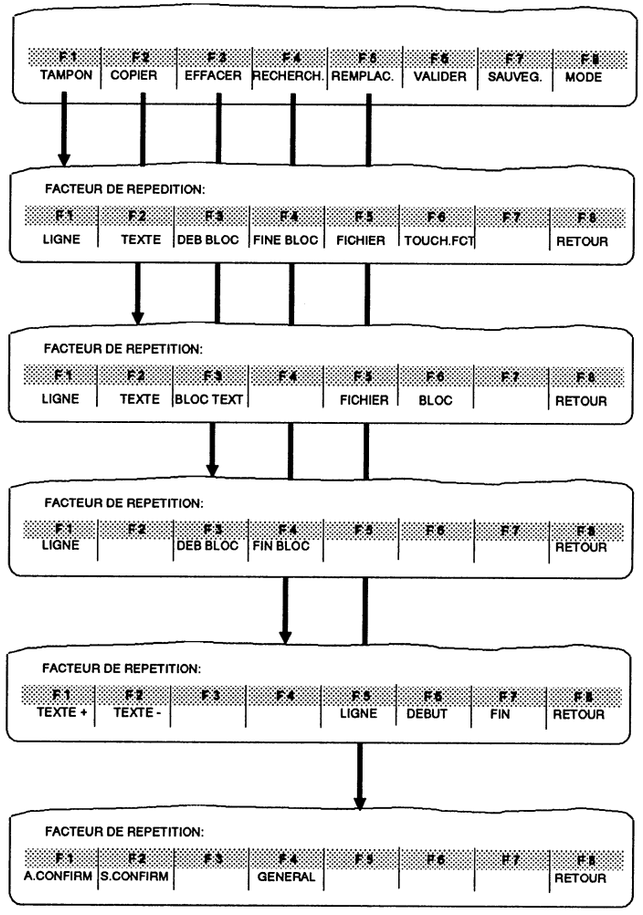
TAMPON

COPIER

EFFACER

RECHERCH

REMPLAC.



3

les touches spéciales

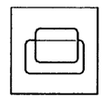
Hardkeys (touches à fonctions invariables)

Outre ces fonctions, les touches spéciales vous offre des "outils" ultérieurs pour traiter votre fichier.

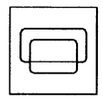
Toutes les touches du curseur sont à votre disposition pour le déplacement du curseur.

Le caractère sur lequel vous vous trouvez sera effacé par ou touche DEL

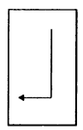
Dans votre fichier, vous tournez la page en avant avec (Remuer le texte vers le haut)



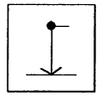
et en arrière avec (Remuer le texte vers le bas)



En MODE INSERER, vous recevez une ligne nouvelle au-dessous de la position du curseur avec



une ligne nouvelle au-dessus de la position du curseur "insertion d'un espace vertical",



un espace également avec "insertion d'un espace horizontal" (en plus de la touche d'espace).

Commande

les touches spéciales

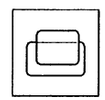
Hardkeys (touches à fonctions invariables)

Outre ces fonctions, les touches spéciales vous offre des "outils" ultérieurs pour traiter votre fichier.

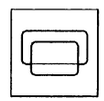
Toutes les touches du curseur sont à votre disposition pour le déplacement du curseur.

Le caractère sur lequel vous vous trouvez sera effacé par ou touche DEL

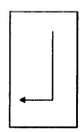
Dans votre fichier, vous tournez la page en avant avec (Remuer le texte vers le haut)



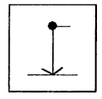
et en arrière avec (Remuer le texte vers le bas)



En MODE INSERER, vous recevez une ligne nouvelle au-dessous de la position du curseur avec



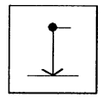
une ligne nouvelle au-dessus de la position du curseur "insertion d'un espace vertical",



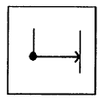
un espace également avec "insertion d'un espace horizontal" (en plus de la touche d'espace).

Vous effacez le caractère
à la gauche du curseur avec
ou touche "-"

En MODE ECRASER
vous n'insérez une ligne nouvelle qu'avec



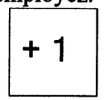
un espace ultérieur uniquement avec



3

Si vous voulez effacer du texte entre les deux deux-points au sein d'une commande dans la ligne au bas de la page, employez:

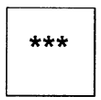
ou touche "+"



Via la touche d'abandon vous terminez chaque fonction activée, mais vous perdez peut-être des données. Si vous avez p. ex. corrigé un fichier et ensuite actionnez la touche d'abandon, toutes les modifications se perdent.

ou touche ESC

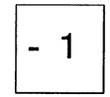
La touche de fin de segment



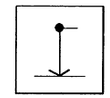
produit "***", c.-à-d., une instruction de fin de segment, si le curseur se trouve dans le champ d'"intruction". En dehors de ce champ ainsi que pour commentaires supplémentaires et commentaires de segment, la touche est fermée.

Commande

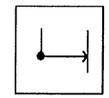
Vous effacez le caractère
à la gauche du curseur avec
ou touche "-"



En MODE ECRASER
vous n'insérez une ligne nouvelle qu'avec



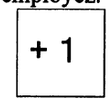
un espace ultérieur uniquement avec



3

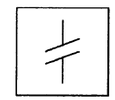
Si vous voulez effacer du texte entre les deux deux-points au sein d'une commande dans la ligne au bas de la page, employez:

ou touche "+"

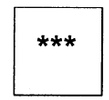


Via la touche d'abandon vous terminez chaque fonction activée, mais vous perdez peut-être des données. Si vous avez p. ex. corrigé un fichier et ensuite actionnez la touche d'abandon, toutes les modifications se perdent.

ou touche ESC



La touche de fin de segment



produit "***", c.-à-d., une instruction de fin de segment, si le curseur se trouve dans le champ d'"intruction". En dehors de ce champ ainsi que pour commentaires supplémentaires et commentaires de segment, la touche est fermée.

3.3.2 Les caractères de commande de l'éditeur LIST/compilateur par lots et leur conventions d'écriture

Pour certaines introductions, l'EDITEUR LIST demande plusieurs caractères de commande afin que la conversion de la liste d'instructions dans un fichier programme STEP 5 soit possible. Il faut p. ex. que les titres et commentaires de segment, les opérandes actuels et identifications de bloc soient reconnaissable comme titres et commentaires de segment, etc.

Colonne INSTRUCTIONS Caractères de commande	Caractéristiques pour	Conventions d'écriture avec exemples
#TAB	fichier source sans tabulateurs véritables	#TAB 1,6,21,46
#TY	type d'AG	avec espace #TY_AG155U #TY_CPU928
#PBn #OBn #FBn, #FXn #DBn, #DXn (#SBn, pas de bloc GRAPH 5)	début du bloc de programme début du bloc d'organisation début du bloc fonctionnel début du bloc de données début du bloc séquentiel	sans espace #PB11 #OB1 #FB25, #FX12 #DB5, #DX33 #SB3
#BI	numéro de bibliothèque	avec espace #BI_12345 pas plus grand que 65535
#N	nom d'un bloc fonctionnel	avec espace #N_GARAGE 6 caract. au max.

Caractères de commande/conventions d'écriture

3.3.2 Les caractères de commande de l'éditeur LIST/compilateur par lots et leur conventions d'écriture

Pour certaines introductions, l'EDITEUR LIST demande plusieurs caractères de commande afin que la conversion de la liste d'instructions dans un fichier programme STEP 5 soit possible. Il faut p. ex. que les titres et commentaires de segment, les opérandes actuels et identifications de bloc soient reconnaissable comme titres et commentaires de segment, etc.

Colonne INSTRUCTIONS Caractères de commande	Caractéristiques pour	Conventions d'écriture avec exemples
#TAB	fichier source sans tabulateurs véritables	#TAB 1,6,21,46
#TY	type d'AG	avec espace #TY_AG155U #TY_CPU928
#PBn #OBn #FBn, #FXn #DBn, #DXn (#SBn, pas de bloc GRAPH 5)	début du bloc de programme début du bloc d'organisation début du bloc fonctionnel début du bloc de données début du bloc séquentiel	sans espace #PB11 #OB1 #FB25, #FX12 #DB5, #DX33 #SB3
#BI	numéro de bibliothèque	avec espace #BI_12345 pas plus grand que 65535
#N	nom d'un bloc fonctionnel	avec espace #N_GARAGE 6 caract. au max.

3 - 14 C79000-B8577-C677-02

La table d'aperçu liste ces caractères de commande. Elle montre l'ordre spécifié pour une conversion sans problème en fichier intermédiaire et programme. Elle vous informe également sur les conventions d'écriture (_ représente un espace) et la position des caractères de commande au sein de la liste d'instructions, et elle donne des explications ultérieures.

Position au sein de la liste d'instruction	Explications
toujours la première ligne d'un fichier	Permet la conversion de fichiers qui sont établis à l'aide d'un éditeur externe, p.ex. 1st Wordplus. Valable pour le compilateur, non pour l'editeur LIST!
toujours la première instruction d'un fichier	Des commentaires éventuels ne se trouvent que dans le fichier source LIST, ils ne seront pas convertis et se perdent au cours de la reconversion.
début d'un bloc; après un BE (Fin de bloc voir plus bas, Opérations)	Portée de valeur: n= 0 - 255, selon le type d'AG Si vous voulez introduire des instructions ultérieures après une fin de bloc, elles doivent être précédées d'un début de bloc nouveau, car autrement ces instructions se perdront pendant la conversion dans la console de programmation. DB0, DB1, DB2 ne sont pas permis.
après le début du bloc ou après le nom du bloc (voir plus bas, #N)	Pour vos propres numéros de bibliothèque; vous ne pouvez et devez pas introduire les numéros de blocs fonctionnels standard. Des commentaires éventuels ne se trouvent que dans le fichier source LIST, ils ne seront pas convertis et se perdent au cours de la reconversion.
avant ou après le numéro de bibliothèque, mais au début du bloc	

Caractères de commande/conventions d'écriture

La table d'aperçu liste ces caractères de commande. Elle montre l'ordre spécifié pour une conversion sans problème en fichier intermédiaire et programme. Elle vous informe également sur les conventions d'écriture (_ représente un espace) et la position des caractères de commande au sein de la liste d'instructions, et elle donne des explications ultérieures.

Position au sein de la liste d'instruction	Explications
toujours la première ligne d'un fichier	Permet la conversion de fichiers qui sont établis à l'aide d'un éditeur externe, p.ex. 1st Wordplus. Valable pour le compilateur, non pour l'editeur LIST!
toujours la première instruction d'un fichier	Des commentaires éventuels ne se trouvent que dans le fichier source LIST, ils ne seront pas convertis et se perdent au cours de la reconversion.
début d'un bloc; après un BE (Fin de bloc voir plus bas, Opérations)	Portée de valeur: n= 0 - 255, selon le type d'AG Si vous voulez introduire des instructions ultérieures après une fin de bloc, elles doivent être précédées d'un début de bloc nouveau, car autrement ces instructions se perdront pendant la conversion dans la console de programmation. DB0, DB1, DB2 ne sont pas permis.
après le début du bloc ou après le nom du bloc (voir plus bas, #N)	Pour vos propres numéros de bibliothèque; vous ne pouvez et devez pas introduire les numéros de blocs fonctionnels standard. Des commentaires éventuels ne se trouvent que dans le fichier source LIST, ils ne seront pas convertis et se perdent au cours de la reconversion.
avant ou après le numéro de bibliothèque, mais au début du bloc	

3

C79000-B8577-C677-02

3 - 15

Colonne INSTRUCTIONS Caractères de commande	**Caractèristiques pour**	**Conventions d'écriture avec exemples**
#UB	titre de segment	Le caractère de commande se trouve dans la colonne INSTRUCTION, le texte de titre dans la colonne COMMENTAIRE D'INSTRUCTION.
( )	Type de paramètre formel	Le type de paramètre formel doit être placé entre parenthèses. (D) (E)
, Opérandes actuels pour le paramétrage d'un bloc fonctionnel	premier caractère dans la colonne; le paramètre suit immédiatement ,E1.0	
#I	fichier Include	avec espace, indication du lecteur et des six premiers caractères du nom de fichier #I_A:UEBUNG
#	nom de bloc symbolique	

Caractères de commande/conventions d'écriture

Colonne INSTRUCTIONS Caractères de commande	**Caractèristiques pour**	**Conventions d'écriture avec exemples**
#UB	titre de segment	Le caractère de commande se trouve dans la colonne INSTRUCTION, le texte de titre dans la colonne COMMENTAIRE D'INSTRUCTION.
( )	Type de paramètre formel	Le type de paramètre formel doit être placé entre parenthèses. (D) (E)
, Opérandes actuels pour le paramétrage d'un bloc fonctionnel	premier caractère dans la colonne; le paramètre suit immédiatement ,E1.0	
#I	fichier Include	avec espace, indication du lecteur et des six premiers caractères du nom de fichier #I_A:UEBUNG
#	nom de bloc symbolique	

3 - 16 C79000-B8577-C677-02

Position au sein de la liste d'instructions	Explications
seulement au début d'un segment	Ces textes de commentaire seront transférés au fichier programmme. Si vous désirez des informations plus détaillées sur la commentation d'un programme STEP 5, consultez la description STEP 5 s.v.p., dans le manuel volume 2 de votre console de programmation.
directement sous le nom de bloc	
au sein d'un bloc	
seulement aux extrémités de blocs: avant le premier bloc ou entre BE et #PBn	Le caractère de commande rend possible d'inclure d'autres fichiers (Include). Cependant, ces fichiers doivent exister en forme de fichiers intermédiaires, c.-à-d., soit terminés via touche de validation dans l'éditeur LIST soit créés via reconversion. Faites attention, qu'il n'y a pas de noms de bloc identiques dans les fichiers que vous voulez relier. Car pendant la création du fichier programme, le dernier bloc du même nom écrase le bloc précédent. Le fichier Include sera relié au fichier symbolique préréglé en conversant. Par conséquent, ce-ci doit être capable de livrer des assignations au fichier Include.

Caractères de commande/conventions d'écriture

Position au sein de la liste d'instructions	Explications
seulement au début d'un segment	Ces textes de commentaire seront transférés au fichier programmme. Si vous désirez des informations plus détaillées sur la commentation d'un programme STEP 5, consultez la description STEP 5 s.v.p., dans le manuel volume 2 de votre console de programmation.
directement sous le nom de bloc	
au sein d'un bloc	
seulement aux extrémités de blocs: avant le premier bloc ou entre BE et #PBn	Le caractère de commande rend possible d'inclure d'autres fichiers (Include). Cependant, ces fichiers doivent exister en forme de fichiers intermédiaires, c.-à-d., soit terminés via touche de validation dans l'éditeur LIST soit créés via reconversion. Faites attention, qu'il n'y a pas de noms de bloc identiques dans les fichiers que vous voulez relier. Car pendant la création du fichier programme, le dernier bloc du même nom écrase le bloc précédent. Le fichier Include sera relié au fichier symbolique préréglé en conversant. Par conséquent, ce-ci doit être capable de livrer des assignations au fichier Include.

3

C79000-B8577-C677-02

3 - 17

Colonne ADR. Caractères de commande	**Caractèristiques pour**	**Conventions d'écriture avec exemples**
Q	Commentaire de segment	Le caractère de commande n'est placé qu'au début d'un segment; s'il y a un titre de segment, il doit précéder immédiatement.
: ;	Commentaire supplémentaire	Le caractère de commande se trouve dans la colonne ADR., toute la largeur de l'écran (sans considération des colonnes) est à la disposition pour le texte.

Caractères de commande/conventions d'écriture

Colonne ADR. Caractères de commande	**Caractèristiques pour**	**Conventions d'écriture avec exemples**
Q	Commentaire de segment	Le caractère de commande n'est placé qu'au début d'un segment; s'il y a un titre de segment, il doit précéder immédiatement.
: ;	Commentaire supplémentaire	Le caractère de commande se trouve dans la colonne ADR., toute la largeur de l'écran (sans considération des colonnes) est à la disposition pour le texte.

3 - 18

C79000-B8577-C677-02

Position au sein de la liste d'instructions	Explications
à n'importe quel endroit au sein d'un bloc	Ces commentaires supplémentaires ne se trouvent que dans le fichier source LIST. Ils ne seront pas tenus en compte en conversant. Si vous reconversez dans le même fichier source LIST, ces commentaires seront perdus là.

Caractères de commande/conventions d'écriture

Position au sein de la liste d'instructions	Explications
à n'importe quel endroit au sein d'un bloc	Ces commentaires supplémentaires ne se trouvent que dans le fichier source LIST. Ils ne seront pas tenus en compte en conversant. Si vous reconversez dans le même fichier source LIST, ces commentaires seront perdus là.

3

C79000-B8577-C677-02

3 - 19

3.3.3 Les opérations STEP 5 dans l'éditeur LIST/compilateur par lots et leurs conventions d'écriture

L'éditeur LIST/compilateur par lots offre le spectre complet des opérations STEP 5. Seule la catégorie de langage de l'appareil d'automatisation ou du CPU détermine les limites. En programmant, consultez donc la liste d'opérations de votre appareil.

	ADRESSE	INSTRUCTION
opération avec opérande absolu		opération et opérande absolu entrée sans format
avec opérande symbolique		opération *U*
opération avec données		opération et format de données *L_KT* entrée sans format
données	adresse *11*	format de données *KH* *KF* *KC* ou *C* *KG* *KT* *KZ* *KY* ou *A* *KM*

Opérations STEP 5

3.3.3 Les opérations STEP 5 dans l'éditeur LIST/compilateur par lots et leurs conventions d'écriture

L'éditeur LIST/compilateur par lots offre le spectre complet des opérations STEP 5. Seule la catégorie de langage de l'appareil d'automatisation ou du CPU détermine les limites. En programmant, consultez donc la liste d'opérations de votre appareil.

	ADRESSE	INSTRUCTION
opération avec opérande absolu		opération et opérande absolu entrée sans format
avec opérande symbolique		opération *U*
opération avec données		opération et format de données *L_KT* entrée sans format
données	adresse *11*	format de données *KH* *KF* *KC* ou *C* *KG* *KT* *KZ* *KY* ou *A* *KM*

3 - 20 C79000-B8577-C677-02

La table suivante, qui tient compte des colonnes de l'écran, liste les conventions d'écriture pour le paramétrage absolu et symbolique.

MNEMONIQUE	COMMENTAIRE D'INSTRUCTION
	touche "extérieur ouvert"
mnémonique *T-OUV E* sans trait d'union	
valeur de la donnée *005.2*	
valeur, 1 mot de donnée par ligne *6248* *+ 13512* 'indication' seuls guillemets simples, jusqu'à 11 mots de donnée par ligne -1169368-38 1 mot de donnée par ligne au max. 123.1 735 125,018 00011100 11101111	

Opérations STEP 5

La table suivante, qui tient compte des colonnes de l'écran, liste les conventions d'écriture pour le paramétrage absolu et symbolique.

MNEMONIQUE	COMMENTAIRE D'INSTRUCTION
	touche "extérieur ouvert"
mnémonique *T-OUV E* sans trait d'union	
valeur de la donnée *005.2*	
valeur, 1 mot de donnée par ligne *6248* *+ 13512* 'indication' seuls guillemets simples, jusqu'à 11 mots de donnée par ligne -1169368-38 1 mot de donnée par ligne au max. 123.1 735 125,018 00011100 11101111	

3

C79000-B8577-C677-02

3 - 21

	ADRESSE	**INSTRUCTION**
opérande formel: Définition	nom *TEMPS* *TO-E* *MBA* 4 caract. au max.	type *(D)* *(E)* *(A)* entre parenthèses
opération avec opérande formel	opération et opérande formel *U_=TO-E* *=_=MBA* entrée sans format, l'opérande formel doit être précédé immédiatement par un signe d'égalité.	
opérandes actuels(=paramètres pour FBs), absolus	caractères de commande avec opérande *,E1.2* *,DW1* sans espace	
symbolique	caractère de commande ,	
données	caractère de commande avec type de donnée *,KT*	
marques de saute	marque *MARCHE* *M003*	
adresses relatives, adresses de mot de donnée	*17*	
fin de bloc		

Opérations STEP 5

	ADRESSE	**INSTRUCTION**
opérande formel: Définition	nom *TEMPS* *TO-E* *MBA* 4 caract. au max.	type *(D)* *(E)* *(A)* entre parenthèses
opération avec opérande formel	opération et opérande formel *U_=TO-E* *=_=MBA* entrée sans format, l'opérande formel doit être précédé immédiatement par un signe d'égalité.	
opérandes actuels(=paramètres pour FBs), absolus	caractères de commande avec opérande *,E1.2* *,DW1* sans espace	
symbolique	caractère de commande ,	
données	caractère de commande avec type de donnée *,KT*	
marques de saute	marque *MARCHE* *M003*	
adresses relatives, adresses de mot de donnée	*17*	
fin de bloc		

3 - 22

C79000-B8577-C677-02

MNEMONIQUE	COMMENTAIRE D'INSTRUCTION
mnémonique *MBA*	
valeur 005.2	
BE	

Opérations STEP 5

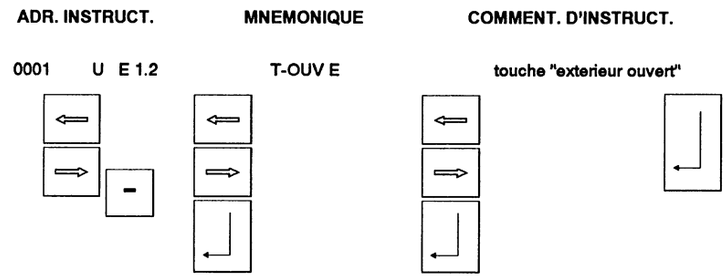
MNEMONIQUE	COMMENTAIRE D'INSTRUCTION
mnémonique *MBA*	
valeur 005.2	
BE	

3

C79000-B8577-C677-02

3 - 23

Utilisez les touches de flèche large et Return entre les colonnes de l'écran pour le déplacement du curseur. La touche Return positionne le curseur toujours sur le 1. caractère de la colonne d'INSTRUCTION.



Symboles

Si vous programmez sous forme symbolique, faites attention de ne pas mettre un trait d'union avant le symbole contrairement au paquet CONT-LOG-LIST. Un début d'un bloc ne peut être représenté par un symbole que dans les cas où il y a une assignation correspondante du type de bloc et du numéro à un symbole. Sinon, programmez le début du bloc sous forme absolu, p. ex. #PB3, car le compilateur par lots nécessite le type de bloc exact et son numéro pour la création du fichier intermédiaire.

Les symboles que vous utilisez dans l'éditeur LIST doivent absolument concorder avec ceux du fichier symbolique. Cela vaut également pour les espaces:

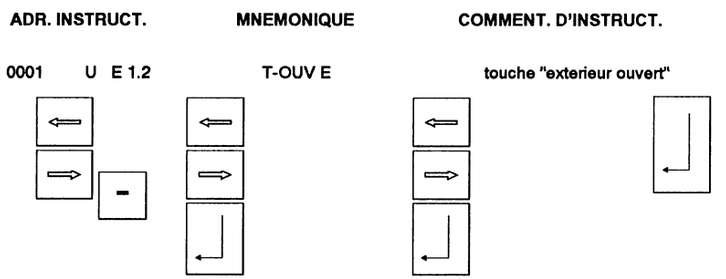
_ARRET D'URGENCE ≠ ARRET D'URGENCE

Des différences ultérieures par rapport au paquet CONT-LOG-LIST sont

- caractères de commande,
- espaces à introduire manuellement pendant des opérations,
- constants de donnée et valeur se trouvent dans des colonnes différentes.

Commande

Utilisez les touches de flèche large et Return entre les colonnes de l'écran pour le déplacement du curseur. La touche Return positionne le curseur toujours sur le 1. caractère de la colonne d'INSTRUCTION.



Symboles

Si vous programmez sous forme symbolique, faites attention de ne pas mettre un trait d'union avant le symbole contrairement au paquet CONT-LOG-LIST. Un début d'un bloc ne peut être représenté par un symbole que dans les cas où il y a une assignation correspondante du type de bloc et du numéro à un symbole. Sinon, programmez le début du bloc sous forme absolu, p. ex. #PB3, car le compilateur par lots nécessite le type de bloc exact et son numéro pour la création du fichier intermédiaire.

Les symboles que vous utilisez dans l'éditeur LIST doivent absolument concorder avec ceux du fichier symbolique. Cela vaut également pour les espaces:

_ARRET D'URGENCE ≠ ARRET D'URGENCE

Des différences ultérieures par rapport au paquet CONT-LOG-LIST sont

- caractères de commande,
- espaces à introduire manuellement pendant des opérations,
- constants de donnée et valeur se trouvent dans des colonnes différentes.

3.3.4 Introduction de blocs programme

Exemple de programmation

Le fichier imprimé sur la page suivante servira d'exemple de travail. Avec cet exemple, nous vous expliquons la commande de l'éditeur LIST/compilateur par lots et des fonctions de ce paquet.

Le programme commande une porte de garage. De l'extérieur, elle est ouverte et fermée par une clé et une touche respective, de l'intérieur, il suffit d'actionner les touches "ouvert" et "fermé". La porte sera fermée avec un retard de 5 secondes.

Commande

3.3.4 Introduction de blocs programme

Exemple de programmation

Le fichier imprimé sur la page suivante servira d'exemple de travail. Avec cet exemple, nous vous expliquons la commande de l'éditeur LIST/compilateur par lots et des fonctions de ce paquet.

Le programme commande une porte de garage. De l'extérieur, elle est ouverte et fermée par une clé et une touche respective, de l'intérieur, il suffit d'actionner les touches "ouvert" et "fermé". La porte sera fermée avec un retard de 5 secondes.

3

C79000-B8577-C677-02

3 - 25

```
Source LIST: B:TEST@@A0.SEQ

ADR.    INSTRUCTION              MNEMONIQUE       COMMENTAIRE D'INSTRUCTION
              ǂPB1
              ǂUB                                 OUVRIR DE L'EXTERIEUR ET DE L'INTERIEUR
✱LA TOUCHE "OUVERT EXTERIEUR" ET LE BOUTON A CLE OU LA TOUCHE "OUVERT INTERIEUR"
✱METTENT EN SERVICE LE MOTEUR VERS LE HAUT. LE MOTEUR TOURNE JUSQU'A CE QUE LA
✱FIN DE COURSE EN HAUT SOIT ATTEINTE OU LA TOUCHE DE SECURITE STOP SOIT
✱APPUYEE.
              U(
              U        E  1.2                   TOUCHE OUVERT EXTERIEUR
              U        E  1.4                   BOUTON A CLE
              O        E  1.5                   TOUCHE OUVERT INTERIEUR
              )
              UN       E  1.0                   FIN DE COURSE EN HAUT
              S        A  1.0                   MOTEUR VERS LE HAUT
              ✱✱✱
              ǂUB                                 OUVRIR DE L'EXTERIEUR ET DE L'INTERIEUR
✱RAZ SORTIE MOTEUR VERS LE HAUT.
              O        E 1.0                    FIN DE COURSE EN HAUT
              O        E 1.7                    TOUCHE DE SECURITE STOP
              R        A 1.0                    MOTEUR VERS LE HAUT
              ✱✱✱
              ǂUB                                 FERMER DE L'EXTERIEUR OU DE L'INTERIEUR
✱LA TOUCHE "FERME EXTERIEUR" ET LE BOUTON A CLE OU LA TOUCHE "FERME INTERIEUR"
✱METTENT EN SERVICE LE MOTEUR VERS LE BAS AVEC UN RETARD A L'ENCHLENCHEMENT DE 5 SEC.
✱LE MOTEUR VERS LE BAS TOURNE JUSQU'A CE QUE LA FIN DE COURSE EN BAS
✱SOIT ATTEINTE OU LA TOUCHE DE SECURITE STOP SOIT APPUYEE.
              U(
              U                T-FER E
              U                BOUTCLE
              O                T-FER I
              )
              UN               FIN-BAS
              L        KT      005.2
              SS               RET-ENCL
              O                FIN-BAS
              O                STOP
              R                RET-ENCL
              L                RET-ENCL
              T   MW 100
              LC               RET-ENCL
              T   MW 102
              U                RET-ENCL
              =                MOT-BAS
              BE
```


Commande

Source LIST: B:TEST@@A0.SEQ

ADR.   INSTRUCTION   MNEMONIQUE   COMMENTAIRE D'INSTRUCTION
              ǂPB1
              ǂUB   OUVRIR DE L'EXTERIEUR ET DE L'INTERIEUR
✱LA TOUCHE "OUVERT EXTERIEUR" ET LE BOUTON A CLE OU LA TOUCHE "OUVERT INTERIEUR"
✱METTENT EN SERVICE LE MOTEUR VERS LE HAUT. LE MOTEUR TOURNE JUSQU'A CE QUE LA
✱FIN DE COURSE EN HAUT SOIT ATTEINTE OU LA TOUCHE DE SECURITE STOP SOIT
✱APPUYEE.
              U(
              U   E 1.2   TOUCHE OUVERT EXTERIEUR
              U   E 1.4   BOUTON A CLE
              O   E 1.5   TOUCHE OUVERT INTERIEUR
)
              UN   E 1.0   FIN DE COURSE EN HAUT
              S   A 1.0   MOTEUR VERS LE HAUT
              ✱✱✱
              ǂUB   OUVRIR DE L'EXTERIEUR ET DE L'INTERIEUR
✱RAZ SORTIE MOTEUR VERS LE HAUT.
              O   E 1.0   FIN DE COURSE EN HAUT
              O   E 1.7   TOUCHE DE SECURITE STOP
              R   A 1.0   MOTEUR VERS LE HAUT
              ✱✱✱
              ǂUB   FERMER DE L'EXTERIEUR OU DE L'INTERIEUR
✱LA TOUCHE "FERME EXTERIEUR" ET LE BOUTON A CLE OU LA TOUCHE "FERME INTERIEUR"
✱METTENT EN SERVICE LE MOTEUR VERS LE BAS AVEC UN RETARD A L'ENCHLENCHEMENT DE 5 SEC.
✱LE MOTEUR VERS LE BAS TOURNE JUSQU'A CE QUE LA FIN DE COURSE EN BAS
✱SOIT ATTEINTE OU LA TOUCHE DE SECURITE STOP SOIT APPUYEE.
              U(
              U   T-FER E
              U   BOUTCLE
              O   T-FER I
)
              UN   FIN-BAS
              L   KT   005.2
              SS   RET-ENCL
              O   FIN-BAS
              O   STOP
              R   RET-ENCL
              L   RET-ENCL
              T   MW 100
              LC   RET-ENCL
              T   MW 102
              U   RET-ENCL
              =   MOT-BAS
              BE

3 - 26

C79000-B8577-C677-02

Introduction

Conditions: Le paquet éditeur LIST/compilateur par lots est chargé, le préréglage est fait, et vous avez appelé fa fonction de l'éditeur.

Déterminer le **MODE** (F8)

Avec cette fonction, vous pouvez choisir entre deux modes d'éditeur: insérer ou écraser. La console de programmation (PG) montre dans la ligne d'en-tête de l'écran quel mode vous avez choisi.

➤ Actionner **MODE** (F8) jusqu'à ce que le mode choisi soit activé.

Début de bloc Procédez de la manière suivante (Les chaînes de caractères que vous introduisez sont imprimées *en italique*, les fonctions à utiliser **en gras**.):

➤ Introduire *#PB1* pour début de bloc,

➤ appuyer deux fois sur la **touche Return**, l'ecpace sert de séparation optique pendant l'introduction,

➤ *#UB* comme titre du segment premier,

➤ appuyer deux fois sur la **touche flèche large vers la droite** afin d'arriver à la colonne COMMENTAIRE D'INSTRUCTION,

➤ *Ouvrir de l'extérieur ou intérieur*

➤ actionner la **touche Return,**

➤ appuyer une fois sur la **touche flèche large vers la gauche** afin d'arriver à la colonne ADResse,

➤ introduire ✱ comme caractère de commande pour le commentaire de segment.

Maintenant, vous pouvez insérer le premier texte de l'exemple. Vous pouvez utiliser toute la largeur de l'écran. Terminez chaque ligne avec **Return**. Vous commencez une nouvelle ligne de texte, comme déjà décrit, avec la **touche flèche large vers la gauche** et ✱, car le curseur ne saute automatiquement qu' à la colonne INSTRUCTION.

Si vous écrivez en mode insérer, faites attention à la fin de ligne! Comme vous ne pouvez insérer qu'au sein d'une ligne, le texte peut glisser en dehors de la fin de ligne et se perd ainsi.

Les touches spéciales et les touches du curseur qui ont été décrit plus haut (chap. 3.3.1.3) sont à votre disposition pour le traitement

Commande

Introduction

Conditions: Le paquet éditeur LIST/compilateur par lots est chargé, le préréglage est fait, et vous avez appelé fa fonction de l'éditeur.

Déterminer le **MODE** (F8)

Avec cette fonction, vous pouvez choisir entre deux modes d'éditeur: insérer ou écraser. La console de programmation (PG) montre dans la ligne d'en-tête de l'écran quel mode vous avez choisi.

➤ Actionner **MODE** (F8) jusqu'à ce que le mode choisi soit activé.

Début de bloc Procédez de la manière suivante (Les chaînes de caractères que vous introduisez sont imprimées *en italique*, les fonctions à utiliser **en gras**.):

3

➤ Introduire *#PB1* pour début de bloc,

➤ appuyer deux fois sur la **touche Return**, l'ecpace sert de séparation optique pendant l'introduction,

➤ *#UB* comme titre du segment premier,

➤ appuyer deux fois sur la **touche flèche large vers la droite** afin d'arriver à la colonne COMMENTAIRE D'INSTRUCTION,

➤ *Ouvrir de l'extérieur ou intérieur*

➤ actionner la **touche Return,**

➤ appuyer une fois sur la **touche flèche large vers la gauche** afin d'arriver à la colonne ADResse,

➤ introduire ✱ comme caractère de commande pour le commentaire de segment.

Maintenant, vous pouvez insérer le premier texte de l'exemple. Vous pouvez utiliser toute la largeur de l'écran. Terminez chaque ligne avec **Return**. Vous commencez une nouvelle ligne de texte, comme déjà décrit, avec la **touche flèche large vers la gauche** et ✱, car le curseur ne saute automatiquement qu' à la colonne INSTRUCTION.

Si vous écrivez en mode insérer, faites attention à la fin de ligne! Comme vous ne pouvez insérer qu'au sein d'une ligne, le texte peut glisser en dehors de la fin de ligne et se perd ainsi.

Les touches spéciales et les touches du curseur qui ont été décrit plus haut (chap. 3.3.1.3) sont à votre disposition pour le traitement

C79000-B8577-C677-02 3 - 27

de votre texte. Cependant, le caractère de commande ✱ ne peut pas être effacé via "effacer caractère" mais seulement via les fonctions **EFFACER** et **LIGNE** (voir plus bas, chap. 3.3.5.1, au paragraphe Traitement de lignes).

Opérandes absolus

Après le commentaire de segment suivent les instructions individuelles pour l'ouverture de la porte de garage, d'abord sous forme absolue avec commentaire d'instruction. Faites attention que vous devez introduire les espaces vous-mêmes.

- ➤ *U(*
- ➤ Appuyer sur la **touche Return,**
- ➤ *U*
- ➤ espace
- ➤ *E 1.2*
- ➤ Appuyer deux fois sur la **touche flèche large vers la droite** afin d'introduire le commentaire
- ➤ *touche ouvert extérieur.*
- ➤ Appuyer sur la **touche Return,**
- ➤ *U*
- ➤ espace
- ➤ *E 1.4*
- ➤ Appuyer deux fois sur la **touche flèche large vers la droite** afin d'introduire le commentaire
- ➤ *bouton à clé.*
- ➤ Appuyer sur la **touche Return,**

et cetera.

Tapez trois astérisques pour la fin de segment.

Sauvegardez l'introduction avec la fonction

- ➤ **SAUVEGARDER** (F7).

SAUVEGARDER (F7)

A l'aide de cette fonction, vous sauvegardez votre fichier source LIST sans quitter l'éditeur. Vous pouvez alors réaliser un sauvegarde intermédiaire sans problème ou interrompre l'édition pour quelque temps.

Commande

de votre texte. Cependant, le caractère de commande ✱ ne peut pas être effacé via "effacer caractère" mais seulement via les fonctions **EFFACER** et **LIGNE** (voir plus bas, chap. 3.3.5.1, au paragraphe Traitement de lignes).

Opérandes absolus

Après le commentaire de segment suivent les instructions individuelles pour l'ouverture de la porte de garage, d'abord sous forme absolue avec commentaire d'instruction. Faites attention que vous devez introduire les espaces vous-mêmes.

- ➤ *U(*
- ➤ Appuyer sur la **touche Return,**
- ➤ *U*
- ➤ espace
- ➤ *E 1.2*
- ➤ Appuyer deux fois sur la **touche flèche large vers la droite** afin d'introduire le commentaire
- ➤ *touche ouvert extérieur.*
- ➤ Appuyer sur la **touche Return,**
- ➤ *U*
- ➤ espace
- ➤ *E 1.4*
- ➤ Appuyer deux fois sur la **touche flèche large vers la droite** afin d'introduire le commentaire
- ➤ *bouton à clé.*
- ➤ Appuyer sur la **touche Return,**

et cetera.

Tapez trois astérisques pour la fin de segment.

Sauvegardez l'introduction avec la fonction

- ➤ **SAUVEGARDER** (F7).

SAUVEGARDER (F7)

A l'aide de cette fonction, vous sauvegardez votre fichier source LIST sans quitter l'éditeur. Vous pouvez alors réaliser un sauvegarde intermédiaire sans problème ou interrompre l'édition pour quelque temps.

3 - 28

C79000-B8577-C677-02

Au contraire du paquet CONT-LOG-LIST où vous devez toujours quitter l'introduction pour la sauvegarde et aller à la sortie pour continuer le travail.

Opérandes symboliques

Ici suivent les instructions pour la fermeture de la porte au segment 3, cette fois programmées sous forme symbolique.

Introduisez le titre et le commentaire comme il est décrit plus haut. Avec les opérandes symboliques, vous procédez de la maniére suivante:

➤ *U*

➤ Appuyer une fois sur la **touche flèche large vers la droite** ou '-' (trait d'union), le curseur se trouve maintenant dans la colonne Mnémonique.

➤ *T-FER E* (sans trait d'union introduisant!)

➤ Appuyez sur la **touche Return**

➤ *U*

➤ Appuyez sur la **touche flèche large vers la droite** pour introduire le symbole

➤ *boutclé*,

➤ appuyez sur la **touche Return**.

Si vous avez atteint la fin de bloc BE, mémorisez vos données avec

➤ **SAUVEGARDER** (F7).

Le paragraphe suivant vous montre comment vous employez les fonctions individuelles en traitant une liste d'instruction. Comme dans l'éditeur symbolique, le traitement ou la correction d'un fichier se fait également avec la fonction d'éditeur.

Commande

Au contraire du paquet CONT-LOG-LIST où vous devez toujours quitter l'introduction pour la sauvegarde et aller à la sortie pour continuer le travail.

Opérandes symboliques

Ici suivent les instructions pour la fermeture de la porte au segment 3, cette fois programmées sous forme symbolique.

Introduisez le titre et le commentaire comme il est décrit plus haut. Avec les opérandes symboliques, vous procédez de la maniére suivante:

➤ *U*

➤ Appuyer une fois sur la **touche flèche large vers la droite** ou '-' (trait d'union), le curseur se trouve maintenant dans la colonne Mnémonique.

➤ *T-FER E* (sans trait d'union introduisant!)

➤ Appuyez sur la **touche Return**

➤ *U*

➤ Appuyez sur la **touche flèche large vers la droite** pour introduire le symbole

➤ *boutclé*,

➤ appuyez sur la **touche Return**.

Si vous avez atteint la fin de bloc BE, mémorisez vos données avec

➤ **SAUVEGARDER** (F7).

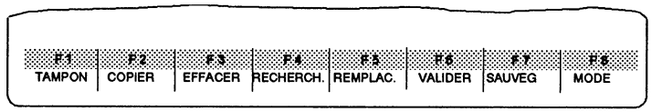
Le paragraphe suivant vous montre comment vous employez les fonctions individuelles en traitant une liste d'instruction. Comme dans l'éditeur symbolique, le traitement ou la correction d'un fichier se fait également avec la fonction d'éditeur.

3

C79000-B8577-C677-02

3 - 29

3.3.5 Utilisation de la barre des touches de fonction EDITEUR



Pour l'explication de l'utilisation de la fonction d'éditeur nous voulons mettre en relief la combinaison de **TAMPON, COPIER** et **EFFACER.** Nous vous montrons comment vous pouvez mémoriser des parties de votre programme sur des fichiers pour le traitement ultérieur et le retraitement et comment vous pouvez allouer des chaînes de caractère à des touches de fonction.

Le facteur de répétition est un outil secourable: Après avoir appelé une fonction, vous introduisez un nombre via le clavier de votre console de programmation. La fonction succédante sera exécutée avec ce facteur, p. ex. copier une ligne 7 fois.

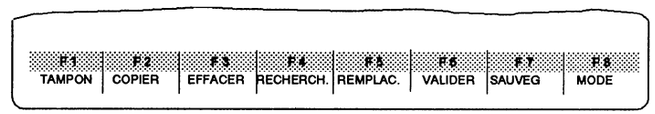
La fonction que vous avez choisie est affichée dans la ligne d'entête au-dessus du COMMENTAIRE D'INSTRUCTION. Il faut toujours quitter une fonction avec **RETOUR** (F8) avant de continuer avec le texte.

Si vous voulez quitter un processus au sein d'une fonction, utilisez la **touche d'abandon** et/ou la touche F8 **RETOUR**.

ATTENTION
Quand vous travaillez avec la touche d'abandon vous perdez peut-être des données! Quand vous avez p. ex. corrigé un fichier et ensuite appuyez sur la touche d'abandon, toutes les modifications seront perdues.

Commande

3.3.5 Utilisation de la barre des touches de fonction EDITEUR



Pour l'explication de l'utilisation de la fonction d'éditeur nous voulons mettre en relief la combinaison de **TAMPON, COPIER** et **EFFACER.** Nous vous montrons comment vous pouvez mémoriser des parties de votre programme sur des fichiers pour le traitement ultérieur et le retraitement et comment vous pouvez allouer des chaînes de caractère à des touches de fonction.

Le facteur de répétition est un outil secourable: Après avoir appelé une fonction, vous introduisez un nombre via le clavier de votre console de programmation. La fonction succédante sera exécutée avec ce facteur, p. ex. copier une ligne 7 fois.

La fonction que vous avez choisie est affichée dans la ligne d'entête au-dessus du COMMENTAIRE D'INSTRUCTION. Il faut toujours quitter une fonction avec **RETOUR** (F8) avant de continuer avec le texte.

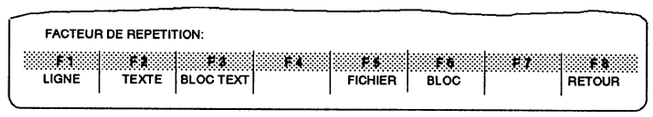
Si vous voulez quitter un processus au sein d'une fonction, utilisez la **touche d'abandon** et/ou la touche F8 **RETOUR**.

ATTENTION
Quand vous travaillez avec la touche d'abandon vous perdez peut-être des données! Quand vous avez p. ex. corrigé un fichier et ensuite appuyez sur la touche d'abandon, toutes les modifications seront perdues.

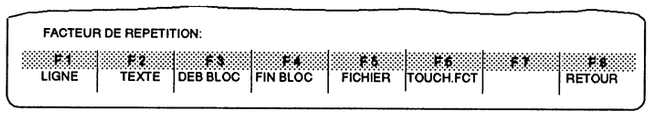
Les fonctions
TAMPON,
COPIER *et*
EFFACER

TAMPON Avec cette fonction, il est possible de retenir des parties de texte. Des chaînes de caractères quelconques (jusqu'à 40 caractères au maximum), des lignes individuelles et des blocs de caractère (500 lignes au maximum) seront tous écrits dans un propre tampon et peuvent être copiés à des endroits quelconques. Le texte qui est retenu dans un tampon peut aussi être mémorisé dans un fichier séquentiel.

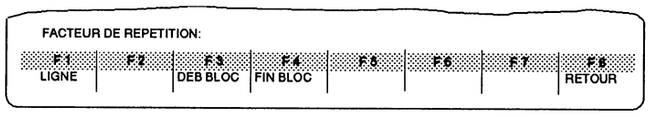
Des touches de fonction seront également allouées à des chaînes de caractères via la fonction de tampon (40 caractères au maximum).



COPIER Le texte retenu dans le tampon ou le fichier séquentiel (fichier tampon, fichier source) sera inséré via cette fonction à la position du curseur. Vous pouvez même copier un fichier source LIST entier ou un bloc à la position du curseur.



EFFACER Le menu effacer vous offre d'effacer des lignes individuelles et des blocs de texte marqués. Le texte effacé sera écrit dans le tampon pour des raisons de sécurité. Ainsi, du texte retenu préalablement sera perdu.



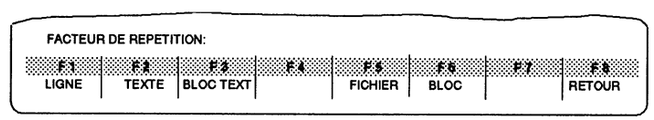
Commande

Les fonctions
TAMPON,
COPIER *et*
EFFACER

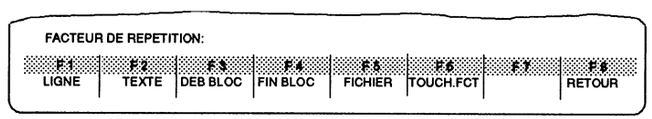
TAMPON Avec cette fonction, il est possible de retenir des parties de texte. Des chaînes de caractères quelconques (jusqu'à 40 caractères au maximum), des lignes individuelles et des blocs de caractère (500 lignes au maximum) seront tous écrits dans un propre tampon et peuvent être copiés à des endroits quelconques. Le texte qui est retenu dans un tampon peut aussi être mémorisé dans un fichier séquentiel.

Des touches de fonction seront également allouées à des chaînes de caractères via la fonction de tampon (40 caractères au maximum).

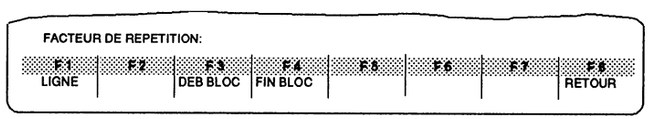
3



COPIER Le texte retenu dans le tampon ou le fichier séquentiel (fichier tampon, fichier source) sera inséré via cette fonction à la position du curseur. Vous pouvez même copier un fichier source LIST entier ou un bloc à la position du curseur.



EFFACER Le menu effacer vous offre d'effacer des lignes individuelles et des blocs de texte marqués. Le texte effacé sera écrit dans le tampon pour des raisons de sécurité. Ainsi, du texte retenu préalablement sera perdu.



Traitement d'une ligne

Exemple: Le titre d'un premier segment doit être identique auquel du deuxième.

1. Tampon

- ➤ Placer le curseur sur cette ligne,
- ➤ appeler →**TAMPON** (F1),
- ➤ actionner **LIGNE** (F1), cette ligne se trouve maintenant dans le tampon,
- ➤ quitter la fonction de tampon avec **RETOUR** (F8).

2. Copier

- ➤ Au deuxième segment, placer le curseur sur l'ancien titre,
- ➤ appeler **COPIER** (F2) (Faites attention au mode d'écriture! Voir plus bas, Remarque),
- ➤ actionner **LIGNE** (F1), le nouveau titre sera inséré,
- ➤ quitter la fonction copier avec **RETOUR** (F8).

3. Effacer

- ➤ Placer le curseur sur l'ancien titre,
- ➤ appeler **EFFACER** (F3),
- ➤ actionner **LIGNE** (F1), l'ancien titre est effacé,
- ➤ quitter la fonction copier avec **RETOUR** (F8).

4. Décaler

- ➤ Placer le curseur sur la ligne qui doit être décalée,
- ➤ appeler **EFFACER** (F3),
- ➤ actionner **LIGNE** (F1), le texte de cette ligne se trouve maintenant dans le tampon.
- ➤ Placer le curseur à l'endroit où il faut décaler la ligne,
- ➤ appeler **COPIER** (F2),
- ➤ actionner **LIGNE** (F1), la ligne sera insérée.
- ➤ Quitter la fonction copier avec **RETOUR** (F8).

Commande

Traitement d'une ligne

Exemple: Le titre d'un premier segment doit être identique auquel du deuxième.

1. Tampon

- ➤ Placer le curseur sur cette ligne,
- ➤ appeler →**TAMPON** (F1),
- ➤ actionner **LIGNE** (F1), cette ligne se trouve maintenant dans le tampon,
- ➤ quitter la fonction de tampon avec **RETOUR** (F8).

2. Copier

- ➤ Au deuxième segment, placer le curseur sur l'ancien titre,
- ➤ appeler **COPIER** (F2) (Faites attention au mode d'écriture! Voir plus bas, Remarque),
- ➤ actionner **LIGNE** (F1), le nouveau titre sera inséré,
- ➤ quitter la fonction copier avec **RETOUR** (F8).

3. Effacer

- ➤ Placer le curseur sur l'ancien titre,
- ➤ appeler **EFFACER** (F3),
- ➤ actionner **LIGNE** (F1), l'ancien titre est effacé,
- ➤ quitter la fonction copier avec **RETOUR** (F8).

4. Décaler

- ➤ Placer le curseur sur la ligne qui doit être décalée,
- ➤ appeler **EFFACER** (F3),
- ➤ actionner **LIGNE** (F1), le texte de cette ligne se trouve maintenant dans le tampon.
- ➤ Placer le curseur à l'endroit où il faut décaler la ligne,
- ➤ appeler **COPIER** (F2),
- ➤ actionner **LIGNE** (F1), la ligne sera insérée.
- ➤ Quitter la fonction copier avec **RETOUR** (F8).

3 - 32

C79000-B8577-C677-02

Remarque
Avec **TAMPON**, une ligne, un texte et un bloc seront tous copiés dans un propre tampon. Dans le tampon, la ligne copiée ne sera écrasée que par une ligne copiée à nouveau, un texte copié par un texte copié à nouveau et un bloc copié par un bloc copié à nouveau. Ainsi, vous avez la possibilité de retenir trois séquences de caractères parallèlement.
En copiant, positionner le curseur avant l'appel de la fonction **COPIER**, car les fonctions seront exécutées immédiatement à la position du curseur. Faites attention au mode d'écriture: en mode insérer, du texte retenu sera inséré, en mode écraser, le texte sera écrasé à partir du curseur par le texte retenu. Il faut donc qu'il y ait assez de place, si vous voulez travaillez en mode écraser.
Des caractères de commande ne peuvent être effacés que par **EFFACER LIGNE**.

Traitement d'un bloc (500 lignes au max.)

1. Tampon et copier p. ex. un commentaire de segment

- ➤ Appeler **TAMPON** (F1),
- ➤ placer le curseur au début d'un commentaire de segment,
- ➤ marquer avec **DEBUTBLOC** (F3),
- ➤ placer le curseur à la fin du commentaire,
- ➤ marquer celle-ci avec **FINBLOC** (F4),
- ➤ quitter la fonction de tampon avec **RETOUR** (F8).
- ➤ appeler **COPIER** (F2),
- ➤ positionner le curseur p. ex. à la fin du fichier,
- ➤ actionner **BLOC** (F3), le commentaire sera inséré ici,
- ➤ quitter la fonction copier avec **RETOUR** (F8).

2. Retenir sur fichier mémoriser p. ex. le retard à l'enclenchement dans le fichier TIMER@.

- ➤ appeler **TAMPON** (F1),
- ➤ positionner le curseur à la ligne L KT 005.2,
- ➤ actionner **DEButBLOC** (F3),
- ➤ déplacer le curseur vers le bas jusqu'à la fin du Timer, ligne U, T1,

Commande

Remarque
Avec **TAMPON**, une ligne, un texte et un bloc seront tous copiés dans un propre tampon. Dans le tampon, la ligne copiée ne sera écrasée que par une ligne copiée à nouveau, un texte copié par un texte copié à nouveau et un bloc copié par un bloc copié à nouveau. Ainsi, vous avez la possibilité de retenir trois séquences de caractères parallèlement.
En copiant, positionner le curseur avant l'appel de la fonction **COPIER**, car les fonctions seront exécutées immédiatement à la position du curseur. Faites attention au mode d'écriture: en mode insérer, du texte retenu sera inséré, en mode écraser, le texte sera écrasé à partir du curseur par le texte retenu. Il faut donc qu'il y ait assez de place, si vous voulez travaillez en mode écraser.
Des caractères de commande ne peuvent être effacés que par **EFFACER LIGNE**.

3

Traitement d'un bloc (500 lignes au max.)

1. Tampon et copier p. ex. un commentaire de segment

- ➤ Appeler **TAMPON** (F1),
- ➤ placer le curseur au début d'un commentaire de segment,
- ➤ marquer avec **DEBUTBLOC** (F3),
- ➤ placer le curseur à la fin du commentaire,
- ➤ marquer celle-ci avec **FINBLOC** (F4),
- ➤ quitter la fonction de tampon avec **RETOUR** (F8).
- ➤ appeler **COPIER** (F2),
- ➤ positionner le curseur p. ex. à la fin du fichier,
- ➤ actionner **BLOC** (F3), le commentaire sera inséré ici,
- ➤ quitter la fonction copier avec **RETOUR** (F8).

2. Retenir sur fichier mémoriser p. ex. le retard à l'enclenchement dans le fichier TIMER@.

- ➤ appeler **TAMPON** (F1),
- ➤ positionner le curseur à la ligne L KT 005.2,
- ➤ actionner **DEButBLOC** (F3),
- ➤ déplacer le curseur vers le bas jusqu'à la fin du Timer, ligne U, T1,

C79000-B8577-C677-02 3 - 33

- ➤ actionner **FINBLOC** (F4), ce bloc est maintenant dupliqué dans le tampon.
- ➤ appeler **FICHIER** (F5) et
- ➤ remplir avec *TIMER@*,
- ➤ actionner **touche Return** et
- ➤ **VALIDER** (F6). Le retard à l'enclenchement se trouve maintenant dans le fichier TIMER@A0.SEQ
- ➤ Quitter la fonction de tampon avec **RETOUR** (F8).
- ➤ Placer le curseur après la fin de bloc (BE),
- ➤ appeler **COPIER** (F2),
- ➤ appeler **FICHIER** (F5) et introduire le fichier qu'on veut copier. Nous prenons le fichier affiché B:TIMER@A0.SEQ,
- ➤ Copier est démarré avec **VALIDER** (F6) ou avec la **touche de validation**.
  Le retard à l'enclenchement sera inséré.
- ➤ Quitter la fonction copier avec **RETOUR** (F8).

3. Effacer On veut effacer les parties insérées.

- ➤ appeler **EFFACER** (F3). Marquer le retard à l'enclenchement
- ➤ avec **DEBut**BLOC (F3) et
- ➤ **FINBLOC** (F4), le texte est effacé.
- ➤ **RETOUR**nez (F8).

Vous pouvez également effacer le commentaire de segment ligne par ligne avec facteur de répétition p. ex.

- ➤ Positionner le curseur,
- ➤ appeler **EFFACER** (F3),
- ➤ introduire un facteur (p. ex. 4) via le clavier
- ➤ actionner **LIGNE** (F1),

Les (quatre) lignes sont effacées.

- ➤ **RETOUR**nez (F8).

Commande

- ➤ actionner **FINBLOC** (F4), ce bloc est maintenant dupliqué dans le tampon.
- ➤ appeler **FICHIER** (F5) et
- ➤ remplir avec *TIMER@*,
- ➤ actionner **touche Return** et
- ➤ **VALIDER** (F6). Le retard à l'enclenchement se trouve maintenant dans le fichier TIMER@A0.SEQ
- ➤ Quitter la fonction de tampon avec **RETOUR** (F8).
- ➤ Placer le curseur après la fin de bloc (BE),
- ➤ appeler **COPIER** (F2),
- ➤ appeler **FICHIER** (F5) et introduire le fichier qu'on veut copier. Nous prenons le fichier affiché B:TIMER@A0.SEQ,
- ➤ Copier est démarré avec **VALIDER** (F6) ou avec la **touche de validation**.
  Le retard à l'enclenchement sera inséré.
- ➤ Quitter la fonction copier avec **RETOUR** (F8).

3. Effacer On veut effacer les parties insérées.

- ➤ appeler **EFFACER** (F3). Marquer le retard à l'enclenchement
- ➤ avec **DEBut**BLOC (F3) et
- ➤ **FINBLOC** (F4), le texte est effacé.
- ➤ **RETOUR**nez (F8).

Vous pouvez également effacer le commentaire de segment ligne par ligne avec facteur de répétition p. ex.

- ➤ Positionner le curseur,
- ➤ appeler **EFFACER** (F3),
- ➤ introduire un facteur (p. ex. 4) via le clavier
- ➤ actionner **LIGNE** (F1),

Les (quatre) lignes sont effacées.

- ➤ **RETOUR**nez (F8).

3 - 34 C79000-B8577-C677-02

4. Décaler Effacer le bloc qu'on veut décaler avec

- ➤ **EFFACER** (F3),
- ➤ **DEButBLOC** (F3) et
- ➤ **FINBLOC** (F4) et l'écrire dans le tampon.
- ➤ **RETOURnez** (F8).
- ➤ Placer le curseur sur la nouvelle position,
- ➤ appeler **COPIER** (F2) et
- ➤ actionner **BLOC** (F3). Le texte est décalé.
- ➤ **RETOURnez** (F8).

Traitement d'une chaîne de caractères ou d'un texte (40 caractères au max.)

Si vous voulez p. ex. insérer ou modifier un texte ou une instruction (plus tard) à plusieurs endroits, ou s'il y a un commentaire d'instruction qui revient fréquemment et que vous ne voulez pas écrire toujours à nouveau, vous pouvez placer de tels textes dans le tampon.

1. Tampon

- ➤ appeler →**TAMPON** (F1),
- ➤ actionner **TEXTE** (F2),
- ➤ introduire du texte entre les deux-points, p. ex. *UE 2.0*.
- ➤ Actionner **VALIDER** (F6). Ce texte se trouve maintenant dans le tampon.
- ➤ Quitter la fonction de tampon avec **RETOUR**.

Vous pouvez effacer du texte entre des deux-points via la touche **+1**.

2. Copier

- ➤ Positionner le curseur,
- ➤ appeler **COPIER** (F2) (Faites attention au mode d'écriture! Voir plus haut, Remarque),
- ➤ placer le curseur à l'endroit qu'on veut traiter,
- ➤ actionner **TEXTE** (F2) et le texte du tampon sera inséré.
- ➤ Quitter la fonction copier avec **RETOUR**.

Commande

4. Décaler Effacer le bloc qu'on veut décaler avec

➤ EFFACER (F3),
➤ DEButBLOC (F3) et
➤ FINBLOC (F4) et l'écrire dans le tampon.
➤ RETOURnez (F8).
➤ Placer le curseur sur la nouvelle position,
➤ appeler COPIER (F2) et
➤ actionner BLOC (F3). Le texte est décalé.
➤ RETOURnez (F8).

Traitement d'une chaîne de caractères ou d'un texte (40 caractères au max.)

Si vous voulez p. ex. insérer ou modifier un texte ou une instruction (plus tard) à plusieurs endroits, ou s'il y a un commentaire d'instruction qui revient fréquemment et que vous ne voulez pas écrire toujours à nouveau, vous pouvez placer de tels textes dans le tampon.

3

1. Tampon
➤ appeler →TAMPON (F1),
➤ actionner TEXTE (F2),
➤ introduire du texte entre les deux-points, p. ex. UE 2.0.
➤ Actionner VALIDER (F6). Ce texte se trouve maintenant dans le tampon.
➤ Quitter la fonction de tampon avec RETOUR.

Vous pouvez effacer du texte entre des deux-points via la touche +1.

2. Copier
➤ Positionner le curseur,
➤ appeler COPIER (F2) (Faites attention au mode d'écriture! Voir plus haut, Remarque),
➤ placer le curseur à l'endroit qu'on veut traiter,
➤ actionner TEXTE (F2) et le texte du tampon sera inséré.
➤ Quitter la fonction copier avec RETOUR.

C79000-B8577-C677-02

3 - 35

Allocation des touches de fonction (avec 40 caractères au max.)

P. ex. si vous voulez allouer la fin du segment, une chaîne de caractère qui revient fréquemment, à une touche de fonction.

➤ Appeler →**TAMPON** (F1),

➤ avec **TOUCHESFCT** (F6) éditer sur écran la liste des touches de fonction. Le curseur clignote à la ligne f1.

➤ Introduire *** entre les deux-points,

➤ **VALIDER** avec F6.

Vous obtenez cette chaîne de caractère à chaque endroit de votre fichier, si vous appuyez sur les touches Shift et F1 des appareils PG 695, PG 695II et PG 750, la touche "Ouvrir" du PG 685.

Si vous insérez p. ex. une fin de segment après une autre fin de segment:

➤ Placer le curseur sur ***,

➤ appuyer sur les touches **SHIFT** et **F1**.

Annulez maintenant toutes ces modifications:

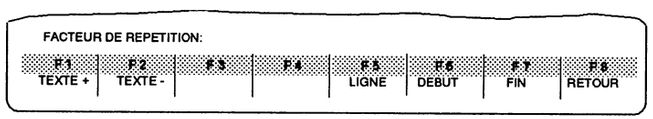
➤ appuyer sur la **touche d'abandon** et

➤ confirmer avec la **touche de validation**. Dans la SELECTION DE FONCTION affichée

➤ appeler **EDITER** (F1) à nouveau.

La fonction RECHERCHE

Via **RECHERCHE**, vous pouvez sauter au début et à la fin de votre fichier et à des lignes individuelles. La ligne 0 n'est pas admissible. De plus, vous pouvez chercher des chaînes de caractère quelconques avec 20 symboles (mots et nombres) au maximum qui se trouvent au sein d'un champ. Texte+ recherche de telles chaînes de caractère en avant à partir de la position du curseur, texte- cherche en arrière. Avec **RECHERCHE**, vous pouvez donc parvenir facilement et aisément à chaque endroit de votre fichier.



Commande

Allocation des touches de fonction (avec 40 caractères au max.)

P. ex. si vous voulez allouer la fin du segment, une chaîne de caractère qui revient fréquemment, à une touche de fonction.

➤ Appeler →**TAMPON** (F1),

➤ avec **TOUCHESFCT** (F6) éditer sur écran la liste des touches de fonction. Le curseur clignote à la ligne f1.

➤ Introduire *** entre les deux-points,

➤ **VALIDER** avec F6.

Vous obtenez cette chaîne de caractère à chaque endroit de votre fichier, si vous appuyez sur les touches Shift et F1 des appareils PG 695, PG 695II et PG 750, la touche "Ouvrir" du PG 685.

Si vous insérez p. ex. une fin de segment après une autre fin de segment:

➤ Placer le curseur sur ***,

➤ appuyer sur les touches **SHIFT** et **F1**.

Annulez maintenant toutes ces modifications:

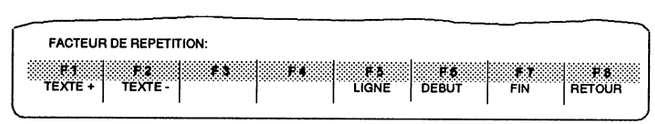
➤ appuyer sur la **touche d'abandon** et

➤ confirmer avec la **touche de validation**. Dans la SELECTION DE FONCTION affichée

➤ appeler **EDITER** (F1) à nouveau.

La fonction RECHERCHE

Via **RECHERCHE**, vous pouvez sauter au début et à la fin de votre fichier et à des lignes individuelles. La ligne 0 n'est pas admissible. De plus, vous pouvez chercher des chaînes de caractère quelconques avec 20 symboles (mots et nombres) au maximum qui se trouvent au sein d'un champ. Texte+ recherche de telles chaînes de caractère en avant à partir de la position du curseur, texte- cherche en arrière. Avec **RECHERCHE**, vous pouvez donc parvenir facilement et aisément à chaque endroit de votre fichier.



3 - 36

C79000-B8577-C677-02

Il faut veiller à ce que le texte à rechercher doive présenter exactement l'écriture en majuscule/en minuscule de la suite de caractères recherchée. Ainsi vous devez introduire en lettres majuscules les textes qui se trouvent dans les champs "**ADR**" et "**instruction**".

Vous sautez p. ex. à la fin avec

➤ **FIN** (F7), le curseur se trouve au-dessous du BE;

au début avec

➤ **DEBUT** (F6),le curseur se trouve sur l'indication du type d'AG (si exist).

à une ligne nouvelle avec

➤ **LIGNE** (F5),

➤ vous introduisez le nombre, p. ex. *12*,

➤ et actionnez **VALIDER**. Le curseur clignote à la ligne 12.

Vous cherchez un certain terme, une adresse, un opérande en avant à partir de votre position du curseur avec

➤ **TEXTE+** (F1),

➤ vous introduisez la chaîne de caractère, p. ex. *FIN-AVANT,*

➤ et actionnez **VALIDER** (F6). Vous trouvez le curseur à la fin de cette chaîne de caractère qui est la plus proche du curseur en avant.

en arrière de votre position avec

➤ **TEXTE-** (F2)

➤ Ici, vous introduisez aussi la chaîne de caractère, p. ex. *touche ouvert*

➤ et actionnez **VALIDER** (F6).

Maintenant, le curseur saute à l'endroit qui est le plus proche en arrière.

Commande

Il faut veiller à ce que le texte à rechercher doive présenter exactement l'écriture en majuscule/en minuscule de la suite de caractères recherchée. Ainsi vous devez introduire en lettres majuscules les textes qui se trouvent dans les champs "**ADR**" et "**instruction**".

Vous sautez p. ex. à la fin avec

➤ **FIN** (F7), le curseur se trouve au-dessous du BE;

au début avec

➤ **DEBUT** (F6),le curseur se trouve sur l'indication du type d'AG (si exist).

à une ligne nouvelle avec

➤ **LIGNE** (F5),

➤ vous introduisez le nombre, p. ex. *12*,

➤ et actionnez **VALIDER**. Le curseur clignote à la ligne 12.

Vous cherchez un certain terme, une adresse, un opérande en avant à partir de votre position du curseur avec

➤ **TEXTE+** (F1),

➤ vous introduisez la chaîne de caractère, p. ex. *FIN-AVANT,*

➤ et actionnez **VALIDER** (F6). Vous trouvez le curseur à la fin de cette chaîne de caractère qui est la plus proche du curseur en avant.

en arrière de votre position avec

➤ **TEXTE-** (F2)

➤ Ici, vous introduisez aussi la chaîne de caractère, p. ex. *touche ouvert*

➤ et actionnez **VALIDER** (F6).

Maintenant, le curseur saute à l'endroit qui est le plus proche en arrière.

3

C79000-B8577-C677-02 3 - 37

Si vous voulez continuer la recherche du terme introduit, réactivez la fonction choisie:

➤ appeler p. ex. **TEXTE-** encore une fois et

➤ **VALIDER**; le curseur saute encore une fois à la chaîne de caractère du terme introduit qui est la plus proche dans la direction choisie.

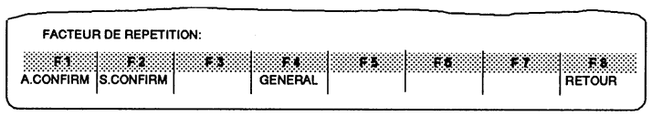
A la fin de la recherche vous terminez par

➤ **RETOUR** (F8).

La fonction REMPLACER

Vous pouvez remplacer des chaînes de caractère quelconques avec 20 caractères au max. (mots et nombres) qui se trouvent dans les colonnes ADR, MNEMONIQUE et COMMENTAIRE D'INSTRUCTION par d'autres chaînes de caractère. Vous choisissez entre un remplacement sélectif avec ou sans confirmation et un remplacement général. Dans le cas du remplacement sélectif, la recherche ne s'effectue que vers l'aval, à partir de la position actuelle du curseur, positionnez donc le curseur au moins une ligne plus haut.

Faites attention que le texte inséré ait exactement la même orthographie que la chaîne de caractères à chercher. Textes qui se trouvent dans les champs "ADR" et "INSTRUCT." sont donc obligatoirement à insérer en majuscules.



REMPLACER vous offre la possibilité de corriger votre fichier rapidement, si p. ex. une instruction ou un symbole doit être remplacé dans le fichier entier. Faites attention à l'identité absolue de la chaîne de caractère recherchée. Des espaces sont également important.

➤ Appeler **REMPLACER** (F5),

➤ appuyer sur **GENERAL** (F4),

➤ introduire la chaîne de caractère à remplacer, p. ex. *RET-ENCL*

➤ appuyer sur la **VALIDER** (F6),

Commande

Si vous voulez continuer la recherche du terme introduit, réactivez la fonction choisie:

➤ appeler p. ex. **TEXTE-** encore une fois et

➤ **VALIDER**; le curseur saute encore une fois à la chaîne de caractère du terme introduit qui est la plus proche dans la direction choisie.

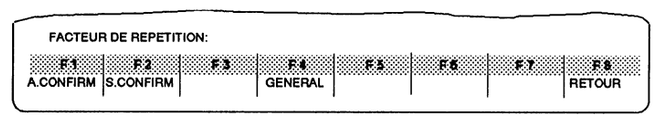
A la fin de la recherche vous terminez par

➤ **RETOUR** (F8).

La fonction REMPLACER

Vous pouvez remplacer des chaînes de caractère quelconques avec 20 caractères au max. (mots et nombres) qui se trouvent dans les colonnes ADR, MNEMONIQUE et COMMENTAIRE D'INSTRUCTION par d'autres chaînes de caractère. Vous choisissez entre un remplacement sélectif avec ou sans confirmation et un remplacement général. Dans le cas du remplacement sélectif, la recherche ne s'effectue que vers l'aval, à partir de la position actuelle du curseur, positionnez donc le curseur au moins une ligne plus haut.

Faites attention que le texte inséré ait exactement la même orthographie que la chaîne de caractères à chercher. Textes qui se trouvent dans les champs "ADR" et "INSTRUCT." sont donc obligatoirement à insérer en majuscules.



REMPLACER vous offre la possibilité de corriger votre fichier rapidement, si p. ex. une instruction ou un symbole doit être remplacé dans le fichier entier. Faites attention à l'identité absolue de la chaîne de caractère recherchée. Des espaces sont également important.

➤ Appeler **REMPLACER** (F5),

➤ appuyer sur **GENERAL** (F4),

➤ introduire la chaîne de caractère à remplacer, p. ex. *RET-ENCL*

➤ appuyer sur la **VALIDER** (F6),

➤ introduire la chaîne de caractère nouvelle, p. ex. *SS-TIMER*

➤ et **VALIDER** (F6). Maintenant, tous les symboles RET-ENCL seront remplacés par SS-TIMER.

remplacer une chaîne de caractère

Si une chaîne de caractère ne doit être remplacée qu'une fois, vous avez une possibilité de contrôle en sélectant **A**(vec)**.CONFIRM**; **S**(ans)**.CONFIRM** effectue ce remplacement immédiatement. En ce cas, le curseur doit être placé au-dessus de la chaîne de caractère à remplacer, car au cours du remplacement sélectif, la recherche ne s'effectue que vers l'aval, à partir de la position actuelle du curseur.

➤ Appeler **REMPLACER** (F5),

➤ appuyer sur **A.CONFIRM** (F1),

➤ introduire la chaîne de caractère à remplacer, p. ex. *EXTERIEUR*

➤ appuyer sur la **VALIDER** (F6),

➤ introduire la chaîne de caractère nouvelle, p. ex. *ZZZZZZ*,

➤ et **VALIDER**. Vous avez maintenant la possibilité de contrôle **OUI** (F1) ou **NON** (F3),

➤ appuyer sur **OUI** (F1). La chaîne de caractère "extérieur" la plus proche du curseur est remplacée par "ZZZZZZ".

Ou bien vous recevez l'indication "Pas trouvé" à la ligne de message. En ce cas, il n'y avait plus de "extérieur" au-dessous du curseur. Placez le curseur au début de

votre fichier et répétez le remplacement avec **REMPLACER** (F5), **A.CONFIRM** (F1), deux fois la **touche de validation** et **OUI** (F1). **S.CONFIRM** fonctionne de manière analogue mais vous n'avez pas de contrôle.

facteur de répétition

Vous pouvez combiner le remplacement sélectif avec le facteur de répétition. Au cours de ce remplacement multiple, le curseur saute également à des chaînes de caractère dans des commentaires. Elles comptent pour l'indication du facteur de répétition mais ne seront pas remplacées même pas si vous appelez **A.CONFIRM, OUI**. Vous terminez le remplacement prématurément avec la **touche d'abandon**, les chaînes de caractère qui sont déjà remplacées seront retenues.

Commande

➤ introduire la chaîne de caractère nouvelle, p. ex. *SS-TIMER*

➤ et **VALIDER** (F6). Maintenant, tous les symboles RET-ENCL seront remplacés par SS-TIMER.

remplacer une chaîne de caractère

Si une chaîne de caractère ne doit être remplacée qu'une fois, vous avez une possibilité de contrôle en sélectant **A**(vec)**.CONFIRM**; **S**(ans)**.CONFIRM** effectue ce remplacement immédiatement. En ce cas, le curseur doit être placé au-dessus de la chaîne de caractère à remplacer, car au cours du remplacement sélectif, la recherche ne s'effectue que vers l'aval, à partir de la position actuelle du curseur.

➤ Appeler **REMPLACER** (F5),

➤ appuyer sur **A.CONFIRM** (F1),

➤ introduire la chaîne de caractère à remplacer, p. ex. *EXTERIEUR*

➤ appuyer sur la **VALIDER** (F6),

➤ introduire la chaîne de caractère nouvelle, p. ex. *ZZZZZZ*,

➤ et **VALIDER**. Vous avez maintenant la possibilité de contrôle **OUI** (F1) ou **NON** (F3),

➤ appuyer sur **OUI** (F1). La chaîne de caractère "extérieur" la plus proche du curseur est remplacée par "ZZZZZZ".

Ou bien vous recevez l'indication "Pas trouvé" à la ligne de message. En ce cas, il n'y avait plus de "extérieur" au-dessous du curseur. Placez le curseur au début de

votre fichier et répétez le remplacement avec **REMPLACER** (F5), **A.CONFIRM** (F1), deux fois la **touche de validation** et **OUI** (F1). **S.CONFIRM** fonctionne de manière analogue mais vous n'avez pas de contrôle.

facteur de répétition

Vous pouvez combiner le remplacement sélectif avec le facteur de répétition. Au cours de ce remplacement multiple, le curseur saute également à des chaînes de caractère dans des commentaires. Elles comptent pour l'indication du facteur de répétition mais ne seront pas remplacées même pas si vous appelez **A.CONFIRM, OUI**. Vous terminez le remplacement prématurément avec la **touche d'abandon**, les chaînes de caractère qui sont déjà remplacées seront retenues.

➤ Quittez la fonction **REMPLACER** avec **RETOUR** (F8) et annulez toutes les modifications avec la

➤ **touche d'abandon,** confirmez avec la

➤ **touche de validation**. Vous êtes encore une fois dans la SELECTION DE FONCTION.

Resortez maintenant le fichier original:

➤ **EDITER** (F1)

Les fonctions de mémorisation SAUVEGARDER et VALIDER

Vous connaissez déjà la fonction **SAUVEGARDER**. Avec elle, vous mémorisez et pouvez interrompre et recommencer votre séance de travail sans sortir le fichier à nouveau.

La fonction **VALIDER** (F6) ou la **touche de validation** mémorise le fichier, génère automatiquement le fichier intermédiaire respective et termine la séance de travail. Au cours de la conversion, la liste d'instructions générée sera contrôlée. S'il y a plus d'une erreur, une liste d'erreurs sera générée. En cas d'une seule erreur, l'endroit incorrect sera affiché sur l'écran, et vous pouvez corriger dans la ligne au bas de la page à l'aide du message d'erreur.

Quant à la liste d'erreurs, lisez au chapitre 3.4, s.v.p.

➤ Mémoriser le fichier avec **VALIDER** (F6): Il sera conversé, l'édition terminée. La SELECTION DE FONCTION est affichée.

Commande

➤ Quittez la fonction **REMPLACER** avec **RETOUR** (F8) et annulez toutes les modifications avec la

➤ **touche d'abandon,** confirmez avec la

➤ **touche de validation**. Vous êtes encore une fois dans la SELECTION DE FONCTION.

Resortez maintenant le fichier original:

➤ **EDITER** (F1)

Les fonctions de mémorisation SAUVEGARDER et VALIDER

Vous connaissez déjà la fonction **SAUVEGARDER**. Avec elle, vous mémorisez et pouvez interrompre et recommencer votre séance de travail sans sortir le fichier à nouveau.

La fonction **VALIDER** (F6) ou la **touche de validation** mémorise le fichier, génère automatiquement le fichier intermédiaire respective et termine la séance de travail. Au cours de la conversion, la liste d'instructions générée sera contrôlée. S'il y a plus d'une erreur, une liste d'erreurs sera générée. En cas d'une seule erreur, l'endroit incorrect sera affiché sur l'écran, et vous pouvez corriger dans la ligne au bas de la page à l'aide du message d'erreur.

Quant à la liste d'erreurs, lisez au chapitre 3.4, s.v.p.

➤ Mémoriser le fichier avec **VALIDER** (F6): Il sera conversé, l'édition terminée. La SELECTION DE FONCTION est affichée.

3.3.6 Introduction de blocs fonctionnels

Exemple Le fichier B:FBTESTA0.SEQ, imprimé sur la page suivante, sert d'exemple de travail. Encore une fois, c'est la commande de la porte de garage, mais maintenant programmée comme bloc fonctionnel pour vous montrer l'édition différente de ces types de bloc.

Ici, l'appel du bloc doit être programmé sous forme symbolique. Générez donc, s.v.p., la liste d'assignations suivante dans le fichier symbolique TEST@@Z0.INI, afin que la conversion fonctionne. Comment vous travaillez avec l'éditeur symbolique est expliqué dans la description STEP 5, manuel volume 2, au chapitre Editeur symbolique.


```
FICHIER SEQ.: TEST@@Z0.INI

E1.0     FIN-HAUT              FIN DE COURSE EN HAUT
E1.1     FIN-BAS               FIN DE COURSE EN BAS
E1.2     T-OUV E               TOUCHE OUVERT EXTERIEUR
E1.3     T-FER E               TOUCHE FERME EXTERIEUR
E1.4     BOUTCLE               BOUTON A CLE EXTERIEUR
E1.5     T-OUV I               TOUCHE OUVERT INTERIEUR
E1.6     T-FER I               TOUCHE FERME INTERIEUR
E1.7     STOP                  TOUCHE DE SECURITE STOP
A1.0     MOT-HAUT              MOTEUR VERS LE HAUT
A1.1     MOT-BAS               MOTEUR VERS LE BAS
T1       RET-ENCL              RETARD A L'ENCLENCHEMENT, 5 SEC.
FB1      GARAGE                FB POUR LA COMMANDE D'UNE PORTE DE GARAGE
```

Si vous désirez de l'information générale sur les blocs fonctionnels, lisez s.v.p. l'introduction et le paragraphe Blocs fonctionnels dans la description STEP 5, manuel volume 2 de votre console de programmation.

Commande

3.3.6 Introduction de blocs fonctionnels

Exemple Le fichier B:FBTESTA0.SEQ, imprimé sur la page suivante, sert d'exemple de travail. Encore une fois, c'est la commande de la porte de garage, mais maintenant programmée comme bloc fonctionnel pour vous montrer l'édition différente de ces types de bloc.

Ici, l'appel du bloc doit être programmé sous forme symbolique. Générez donc, s.v.p., la liste d'assignations suivante dans le fichier symbolique TEST@@Z0.INI, afin que la conversion fonctionne. Comment vous travaillez avec l'éditeur symbolique est expliqué dans la description STEP 5, manuel volume 2, au chapitre Editeur symbolique.

3

FICHIER SEQ.: TEST@@Z0.INI

E1.0 FIN-HAUT FIN DE COURSE EN HAUT
E1.1 FIN-BAS FIN DE COURSE EN BAS
E1.2 T-OUV E TOUCHE OUVERT EXTERIEUR
E1.3 T-FER E TOUCHE FERME EXTERIEUR
E1.4 BOUTCLE BOUTON A CLE EXTERIEUR
E1.5 T-OUV I TOUCHE OUVERT INTERIEUR
E1.6 T-FER I TOUCHE FERME INTERIEUR
E1.7 STOP TOUCHE DE SECURITE STOP
A1.0 MOT-HAUT MOTEUR VERS LE HAUT
A1.1 MOT-BAS MOTEUR VERS LE BAS
T1 RET-ENCL RETARD A L'ENCLENCHEMENT, 5 SEC.
FB1 GARAGE FB POUR LA COMMANDE D'UNE PORTE DE GARAGE

Si vous désirez de l'information générale sur les blocs fonctionnels, lisez s.v.p. l'introduction et le paragraphe Blocs fonctionnels dans la description STEP 5, manuel volume 2 de votre console de programmation.

C79000-B8577-C677-02

3 - 41

```
Source LIST: B: FBTESTA0.SEQ

ADR    INSTRUCTION      MNEMONIQUE              COMMENTAIRE D'INSTRUCTION
              #         GARAGE                  FB 1 POUR UNE PORTE DE GARAGE
       #N  GARAGE
FINH   (E)                                      FIN DE COURSE HAUT
FINB   (E)                                      FIN DE COURSE BAS
TO-I   (E)                                      TOUCHE OUVERT INTERIEUR
TO-E   (E)                                      TOUCHE OUVERT EXTERIEUR
TF-I           (E)                              TOUCHE FERME INTERIEUR
TF-E   (E)                                      TOUCHE FERME EXTERIEUR
CLE    (E)                                      BOUTON A CLE
STOP   (E)                                      TOUCHE DE SECURITE STOP
MHAUT  (A)                                      MOTEUR VERS LE HAUT
MBAS   (A)                                      MOTEUR VERS LE BAS
               #UB                              OUVRIR DE L'EXTERIEUR OU L'INTERIEUR
               UN       =STOP
               U(
               U        =TO-E
               U        =CLE
               O        =TO-I
               )
               UN       =FINH
               S        = MHAUT
               ***
               #UB                              FERMER DE L'EXTERIEUR OU L'INTERIEUR
*RAZ SORTIE MOTEUR VERS LE HAUT
               O        =FINH
               O        =STOP
               RB       =MHAUT
               ***
               #UB                              FERMER DE L'EXTERIEUR OU L'INTERIEUR
*LA PORTE SE FERME IMMEDIATEMENT
               U(
               U        =TF-E
               U        =CLE
               O        =TF-I
               )
               UN       =FINB
               S        =MBAS
               ***
               #UB                              FERMER DE L'EXTERIEUR OU L'INTERIEUR
*RAZ SORTIE MOTEUR VERS LE BAS
               O        =FINB
               O        =STOP
               RB       =MBAS
               BE
```


Commande

Source LIST: B: FBTESTA0.SEQ

ADR   INSTRUCTION   MNEMONIQUE   COMMENTAIRE D'INSTRUCTION
              #   GARAGE   FB 1 POUR UNE PORTE DE GARAGE
       #N GARAGE
FINH   (E)   FIN DE COURSE HAUT
FINB   (E)   FIN DE COURSE BAS
TO-I   (E)   TOUCHE OUVERT INTERIEUR
TO-E   (E)   TOUCHE OUVERT EXTERIEUR
TF-I   (E)   TOUCHE FERME INTERIEUR
TF-E   (E)   TOUCHE FERME EXTERIEUR
CLE   (E)   BOUTON A CLE
STOP   (E)   TOUCHE DE SECURITE STOP
MHAUT (A)   MOTEUR VERS LE HAUT
MBAS   (A)   MOTEUR VERS LE BAS
               #UB   OUVRIR DE L'EXTERIEUR OU L'INTERIEUR
               UN   =STOP
               U(
               U   =TO-E
               U   =CLE
               O   =TO-I
)
               UN   =FINH
               S   = MHAUT
               ***
               #UB   FERMER DE L'EXTERIEUR OU L'INTERIEUR
*RAZ SORTIE MOTEUR VERS LE HAUT
               O   =FINH
               O   =STOP
               RB   =MHAUT
               ***
               #UB   FERMER DE L'EXTERIEUR OU L'INTERIEUR
*LA PORTE SE FERME IMMEDIATEMENT
               U(
               U   =TF-E
               U   =CLE
               O   =TF-I
)
               UN   =FINB
               S   =MBAS
               ***
               #UB   FERMER DE L'EXTERIEUR OU L'INTERIEUR
*RAZ SORTIE MOTEUR VERS LE BAS
               O   =FINB
               O   =STOP
               RB   =MBAS
               BE

3 - 42

C79000-B8577-C677-02

Introduction

Condition: Le paquet éditeur LIST/compilateur par lots est chargé.

Recommencez l'exemple, remplissez le PREREGLAGE avec les noms de fichier FBTEST pour le fichier source LIST et le fichier intermédiaire, et TEST@@ pour le fichier programme et symbolique. Actionnez *valider* et appelez la fonction d'éditeur.

Si vous n'avez pas quitté le paquet éditeur LIST/compilateur par lots, la SELECTION DE FONCTION est affichée.

Allez au PREREGLAGE et modifiez le nom du fichier source LIST en FBTEST.

Changer le PREREGLAGE

➤ Appeler PREREGLage (F6); le curseur clignote au champ FICHIER SOURCE LIST.

➤ Placer le curseur sur le nom de fichier avec la **touche flèche fine vers la droite** et l'écraser avec *FBTEST*.

➤ Appuyer sur la touche Return et maintenant, le fichier intermédiaire porte également le nom FBTEST. Les données qui restent ne seront pas modifiées.

➤ **VALIDER** (F6).

➤ Appelez la fonction d'éditeur avec **EDITER** (F1).

Maintenant, vous pouvez commencer à programmer votre bloc fonctionnel "Garage".

Explications sur l'édition de blocs fonctionnels:

Les conventions d'écriture des caractères de commande et des opérations sont décrites aux paragraphes 3.3.2 et 3.3.3. En introduisant, faites attention aux particularités des blocs fonctionnels:

- Des opérandes formels sont définés dans les colonnes ADR et INSTRUCTION. Ces définitions correspondent à la liste d'identifications dans des blocs de fonction qui sont créés au sein du paquet CONT-LOG-LIST.
- Les opérandes formels consistent en 4 caractères au maximum.
- La définition comme entrée, sortie etc. est mise entre parenthèses.
- Lors de la programmation, l'opérande formel doit être précédé par un signe d'égalité (comme dans le paquet CONT-LOG-LIST).

Commande

Introduction

Condition: Le paquet éditeur LIST/compilateur par lots est chargé.

Recommencez l'exemple, remplissez le PREREGLAGE avec les noms de fichier FBTEST pour le fichier source LIST et le fichier intermédiaire, et TEST@@ pour le fichier programme et symbolique. Actionnez *valider* et appelez la fonction d'éditeur.

Si vous n'avez pas quitté le paquet éditeur LIST/compilateur par lots, la SELECTION DE FONCTION est affichée.

Allez au PREREGLAGE et modifiez le nom du fichier source LIST en FBTEST.

Changer le PREREGLAGE

➤ Appeler PREREGLage (F6); le curseur clignote au champ FICHIER SOURCE LIST.

➤ Placer le curseur sur le nom de fichier avec la **touche flèche fine vers la droite** et l'écraser avec *FBTEST*.

➤ Appuyer sur la touche Return et maintenant, le fichier intermédiaire porte également le nom FBTEST. Les données qui restent ne seront pas modifiées.

➤ **VALIDER** (F6).

➤ Appelez la fonction d'éditeur avec **EDITER** (F1).

Maintenant, vous pouvez commencer à programmer votre bloc fonctionnel "Garage".

Explications sur l'édition de blocs fonctionnels:

Les conventions d'écriture des caractères de commande et des opérations sont décrites aux paragraphes 3.3.2 et 3.3.3. En introduisant, faites attention aux particularités des blocs fonctionnels:

- Des opérandes formels sont définés dans les colonnes ADR et INSTRUCTION. Ces définitions correspondent à la liste d'identifications dans des blocs de fonction qui sont créés au sein du paquet CONT-LOG-LIST.
- Les opérandes formels consistent en 4 caractères au maximum.
- La définition comme entrée, sortie etc. est mise entre parenthèses.
- Lors de la programmation, l'opérande formel doit être précédé par un signe d'égalité (comme dans le paquet CONT-LOG-LIST).

C79000-B8577-C677-02

3 - 43

Le bloc fonctionnel FB1 nommé Garage est alloué au symbole "Garage" dans le fichier symbolique. Pour cette raison, le début de bloc peut être programmé de façon symbolique.

Des commentaires et des titres de segments seront traités comme ceux du bloc de programme. Vous introduisez la fin de segment via la touche de fonction allouée par vous-même (voir plus haut 3.3.5.2) et mémorisez par **SAUVEGARDER** (F7).

Suite d'opérations (du texte à introduire est imprimé *en italique*, des fonctions à utiliser **en gras**)

- ➤ #
- ➤ Appuyer sur la **touche flèche large vers la droite,** introduire *Garage* dans la colonne MNEMONIQUE,
- ➤ appuyer sur la **touche flèche large vers la droite** pour le commentaire
- ➤ *FB1 pour une porte de garage*
- ➤ appuyer sur la **touche Return,**
- ➤ *#N*
- ➤ espace
- ➤ *Garage*
- ➤ appuyer sur la **touche Return,**
- ➤ appuyer sur la **touche flèche large vers la gauche,** le curseur se trouve dans la colonne ADResses.
- ➤ Ecrire *FINH*, après le quatriéme caractère, le curseur saute à la colonne INSTRUCTION
- ➤ *(E)*
- ➤ appuyer deux fois sur la **touche flèche large vers la droite.** Vous écrivez comme commentaire d'instruction
- ➤ *fin de course en haut*
- ➤ appuyer sur la **touche Return,**

et cetera.

Commande

Le bloc fonctionnel FB1 nommé Garage est alloué au symbole "Garage" dans le fichier symbolique. Pour cette raison, le début de bloc peut être programmé de façon symbolique.

Des commentaires et des titres de segments seront traités comme ceux du bloc de programme. Vous introduisez la fin de segment via la touche de fonction allouée par vous-même (voir plus haut 3.3.5.2) et mémorisez par **SAUVEGARDER** (F7).

Suite d'opérations (du texte à introduire est imprimé *en italique*, des fonctions à utiliser **en gras**)

- ➤ #
- ➤ Appuyer sur la **touche flèche large vers la droite,** introduire *Garage* dans la colonne MNEMONIQUE,
- ➤ appuyer sur la **touche flèche large vers la droite** pour le commentaire
- ➤ *FB1 pour une porte de garage*
- ➤ appuyer sur la **touche Return,**
- ➤ *#N*
- ➤ espace
- ➤ *Garage*
- ➤ appuyer sur la **touche Return,**
- ➤ appuyer sur la **touche flèche large vers la gauche,** le curseur se trouve dans la colonne ADResses.
- ➤ Ecrire *FINH*, après le quatriéme caractère, le curseur saute à la colonne INSTRUCTION
- ➤ *(E)*
- ➤ appuyer deux fois sur la **touche flèche large vers la droite.** Vous écrivez comme commentaire d'instruction
- ➤ *fin de course en haut*
- ➤ appuyer sur la **touche Return,**

et cetera.

Les instructions correspondent au schéma du bloc de programme:

- *U(*
- appuyer sur la **touche Return,**
- *U*
- espace
- *=PO-E*
- appuyer sur la **touche Return,**

et cetera.

Commande

Les instructions correspondent au schéma du bloc de programme:

- *U(*
- appuyer sur la **touche Return,**
- *U*
- espace
- *=PO-E*
- appuyer sur la **touche Return,**

et cetera.

3

C79000-B8577-C677-02

3 - 45

Paramétrage d'un bloc fonctionnel

Pour paramétrer le bloc fonctionnel, c.-à-d., le pourvoir d'opérandes actuels, vous écrivez un bloc de programme:

```
Source LIST:  B: FBTESTA0.SEQ

ADR    INSTRUCTION      MNEMONIQUE              COMMENTAIRE D'INSTRUCTION
       #PB2
       SPA              GARAGE                  Parametrage de FB1
       ,E 1.0
       ,E 1.1
       ,                T-OUV I
       ,                T-OUV E
       ,                T-FER I
       ,                T-FER E
       ,                BOUTCLE
       ,                STOP
       ,                MOT-HAUT
       ,                MOT-BAS
       BE
```

Vous pouvez introduire les opérandes actuels de façon absolue ou symbolique. Faites attention, s.v.p.,

- que chaque opérande actuel doit être précédé d'une virgule comme caractère de commande, et
- que les paramètres concordent dans leur ordre avec la liste d'identifications des opérandes formels dans le bloc fonctionnel.

Commande

Paramétrage d'un bloc fonctionnel

Pour paramétrer le bloc fonctionnel, c.-à-d., le pourvoir d'opérandes actuels, vous écrivez un bloc de programme:

```
Source LIST:  B: FBTESTA0.SEQ

ADR    INSTRUCTION      MNEMONIQUE              COMMENTAIRE D'INSTRUCTION
       #PB2
       SPA              GARAGE                  Parametrage de FB1
       ,E 1.0
       ,E 1.1
       ,                T-OUV I
       ,                T-OUV E
       ,                T-FER I
       ,                T-FER E
       ,                BOUTCLE
       ,                STOP
       ,                MOT-HAUT
       ,                MOT-BAS
       BE
```

Vous pouvez introduire les opérandes actuels de façon absolue ou symbolique. Faites attention, s.v.p.,

- que chaque opérande actuel doit être précédé d'une virgule comme caractère de commande, et
- que les paramètres concordent dans leur ordre avec la liste d'identifications des opérandes formels dans le bloc fonctionnel.

3 - 46

C79000-B8577-C677-02

Suite d'opérations:

➤ *#PB1*

➤ appuyer sur la **touche Return,**

➤ *SPA*

➤ appuyer une fois sur la **touche flèche large vers la droite,**

➤ *GARAGE*

➤ appuyer une fois sur la **touche flèche large vers la droite,**

➤ *paramétrage du FB1*

➤ appuyer sur la **touche Return,**

➤ *,E 1.0*

➤ appuyer sur la **touche Return,**

ou de façon symbolique:

➤ ,

➤ appuyer une fois sur la **touche flèche large vers la droite,**

➤ *T-OUV E*

➤ appuyer sur la **touche Return.**

Mémorisez temporairement avec **SAUVEGARDER** (F7), car le bloc de données qui suit au paragraphe 3.3.7 sera également écrit dans ce fichier source LIST.

Commande

Suite d'opérations:

➤ *#PB1*

➤ appuyer sur la **touche Return,**

➤ *SPA*

➤ appuyer une fois sur la **touche flèche large vers la droite,**

➤ *GARAGE*

➤ appuyer une fois sur la **touche flèche large vers la droite,**

➤ *paramétrage du FB1*

➤ appuyer sur la **touche Return,**

➤ *,E 1.0*

➤ appuyer sur la **touche Return,**

ou de façon symbolique:

➤ ,

➤ appuyer une fois sur la **touche flèche large vers la droite,**

➤ *T-OUV E*

➤ appuyer sur la **touche Return.**

Mémorisez temporairement avec **SAUVEGARDER** (F7), car le bloc de données qui suit au paragraphe 3.3.7 sera également écrit dans ce fichier source LIST.

3

C79000-B8577-C677-02

3 - 47

3.3.7 Introduction de blocs de données

Exemple Le bloc de données qui suit est la base de travail de ce paragraphe.

Si vous désirez de l'information sur les blocs de données, consultez, s.v.p., la description STEP 5 de votre manuel PG, volume 2, Introduction et chapitre Blocs de données.

```
Source LIST:  B:FBTESTA0.SEQ

ADR    INSTRUCTION      MNEMONIQUE               COMMENTAIRE D'INSTRUCTION

       #DB 12
0      KH                      FFFF
1      KM                      11111111 11000000        NOMBRE DE PIECES
2      KH                      0013
3      KF                      -32768
4      KF                      +32767
5      KG                      -2740000+22
7      KG                      -1234000+05
100    KY                      022,033
111    KY                      022,033
       KY                      022,033
       KY                      022,033
       KH                      ADAC
       KH                      4538
       KF                      +32767
       KF                      +32767
       KF                      +32767
       KF                      +32767
       KF                      +32767
       KY                      022,033
       KY                      022,033
       BE
```


Commande

3.3.7 Introduction de blocs de données

Exemple Le bloc de données qui suit est la base de travail de ce paragraphe.

Si vous désirez de l'information sur les blocs de données, consultez, s.v.p., la description STEP 5 de votre manuel PG, volume 2, Introduction et chapitre Blocs de données.

Source LIST: B:FBTESTA0.SEQ

ADR   INSTRUCTION   MNEMONIQUE   COMMENTAIRE D'INSTRUCTION

       #DB 12
0   KH   FFFF
1   KM   11111111 11000000   NOMBRE DE PIECES
2   KH   0013
3   KF   -32768
4   KF   +32767
5   KG   -2740000+22
7   KG   -1234000+05
100   KY   022,033
111   KY   022,033
       KY   022,033
       KY   022,033
       KH   ADAC
       KH   4538
       KF   +32767
       KF   +32767
       KF   +32767
       KF   +32767
       KF   +32767
       KY   022,033
       KY   022,033
       BE

3 - 48

C79000-B8577-C677-02

Introduction

Condition: Le paquet éditeur LIST/compilateur par lots est chargé.

Si vous venez d'avoir fait l'exemple pour les blocs fonctionnels, vous vous trouvez dans la fonction d'éditeur, et le fichier FBTEST est affiché.

Recommencez l'exemple, remplissez le PREREGLAGE avec les noms de fichier FBTEST pour le fichier source LIST et fichier intermédiaire, et TEST@@ pour le fichier programme et symbolique. Actionnez *valider* et appelez la fonction d'éditeur.

Explications sur l'édition de blocs de données

Les conventions d'écriture des caractères de commande et des opérations sont décrites aux paragraphes 3.3.2 et 3.3.3.
En introduisant, faites attention aux particularités des blocs de données:

- Le format de donnée se trouve dans la colonne INSTRUCTIONS
- La valeur dans la colonne MNEMONIQUE
- L'adresse du mot de données, par rapport au début de bloc, se trouve dans la colonne ADR.
- Vous ne devez ni marquer vos mots de données avec des signes d'égalité ni les terminer avec point-virgule.

TUYAU: Si vous introduisez une adresse qui n'est pas identique à l'adresse effective dans DB, l'espace sera rempli en conversant avec KH 0000 (dans l'exemple, les adresses 9 a 99). Ainsi, vous créez de la place pour des données du processus.

Au contraire du paquet CONT-LOG-LIST, le facteur de répétition ne peut pas être employé directement mais seulement via la combinaison avec la fonction **COPIER**.

Suite d'opérations:

➤ *#DB12*

➤ appuyer sur la **touche Return**

➤ appuyer une fois sur la **touche flèche large vers la gauche** et

➤ introduire l'adresse *0*.

➤ Appuyer une fois sur la **touche flèche large vers la droite**. Introduisez

➤ *KH* dans la colonne d'INSTRUCTION.

Commande

Introduction

Condition: Le paquet éditeur LIST/compilateur par lots est chargé.

Si vous venez d'avoir fait l'exemple pour les blocs fonctionnels, vous vous trouvez dans la fonction d'éditeur, et le fichier FBTEST est affiché.

Recommencez l'exemple, remplissez le PREREGLAGE avec les noms de fichier FBTEST pour le fichier source LIST et fichier intermédiaire, et TEST@@ pour le fichier programme et symbolique. Actionnez *valider* et appelez la fonction d'éditeur.

Explications sur l'édition de blocs de données

Les conventions d'écriture des caractères de commande et des opérations sont décrites aux paragraphes 3.3.2 et 3.3.3.
En introduisant, faites attention aux particularités des blocs de données:

- Le format de donnée se trouve dans la colonne INSTRUCTIONS
- La valeur dans la colonne MNEMONIQUE
- L'adresse du mot de données, par rapport au début de bloc, se trouve dans la colonne ADR.
- Vous ne devez ni marquer vos mots de données avec des signes d'égalité ni les terminer avec point-virgule.

TUYAU: Si vous introduisez une adresse qui n'est pas identique à l'adresse effective dans DB, l'espace sera rempli en conversant avec KH 0000 (dans l'exemple, les adresses 9 a 99). Ainsi, vous créez de la place pour des données du processus.

Au contraire du paquet CONT-LOG-LIST, le facteur de répétition ne peut pas être employé directement mais seulement via la combinaison avec la fonction **COPIER**.

Suite d'opérations:

➤ *#DB12*

➤ appuyer sur la **touche Return**

➤ appuyer une fois sur la **touche flèche large vers la gauche** et

➤ introduire l'adresse *0*.

➤ Appuyer une fois sur la **touche flèche large vers la droite**. Introduisez

➤ *KH* dans la colonne d'INSTRUCTION.

C79000-B8577-C677-02

3 - 49

➤ Appuyer une fois sur la **touche flèche large vers la droite.** Introduisez la valeur

➤ *FFFF* dans la colonne de MNEMONIQUE,

➤ appuyer sur la **touche Return.**

➤ Appuyer une fois sur la **touche flèche large vers la gauche** et

➤ introduire l'adresse *1.*

➤ Appuyer sur la **touche flèche large vers la droite,**

➤ introduire *KM,*

➤ appuyer sur la **touche flèche large vers la droite,**

➤ introduire *11111111 11000000,*

➤ appuyer sur la **touche flèche large vers la droite,**

➤ *écrire NOMBRE DE PIECES* comme commentaire,

➤ appuyer sur la **touche Return**

et cetera.

Avec KY 22,23

➤ vous écrivez la ligne une fois,

➤ la retenez avec →**TAMPON** (F1),

➤ **LIGNE** (F1), **RETOURnez** (F8) et **touche Return,** et

➤ la copiez avec **COPIER** (F2),

➤ facteur de répétition 3 et

➤ **LIGNE** (F1).

Vous pouvez également employer la fonction COPIER (F2) BLOC (F4) pour multiplier.

Terminez maintenant vos exemples d'exercices avec **VALIDER** (F6). Le fichier source LIST FBTEST sera alors mémorisé et conversé dans le fichier intermédiaire. La fonction *éditeur* est aussi terminée.

Commande

➤ Appuyer une fois sur la **touche flèche large vers la droite.** Introduisez la valeur

➤ *FFFF* dans la colonne de MNEMONIQUE,

➤ appuyer sur la **touche Return.**

➤ Appuyer une fois sur la **touche flèche large vers la gauche** et

➤ introduire l'adresse *1.*

➤ Appuyer sur la **touche flèche large vers la droite,**

➤ introduire *KM,*

➤ appuyer sur la **touche flèche large vers la droite,**

➤ introduire *11111111 11000000,*

➤ appuyer sur la **touche flèche large vers la droite,**

➤ *écrire NOMBRE DE PIECES* comme commentaire,

➤ appuyer sur la **touche Return**

et cetera.

Avec KY 22,23

➤ vous écrivez la ligne une fois,

➤ la retenez avec →**TAMPON** (F1),

➤ **LIGNE** (F1), **RETOURnez** (F8) et **touche Return,** et

➤ la copiez avec **COPIER** (F2),

➤ facteur de répétition 3 et

➤ **LIGNE** (F1).

Vous pouvez également employer la fonction COPIER (F2) BLOC (F4) pour multiplier.

Terminez maintenant vos exemples d'exercices avec **VALIDER** (F6). Le fichier source LIST FBTEST sera alors mémorisé et conversé dans le fichier intermédiaire. La fonction *éditeur* est aussi terminée.

3 - 50

C79000-B8577-C677-02

3.3.8 Modification d'un fichier source LIST

Si vous voulez modifier un fichier source LIST au sein de l'éditeur LIST/compilateur par lots, éditez-le sur l'écran avec *éditeur* et traitez-le à l'aide des fonctions d'éditeur.

Dans notre exemple, le fichier FBTEST doit être intégré au fichier source LIST TEST@@ via l'instruction Include. A cet effet, FBTESTA0.SEQ doit exister comme fichier intermédiaire. Dans le cas de notre exemple, cette condition est déjà remplie (voir plus haut).

Condition: Introduire TEST@@ comme fichier source LIST au préréglage.

➤ Editez le fichier TEST@@ avec **EDITEUR** (F1).

Sautez à la fin du fichier avec

➤ **RECHERCHE** (F4),

➤ **FIN** (F7), et

➤ **RETOURnez** (F8) ensuite au mode d'éditeur.

Le mode insérer est préréglé.

➤ Placez le curseur avant le premier bloc, entre BE et #PBn ou à la fin du fichier aprés la dernière fin de bloc BE;

➤ insérez un espace vertical; maintenant vous avez de la place pour l'instruction Include.

➤ *#I*

➤ espace

➤ *B:FBTEST*

➤ appuyez sur **VALIDER** (F6) pour mémoriser et convertir. Maintenant, votre fichier intermédiaire est actualisé.

Si vous conversez ensuite le fichier source LIST TEST@@A0.SEQ dans le fichier programme STEP 5 TEST@@ST.S5D, FBTESTA1.SEQ sera aussi converti et transféré au fichier programme. Dans ce fichier-là se trouvent alors tous les blocs édités pendant cette séance d'exercice.

Commande

3.3.8 Modification d'un fichier source LIST

Si vous voulez modifier un fichier source LIST au sein de l'éditeur LIST/compilateur par lots, éditez-le sur l'écran avec *éditeur* et traitez-le à l'aide des fonctions d'éditeur.

Dans notre exemple, le fichier FBTEST doit être intégré au fichier source LIST TEST@@ via l'instruction Include. A cet effet, FBTESTA0.SEQ doit exister comme fichier intermédiaire. Dans le cas de notre exemple, cette condition est déjà remplie (voir plus haut).

Condition: Introduire TEST@@ comme fichier source LIST au préréglage.

➤ Editez le fichier TEST@@ avec **EDITEUR** (F1).

Sautez à la fin du fichier avec

➤ **RECHERCHE** (F4),

➤ **FIN** (F7), et

➤ **RETOURnez** (F8) ensuite au mode d'éditeur.

Le mode insérer est préréglé.

➤ Placez le curseur avant le premier bloc, entre BE et #PBn ou à la fin du fichier aprés la dernière fin de bloc BE;

➤ insérez un espace vertical; maintenant vous avez de la place pour l'instruction Include.

➤ *#I*

➤ espace

➤ *B:FBTEST*

➤ appuyez sur **VALIDER** (F6) pour mémoriser et convertir. Maintenant, votre fichier intermédiaire est actualisé.

Si vous conversez ensuite le fichier source LIST TEST@@A0.SEQ dans le fichier programme STEP 5 TEST@@ST.S5D, FBTESTA1.SEQ sera aussi converti et transféré au fichier programme. Dans ce fichier-là se trouvent alors tous les blocs édités pendant cette séance d'exercice.

C79000-B8577-C677-02

3 - 51

3.4 Conversion avec la fonction COMPILER

Après cette conversion, votre fichier source LIST mémorisé avec *valider* existe comme fichier intermédiaire (INT). Pour le convertir ensuite dans un fichier programme STEP 5, appelez la fonction **COMPILER**. Là, vous pouvez convertir votre liste d'instructions dans le fichier programme nommé au préréglage. Avec **INT>MC5** le fichier intermédiaire est transformé en code machine MC5 ; avec **SEQ>MC5** le fichier source LIST est transformé en code machine et un fichier intermédiaire est créé automatiquement.

La fonction **SEQ>MC5** effectue d'abord la conversion **SEQ>INT**. Si des erreurs apparaissent, la conversion **INT>MC5** n'est pas démarrée et la fonction est au contraire terminée. Ainsi sont dressés dans la liste d'erreurs les messages d'erreurs qui sont apparus lors de la création du fichier intermédiaire. De manière analogue, la fonction **MC5>SEQ** démarre d'abord la conversion **MC5>INT**, puis la conversion **INT>SEQ** à condition que la création du fichier intermédiaire soit sans erreurs.

3.4.5 Suite d'opérations: Conversion dans le fichier programme

Condition: Le fichier source LIST FBTESTA0.SEQ se trouve dans le préréglage.

Convertir

➤ Appeler la fonction **COMPILER** avec F2,

➤ appuyer sur **INT>MC5** (F2) ou sur **SEQ>MC5** (F1).

➤ Remplir la ligne de commande suivante:

Convertir blocs:	OPT:	IMP:

La **touche Help** vous donne du renseignement sur les introductions possibles pour chaque champ d'introduction.

Commande

3.4 Conversion avec la fonction COMPILER

Après cette conversion, votre fichier source LIST mémorisé avec *valider* existe comme fichier intermédiaire (INT). Pour le convertir ensuite dans un fichier programme STEP 5, appelez la fonction **COMPILER**. Là, vous pouvez convertir votre liste d'instructions dans le fichier programme nommé au préréglage. Avec **INT>MC5** le fichier intermédiaire est transformé en code machine MC5 ; avec **SEQ>MC5** le fichier source LIST est transformé en code machine et un fichier intermédiaire est créé automatiquement.

La fonction **SEQ>MC5** effectue d'abord la conversion **SEQ>INT**. Si des erreurs apparaissent, la conversion **INT>MC5** n'est pas démarrée et la fonction est au contraire terminée. Ainsi sont dressés dans la liste d'erreurs les messages d'erreurs qui sont apparus lors de la création du fichier intermédiaire. De manière analogue, la fonction **MC5>SEQ** démarre d'abord la conversion **MC5>INT**, puis la conversion **INT>SEQ** à condition que la création du fichier intermédiaire soit sans erreurs.

3.4.5 Suite d'opérations: Conversion dans le fichier programme

Condition: Le fichier source LIST FBTESTA0.SEQ se trouve dans le préréglage.

Convertir

➤ Appeler la fonction **COMPILER** avec F2,

➤ appuyer sur **INT>MC5** (F2) ou sur **SEQ>MC5** (F1).

➤ Remplir la ligne de commande suivante:

Convertir blocs:	OPT:	IMP:

La **touche Help** vous donne du renseignement sur les introductions possibles pour chaque champ d'introduction.

3 - 52 C79000-B8577-C677-02

➤ Au champ BLOCS, appuyer sur la **touche Help**:

En plus des introductions usuelles du paquet de base STEP 5, vous pouvez indiquer des domaines de bloc à traiter, p. ex. PB12 - PB21.

➤ Ecrivez *B* dans ce champ et terminez avec la

➤ **touche Return.**

➤ Actionner la **touche Help** au champ OPTion.

Via "2" vous pouvez démarrer un test de conversion: Votre fichier intermédiaire sera converti et en même temps vérifié, cependant, il ne sera pas mémorisé dans le fichier programme. D'éventuelles erreurs sont accessible via la liste d'erreurs.

➤ Introduisez *2* et

➤ appuyez sur la **touche Return.**

Si votre fichier programme contient déjà des blocs du même nom et le champ OPTion est vide, la console de programmation vous invite alors de confirmer chaque processus de copiage; dans ce cas les blocs seront écrasés par les nouveaux blocs du même nom.
Si vous utilisez l'option "1", les blocs seront écrasés dans le fichier programme sans votre confirmation.

➤ Au champ IMPrimante, appuyer sur la **touche Help.**

Les formats d'impression sont les formats usuels du paquet de base STEP 5: écriture normale, écriture comprimée et écriture super-comprimée. La grandeur de la feuille (DIN A3 ou A4) dépend de l'imprimante reliée.

Ce champ reste vide dans notre exemple.

Convertir blocs: **B**	OPT: **2**	IMP:

➤ Appuyer sur la **touche de validation.**

La console de programmation maintenant réalise la conversion et le contrôle. Elle affiche quels blocs sont en train d'être traités et combien d'erreurs ont été trouvé ou s'il n'y a pas d'erreurs dans la conversion. Après cela, la console de programmation retourne à la sélection de fonction.

Commande

➤ Au champ BLOCS, appuyer sur la **touche Help**:

En plus des introductions usuelles du paquet de base STEP 5, vous pouvez indiquer des domaines de bloc à traiter, p. ex. PB12 - PB21.

➤ Ecrivez *B* dans ce champ et terminez avec la

➤ **touche Return.**

➤ Actionner la **touche Help** au champ OPTion.

Via "2" vous pouvez démarrer un test de conversion: Votre fichier intermédiaire sera converti et en même temps vérifié, cependant, il ne sera pas mémorisé dans le fichier programme. D'éventuelles erreurs sont accessible via la liste d'erreurs.

➤ Introduisez *2* et

➤ appuyez sur la **touche Return.**

Si votre fichier programme contient déjà des blocs du même nom et le champ OPTion est vide, la console de programmation vous invite alors de confirmer chaque processus de copiage; dans ce cas les blocs seront écrasés par les nouveaux blocs du même nom.
Si vous utilisez l'option "1", les blocs seront écrasés dans le fichier programme sans votre confirmation.

➤ Au champ IMPrimante, appuyer sur la **touche Help.**

Les formats d'impression sont les formats usuels du paquet de base STEP 5: écriture normale, écriture comprimée et écriture super-comprimée. La grandeur de la feuille (DIN A3 ou A4) dépend de l'imprimante reliée.

Ce champ reste vide dans notre exemple.

Convertir blocs: **B**	OPT: **2**	IMP:

➤ Appuyer sur la **touche de validation.**

La console de programmation maintenant réalise la conversion et le contrôle. Elle affiche quels blocs sont en train d'être traités et combien d'erreurs ont été trouvé ou s'il n'y a pas d'erreurs dans la conversion. Après cela, la console de programmation retourne à la sélection de fonction.

C79000-B8577-C677-02

3 - 53

Si la conversion est sans erreurs, vous répétez la conversion. Laissez le champ d'option vide pourque le fichier programme soit généré et introduisez ✱ au champ IMPression pour obtenir une impression.

Convertir blocs: **B**	OPT:	IMP: **Q**

➤ Appuyer sur la **touche de validation**.

Puis, les blocs du fichier source LIST FBTESTA0.SEQ seront transférés au fichier programme TEST@@ST.S5D en forme de code machine et mémorisés. Vous pouvez maintenant traiter ultérieurement les blocs FB1 et PB2 dans le paquet CONT-LOG-LIST (p. ex. tester dans l'AG).

3.4.6 Suite d'opérations: Reconvertir dans le fichier programme

Les opérations sont analogues de ceux de la conversion. Vous n'utilisez que la fonction **MC5>INT** (F4) ou **MC5>SEQ** (F5). Pour la *reconversion*, il est important que les fichiers correspondants se trouvent dans le préréglage. Les mêmes conventions sont valables dans la ligne de commande que pour la *conversion*. La touche Help dans le champ d'OPTion vous offre les variantes suivantes de transfert

➤ Appeler **COMPILER** (2),

➤ sélectionner **MC5>INT** (F4),

➤ remplir la ligne de commande,

➤ appuyer sur la **touche Return**.

Le fichier intermédiaire sera maintenant dressé à nouveau ou un fichier du même nom sera écrasé (après interrogation). La fonction **MC5>SEQ** crée automatiquement le fichier source LIST que vous pouvez traiter avec l'éditeur LIST (voir 3.4). Avec la fonction **MC5>INT**, vous devez vous-même créer du fichier intermédiaire un fichier source séquentiel via la fonction SPECIALE INT>SEQ (voir ci-dessous parag. 3.7).

Commande

Si la conversion est sans erreurs, vous répétez la conversion. Laissez le champ d'option vide pourque le fichier programme soit généré et introduisez ✱ au champ IMPression pour obtenir une impression.

Convertir blocs: **B**	OPT:	IMP: **Q**

➤ Appuyer sur la **touche de validation**.

Puis, les blocs du fichier source LIST FBTESTA0.SEQ seront transférés au fichier programme TEST@@ST.S5D en forme de code machine et mémorisés. Vous pouvez maintenant traiter ultérieurement les blocs FB1 et PB2 dans le paquet CONT-LOG-LIST (p. ex. tester dans l'AG).

3.4.6 Suite d'opérations: Reconvertir dans le fichier programme

Les opérations sont analogues de ceux de la conversion. Vous n'utilisez que la fonction **MC5>INT** (F4) ou **MC5>SEQ** (F5). Pour la *reconversion*, il est important que les fichiers correspondants se trouvent dans le préréglage. Les mêmes conventions sont valables dans la ligne de commande que pour la *conversion*. La touche Help dans le champ d'OPTion vous offre les variantes suivantes de transfert

➤ Appeler **COMPILER** (2),

➤ sélectionner **MC5>INT** (F4),

➤ remplir la ligne de commande,

➤ appuyer sur la **touche Return**.

Le fichier intermédiaire sera maintenant dressé à nouveau ou un fichier du même nom sera écrasé (après interrogation). La fonction **MC5>SEQ** crée automatiquement le fichier source LIST que vous pouvez traiter avec l'éditeur LIST (voir 3.4). Avec la fonction **MC5>INT**, vous devez vous-même créer du fichier intermédiaire un fichier source séquentiel via la fonction SPECIALE INT>SEQ (voir ci-dessous parag. 3.7).

3 - 54 C79000-B8577-C677-02

3.5 Liste d'erreurs

La liste d'erreurs contient non seulement les erreurs qui se sont produits lors de la *conversion* mais représente un procès-verbal complet de la conversion: En plus, elle liste les blocs étant convertis sans erreurs et indique, en cas d'abandon, l'endroit concerné.

Pour avoir un véritable exemple, vous installez une erreur dans votre FB1 (programmé au paragraphe 3.3.5): Editez le FB1 avec **EDITEUR** et écrivez p. ex. les instructions raz seulement avec *R*. Pendant la mémorisation avec **VALIDER**, on vous indique déjà qu'il y a des erreurs.

➤ Appelez maintenant la **LISTE ERR** (F3).

➤ Remplissez le champ d'introduction IMPression dans la ligne de commande afin de pouvoir corriger aisément la source LIST à l'aide de cette impression. La touche Help vous indique les paramètres pour le champ IMPression.

➤ Appuyer sur la **touche de validation**: Chaque instruction incorrecte est affichée avec identification de bloc et nombre de ligne et expliquée, et en plus, les blocs sont listés qui ont été converti correctement.


```
Fichier B:FBTESTAF.SEQ
CONVERSION SOURCE LIST B:FBTESTA0.SEQ  ⇒ FICHIER INTERMEDIAIRE B:FBTESTA1.SEQ
      R =MHAUT
*** ERREUR A LA LIGNE    28: OPERANDE NON AUTORISE   ***
      R =MBAS
*** ERREUR A LA LIGNE    45: OPERANDE NON AUTORISE   ***
*** FB 1  CONVERTI,     2 ERREURS TROUVE   ***
*** PB 1  CONVERTI,     BLOC SANS ERREUR   ***
*** CONVERSION TERMINEE,     2  ERREURS,  PAS D'AVERTISSEMENT(S)   ***
```

Edition de la liste d'erreurs

La liste d'erreurs sera éditée sur l'écran, si vous ne remplissez pas le champ IMPression dans la ligne de commande. Vous pouvez arrêter l'édition sur écran avec la touche d'abandon.Lorsque les listes d'erreurs sont longues, l'émission sur écran s'arrête toutes les 20 lignes ; vous pouvez alors soit arrêter l'émission avec la touche d'abandon, soit afficher la page d'écran suivante via la touche de validation.

Commande

3.5 Liste d'erreurs

La liste d'erreurs contient non seulement les erreurs qui se sont produits lors de la *conversion* mais représente un procès-verbal complet de la conversion: En plus, elle liste les blocs étant convertis sans erreurs et indique, en cas d'abandon, l'endroit concerné.

Pour avoir un véritable exemple, vous installez une erreur dans votre FB1 (programmé au paragraphe 3.3.5): Editez le FB1 avec **EDITEUR** et écrivez p. ex. les instructions raz seulement avec *R*. Pendant la mémorisation avec **VALIDER**, on vous indique déjà qu'il y a des erreurs.

➤ Appelez maintenant la **LISTE ERR** (F3).

➤ Remplissez le champ d'introduction IMPression dans la ligne de commande afin de pouvoir corriger aisément la source LIST à l'aide de cette impression. La touche Help vous indique les paramètres pour le champ IMPression.

➤ Appuyer sur la **touche de validation**: Chaque instruction incorrecte est affichée avec identification de bloc et nombre de ligne et expliquée, et en plus, les blocs sont listés qui ont été converti correctement.

3

Fichier B:FBTESTAF.SEQ
CONVERSION SOURCE LIST B:FBTESTA0.SEQ ⇒ FICHIER INTERMEDIAIRE B:FBTESTA1.SEQ
      R =MHAUT
*** ERREUR A LA LIGNE   28: OPERANDE NON AUTORISE   ***
      R =MBAS
*** ERREUR A LA LIGNE   45: OPERANDE NON AUTORISE   ***
*** FB 1 CONVERTI,   2 ERREURS TROUVE   ***
*** PB 1 CONVERTI,   BLOC SANS ERREUR   ***
*** CONVERSION TERMINEE,   2 ERREURS, PAS D'AVERTISSEMENT(S)   ***

Edition de la liste d'erreurs

La liste d'erreurs sera éditée sur l'écran, si vous ne remplissez pas le champ IMPression dans la ligne de commande. Vous pouvez arrêter l'édition sur écran avec la touche d'abandon.Lorsque les listes d'erreurs sont longues, l'émission sur écran s'arrête toutes les 20 lignes ; vous pouvez alors soit arrêter l'émission avec la touche d'abandon, soit afficher la page d'écran suivante via la touche de validation.

C79000-B8577-C677-02

3 - 55

3.6 Imprimer

Cette fonction ne sert qu'à imprimer le fichier source LIST préréglé. Vous ne devez par conséquent que déterminer le format d'impression dans la ligne de commande. (Vous trouvez des détails sur le format d'impression dans le description STEP 5, manuel volume 2, au chapitre sur l'entrée/la sortie de blocs LIST). Des fichiers convertis ne peuvent être sortis sur imprimante que via la ligne de commande de la fonction COMPILER.

Condition: L'imprimante est reliée et en ordre de marche; si vous avez une imprimante non Siemens, il faut la paramétrer dans le programme UTILITAIR IMPRIM. L'imprimante PT 88 est préréglé. Au préréglage de l'éditeur LIST/compilateur par lots se trouve le nom du fichier à imprimer, p. ex. FBTEST. La sélection de fonction est éditée sur l'écran.

➤ Appeler **IMPRIMER** (F4)

➤ remplir le champ IMP, l'édition standard en écriture normale est préréglée.

➤ Appuyer sur la **touche de validation.**

Le fichier source LIST FBTESTAQ.SEQ sera imprimé. La console de programmation retourne à la sélection de fonction.

Comme dans le paquet CONT-LOG-LIST, vous pouvez directer l'impression via un fichier. Vous définez ce fichier dans le programme UTILITAIR IMPRIM et inscrivez ensuite le nom du fichier imprimante au PREREGLAGE. Pour votre ordre d'impression actuelle, vous déterminez un format d'impression dans la ligne de commande comme d'habitude. Ce format d'impression sera également transféré au fichier imprimante.

Commande

3.6 Imprimer

Cette fonction ne sert qu'à imprimer le fichier source LIST préréglé. Vous ne devez par conséquent que déterminer le format d'impression dans la ligne de commande. (Vous trouvez des détails sur le format d'impression dans le description STEP 5, manuel volume 2, au chapitre sur l'entrée/la sortie de blocs LIST). Des fichiers convertis ne peuvent être sortis sur imprimante que via la ligne de commande de la fonction COMPILER.

Condition: L'imprimante est reliée et en ordre de marche; si vous avez une imprimante non Siemens, il faut la paramétrer dans le programme UTILITAIR IMPRIM. L'imprimante PT 88 est préréglé. Au préréglage de l'éditeur LIST/compilateur par lots se trouve le nom du fichier à imprimer, p. ex. FBTEST. La sélection de fonction est éditée sur l'écran.

➤ Appeler **IMPRIMER** (F4)

➤ remplir le champ IMP, l'édition standard en écriture normale est préréglée.

➤ Appuyer sur la **touche de validation.**

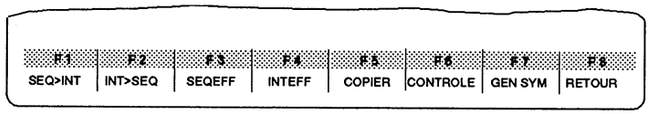
Le fichier source LIST FBTESTAQ.SEQ sera imprimé. La console de programmation retourne à la sélection de fonction.

Comme dans le paquet CONT-LOG-LIST, vous pouvez directer l'impression via un fichier. Vous définez ce fichier dans le programme UTILITAIR IMPRIM et inscrivez ensuite le nom du fichier imprimante au PREREGLAGE. Pour votre ordre d'impression actuelle, vous déterminez un format d'impression dans la ligne de commande comme d'habitude. Ce format d'impression sera également transféré au fichier imprimante.

3.7 Fonctions SPECIALES pour le traitement de fichiers intermédiaires et source

Les **fonctions spéciales** servent à traiter et transformer des fichiers séquentiels et intermédiaires et vous offrent un contrôle pour le fichier programme converti. Tous les processus se réfèrent aux fichiers qui sont déterminés au préréglage. Pour cette raison, faites attention que vous y avez inscrit les fichiers que vous pouvez utiliser pour votre travail. Dans notre exemple, le préréglage reste tel qu'il est.

Chaque processus peut être arrêté via la touche d'abandon.



3.7.5 COPIER

Vous utilisez la fonction COPIER pour faire des copies de sécurité. Elle rend possible non seulement la copie du fichier intermédiaire mais aussi une copie du fichier source LIST sur un autre lecteur. Pendant qu'une copie est faite, le console de programmation édite un commentaire, p. ex. "erreur de hardware", si le lecteur n'est pas fermé.

Pour un traitement sans danger des fichiers d'exemple, copiez les sur une disquette.

- ➤ Inscrire le fichier source LIST au PREREGLAGE,
- ➤ appeler **fonctions SPECIALES** (F5),
- ➤ appeler **COPIER** (F5),
- ➤ introduire lecteur: *A*
- ➤ appuyer sur la **touche de validation**. Le fichier intermédiaire est maintenant sauvegardé sur disquette. Sur la question "Copier fichier source SEQ. également?"
- ➤ appuyer sur la **touche de validation** (oui).

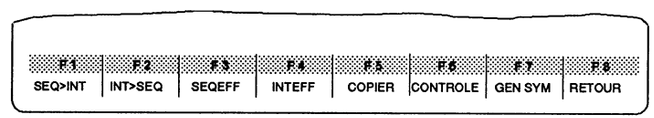
Les fichiers intermédiaires et source LIST sont maintenant copiés sur disquette. La console de programmation retourne à la sélection de fonction.

Commande

3.7 Fonctions SPECIALES pour le traitement de fichiers intermédiaires et source

Les **fonctions spéciales** servent à traiter et transformer des fichiers séquentiels et intermédiaires et vous offrent un contrôle pour le fichier programme converti. Tous les processus se réfèrent aux fichiers qui sont déterminés au préréglage. Pour cette raison, faites attention que vous y avez inscrit les fichiers que vous pouvez utiliser pour votre travail. Dans notre exemple, le préréglage reste tel qu'il est.

Chaque processus peut être arrêté via la touche d'abandon.



3

3.7.5 COPIER

Vous utilisez la fonction COPIER pour faire des copies de sécurité. Elle rend possible non seulement la copie du fichier intermédiaire mais aussi une copie du fichier source LIST sur un autre lecteur. Pendant qu'une copie est faite, le console de programmation édite un commentaire, p. ex. "erreur de hardware", si le lecteur n'est pas fermé.

Pour un traitement sans danger des fichiers d'exemple, copiez les sur une disquette.

- ➤ Inscrire le fichier source LIST au PREREGLAGE,
- ➤ appeler **fonctions SPECIALES** (F5),
- ➤ appeler **COPIER** (F5),
- ➤ introduire lecteur: *A*
- ➤ appuyer sur la **touche de validation**. Le fichier intermédiaire est maintenant sauvegardé sur disquette. Sur la question "Copier fichier source SEQ. également?"
- ➤ appuyer sur la **touche de validation** (oui).

Les fichiers intermédiaires et source LIST sont maintenant copiés sur disquette. La console de programmation retourne à la sélection de fonction.

3.7.6 SEQ>INT

Vous appelez **SEQ>INT** (F1), si vous voulez p. ex. convertir un fichier source LIST qui a été dressé à l'aide d'un autre éditeur de texte. Avec cette fonction, le fichier texte est transformé en un fichier intermédiaire - la condition pour une conversion dans le fichier programme.

- ➤ Inscrire le fichier texte au préréglage,
- ➤ activer **fonctions SPECIALES** (F5),
- ➤ appeler **SEQ>INT** (F1), le message: "Conversion du fichier source LIST dans le fichier intermédiaire?" est affiché.
- ➤ Appuyer sur la **touche de validation** (oui).

Maintenant, un fichier intermédiaire existe pour le traitement ultérieur. La console de programmation retourne à la sélection de fonction.

3.7.7 INT>SEQ

Vous appelez **INT>SEQ** (F2), quand vous avez p. ex. reconverti un fichier programme (avec COMPILER, MC5>INT) et voulez le modifier dans l'éditeur. A cet effet, vous devez transformer le fichier intermédiaire dans un fichier séquentiel. Des commentaires supplémentaires de l'ancien fichier source LIST se perdent.

- ➤ Déterminer les fichiers concernés au PREREGLAGE.
- ➤ Activer **fonctions SPECIALES,**
- ➤ Appeler **INT>SEQ** (F2), l'appareil demande: "Conversion du fichier intermédiaire dans le fichier source LIST?"
- ➤ Appuyer sur la **touche de validation** (oui).

Maintenant, un fichier source LIST est généré à nouveau. La console de programmation retourne à la sélection de fonction.

Commande

3.7.6 SEQ>INT

Vous appelez **SEQ>INT** (F1), si vous voulez p. ex. convertir un fichier source LIST qui a été dressé à l'aide d'un autre éditeur de texte. Avec cette fonction, le fichier texte est transformé en un fichier intermédiaire - la condition pour une conversion dans le fichier programme.

- ➤ Inscrire le fichier texte au préréglage,
- ➤ activer **fonctions SPECIALES** (F5),
- ➤ appeler **SEQ>INT** (F1), le message: "Conversion du fichier source LIST dans le fichier intermédiaire?" est affiché.
- ➤ Appuyer sur la **touche de validation** (oui).

Maintenant, un fichier intermédiaire existe pour le traitement ultérieur. La console de programmation retourne à la sélection de fonction.

3.7.7 INT>SEQ

Vous appelez **INT>SEQ** (F2), quand vous avez p. ex. reconverti un fichier programme (avec COMPILER, MC5>INT) et voulez le modifier dans l'éditeur. A cet effet, vous devez transformer le fichier intermédiaire dans un fichier séquentiel. Des commentaires supplémentaires de l'ancien fichier source LIST se perdent.

- ➤ Déterminer les fichiers concernés au PREREGLAGE.
- ➤ Activer **fonctions SPECIALES,**
- ➤ Appeler **INT>SEQ** (F2), l'appareil demande: "Conversion du fichier intermédiaire dans le fichier source LIST?"
- ➤ Appuyer sur la **touche de validation** (oui).

Maintenant, un fichier source LIST est généré à nouveau. La console de programmation retourne à la sélection de fonction.

3.7.8 SEQEFF et INTEFF

SEQEFF et **INTEFF** effacent le fichier intermédiaire et source LIST préréglé.

➤ Déterminer les fichiers concernés au PREREGLAGE.

➤ Appeler **fonctions SPECIALES** (F5),

➤ appeler **SEQEFF** (F3), l'appareil demande: "Effacer le fichier source LIST?"

➤ Appuyer sur la **touche de validation** (oui).

La console de programmation retourne à la sélection de fonction.

➤ Appeler **fonctions SPECIALES** (F5),

➤ appeler **INTEFF** (F4), l'appareil demande: "Effacer le fichier intermédiaire?"

➤ Appuyer sur la **touche de validation** (oui) ou abandonner (non).

La console de programmation retourne à la sélection de fonction.

3.7.9 CONTROLE

Le **CONTROLE** est une vérification ultérieure de blocs dans le fichier programme préréglé. Il vérifie également, si des blocs fonctionnels standard sont pourvus de paramètres corrects. D'éventuelles erreurs sont accessibles via la liste d'erreurs.

Dans les lignes de commande de cette fonction, vous pouvez inscrire des blocs individuels, des groupes de bloc, des types de bloc ou tous les blocs d'un fichier programme; la touche Help vous donnera des informations.

Commande

3.7.8 SEQEFF et INTEFF

SEQEFF et INTEFF effacent le fichier intermédiaire et source LIST préréglé.

➤ Déterminer les fichiers concernés au PREREGLAGE.

➤ Appeler **fonctions SPECIALES** (F5),

➤ appeler **SEQEFF** (F3), l'appareil demande: "Effacer le fichier source LIST?"

➤ Appuyer sur la **touche de validation** (oui).

La console de programmation retourne à la sélection de fonction.

➤ Appeler **fonctions SPECIALES** (F5),

➤ appeler **INTEFF** (F4), l'appareil demande: "Effacer le fichier intermédiaire?"

➤ Appuyer sur la **touche de validation** (oui) ou abandonner (non).

La console de programmation retourne à la sélection de fonction.

3.7.9 CONTROLE

Le **CONTROLE** est une vérification ultérieure de blocs dans le fichier programme préréglé. Il vérifie également, si des blocs fonctionnels standard sont pourvus de paramètres corrects. D'éventuelles erreurs sont accessibles via la liste d'erreurs.

Dans les lignes de commande de cette fonction, vous pouvez inscrire des blocs individuels, des groupes de bloc, des types de bloc ou tous les blocs d'un fichier programme; la touche Help vous donnera des informations.

➤ Au PREREGLAGE, déterminer le fichier programme à contrôler et, le cas échéant, le type d'AG voulu,

➤ appeler **fonctions SPECIALES** (F5),

➤ appeler **CONTROLE** (F6),

➤ remplir la ligne de commande: introduire p. ex. ✱,

➤ appuyer sur la **touche de validation**.

➤ Remplir la liste de bloc avec *PB1*.

➤ Appuyer sur la **touche Return**,

➤ introduction analogue de *FB1, DB12*,

➤ appuyer sur la **touche de validation**.

La console de programmation commente le contrôle. En cas d'erreurs, vous pouvez éditer la liste d'erreurs.

3.7.10 GEN-SYM

GEN-SYM crée du fichier source LIST un fichier source symbolique qui contient tous les paramètres absolus et symboles utilisés. Vous pouvez élargir le fichier source symbolique à l'aide de l'éditeur symbolique en y complétant p.ex. les affectations et en y inscrivant des commentaires. Les symboles et paramètres absolus apparaissent dans le fichier source symbolique autant de fois qu'ils ont été utilisés dans le fichier source LIST. Afin d'éliminer les éléments répétés, il vous faut procéder comme suit:

➤ Créer le fichier source symbolique avec GEN-SYM

Exemple:

absolu	symbole	commentaire
	ARRET D'URGENCE	
	ARRET D'URGENCE	
	ARRET D'URGENCE	

➤ Passer à l'éditeur symbolique (paquet 8)

➤ Compléter l'affectation à la première apparition du symbole ou du paramètre absolu

Commande

➤ Au PREREGLAGE, déterminer le fichier programme à contrôler et, le cas échéant, le type d'AG voulu,

➤ appeler **fonctions SPECIALES** (F5),

➤ appeler **CONTROLE** (F6),

➤ remplir la ligne de commande: introduire p. ex. ✱,

➤ appuyer sur la **touche de validation**.

➤ Remplir la liste de bloc avec *PB1*.

➤ Appuyer sur la **touche Return**,

➤ introduction analogue de *FB1, DB12*,

➤ appuyer sur la **touche de validation**.

La console de programmation commente le contrôle. En cas d'erreurs, vous pouvez éditer la liste d'erreurs.

3.7.10 GEN-SYM

GEN-SYM crée du fichier source LIST un fichier source symbolique qui contient tous les paramètres absolus et symboles utilisés. Vous pouvez élargir le fichier source symbolique à l'aide de l'éditeur symbolique en y complétant p.ex. les affectations et en y inscrivant des commentaires. Les symboles et paramètres absolus apparaissent dans le fichier source symbolique autant de fois qu'ils ont été utilisés dans le fichier source LIST. Afin d'éliminer les éléments répétés, il vous faut procéder comme suit:

➤ Créer le fichier source symbolique avec GEN-SYM

Exemple:

absolu	symbole	commentaire
	ARRET D'URGENCE	
	ARRET D'URGENCE	
	ARRET D'URGENCE	

➤ Passer à l'éditeur symbolique (paquet 8)

➤ Compléter l'affectation à la première apparition du symbole ou du paramètre absolu

3 - 60 C79000-B8577-C677-02

Exemple:

absolu	symbole	commentaire
E 1.0	ARRET D'URGENCE	arrêt d'urgence
	ARRET D'URGENCE	
	ARRET D'URGENCE	

➤ Convertir dans le fichier symbolique Ici vous pouvez ignorer les messages d'erreurs "symbole déjà existant"

➤ Reconvertir le fichier symbolique dans le fichier source symbolique (INT > SEQ)

Exemple:

absolu	symbole	commentaire
E 1.0	ARRET D'URGENCE	arrêt d'urgence

Le fichier source symbolique ne contient maintenant qu'une affectation qui porte le symbole ARRET D'URGENCE.

Commande

Exemple:

absolu	symbole	commentaire
E 1.0	ARRET D'URGENCE	arrêt d'urgence
	ARRET D'URGENCE	
	ARRET D'URGENCE	

➤ Convertir dans le fichier symbolique Ici vous pouvez ignorer les messages d'erreurs "symbole déjà existant"

➤ Reconvertir le fichier symbolique dans le fichier source symbolique (INT > SEQ)

Exemple:

absolu	symbole	commentaire
E 1.0	ARRET D'URGENCE	arrêt d'urgence

Le fichier source symbolique ne contient maintenant qu'une affectation qui porte le symbole ARRET D'URGENCE.

3

C79000-B8577-C677-02

3 - 61

Commande

3 - 62

C79000-B8577-C677-02

Version lignes de commande

4

Version lignes de commande

4

Le programme **COMPILE.CMD**

Le programme **COMPILE.CMD** est un programme normal CP/M qui doit être appelé du système d'exploitation, comme p.ex. **KONVER S5**. De ce fait, vous pouvez également l'utiliser dans des fichiers SUBMIT afin de faire défiler automatiquement des suites d'ordres. **COMPILE.CMD** exécute les conversions sélectionnables dans le paquet éditeur LIST/compilateur par lots, sans d'autres données utilisateur. **COMPILE.CMD** charge à cet effet les driver S5 nécessaires, puis les efface à nouveau. Les fichiers S5WX*.* doivent donc se trouver sur le PG, sans quoi **COMPILE** s'arrête par un message d'erreur.

fichiers à traiter

Vous pouvez indiquer les fichiers à traiter au cours de l'appel, ou bien les écrire dans le fichier dit fichier input (voir 4.2). Le fichier input peut aussi contenir les données pour plusieurs processus de conversions. Aucun signe joker (?,*) n'est admis dans les noms de fichiers.

Les fonctions de conversion sont identiques au déroulement de programme du paquet LIST/compilateur par lots (paquet 9), avec sélection des conversions correspondantes. Les options que vous pouvez introduire dans l'éditeur LIST/compilateur par lots respectivement dans la ligne de commande sont réglables uniquement via le fichier input. Les valeurs par défaut (default) utilisées par **COMPILE.CMD** sont expliquées sous le paragraphe 4.3.

Version ligne de commande

Le programme **COMPILE.CMD**

Le programme **COMPILE.CMD** est un programme normal CP/M qui doit être appelé du système d'exploitation, comme p.ex. **KONVER S5**. De ce fait, vous pouvez également l'utiliser dans des fichiers SUBMIT afin de faire défiler automatiquement des suites d'ordres. **COMPILE.CMD** exécute les conversions sélectionnables dans le paquet éditeur LIST/compilateur par lots, sans d'autres données utilisateur. **COMPILE.CMD** charge à cet effet les driver S5 nécessaires, puis les efface à nouveau. Les fichiers S5WX*.* doivent donc se trouver sur le PG, sans quoi **COMPILE** s'arrête par un message d'erreur.

fichiers à traiter

Vous pouvez indiquer les fichiers à traiter au cours de l'appel, ou bien les écrire dans le fichier dit fichier input (voir 4.2). Le fichier input peut aussi contenir les données pour plusieurs processus de conversions. Aucun signe joker (?,*) n'est admis dans les noms de fichiers.

Les fonctions de conversion sont identiques au déroulement de programme du paquet LIST/compilateur par lots (paquet 9), avec sélection des conversions correspondantes. Les options que vous pouvez introduire dans l'éditeur LIST/compilateur par lots respectivement dans la ligne de commande sont réglables uniquement via le fichier input. Les valeurs par défaut (default) utilisées par **COMPILE.CMD** sont expliquées sous le paragraphe 4.3.

4 - 2

C79000-B8577-C677-02

4.1 Appel

4.1.1 Appel sans donnée de paramètre

Syntaxe: COMPILE <SOURCE> <DESTINATION> <IDENTIF.LANGAGE>

Exemple: COMPILE @@@@@@AO.SEQ @@@@@@ST.S5D F

Les six combinaisons suivantes pour source et destination sont possibles:

Fich.source LIST	->	fich.progr.:	nnnnnnA0.SEQ	nnnnnnST.S5D
Fich.source LIST	->	fich.interm:	nnnnnnA0.SEQ	nnnnnnA1.SEQ
Fich.interm.	->	fich.progr.:	nnnnnnA1.SEQ	nnnnnnST.S5D
Fich.interm.	->	fich.source LIST:	nnnnnnA1.SEQ	nnnnnnA0.SEQ
Fich.progr.	->	fich.source LIST:	nnnnnnST.S5D	nnnnnnA0.SEQ
Fich.progr.	->	fich.interm:	nnnnnnST.S5D	nnnnnnA1.SEQ

4

Les noms de fichiers doivent répondre aux conventions S5, c.-à-d. qu'aux 6 caractères du nom facultatifs suit l'identification du fichier autorisée (A0.SEQ, A1.SEQ, ST.S5D). Il n'est pas fait de distinction entre l'écriture en minuscule et l'écriture en majuscule. Si le lecteur n'est pas précisé, **COMPILE** travaillera avec le lecteur préréglé (default) momentané.

En appelant **COMPILE**, vous pouvez indiquer l'identification de langue en option et avez les possibilités suivantes:

D (allemand), E (anglais) et F (français). La valeur standard (default) est D (allemand). Caractères de tabulation, espaces et virgules sont admis comme séparateurs entre les arguments.

Version ligne de commande

4.1 Appel

4.1.1 Appel sans donnée de paramètre

Syntaxe: COMPILE <SOURCE> <DESTINATION> <IDENTIF.LANGAGE>

Exemple: COMPILE @@@@@@AO.SEQ @@@@@@ST.S5D F

Les six combinaisons suivantes pour source et destination sont possibles:

Fich.source LIST	->	fich.progr.:	nnnnnnA0.SEQ	nnnnnnST.S5D
Fich.source LIST	->	fich.interm:	nnnnnnA0.SEQ	nnnnnnA1.SEQ
Fich.interm.	->	fich.progr.:	nnnnnnA1.SEQ	nnnnnnST.S5D
Fich.interm.	->	fich.source LIST:	nnnnnnA1.SEQ	nnnnnnA0.SEQ
Fich.progr.	->	fich.source LIST:	nnnnnnST.S5D	nnnnnnA0.SEQ
Fich.progr.	->	fich.interm:	nnnnnnST.S5D	nnnnnnA1.SEQ

4

Les noms de fichiers doivent répondre aux conventions S5, c.-à-d. qu'aux 6 caractères du nom facultatifs suit l'identification du fichier autorisée (A0.SEQ, A1.SEQ, ST.S5D). Il n'est pas fait de distinction entre l'écriture en minuscule et l'écriture en majuscule. Si le lecteur n'est pas précisé, **COMPILE** travaillera avec le lecteur préréglé (default) momentané.

En appelant **COMPILE**, vous pouvez indiquer l'identification de langue en option et avez les possibilités suivantes:

D (allemand), E (anglais) et F (français). La valeur standard (default) est D (allemand). Caractères de tabulation, espaces et virgules sont admis comme séparateurs entre les arguments.

C79000-B8577-C677-02 4 - 3

4.1.2 Appel avec fichier input

Syntaxe: COMPILE	# <NOM DE FICH.INPUT>	<IDENTIF.LANGUE>
Exemple: COMPILE	# A.TESTINP1.INP	F

Le nom du fichier input doit être conforme aux conventions CPM, c.-à-d. que le nom du fichier peut consister en 8 caractères au max. et le type de fichier en 3 caractères au max. (lettres ou chiffres). Si vous ne précisez pas le lecteur ou l'identification de langue, COMPILE travaillera avec le lecteur pré-affecté (default) et la valeur standard D allemand) sera prise pour l'identification de la langue).

Pour identifier un fichier en qualité de fichier input, le caractère # doit obligatoirement précéder le nom du fichier, autrement COMPILE s'arrête par un message d'erreur.

Version ligne de commande

4.1.2 Appel avec fichier input

Syntaxe: COMPILE # <NOM DE FICH.INPUT> <IDENTIF.LANGUE>
Exemple: COMPILE # A.TESTINP1.INP F

Le nom du fichier input doit être conforme aux conventions CPM, c.-à-d. que le nom du fichier peut consister en 8 caractères au max. et le type de fichier en 3 caractères au max. (lettres ou chiffres). Si vous ne précisez pas le lecteur ou l'identification de langue, COMPILE travaillera avec le lecteur pré-affecté (default) et la valeur standard D allemand) sera prise pour l'identification de la langue).

Pour identifier un fichier en qualité de fichier input, le caractère # doit obligatoirement précéder le nom du fichier, autrement COMPILE s'arrête par un message d'erreur.

4 - 4 C79000-B8577-C677-02

4.2 Format du fichier input

Le fichier input contient une liste des fichiers à traiter et des réglages désirés. Tous les réglages sont identiques aux paramètres pouvant être introduits dans le paquet LIST/compilateur par lots sur la ligne de commande des fonctions de conversion individuelles. Les réglages qui suivent après l'indication de fichier source et destination conservent leur valeur jusqu'à ce qu'ils soient modifiés explicitement. Lorsqu'aucun paramètre précis n'est indiqué, COMPILE travaillera avec l'affectation implicite (default), (voir 4.3). Après chaque bloc de paramètre, vous pouvez réindiquer un nouveau fichier source et destination avec de nouveaux paramètres.

Syntaxe:

<FICHIER SOURCE 1> <FICHIER DESTINATION 1>
$BLOC: <INDICATION BLOC>
$SYMB: <NOM FICHIER SYMBOLIQUE>
$OPT: <OPTIONS>
$TYPE D'AG: <IDENTIFICATION LANGAGE>
$IMP: <OPTION> NOM FICHIER IMPRIMANTE>
<FICHIER SOURCE 2> <FICHIER DESTINATION 2>
$..
$..

Exemple:

```
TEST1@A0.SEQ  TEST1@ ST.S5D
$BLOC:        FB3-10
$SYMB:        SYMB23Z0.INI
$OPT:         1
$TYPE D'AG:   AG135W
$IMP:         *_B:IMP1IM.INI

TEST2ST.S5D  TEST2A1.SEQ
$BLOC:       B
$OPT:        2
```

4

Version ligne de commande

4.2 Format du fichier input

Le fichier input contient une liste des fichiers à traiter et des réglages désirés. Tous les réglages sont identiques aux paramètres pouvant être introduits dans le paquet LIST/compilateur par lots sur la ligne de commande des fonctions de conversion individuelles. Les réglages qui suivent après l'indication de fichier source et destination conservent leur valeur jusqu'à ce qu'ils soient modifiés explicitement. Lorsqu'aucun paramètre précis n'est indiqué, COMPILE travaillera avec l'affectation implicite (default), (voir 4.3). Après chaque bloc de paramètre, vous pouvez réindiquer un nouveau fichier source et destination avec de nouveaux paramètres.

Syntaxe:

<FICHIER SOURCE 1> <FICHIER DESTINATION 1>
$BLOC: <INDICATION BLOC>
$SYMB: <NOM FICHIER SYMBOLIQUE>
$OPT: <OPTIONS>
$TYPE D'AG: <IDENTIFICATION LANGAGE>
$IMP: <OPTION> NOM FICHIER IMPRIMANTE>
<FICHIER SOURCE 2> <FICHIER DESTINATION 2>
$..
$..

Exemple:

TEST1@A0.SEQ TEST1@ ST.S5D
$BLOC: FB3-10
$SYMB: SYMB23Z0.INI
$OPT: 1
$TYPE D'AG: AG135W
$IMP: *_B:IMP1IM.INI

TEST2ST.S5D TEST2A1.SEQ
$BLOC: B
$OPT: 2

4

4.2.1
Indication de fichier source et destination

Vous devez inscrire le fichier source et destination sur la première ligne du fichier input. Le format et les possibilités de combinaisons sont identiques à l'appel de COMPILE sans paramètre (voir 4.1). Mais, ici, vous ne pouvez pas spécifier d'identification de langue, car celle-ci se trouve déjà sur la ligne d'appel après le nom du fichier input.

4.2.2
Paramètre $BLOC:

Pour toutes les conversions, excepté celle de fichier source LIST en fichier intermédiaire et vice versa, vous pouvez mettre en paramètre les données de blocs suivantes:

OBn, FBn, PBn:	conversion de blocs individuels
*:	conversion de plusieurs blocs individuels
OBn-m, FBn-m:	conversion de zones de blocs
OB,FB,PB:	conversion de types de blocs
B:	conversion de tous les blocs (Préréglage)

(Voir manuel, chapitre 3.4.1)

4.2.3
Paramètre $SYMB:

Par cette option, vous pouvez communiquer le nom du fichier symbolique à la conversion. Ceci correspond dans le paquet LIST/compilateur par lots à l'inscription du fichier symbolique dans le masque de préréglage. Le nom du fichier peut comprendre 6 caractères au max. et le type de lecteur, car l'extension Z0.SEQ est automa-tiquement rattachée. Le nom du fichier source LIST ou intermédiaire est préréglé.

Version ligne de commande

4.2.1
Indication de fichier source et destination

Vous devez inscrire le fichier source et destination sur la première ligne du fichier input. Le format et les possibilités de combinaisons sont identiques à l'appel de COMPILE sans paramètre (voir 4.1). Mais, ici, vous ne pouvez pas spécifier d'identification de langue, car celle-ci se trouve déjà sur la ligne d'appel après le nom du fichier input.

4.2.2
Paramètre $BLOC:

Pour toutes les conversions, excepté celle de fichier source LIST en fichier intermédiaire et vice versa, vous pouvez mettre en paramètre les données de blocs suivantes:

OBn, FBn, PBn:	conversion de blocs individuels
*:	conversion de plusieurs blocs individuels
OBn-m, FBn-m:	conversion de zones de blocs
OB,FB,PB:	conversion de types de blocs
B:	conversion de tous les blocs (Préréglage)

(Voir manuel, chapitre 3.4.1)

4.2.3
Paramètre $SYMB:

Par cette option, vous pouvez communiquer le nom du fichier symbolique à la conversion. Ceci correspond dans le paquet LIST/compilateur par lots à l'inscription du fichier symbolique dans le masque de préréglage. Le nom du fichier peut comprendre 6 caractères au max. et le type de lecteur, car l'extension Z0.SEQ est automa-tiquement rattachée. Le nom du fichier source LIST ou intermédiaire est préréglé.

4 - 6 C79000-B8577-C677-02

4.2.4
Paramètre $OPT

A la conversion du fichier source AWL dans le fichier programme ou bien du fichier intermédiaire dans le fichier programme, les options suivantes sont possibles:

VIDE: créer code-MC5 et interroger avant écrasement

1: créer code-MC5 et écraser sans interroger

2: créer aucun code-M, uniquement test

(Voir manuel chapitre 3.4.1.)

A la conversion du fichier programme dans le fichier source LIST ou bien du fichier programme dans le fichier intermédiaire, les options suivantes sont possibles:

VIDE: incrire uniquement symboles dans fich.source LIST

1: inscrire symboles et param.absolus dans fich.source LIST

2: inscrire uniquement param.absolus dans fich.source LIST

(Voir manuel chapitre 3.4.2)

4.2.5
Paramètre $ type d'AG

Ce paramètre détermine le langage à la conversion, c.-à-d. le jeu d'instructions pour le type d'AG correspondant. Le préréglage est NON (voir manuel chapitre 2.2.3).

4.2.6
Paramètre $IMP

L'indication d'un fichier imprimante n'est valable que pour la conversion du fichier source LIST ou du fichier intermédiaire dans le fichier programme. Le paramètre consiste en deux parties: premièrement en l'option, deuxièmement en le nom du fichier imprimante. L'option doit toujours être placée devant le nom de fichier et présente les trois possibilités suivantes:

*: impression standard

1: impression en écriture normale

2: impression en écriture comprimée (avec marge)

Le préréglage est VIDE (pas de listing);

(voir manuel chapitre 3.4.1).

Version ligne de commande

4.2.4
Paramètre $OPT

A la conversion du fichier source AWL dans le fichier programme ou bien du fichier intermédiaire dans le fichier programme, les options suivantes sont possibles:

VIDE: créer code-MC5 et interroger avant écrasement

1: créer code-MC5 et écraser sans interroger

2: créer aucun code-M, uniquement test

(Voir manuel chapitre 3.4.1.)

A la conversion du fichier programme dans le fichier source LIST ou bien du fichier programme dans le fichier intermédiaire, les options suivantes sont possibles:

VIDE: incrire uniquement symboles dans fich.source LIST

1: inscrire symboles et param.absolus dans fich.source LIST

2: inscrire uniquement param.absolus dans fich.source LIST

(Voir manuel chapitre 3.4.2)

4.2.5
Paramètre $ type d'AG

Ce paramètre détermine le langage à la conversion, c.-à-d. le jeu d'instructions pour le type d'AG correspondant. Le préréglage est NON (voir manuel chapitre 2.2.3).

4.2.6
Paramètre $IMP

L'indication d'un fichier imprimante n'est valable que pour la conversion du fichier source LIST ou du fichier intermédiaire dans le fichier programme. Le paramètre consiste en deux parties: premièrement en l'option, deuxièmement en le nom du fichier imprimante. L'option doit toujours être placée devant le nom de fichier et présente les trois possibilités suivantes:

*: impression standard

1: impression en écriture normale

2: impression en écriture comprimée (avec marge)

Le préréglage est VIDE (pas de listing);

(voir manuel chapitre 3.4.1).

C79000-B8577-C677-02

4 - 7

4.3 Valeurs par défaut (default) des paramètres

COMPILE.CMD utilise les affectations implicites suivantes:

$BLOC:	B	tous les blocs
$OPT:		
SEQ>MC5 et INT>MC5:	1	
		créer code et écraser données sans interroger
MC5>SEQ et MC5>INT:	VIDE	
		inscrire uniquement symboles
$IMP:	VIDE	pas de listing
$TYPE D'AG:	NON	langage NON

Version ligne de commande

4.3 Valeurs par défaut (default) des paramètres

COMPILE.CMD utilise les affectations implicites suivantes:

$BLOC:	B	tous les blocs
$OPT:		
SEQ>MC5 et INT>MC5:	1	
		créer code et écraser données sans interroger
MC5>SEQ et MC5>INT:	VIDE	
		inscrire uniquement symboles
$IMP:	VIDE	pas de listing
$TYPE D'AG:	NON	langage NON

Annexe 5

Annexe 5

C79000-B8577-C677-02 5 - 1

Messages d'erreurs

* Adresse de DB non valable:
  Adresse de DB trop longue, ou bien caractères contenus non admis (max. de 5 caractères).

* Article faux dans fichier SEQ:
  (A1.SEQ) Fichier de travail séquentiel (A0.SEQ) défectueux (défaut de format)

* Bloc commentaire trop long:
  Répartir ou raccourcir programme (max. de 16 kB).

* Bloc dans fichier intermédiaire incorrect:
  Fichier code intermédiaire (A1.SEQ) est défectueux (défaut de format).
  Générer de nouveau fichier code intermédiaire par la fonction "SEQINT" dans le fichier séquentiel (A0.SEQ).

* Bloc doc. trop long:
  Répartir ou raccourcir documentation programme (max. de 16kB).

* Bloc sans BE:
  Instruction BE (code de fin de bloc) manque.

* Bloc trop long:
  répartir programme (max. de 8kB)

* Caractère de commande non valable:
  Au caractère # succède un caractère de commande non autorisé.

* Caractère non valable:
  CaractèUre non valable existant

Annexe

Messages d'erreurs

* Adresse de DB non valable:
  Adresse de DB trop longue, ou bien caractères contenus non admis (max. de 5 caractères).

* Article faux dans fichier SEQ:
  (A1.SEQ) Fichier de travail séquentiel (A0.SEQ) défectueux (défaut de format)

* Bloc commentaire trop long:
  Répartir ou raccourcir programme (max. de 16 kB).

* Bloc dans fichier intermédiaire incorrect:
  Fichier code intermédiaire (A1.SEQ) est défectueux (défaut de format).
  Générer de nouveau fichier code intermédiaire par la fonction "SEQINT" dans le fichier séquentiel (A0.SEQ).

* Bloc doc. trop long:
  Répartir ou raccourcir documentation programme (max. de 16kB).

* Bloc sans BE:
  Instruction BE (code de fin de bloc) manque.

* Bloc trop long:
  répartir programme (max. de 8kB)

* Caractère de commande non valable:
  Au caractère # succède un caractère de commande non autorisé.

* Caractère non valable:
  CaractèUre non valable existant

5 - 2

C79000-B8577-C677-02

* Code d'opérande non admis:
  Code d'opérande incompatible avec l'opérateur.

* Code d'opérande non défini:
  Code d'opérande non défini dans STEP 5.

* Commentaire trop long:
  Erreur système!
  Commentaire d'instruction trop long (max. de 40 caractères).

* Début de bloc manque:
  Pas de caractère # existant avec désignation de bloc absolue et/ou symbolique.

* Défaut de conversion:
  Erreur système!

* Défaut de conversion:
  Plage numérique limite supérieure

* Défaut de lecture:
  Défaut de disquette, fichier défectueux.

* Écraser fichier source symbolique existant?
  Un fichier source symbolique avec nom de fichier identique existe déjà (fonction GEN-SYM)

* Fichier intermédiaire existe déjà, effacer?
  Un fichier intermédiare avec nom de fichier identique existe déjà.

Annexe

* Code d'opérande non admis:
  Code d'opérande incompatible avec l'opérateur.

* Code d'opérande non défini:
  Code d'opérande non défini dans STEP 5.

* Commentaire trop long:
  Erreur système!
  Commentaire d'instruction trop long (max. de 40 caractères).

* Début de bloc manque:
  Pas de caractère # existant avec désignation de bloc absolue et/ou symbolique.

* Défaut de conversion:
  Erreur système!

* Défaut de conversion:
  Plage numérique limite supérieure

* Défaut de lecture:
  Défaut de disquette, fichier défectueux.

* Écraser fichier source symbolique existant?
  Un fichier source symbolique avec nom de fichier identique existe déjà (fonction GEN-SYM)

* Fichier intermédiaire existe déjà, effacer?
  Un fichier intermédiare avec nom de fichier identique existe déjà.

C79000-B8577-C677-02

5 - 3

* Fichier intermédiaire incorrect:
  Fichier code intermédiaire (A1.SEQ) est défectueux (défaut de format).
  Générer de nouveau fichier code intermédiaire par la fonction "SEQINT" dans le fichier de travail séquentiel (A0.SEQ).

* Fichier symbolique inexistant:
  Fichier symbolique manque pour programmation purement symbolique.

* Fin de segment manque ou segment trop long:
  Caractère de fin de segment (***) ou instruction de génération d'image pour fin de segment (BLD 255) manque, ou bien segment trop long (max. de 255 lignes).

* Format fichier intermédiare non valable:
  Fichier code intermédiaire (A1.SEQ) est défectueux (défaut de format).
  Générer de nouveau fichier code intermédiaire par la fonction "SEQINT" dans le fichier de travail séquentiel (A0.SEQ).

* Format incorrect:
  Format avec default

* Indicatif de fichier intermédiaire incorrect:
  Fichier a été créé par des outils d'un autre état d'édition. Générer de nouveau fichier code intermédiare par la fonction "SEQINT" dans le fichier de travail séquentiel (A0.SEQ).

* Instruction dans bloc non autorisée:
  Instructions dans la "reserve d'opération complémentaire" sont uniquement autorisées dans le FB.

Annexe

* Fichier intermédiaire incorrect:
  Fichier code intermédiaire (A1.SEQ) est défectueux (défaut de format).
  Générer de nouveau fichier code intermédiaire par la fonction "SEQINT" dans le fichier de travail séquentiel (A0.SEQ).

* Fichier symbolique inexistant:
  Fichier symbolique manque pour programmation purement symbolique.

* Fin de segment manque ou segment trop long:
  Caractère de fin de segment (***) ou instruction de génération d'image pour fin de segment (BLD 255) manque, ou bien segment trop long (max. de 255 lignes).

* Format fichier intermédiare non valable:
  Fichier code intermédiaire (A1.SEQ) est défectueux (défaut de format).
  Générer de nouveau fichier code intermédiaire par la fonction "SEQINT" dans le fichier de travail séquentiel (A0.SEQ).

* Format incorrect:
  Format avec default

* Indicatif de fichier intermédiaire incorrect:
  Fichier a été créé par des outils d'un autre état d'édition. Générer de nouveau fichier code intermédiare par la fonction "SEQINT" dans le fichier de travail séquentiel (A0.SEQ).

* Instruction dans bloc non autorisée:
  Instructions dans la "reserve d'opération complémentaire" sont uniquement autorisées dans le FB.

5 - 4

C79000-B8577-C677-02

* Instruction Include emboîtée non admise
  Un fichier intermédiaire, qui a été appelé avec #INCLUDE, contient une #instruction INCLUDE

* Instruction non autorisée:
  Pas d'instruction STEP 5 admise.

* Instruction non définie:
  Pas d'instruction MC5 admise. Fichier programme (ST.S5D) défectueux.

* Instruction pour type d'AG non admise:
  Instruction pour type d'AG donné non autorisée.

* Instruction SYS non autorisée:
  Supprimé.

* Instructions système non autorisées:
  Supprimé.

* Ligne non autorisée:
  Veiller à la suite (caractère de commande) lors de l'entrée de bloc.

* Ligne non traitée:
  type de bloc indéterminé.

* Ligne non valable:
  Veiller à la suite (caractère de commande) lors de l'entrée de bloc.

* Marque non admise:
  Repère de saut à position incorrecte.

Annexe

* Instruction Include emboîtée non admise
  Un fichier intermédiaire, qui a été appelé avec #INCLUDE, contient une #instruction INCLUDE

* Instruction non autorisée:
  Pas d'instruction STEP 5 admise.

* Instruction non définie:
  Pas d'instruction MC5 admise. Fichier programme (ST.S5D) défectueux.

* Instruction pour type d'AG non admise:
  Instruction pour type d'AG donné non autorisée.

* Instruction SYS non autorisée:
  Supprimé.

* Instructions système non autorisées:
  Supprimé.

* Ligne non autorisée:
  Veiller à la suite (caractère de commande) lors de l'entrée de bloc.

* Ligne non traitée:
  type de bloc indéterminé.

* Ligne non valable:
  Veiller à la suite (caractère de commande) lors de l'entrée de bloc.

* Marque non admise:
  Repère de saut à position incorrecte.

C79000-B8577-C677-02

5 - 5

* Marque non définie:
  Repère de saut à la destination de saut donnée (adresse de symbole) non indiqué dans le champ "ADR".

* Marque non valable:
  Repère de saut avec caractères incorrects.

* Marque trop longue:
  Destination de saut (adresse de symbole) inscrite dans le champ "INSTRUCTION", trop longue (max. de 4 caractères).

* Marques identiques:
  Repère de saut existant à plusieurs reprises.

* No. de BIB déjà existant:
  Le caractère de commande #BI est indiqué à plusieurs reprises.

* No. de BIB non valable:
  Numéro de bibliothèque trop long, ou bien caractères contenus non admis (max. de 5 chiffres).

* Nom de bloc déjà existant:
  Le caractère de commande #N est indiqué à plusieurs reprises.

* Nombre de paramètres incorrect:
  Nombre de paramètres formels donné dans FB diffère du nombre de paramètres actuels indiqué après appel de FB (contrôle).

* Numéro de fonction faux:
  Erreur système!

* Opérande non admis:
  aucun opérande autorisé.

Annexe

* Marque non définie:
  Repère de saut à la destination de saut donnée (adresse de symbole) non indiqué dans le champ "ADR".

* Marque non valable:
  Repère de saut avec caractères incorrects.

* Marque trop longue:
  Destination de saut (adresse de symbole) inscrite dans le champ "INSTRUCTION", trop longue (max. de 4 caractères).

* Marques identiques:
  Repère de saut existant à plusieurs reprises.

* No. de BIB déjà existant:
  Le caractère de commande #BI est indiqué à plusieurs reprises.

* No. de BIB non valable:
  Numéro de bibliothèque trop long, ou bien caractères contenus non admis (max. de 5 chiffres).

* Nom de bloc déjà existant:
  Le caractère de commande #N est indiqué à plusieurs reprises.

* Nombre de paramètres incorrect:
  Nombre de paramètres formels donné dans FB diffère du nombre de paramètres actuels indiqué après appel de FB (contrôle).

* Numéro de fonction faux:
  Erreur système!

* Opérande non admis:
  aucun opérande autorisé.

5 - 6

C79000-B8577-C677-02

* Opérande trop long:
  Code d'opérande trop long (max. de 2 caractères)

* Opérateur non indiqué:
  Donnée d'opérateur manquante pour programmation symbolique.

* Opérateur non valable:
  Opérateur non défini dans STEP 5.

* Opérateur trop long:
  (max. de 3 caractères).

* Paramètre absolu trop long:
  Erreur système!

* Paramètre formel déjà existant:
  Le nom de paramètre est attribué à diverses reprises.

* Paramètre formel non défini:
  Nom et type de paramètre ne sont pas définis dans le FB.

* Paramètre formel non valable:
  Le nom de paramètre contient des caractères non valables, ou bien le type de paramètre n'est pas admis.

* Paramètre incorrect:
  Paramètre non valable.

* Paramètre non admis:
  Aucun paramètre autorisé.

Annexe

* Opérande trop long:
  Code d'opérande trop long (max. de 2 caractères)

* Opérateur non indiqué:
  Donnée d'opérateur manquante pour programmation symbolique.

* Opérateur non valable:
  Opérateur non défini dans STEP 5.

* Opérateur trop long:
  (max. de 3 caractères).

* Paramètre absolu trop long:
  Erreur système!

* Paramètre formel déjà existant:
  Le nom de paramètre est attribué à diverses reprises.

* Paramètre formel non défini:
  Nom et type de paramètre ne sont pas définis dans le FB.

* Paramètre formel non valable:
  Le nom de paramètre contient des caractères non valables, ou bien le type de paramètre n'est pas admis.

* Paramètre incorrect:
  Paramètre non valable.

* Paramètre non admis:
  Aucun paramètre autorisé.

5

C79000-B8577-C677-02

5 - 7

* Paramètre trop long (max. de 4 caractères):
  Paramètre formel inscrit dans le champ "INSTRUCTION" trop long (max. de 4 caractères).

* Paramètres actuels non autorisés:
  L'indication de paramètres actuels est autorisée uniquement après un appel de FB.

* Parenthèses incorrectes:
  Fin de parenthèses non compensée (veiller aux niveaux des paranthèses)

* Pas d'indication d'opérande formel:
  Au paramètre actuel donné après l'appel de FB, manque l'indication attribuée de paramètre formel dans le champ "ADR" du FB.

* Pas d'indication de date:
  Après indication d'un type de constante dans champ "INSTRUCTION", l'indication de la valeur de constante manque dans le champ "SYMBOLE D'OPERANDE".

* Pas de code d'opérande indiqué:
  Code d'opérande manque.

* Pas de nom de bloc indiqué:
  Caractère de commande #N et nom de bloc manquent dans le bloc fonctionnel.

* Pas de numéro de BIB indiqué:
  Après #BI, l'indication du numéro de bibliothèque manque.

Annexe

* Paramètre trop long (max. de 4 caractères):
  Paramètre formel inscrit dans le champ "INSTRUCTION" trop long (max. de 4 caractères).

* Paramètres actuels non autorisés:
  L'indication de paramètres actuels est autorisée uniquement après un appel de FB.

* Parenthèses incorrectes:
  Fin de parenthèses non compensée (veiller aux niveaux des paranthèses)

* Pas d'indication d'opérande formel:
  Au paramètre actuel donné après l'appel de FB, manque l'indication attribuée de paramètre formel dans le champ "ADR" du FB.

* Pas d'indication de date:
  Après indication d'un type de constante dans champ "INSTRUCTION", l'indication de la valeur de constante manque dans le champ "SYMBOLE D'OPERANDE".

* Pas de code d'opérande indiqué:
  Code d'opérande manque.

* Pas de nom de bloc indiqué:
  Caractère de commande #N et nom de bloc manquent dans le bloc fonctionnel.

* Pas de numéro de BIB indiqué:
  Après #BI, l'indication du numéro de bibliothèque manque.

5 - 8

C79000-B8577-C677-02

* Pas de paramètre actuel indiqué:
  Après l'appel de FB, l'indication nécessaire de paramètre actuel manque dans le champ "INSTRUCTION".

* Pas de paramètre indiqué:
  Indication de paramètre manque (pour programmation purement absolue).

* Pas de symbole indiqué:
  Symbole d'opérande manque (pour programmation purement symbolique)

* Plus de commentaire que d'instructions

* Position de données incorrecte:
  Pour des constantes, la valeur est à entrer dans le champ "SYMBOLE OPERANDE".

* Symbole incompatible avec paramètre absolu:
  Opérande absolu et opérande symbolique sont assignés différemment dans fichier source LIST et fichier symbolique.

* Symbole non autorisé:
  Instruction autorise aucune indication d'opérande.

* Symbole trop long:
  Erreur système! (max. de 24 caractères)

* Trop de paramètres actuels:
  (40 au max.)

* Trop de paramètres formels:
  (40 au max.)

Annexe

* Pas de paramètre actuel indiqué:
  Après l'appel de FB, l'indication nécessaire de paramètre actuel manque dans le champ "INSTRUCTION".

* Pas de paramètre indiqué:
  Indication de paramètre manque (pour programmation purement absolue).

* Pas de symbole indiqué:
  Symbole d'opérande manque (pour programmation purement symbolique)

* Plus de commentaire que d'instructions

* Position de données incorrecte:
  Pour des constantes, la valeur est à entrer dans le champ "SYMBOLE OPERANDE".

* Symbole incompatible avec paramètre absolu:
  Opérande absolu et opérande symbolique sont assignés différemment dans fichier source LIST et fichier symbolique.

* Symbole non autorisé:
  Instruction autorise aucune indication d'opérande.

* Symbole trop long:
  Erreur système! (max. de 24 caractères)

* Trop de paramètres actuels:
  (40 au max.)

* Trop de paramètres formels:
  (40 au max.)

C79000-B8577-C677-02

5 - 9

* Type d'AG non admis:
  Indication d'un type d'AG non valable.

* Type de bloc indéterminé (symbole non trouvé):
  Pour programmation purement symbolique, la désignation de bloc symbolique manque.

* Type de paramètre incorrect:
  Type de paramètre formel indiqué sous "INSTRUCTION" dans le FB diffère du type de paramètre actuel assigné après appel de FB (contrôle).

* Un seul titre par segment:
  Le caractère de commande #UB est indiqué à plusieurs reprises en début de segment.

* Uniquement après appel de FB:
  Paramètres actuels ne sont admis qu'immediatement après un appel de FB.

* Uniquement autorisé pour blocs AWL:
  bloc

* Uniquement pour blocs fonctionnels:
  Article d'instruction élargi autorisé.

* Zone de paramètre du type d'AG limit supérieure:
  Cette valeur de paramètre n'est pas autorisée pour le type d'AG donné.

Annexe

* Type d'AG non admis:
  Indication d'un type d'AG non valable.

* Type de bloc indéterminé (symbole non trouvé):
  Pour programmation purement symbolique, la désignation de bloc symbolique manque.

* Type de paramètre incorrect:
  Type de paramètre formel indiqué sous "INSTRUCTION" dans le FB diffère du type de paramètre actuel assigné après appel de FB (contrôle).

* Un seul titre par segment:
  Le caractère de commande #UB est indiqué à plusieurs reprises en début de segment.

* Uniquement après appel de FB:
  Paramètres actuels ne sont admis qu'immediatement après un appel de FB.

* Uniquement autorisé pour blocs AWL:
  bloc

* Uniquement pour blocs fonctionnels:
  Article d'instruction élargi autorisé.

* Zone de paramètre du type d'AG limit supérieure:
  Cette valeur de paramètre n'est pas autorisée pour le type d'AG donné.

5 - 10

C79000-B8577-C677-02

Intercalaire

C79000-D8377-C018-01

1

Intercalaire

2

C79000-D8377-C018-01

Anmerkungen/Vorschläge
Ihre Anmerkungen und Vorschläge helfen uns, die Qualität und Benutzbarkeit unserer Dokumentation zu verbessern. Bitte füllen Sie diesen Fragebogen bei der nächsten Gelegenheit aus und senden Sie ihn an Siemens zurück.

Vergessen Sie dabei nicht, Titel und Bestell-Nummer mit Ausgabestand anzugeben.

Titel Ihres Handbuchs: .....

Bestell-Nr. Ihres Handbuchs: ..... Ausgabestand: .....

Geben Sie bitte bei den folgenden Fragen Ihre persönliche Bewertung mit Werten von 1 ≙ gut bis 5 ≙ schlecht an.

1. Entspricht der Inhalt Ihren Anforderungen? ☐
2. Sind die benötigten Informationen leicht zu finden? ☐
3. Sind die Texte leicht verständlich? ☐
4. Entspricht der Grad der technischen Einzelheiten Ihren Anforderungen? ☐
5. Wie bewerten Sie die Qualität der Abbildungen/Tabellen? ☐

Falls Sie auf konkrete Probleme gestoßen sind, erläutern Sie diese bitte in den folgenden Zeilen:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Anmerkungen/Vorschläge
Ihre Anmerkungen und Vorschläge helfen uns, die Qualität und Benutzbarkeit unserer Dokumentation zu verbessern. Bitte füllen Sie diesen Fragebogen bei der nächsten Gelegenheit aus und senden Sie ihn an Siemens zurück.

Vergessen Sie dabei nicht, Titel und Bestell-Nummer mit Ausgabestand anzugeben.

Titel Ihres Handbuchs: .....

Bestell-Nr. Ihres Handbuchs: ..... Ausgabestand: .....

Geben Sie bitte bei den folgenden Fragen Ihre persönliche Bewertung mit Werten von 1 ≙ gut bis 5 ≙ schlecht an.

1. Entspricht der Inhalt Ihren Anforderungen? ☐
2. Sind die benötigten Informationen leicht zu finden? ☐
3. Sind die Texte leicht verständlich? ☐
4. Entspricht der Grad der technischen Einzelheiten Ihren Anforderungen? ☐
5. Wie bewerten Sie die Qualität der Abbildungen/Tabellen? ☐

Falls Sie auf konkrete Probleme gestoßen sind, erläutern Sie diese bitte in den folgenden Zeilen:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2

Absender:

Ihr Name: . . . . . .

Ihre Funktion: . . . . . .

Ihre Firma: . . . . . .

Straße: . . . . . .

PLZ, Ort: . . . . . .

Telefon: . . . . . .

An
Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

76181 Karlsruhe

Bitte kreuzen Sie Ihren Industriezweig an:

- ☐ Automobilindustrie
- ☐ Chemische Industrie
- ☐ Elektroindustrie
- ☐ Nahrungsmittel
- ☐ Leittechnik
- ☐ Maschinenbau
- ☐ Pharmazeutische Industrie
- ☐ Kunststoffverarbeitung
- ☐ Papierindustrie
- ☐ Textilindustrie
- ☐ Transportwesen
- ☐ Petrochemie

Andere . . . . . .

C79000-V8500-C066-01

2

Absender:

Ihr Name: . . . . . .

Ihre Funktion: . . . . . .

Ihre Firma: . . . . . .

Straße: . . . . . .

PLZ, Ort: . . . . . .

Telefon: . . . . . .

An
Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

76181 Karlsruhe

Bitte kreuzen Sie Ihren Industriezweig an:

- ☐ Automobilindustrie
- ☐ Chemische Industrie
- ☐ Elektroindustrie
- ☐ Nahrungsmittel
- ☐ Leittechnik
- ☐ Maschinenbau
- ☐ Pharmazeutische Industrie
- ☐ Kunststoffverarbeitung
- ☐ Papierindustrie
- ☐ Textilindustrie
- ☐ Transportwesen
- ☐ Petrochemie

Andere . . . . . .

C79000-V8500-C066-01

Anmerkungen/Vorschläge
Ihre Anmerkungen und Vorschläge helfen uns, die Qualität und Benutzbarkeit unserer Dokumentation zu verbessern. Bitte füllen Sie diesen Fragebogen bei der nächsten Gelegenheit aus und senden Sie ihn an Siemens zurück.

Vergessen Sie dabei nicht, Titel und Bestell-Nummer mit Ausgabestand anzugeben.

Titel Ihres Handbuchs: .....

Bestell-Nr. Ihres Handbuchs: ..... Ausgabestand: .....

Geben Sie bitte bei den folgenden Fragen Ihre persönliche Bewertung mit Werten von 1 ≙ gut bis 5 ≙ schlecht an.

1. Entspricht der Inhalt Ihren Anforderungen? ☐
2. Sind die benötigten Informationen leicht zu finden? ☐
3. Sind die Texte leicht verständlich? ☐
4. Entspricht der Grad der technischen Einzelheiten Ihren Anforderungen? ☐
5. Wie bewerten Sie die Qualität der Abbildungen/Tabellen? ☐

Falls Sie auf konkrete Probleme gestoßen sind, erläutern Sie diese bitte in den folgenden Zeilen:

. . . . . .

Anmerkungen/Vorschläge
Ihre Anmerkungen und Vorschläge helfen uns, die Qualität und Benutzbarkeit unserer Dokumentation zu verbessern. Bitte füllen Sie diesen Fragebogen bei der nächsten Gelegenheit aus und senden Sie ihn an Siemens zurück.

Vergessen Sie dabei nicht, Titel und Bestell-Nummer mit Ausgabestand anzugeben.

Titel Ihres Handbuchs: .....

Bestell-Nr. Ihres Handbuchs: ..... Ausgabestand: .....

Geben Sie bitte bei den folgenden Fragen Ihre persönliche Bewertung mit Werten von 1 ≙ gut bis 5 ≙ schlecht an.

1. Entspricht der Inhalt Ihren Anforderungen? ☐
2. Sind die benötigten Informationen leicht zu finden? ☐
3. Sind die Texte leicht verständlich? ☐
4. Entspricht der Grad der technischen Einzelheiten Ihren Anforderungen? ☐
5. Wie bewerten Sie die Qualität der Abbildungen/Tabellen? ☐

Falls Sie auf konkrete Probleme gestoßen sind, erläutern Sie diese bitte in den folgenden Zeilen:

. . . . . .

C79000-V8500-C066-01 1

2

Absender:

Ihr Name: . . . . . .

Ihre Funktion: . . . . . .

Ihre Firma: . . . . . .

Straße: . . . . . .

PLZ, Ort: . . . . . .

Telefon: . . . . . .

An
Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

76181 Karlsruhe

Bitte kreuzen Sie Ihren Industriezweig an:

- ☐ Automobilindustrie
- ☐ Chemische Industrie
- ☐ Elektroindustrie
- ☐ Nahrungsmittel
- ☐ Leittechnik
- ☐ Maschinenbau
- ☐ Pharmazeutische Industrie
- ☐ Kunststoffverarbeitung
- ☐ Papierindustrie
- ☐ Textilindustrie
- ☐ Transportwesen
- ☐ Petrochemie

Andere . . . . . .

C79000-V8500-C066-01

2

Absender:

Ihr Name: . . . . . .

Ihre Funktion: . . . . . .

Ihre Firma: . . . . . .

Straße: . . . . . .

PLZ, Ort: . . . . . .

Telefon: . . . . . .

An
Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

76181 Karlsruhe

Bitte kreuzen Sie Ihren Industriezweig an:

- ☐ Automobilindustrie
- ☐ Chemische Industrie
- ☐ Elektroindustrie
- ☐ Nahrungsmittel
- ☐ Leittechnik
- ☐ Maschinenbau
- ☐ Pharmazeutische Industrie
- ☐ Kunststoffverarbeitung
- ☐ Papierindustrie
- ☐ Textilindustrie
- ☐ Transportwesen
- ☐ Petrochemie

Andere . . . . . .

C79000-V8500-C066-01

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1. Do the contents meet your requirements? ☐
2. Is the information you need easy to find? ☐
3. Is the text easy to understand? ☐
4. Does the level of technical detail meet your requirements? ☐
5. Please rate the quality of the graphics/tables: ☐

Additional comments:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1. Do the contents meet your requirements? ☐
2. Is the information you need easy to find? ☐
3. Is the text easy to understand? ☐
4. Does the level of technical detail meet your requirements? ☐
5. Please rate the quality of the graphics/tables: ☐

Additional comments:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

C79000-V8576-C066-01 1

Siemens AG
AUT E 1163

D-76181 Karlsruhe
Federal Republic of Germany

From:

Your Name: . . . . . .

Your Title: . . . . . .

Company Name: . . . . . .

Street: . . . . . .

City, Zip Code: . . . . . .

Phone: . . . . . .

Please check any industry that applies to you:

- ☐ Automotive
- ☐ Chemical
- ☐ Electrical Machinery
- ☐ Food
- ☐ Instrument and Control
- ☐ Nonelectrical Machinery
- ☐ Pharmaceutical
- ☐ Plastic
- ☐ Pulp and Paper
- ☐ Textiles
- ☐ Transportation
- ☐ Petrochemical

Other . . . . . .

C79000-V8576-C066-01

2

Siemens AG
AUT E 1163

D-76181 Karlsruhe
Federal Republic of Germany

From:

Your Name: . . . . . .

Your Title: . . . . . .

Company Name: . . . . . .

Street: . . . . . .

City, Zip Code: . . . . . .

Phone: . . . . . .

Please check any industry that applies to you:

- ☐ Automotive
- ☐ Chemical
- ☐ Electrical Machinery
- ☐ Food
- ☐ Instrument and Control
- ☐ Nonelectrical Machinery
- ☐ Pharmaceutical
- ☐ Plastic
- ☐ Pulp and Paper
- ☐ Textiles
- ☐ Transportation
- ☐ Petrochemical

Other . . . . . .

C79000-V8576-C066-01

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1. Do the contents meet your requirements? ☐
2. Is the information you need easy to find? ☐
3. Is the text easy to understand? ☐
4. Does the level of technical detail meet your requirements? ☐
5. Please rate the quality of the graphics/tables: ☐

Additional comments:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1. Do the contents meet your requirements? ☐
2. Is the information you need easy to find? ☐
3. Is the text easy to understand? ☐
4. Does the level of technical detail meet your requirements? ☐
5. Please rate the quality of the graphics/tables: ☐

Additional comments:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

C79000-V8576-C066-01 1

From:

Your Name: . . . . . .

Your Title: . . . . . .

Company Name: . . . . . .

Street: . . . . . .

City, Zip Code: . . . . . .

Phone: . . . . . .

Siemens AG
AUT E 1163

D-76181 Karlsruhe
Federal Republic of Germany

Please check any industry that applies to you:

- ☐ Automotive
- ☐ Chemical
- ☐ Electrical Machinery
- ☐ Food
- ☐ Instrument and Control
- ☐ Nonelectrical Machinery
- ☐ Pharmaceutical
- ☐ Plastic
- ☐ Pulp and Paper
- ☐ Textiles
- ☐ Transportation
- ☐ Petrochemical

Other . . . . . .

C79000-V8578-C066-01

2

From:

Your Name: . . . . . .

Your Title: . . . . . .

Company Name: . . . . . .

Street: . . . . . .

City, Zip Code: . . . . . .

Phone: . . . . . .

Siemens AG
AUT E 1163

D-76181 Karlsruhe
Federal Republic of Germany

Please check any industry that applies to you:

- ☐ Automotive
- ☐ Chemical
- ☐ Electrical Machinery
- ☐ Food
- ☐ Instrument and Control
- ☐ Nonelectrical Machinery
- ☐ Pharmaceutical
- ☐ Plastic
- ☐ Pulp and Paper
- ☐ Textiles
- ☐ Transportation
- ☐ Petrochemical

Other . . . . . .

C79000-V8578-C066-01

Remarques/suggestions
Vos remarques et suggestions nous permettent d'améliorer la qualité générale de notre documentation. C'est pourquoi nous vous serions reconnaissants de compléter et de renvoyer ces formulaires à Siemens.

N'oubliez pas d'indiquer le titre, le numéro de référence et l'édition de votre manuel.

Titre de votre manuel : .....

Numéro de référence de votre manuel : ..... Edition : .....

Répondez aux questions suivantes en donnant votre évaluation comprise entre 1 pour très bien et 5 pour très mauvais.

1. Le contenu du manuel répond-il à votre attente ? ☐
2. Les informations requises peuvent-elles facilement être trouvées ? ☐
3. Le texte est-il compréhensible ? ☐
4. Le niveau des détails techniques répond-il à votre attente ? ☐
5. Quelle évaluation attribuez-vous aux figures et tableaux ? ☐

Les lignes suivantes vous permettent d'exposer des problèmes concrets que vous auriez éventuellement rencontrés :

. . . . . .

Remarques/suggestions
Vos remarques et suggestions nous permettent d'améliorer la qualité générale de notre documentation. C'est pourquoi nous vous serions reconnaissants de compléter et de renvoyer ces formulaires à Siemens.

N'oubliez pas d'indiquer le titre, le numéro de référence et l'édition de votre manuel.

Titre de votre manuel : .....

Numéro de référence de votre manuel : ..... Edition : .....

Répondez aux questions suivantes en donnant votre évaluation comprise entre 1 pour très bien et 5 pour très mauvais.

1. Le contenu du manuel répond-il à votre attente ? ☐
2. Les informations requises peuvent-elles facilement être trouvées ? ☐
3. Le texte est-il compréhensible ? ☐
4. Le niveau des détails techniques répond-il à votre attente ? ☐
5. Quelle évaluation attribuez-vous aux figures et tableaux ? ☐

Les lignes suivantes vous permettent d'exposer des problèmes concrets que vous auriez éventuellement rencontrés :

. . . . . .

C79000-V8577-C066-01 1

2

Expéditeur :

Vos Nom : . . . . . .

Fonction : . . . . . .

Entreprise : . . . . . .

Rue : . . . . . .

Code postal : . . . . . .

Ville : . . . . . .

Pays : . . . . . .

Téléphone : . . . . . .

Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

D-76181 Karlsruhe
République Fédérale d'Allemagne

Indiquez votre secteur industriel :

- ☐ Industrie automobile
- ☐ Industrie chimique
- ☐ Industrie électrique
- ☐ Industrie alimentaire
- ☐ Contrôle/commande
- ☐ Construction mécanique
- ☐ Industrie pharmaceutique
- ☐ Traitement des matières plastiques
- ☐ Industrie du papier
- ☐ Industrie textile
- ☐ Transports
- ☐ Pétrochimie

Autres . . . . . .

C79000-V8577-C066-01

2

Expéditeur :

Vos Nom : . . . . . .

Fonction : . . . . . .

Entreprise : . . . . . .

Rue : . . . . . .

Code postal : . . . . . .

Ville : . . . . . .

Pays : . . . . . .

Téléphone : . . . . . .

Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

D-76181 Karlsruhe
République Fédérale d'Allemagne

Indiquez votre secteur industriel :

- ☐ Industrie automobile
- ☐ Industrie chimique
- ☐ Industrie électrique
- ☐ Industrie alimentaire
- ☐ Contrôle/commande
- ☐ Construction mécanique
- ☐ Industrie pharmaceutique
- ☐ Traitement des matières plastiques
- ☐ Industrie du papier
- ☐ Industrie textile
- ☐ Transports
- ☐ Pétrochimie

Autres . . . . . .

C79000-V8577-C066-01

Remarques/suggestions
Vos remarques et suggestions nous permettent d'améliorer la qualité générale de notre documentation. C'est pourquoi nous vous serions reconnaissants de compléter et de renvoyer ces formulaires à Siemens.

N'oubliez pas d'indiquer le titre, le numéro de référence et l'édition de votre manuel.

Titre de votre manuel : .....

Numéro de référence de votre manuel : ..... Edition : .....

Répondez aux questions suivantes en donnant votre évaluation comprise entre 1 pour très bien et 5 pour très mauvais.

1. Le contenu du manuel répond-il à votre attente ? ☐
2. Les informations requises peuvent-elles facilement être trouvées ? ☐
3. Le texte est-il compréhensible ? ☐
4. Le niveau des détails techniques répond-il à votre attente ? ☐
5. Quelle évaluation attribuez-vous aux figures et tableaux ? ☐

Les lignes suivantes vous permettent d'exposer des problèmes concrets que vous auriez éventuellement rencontrés :

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Remarques/suggestions
Vos remarques et suggestions nous permettent d'améliorer la qualité générale de notre documentation. C'est pourquoi nous vous serions reconnaissants de compléter et de renvoyer ces formulaires à Siemens.

N'oubliez pas d'indiquer le titre, le numéro de référence et l'édition de votre manuel.

Titre de votre manuel : .....

Numéro de référence de votre manuel : ..... Edition : .....

Répondez aux questions suivantes en donnant votre évaluation comprise entre 1 pour très bien et 5 pour très mauvais.

1. Le contenu du manuel répond-il à votre attente ? ☐
2. Les informations requises peuvent-elles facilement être trouvées ? ☐
3. Le texte est-il compréhensible ? ☐
4. Le niveau des détails techniques répond-il à votre attente ? ☐
5. Quelle évaluation attribuez-vous aux figures et tableaux ? ☐

Les lignes suivantes vous permettent d'exposer des problèmes concrets que vous auriez éventuellement rencontrés :

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2

Expéditeur :

Vos Nom : . . . . . .

Fonction : . . . . . .

Entreprise : . . . . . .

Rue : . . . . . .

Code postal : . . . . . .

Ville : . . . . . .

Pays : . . . . . .

Téléphone : . . . . . .

Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

D-76181 Karlsruhe
République Fédérale d'Allemagne

Indiquez votre secteur industriel :

- ☐ Industrie automobile
- ☐ Industrie chimique
- ☐ Industrie électrique
- ☐ Industrie alimentaire
- ☐ Contrôle/commande
- ☐ Construction mécanique
- ☐ Industrie pharmaceutique
- ☐ Traitement des matières plastiques
- ☐ Industrie du papier
- ☐ Industrie textile
- ☐ Transports
- ☐ Pétrochimie

Autres . . . . . .

C79000-V8577-C066-01

2

Expéditeur :

Vos Nom : . . . . . .

Fonction : . . . . . .

Entreprise : . . . . . .

Rue : . . . . . .

Code postal : . . . . . .

Ville : . . . . . .

Pays : . . . . . .

Téléphone : . . . . . .

Siemens AG
AUT E 1163
Östl. Rheinbrückenstraße 50

D-76181 Karlsruhe
République Fédérale d'Allemagne

Indiquez votre secteur industriel :

- ☐ Industrie automobile
- ☐ Industrie chimique
- ☐ Industrie électrique
- ☐ Industrie alimentaire
- ☐ Contrôle/commande
- ☐ Construction mécanique
- ☐ Industrie pharmaceutique
- ☐ Traitement des matières plastiques
- ☐ Industrie du papier
- ☐ Industrie textile
- ☐ Transports
- ☐ Pétrochimie

Autres . . . . . .

C79000-V8577-C066-01