Handbuch
Steuerung
Ein- und Ausgänge
Ablaufprogrammierung

Andras Steuerungssysteme GmbH
Westerstraße 93A
28199 Bremen

ANDRAS SYSTEM

Wir finden den Weg

- Irrtümer und Änderungen vorbehalten -

Ergänzungen / Änderungen

- Dokumentation der neuen Funktion „Alle Texte drucken“
- Ergänzung im Kapitel „Betextung ablaufabhängiger Merker“
- Dokumentation der neuen SPS-Befehle 209, 249, 250, 251
- Sonderbefehl 10 für SPS-Befehl 8
- Bedeutung der globalen Merker 1000 und höher
- SPS-Sicherheitsräume
- Ergänzung im Kapitel Betextungen
- Zuordnungstabelle
- Sps-Befehle 211, 212 und 216
- Ergänzung der möglichen Operanden für SPS-Opcode 8
- Ergänzung der globalen Merker
- Sps-Befehle 180 bis 186 und 189
- Externe Daten 2100 bis 2299
- Ergänzung bei den Befehlen 4, 104, 105
- Sps-Befehl 248
- Ergänzung bei den Befehlen 200 und 201
- SPS-Befehl 207
- SPS-Befehl 35 und 85, Operand 25
- SPS-Befehl 35 und 85, Operand 26
- SPS-Befehl 109 geändert
- SPS-Befehl 46 geändert
- SPS-Befehl 35 und 85, Operand 27 bis 32
- Externe Daten ab 2000
- Änderung im Kapitel Tabelle der externen Zeiten
- SPS-Befehle 188, 208, 209
- SPS-Befehl 35 und 85, Operand 33
- Neues Kapitel BDE
- Ergänzung in den Kapiteln zu den Achsbefehlen
- SPS-Befehl 247
- Konfiguration von CAN-Modulen im EA-Strang
- SPS-Befehle 116,117,126,127,210
- SPS-Befehl 215
- SPS-Befehl 114,115

OAL-HB-EAABPR-PT-101.09-DE

Inhaltsverzeichnis

Ergänzungen / Änderungen ... 2
Inhaltsverzeichnis ... 3
Vorwort ... 8
Ein- und Ausgänge ... 9
Bedeutung von Sekundäreingängen ... 9
Betextung von Ein- und Ausgängen ... 10
Konfiguration von Ein- und Ausgängen ... 11
Testen von Ein- und Ausgängen ... 14
Ablaufprogrammierung, grundlegende Arbeitsweise ... 16
Was ist ein Ablauf? ... 16
Wie werden Abläufe gestartet? ... 16
Verteilung der Rechenzeit ... 16
Generelle Befehlssyntax einer Zeile im Ablauf ... 16
Strukturänderungen innerhalb eines Ablaufes ... 17
Notaussprung ... 17
Globale und lokale Merker ... 17
Ablaufprogrammierung, Befehlsbeschreibung ... 18
Opcode 0: NOP ... 18
Opcode 1: Fertig ... 18
Opcode 2: Melde Störung ... 19
Opcode 3: Melde Bedienerführung ... 20
Opcode 4: Notstop Achse ... 20
Opcode 5: Lade Stepfreigabe ... 20
Opcode 6: Pointer Externe Daten ... 21
Opcode 7: System-Auftrag ... 21
Opcode 8: System-Funktion ... 21
Opcode 9: Lade Wartebedingung ... 22
Opcode 10: Warte bis Eingang low ... 22
Opcode 11: Warte bis Eingang high ... 22
Opcode 12: Test Eingang low ... 23
Opcode 13: Test Eingang high ... 23
Opcode 14: Externen Timer stoppen ... 23
Opcode 15: Externen Timer starten ... 24
Opcode 16: Warte Eingang low / Zeit ... 24
Opcode 17: Warte Eingang high / Zeit ... 24
Opcode 18: Lade Eingangsgruppe (Byte) in Zähler ... 25
Opcode 19: Lade Eingangsgruppe (Byte) in Akku ... 25
Opcode 20: Warte bis Ausgang low ... 26
Opcode 21: Warte bis Ausgang high ... 26
Opcode 22: Test Ausgang low ... 26
Opcode 23: Test Ausgang high ... 27
Opcode 24: Ausgang löschen ... 27
Opcode 25: Ausgang setzen ... 27
Opcode 26: Warte Ausgang low / Zeit ... 28
Opcode 27: Warte Ausgang high / Zeit ... 28
Opcode 28: Lade Zähler in Ausgangsgruppe (Byte) ... 29
Opcode 29: Lade Akku in Ausgangsgruppe (Byte) ... 29
Opcode 30: Zähler = Zähler - Operand ... 30
Opcode 31: Zähler = Zähler + Operand ... 30
Opcode 32: Sprung wenn Zähler = 0 ... 30
Opcode 33: Sprung wenn Zähler <> 0 ... 31
Opcode 34: Zähler laden mit Operand ... 31
Opcode 35: Zähler laden Sonderfunktion ... 31
Opcode 36: Teste Zähler < Operand ... 32
Opcode 37: Teste Zähler > Operand ... 32
Opcode 38: Teste Zähler = Operand ... 32

Opcode 39: Teste Zähler <> Operand 33
Opcode 40: Warte bis Timer = 0 33
Opcode 41: Lade Timer und warte 33
Opcode 42: Sprung wenn Timer = 0 34
Opcode 43: Sprung wenn Timer <> 0 34
Opcode 44: Lade Timer in 1/100 Sekunden 34
Opcode 45: Lade Timer in Sekunden 35
Opcode 46: Lade Timer aus Tabelle 35
Opcode 47: Lade Timer aus Tabelle 35
Opcode 48: Kopplung auf Startbedingung setzen 36
Opcode 49: Kopplung auf Startbedingung löschen 36
Opcode 50: Warte bis Auftrag = 0 36
Opcode 51: Warte bis Auftrag <> 0 37
Opcode 52: Test Auftrag = 0 37
Opcode 53: Lösche Auftrag 38
Opcode 54: Auftragspointer laden 38
Opcode 55: Auftrag laden 38
Opcode 56: Teste Auftrag < Operand 39
Opcode 57: Teste Auftrag > Operand 39
Opcode 58: Teste Auftrag = Operand 39
Opcode 59: Teste Auftrag <> Operand 40
Opcode 60: Warte bis Step = 0 40
Opcode 61: Warte bis Step <> 0 40
Opcode 62: Teste Step = 0 41
Opcode 63: Sprung wenn Step <> 0 41
Opcode 64: Steppointer laden 41
Opcode 65: Step laden 42
Opcode 66: Teste Step < Operand 42
Opcode 67: Teste Step > Operand 42
Opcode 68: Teste Step = Operand 43
Opcode 69: Teste Step <> Operand 43
Opcode 70: Sprung wenn Nein 43
Opcode 71: Sprung wenn Ja 44
Opcode 72: Sprung unbedingt 44
Opcode 73: Sprung wenn Störung 44
Opcode 74: Sprung wenn Notaus 45
Opcode 75: Sprungziel für Notaus 45
Opcode 76: Sprung wenn eigene Störung 45
Opcode 77: Sprung wenn Startbedingung 46
Opcode 78: Sprung wenn Folgeauftrag 46
Opcode 79: Unterprogrammsprung 46
Opcode 80: Akku = Akku – Operand 47
Opcode 81: Akku = Akku + Operand 47
Opcode 82: Akku = Akku / Operand 47
Opcode 83: Akku = Akku * Operand 47
Opcode 84: Akku laden mit Operand 48
Opcode 85: Akku laden Sonderfunktion 48
Opcode 86: Teste Akku < Operand 48
Opcode 87: Teste Akku > Operand 49
Opcode 88: Teste Akku = Operand 49
Opcode 89: Teste Akku <> Operand 49
Opcode 90: Warte bis Merker low 50
Opcode 91: Warte bis Merker high 50
Opcode 92: Teste Merker low 50
Opcode 93: Teste Merker high 51
Opcode 94: Merker löschen 51
Opcode 95: Merker setzen 51
Opcode 96: Warte Merker low / Zeit 52
Opcode 97: Warte Merker high / Zeit 52
Opcode 98: Teste Bit im Zähler 52

Opcode 99: Teste Bit im Akku ... 53
Opcode 100: Reset Ablauf Operand ... 53
Opcode 101: Teste Achse vor ... 54
Opcode 102: Teste Achse rück ... 54
Opcode 103: Teste Achse Stillstand ... 54
Opcode 104: Stop Achse ... 55
Opcode 105: Reset Achse ... 55
Opcode 106: Positioniersperre setzen ... 55
Opcode 107: Positioniersperre löschen ... 56
Opcode 108: Vergleiche Gruppe Achse/Ablauf ... 56
Opcode 109: Status Achse in Operand ... 56
Opcode 110: Oder Eingang low ... 57
Opcode 111: Oder Eingang high ... 57
Opcode 114: Pseudo-Eingang löschen ... 58
Opcode 115: Pseudo-Eingang setzen ... 58
Opcode 112: Und Eingang low ... 59
Opcode 113: Und Eingang high ... 59
Opcode 116: Analogwert in Zähler laden ... 60
Opcode 117: Analogwert in Akku laden ... 60
Opcode 118: Lade Eingangsgruppe (Word) in Zähler ... 61
Opcode 119: Lade Eingangsgruppe (Word) in Akku ... 61
Opcode 120: Oder Ausgang low ... 62
Opcode 121: Oder Ausgang high ... 62
Opcode 122: Und Ausgang low ... 63
Opcode 123: Und Ausgang high ... 63
Opcode 124: Lösche wenn ja, setzte wenn nein ... 64
Opcode 125: Setze wenn ja, lösche wenn nein ... 64
Opcode 126: Analogwert setzen aus Zähler ... 65
Opcode 127: Analogwert setzen aus Akku ... 65
Opcode 128: Lade Zähler in Ausgangsgruppe (Word) ... 66
Opcode 129: Lade Akku in Ausgangsgruppe (Word) ... 66
Opcode 180: Akku komplementieren ... 67
Opcode 181: Akku negieren ... 67
Opcode 182: Sprung wenn Akku = 0 ... 67
Opcode 183: Sprung wenn Akku <> 0 ... 68
Opcode 184: Akku = Akku „and“ Operand ... 68
Opcode 185: Akku = Akku „or“ Operand ... 68
Opcode 186: Akku = Akku „xor“ Operand ... 69
Opcode 188: BCD -> Binär ... 69
Opcode 189: Graycodekonvertierung ... 69
Opcode 190: Warte bis globaler Merker low ... 70
Opcode 191: Warte bis globaler Merker high ... 70
Opcode 192: Teste globaler Merker low ... 70
Opcode 193: Teste globaler Merker high ... 71
Opcode 194: Globalen Merker löschen ... 71
Opcode 195: Globalen Merker setzen ... 71
Opcode 196: Warte globaler Merker low / Zeit ... 72
Opcode 197: Warte globaler Merker high / Zeit ... 72
Opcode 198: Eigene Störung löschen ... 72
Opcode 199: Merker in allen Abläufen löschen ... 73
Opcode 200: Eigenen WCB schieben ... 73
Opcode 201: Eigenen WCB ringförmig schieben ... 73
Opcode 202: Meldung in Logdatei schreiben ... 74
Opcode 203: WCBs rückwärts durchsuchen ... 74
Opcode 204: WCB-Pointer auf eigenen Platz ... 74
Opcode 205: WCB-Pointer minus Operand ... 75
Opcode 206: WCB-Pointer plus Operand ... 75
Opcode 207: Eigenen WCB auf Platz (Operand) schieben ... 75
Opcode 208: BDE-Wert addieren ... 75
Opcode 209: Zykluszähler plus 1 ... 76

Opcode 210: Achse positionieren ... 76
Opcode 211: Achse positionieren ... 77
Opcode 212: Achse Massabfrage ... 78
Opcode 215: Pointer WCA auf Feld laut DBH ... 78
Opcode 216: Pointer WCB auf Feld laut DBH ... 78
Opcode 247: Bild-Refresh ... 79
Opcode 248: Init Abläufe Gruppe ... 79
Opcode 249: Init Abläufe Gruppe Konfig ... 79
Opcode 250: Reset Abläufe Gruppe ... 80
Opcode 251: System-Auftrag ... 80
Opcode 252: Öffne Editor ... 80
Opcode 253: Projektiere ASI ... 80
Opcode 254: Sicherheitsraum testen ... 81
Opcode 255: Reset System ... 81
Editierung der Ablaufprogrammierung ... 82
Verwaltung der SPS ... 82
Anwahl von Abläufen zur Editierung ... 83
Editierung eines einzelnen Ablaufs ... 86
Zeilen einfügen und Zeilen löschen ... 86
Die eigentliche Editierung ... 87
Die Hilfsfunktion Liste ... 87
Suchen innerhalb eines Ablaufs ... 88
Ersetzen innerhalb eines Ablaufs ... 88
Drucken eines einzelnen Ablaufs ... 88
Betextungen ... 89
Überblick der möglichen Betextungen ... 89
Betextung von ablaufabhängigen Merken ... 90
Funktion „Alle Texte drucken“ ... 90
Onlinehilfe der Opcodes ... 91
Testen der Ablaufprogrammierung ... 92
Verbundtest ... 92
Einzeltest eines Ablaufs ... 93
Testen von globalen Merkern ... 94
Zusatzdaten der Ablaufprogrammierung ... 96
Tabelle der externen Zeiten ... 97
Benutzerdefinierte Softkeys ... 98
Aufrufbare Editoren ... 100
Überwachungen ... 101
SPS-Sicherheitsräume ... 102
Zuordnungstabelle ... 103
Externe Daten ... 104
Anwenderspezifische externe Daten ... 104
Fixe externe Daten 2000 ... 104
Fixe externe Daten 2100 bis 2299 ... 104
Betriebsdaten BDE ... 105
Anhang ... 106
Liste der möglichen Operanden für den SPS-Opcode 7 im OALSYS-System ... 106
Liste der möglichen Operanden für den SPS-Opcode 8 im OALSYS-System ... 106
Liste der möglichen Operanden für den Opcode 251 im OALSYS-System ... 106
Liste der möglichen Operanden für den SPS-Opcode 35 und 85 ... 107
Liste der Bedeutung der globalen Merker 1000 und höher ... 108
Liste der Bedeutung der externen Daten ab 2000 ... 108

Abbildungsverzeichnis

Abbildung 1: Textbildschirm für Eingangsbezeichnungen 10
Abbildung 2: Konfiguration von Ein- und Ausgängen 11
Abbildung 3: Testen von Ein- und Ausgängen 14
Abbildung 4: Test der Ein- und Ausgänge mit eingeblendeter Liste 15
Abbildung 5: Verwaltung der Abläufe 82
Abbildung 6: Anwahl zur Editierung von Abläufen 83
Abbildung 7: Hilfsmenü in der Anwahl der Ablaufprogrammierung 84
Abbildung 8: Beispiel zum Inhaltsverzeichnis der Ablaufprogrammierung 84
Abbildung 9: Beispiel für die Suche über alle Abläufe 85
Abbildung 10: Beispiel Editierung eines einzelnen Ablaufs 86
Abbildung 11: Betextung ablaufabhängiger Merker 90
Abbildung 12: Onlinehilfe zu den SPS-Befehlen 91
Abbildung 13: Verbundtest 92
Abbildung 14: Interne Daten des Ablaufs im Einzeltest 93
Abbildung 15: Anzeige des Ausgangsstatus im Einzeltest 93
Abbildung 16: Inaktiver Cursor bei laufendem Ablauf im Einzeltest 94
Abbildung 17: Übersicht der globalen Merker 94
Abbildung 18: Übersicht aller globalen Merker 95
Abbildung 19: Beispiel für externe Zeiten 97
Abbildung 20: Editierung von Automatik-Softkeys 99
Abbildung 21: Beispiel aufrufbarer Editoren für die Automatikmaske 100
Abbildung 22: Beispiel für Überwachungen 101
Abbildung 23: SPS-Sicherheitsräume 102
Abbildung 24: Zuordnungstabelle der SPS 103
Abbildung 25: Editierung der fixen externen Daten ab 2100 105

Vorwort

Diese Dokumentation beschäftigt sich mit den Ein- und Ausgängen einer Steuerung, sowie mit der Ablaufprogrammierung des Systems. In den Steuerungen der Andras Steuerungssysteme GmbH bildet die Ablaufprogrammierung die Funktionen der klassischen SPS nach, kann aber noch erheblich mehr. Diese Dokumentation wendet sich in erster Linie an die Programmierer einer Maschine. Das System ist grundsätzlich frei konfigurierbar, so dass auf die Nennung von Menüpunkten in dieser Dokumentation verzichtet wird. Wo die einzelnen Masken und Funktionen zu finden sind, ist von den Programmierern der Steuerung frei einstellbar und aus diesem Grund vom Lieferanten der Steuerung nicht dokumentierbar.

Auch der Aufbau und das Layout der einzelnen Masken kann verändert werden. Wie dies geschieht ist in der Dokumentation „Handbuch Editor-Konzept“ beschrieben. Alle Funktionen und Masken, die hier beschrieben werden, entsprechen in ihrem Layout der Version, die von der Andras Steuerungssysteme GmbH als Auslieferungszustand festgelegt worden ist.

Ein- und Ausgänge

Die Steuerungen der Andras Steuerungssysteme GmbH können prinzipiell mit 1024 Eingängen und 1024 Ausgängen arbeiten. Dabei besteht die Möglichkeit, die logische Nummerierung der Ein- und Ausgänge frei auf die physikalische Anordnung zu zu weisen. Alle Masken der Steuerung sowie alle Programmierebenen arbeiten grundsätzlich mit der Nummerierung des logischen Systems.

Bedeutung von Sekundäreingängen

Das System bietet grundsätzlich die Möglichkeit, zusätzlich zu den 1024 Eingängen weitere 1024 Sekundäreingänge zu verwenden. Sekundäreingänge sind die Möglichkeit, die Ausfallzeiten von Maschinen zu verringern, da die Maschine auch dann weiter arbeiten kann, wenn ein Eingang physikalisch ausfällt.

Werden Sekundäreingänge verwendet, so ist das Resultat einer Eingangsabfrage grundsätzlich das Oder-Ergebnis aus dem Eingang und dem entsprechenden Sekundäreingang. Werden Sekundäreingänge verwendet, so sind die Ergebnisse aus dem Eingang und dem Sekundäreingang nicht einzelnd abfragbar! Die logische Nummer eines Sekundäreingangs muss in ihrer Funktionalität der logischen Nummer des dazu gehörenden Eingangs entsprechen.

Die nachfolgende Tabelle stellt dar, wie das Abfrageergebnis eines Eingangs aussieht, wenn Sekundäreingänge verwendet werden:

Status des Eingangs	Status des Sekundäreingangs	Abfrageergebnis
LOW	LOW	LOW
HIGH	LOW	HIGH
LOW	HIGH	HIGH
HIGH	HIGH	HIGH

Betextung von Ein- und Ausgängen

Ein- und Ausgänge können frei betextet werden, dabei kann jedem Eingang und jedem Ausgang ein Text zugewiesen werden. Die Texte werden in der zentralen Textdatei des Systems gespeichert. Als Textschlüssel wird `SPSEIN.....+Eingangsnummer`, bzw. `SPSAUS.....+Ausgangsnummer` verwendet, also beispielweise `SPSEIN.....0` für den logischen Eingang 0 und `SPSAUS.....13` für den logischen Ausgang 13.

Die entsprechenden Texte können an verschiedenen Stellen des Systems eingegeben und bearbeitet werden. So ist die Bearbeitung sowohl in der Konfigurationsmaske, als auch in einem separaten Textbildschirm möglich. Der separate Textbildschirm wird in der nachfolgenden Abbildung am Beispiel der Eingangsbezeichnungen dargestellt:

Abbildung 1: Textbildschirm für Eingangsbezeichnungen

Diese Maske ermöglicht die Editierung der Eingangstexte. Angezeigt werden die Eingangsnummer und der dazugehörige Text. Um eine neue Nummer anzulegen, muss im Feld „Nr.“ lediglich die gewünschte neue Nummer eingegeben werden. Um einen Text zu löschen, wird der Softkey „Satz Entf.“ Verwendet, der sich in der zweiten Softkeyleiste befindet, die über den Softkey „<<“ erreichbar ist. Die gesamten Eingangsbezeichnungen können über den Softkey „Drucken“ in eine Textdatei auf der Diskette ausgegeben werden. Auch ein Export in eine Datei, die mit Excel lesbar ist, ist über die Funktion „Drucken“ erreichbar. Die Editierung der Ausgangstexte geschieht analog nach dem gleichen Verfahren.

Sekundäreingänge erhalten den gleichen Text wie die dazu gehörenden Eingänge und können nicht mit einer gesonderten Betextung versehen werden.

Konfiguration von Ein- und Ausgängen

In der Konfiguration der Ein- und Ausgänge wird festgelegt, zu welcher logischen Nummer welches physikalische Bauelement gehört. Es handelt sich dabei also um die Übersetzungstabelle, mit deren Hilfe das Steuerungssystem feststellen kann, welche physikalische Ein- bzw. Ausgang gemeint ist. Die Konfiguration erfolgt getrennt nach Eingängen, Sekundäreingängen und Ausgängen, folgt aber stets dem gleichen Schema und hat die gleiche Editierungsmaske.

EINGAENGE ZUWEISEN

LFD.NR.	MODUL	C	ADR.	BIT	BEZEICHNUNG
0	17	0	0	0	Kein Not Aus
1	17	0	0	1	Gruppe 0: Frei
2	17	0	0	2	Gruppe 2: Z-Achse im SGM-Berei
3	17	0	0	3	Gruppe 2: Y-Achse abgesenkt
4	17	0	0	4	Gruppe 2: Kein Einrichtbetrieb
5	17	0	0	5	Gruppe 2: SGM-Schutztuer gesch
6	17	0	0	6	Gruppe 2: Betrieb "Mit Handlin
7	17	0	0	7	Gruppe 0: Frei
8	103	0	1	0	Gruppe 2: Werkzeug geschlossen
9	103	0	1	1	Gruppe 2: Start Handling
10	103	0	1	2	Gruppe 2: Auswerfer "ist zurue
11	103	0	1	3	Gruppe 2: Auswerfer "ist vorn"
12	103	0	2	0	Gruppe 2: Kern in Stellung HG

<< | DRUCKEN | SEITE- | SEITE+

Abbildung 2: Konfiguration von Ein- und Ausgängen

Um eine neue logische Nummer an zu legen, wird die entsprechende Nummer einfach im Feld „Lfd.Nr.“ eingegeben. Um eine Zuordnung zu löschen, wird der Softkey „Satz.Entf.“ verwendet, der sich in der zweiten Softkeyleiste befindet. Die zweite Softkeyleiste erreicht man über den Softkey „<<“. Mit dem Softkey „Drucken“ besteht die Möglichkeit, die gesamte Zuordnung in eine Textdatei auf Diskette aus zu geben.

Zu jeder logischen Nummer gehören Angaben, die die genaue physikalische Position des Ein- bzw. Ausgangs festlegen. Die Bedeutung der einzelnen Felder eines Datensatzes wird in der nachfolgenden Tabellen exemplarisch an den Eingängen erläutert:

Feld	Bedeutung
Lfd.Nr.	Logische Nummer des Eingangs
Modul	Modul, an dass der Eingang angeschlossen ist. Die möglichen Modulnummern sind in der nachfolgenden Tabelle erläutert.
C	Kanal innerhalb des Moduls
Adr.	Adresse innerhalb des Moduls (z.B. Nummer der ASI-Insel innerhalb eines ASI-Kopplers)
Bit	Bit innerhalb der Adresse
Bezeichnung	Betextung des Eingangs. Hier wird der Text angezeigt, der – wie im vorherigen Kapitel beschrieben – dem Eingang zugewiesen worden ist. Der Text kann an dieser Stelle auch eingegeben und bearbeitet werden. Ist für den Eingang kein Text vorhanden, steht hier der entsprechende Textschlüssel für die Textdatei.

Eine zentrale Bedeutung innerhalb dieser Konfiguration kommt dem Modul zu. Hier wird die Moduladresse eingetragen. Folgende Kombinationen sind denkbar:

Modul	Bedeutung
0	NICHT MÖGLICH!
1..15	EA-Modul im TSM-Strang, Adresse des entsprechenden Moduls laut Drehschalter
16	Ausgänge auf der CPU
17	Eingänge auf der CPU
100 ...	Der Ein- bzw. Ausgang ist an einen seriellen ASI-Koppler angeschlossen. Die Modulnummer setzt sich zusammen aus der Zahl 100 und der Adresse des TSM-Bausteins, an den der ASI-Koppler angeschlossen ist
200 ...	Der Ein- bzw. Ausgang ist an einen Bitbus-ASI-Koppler angeschlossen. Die Modulnummer entspricht der Knotenadresse des Kopplers.
254	Die Modulnummer 254 ist für EA-Module reserviert, die im CAN-Bus liegen. In einem solchen Fall wird im Feld Adr. die CAN-Adresse des Moduls hinterlegt und es sind die Bits 0 bis 63 möglich. **Dieses Verfahren kann nur für FESTO-Ventilinseln verwendet werden!**

In der nachfolgenden Tabelle sind zur Verdeutlichung der Konfiguration noch einmal ein paar Beispiele aufgeführt:

Lfd.Nr	Modul	C	Adr.	Bit
Eingang mit der logischen Nummer 0, direkt an der CPU angeschlossen				
0	17	0	0	0
Eingang mit der logischen Nummer 23, angeschlossen an einen seriellen ASI-Koppler. Innerhalb des Koppler mit der Adresse 7 versehen, innerhalb der Adresse Bit 1. Der Koppler ist an Kanal 1 eines RS232-Modul angeschlossen. Das RS232-Modul hat im TSM-Strang die Adresse 3.				
23	103	1	7	1
Eingang mit der logischen Nummer 105, angeschlossen an einen seriellen ASI-Koppler. Innerhalb des Koppler mit der Adresse 1 versehen, innerhalb der Adresse Bit 0. Der Koppler ist an Kanal 0 eines RS232-Modul angeschlossen. Das RS232-Modul hat im TSM-Strang die Adresse 3.				
105	103	0	1	0
Eingang mit der logischen Nummer 154, angeschlossen an einen Bitbus-ASI-Koppler. Innerhalb des Koppler mit der Adresse 5 versehen, innerhalb der Adresse Bit 3. Der Koppler hat die Knotennummer 201.				
154	201	0	5	3
Eingang mit der logischen Nummer 158, physikalisch auf einer CAN-Insel mit der CAN-Adresse 3 und dort Bit 2 angesiedelt				
158	254	0	3	2

Testen von Ein- und Ausgängen

Um die Ein- und Ausgänge zu testen wird das entsprechende Testbild aufgerufen. In diesem Bild wird der Status der logischen Ein- und Ausgänge angezeigt.

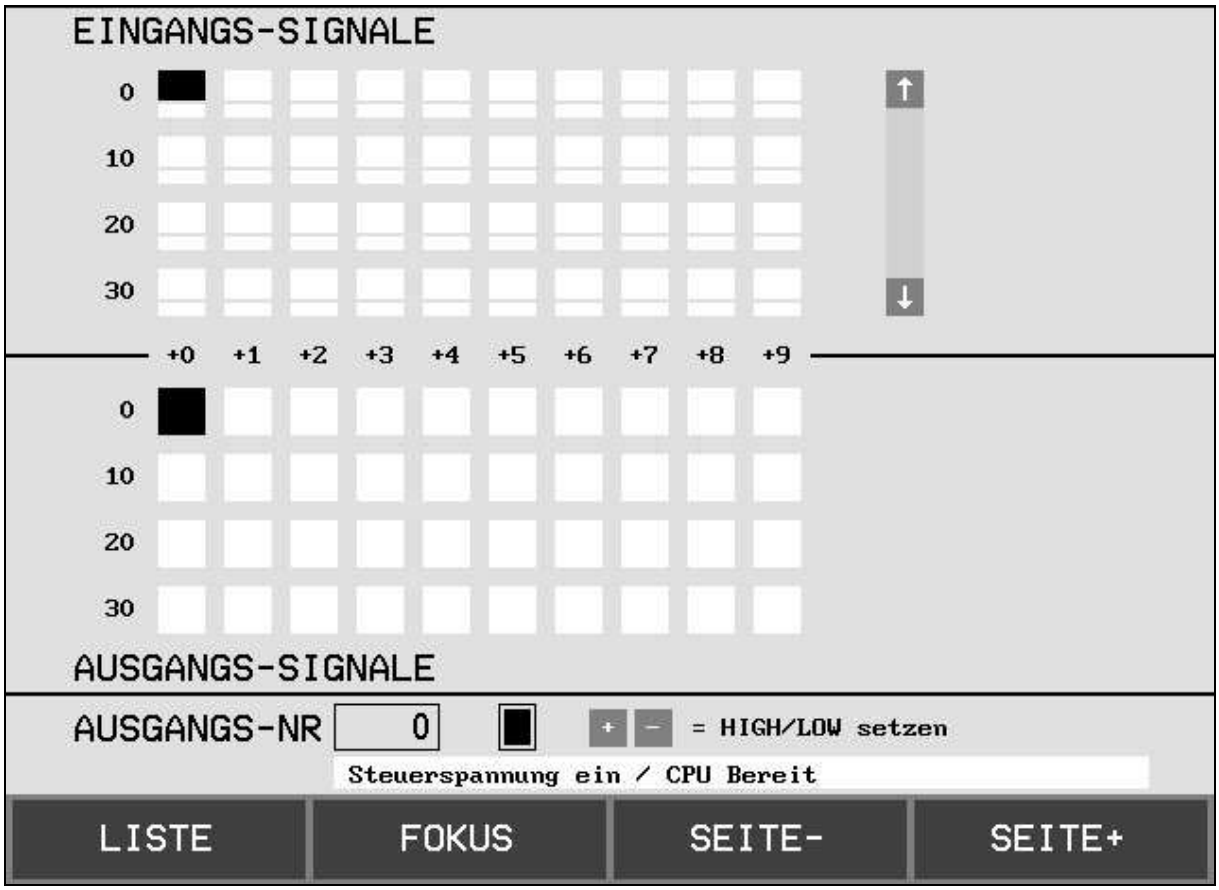


Abbildung 3: Testen von Ein- und Ausgängen

Angezeigt werden immer 40 Ein- bzw. Ausgänge als Block. Mit den Cursortasten kann dieser Block verändert werden. Dabei wird immer der Teil verändert, der gerade den Fokus hat (im oberen Beispiel die Eingänge). Der Fokus kann mit dem Softkey „Fokus“ gewechselt werden. Bei den Eingängen wird im großen Kästchen der Status des Eingangs, im darunter liegenden kleinen Kästchen der Status des entsprechenden Sekundäreingangs angezeigt. Sowohl bei Ein- als auch bei Ausgängen bedeutet ein schwarzes Kästchen den Status „High“, ein weißes Kästchen den Status „Low“. Mit der Plus-Taste kann der unter Ausgangs-Nr. angegeben Ausgang gesetzt, mit der Minus-Taste gelöscht werden. Um einen anderen Ausgang zum Setzen oder Löschen zu wählen, wird zunächst die Enter-Taste gedrückt und anschließend die gewünschte Ausgangsnummer eingegeben. Der Text unterhalb der Ausgangsnummer zeigt die Betextung des aktiv angewählten Ausgangs, also des Ausgangs, der sich verändern würde, wenn die Plus- oder Minus-Taste betätigt werden würde.

Der Softkey „Liste“ bietet eine Auflistung der Eingangs- bzw. Ausgangstexte an. Hierzu erscheint ein neues Fenster, in dem die entsprechende Betextung verändert oder gedruckt werden kann. Dieses Fenster wird zusätzlich in das Bild eingeblendet, entspricht aber der Maske zur Betextung von Ein- und Ausgängen (siehe entsprechendes Kapitel der Dokumentation). Befindet sich der Fokus bei den Eingängen, dann werden die Eingangstexte angezeigt. Befindet sich der Fokus im Bereich der Ausgänge, dann werden die Ausgangstexte angezeigt.

Die nachfolgende Abbildung zeigt das Bild zum Testen der Ein- und Ausgänge mit eingeblendeter Liste der Eingangstexte.

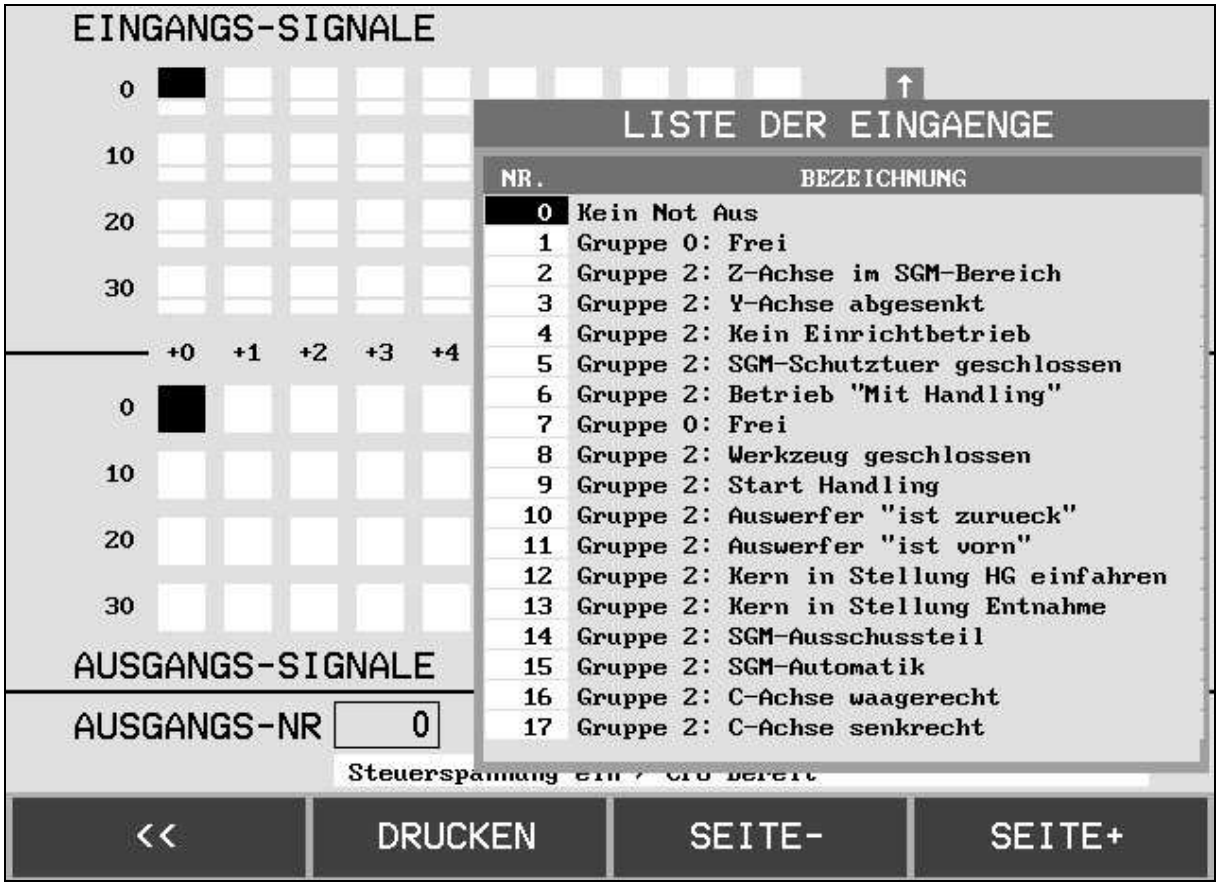


Abbildung 4: Test der Ein- und Ausgänge mit eingeblendeter Liste

Solange im Testbild die Liste der Texte eingeblendet ist, wird das Statusbild der Ein- und Ausgänge nicht aktualisiert!

Ablaufprogrammierung, grundlegende Arbeitsweise

Bei der Ablaufprogrammierung handelt es sich um Programmierung, die unter anderem auch die Funktionen einer klassischen SPS abbilden kann. Je nach Ausbaustufe der Steuerung können bis zu 1024 Abläufe parallel im System arbeiten.

Was ist ein Ablauf?

Ein Ablauf ist ein eigenständiger Prozess innerhalb der Steuerung. Er verfügt über 16 eigene Merker und einen eigenen Timer. Mit anderen Prozessen oder anderen Teilbereichen einer Steuerung kommuniziert der Ablauf über seinen Auftrag und seinen Step. Dabei sollte der Auftrag dem Ablauf grundsätzlich mitteilen, was er zu tun hat – über den Step meldet der Ablauf dann zurück, in welchem Zustand er sich gerade befindet.

Ein Ablauf kann entweder permanent Rechenzeit erhalten oder aber nur dann, wenn er einen Auftrag ungleich Null hat. Prinzipiell erhält jeder Ablauf zunächst einmal Rechenzeit. Steht der Ablauf in seiner Abarbeitung aber auf einem Befehl 51, so erhält er nur dann Rechenzeit, wenn sein Auftrag ungleich Null wird. Der Befehl 51 erfordert also keine Rechenzeit für einen Ablauf.

Wie werden Abläufe gestartet?

Abläufe, die nicht auf einem Befehl 51 stehen, laufen, sobald die Ablaufprogrammierung frei gegeben worden ist. Alle anderen Abläufe laufen dann, wenn ihr Auftrag ungleich 0 ist. Der Auftrag eines Ablaufs kann von anderen Abläufen, vom System oder von anderen Teilbereichen des System gesetzt werden. Damit systemrelevante Abläufe überhaupt ersteinmal angestartet werden, erhalten alle Abläufe mit einer Nummer größer 900 beim Start des Systems einen Initialisierungsauftrag mit dem Wert 32767. Diese Abläufe müssen dann dafür sorgen, dass andere benötigte Abläufe gestartet werden.

In manchen Betriebssystem (z.B. OALSYS) gibt es noch weitere Vereinbarungen. Hier können Abläufe beispielsweise auch dann gestartet werden, wenn sie in einer Maschinenkonfiguration eingetragen sind. Solche Vereinbarungen sind aber von dem verwendeten Betriebssystem abhängig.

Verteilung der Rechenzeit

Die gesamte Ablaufprogrammierung wird vom System zyklisch aufgerufen. Sobald dies der Fall ist, liest die Ablaufprogrammierung zunächst den Status aller Eingänge ein. Anschließend weist die Ablaufprogrammierung den einzelnen Abläufen Rechenzeit zu. Sie beginnt dabei immer mit dem Ablauf, der die niedrigste Nummer hat und geht dann in numerischer Reihenfolge weiter. Jeder Ablauf erhält dann solange Rechenzeit, bis er auf einen Sprungbefehl oder auf eine Wartebedingung trifft. Wenn der letzte Ablauf seine Rechenzeit erhalten hat, werden die veränderten Ausgänge an das System zurück gemeldet und von dort aus gesetzt bzw. gelöscht.

Generelle Befehlssyntax einer Zeile im Ablauf

Jeder Befehl innerhalb eines Ablaufs besteht aus einem Opcode (dem eigentlich Befehl) und einem Operanden. Beim Opcode handelt es sich um ein Byte, beim Operanden um ein Wort. Das oberste Bit des Operanden kann benutzt werden, um eine Indizierung festzulegen – womit sich als höchster Wert für einen Operanden 32767 ergibt. Die Indizierung ist vor allem zur Vereinfachung von Schleifenprogrammierungen vorgesehen. Ist das Indizierungsbit gesetzt, so wird auf den Operanden automatisch der Inhalt des Zähler aufaddiert.

Strukturänderungen innerhalb eines Ablaufes

Die Struktur eines Ablaufs kann nur dann verändert werden, wenn die Ablaufprogrammierung nicht aktiv ist. Damit können in einem Ablauf nur dann Zeilen eingefügt oder gelöscht werden, wenn die Ablaufprogrammierung steht. Dies hat zur Folge, dass in allen Bildern und Masken der Steuerung, in denen innerhalb eines Ablaufs Zeilen eingefügt oder gelöscht werden können, die Ablaufprogrammierung angehalten wird. Die Ablaufprogrammierung kann bei allen Systemen auch grundsätzlich über die allgemeinen Konstanten abgeschaltet werden. Bei abgeschalteter Ablaufprogrammierung erhalten nur die Abläufe mit einer Nummer größer 999 Rechenzeit.

Notaussprung

Für jeden Ablauf kann ein Notaussprung armiert werden. Bei der Armierung, die über einen entsprechenden Befehl erfolgt, wird angegeben, auf welche Zeile der Ablauf bei Notaus springen soll. Stellt das System dann die Notausbedingung fest, springen alle Abläufe mit armiertem Notaussprung auf die entsprechende Zeile und machen dort weiter. Ein Notaussprung mit dem Operand Null entfernt die entsprechende Armierung.

Globale und lokale Merker

Jedes System verfügt über 1024 globale Merker, die von 0 bis 1023 nummeriert sind. Auf die globalen Merker kann jeder Ablauf zugreifen, sie lesen und verändern. Zusätzlich hat jeder Ablauf 16 eigene Merker (sogenannte lokale Merker), die von 0 bis 15 nummeriert sind. Lokale Merker können zwar von jedem Ablauf gelesen werden, eine Veränderung kann aber nur der Ablauf bewirken, zu dem diese lokalen Merker gehören. Sowohl die globalen, als auch die lokalen Merker sind sogenannte boolsche Variable, d.h. sie können ausschließlich den Wert 0 oder 1 haben.

Die Globalen Merker mit den Nummern 1000 und höher stehen nicht zur freien Verfügung, sondern werden zum Informationsaustausch und zur Synchronisation zwischen der Ablaufprogrammierung und dem Betriebssystem benutzt. Eine Aufstellung der Bedeutung dieser Merker befindet sich im Anhang.

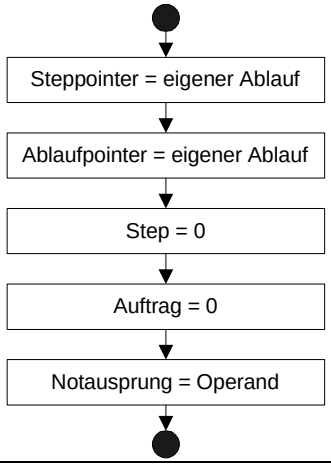
Ablaufprogrammierung, Befehlsbeschreibung

In diesem Kapitel werden die einzelnen Befehle (Opcodes) der Ablaufprogrammierung dargestellt und in ihrer Funktionsweise erläutert.

Opcode 0: NOP

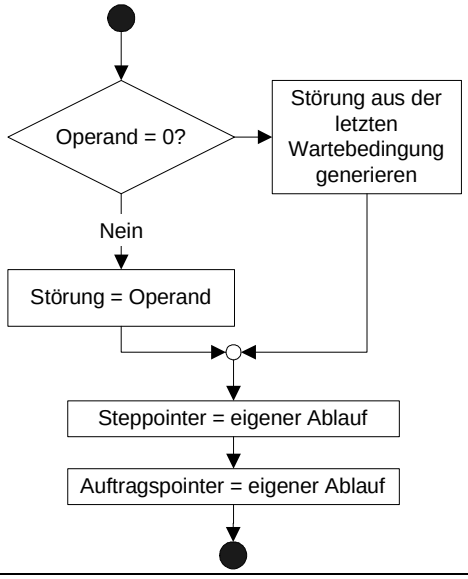
Der Opcode 0 ist kein Befehl, er hat keinerlei Funktion und verbraucht keine Rechenzeit. Wird dem Operanden bei diesem Befehl ein Wert ungleich Null zugewiesen, so wird dieser Operand als Kommentar aufgefasst. Es wird dann in der zentralen Textdatei nach einem Schlüssel mit der Syntax SPSKOMM...+Operand gesucht, beispielsweise SPSKOMM.....5, wenn der Operand 5 ist. Der Opcode 0 ist in Verbindung mit einem Operanden ungleich Null also die Möglichkeit, den Programmcode eines Ablaufes zu kommentieren.

Opcode 1: Fertig

Blockschaltbild	***Funktionsbeschreibung***
	Auftragspointer und Steppointer werden auf den eigenen Ablauf gesetzt. Auftrag und Step werden auf Null gesetzt. Als Notaus-Sprung wird der Operand armiert. Ist der Operand Null, wird die Armierung eines Notaussprungs aufgehoben.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 2: Melde Störung

Blockschaltbild	***Funktionsbeschreibung***
	Veranlasst das System, die im Operanden genannte Nummer als Störung zu melden. Angezeigt wird dann vom System die Ablaufnummer, der Ablaufname und der Störungstext, der unter der Nummer angegeben worden ist. Ist der Operand Null, so wird die Störung aus der letzten Wartebedingung generiert (siehe Beispiel). Steppointer und Auftragspointer werden auf den eigenen Ablauf gesetzt, der Step wird gelöscht.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

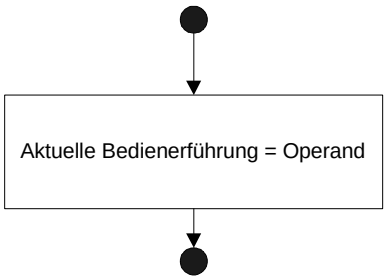
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

<u>Beispiel zur Generierung der Störung aus der Wartebedingung</u>:

```
16     3    -> Warte Eingang low / Zeit
70    23    -> Sprung auf Zeile 23, wenn der Eingang low ist
 2     0    -> Generiere Störung aus der Wartebedingung
```

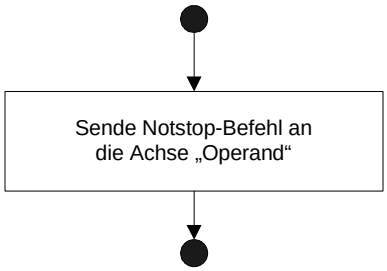
Wenn die Zeit abgelaufen und der Eingang nicht low ist, dann führt der Befehl 2 mit dem Operanden 0 dazu, dass als Störung „Zeit Eingang 3 low“ angezeigt wird. Die Störung wurde direkt aus der Wartebedingung generiert.

Opcode 3: Melde Bedienerführung

Blockschaltbild	***Funktionsbeschreibung***
	Veranlasst das System, die im Operanden genannte Nummer als Bedienerführung einzutragen. Ist im aktiven Bild die Anzeige einer Bedienerführung vorgesehen, dann wird anschließend dort der Text angezeigt, der unter der Nummer des Operanden gefunden worden ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

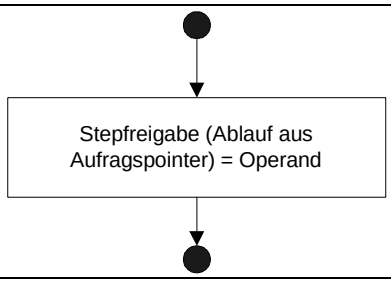
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 4: Notstop Achse

Blockschaltbild	***Funktionsbeschreibung***
	Sendet an die motorische Achse mit der im Operanden angegebenen Nummer einen Notstop-Befehl. Ist der Operand Null, so wird der Befehl an alle Achsen gesendet, die zur gleichen Gruppe wie der aufrufenden Ablauf gehören.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 5: Lade Stepfreigabe

Blockschaltbild	***Funktionsbeschreibung***
	Für den Ablauf, auf den aktuell der Auftragspointer steht, wird der Operand als Stepfreigabe geladen.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 6: Pointer Externe Daten

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den Pointer innerhalb des WCBs auf das Datenfeld mit der im Operanden angegebenen Nummer (nullbasiert). Der Rekord, auf dem der WCB-Pointer steht wird nicht verändert. Der Pointer wird innerhalb des Rekords als physikalischer Offset gesetzt, die in der WCB.DBH hinterlegte Datenstruktur wird dabei nicht betrachtet!
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 7: System-Auftrag

Blockschaltbild	***Funktionsbeschreibung***
	Erteilt dem System den Auftrag, der im Operanden angegebene worden ist. Die möglichen Systemaufträge sind systemabhängig und im Anhang aufgelistet.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 8: System-Funktion

Blockschaltbild	***Funktionsbeschreibung***
	Führt innerhalb des Systems eine Systemfunktion aus. Welche Funktion ausgeführt wird, ist vom Operanden abhängig. Die möglichen Systemfunktionen sind systemabhängig und im Anhang aufgelistet.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

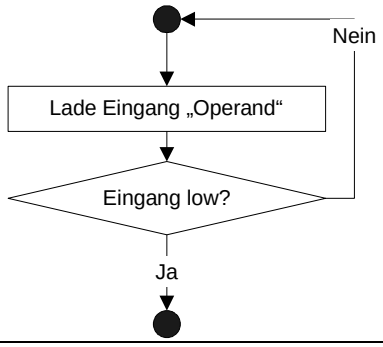
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 9: Lade Wartebedingung

Blockschaltbild	***Funktionsbeschreibung***
	Dieser Befehl wird nicht mehr unterstützt!
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

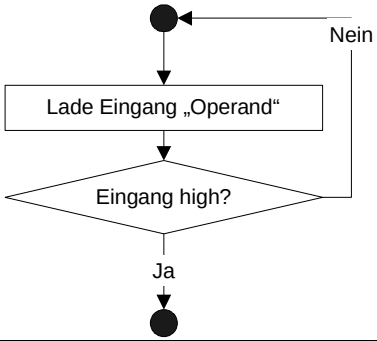
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 10: Warte bis Eingang low

Blockschaltbild	***Funktionsbeschreibung***
	Wartet bis der Eingang mit der logischen Nummer, die im Operanden angegeben ist, low ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf arbeitet weiter wenn der Eingang low ist; Ablaufprogrammierung macht mit dem nächsten Ablauf weiter, wenn der Eingang high ist.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

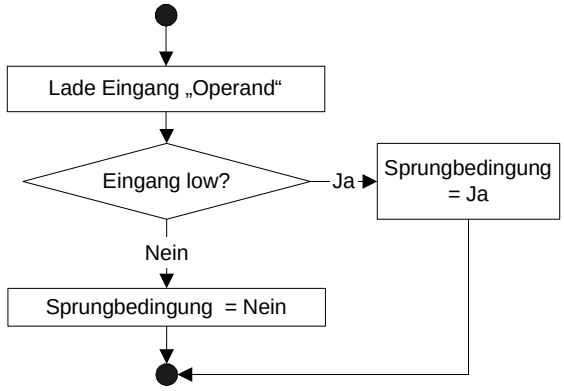
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 11: Warte bis Eingang high

Blockschaltbild	***Funktionsbeschreibung***
	Wartet bis der Eingang mit der logischen Nummer, die im Operanden angegeben ist, high ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf arbeitet weiter wenn der Eingang high ist; Ablaufprogrammierung macht mit dem nächsten Ablauf weiter, wenn der Eingang low ist.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

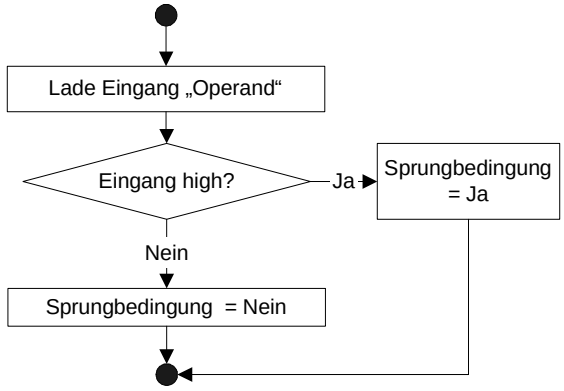
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 12: Test Eingang low

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf *Ja*, wenn der logische Eingang mit der im Operanden angegebenen Nummer low ist. Setzt die Bedingung auf *Nein* wenn der Eingang high ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

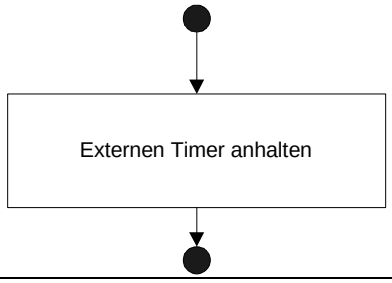
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 13: Test Eingang high

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf *Ja*, wenn der logische Eingang mit der im Operanden angegebenen Nummer high ist. Setzt die Bedingung auf *Nein* wenn der Eingang low ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

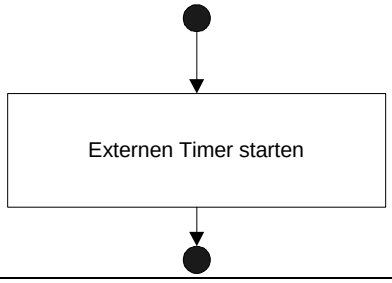
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 14: Externen Timer stoppen

Blockschaltbild	***Funktionsbeschreibung***
	Hält einen externen Timer an. Der Timer wird dabei nicht zurück gesetzt. Der Operand gibt an, welcher Rekord aus der BDE-Datei verwendet werden soll.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

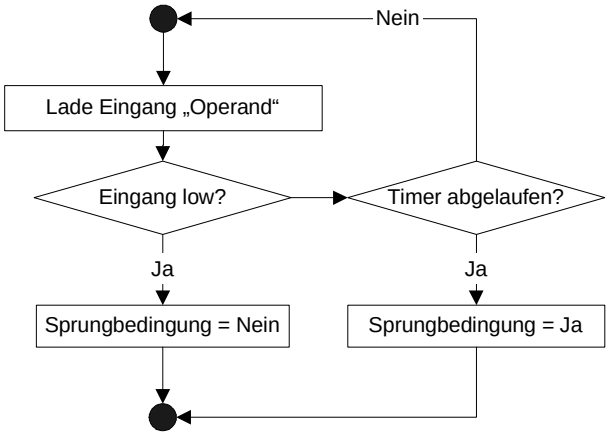
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 15: Externen Timer starten

Blockschaltbild	*Funktionsbeschreibung*
	Startet einen externen Timer an mit dem Wert, den der Timer aktuell hat. Der Operand gibt an, welcher Rekord aus der BDE-Datei verwendet werden soll.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

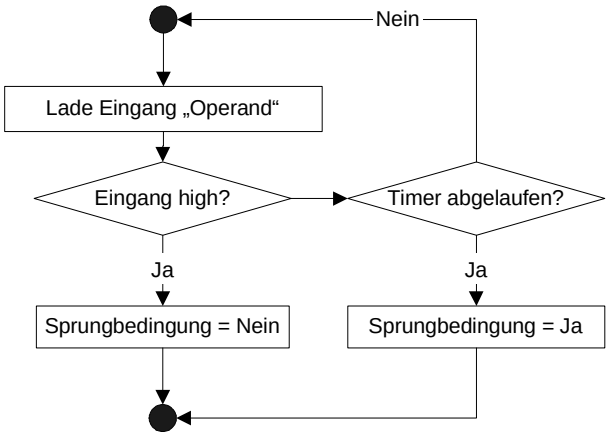
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 16: Warte Eingang low / Zeit

Blockschaltbild	*Funktionsbeschreibung*
	Der Ablauf wartet, bis der logische Eingang mit der im Operanden angegebenen Nummer low ist oder bis der interne Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Eingang low ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

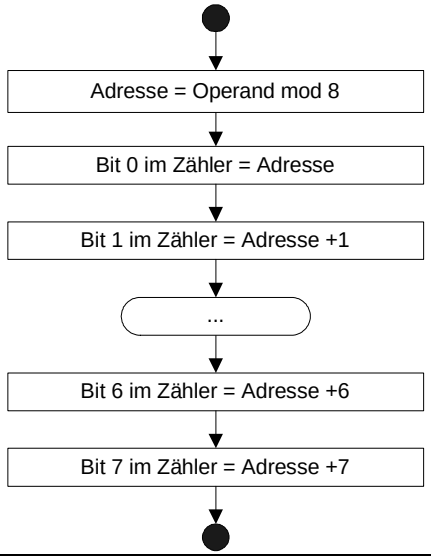
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 17: Warte Eingang high / Zeit

Blockschaltbild	*Funktionsbeschreibung*
	Der Ablauf wartet, bis der logische Eingang mit der im Operanden angegebenen Nummer high ist oder bis der interne Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Eingang high ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

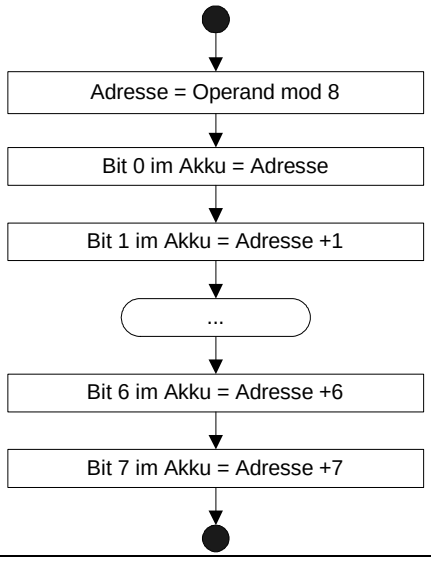
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 18: Lade Eingangsgruppe (Byte) in Zähler

Blockschaltbild	***Funktionsbeschreibung***
	Liest ein vollständiges Byte aus dem Mapping der Eingänge. Das zu lesende Byte wird in den Zähler geschrieben. Der erste Eingang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

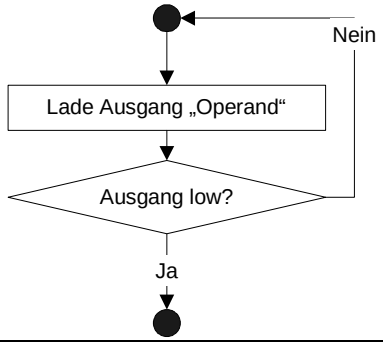
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 19: Lade Eingangsgruppe (Byte) in Akku

Blockschaltbild	***Funktionsbeschreibung***
	Liest ein vollständiges Byte aus dem Mapping der Eingänge. Das zu lesende Byte wird in den Akku geschrieben. Der erste Eingang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

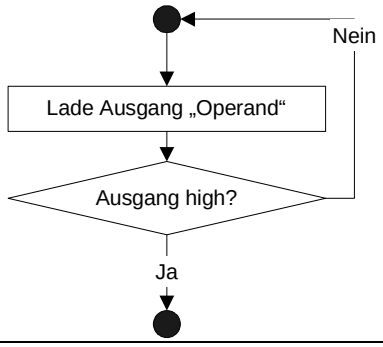
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 20: Warte bis Ausgang low

Blockschaltbild	*Funktionsbeschreibung*
	Wartet bis der Ausgang mit der logischen Nummer, die im Operanden angegeben ist, low ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf arbeitet weiter wenn der Ausgang low ist; Ablaufprogrammierung macht mit dem nächsten Ablauf weiter, wenn der Ausgang high ist.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

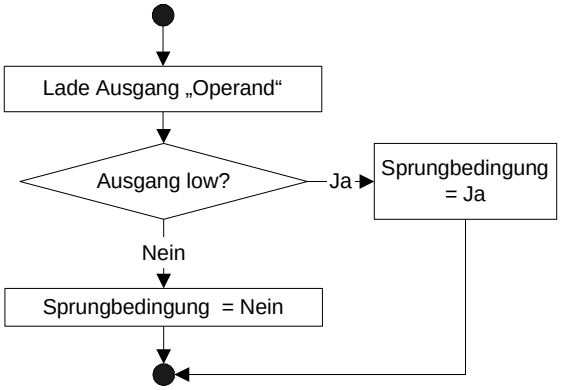
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 21: Warte bis Ausgang high

Blockschaltbild	*Funktionsbeschreibung*
	Wartet bis der Ausgang mit der logischen Nummer, die im Operanden angegeben ist, high ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf arbeitet weiter wenn der Ausgang high ist; Ablaufprogrammierung macht mit dem nächsten Ablauf weiter, wenn der Ausgang low ist.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

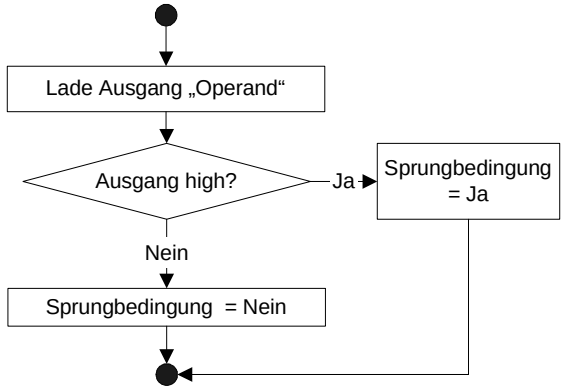
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 22: Test Ausgang low

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf *Ja*, wenn der logische Ausgang mit der im Operanden angegebenen Nummer low ist. Setzt die Bedingung auf *Nein* wenn der Ausgang high ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

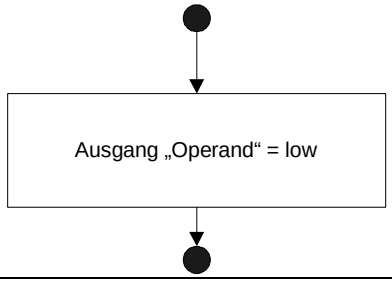
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 23: Test Ausgang high

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf *Ja*, wenn der logische Ausgang mit der im Operanden angegebenen Nummer low ist. Setzt die Bedingung auf *Nein* wenn der Ausgang high ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

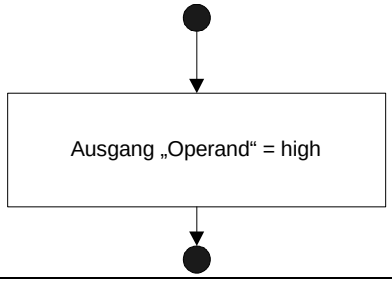
Opcode 24: Ausgang löschen

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den logischen Ausgang mit der im Operanden angegebenen Nummer auf low.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Hinweis: Die Ausgänge werden erst am Ende des Zyklus der gesamten Ablaufprogrammierung tatsächlich gesetzt! Wird der gleiche Ausgang innerhalb des selben Zyklus der Ablaufprogrammierung an anderer Stelle wieder gesetzt, dann geht das Löschen dieses Ausgangs verloren!

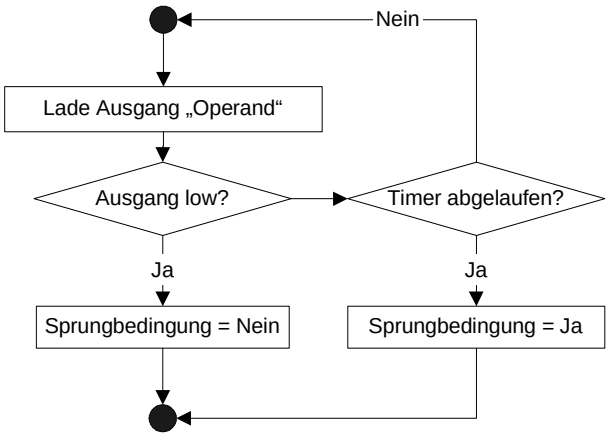
Opcode 25: Ausgang setzen

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den logischen Ausgang mit der im Operanden angegebenen Nummer auf high.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

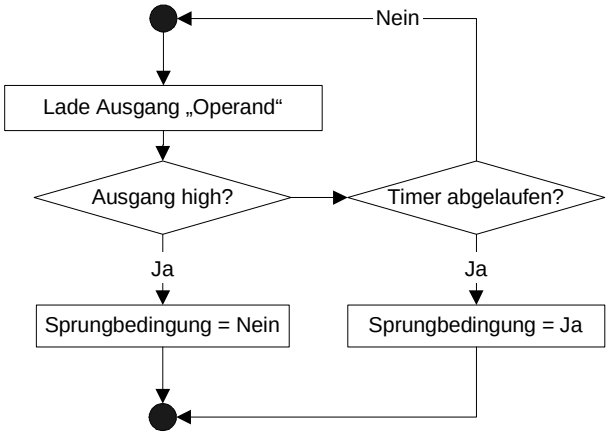
Hinweis: Die Ausgänge werden erst am Ende des Zyklus der gesamten Ablaufprogrammierung tatsächlich gesetzt! Wird der gleiche Ausgang innerhalb des selben Zyklus der Ablaufprogrammierung an anderer Stelle wieder gelöscht, dann geht das Setzen dieses Ausgangs verloren!

Opcode 26: Warte Ausgang low / Zeit

Blockschaltbild	***Funktionsbeschreibung***
	Der Ablauf wartet, bis der logische Ausgang mit der im Operanden angegebenen Nummer low ist oder bis der interne Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Ausgang low ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

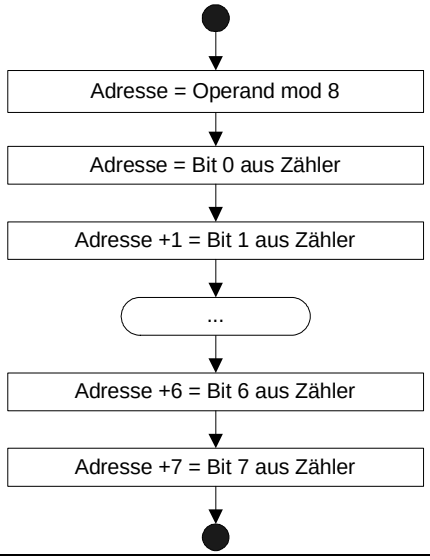
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 27: Warte Ausgang high / Zeit

Blockschaltbild	***Funktionsbeschreibung***
	Der Ablauf wartet, bis der logische Ausgang mit der im Operanden angegebenen Nummer high ist oder bis der interne Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Ausgang high ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

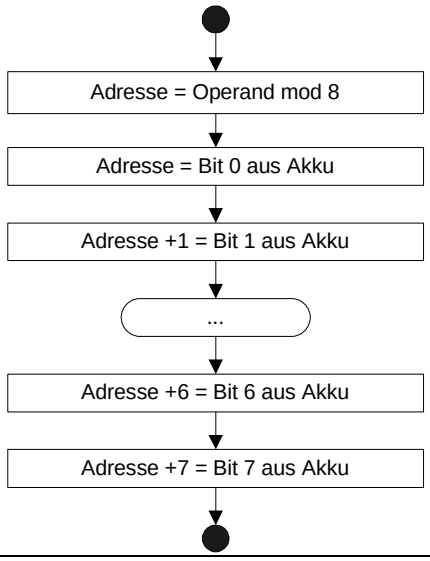
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 28: Lade Zähler in Ausgangsgruppe (Byte)

Blockschaltbild	*Funktionsbeschreibung*
	Schreibt ein vollständiges Byte in das Mapping der Ausgänge. Das zu schreibende Byte steht im Zähler. Der erste Ausgang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

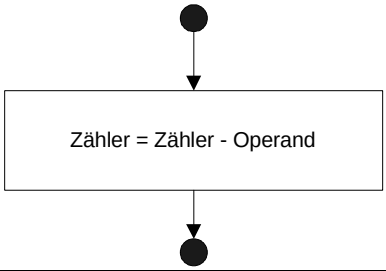
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 29: Lade Akku in Ausgangsgruppe (Byte)

Blockschaltbild	*Funktionsbeschreibung*
	Schreibt ein vollständiges Byte in das Mapping der Ausgänge. Das zu schreibende Byte steht im Akku. Der erste Ausgang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

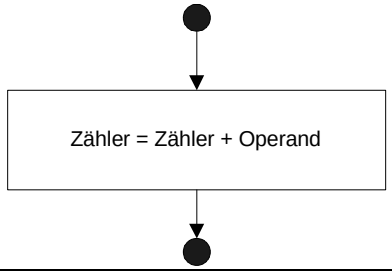
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 30: Zähler = Zähler - Operand

Blockschaltbild	*Funktionsbeschreibung*
	Subtrahiert den Operanden vom Zähler und schreibt den Wert in den Zähler.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

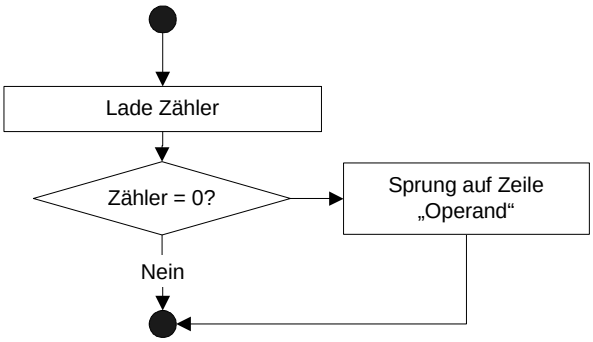
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 31: Zähler = Zähler + Operand

Blockschaltbild	*Funktionsbeschreibung*
	Addiert den Zähler und den Operanden und schreibt den Wert in den Zähler.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

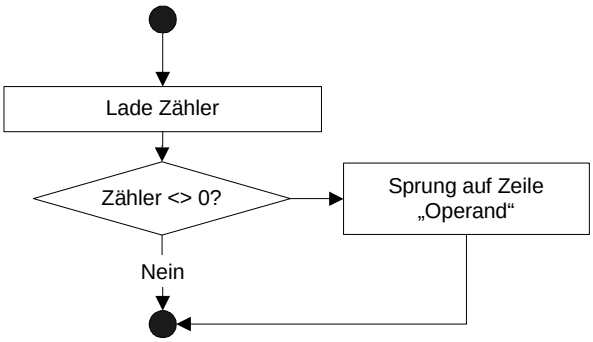
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 32: Sprung wenn Zähler = 0

Blockschaltbild	*Funktionsbeschreibung*
	Springt auf die im Operanden angegebene Zeile, wenn der Zähler Null ist.
	Ablauf macht nach Ausführung weiter?
	Ist der Zähler Null, dann springt der Ablauf auf die angegebene Zeile, anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter. Ist der Zähler ungleich Null, macht der Ablauf mit der nächsten Zeile weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

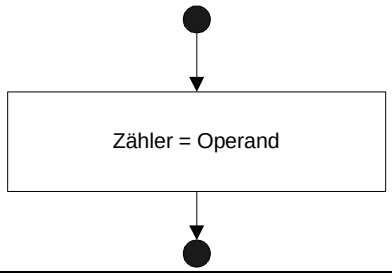
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 33: Sprung wenn Zähler <> 0

Blockschaltbild	*Funktionsbeschreibung*
	Springt auf die im Operanden angegebene Zeile, wenn der Zähler ungleich Null ist.
	Ablauf macht nach Ausführung weiter?
	Ist der Zähler ungleich Null, dann springt der Ablauf auf die angegebene Zeile, anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter. Ist der Zähler Null, macht der Ablauf mit der nächsten Zeile weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 34: Zähler laden mit Operand

Blockschaltbild	*Funktionsbeschreibung*
	Lädt den Operanden in den Zähler.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

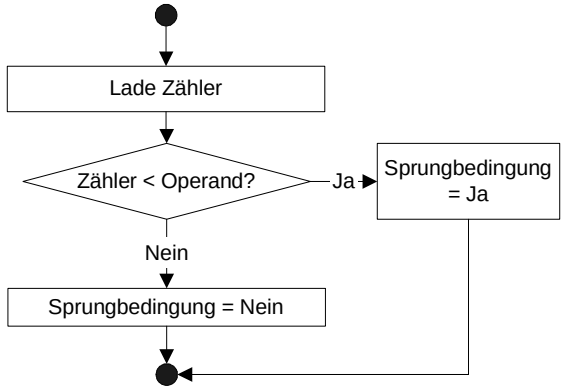
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 35: Zähler laden Sonderfunktion

Blockschaltbild	*Funktionsbeschreibung*
	Lädt den Zähler mit dem Ergebnis einer Sonderfunktion. Die Sonderfunktion wird über den Operanden angegeben. Eine Liste der Sonderfunktionen befindet sich im Anhang.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

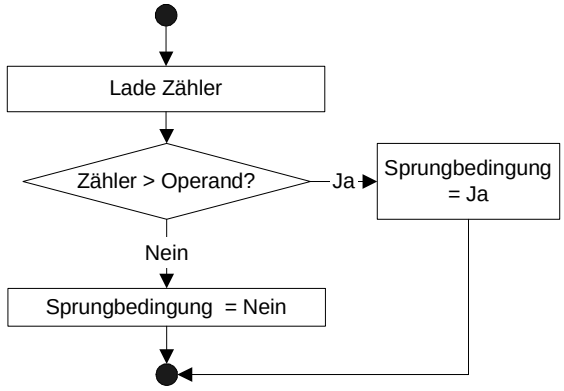
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 36: Teste Zähler < Operand

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf ***Ja***, wenn der Zähler kleiner als der Operand ist. Setzt die Bedingung auf ***Nein***, wenn der Zähler größer oder gleich dem Operanden ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

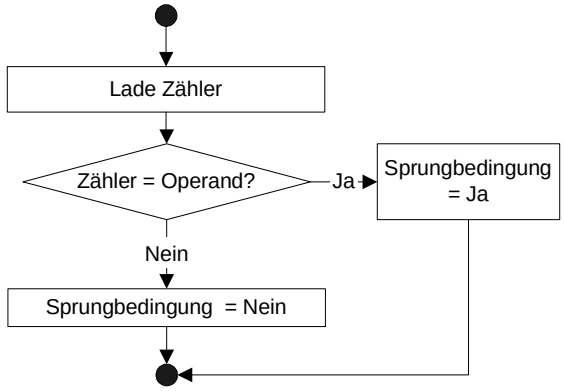
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 37: Teste Zähler > Operand

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf ***Ja***, wenn der Zähler größer als der Operand ist. Setzt die Bedingung auf ***Nein***, wenn der Zähler kleiner oder gleich dem Operanden ist .
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

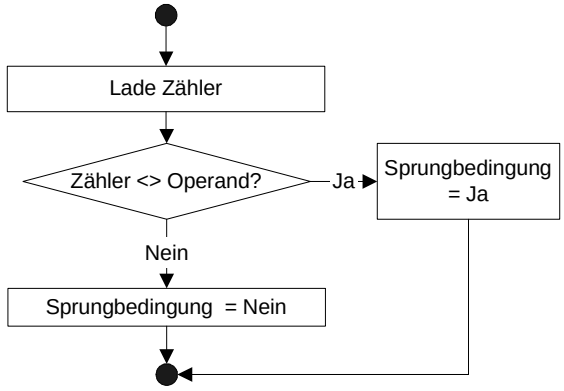
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 38: Teste Zähler = Operand

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf ***Ja***, wenn der Zähler gleich dem Operanden ist. Setzt die Bedingung auf ***Nein***, wenn der Zähler größer oder kleiner als der Operand ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

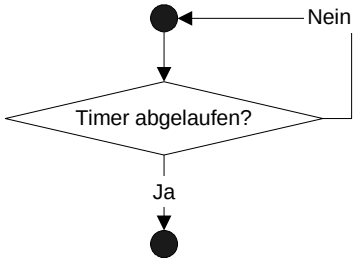
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 39: Teste Zähler <> Operand

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf ***Ja***, wenn der Zähler ungleich dem Operanden ist. Setzt die Bedingung auf ***Nein***, wenn der Zähler gleich dem Operanden ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

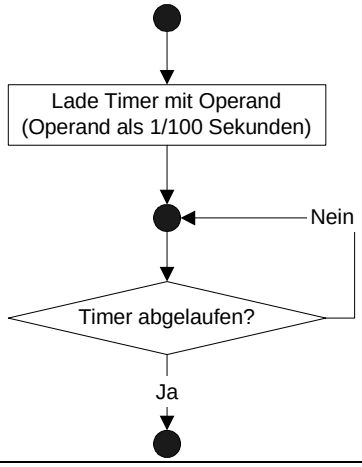
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 40: Warte bis Timer = 0

Blockschaltbild	***Funktionsbeschreibung***
	Wartet bis der Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf macht mit der nächsten Zeile weiter, wenn der Timer abgelaufen ist. Ablaufprogrammierung macht mit dem nächsten Ablauf weiter, wenn der Timer nicht abgelaufen ist.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

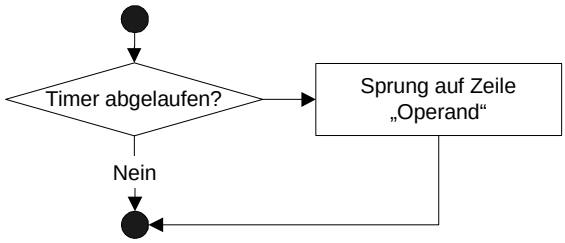
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 41: Lade Timer und warte

Blockschaltbild	***Funktionsbeschreibung***
	Lädt den Timer mit dem Operanden. Dabei wird der Operand als Angabe in 1/100 Sekunden verstanden. Anschließend wartet der Befehl, bis der Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf macht mit der nächsten Zeile weiter, wenn der Timer abgelaufen ist. Ablaufprogrammierung macht mit dem nächsten Ablauf weiter, wenn der Timer nicht abgelaufen ist.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

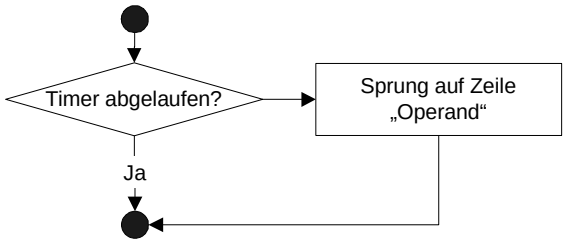
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 42: Sprung wenn Timer = 0

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile, wenn der Timer abgelaufen ist. Macht mit der nächsten Zeile weiter, wenn der Timer nicht abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf macht mit der nächsten Zeile weiter, wenn der Timer nicht abgelaufen ist. Ist der Timer abgelaufen springt der Ablauf auf die angegebene Zeile und anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

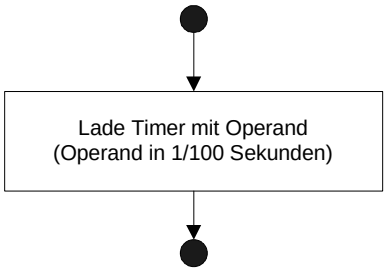
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 43: Sprung wenn Timer <> 0

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile, wenn der Timer nicht abgelaufen ist. Macht mit der nächsten Zeile weiter, wenn der Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Ablauf macht mit der nächsten Zeile weiter, wenn der Timer abgelaufen ist. Ist der Timer nicht abgelaufen springt der Ablauf auf die angegebene Zeile und anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

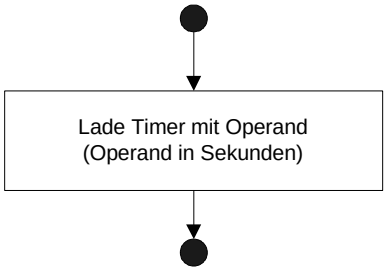
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 44: Lade Timer in 1/100 Sekunden

Blockschaltbild	***Funktionsbeschreibung***
	Lädt den Timer. Der Operand wird dabei als Angabe in 1/100 Sekunden interpretiert.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

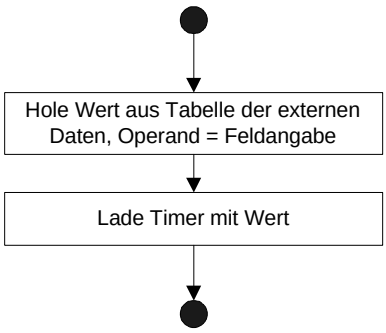
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 45: Lade Timer in Sekunden

Blockschaltbild	***Funktionsbeschreibung***
	Lädt den Timer. Der Operand wird dabei als Angabe in Sekunden interpretiert.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

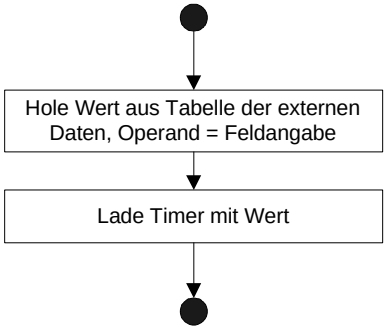
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 46: Lade Timer aus Tabelle

Blockschaltbild	***Funktionsbeschreibung***
	Lädt den Timer aus der Tabelle der externen Zeiten (SYSTEM.ZEI). Der Operand gibt dabei an, welches Feld in der Tabelle benutzt werden soll. Die Angabe ist einsbasiert.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 47: Lade Timer aus Tabelle

Blockschaltbild	***Funktionsbeschreibung***
	Lädt den Timer aus der Tabelle der externen Zeiten (PROZESS.ZEI). Der Operand gibt dabei an, welches Feld in der Tabelle benutzt werden soll. Die Angabe ist einsbasiert.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 48: Kopplung auf Startbedingung setzen

Blockschaltbild	*Funktionsbeschreibung*
	Mit dieser Funktion wird die Ausführung des Ablaufs an die Startbedingung gekoppelt. Der Ablauf erhält nur dann Rechenzeit, wenn die Startbedingung anliegt. Während der Ablauf auf die Startbedingung gekoppelt ist, merkt er sich, welchen Ausgang er zuletzt verändert hat. Wird die Startbedingung gelöscht, hält der Ablauf inklusive seines Timer an und löscht ggfs. den Ausgang. Kommt die Startbedingung wieder, läuft der Ablauf inklusive seiner Timer weiter und setzt dabei ggfs. den gelöschten Ausgang wieder.
	Ablauf macht nach Ausführung weiter?
	Abhängig von der Startbedingung
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

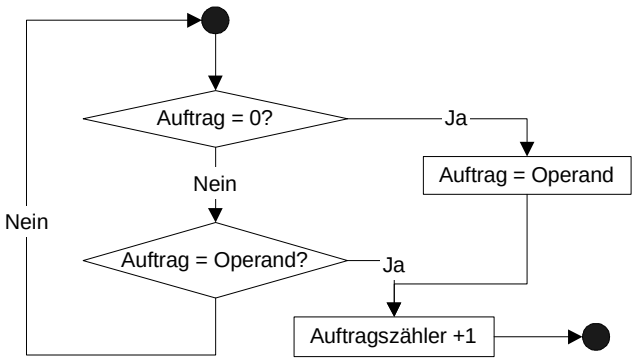
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 49: Kopplung auf Startbedingung löschen

Blockschaltbild	*Funktionsbeschreibung*
	Die Kopplung an die Startbedingung wird aufgehoben. Anschließend ist die Ausführung des Ablaufs nicht mehr an die Startbedingung gebunden.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

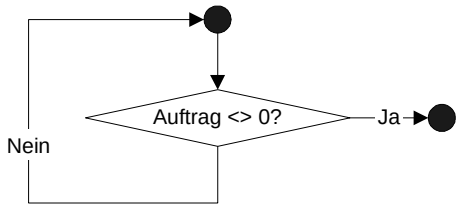
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 50: Warte bis Auftrag = 0

Blockschaltbild	*Funktionsbeschreibung*
	Wartet, bis der Auftrag des Ablaufs, auf den der Auftragspointer aktuell zeigt, Null geworden ist. Ist der entsprechende Auftrag Null geworden, wird dem Ablauf, auf den aktuell der Auftragspointer zeigt, der Auftrag erteilt, der im Operanden angegeben ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf macht weiter, wenn der Operand in den Auftrag geschrieben worden ist, ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
X	-	-

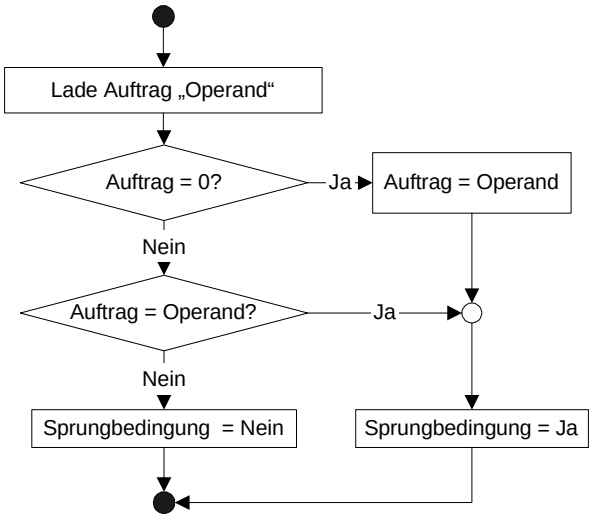
Opcode 51: Warte bis Auftrag <> 0

Blockschaltbild	***Funktionsbeschreibung***
	Wartet, bis der Auftrag des Ablaufs, auf den der Auftragspointer aktuell zeigt, ungleich Null geworden ist.
	Ablauf macht nach Ausführung weiter?
	Siehe Hinweis
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
X	-	-

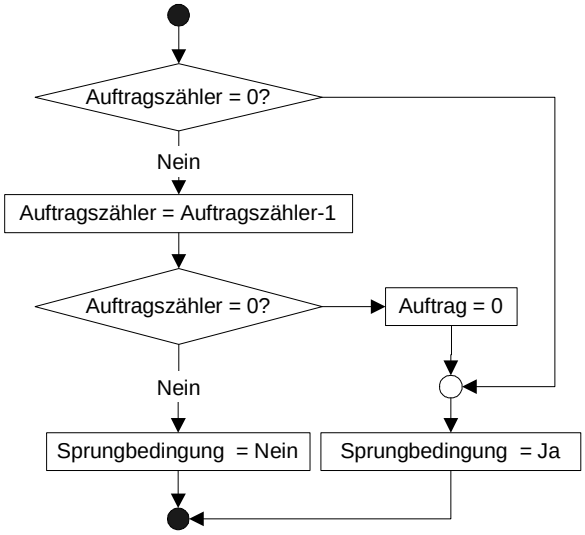
Hinweis: Wenn dieser Opcode benutzt wird, während der Auftragspointer auf den eigenen Ablauf zeigt, wird der gesamte Ablauf aus dem Tasking der Ablaufprogrammierung ausgekoppelt, wenn der Auftrag Null ist. Erhält der Ablauf dann einen Auftrag, wird er wieder in das Tasking der Ablaufprogrammierung eingehängt. Durch diese Vorgehensweise wird die Grundbelastung der Ablaufprogrammierung reduziert. Alle Abläufe, die zeitlich begrenzte Vorgänge steuern, sollten auf jeden Fall mit einem Opcode 51 0 beginnen, um die Belastung des Systems möglichst gering zu halten. Dies gilt natürlich nicht für Abläufe, die ständig laufen müssen, wie beispielsweise Schutzzaunüberwachungen.

Opcode 52: Test Auftrag = 0

Blockschaltbild	***Funktionsbeschreibung***
	Testet den Auftrag, auf den der Auftragspointer gerade steht. Setzt die Sprungbedingung auf Ja, wenn der Auftrag Null ist, setzt die Sprungbedingung auf Nein, wenn der Auftrag ungleich Null ist. Ist der Auftrag beim Testen Null, dann wird der Operand als Auftrag eingetragen.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

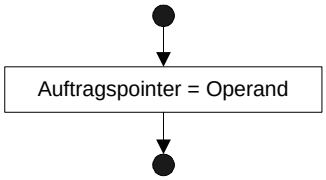
Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 53: Lösche Auftrag

Blockschaltbild	***Funktionsbeschreibung***
	Löscht den Auftrag auf den der Auftragspointer zeigt, wenn der Auftragszähler Null ist. Setzt die Sprungbedingung, wenn der Auftragszähler nach Ausführung des Befehls Null ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

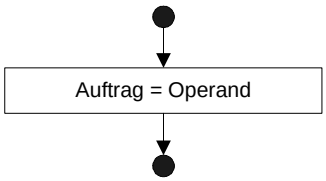
Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 54: Auftragspointer laden

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den Auftragspointer auf den Ablauf mit der im Operanden angegebenen Nummer. Ein Operand Null setzt den Auftragspointer immer auf den eigenen Ablauf.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

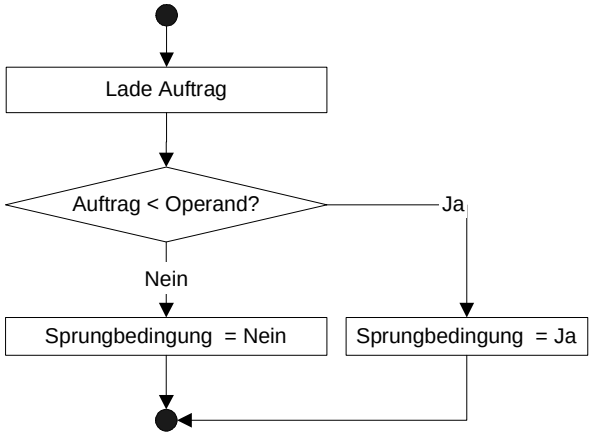
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 55: Auftrag laden

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den Auftrag, auf den der Auftragspointer aktuell zeigt, auf die im Operanden angegebenen Nummer
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

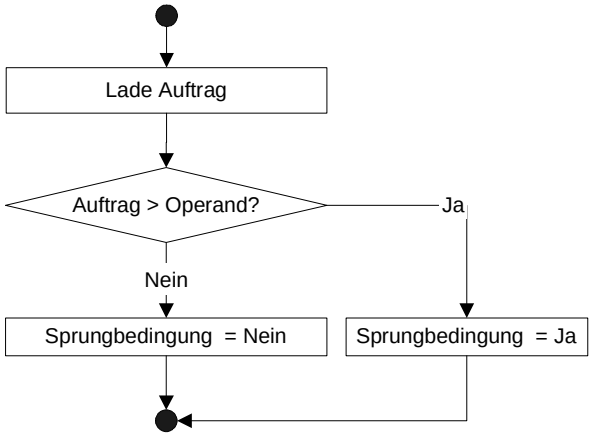
Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 56: Teste Auftrag < Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Auftrag, auf den der Auftragspointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Auftrag kleiner als der Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

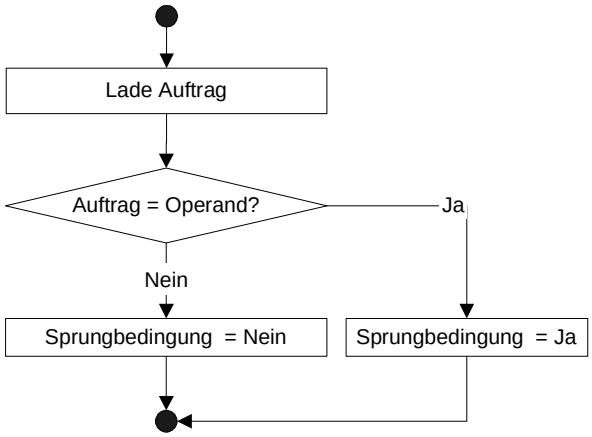
Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 57: Teste Auftrag > Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Auftrag, auf den der Auftragspointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Auftrag größer als der Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

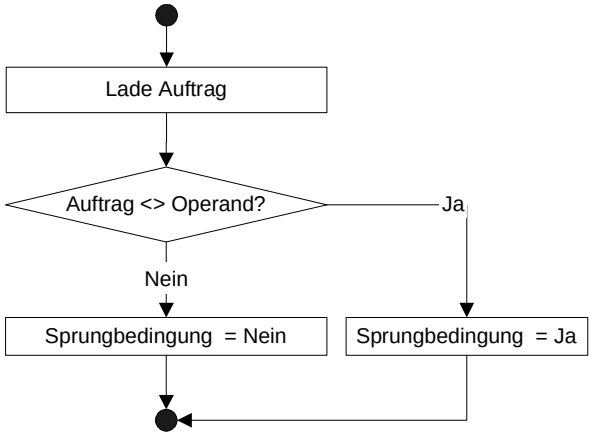
Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 58: Teste Auftrag = Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Auftrag, auf den der Auftragspointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Auftrag gleich dem Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

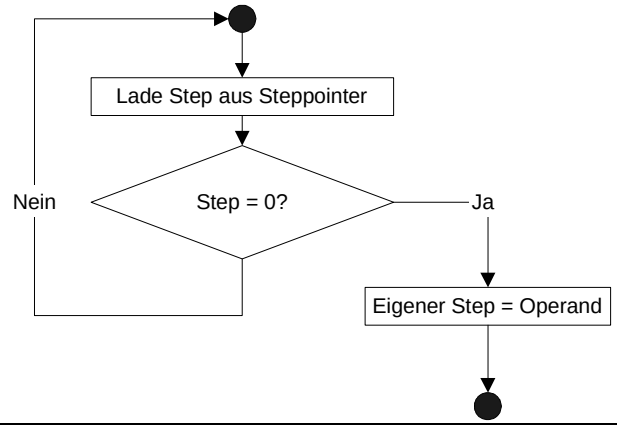
Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 59: Teste Auftrag <> Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Auftrag, auf den der Auftragspointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Auftrag ungleich dem Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

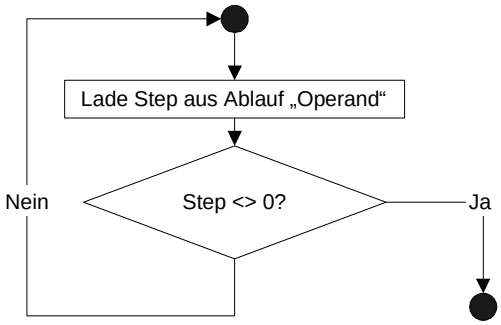
Auftragspointer	Steppointer	Sprungbedingung
X	-	-

Opcode 60: Warte bis Step = 0

Blockschaltbild	*Funktionsbeschreibung*
	Wartet, bis der Step, auf den der Steppointer zeigt, Null ist. Ist der Step Null, wird der Operand in den **eigenen** Step geladen.
	Ablauf macht nach Ausführung weiter?
	Ist der Step Null, macht der Ablauf weiter. Ist der Step ungleich Null, macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

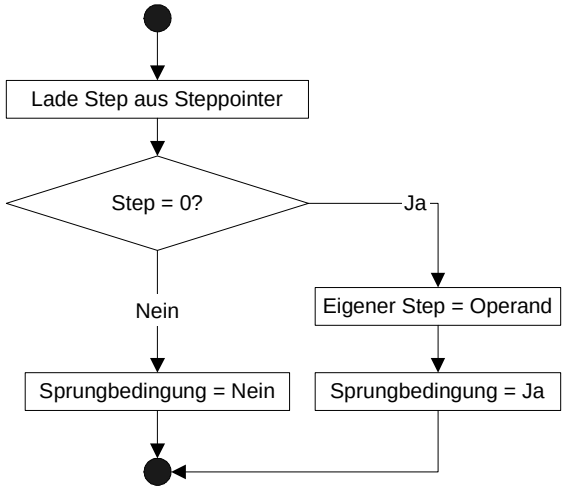
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 61: Warte bis Step <> 0

Blockschaltbild	*Funktionsbeschreibung*
	Wartet, bis der Step des Ablaufes mit der im Operanden angegebenen Nummer ungleich Null geworden ist. Der Steppointer wird nicht berücksichtigt!
	Ablauf macht nach Ausführung weiter?
	Ist der Step ungleich Null, macht der Ablauf weiter. Ist der Step Null, macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

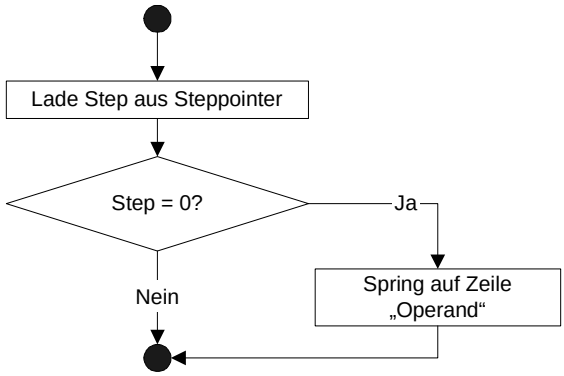
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 62: Teste Step = 0

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Step, auf den der Steppointer zeigt. Ist der Step Null, wird der Operand in den eigenen Step geladen und die Sprungbedingung auf Ja gesetzt. Ist der Step ungleich Null, wird die Sprungbedingung gelöscht.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

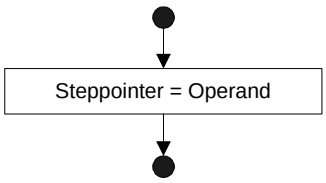
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 63: Sprung wenn Step <> 0

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Step, auf den der Steppointer zeigt. Ist der Step Null, macht der Ablauf mit dem nächsten Befehl weiter. Ist der Step ungleich Null, springt der Ablauf auf die Zeile, die im Operanden angegeben ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf macht weiter wenn der Step Null ist. Der Ablauf springt auf die angegebene Zeile und anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter, wenn der Step ungleich Null ist.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

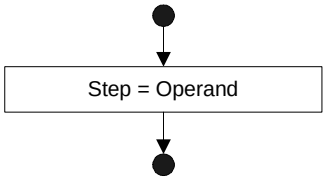
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 64: Steppointer laden

Blockschaltbild	*Funktionsbeschreibung*
	Setzt den Steppointer auf den Ablauf mit der im Operanden angegebenen Nummer. Ein Operand Null setzt den Steppointer immer auf den eigenen Ablauf.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

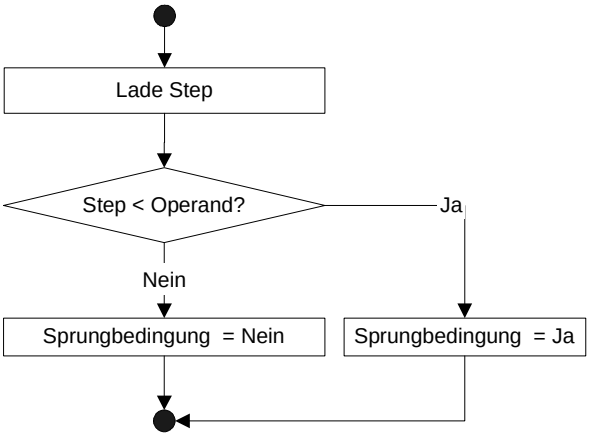
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 65: Step laden

Blockschaltbild	*Funktionsbeschreibung*
	Setzt den eigenen Step auf die im Operanden angegebenen Nummer
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

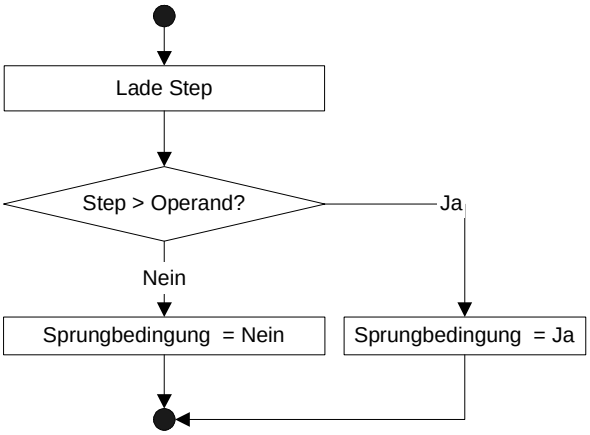
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 66: Teste Step < Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Step, auf den der Steppointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Step kleiner als der Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

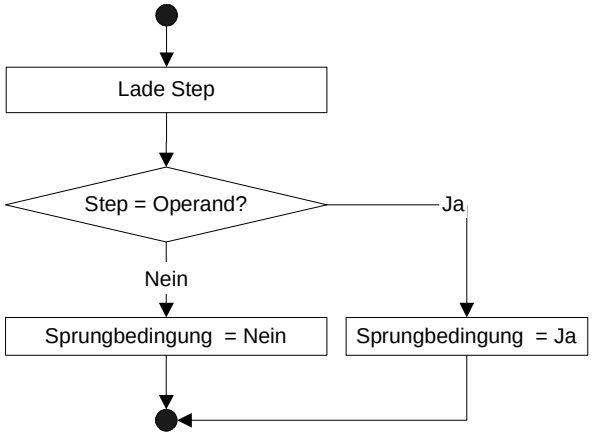
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 67: Teste Step > Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Step, auf den der Steppointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Step größer als der Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

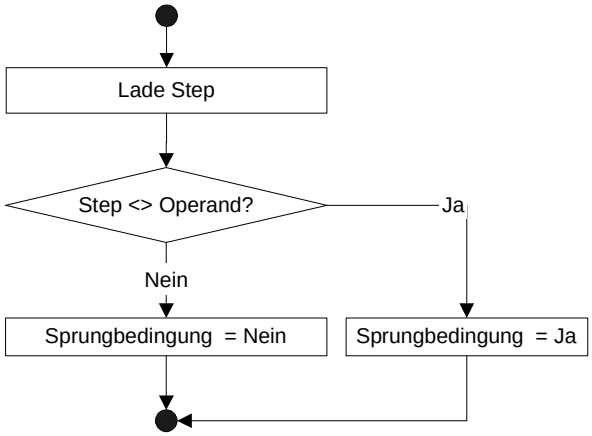
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 68: Teste Step = Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Step, auf den der Steppointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Step gleich dem Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

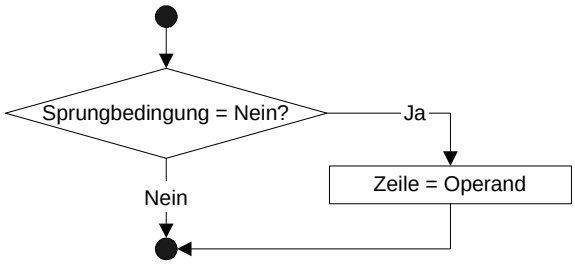
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 69: Teste Step <> Operand

Blockschaltbild	*Funktionsbeschreibung*
	Testet den Step, auf den der Steppointer aktuell zeigt. Setzt die Sprungbedingung auf Ja, wenn der Step ungleich dem Operand ist. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

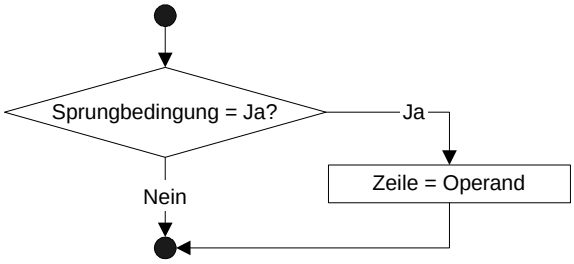
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 70: Sprung wenn Nein

Blockschaltbild	*Funktionsbeschreibung*
	Springt auf die im Operanden angegebene Zeile, wenn die Sprungbedingung auf Nein steht. Macht im anderen Fall mit der nächsten Zeile weiter.
	Ablauf macht nach Ausführung weiter?
	Wenn der Sprung ausgeführt wird, macht die Ablaufprogrammierung anschließend mit dem nächsten Ablauf weiter. Der Ablauf macht mit der nächsten Zeile weiter, wenn der Sprung nicht ausgeführt wird.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

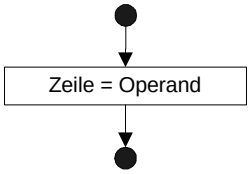
Auftragspointer	Steppointer	Sprungbedingung
-	-	X

Opcode 71: Sprung wenn Ja

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile, wenn die Sprungbedingung auf Ja steht. Macht im anderen Fall mit der nächsten Zeile weiter.
	Ablauf macht nach Ausführung weiter?
	Wenn der Sprung ausgeführt wird, macht die Ablaufprogrammierung anschließend mit dem nächsten Ablauf weiter. Der Ablauf macht mit der nächsten Zeile weiter, wenn der Sprung nicht ausgeführt wird.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

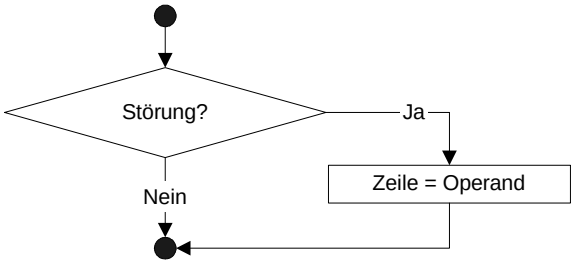
Auftragspointer	Steppointer	Sprungbedingung
-	-	X

Opcode 72: Sprung unbedingt

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile.
	Ablauf macht nach Ausführung weiter?
	Der Sprung wird ausgeführt. Anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

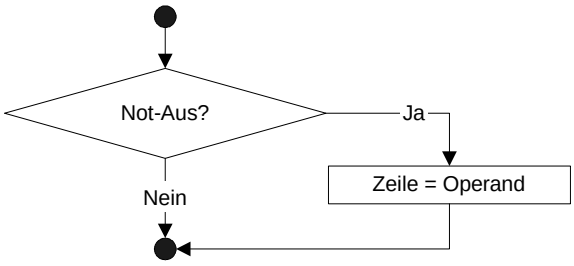
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 73: Sprung wenn Störung

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile, wenn zum Zeitpunkt der Ausführung eine Störung ansteht. Macht im anderen Fall mit der nächsten Zeile weiter.
	Ablauf macht nach Ausführung weiter?
	Wenn der Sprung ausgeführt wird, macht die Ablaufprogrammierung anschließend mit dem nächsten Ablauf weiter. Der Ablauf macht mit der nächsten Zeile weiter, wenn der Sprung nicht ausgeführt wird.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 74: Sprung wenn Notaus

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile, wenn zum Zeitpunkt der Ausführung die Notausbedingung ansteht. Macht im anderen Fall mit der nächsten Zeile weiter.
	Ablauf macht nach Ausführung weiter?
	Wenn der Sprung ausgeführt wird, macht die Ablaufprogrammierung anschließend mit dem nächsten Ablauf weiter. Der Ablauf macht mit der nächsten Zeile weiter, wenn der Sprung nicht ausgeführt wird.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

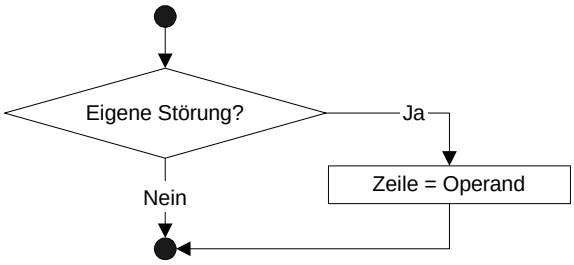
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 75: Sprungziel für Notaus

Blockschaltbild	***Funktionsbeschreibung***
	Dieser Befehl armiert einen Notaussprung. Solange der Notaussprung armiert ist, wird die Ausführung des Ablaufs unterbrochen, sobald die Notausbedingung eintrifft. Der Ablauf springt dann – unabhängig seiner aktuelle Ausführungszeile – auf die armierte Zeile, löscht den Steppointer, den Auftragspointer, den Step den Notaussprung und setzt das Störflag. Mit dem Operanden wird die entsprechende Sprungzeile angegeben, ein Operand Null löscht die Armierung des Notaussprungs.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

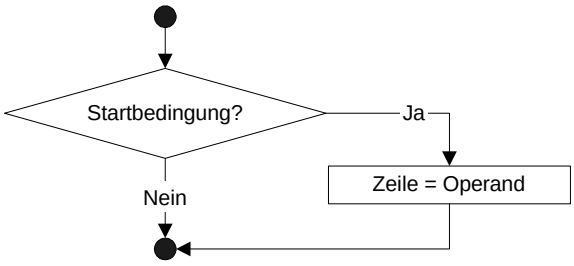
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 76: Sprung wenn eigene Störung

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile, wenn zum Zeitpunkt der Befehlsausführung eine Störung ansteht, die durch den eigenen Ablauf verursacht worden ist. Macht im anderen Fall mit der nächsten Zeile weiter.
	Ablauf macht nach Ausführung weiter?
	Wenn der Sprung ausgeführt wird, macht die Ablaufprogrammierung anschließend mit dem nächsten Ablauf weiter. Der Ablauf macht mit der nächsten Zeile weiter, wenn der Sprung nicht ausgeführt wird.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 77: Sprung wenn Startbedingung

Blockschaltbild	***Funktionsbeschreibung***
	Springt auf die im Operanden angegebene Zeile, wenn zum Zeitpunkt der Befehlsausführung die Startbedingung ansteht. Macht im anderen Fall mit der nächsten Zeile weiter.
	Ablauf macht nach Ausführung weiter?
	Wenn der Sprung ausgeführt wird, macht die Ablaufprogrammierung anschließend mit dem nächsten Ablauf weiter. Der Ablauf macht mit der nächsten Zeile weiter, wenn der Sprung nicht ausgeführt wird.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 78: Sprung wenn Folgeauftrag

Blockschaltbild	***Funktionsbeschreibung***
	Systemabhängig und nicht in allen Systemen implementiert.
	Ablauf macht nach Ausführung weiter?
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

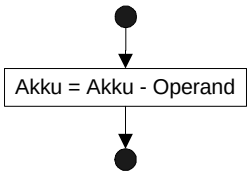
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 79: Unterprogrammsprung

Blockschaltbild	***Funktionsbeschreibung***
	Der Ablauf springt auf die im Operanden angegebene Zeile und betrachtet die dortigen Befehle als Unterprogramm. Die Befehle werden ausgeführt. Folgt dann ein Befehl 79 mit dem Operanden Null, so gilt das Unterprogramm als abgeschlossen und der Ablauf springt auf die Zeile hinter der Zeile, die den Unterprogrammsprung ausgelöst hat. Innerhalb von Unterprogrammen ist kein weiterer Unterprogrammsprung möglich.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf führt den Sprung aus, anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

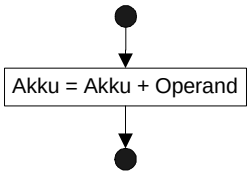
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 80: Akku = Akku – Operand

Blockschaltbild	*Funktionsbeschreibung*
	Subtrahiert den Operanden vom Akku und schreibt den neuen Wert zurück in den Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

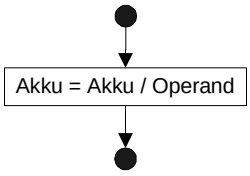
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 81: Akku = Akku + Operand

Blockschaltbild	*Funktionsbeschreibung*
	Addiert den Operanden zum Akku und schreibt den neuen Wert zurück in den Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

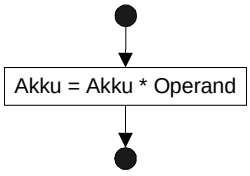
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 82: Akku = Akku / Operand

Blockschaltbild	*Funktionsbeschreibung*
	Dividiert den Akku durch den Operanden und schreibt den ganzzahligen Teil des Ergebnisses zurück in den Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

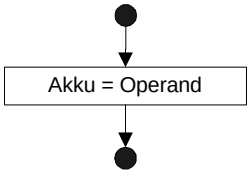
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 83: Akku = Akku * Operand

Blockschaltbild	*Funktionsbeschreibung*
	Multipliziert den Operanden und den Akku und schreibt den neuen Wert zurück in den Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 84: Akku laden mit Operand

Blockschaltbild	***Funktionsbeschreibung***
	Lädt den Operanden in den Akku
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

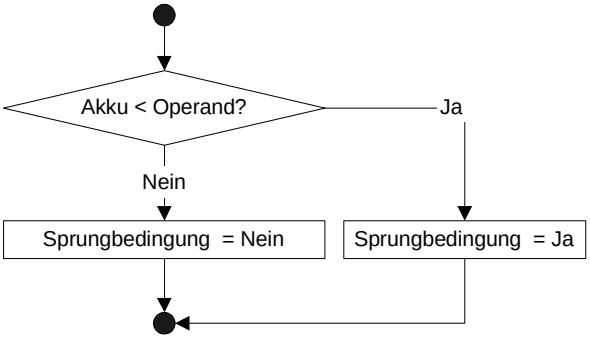
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 85: Akku laden Sonderfunktion

Blockschaltbild	***Funktionsbeschreibung***
	Lädt den Zähler mit dem Ergebnis einer Sonderfunktion. Die Sonderfunktion wird über den Operanden angegeben. Eine Liste der Sonderfunktionen befindet sich im Anhang.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

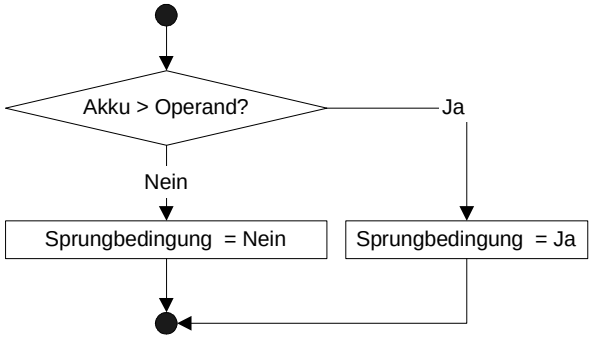
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 86: Teste Akku < Operand

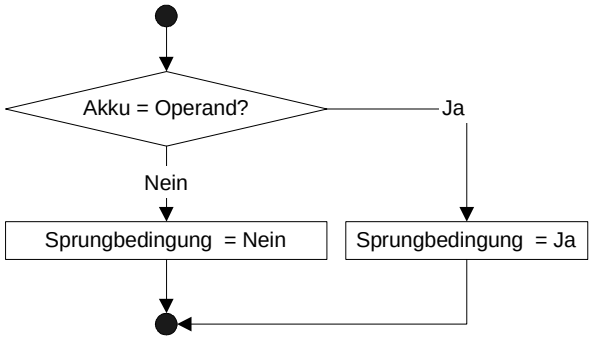
Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf Ja, wenn der Akku kleiner als der Operand ist. Setzt die Sprungbedingung im anderen Fall auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

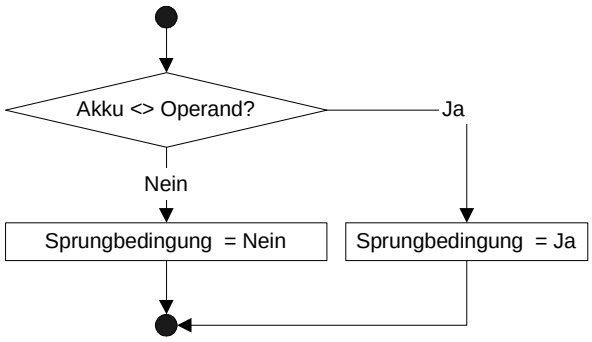
Opcode 87: Teste Akku > Operand

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf Ja, wenn der Akku größer als der Operand ist. Setzt die Sprungbedingung im anderen Fall auf Nein. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - | Steppointer: - | Sprungbedingung: -

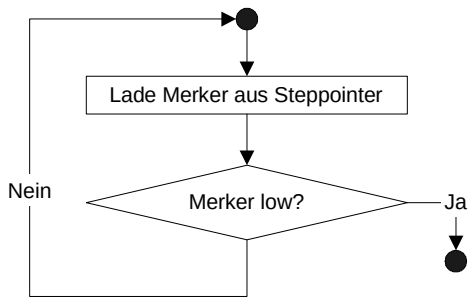
Opcode 88: Teste Akku = Operand

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf Ja, wenn der Akku gleich dem Operand ist. Setzt die Sprungbedingung im anderen Fall auf Nein. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - | Steppointer: - | Sprungbedingung: -

Opcode 89: Teste Akku <> Operand

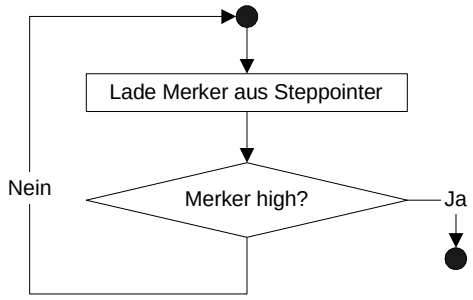
Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf Ja, wenn der Akku ungleich dem Operand ist. Setzt die Sprungbedingung im anderen Fall auf Nein. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - | Steppointer: - | Sprungbedingung: -

Opcode 90: Warte bis Merker low

Blockschaltbild	*Funktionsbeschreibung*
	Wartet, bis der im Operanden angegebene Merker low ist. Kontrolliert wird der Merker des Ablaufes, auf den aktuell der Steppointer zeigt.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf macht weiter, wenn der Merker low ist. Im anderen Fall macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

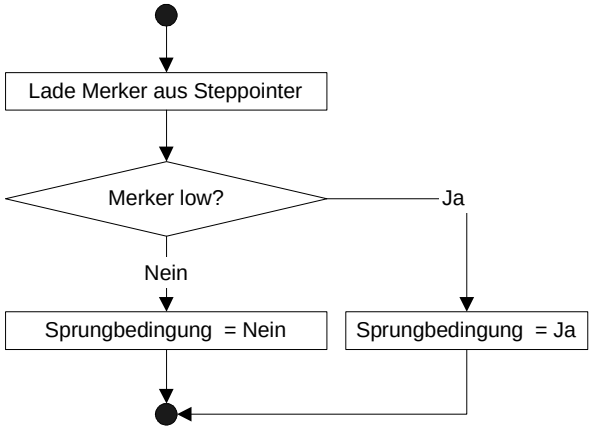
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 91: Warte bis Merker high

Blockschaltbild	*Funktionsbeschreibung*
	Wartet, bis der im Operanden angegebene Merker low ist. Kontrolliert wird der Merker des Ablaufes, auf den aktuell der Steppointer zeigt.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf macht weiter, wenn der Merker high ist. Im anderen Fall macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

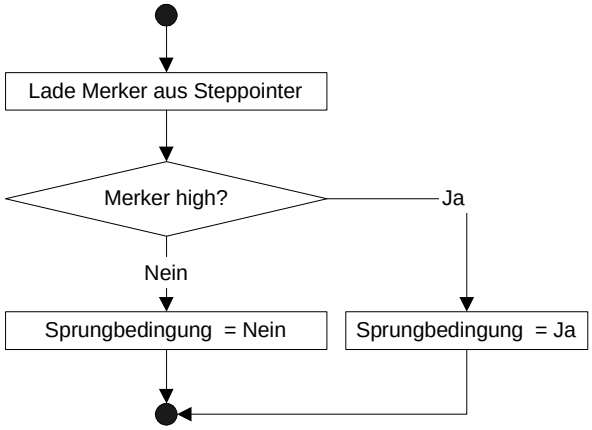
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 92: Teste Merker low

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf Ja, wenn der im Operanden angegebene Merker low ist, setzt die Sprungbedingung im anderen Fall auf Nein. Kontrolliert wird der Merker des Ablaufes, auf den aktuell der Steppointer zeigt.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

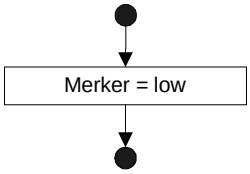
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 93: Teste Merker high

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf Ja, wenn der im Operanden angegebene Merker high ist, setzt die Sprungbedingung im anderen Fall auf Nein. Kontrolliert wird der Merker des Ablaufes, auf den aktuell der Steppointer zeigt.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

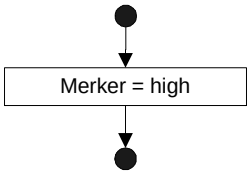
Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 94: Merker löschen

Blockschaltbild	***Funktionsbeschreibung***
	Löscht den Merker, der durch den Operanden angegeben wird. Gelöscht wird immer der eigene Merker, unabhängig davon, auf welchen Ablauf der Steppointer zeigt.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

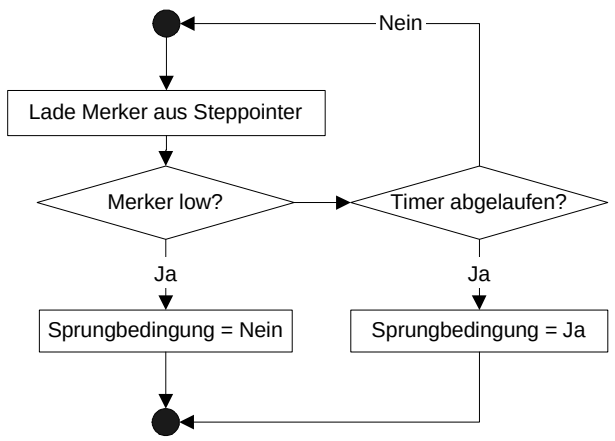
Opcode 95: Merker setzen

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den Merker, der durch den Operanden angegeben wird. Gesetzt wird immer der eigene Merker, unabhängig davon, auf welchen Ablauf der Steppointer zeigt.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 96: Warte Merker low / Zeit

Blockschaltbild



Funktionsbeschreibung

Der Ablauf wartet, bis der Merker mit der im Operanden angegebenen Nummer low ist oder bis der interne Timer abgelaufen ist. Kontrolliert wird der Merker des Ablaufs, auf den aktuell der Steppointer zeigt.

Ablauf macht nach Ausführung weiter?

Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Merker low ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.

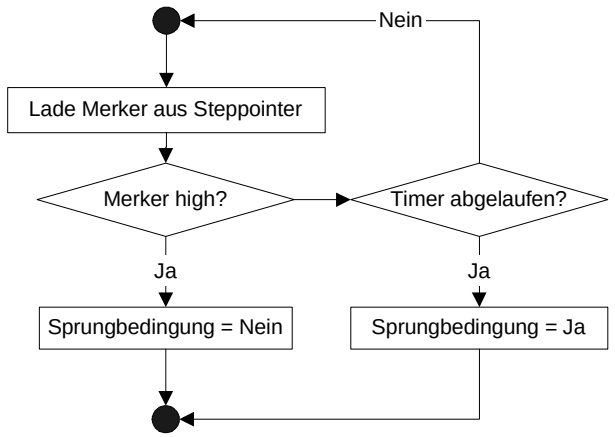
Sprungbedingung wird verändert: Ja

Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 97: Warte Merker high / Zeit

Blockschaltbild



Funktionsbeschreibung

Der Ablauf wartet, bis der Merker mit der im Operanden angegebenen Nummer high ist oder bis der interne Timer abgelaufen ist. Kontrolliert wird der Merker des Ablaufs, auf den aktuell der Steppointer zeigt.

Ablauf macht nach Ausführung weiter?

Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Merker high ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.

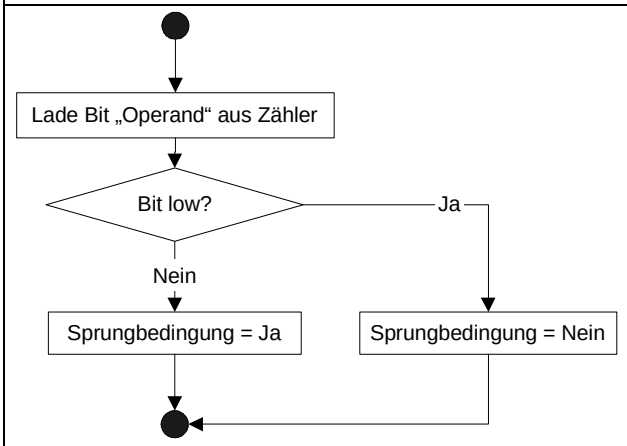
Sprungbedingung wird verändert: Ja

Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	X	-

Opcode 98: Teste Bit im Zähler

Blockschaltbild



Funktionsbeschreibung

Setzt die Sprungbedingung auf Ja, wenn das durch den Operanden angegebene Bit im Zähler high ist. Setzt die Sprungbedingung im anderen Fall auf Nein.

Ablauf macht nach Ausführung weiter?

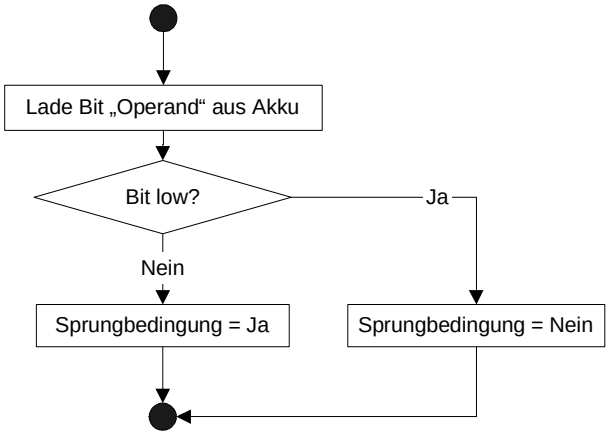
Ja

Sprungbedingung wird verändert: Ja

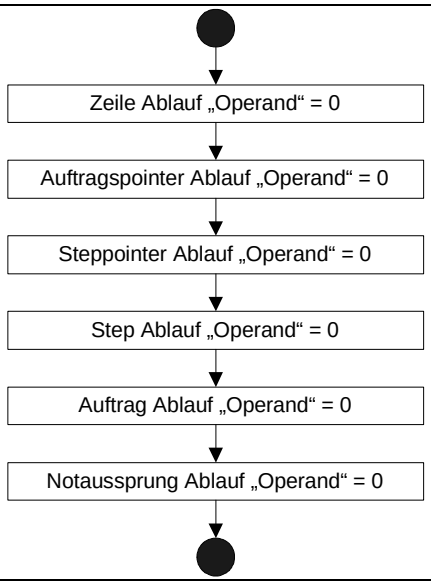
Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

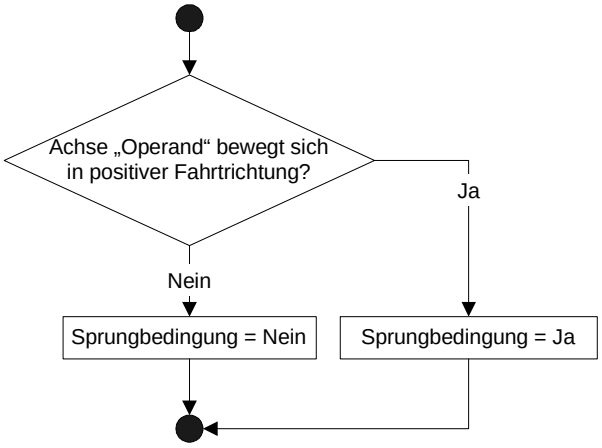
Opcode 99: Teste Bit im Akku

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf Ja, wenn das durch den Operanden angegebene Bit im Akku high ist. Setzt die Sprungbedingung im anderen Fall auf Nein. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: -

Opcode 100: Reset Ablauf Operand

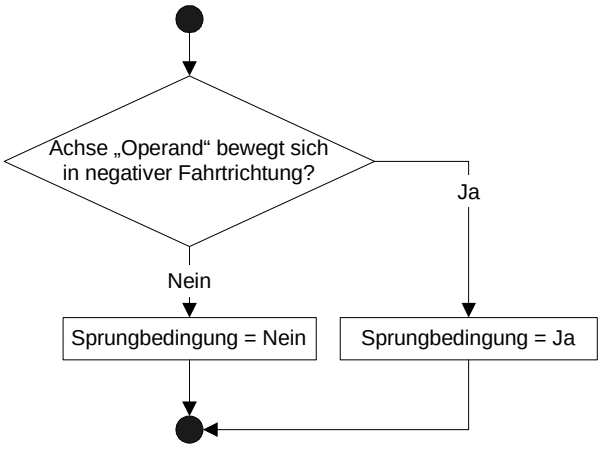
Blockschaltbild	***Funktionsbeschreibung***
	Der im Operanden angegebene Ablauf wird auf die Zeile 1 gesetzt, der Auftragspointer, der Steppointer, der Auftrag, der Step und der Notaussprung des entsprechenden Ablaufs werden gelöscht. Sollte der Ablauf nicht in die Ausführung eingehängt sein, wird er eingehängt. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: -

Opcode 101: Teste Achse vor

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf Ja, wenn die durch den Operanden angegebene Achse sich in positiver Richtung bewegt. Setzt die Sprungbedingung im anderen Fall auf Nein. *In Softwareversionen ab 8.00: Der Operand 0 benutzt die Achse, die dem Ablauf in der Zuordnungstabelle zugewiesen ist*
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

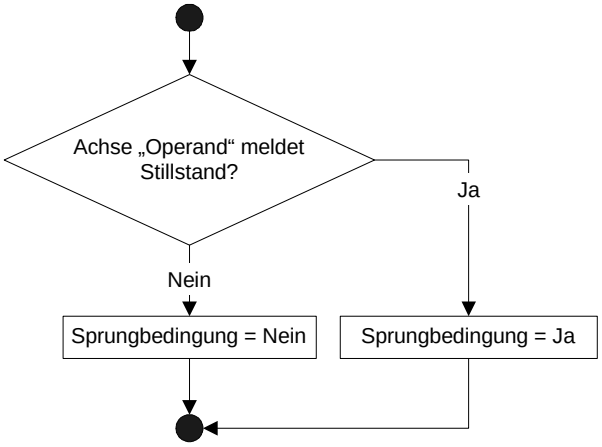
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 102: Teste Achse rück

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf Ja, wenn die durch den Operanden angegebene Achse sich in negativer Richtung bewegt. Setzt die Sprungbedingung im anderen Fall auf Nein. *In Softwareversionen ab 8.00: Der Operand 0 benutzt die Achse, die dem Ablauf in der Zuordnungstabelle zugewiesen ist*
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

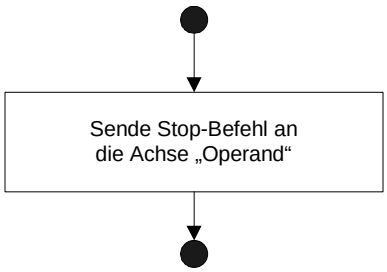
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 103: Teste Achse Stillstand

Blockschaltbild	***Funktionsbeschreibung***
	Setzt die Sprungbedingung auf Ja, wenn die durch den Operanden angegebene Achse Stillstand meldet. Setzt die Sprungbedingung im anderen Fall auf Nein. *In Softwareversionen ab 8.00: Der Operand 0 benutzt die Achse, die dem Ablauf in der Zuordnungstabelle zugewiesen ist*
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

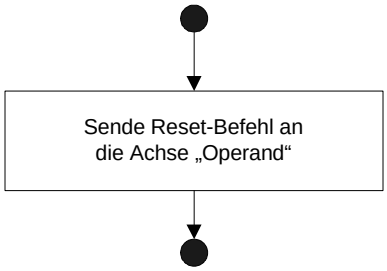
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 104: Stop Achse

Blockschaltbild	***Funktionsbeschreibung***
	Sendet an die im Operanden angegebene Achse einen Stop-Befehl. Ist der Operand Null, so wird der Befehl an alle Achsen gesendet, die zur gleichen Gruppe wie der aufrufenden Ablauf gehören.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 105: Reset Achse

Blockschaltbild	***Funktionsbeschreibung***
	Sendet an die im Operanden angegebene Achse einen Reset-Befehl. Ist der Operand Null, so wird der Befehl an alle Achsen gesendet, die zur gleichen Gruppe wie der aufrufenden Ablauf gehören.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 106: Positioniersperre setzen

Blockschaltbild	***Funktionsbeschreibung***
	Armiert für die im Operanden angegebene Achse eine Positioniersperre. Solange diese Sperre armiert ist, nimmt die Achse keine weiteren Positionierbefehle an. Sollte zum Zeitpunkt der Ausführung die Achse bereits fahren, führt sie diese Fahrbewegung zu Ende. *In Softwareversionen ab 8.00: Der Operand 0 benutzt die Achse, die dem Ablauf in der Zuordnungstabelle zugewiesen ist*
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

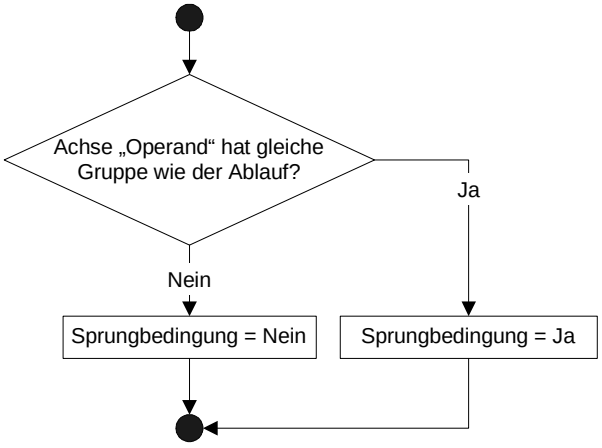
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 107: Positioniersperre löschen

Blockschaltbild	*Funktionsbeschreibung*
	Löscht die Positioniersperre für die im Operanden angegebene Achse. Sollte nach dem Löschen ein gültiger Fahrbefehl für die Achse anliegen, der während der Armierung der Sperre eingetragen worden ist, wird dieser Befehl ausgeführt. *In Softwareversionen ab 8.00: Der Operand 0 benutzt die Achse, die dem Ablauf in der Zuordnungstabelle zugewiesen ist*
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

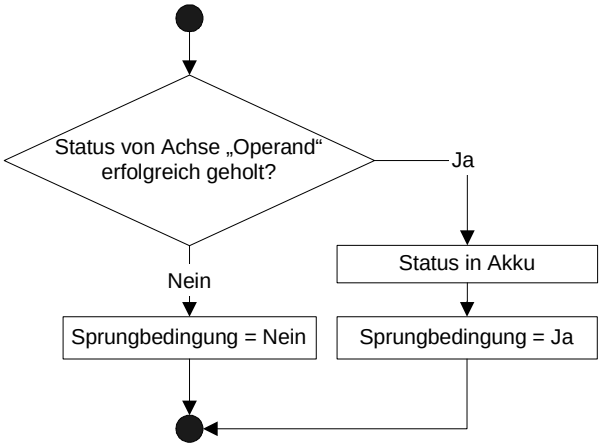
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 108: Vergleiche Gruppe Achse/Ablauf

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf Ja, wenn die im Operanden angegebene Achse zur gleichen Gruppe gehört wie der Ablauf. Setzt im anderen Fall die Sprungbedingung auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

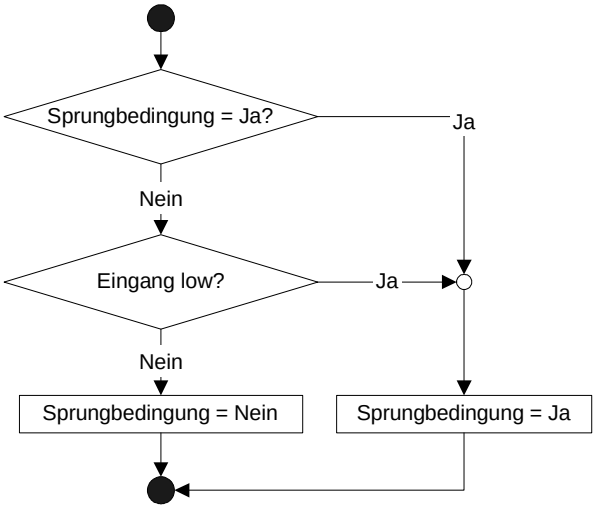
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 109: Status Achse in Operand

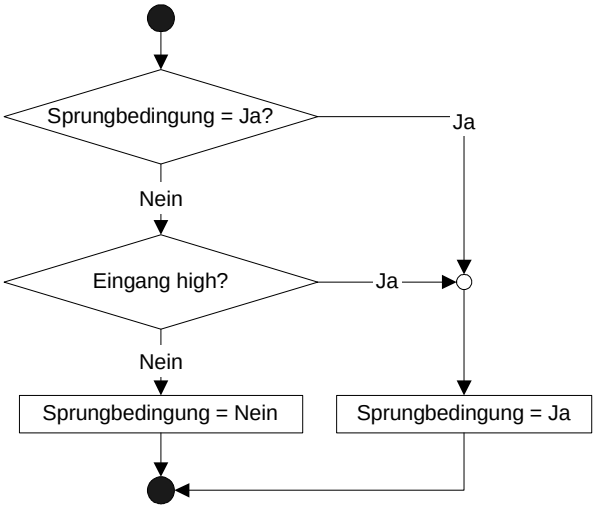
Blockschaltbild	*Funktionsbeschreibung*
	Lädt den Status der im Operanden angegebenen Achse in den Akku. Setzt die Sprungbedingung auf Ja, wenn die Anfrage okay ist. Setzt die Sprungbedingung auf Nein, wenn die Anfrage ungültig ist. ***Implementiert in Softwareständen ab 10.03.2006*** *In Softwareversionen ab 8.00: Der Operand 0 benutzt die Achse, die dem Ablauf in der Zuordnungstabelle zugewiesen ist*
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 110: Oder Eingang low

Blockschaltbild	***Funktionsbeschreibung***
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Eingang low“ eine boolsche Oder-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - ; Steppointer: - ; Sprungbedingung: X

Opcode 111: Oder Eingang high

Blockschaltbild	***Funktionsbeschreibung***
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Eingang high“ eine boolsche Oder-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - ; Steppointer: - ; Sprungbedingung: X

Opcode 114: Pseudo-Eingang löschen

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den Pseudo-Eingang mit der im Operanden angegebenen Nummer auf low. Dieser Eingang darf in der Konfiguration nicht angelegt sein. ***Implementiert in Systemen ab 08.2011***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

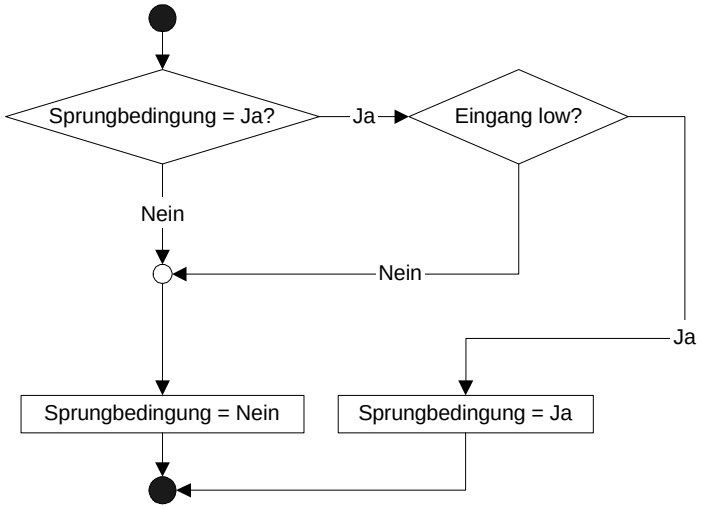
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 115: Pseudo-Eingang setzen

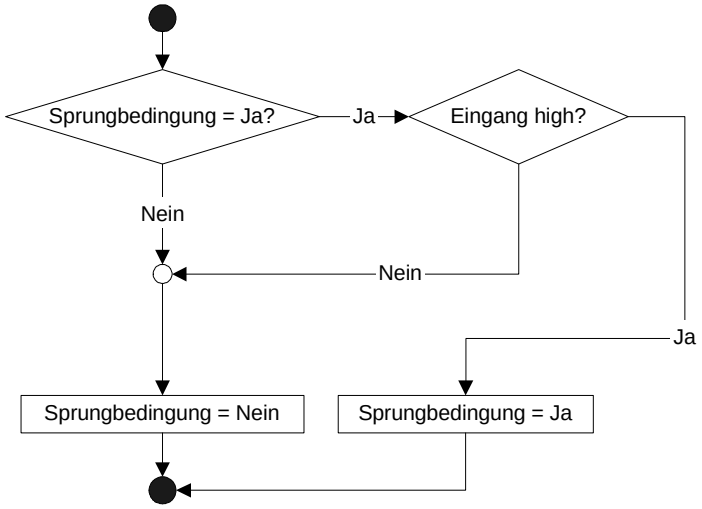
Blockschaltbild	***Funktionsbeschreibung***
	Setzt den Pseudo-Eingang mit der im Operanden angegebenen Nummer auf high. Dieser Eingang darf in der Konfiguration nicht angelegt sein. ***Implementiert in Systemen ab 08.2011***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 112: Und Eingang low

Blockschaltbild	*Funktionsbeschreibung*
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Eingang low“ eine boolsche Und-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: X

Opcode 113: Und Eingang high

Blockschaltbild	*Funktionsbeschreibung*
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Eingang high“ eine boolsche Und-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: X

Opcode 116: Analogwert in Zähler laden

Blockschaltbild	***Funktionsbeschreibung***
	Der Analogwert aus dem im Operanden angegebenen logischen Analog-Eingang wird in den Zähler geladen. Die Adressierung der analogen Eingänge erfolgt dabei über die entsprechende Konfigurationsdatei. ***Implementiert in Softwareständen ab Januar 2008***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

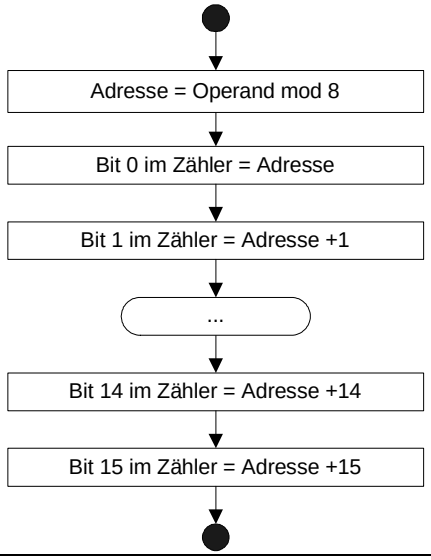
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 117: Analogwert in Akku laden

Blockschaltbild	***Funktionsbeschreibung***
	Der Analogwert aus dem im Operanden angegebenen logischen Analog-Eingang wird in den Akku geladen. Die Adressierung der analogen Eingänge erfolgt dabei über die entsprechende Konfigurationsdatei. ***Implementiert in Softwareständen ab Januar 2008***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

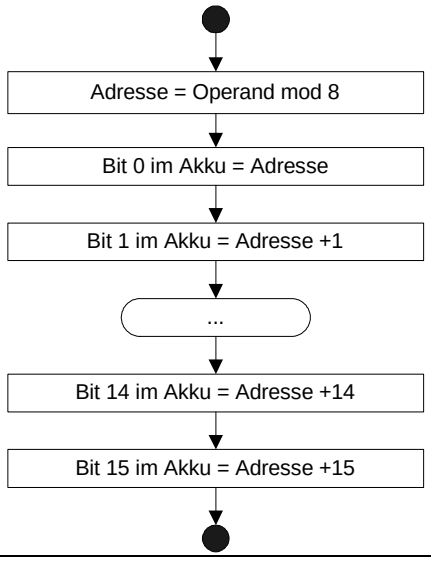
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 118: Lade Eingangsgruppe (Word) in Zähler

Blockschaltbild	***Funktionsbeschreibung***
	Liest ein vollständiges Word aus dem Mapping der Eingänge. Das Word wird in den Zähler geschrieben. Der erste Eingang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

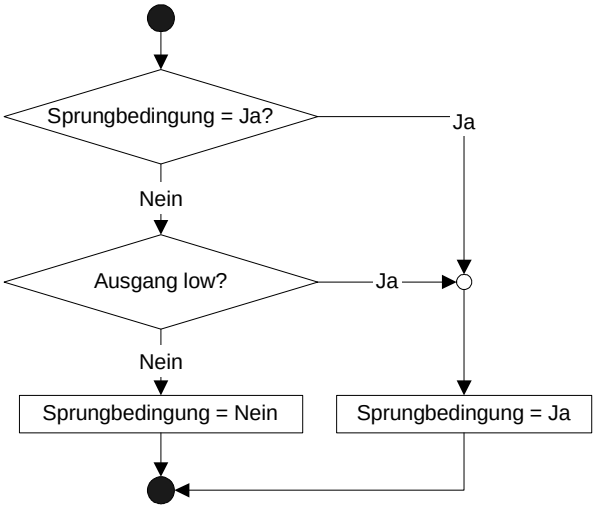
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 119: Lade Eingangsgruppe (Word) in Akku

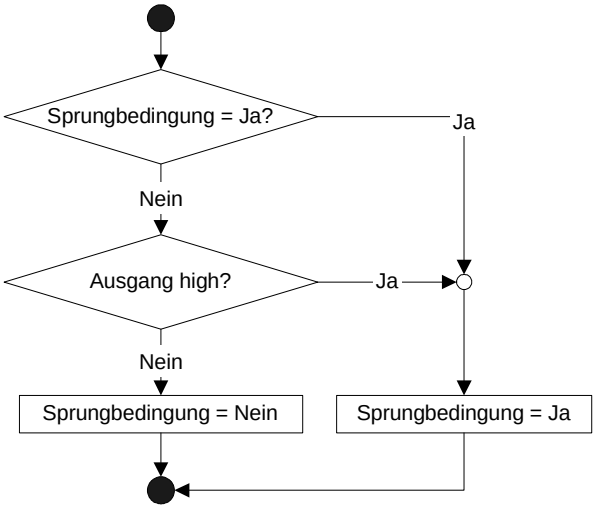
Blockschaltbild	***Funktionsbeschreibung***
	Lädt ein vollständiges Word aus dem Mapping der Eingänge. Das Word wird in den Akku geschrieben. Der erste Eingang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

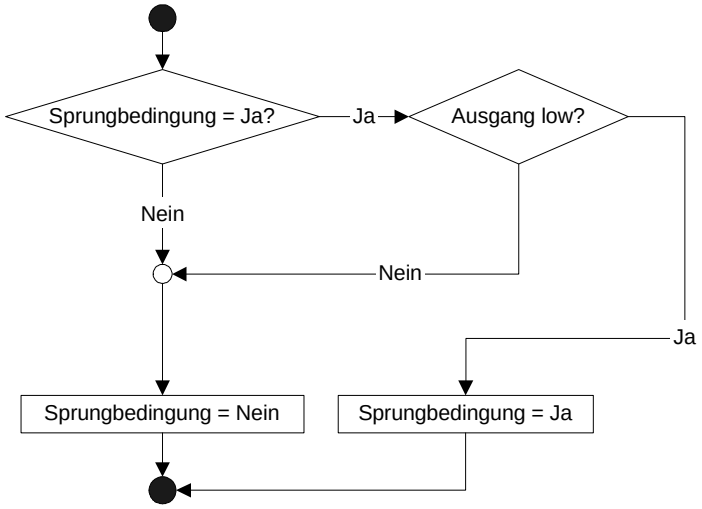
Opcode 120: Oder Ausgang low

Blockschaltbild	*Funktionsbeschreibung*
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Ausgang low“ eine boolsche Oder-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: X

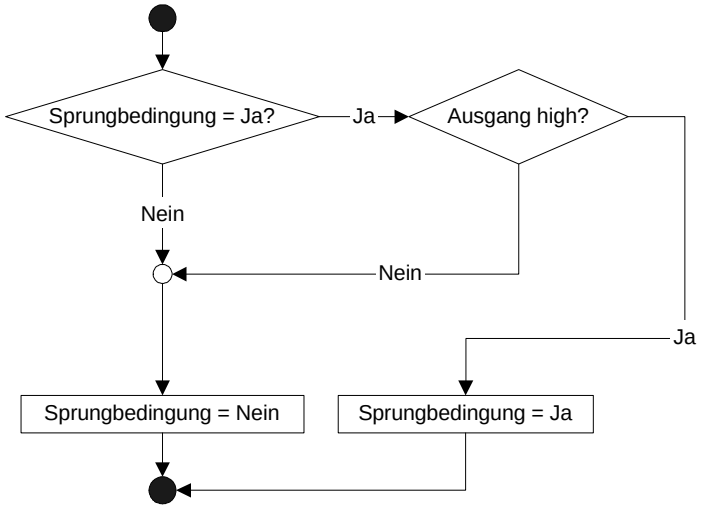
Opcode 121: Oder Ausgang high

Blockschaltbild	*Funktionsbeschreibung*
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Ausgang high“ eine boolsche Oder-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: X

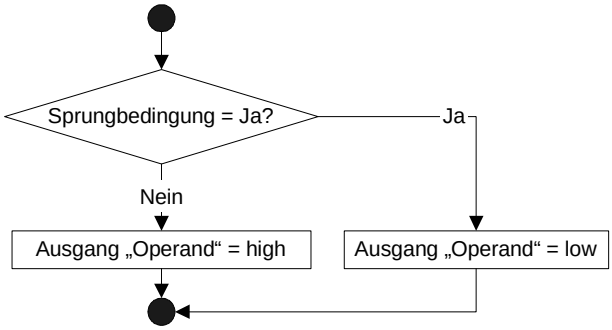
Opcode 122: Und Ausgang low

Blockschaltbild	*Funktionsbeschreibung*
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Ausgang low“ eine boolsche Und-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: X

Opcode 123: Und Ausgang high

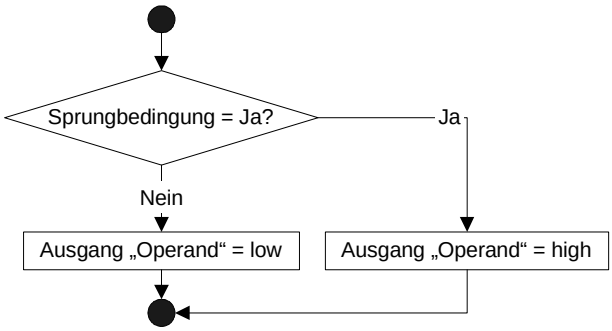
Blockschaltbild	*Funktionsbeschreibung*
	Führt zwischen der aktuellen Sprungbedingung und der Abfrage „Ausgang high“ eine boolsche Und-Verknüpfung durch und schreibt das Ergebnis in die Sprungbedingung. ***Ablauf macht nach Ausführung weiter?*** Ja ***Sprungbedingung wird verändert:*** Ja ***Einfluss auf die Befehlsausführung*** Auftragspointer: - Steppointer: - Sprungbedingung: X

Opcode 124: Lösche wenn ja, setzte wenn nein

Blockschaltbild	***Funktionsbeschreibung***
	Löscht den im Operanden angegebenen Ausgang, wenn die Sprungbedingung auf Ja steht. Setzt den Ausgang im anderen Fall.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	X

Opcode 125: Setze wenn ja, lösche wenn nein

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den im Operanden angegebenen Ausgang, wenn die Sprungbedingung auf Ja steht. Löscht den Ausgang im anderen Fall.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	X

Opcode 126: Analogwert setzen aus Zähler

Blockschaltbild	***Funktionsbeschreibung***
	Der aktuelle Zählerinhalt wird in den logischen Analog-Ausgang geschrieben, der im Operanden angegeben ist. Die Adressierung erfolgt anhand der Konfiguration der analogen Ausgänge. ***Implementiert in Softwareständen ab Januar 2008***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

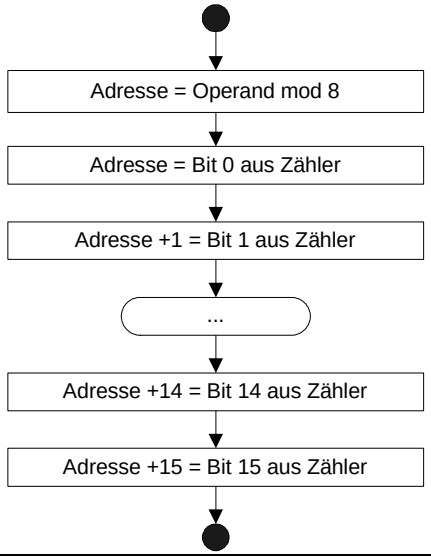
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 127: Analogwert setzen aus Akku

Blockschaltbild	***Funktionsbeschreibung***
	Der aktuelle Akkuinhalt wird in den logischen Analog-Ausgang geschrieben, der im Operanden angegeben ist. Die Adressierung erfolgt anhand der Konfiguration der analogen Ausgänge. ***Implementiert in Softwareständen ab Januar 2008***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

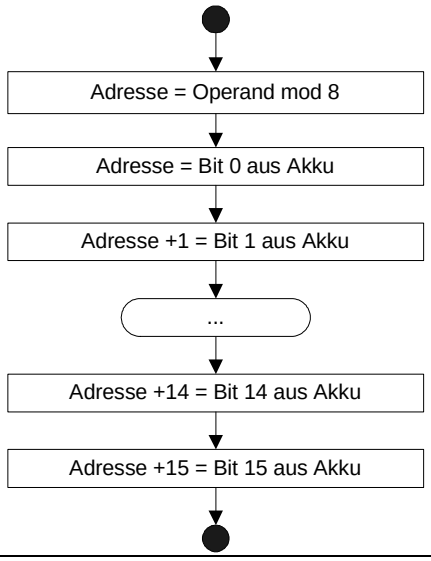
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 128: Lade Zähler in Ausgangsgruppe (Word)

Blockschaltbild	***Funktionsbeschreibung***
	Schreibt ein vollständiges Word in das Mapping der Ausgänge. Das zu schreibende Word steht im Zähler. Der erste Ausgang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 129: Lade Akku in Ausgangsgruppe (Word)

Blockschaltbild	***Funktionsbeschreibung***
	Schreibt ein vollständiges Word in das Mapping der Ausgänge. Das zu schreibende Word steht im Akku. Der erste Ausgang, der betroffen ist, ergibt sich aus dem Operanden modulo 8.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 180: Akku komplementieren

Blockschaltbild	*Funktionsbeschreibung*
	Erzeugt ein bitweises Komplement des Akkus. Alle Bits, die im Wert des Akkus gesetzt waren, sind anschließend gelöscht. Alle Bits, die im Wert des Akkus gelöscht waren, sind anschließend gesetzt. Der neue Wert steht wieder im Akku und überschreibt damit den alten Wert.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 13.10.2004

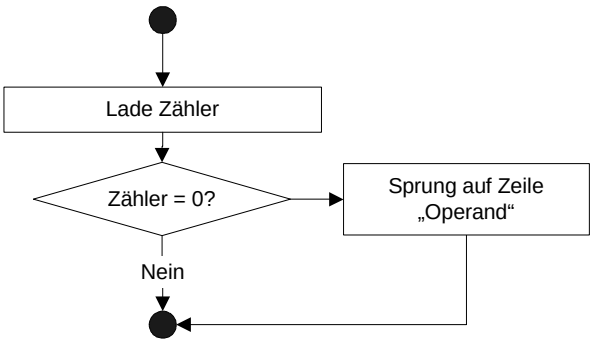
Opcode 181: Akku negieren

Blockschaltbild	*Funktionsbeschreibung*
	Multipliziert den Wert im Akku mit –1 und schreibt den neuen Wert zurück in den Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 13.10.2004

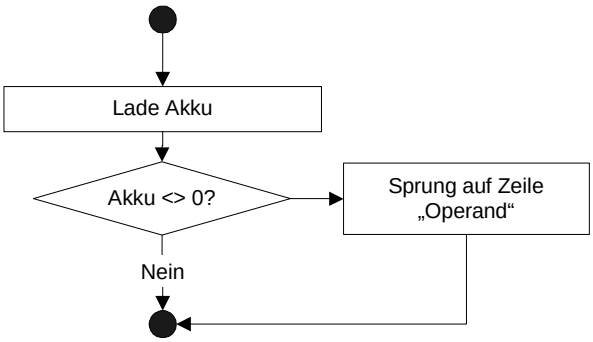
Opcode 182: Sprung wenn Akku = 0

Blockschaltbild	*Funktionsbeschreibung*
	Springt auf die im Operanden angegebene Zeile, wenn der Akku Null ist.
	Ablauf macht nach Ausführung weiter?
	Ist der Akku Null, dann springt der Ablauf auf die angegebene Zeile, anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter. Ist der Akku ungleich Null, macht der Ablauf mit der nächsten Zeile weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 13.10.2004

Opcode 183: Sprung wenn Akku <> 0

Blockschaltbild	*Funktionsbeschreibung*
	Springt auf die im Operanden angegebene Zeile, wenn der Akku ungleich Null ist.
	Ablauf macht nach Ausführung weiter?
	Ist der Akku ungleich Null, dann springt der Ablauf auf die angegebene Zeile, anschließend macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter. Ist der Akku Null, macht der Ablauf mit der nächsten Zeile weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 13.10.2004

Opcode 184: Akku = Akku „and“ Operand

Blockschaltbild	*Funktionsbeschreibung*
	Führt ein binäres AND zwischen Akku und Operand aus. Der neue Wert steht anschließend im Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 13.10.2004

Opcode 185: Akku = Akku „or“ Operand

Blockschaltbild	*Funktionsbeschreibung*
	Führt ein binäres OR zwischen Akku und Operand aus. Der neue Wert steht anschließend im Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 13.10.2004

Opcode 186: Akku = Akku „xor“ Operand

Blockschaltbild	*Funktionsbeschreibung*
	Führt ein binäres XOR zwischen Akku und Operand aus. Der neue Wert steht anschließend im Akku.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 13.10.2004

Opcode 188: BCD -> Binär

Blockschaltbild	*Funktionsbeschreibung*
	Der BCD-Wert im Akku wird in einen binären Wert gewandelt. Der neue Wert steht dann wieder im Akku
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 20.08.2006

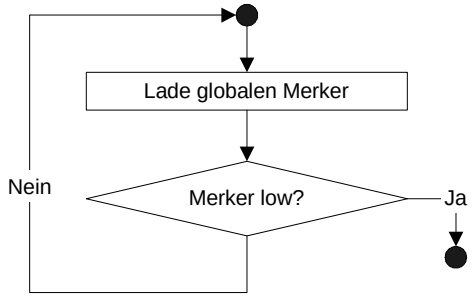
Opcode 189: Graycodekonvertierung

Blockschaltbild	*Funktionsbeschreibung*
	Benutzt den Wert im Akku, um eine Graycodekonvertierung durch zu führen. Dabei gibt der Operand die Breite des Graycode in Bits inklusive Invertierbit an. Die oberen Bits im Akku bleiben unberücksichtigt und müssen nicht gelöscht werden.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

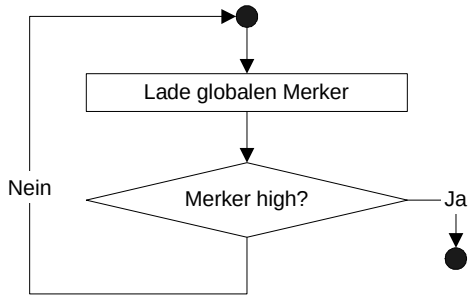
Implementiert in Softwareständen ab 13.10.2004

Opcode 190: Warte bis globaler Merker low

Blockschaltbild	*Funktionsbeschreibung*
	Wartet, bis der im Operanden angegebene globale Merker low ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf macht weiter, wenn der Merker low ist. Im anderen Fall macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

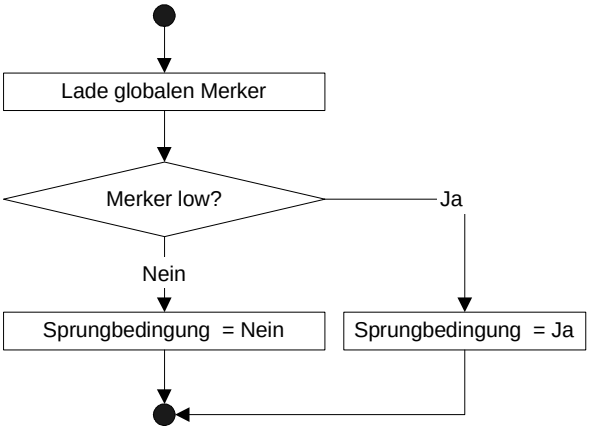
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 191: Warte bis globaler Merker high

Blockschaltbild	*Funktionsbeschreibung*
	Wartet, bis der im Operanden angegebene globale Merker high ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf macht weiter, wenn der Merker high ist. Im anderen Fall macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

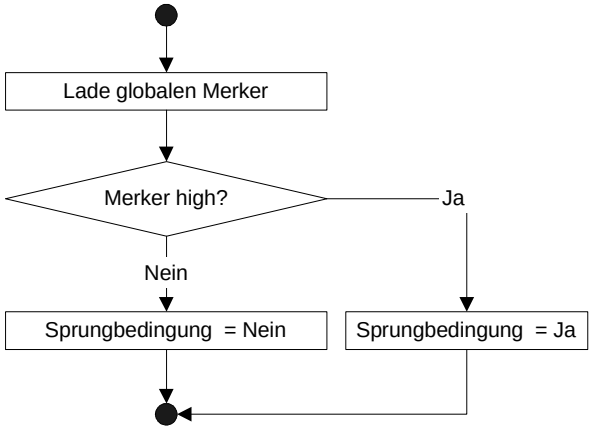
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 192: Teste globaler Merker low

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf Ja, wenn der im Operanden angegebene globale Merker low ist, setzt die Sprungbedingung im anderen Fall auf Nein.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

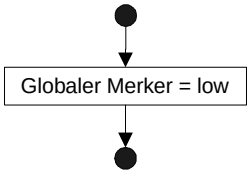
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 193: Teste globaler Merker high

Blockschaltbild	*Funktionsbeschreibung*
	Setzt die Sprungbedingung auf Ja, wenn der im Operanden angegebene globale Merker high ist, setzt die Sprungbedingung im anderen Fall auf Nein
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

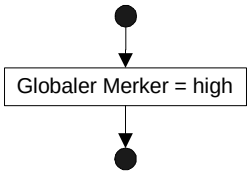
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 194: Globalen Merker löschen

Blockschaltbild	*Funktionsbeschreibung*
	Löscht den globalen Merker, der durch den Operanden angegeben wird.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

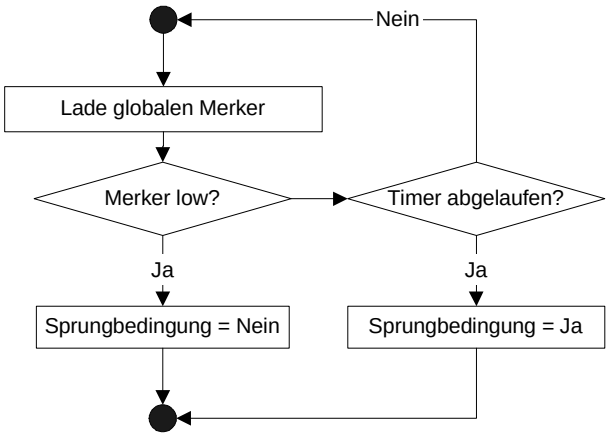
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 195: Globalen Merker setzen

Blockschaltbild	*Funktionsbeschreibung*
	Setzt den globalen Merker, der durch den Operanden angegeben wird.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

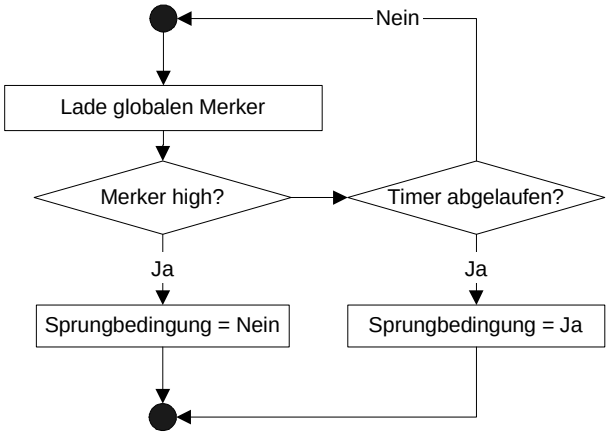
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 196: Warte globaler Merker low / Zeit

Blockschaltbild	*Funktionsbeschreibung*
	Der Ablauf wartet, bis der globale Merker mit der im Operanden angegebenen Nummer low ist oder bis der interne Timer abgelaufen ist
	Ablauf macht nach Ausführung weiter?
	Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Merker low ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 197: Warte globaler Merker high / Zeit

Blockschaltbild	*Funktionsbeschreibung*
	Der Ablauf wartet, bis der globale Merker mit der im Operanden angegebenen Nummer high ist oder bis der interne Timer abgelaufen ist.
	Ablauf macht nach Ausführung weiter?
	Der Ablauf arbeitet auf der nächsten Zeile weiter, wenn der Merker high ist oder wenn der interne Timer abgelaufen ist. Ansonsten macht die Ablaufprogrammierung mit dem nächsten Ablauf weiter.
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 198: Eigene Störung löschen

Blockschaltbild	*Funktionsbeschreibung*
	Löscht die eigene Störung, die durch den Befehl 2 eingetragen worden ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 199: Merker in allen Abläufen löschen

Blockschaltbild	*Funktionsbeschreibung*
	Dieser Befehl bricht die Regel, dass ablaufabhängige Merker nur vom eigenen Ablauf verändert werden können! Der Befehl löscht den im Operanden angegebenen, ablaufabhängigen Merker in allen existierenden Abläufen. Ein Operand 999 löscht alle ablaufabhängigen Merker in allen Abläufen.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 200: Eigenen WCB schieben

Blockschaltbild	*Funktionsbeschreibung*
	Es werden der eigene Rekord plus Anzahl Operand Folgerekords geschoben. Das Merkerbyte im eigenen Rekord wird gelöscht. Wenn die Funktion ausgeführt ist, wird die Sprungbedingung gesetzt, ansonsten wird sie gelöscht.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 201: Eigenen WCB ringförmig schieben

Blockschaltbild	*Funktionsbeschreibung*
	Es werden der eigene Rekord plus Anzahl Operand Folgerekords geschoben. Anschliessend wird der letzte geschobene Record auf den eigenen WCB kopiert. Daraus ergibt sich, dass für n Plätze n+1 WCB's gebraucht werden. Wenn die Funktion ausgeführt ist, wird die Sprungbedingung gesetzt, ansonsten wird sie gelöscht.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 202: Meldung in Logdatei schreiben

Blockschaltbild	***Funktionsbeschreibung***
	Schreibt die als Bedienerführung gedachte SPS-Meldung mit der im Operanden angegebenen Nummer in die Logdatei.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 203: WCBs rückwärts durchsuchen

Blockschaltbild	***Funktionsbeschreibung***
	Durchsucht die WCB-Datei rückwärts, beginnend beim eigenen Platz, endend beim eigenen Platz minus Operand. Wird ein belegter Platz gefunden, der nicht der eigene Platz ist, wird die Sprungbedingung gesetzt und in den Zähler der Wert des über den Befehl 6 adressierten Datenfeldes geschrieben. Wird kein Platz gefunden, dann wird der Zähler gelöscht.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 204: WCB-Pointer auf eigenen Platz

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den WCB-Pointer auf den Rekord, der diesem Ablauf laut Zuordnungstabelle zugewiesen ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 205: WCB-Pointer minus Operand

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den WCB-Pointer auf den aktuellen Rekord minus den Operanden.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 206: WCB-Pointer plus Operand

Blockschaltbild	***Funktionsbeschreibung***
	Setzt den WCB-Pointer auf den aktuellen Rekord plus den Operanden.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 207: Eigenen WCB auf Platz (Operand) schieben

Blockschaltbild	***Funktionsbeschreibung***
	Es wird der eigene Rekord auf den durch den Operanden adressierten Platz geschoben. Das Merkerbyte im eigenen Rekord wird gelöscht. Wenn die Funktion ausgeführt ist, wird die Sprungbedingung gesetzt, ansonsten wird sie gelöscht.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 208: BDE-Wert addieren

Blockschaltbild	***Funktionsbeschreibung***
	Das Feld Wert im Rekord Operand der BDE-Datei wird um den Wert erhöht, auf den der Pointer der externen Daten zeigt.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 209: Zykluszähler plus 1

Blockschaltbild	*Funktionsbeschreibung*
	Erhöht den Zykluszähler (der Zähler, der auch in der Logdatei angezeigt wird) um den Wert 1. ***Implementiert in Softwareständen ab 12.12.2003.*** Ist der Operand ungleich Null, so wird nicht der Standard-Zähler erhöht, sondern der Zähler aus dem entsprechenden Rekord der BDE-Datei. ***Implementiert in Softwareständen ab 20.08.2006***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 210: Achse positionieren

Blockschaltbild	*Funktionsbeschreibung*
	Dieser Befehl fährt die Achse, die in der Zuordnungstabelle angegeben ist, auf ein Zielmass. Im Gegensatz zum Befehl 211 wird eine zugeordneter Achsprozess ignoriert und nicht ausgeführt. Die Zielposition wird aus dem Werkstückkontrollblock entnommen und zwar von der Stelle, an der aktuell der Datenpointer steht. Ist der Datenpointer 0, wird die Achse auf ihren Minimalwert gefahren. Ist der Datenpointer 2000, wird die Achse auf ihren Maximalwert gefahren. Zu Beginn des Befehl werden die ablaufabhängigen Merker 0 und 1 gelöscht. Ist die Achse im Ziel, wird der Merker 0 gesetzt. ***Implementiert in Softwareständen ab Januar 2008***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Wichtig! **Dieser Befehl startet die Zielfahrt der Achse nur an. Die SPS macht anschließend sofort mit der nächsten Zeile weiter und die Achsbewegung läuft im Hintergrund. Soll der Ablauf darauf warten, dass die Achse im Ziel ist, so muss dies über eine Abfrage des Merkers gelöst werden!**

Beispiel: Das nachfolgende Beispiel zeigt, wie ein Achse gestartet wird und der Ablauf anschließend darauf wartet, dass die Achse ins Ziel kommt.

```
1   211 0     // Starte Achse
2   93        0     // Teste Merker 0 auf high
3   70        2     // Springen, wenn nicht
```

Opcode 211: Achse positionieren

Blockschaltbild	***Funktionsbeschreibung***
	Dieser Befehl fährt die Achse, die in der Zuordnungstabelle angegeben ist, auf ein Zielmass. Ist in der Zuordnungstabelle ein Achsprozess angegeben, so wird dieser gestartet um das Ziel zu ermitteln, ansonsten wird die Zielposition aus dem Werkstückkontrollblock entnommen und zwar von der Stelle, an der aktuell der Datenpointer steht. Ist der Datenpointer 0, wird die Achse auf ihren Minimalwert gefahren. Ist der Datenpointer 2000, wird die Achse auf ihren Maximalwert gefahren. Zu Beginn des Befehl werden die ablaufabhängigen Merker 0 und 1 gelöscht. Ist die Achse im Ziel, wird der Merker 0 gesetzt. ***Implementiert in Softwareständen ab 08.08.2004***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Wichtig! **Dieser Befehl startet die Zielfahrt der Achse nur an. Die SPS macht anschließend sofort mit der nächsten Zeile weiter und die Achsbewegung läuft im Hintergrund. Soll der Ablauf darauf warten, dass die Achse im Ziel ist, so muss dies über eine Abfrage des Merkers gelöst werden!**

Beispiel: Das nachfolgende Beispiel zeigt, wie ein Achse gestartet wird und der Ablauf anschließend darauf wartet, dass die Achse ins Ziel kommt.

```
4   211 0     // Starte Achse
5   93        0     // Teste Merker 0 auf high
6   70        2     // Springen, wenn nicht
```

Opcode 212: Achse Massabfrage

Blockschaltbild	***Funktionsbeschreibung***
	Dieser Befehl testet die Achse, die in der Zuordnungstabelle angegeben ist, auf ein Zielmass. Ist in der Zuordnungstabelle ein Achsprozess angegeben, so wird dieser gestartet um das Ziel zu ermitteln, ansonsten wird die Zielposition aus dem Werkstückkontrollblock entnommen und zwar von der Stelle, an der aktuell der Datenpointer steht. Ist der Datenpointer 0, wird die Achse auf ihren Minimalwert getestet. Ist der Datenpointer 2000, wird die Achse auf ihren Maximalwert getestet. Zu Beginn des Befehl werden die ablaufabhängigen Merker 0 und 1 gelöscht. Ist die Achse zum Zeitpunkt der Abfrage auf der Zielposition, wird der Merker 1 gesetzt. ***Implementiert in Softwareständen ab 08.08.2004***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 215: Pointer WCA auf Feld laut DBH

Blockschaltbild	***Funktionsbeschreibung***
	Dieser Befehl setzt den Datenpointer der WCA auf das Feld mit der Nummer, die im Operanden angegeben worden ist. Die Nummerierung bezieht sich dabei auf die DBH der WCA und der erste Eintrag der DBH (in der Regel die Satznummer) zählt bei der Adressierung bereits mit. Wird das angegebene Feld nicht gefunden, so wird die Sprungbedingung auf Nein gesetzt
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 216: Pointer WCB auf Feld laut DBH

Blockschaltbild	***Funktionsbeschreibung***
	Dieser Befehl setzt den Datenpointer der WCB auf das Feld mit der Nummer, die im Operanden angegeben worden ist. Die Nummerierung bezieht sich dabei auf die DBH der WCB und der erste Eintrag der DBH (in der Regel die Satznummer) zählt bei der Adressierung bereits mit. Wird das angegebene Feld nicht gefunden, so wird die Sprungbedingung auf Nein gesetzt. ***Implementiert in Softwareständen ab 08.08.2004***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 247: Bild-Refresh

Blockschaltbild	*Funktionsbeschreibung*
	Dieser Befehl veranlasst, dass das aktuell angezeigte Bild auf der Steuerung neu angezeigt wird. Dabei werden auch alle Dateninhalte des Bildes neu gelesen.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Implementiert in Softwareständen ab 18.01.2007

Opcode 248: Init Abläufe Gruppe

Blockschaltbild	*Funktionsbeschreibung*
	Reset auf alle Abläufe, die zu der Gruppe gehört, die im Operanden aufgeführt ist. Nach dem Reset wird für alle Abläufe ein Initialisierungsauftrage eingetragen. ***Implementiert in Softwareständen ab 13.10.2004***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 249: Init Abläufe Gruppe Konfig

Blockschaltbild	*Funktionsbeschreibung*
	Reset auf alle Abläufe, die in der Konfig stehen, die zu der Gruppe gehört, die im Operanden aufgeführt ist. Nach dem Reset wird für alle Abläufe ein Initialisierungsauftrage eingetragen. ***Implementiert in Softwareständen ab 12.12.2003***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 250: Reset Abläufe Gruppe

Blockschaltbild	*Funktionsbeschreibung*
	Führt einen Reset von allen Abläufen durch, die zu der Gruppe gehören, die im Operanden angegeben sind.Dieser Befehl trägt für die Abläufe KEINEN Initialisierungsauftrag ein. ***Implementiert in Softwareständen ab 12.12.2003***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 251: System-Auftrag

Blockschaltbild	*Funktionsbeschreibung*
	Erteilt dem System den Auftrag, der im Operanden angegebene worden ist. Die möglichen Systemaufträge sind systemabhängig und im Anhang aufgelistet. ***Implementiert in Softwareständen ab 12.12.2003***
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 252: Öffne Editor

Blockschaltbild	*Funktionsbeschreibung*
	Veranlasst, dass ein Editor geöffnet wird. Der Editor ist in einer entsprechenden Definitionsdatei hinterlegt. Der Operand gibt an, welcher Editor aus der Definitionsdatei geöffnet werden soll (einsbasiert). Tatsächlich geöffnet wird der Editor nur im Automatikbild.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

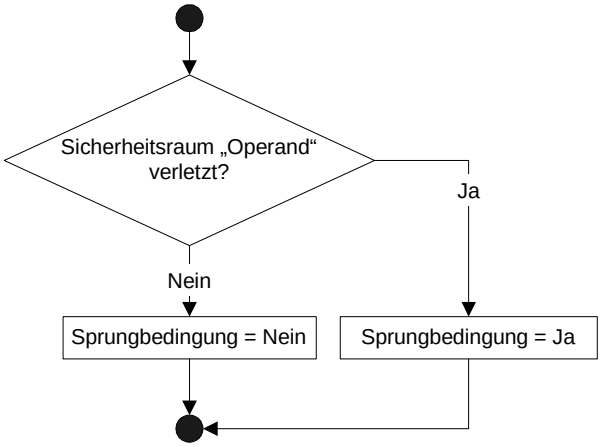
Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 253: Projektiere ASI

Blockschaltbild	*Funktionsbeschreibung*
	Sendet an alle ASI-Koppler einen Projektierungsbefehl.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 254: Sicherheitsraum testen

Blockschaltbild	***Funktionsbeschreibung***
	Testet den im Operanden angegebenen Sicherheitsraum innerhalb der SPS-Sicherheitsraumdatei (einsbasiert). Setzt die Sprungbedingung auf Ja, wenn der Sicherheitsraum verletzt ist.
	Ablauf macht nach Ausführung weiter?
	Ja
	Sprungbedingung wird verändert: Ja
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Opcode 255: Reset System

Blockschaltbild	***Funktionsbeschreibung***
	Führt einen Warmstart des Systems aus und löscht dabei alle Startbedingungen.
	Ablauf macht nach Ausführung weiter?
	Nein. Nach der Ausführung ist auch der Ablauf, der diesen Befehl abgesetzt hat, neu gestartet worden.
	Sprungbedingung wird verändert: Nein
	Einfluss auf die Befehlsausführung

Auftragspointer	Steppointer	Sprungbedingung
-	-	-

Editierung der Ablaufprogrammierung

Dieses Kapitel beschäftigt sich mit allen Dingen, die mit der Editierung von Abläufen und mit der Verwaltung der Ablaufdateien zu tun hat.

Verwaltung der SPS

Die Verwaltung der SPS zeigt eine Übersicht aller Abläufe, die auf der Ramdisk des System gefunden worden sind. Zu jedem Ablauf wird nicht nur die entsprechende Nummer angezeigt, sondern auch der Text, der diesem Ablauf zugeordnet ist. Der Text kann an dieser Stelle auch verändert werden, in dem auf dem zu ändernden Text durch die Taste Enter der Editiermodus geöffnet wird und anschließend der gewünschte Text eingegeben wird. Die Texte werden als Ablaufbezeichnung unter dem Schlüssel `SPSNAME....+Ablaufnummer` in der zentralen Textdatei gespeichert.

Abbildung 5: Verwaltung der Abläufe

Über den Softkey Funktionen im Hauptbild der Verwaltung gelangt man an die einzelnen Funktionen, mit denen Abläufe kopiert, gelöscht und neu angelegt werden können. Von hier aus können auch alle Abläufe gedruckt werden.

Der Softkey *Alle drucken* führt zu einem weiteren Untermenü, in dem die Druckfunktionen ausgewählt werden können. *In Datei* druckt dabei alle Abläufe in Textdateien, wobei für jeden Ablauf eine neue Textdatei angelegt wird, die den Namen des Ablaufs und die Dateiendung TXT trägt. Die Funktion *In Eine* druckt alle Abläufe in eine einzige Textdatei, wobei die Abläufe in dieser Datei numerisch sortiert sind.

Mit der Funktion *Kopieren* kann ein Ablauf kopiert werden. Kopiert wird dabei der Ablauf, auf dem aktuell der Cursor im Bild steht. Die Nummer des Zielablaufs muss eingegeben werden. Ist die eingegebene Zielnummer bereits als Ablauf vorhanden, erscheint eine Fehlermeldung und es wird nicht kopiert. Ist die eingegebene Nummer nicht vorhanden, so wird der Ablauf auf dem der Cursor steht auf den Ablauf mit der eingegebenen Nummer kopiert. Dabei wird auch die Bezeichnung des Ablaufs kopiert.

Die Funktion *Löschen* löscht den Ablauf, auf dem aktuell der Cursor steht. Vor dem Löschen erfolgt eine Sicherheitsabfrage, die mit Ja beantwortet werden muss, wenn der Ablauf wirklich gelöscht werden soll. Gelöscht wird neben dem Ablauf auch der Text, der diesem Ablauf zugeordnet ist.

Mit der Funktion *Neu* kann ein neuer Ablauf angelegt werden. Dazu muss zunächst die gewünschte Ablaufnummer eingegeben werden. Existiert bereits ein Ablauf mit der eingegebenen Nummer, so erscheint eine Fehlermeldung und der Vorgang wird abgebrochen. Existiert die angegebene Nummer nicht, so wird eine entsprechende Datei angelegt. Anschließend fragt das System nach der Anzahl der Zeilen, die dieser Ablauf als Vorgabe erhalten soll. Die Anzahl ist im einzelnen Editieren eines Ablaufs noch veränderbar, es geht in diesem Fall nur darum, dem Ablauf bereits eine Größe geben zu können, um die spätere Editierung zu vereinfachen. Nachdem der Ablauf neu angelegt worden ist, steht der Cursor auf diesem Ablauf. Der Ablauf hat noch keine Bezeichnung – an Stelle der Bezeichnung steht der Schlüssel für die Textdatei. Die Bezeichnung sollte nun an dieser Stelle angegeben werden.

Anwahl von Abläufen zur Editierung

Um einzelne Abläufe editieren zu können, muss die gewünschte Nummer des Ablaufs in die entsprechende Anwahlmaske eingegeben werden. Wird diese Anwahl durch Enter im Nummernfeld oder durch den Softkey Übernahme bestätigt, gelangt man in die direkte Editierung eines Ablaufs.

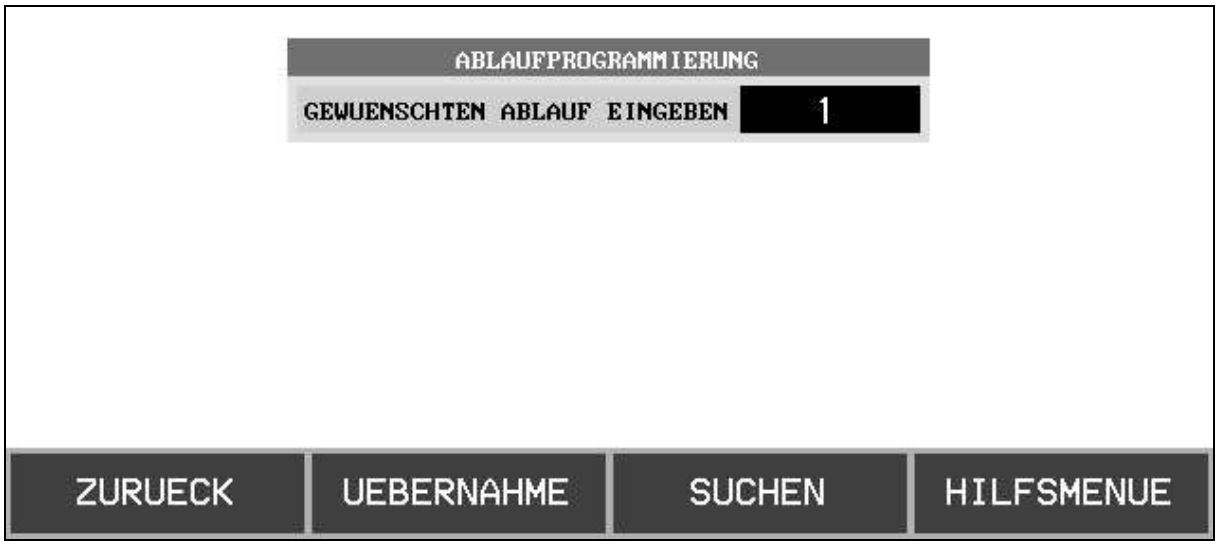


Abbildung 6: Anwahl zur Editierung von Abläufen

Aus der Anwahlmaske heraus gibt es aber noch eine Vielzahl von Funktionen, die sich auf die gesamte Ablaufprogrammierung beziehen. Hinter dem Softkey Hilfsmenue verbirgt sich ein weiteres Menü, mit dem das Inhaltsverzeichnis der Ablaufprogrammierung angezeigt werden kann und mit dem alle Abläufe gedruckt werden können.

Abbildung 7: Hilfsmenü in der Anwahl der Ablaufprogrammierung

Der Softkey *Alle drucken* führt zu einem weiteren Untermenü, in dem die Druckfunktionen ausgewählt werden können. *In Datei* druckt dabei alle Abläufe in Textdateien, wobei für jeden Ablauf eine neue Textdatei angelegt wird, die den Namen des Ablaufs und die Dateiendung TXT trägt. Die Funktion *In Eine* druckt alle Abläufe in eine einzige Textdatei, wobei die Abläufe in dieser Datei numerisch sortiert sind.

Der Softkey *Inhalt* baut ein Inhaltsverzeichnis der Ablaufprogrammierung auf – ähnlich dem Bild in der Verwaltung der Abläufe. In diesem Inhaltsverzeichnis kann die Bezeichnung eines Ablaufs verändert werden. Außerdem kann der Ablauf, auf den aktuell der Cursor steht, gedruckt werden. Weitere Funktionen sind im Inhaltsverzeichnis an dieser Stelle nicht möglich.

Abbildung 8: Beispiel zum Inhaltsverzeichnis der Ablaufprogrammierung

Ebenfalls aus der Hauptmaske der Anwahl heraus steht die Funktion *Suchen* zur Verfügung. Diese Funktion ermöglicht es, über alle Abläufe hinweg zu suchen. Gesucht werden kann nach Eingängen, Ausgängen, Sprungzielen, Ablaufnummer, Befehlen, Meldungen, Merkern und externen Daten. Wonach gesucht werden soll, wird anhand des entsprechenden Softkeys festgelegt – anschließend wird der Wert, nach dem gesucht werden soll eingegeben werden. Softkey 2 „Ausgang“ und die anschließende Eingabe der Zahl 1 sucht also nach allen Befehlen, die den Ausgang 1 ansprechen – und zwar in allen Abläufen.

Das Suchergebnis ist ein Bild, in dem die entsprechenden Fundstellen der Suche aufgeführt werden:

Abbildung 9: Beispiel für die Suche über alle Abläufe

Das Bild listet alle Fundstellen, wobei neben der Ablaufnummer auch die entsprechende Zeile, die Ablaufbezeichnung und der Befehl mit dem (in diesem Fall) der Ausgang angesprochen wird, genannt werden. Diese Liste kann also als Referenzliste für die entsprechende Suche verwendet werden. Die Liste kann über die Funktion *Drucken* in eine Textdatei auf der Diskette ausgegebenen werden.

Mit der Funktion *Übernahme* ist es möglich, direkt an die entsprechende Stelle zu springen. Dabei öffnet die Software den Ablauf, auf den der Cursor aktuell steht zur Editierung und springt dann auf die Zeile, auf der der Cursor innerhalb des Suchergebnisses steht. Im oben gezeigten Bild würde also der Ablauf 229 zur Editierung geöffnet werden und innerhalb der Editierung würde die Zeile 2 als den Cursor erhalten.

Editierung eines einzelnen Ablaufs

In der Maske zur Editierung eines einzelnen Ablaufs werden die Zeilen des Ablaufs untereinander dargestellt. Aufgeführt sind dabei die Zeilennummer, der Befehl, die Kennzeichnung auf eine Indizierung (dargestellt durch die Spalte I.), der Operand und die Beschreibung. Ein Sternchen vor der Zeilennummer signalisiert, dass innerhalb dieses Ablaufs die Zeile als Sprungziel eingetragen ist. Die Beschreibung ist grundsätzlich zweizeilig, wobei die obere Zeile die textmässige Übersetzung des Befehls darstellt. In der unteren Zeile werden – soweit vorhanden – die Texte für die Operanden angezeigt. So steht bei einem Ausgangsbefehl an dieser Stelle die Ausgangsbezeichnung für den Ausgang, der im Operanden angegeben ist.

*	NR.	BEF.	I.	OPERAND	BESCHREIBUNG
*	1	51	0	0	WARTE BIS AUFTRAG <> 0
	2	58	0	1	TESTE AUFTRAG = OPERAND ETAGENBAND 1: TEILEÜBERNAHME VON LIFT 1
	3	71	0	30	SPRUNG WENN JA ****** AUFTRAG 1 ******
	4	58	0	2	TESTE AUFTRAG = OPERAND ETAGENBAND 1: TEILEÜBERGABE AN LIFT 2
	5	71	0	71	SPRUNG WENN JA ****** AUFTRAG 2 ******
	6	58	0	3	TESTE AUFTRAG = OPERAND RESET DER MERKER
	7	71	0	109	SPRUNG WENN JA ****** AUFTRAG 3 ******
	8	72	0	114	SPRUNG UNBEDINGT ****** AUFTRAG > X ******

Abbildung 10: Beispiel Editierung eines einzelnen Ablaufs

Zeilen einfügen und Zeilen löschen

Die Funktionen zum einfügen und löschen einer Zeile befinden sich in der zweiten Softkeyleiste. Gelöscht wird immer die Zeile auf der der Cursor gerade steht. Beim Einfügen wird immer in der Zeile eingefügt, auf der der Cursor gerade steht, d.h. diese Zeile wird nach unten verschoben. Wird im Feld Operand in der letzten Zeile die Taste *ENTER* benutzt, so wird automatisch eine weitere Zeile angehängt.
Die Editierung von Abläufen verfügt über eine automatische Sprungverschiebung. Wird eine Zeile eingefügt oder gelöscht, so werden alle Sprünge innerhalb des Ablaufs automatisch so korrigiert, dass der Sprung immer noch auf dem gleichen Befehl endet. Damit bleiben die Sprünge korrekt erhalten, geändert wird jeweils die als Operand eingetragene Sprungzeile und zwar für jeden Befehl, für den die Änderung notwendig ist.

Die eigentliche Editierung

Die eigentliche Editierung eines Ablaufs verläuft feldweise. Dabei wird jede Eingabe durch *Enter* abgeschlossen und der Cursor springt automatisch in das nächste Feld. Wird eine Eingabe mit *Zurück* abgebrochen, so wird die letzte Eingabe nicht übernommen und der Editiermodus wird beendet. Eine Ausnahme stellt das Feld für die Indizierung dar. Da dieses Feld nur sehr selten benötigt wird, wird es im normalen Editierungsmodus übersprungen, d.h. der Cursor wechselt aus dem Feld Befehl nicht in das Feld Editierung sondern in das Feld Operand. Um das Feld Indizierung bearbeiten zu können, muss im Feld Operand die *Links*-Taste verwendet werden.

Eine Eingabe im Feld Zeilennummer führt dazu, dass auf die angegebene Zeile gesprungen wird, um dort die Editierung fort zu setzen. Wird eine Zeilennummer angegeben, die größer als die Anzahl der Zeilen des Ablaufs ist, so springt die Editierung auf die letzte Zeile.

Im Feld Befehl wird der auszuführende Befehl eingegeben. Möglich sind Werte zwischen 0 und 255.

Im Feld Indizierung wird ausschließlich angegeben, ob der Befehl indiziert ist oder nicht. Möglich sind die Werte 0 (nicht indiziert) und 1 (indiziert).

Im Feld Operand wird der gewünschte Operand angegeben. Der mögliche Zahlenbereich liegt zwischen 0 und 32767.

Die Hilfsfunktion Liste

Der Softkey Liste führt zu einer Hilfsfunktion, die abhängig von dem eingegebenen Befehl ist. Es wird dabei jeweils eine Liste der möglichen Operanden angezeigt. So wird bei allen Ausgangsbefehlen eine Liste aller Ausgänge angeboten, für die ein Text besteht; bei allen Eingangsbefehlen wird eine Liste aller Eingänge angeboten, für die ein Text besteht; bei allen Befehlen für ablaufabhängige Merker wird eine Liste aller ablaufabhängigen Merker angeboten, für die ein Text besteht; usw.

Die entsprechenden Texte können in dieser Liste auch verändert und ausgedruckt werden. Da es sich um das gleiche Prinzip wie bei der Eingabe von Texten handelt, sei an dieser Stelle auf das Kapitel Betextung von Ein- und Ausgängen verwiesen.

Außer bei ablaufabhängigen Merkern, Jobs und Steps springt die Liste beim Öffnen automatisch auf den Wert, der aktuell im Operanden eingetragen ist. Ist der entsprechende Text nicht vorhanden, wird er automatisch angelegt und der Cursor steht auf dem Feld Bezeichnung, so dass der Text eingegeben werden kann. In der zweiten Softkeyleiste des Listen-Bilds befindet sich der Softkey Übernahme. Dieser Softkey führt dazu, dass die Nummer, auf der der Cursor der Liste aktuell steht als Operand in die Ablaufprogrammierung übernommen und die Listenanzeige beendet wird.

Suchen innerhalb eines Ablaufs

Die Funktionen zum Suchen innerhalb eines Ablaufs befinden sich im Bereich Funktionen und sind durch den gleichnamigen Softkey erreichbar. Gesucht werden kann nach Eingängen, Ausgängen, Sprungzielen, Ablaufnummer, Befehlen, Meldungen, Merkern und externen Daten. Was gesucht werden soll wird durch den entsprechenden Softkey ausgewählt und anschließend wird der gewünschte Werte eingegeben. Die Suche beginnt immer in der ersten Zeile des Ablaufs und der Cursor springt automatisch auf die erste Fundstelle. Anschließend steht der Softkey Wiederholen zur Verfügung. Dieser Softkey setzt die Suche an der Stelle fort, an der der Cursor gerade steht, sucht also innerhalb des Ablaufs nach dem nächsten Eintrag. Werden keine weiteren passenden Einträge gefunden, so erscheint kurz die Meldung „keine weiteren Einträge“.

Ersetzen innerhalb eines Ablaufs

Die Funktionen zum Ersetzen innerhalb eines Ablaufs befinden sich im Bereich Funktionen und sind durch den gleichnamigen Softkey erreichbar. Ersetzt werden können Eingängen, Ausgängen, Sprungzielen, Ablaufnummer, Befehlen, Meldungen, Merkern und externen Daten. Was ersetzt werden soll wird durch den entsprechenden Softkey ausgewählt und anschließend wird der alte Wert und dann der neue Wert eingegeben. Das Ersetzen beginnt immer in der ersten Zeile des Ablaufs und der Cursor springt automatisch auf die erste Fundstelle. Hier stehen dann die Softkeys Nein, Ja, Alle zur Verfügung. Der Softkey Nein führt die Ersetzung an dieser Stelle nicht aus, der Softkey Ja führt die Ersetzung an dieser Stelle durch und der Softkey ersetzt diese und alle weiteren Stellen innerhalb des Ablaufs ohne weitere Nachfrage. Wurde nicht der Softkey Alle benutzt, steht anschließend steht der Softkey Wiederholen zur Verfügung. Dieser Softkey setzt die Ersetzung an der Stelle fort, an der der Cursor gerade steht, sucht also innerhalb des Ablaufs nach dem nächsten Eintrag. Werden keine weiteren passenden Einträge gefunden, so erscheint kurz die Meldung „keine weiteren Einträge“.

Drucken eines einzelnen Ablaufs

Mit dem Softkey Drucken kann ein einzelner Ablauf als Textdatei auf Diskette ausgegeben werden. Der entsprechende Softkey befindet sich im Hauptmenü der Editierung. Die Textdatei, die anschließend den „Ausdruck“ beinhaltet trägt den Namen der Ablaufdatei und die Dateiendung TXT, also beispielsweise PROZ0001.TXT für den Ablauf 1.

Betextungen

Innerhalb der Ablaufprogrammierung können die verschiedensten Element betextet werden. Dabei werden alle Texte in der zentralen Textdatei gespeichert.

Überblick der möglichen Betextungen

Die nachfolgende Tabelle beschreibt, welche Elemente betextet werden können, wie der entsprechende Textschlüssel für die Textdatei aussieht und nach welchem Verfahren die Betextung erfolgt.

Element	Schlüssel	Beispiel	Betextungsverfahren
Name eines Ablaufs	`SPSNAME...+Nummer`	`SPSNAME.....7`	Siehe Kapitel Verwaltung der SPS
Eingänge	`SPSEIN....+Nummer`	`SPSEIN......3`	Siehe Kapitel Betextung von Ein- und Ausgängen
Ausgänge	`SPSAUS....+Nummer`	`SPSAUS......3`	Siehe Kapitel Betextung von Ein- und Ausgängen
Meldungen	`SPSMELD....+Nummer`	`SPSMELD.....5`	Gleiches Verfahren wie bei den Eingängen
Kommentare	`SPSKOMM....+Nummer`	`SPSKOMM.....6`	Gleiches Verfahren wie bei den Eingängen
Globale Merker	`SPSGMERK...+Nummer`	`SPSGMERK.1024`	Gleiches Verfahren wie bei den Eingängen
Bezeichnung der externen Daten	`WCBTEXT...+Nummer`	`WCBTEXT.....5`	Gleiches Verfahren wie bei den Eingängen
Ablaufabh. Merker	`SPSM + Ablaufnummer + ....+Nummer`	`SPSM0004....1`	Siehe Kapitel Betextung von ablaufabhängigen Merkern
Jobs der Abläufe	`SPSJ + Ablaufnummer + ....+Nummer`	`SPSJ0005....3`	Gleiches Verfahren wie bei ablaufabh. Merkern
Steps der Abläufe	`SPSS + Ablaufnummer + ....+Nummer`	`SPSS1023....1`	Gleiches Verfahren wie bei ablaufabh. Merkern
SPS-Sicherheitsräume	`SPSSICH + ... + Nummer`	`SPSSICH.....4`	Innerhalb des Editors für die SPS-Sicherheitsräume

Betextung von ablaufabhängigen Merken

Die Texte für ablaufabhängige Merker werden in der zentralen Textdatei gespeichert. Der entsprechende Schlüssel ergibt sich aus der Kennung SPSM plus der Ablaufnummer mit führenden Nullen plus der Merkernummer ohne führende Nullen, also beispielsweise `SPSM0004....1` für den Merker 1 im Ablauf 4.

Abbildung 11: Betextung ablaufabhängiger Merker

Diese Maske ermöglicht die Editierung der Texte für ablaufabhängige. Angezeigt werden die Ablaufssnummer, die Merkernummer und der dazugehörige Text. Um eine neue Nummer anzulegen, muss im Feld „Abl.“ die gewünschte Ablaufnummer und im Feld „Nr.“ die gewünschte Merkernummer eingegeben werden. Um einen Text zu löschen, wird der Softkey „Satz Entf.“ Verwendet, der sich in der zweiten Softkeyleiste befindet, die über den Softkey „<<“ erreichbar ist. Die gesamten Bezeichnungen der ablaufabhängigen Merker können über den Softkey „Drucken“ in eine Textdatei auf der Diskette ausgegeben werden. Auch ein Export in eine Datei, die mit Excel lesbar ist, ist über die Funktion „Drucken“ erreichbar.

Nach dem gleichen Prinzip und mit dem gleichen Editor werden auch die ablaufabhängigen Job- und Stepbezeichnungen verarbeitet.

Funktion „Alle Texte drucken“

In allen Softwareständen **ab dem 09.12.2003** befindet sich eine Funktion „Alle Texte drucken“. Mit dieser Funktion werden alle vorhandenen Texttyp (Eingangs- und Ausgangsbezeichnungen, Jobs, Steps, Merker, Achs- und Ablaufbezeichnungen usw.) in Textdateien auf Diskette gedruckt. Die Funktion arbeitet dabei so, also ob die einzelnen Textteile einzelnd gedruckt werden, d.h. jeder Texttyp wird in eine eigene Textdatei auf der Diskette gedruckt. Beim Aufruf der Funktion muss einmalig die verwendete Diskette bestätigt werden.

Onlinehilfe der Opcodes

Die Ablaufprogrammierung verfügt über ein Bild, in dem alle verfügbaren Opcodes mit ihrer Betextung aufgeführt sind. Dieses Bild kann als Onlinehilfe verwendet werden. Die entsprechende Opcodetabelle kann mit dem Softkey Drucken in eine Textdatei oder in eine Excel-lesbare Datei ausgegeben werden.

Abbildung 12: Onlinehilfe zu den SPS-Befehlen

Testen der Ablaufprogrammierung

Zum Testen der Ablaufprogrammierung gibt es verschiedene Möglichkeiten. Zum einen kann die Funktionalität eines einzelnen Ablaufs getestet werden und zum anderen kann das Zusammenspiel mehrerer Abläufe untersucht werden.

Verbundtest

Der Verbundtest bietet die Möglichkeit, das Zusammenspiel verschiedener Abläufe zu untersuchen. In dem entsprechenden Bild werden die internen Daten der Ablaufprogrammierung für mehrere Abläufe untereinander dargestellt.

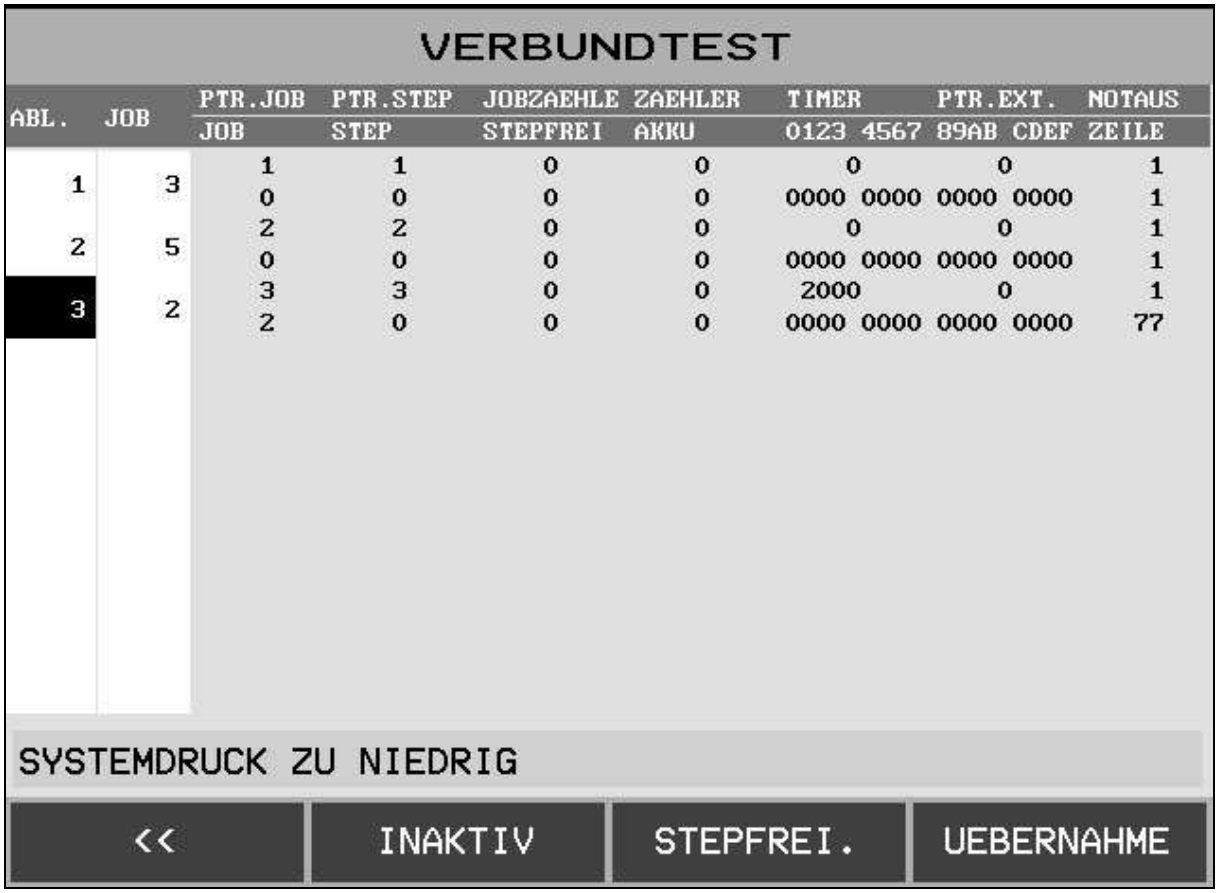


Abbildung 13: Verbundtest

Neben den Daten der angewählten Abläufe wird in dem entsprechenden Bild auch die aktuelle Bedienerführung angezeigt (im obigen Beispiel der Text „Systemdruck zu niedrig“).

Um einen weiteren Ablauf in der Beobachtung hinzu zu fügen, wird der Softkey Satz einf. in der zweiten Softkeyleiste benutzt. Hier kann auch der Ablauf, auf den aktuell der Cursor steht, aus der Beobachtung entfernt werden. Es können beliebig viele Abläufe in die Beobachtungsliste aufgenommen werden, jeder weitere Ablauf erhöht aber die Zeit, die für den Refresh der Anzeige benötigt wird.

In jeder Zeile sind zwei Felder editierbar. Im ersten Feld wird angegeben, welcher Ablauf an dieser Stelle angezeigt werden soll. Im zweiten Feld kann ein Auftrag vorgegeben werden. Dieser Auftrag wird in dem entsprechenden Ablauf eingetragen, wenn der Softkey Übernahme gedrückt wird.

Der zyklische Refresh der Daten kann aktiviert und deaktiviert werden. Standardmäßig ist er aktiviert. Bei aktivierten Refresh wird der Softkey Inaktiv angezeigt, mit dem der Refresh deaktiviert werden kann. Bei deaktiviertem Refresh wird der Softkey Aktiv angezeigt, mit dem der Refresh aktiviert werden kann.

Mit dem Softkey Stepfreigabe kann für den Ablauf, auf dem aktuell der Cursor steht, eine Stepfreigabe gesetzt werden. Die Stepfreigabe wird sofort gesetzt, nach dem der entsprechende Wert mit Enter bestätigt worden ist.

Der Softkey Übernahme dient dazu, den Abläufen die eingetragenen Aufträge zu erteilen. In dem Moment, in dem der Softkey gedrückt wird, erhalten alle angezeigten Abläufe den Auftrag, der im entsprechenden Feld Job eingetragen ist.

Einzeltest eines Ablaufs

Jeder Ablauf kann einem Einzeltest unterzogen werden. Die entsprechende Funktion erreicht man, wenn in der Maske zur Editierung eines Ablaufs der Softkey Test gedrückt wird.

Im Einzeltest eines Ablaufs werden neben den einzelnen Programmzeilen die internen Daten des zu testenden Ablaufs eingeblendet und ständig neu angezeigt:

PTR.JOB	PTR.STEP	JOBZAEHLE	ZAEHLER	TIMER	PTR.EXT.	NOTAUS
1	1	0	0	0	0	0
JOB	STEP	STEPFREI	AKKU	0123 4567	89AB CDEF	S(MERKER)
0	0	0	0	0000 0000	0000 0000	-

Abbildung 14: Interne Daten des Ablaufs im Einzeltest

Neben den zu dem Ablauf gehörenden Merkern wird unter dem Buchstaben S der Zustand der Sprungbedingung angezeigt. Ein X an dieser Stelle bedeutet, dass die Sprungbedingung auf Ja steht, ein Minus bedeutet, dass die Sprungbedingung auf Nein steht.

In den Programmzeilen selber wird im Einzeltest der Zustand von Eingängen, Ausgängen, globalen und ablaufabhängigen Merkern angezeigt:

21	24	0	208	.	AUSGANG LOESCHEN Gruppe 0: Förderband 1 Ein

Abbildung 15: Anzeige des Ausgangsstatus im Einzeltest

Hinter dem Operanden wird bei den entsprechenden Daten der Zustand Low durch einen Punkt, der Zustand High durch ein Ausrufezeichen signalisiert. Im obigen Beispiel steht der Ausgang 208 aktuell also auf low.

Damit der Test durchgeführt werden kann, muss dem Ablauf in der Regel ein Auftrag erteilt werden. Dies geschieht durch den Softkey Job und die Eingabe des gewünschten Auftrags. Zusätzlich können dem Ablauf eine Stepfreigabe (Softkey Stepfrei) und ein Step (Softkey Step) gegeben werden. Alle drei Softkeys befinden sich in der zweiten Softkeyleiste.

Durch das Eintragen eines Auftrags beginnt der Ablauf im Einzeltest jedoch noch nicht zu arbeiten. Hierfür muss entweder der Softkey Einzel oder die Start-Taste verwendet werden. Der Softkey Einzel führt nur den einen Befehl aus, auf dem der Cursor aktuell steht und kann damit verwendet werden, um einen Ablauf schrittweise zu testen.

Die Starttaste führt dazu, dass der Ablauf zu laufen beginnt. Ein laufender Ablauf kann nur durch die Stoptaste angehalten werden. Während der Ablauf läuft, wird die aktuelle Zeile ständig nachgeführt und der Cursor steht auf inaktiv:

Abbildung 16: Inaktiver Cursor bei laufendem Ablauf im Einzeltest

Auch beim Softkey Einzel kann es vorkommen, dass der Cursor inaktiv ist. Dies geschieht immer dann, wenn ein Befehl noch nicht zu Ende abgearbeitet worden ist (z.B. bei einem Warte-Befehl). Während der Cursor inaktiv ist, ist keine Bedienung des Bildes möglich.

Eine besondere Bedeutung kommt dem Softkey Breakpoint zu. Mit diesem Softkey ist es möglich, einen Breakpoint innerhalb des Ablaufs zu setzen. Hierfür muss die Zeile eingegeben werden, auf der der Breakpoint sitzen soll. Der Ablauf läuft dann solange, bis die Zeile mit dem Breakpoint erreicht ist. Wird diese Zeile erreicht, wird die Ausführung des Ablaufs beendet und der Cursor erscheint wieder im Bild, so dass von dieser Stelle aus beispielsweise im Einzelschritt weiter getestet werden kann. Ein Breakpoint wird gelöscht, in dem als Zeilennummer eine Null eingegeben wird. Beim Verlassen des Tests wird ein eingetragener Breakpoint ebenfalls gelöscht.

Testen von globalen Merkern

Ähnlich der Übersicht über die Ein- und Ausgänge verfügt das System über ein Bild, in dem der Zustand der globalen Merker angezeigt werden kann.

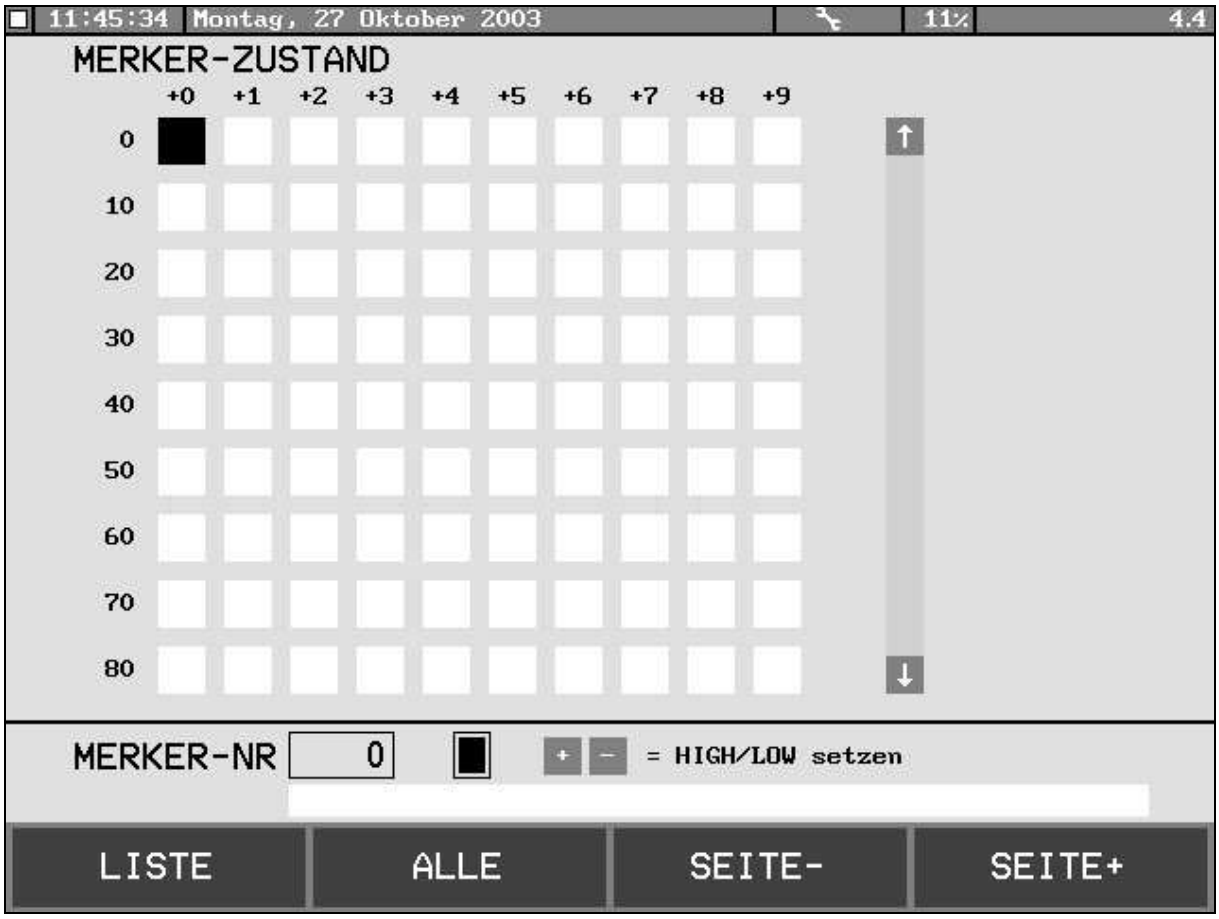


Abbildung 17: Übersicht der globalen Merker

In diesem Bild werden pro Seite 80 globale Merker angezeigt, ist das Kästchen schwarz, so ist der entsprechende Merker high, ansonsten low. Mit den Softkeys *Seite+* und *Seite-* kann das Bild verschoben werden, so dass auch höhere Merkernummern sichtbar sind. Die Funktion *Liste* ruft eine Liste auf, in der die Texte der globalen Merker angezeigt, verändert und gedruckt werden können. Um einen Merker manuell zu verändern, wird die *Enter*-Taste benutzt, um in den Editermodus zu gelangen, anschließend wird die gewünschte Merker-Nummer eingegeben. Sie steht dann in dem Feld Merker-Nummer im unteren Bildteil. Die Taste *Plus* setzt den hier angezeigten Merker auf High, die Taste *Minus* auf Low.

Mit dem Softkey Alle gelangt man in ein Übersichtsbild, in dem alle im System verfügbaren 1024 globalen Merker angezeigt werden:

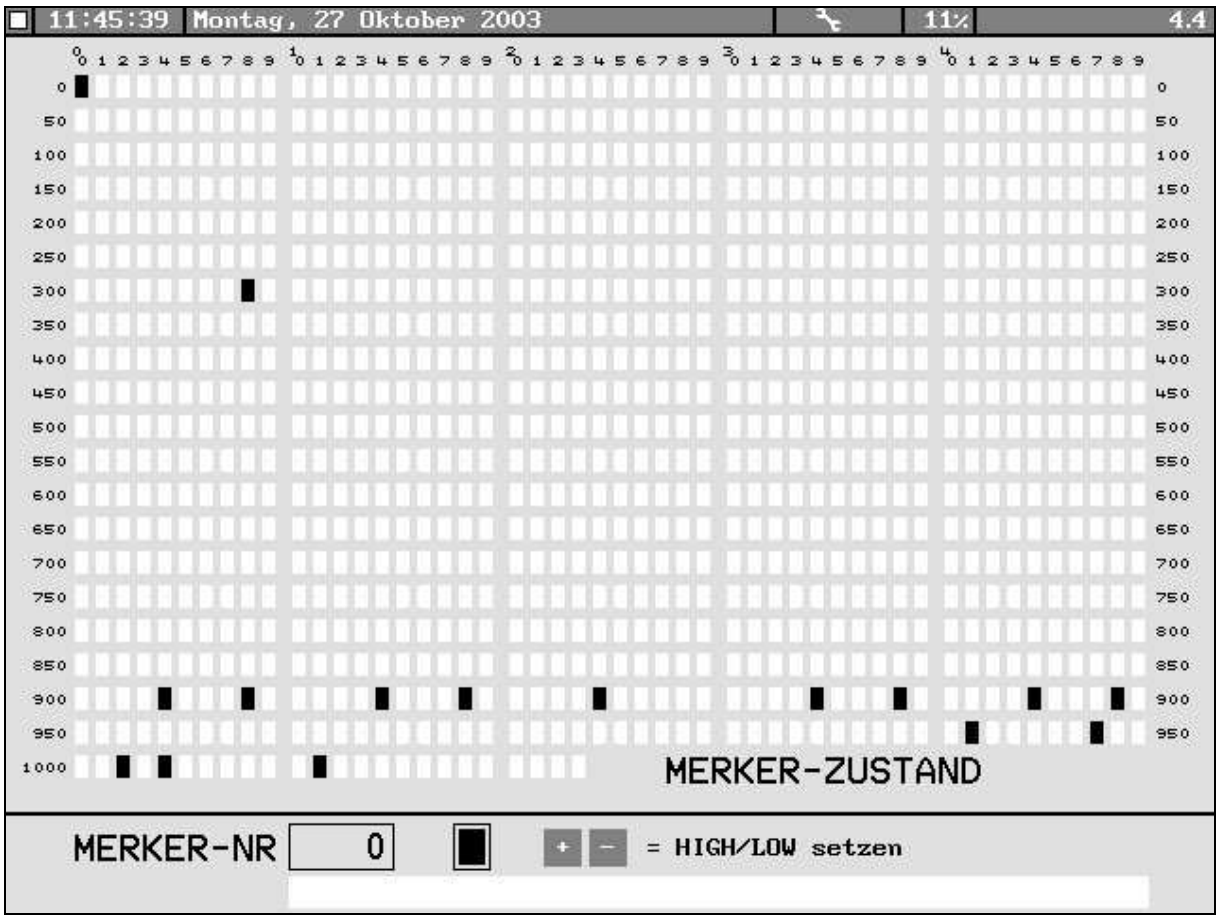


Abbildung 18: Übersicht aller globalen Merker

In dieser Komplettübersicht stehen keine Softkeys zur Verfügung, es kann aber ein Merker nach dem oben beschriebenen Verfahren gesetzt oder gelöscht werden. Um in das vorherige Bild zu gelangen wird die Taste *Zurück* verwendet.

Zusatzdaten der Ablaufprogrammierung

In diesem Kapitel geht es darum, welche zusätzlichen Möglichkeiten die Ablaufprogrammierung bietet. In erster Linie geht es dabei um Daten, die innerhalb der Ablaufprogrammierung verwendet werden können oder die mit der Ablaufprogrammierung verwandt sind.

Tabelle der externen Zeiten

Bei der Tabelle der externen Zeiten handelt es sich um einen Datentabelle (Dateiname PROZESS.ZEI), die es ermöglicht, Zeitangaben editierbar zu machen, ohne dass der entsprechende Ablauf dafür geändert werden muss. Dies ist beispielsweise immer dann sinnvoll, wenn Zeitangaben vom Maschinenführer veränderbar sein sollen. Auf die Tabelle der externen Zeiten kann mittels Opcode aus dem Ablauf heraus zugegriffen werden.
Bei der Tabelle handelt es sich um eine einfache Rekorddatei, die beliebig groß angelegt werden kann. Jeder Eintrag besteht aus einer Zeitangabe *in Sekunden* und kann drei Vorkomma und zwei Nachkommastellen haben. Die entsprechenden Zeitangaben können betextet werden, wobei der entsprechende Text in der zentralen Textdatei gespeichert wird. Als Textschlüssel wird `SPSZEIT.....+Nummer des Rekords in der Datei` verwendet, also beispielsweise `SPSZEIT.....1` für die erste Zeitangabe innerhalb der Datei.

Die nachfolgende Abbildung zeigt ein Beispiel für eine entsprechende Tabelle mit externen Zeiten:

ABLAUFZEITEN

ABLAUFZEIT	ZEIT IN SEKUNDEN
MELDEZEILE SEC.	2.00
MAX.ZYKLUSZEIT SGM	90.00
VORLAUFZEIT ETAGENBAND LEEREN	20.00
NACHLAUFZEIT LIFT 1 AUF ETAGENBAND	0.50
NACHLAUFZEIT ETAGENBAND ENTLEEREN	12.00
LEERFAHRZEIT	60.00
NACHLAUFZEIT TRAY AUSLAUF	10.00
LEERFAHRZEIT TRAY EINLAUF	30.00
LEERFAHRZEIT TRAY AUSLAUF	30.00
NACHLAUFZEIT TRAY EINLAUF	10.00

<< | | SEITE- | SEITE+

Abbildung 19: Beispiel für externe Zeiten

In der Tabelle kann mit den Softkeys Satz entfernen ein Datensatz gelöscht werden und mit dem Softkey Satz einfügen ein Datensatz angelegt werden. Die beiden Softkeys befinden sich in der zweiten Softkeyleiste. Die Betextung der Zeit kann direkt in der Tabelle erfolgen.

In Systemen ab 10.03.2006 gibt es zusätzlich eine Tabelle mit SPS-Systemzeiten. Diese Tabelle hat die gleiche Struktur wie die Standard-Zeitentabelle und bildet Ihren Textschlüssel aus **`SYSZEIT+Nummer`**. Der Dateiname lautet SYSTEM.ZEI. Der SPS-Befehl 46 greift auf diese Tabelle zu, der SPS-Befehl 47 auf die o.g. Standardtabelle.

Benutzerdefinierte Softkeys

Die Ablaufprogrammierung gibt die Möglichkeit, aus einem Ablauf heraus, benutzerdefinierte Softkeys anzuzeigen. Die entsprechende Anzeige erfolgt nur im Automatikbild. Die entsprechenden Befehle innerhalb der Ablaufprogrammierung geben dabei an, welcher Softkey aus der Softkeydatei angezeigt werden soll. Die benutzerdefinierten Softkeys werden auch als Automatik-Softkeys bezeichnet, da sie nur in im Automatikbild verwendet werden können.

Jeder Softkey wird dabei innerhalb der Softkeydatei durch verschiedene Parameter definiert, die in der nachfolgenden Tabelle näher erläutert sind:

Parameter	Bedeutung
Lfd.Nr.	Laufende Nummer des Softkeys. Mit dieser Nummer wird der entsprechende Softkey aus der Ablaufprogrammierung heraus angesprochen.
Text	Text, der angezeigt werden soll. Die Texte werden in der zentralen Textdatei gespeichert und haben als Schlüssel `USERSOFT.....+Softkeynummer`, also beispielsweise `USERSOFT....3` für den Softkey mit der laufenden Nummer 3.
Softkey	Auf welcher Position innerhalb der Softkeyleiste soll der benutzerdefinierte Softkey angezeigt werden? Softkey 1 = links, Softkey 4 = rechts. Möglich sind nur Werte zwischen 1 und 4.
Ablauf	An welchen Ablauf soll ein Auftrag gesendet werden, wenn dieser Softkey gedrückt wird?
Auftrag	Welcher Auftrag soll an den Ablauf gesendet werden, wenn dieser Softkey gedrückt wird?

Anhand des nachfolgenden Beispiels soll die Funktionsweise näher erläutert werden. Die Tabelle zeigt dabei, wie der entsprechende Softkey definiert worden ist:

Lfd.Nr	Text	Softkey	Ablauf	Auftrag
3	JA	2	991	27

Wenn die Ablaufprogrammierung einen Befehl ausführt, der den Softkey 3 anzeigen soll, dann wird im Automatikbild auf dem zweiten Softkey von links der Text Ja angezeigt. Wird dieser Softkey dann innerhalb des Automatikbildes gedrückt, so erhält der Ablauf 991 einen Auftrag 27.

Zur Verwaltung der benutzerdefinierten Softkeys stellt die Steuerung eine entsprechende Eingabemaske zur Verfügung. In dieser Eingabemaske können Datensätze eingefügt und gelöscht werden. Die Betextung der benutzerdefinierten Softkeys erfolgt ebenfalls in dieser Eingabemaske, die Texte selber werden aber in der zentralen Textdatei gespeichert.

Die nachfolgende Abbildung zeigt die Eingabemaske für die benutzerdefinierten Softkeys:

AUTOMATIK - SOFTKEYS

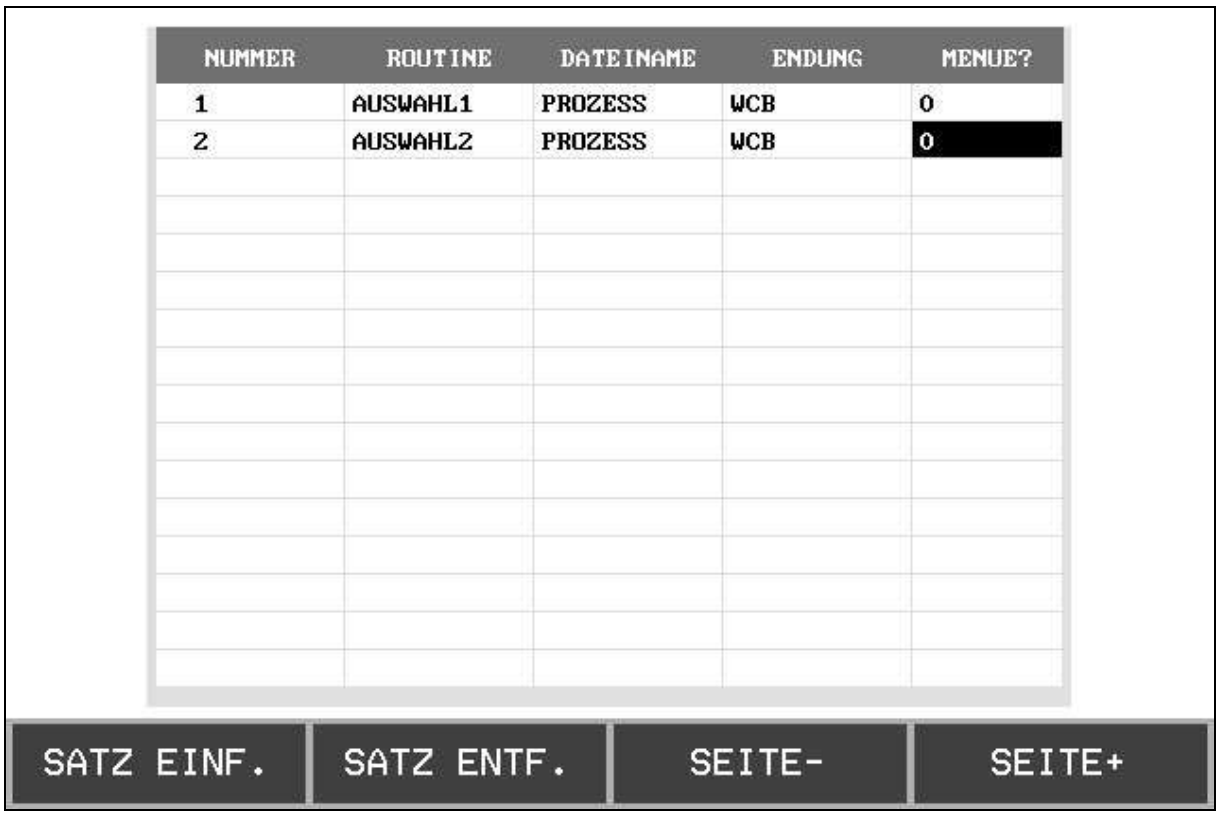
LFD.NR.	TEXT	SOFTKEY	ABLAUF	AUFTRAG
1	JA	2	991	2
2	NEIN	3	991	3
3	LEERFAHREN	2	991	2
4	USERSOFT....4	0	0	0

SATZ EINF. | SATZ ENTF. | SEITE- | SEITE+

Abbildung 20: Editierung von Automatik-Softkeys

Aufrufbare Editoren

Die Ablaufprogrammierung bietet die Möglichkeit, mittels Befehl einen Editor innerhalb der Automatikmaske zu öffnen und somit dem Bediener einer Eingabemöglichkeit zu geben. Der Befehl in der Ablaufprogrammierung gibt an, welcher Editor aus der Liste der zur Verfügung stehenden Editoren geöffnet werden soll. Die Liste der Editoren kann individuell angepasst werden und hat prinzipiell folgenden Aufbau:



NUMMER	ROUTINE	DATEINAME	ENDUNG	MENUE?
1	AUSWAHL1	PROZESS	WCB	0
2	AUSWAHL2	PROZESS	WCB	0

Abbildung 21: Beispiel aufrufbarer Editoren für die Automatikmaske

Die Nummer ist die Datensatznummer, die innerhalb der Ablaufprogrammierung angegeben wird, um einen Editor zu öffnen. Das Feld *Routine* gibt an, welcher Editor geöffnet werden soll. Dabei ist sicherzustellen, dass der entsprechende Editor (Dateien: Routine.edf, Routine.edh, Routine.edl – wobei Routine für den eingetragenen Namen steht) sich auch auf der Ramdisk des Systems befindet. Im Feld *Dateiname* wird angegeben, wie die Datendatei heißt, die mit diesem Editor bearbeitet werden soll. Im Feld *Endung* wird festgelegt, welche Dateiendung die Datendatei hat, die mit diesem Editor bearbeitet werden soll.
Steht im Feld *Menue?* eine Null, so handelt es sich um einen Editor zur Datenediterung. Steht in dem Feld ein Wert ungleich Null, so handelt es sich um ein Untermenü, welches geöffnet werden soll. In dem Fall gibt der Routinennamen den Namen der zu öffnenden Menüdatei an.

Die Funktionen zum Öffnen eines Editors funktionieren nur in der Automatikmaske des Systems! Wird der Editor oder das Menü beendet, so setzt die Ablaufprogrammierung in dem Ablauf, der den Aufruf verursacht hat, den ablaufabhängigen Merker Null.

Überwachungen

Die Tabelle der Überwachungen gibt die Möglichkeit, Abläufe mit bis zu vier Eingängen zu verknüpfen und ist nur in Betriebssystemen der OALSYS-Reihe implementiert. Die Überwachungen finden nur statt, wenn der Ablauf keinen Auftrag hat und somit keine Rechenzeit erhält. Die Überwachungen geben eine komfortable Möglichkeit, eine ständige Überprüfung eines Eingangs innerhalb eines Ablaufs zu verhindern und somit die Systemlast zu reduzieren. Die Überwachung trägt die Platznummer des Eingangs, der High ist, als Step in den Ablauf ein. Ist keiner der definierten Eingänge high, kommt es genauso zu einer Störung, als wenn mehr als ein Eingang high ist.

Wenn in der Liste der Überwachungen ein Ablauf mit Eingängen verbunden ist und dieser Ablauf im System nicht existiert, so kann dies zu Störungsmeldungen von nicht existierenden Abläufen führen!

Prinzipiell kann man die Arbeitsweise der Überwachung am nachstehenden Beispiel erklären:

AKTUELLE DATEI.....: TEACHO2.UBW

ABLAUF	EINGANG 1	EINGANG 2	EINGANG 3	EINGANG 4
5	57	58	0	0
27	3	4	0	0

Abbildung 22: Beispiel für Überwachungen

Der Ablauf 5 ist mit den Eingängen 57 und 58 verbunden. Wenn der Ablauf 5 keinen Auftrag hat, werden die Überwachungen durchgeführt und das System setzt für den Ablauf 5 den Step 1, wenn der Eingang 57 high ist oder den Step 2, wenn der Eingang 58 high ist. Welcher Step gesetzt wird, hängt von der Platzierung der Eingänge innerhalb der Tabelle ab. Die Eingänge in der Spalte *Eingang 1* setzen den Step 1, die Eingänge in der Spalte *Eingang 2* setzen den Step 2 usw. Sind in diesem Fall sowohl der Eingang 57 als auch der Eingang 58 high, kommt es zur Störung „Eingang high, Ablauf 5, Eingang 58“. Sind beide Eingänge low, kommt es zur Störung „Eingang low, Ablauf 5, Eingang 58“.
Eine Eingangsnummer Null steht für nicht belegt – damit kann auch kein Ablauf mit dem Eingang Null verbunden werden.

SPS-Sicherheitsräume

Ein Sicherheitsraum stellt einen Block von maximal vier Bewegungen dar, mit dem eine Position im Raum beschrieben werden kann. Bei den Bewegungen kann es sich um eine Achse oder um einen Ablauf handeln. Für jede einzelne Bewegung wird festgelegt, von welchem Mass bis zu welchem Mass die Bedingung gelten soll. Handelt es sich bei der Bewegung um einen Ablauf, so gilt das Mass als der Step, den der entsprechende Ablauf aktuell meldet. Ein Sicherheitsraum gilt als verletzt, wenn alle Bewegungen die beschriebene Bedingung erfüllen. Erfüllt nur eine der genannten Bewegungen die Bedingung nicht, so ist der Sicherheitsraum nicht verletzt. Melden Abläufe einen Step Null, so gilt die entsprechende Bedingung als erfüllt.

Die SPS-Sicherheitsräume werden vom System nicht automatisch getestet, sondern müssen innerhalb der Ablaufprogrammierung mit dem Opcode 254 abgefragt werden.

Die Datei mit den Sicherheitsräumen ist eine rekordbasierte Datei. Wird ein Datensatz eingefügt oder gelöscht, so verschieben sich alle dahinter liegenden Sicherheitsräume und es ist eventuell notwendig, die abfragenden Prozesse entsprechend zu ändern!

Sicherheitsräume können mit einem Text versehen werden. Der Textschlüssel für SPS-Sicherheitsräume lautet `SPSSICH.....+Nummer`, also beispielsweise `SPSSICH.....1` für den Sicherheitsraum 1. In dem entsprechenden Editor können die Sicherheitsräume in eine Textdatei ausgegeben werden (Softkey „Drucken“).

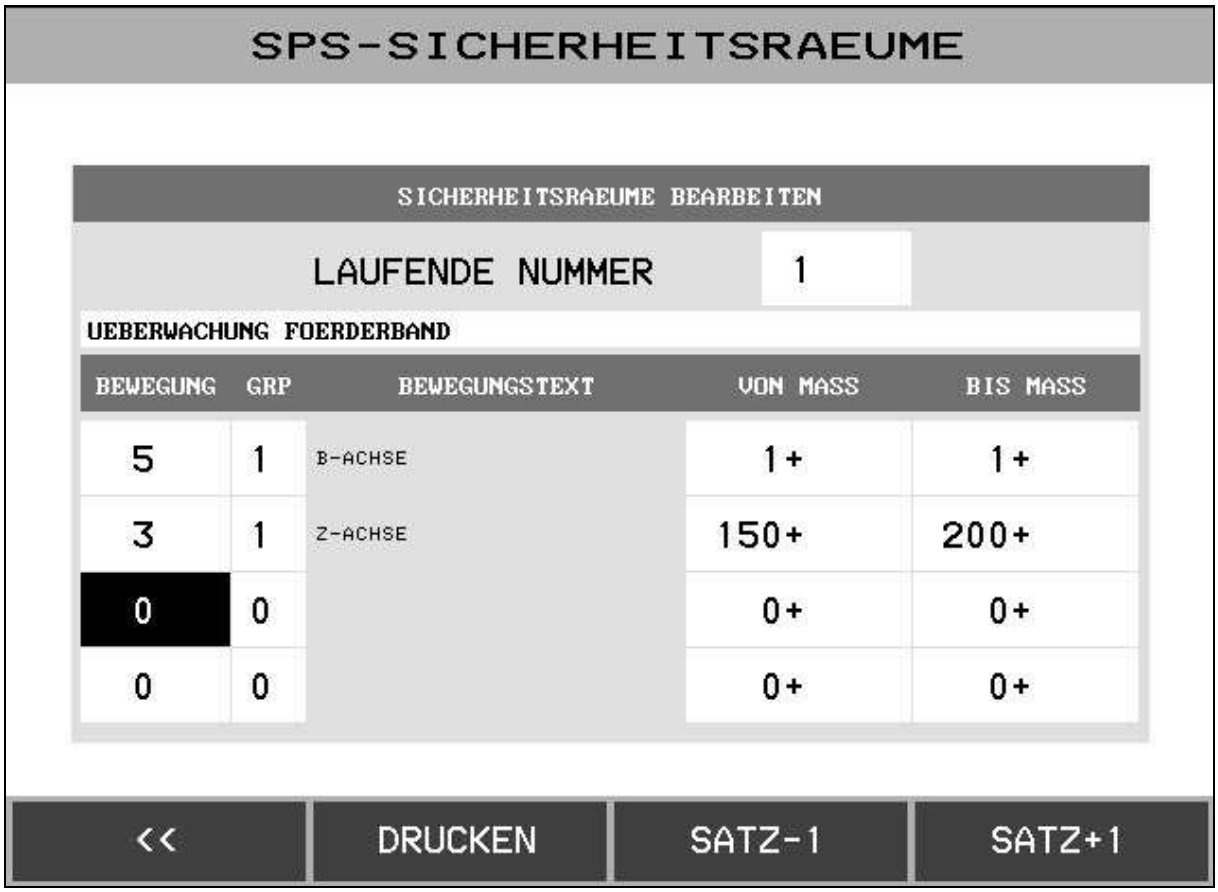


Abbildung 23: SPS-Sicherheitsräume

Der Sicherheitsraum im oben gezeigten Bild gilt als verletzt, wenn die Z-Achse auf einem Mass zwischen 150 und 200 steht und der Ablauf der B-Achse einen Step 0 oder 1 meldet.

Zuordnungstabelle

Mit Hilfe der Zuordnungstabelle können jedem Ablauf verschiedene Daten und Prozesse zugeordnet werden. Die Ablaufprogrammierung greift dann direkt auf diese Zuordnungen zu.

ZUORDNUNGSTABELLE

ABLAUF	GRUPPE	PLATZ	PROZESS	ACHSE	A.PROZ.
1	0	0	0	0	0
2	0	2	3	1	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0
13	0	0	0	0	0
14	0	0	0	0	0
15	0	0	0	0	0
16	0	0	0	0	0

Abbildung 24: Zuordnungstabelle der SPS

Gruppe	Dieser Wert legt fest, zu welcher Gruppe innerhalb des Gruppensystems der Ablauf gehört. Der Opcode 35 mit dem Operanden 24 liefert beispielsweise die hier eingetragene Gruppe.
Platz	Dieser Wert legt fest, welcher Rekord (welche Rekordnummer) innerhalb des Werkstückkontrollblocks diesem Ablauf zugeordnet ist. Der Opcode 204 setzt den Pointer für den WCB auf den Platz, der an dieser Stelle angegeben worden ist.
Prozess	Dieser Wert gibt an, welcher MPI-Prozess dem Ablauf zugeordnet ist. Der Opcode 8 mit dem Operanden 10 startet beispielsweise den Prozess, der hier eingetragen ist.
Achse	Dieser Wert gibt an, welcher Achse dem Ablauf zugeordnet ist. Der Opcode 211 arbeitet beispielsweise mit der Achse, die an dieser Stelle eingetragen ist.
Achsprozess	Dieser Wert gibt an, welcher MPI-Prozess genutzt werden soll, um die Zielposition einer Achse (Opcode 211 und 212) zu ermitteln.

Externe Daten

Die externen Daten stellen eine Möglichkeit dar, Werte dauerhaft zu speichern oder eine Werkstückverfolgung aufzubauen. Für die externen Daten gibt es verschiedene Opcodes, mit deren Hilfe der aktuelle Pointer gesetzt werden kann oder ein Wert aus den Daten gelesen oder in die Daten geschrieben werden kann.

Anwenderspezifische externe Daten

Externe Daten können anwenderspezifisch sein. Die Datenstruktur wird dabei in der Datei `WCB.DBH` definiert und die Daten selber werden in einer Rekorddatei mit dem Namen `PROZESS.WCB` gespeichert. Diese Datei kann beliebig viele Rekords haben. Das erste Byte innerhalb der externen Daten ist das so genannte Merkerbyte und sollte immer definiert sein. Es teilt dem System mit, ob die Daten gültig sind. Dieses Byte wird beim Aufruf der Funktion „Maschine löschen“ für alle Rekords der Datei auf Null gesetzt.
Mit Hilfe dieser Form der externen Daten kann auch eine Werkstückverfolgung aufgebaut werden, in dem mehrere Rekords angelegt werden. Das Feld „Platz“ innerhalb der Zuordnungstabelle der SPS teilt dem Ablauf dabei mit, welcher Rekord zu ihm gehört.
Externe Daten können betextet werden. Da die Datenstruktur anwenderspezifisch ist, muss auch eine eventuelles Anzeige- und Editierbild anwenderspezifisch gestaltet werden.

Fixe externe Daten 2000

Im Bereich der externen Daten ab 2000 stellt das System von sich aus Parameter ab, die von der SPS aus gelesen werden können. In diesem Bereich informiert das System über bestimmte Zustände. Die Daten werden vom System nur gelesen, Schreibzugriffe seitens der SPS haben also keine Auswirkung. Diese Daten sind wordweise angelegt, können aber auch byteweise gelesen werden. Nicht in jeder Betriebssystemart wird jeder Datensatz unterstützt. Diese externen Daten sind vom System aus betextet.

Fixe externe Daten 2100 bis 2299

Das System stellt in ***Softwareständen ab 19.10.2004*** einen fixen Bereich von 200 Byte zur Verfügung, in dem die SPS Daten hinterlegen kann. Dieser fixe Bereich wird durch den Pointer für die externen Daten referenziert. Dabei steht der Wert 2100 für das erste Byte innerhalb dieses fixen Speicherbereichs, der Wert 2101 für das zweite Byte usw. Der Wert 2299 steht für das Byte 200 innerhalb des Speicherbereichs. Die Daten sind intern gespeichert und werden beim Ausführen der Funktion „Maschine löschen“ <u>nicht</u> gelöscht. Sie können <u>nicht</u> auf Diskette gesichert oder von Diskette eingelesen werden!
Die hier hinterlegten Daten können byteweise geschrieben oder gelesen werden. Die Daten können auch wordweise gelesen oder geschrieben werden. Beim wordweisen Schreiben ist darauf zu achten, dass das Byte hinter der angegebenen Adresse ebenfalls geschrieben wird.

Auch diese externen Daten können betextet werden. Zur Editierung steht ein entsprechendes Arbeitsbild zur Verfügung:

WCB-NR.	WERT
2100	1
2101	2
2102	3
2103	4
2104	5
2105	6
2106	7
2107	0
2108	0
2109	0
2110	0
2111	0
2112	0

Abbildung 25: Editierung der fixen externen Daten ab 2100

Betriebsdaten BDE

Das System kann aus der SPS heraus Betriebsdaten pflegen. Hierfür besteht eine Datenbeschreibung in der BDE.DBH. Die Betriebsdaten sind in der Rekorddatei PROZESS.BDE gespeichert und bestehen grundsätzlich aus einem Merkerbyte (nicht veränderbar), einem Timer zum Aufaddieren von Zeiten, einem Zähler zum Aufaddieren von Stückzahlen und einem Wert mit Nachkommastelle, um beispielsweise Längen zu zählen. Mit den SPS-Befehlen 14, 15, 208 und 209 können die entsprechenden Daten modifiziert werden.

In dem Bild BDEEDIT können die Daten angezeigt, exportiert und gelöscht werden. Dieses Bild ist standardmäßig nicht im Menü eingetragen und muss auf Wunsch in die Menüdatei eingefügt werden. Dabei ist darauf zu achten, dass beim Editieren der Daten der Runlevel so gewählt wird, dass die SPS gesperrt ist!

Anhang

Liste der möglichen Operanden für den SPS-Opcode 7 im OALSYS-System

Operand	Bedeutung
1	TEILENUMMER PLUS 1 (bezogen auf die Defaulttabelle)
2	TEILENUMMER = 1 (bezogen auf die Defaulttabelle)
3	TEILENUMMER MINUS 1 (bezogen auf die Defaulttabelle)
101..199	DEFAULTTABELLE AUF 1..99 setzen

Liste der möglichen Operanden für den SPS-Opcode 8 im OALSYS-System

Die nachfolgende Tabelle listet die möglichen Operanden für den SPS-Opcode 8 innerhalb des OALSYS-Betriebssystems und erläutert die entsprechende Bedeutung.

Operand	Bedeutung
1	UNTERBRECHUNG TEACHEBENE 1
2	UNTERBRECHUNG TEACHEBENE (AKKU)
3	SETZE SOFTKEY AUS AKKU
4	LOESCHE SOFTKEY AUS AKKU (0=ALLE)
5	AKTUELLEN TEILEZAEHLER SCHREIBEN IN AKKU
6	AKTUELLEN TEILEZAEHLER LADEN AUS AKKU
7	AKTUELLEN ETAGENZAEHLER SCHREIBEN IN AKKU
8	AKTUELLEN ETAGENZAEHLER LADEN AUS AKKU
9	SICHERHEITSRAUM AUS AKKU LADEN
10	MPI-PORZESS STARTEN (PROZESSNUMMER AUS AKKU)
11	TESTE STOP PASSIV GRUPPE (Sprungbedingung=ja, wenn Stop passiv) ***implementiert in Softwareständen ab 21.09.2004***

Liste der möglichen Operanden für den Opcode 251 im OALSYS-System

Operand	Bedeutung
1	TEILENUMMER PLUS 1 (bezogen auf die Tabelle, die im Akku steht)
2	TEILENUMMER = 1 (bezogen auf die Tabelle, die im Akku steht)
3	TEILENUMMER MINUS 1 (bezogen auf die Tabelle, die im Akku steht)

Liste der möglichen Operanden für den SPS-Opcode 35 und 85

Die nachfolgende Tabelle listet die möglichen Operanden für den SPS-Opcode 35 und 85 und erläutert die entsprechende Bedeutung.

Operand	Bedeutung
0	AUFTRAG (POINTER)
1	STEP (POINTER)
2	SYSTEM AUFTRAG
3	TASTE START=1 / STOP=2
4	AKKUINHALT
5	STEPFREIGABE
6	AUFTRAGSZAEHLER
7	EXTERNE DATEN (BYTE) LESEN
8	EXTERNE DATEN (BYTE) SCHREIBEN
9	UPRO-ADRESSE LESEN
10	UPRO-ADRESSE SCHREIBEN
11	ABLAUF MERKER LESEN
12	ABLAUF MERKER SCHREIBEN
13	TIMER INHALT
14	NUMMER DER STOERUNG
15	FOLGEAUFTRAGSZAEHLER
16	USERLEVEL LESEN
17	EXTERNE DATEN (WORD) LESEN
18	EXTERNE DATEN (WORD) SCHREIBEN
19	RUNLEVEL LESEN
20	ANZAHL AKTIVER ACHSEN
21	ANZAHL AKTIVER GRUPPEN
22	OVERRIDE LESEN
23	OVERRIDE SCHREIBEN
24	EIGENE GRUPPE LESEN Implementiert in Softwareständen ab 12.12.2003
25	PROGRAMMNUMER LESEN Implementiert in Softwareständen ab 10.08.2005
26	EIGENE ACHSPOSITION LESEN Implementiert in Softwareständen ab 26.10.2005
27	STUNDE AUS DATUM Implementiert in Softwareständen ab 10.03.2006
28	MINUTE AUS DATUM Implementiert in Softwareständen ab 10.03.2006
29	SEKUNDE AUS DATUM Implementiert in Softwareständen ab 10.03.2006
30	TAG AUS DATUM Implementiert in Softwareständen ab 10.03.2006
31	MONAT AUS DATUM Implementiert in Softwareständen ab 10.03.2006
32	JAHR AUS DATUM Implementiert in Softwareständen ab 10.03.2006
33	BCD-Wert ganzzahlig holen Implementiert in Softwareständen ab 20.08.2006
34	AKTUELLER WOCHENTAG AUS DATUM Implementiert in Softwareständen ab 08.2011

Liste der Bedeutung der globalen Merker 1000 und höher

Die nachfolgende Liste zeigt auf, welche Bedeutung die globalen Merker mit einer Nummer größer 1000 haben.

Globaler Merker	Bedeutung
1000	Automatikmaske ist angewählt, Runlevel 64 liegt an
1001	Anlage ist scharf, Arbeitstakt aktiv
1002	Programmnummer wurde verändert
1003	Anlage wurde neu eingeschaltet, Netzausfall
1004	Verbindung zum Bedienpult ist aktiv
1005	Maschine wurde gelöscht
1006	Der Einzeltest ist aktiv
1007	Die Achsparameter wurden an die Antriebe gesendet
1008	Die Abläufe sind (logisch) gesperrt
1009	Runlevel 128 liegt an
1010	Die Starttaste am Bedienpult soll keinen Startmerker setzen
1011	Es steht eine Störung an
1012	Anlage ist betriebsbereit gesetzt, wenn die CPU mit allen Initialisierungen fertig ist implementiert in Softwareständen ab 21.09.2004
1022	MPI-Fehler
1023	Asi-Störung

Liste der Bedeutung der externen Daten ab 2000

Die nachfolgende Liste zeigt auf, welche Bedeutung die externen Daten mit einer Nummer größer 2000 haben.

Externe Daten	Bedeutung
2000	AKTUELLE PALETTIERTABELLE
2002	BILDINDEX AUTOMATIK (THEUNISSEN)
2004	AENDERUNGSBYTE EDITOR STUECKLISTEN
2006	ZEILENNUMMER STUECKLISTEN
2008	TASTE AUS 2-ZEILEN-LCD-TERMINAL
2010	ADRESSE FEHLERHAFTES ASI-MODUL
2012	TEACHEBENE UNTER TEST
2014	TEACHEBENE OBEN
2016	ANGEZEIGTER SATZ IM WCBAUTOM