Benutzerhandbuch

Schrittmotortreiber
RS Stock No.: 434540
RS Stock No.: 434542
RS Stock No.: 434543

Vorsichtsmaßnahmen und Hinweise im Text:

Achtung
Verschiedene Arten von Gefahren, die zu Sachschäden oder Verletzungen führen können.

Informationen
Empfehlungen, Ratschläge oder Verweise auf ein anderes Dokument.

Hervorhebungen und Formatierungen von Textfragmenten:

- Bezeichnung oder Markierung nach Wahl

1) Einige Aktionen sollten in einer bestimmten Reihenfolge durchgeführt werden

— Gemeinsame Aufzählungen

Herstellerinformationen

RS Components verpflichtet sich zur kontinuierlichen Weiterentwicklung und behält sich das Recht vor, Änderungen und Verbesserungen am Design und der Software des Produkts ohne vorherige Ankündigung vorzunehmen.
Die Informationen in diesem Handbuch können jederzeit und ohne vorherige Ankündigung geändert werden.

Inhalt

1. Zweck des Controllers

Der Controller ist für den Betrieb mit Schrittmotoren konzipiert. Das Gerät kann über eine SPS (via RS-485 Modbus ASCII/RTU) gesteuert werden oder im Standalone-Modus gemäß dem voreingestellten Ausführungsprogramm arbeiten.

Der Controller bietet Vollschrittbetrieb oder Mikroschrittbetrieb bis zu 1/256. Der Controller sorgt für eine gleichgängige Bewegung mit geringen Vibrationen und hoher Positioniergenauigkeit.

Der Controller bietet die folgenden Steuerungsmodi:

- Programmsteuerungsmodus – Standalone-Betrieb gemäß dem voreingestellten Ausführungsalgorithmus oder Echtzeitsteuerung von einem PC oder einer SPS durch Befehle über das Modbus-Protokoll.
- Analoge Drehzahlregelung – Die Drehzahl des Motors wird durch das Potentiometer an der Vorderseite des Controllers eingestellt.
- STEP/DIR-Steuerungsmodus – Steuerung der Motorposition durch logische Pulssignale.

Der Controller verfügt über 8 Logikeingänge und 10 Logikausgänge (2 dieser Ausgänge sind Hochspannungsausgänge). Der Zustand der E/A kann von einem vom Benutzer ausgeführten Programm oder direkt über Modbus-Befehle gelesen oder gesetzt werden. Ein internes Ausführungsprogramm kann über Mini-USB-Anschluss oder RS-485 vom Controller heruntergeladen und auf den Controller hochgeladen werden.

Wir bieten Software zur Einstellung, Konfiguration oder Bearbeitung von Ausführungsprogrammen und zum Hochladen des Ausführungsprogramms in den Speicher des Controllers.

Der Controller verfügt über einen Überhitzungsschutz.

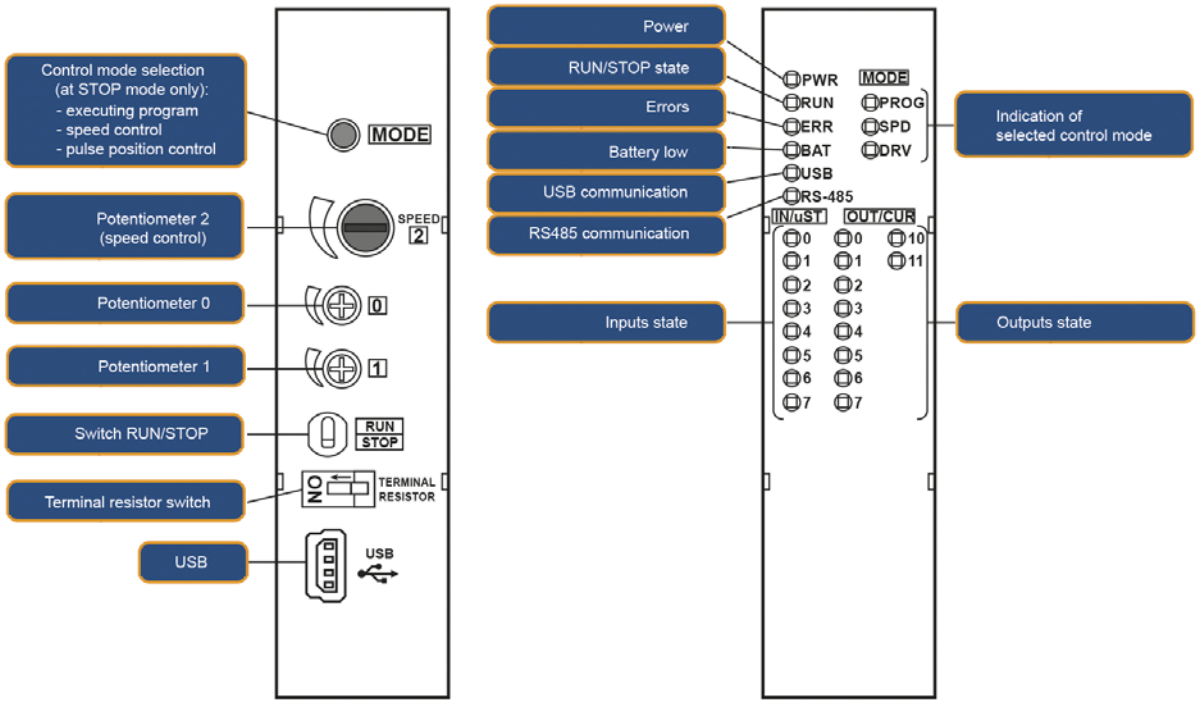


Abb. 1. Zweck der Bedienelemente

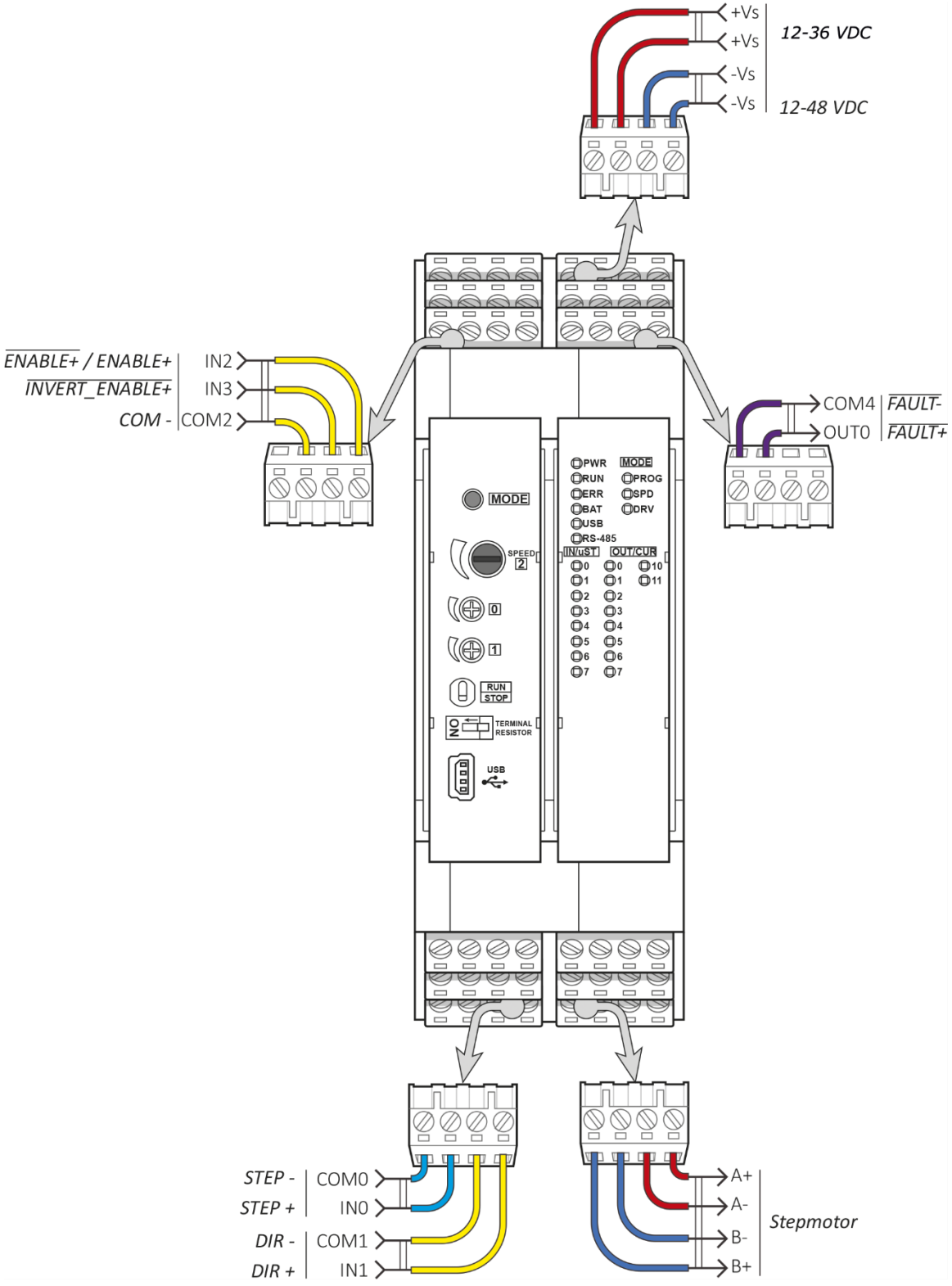


Abb. 2a. Klemmenbelegung – Treiber-Steuermodus

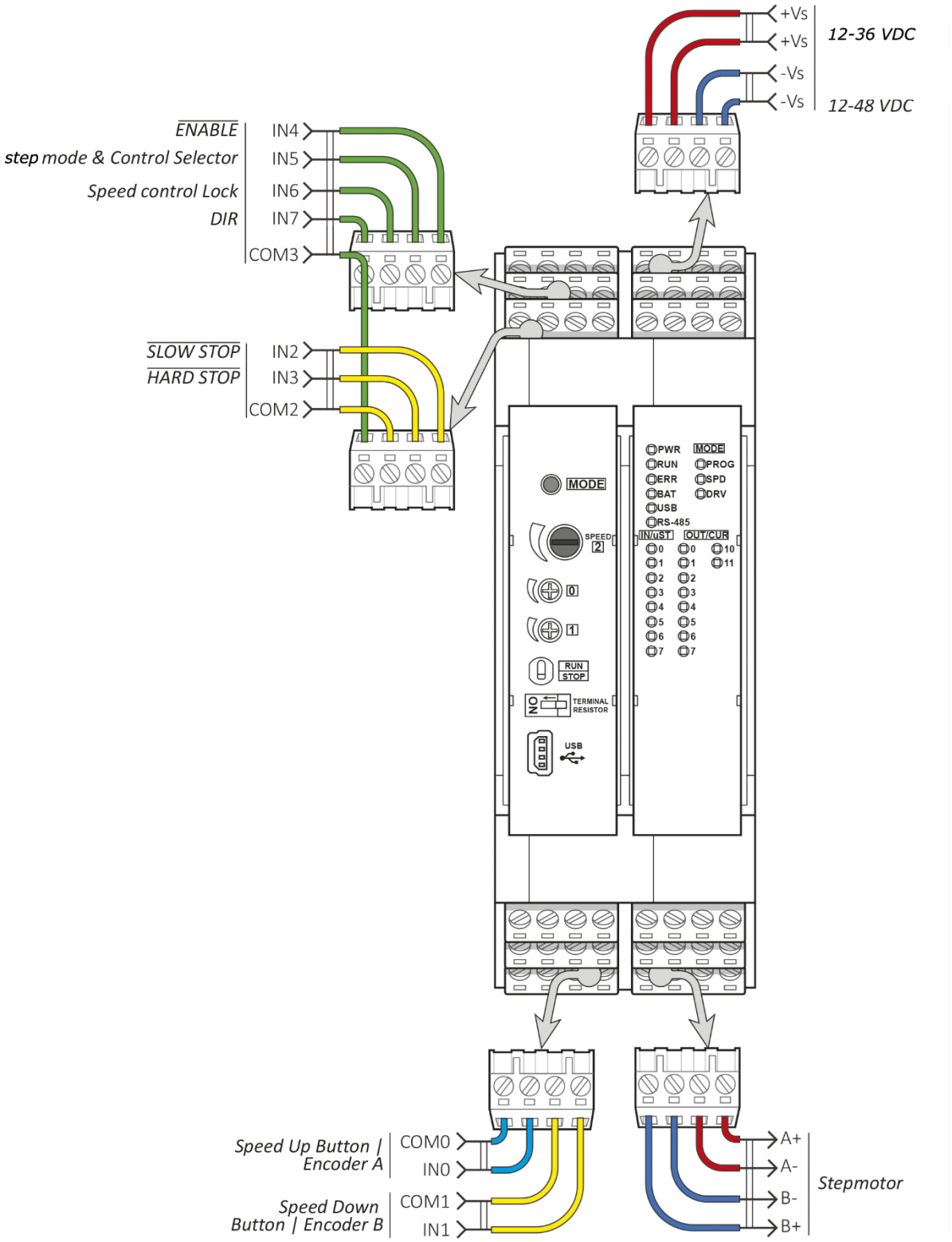


Abb. 3b. Klemmenbelegung – Drehzahlregelungsmodus

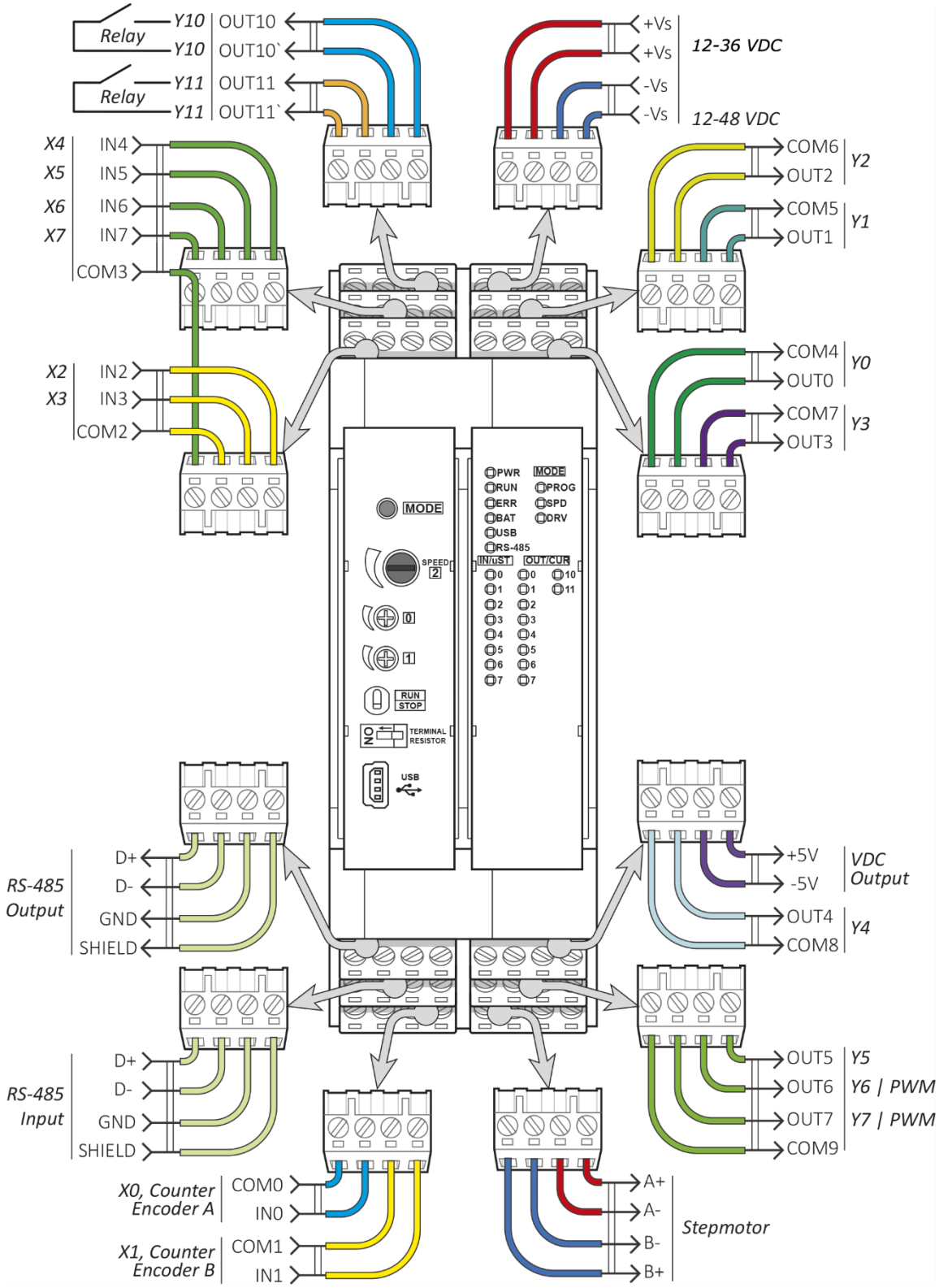


Abb. 4c. Klemmenbelegung – Benutzerprogramm-Steuermodus

Abb. 1 zeigt die Vorderseite des Controllers mit Bedien- und Anzeigeelementen. Die **PWR**-Anzeige zeigt das Vorhandensein der Versorgungsspannung an. **RUN** zeigt den aktuellen Zustand des Controllers an (RUN oder STOP). Im **PROG**-Modus (Ausführung eines Anwenderprogramms) und

im **SPD**-Modus (Drehzahlregelung) zeigt der aktive Zustand der **RUN**-Anzeige die Ausführung des Programms an, der inaktive Zustand den Stoppzustand. Im **DRV**-Modus (Step/Dir-Modus) zeigt der inaktive **RUN**-Zustand die Möglichkeit an, Treiberparameter über die Bedienelemente des Controllers einzustellen. Der aktive **RUN**-Zustand zeigt den Eintritt in den Betriebsmodus an, Parameteränderungen sind deaktiviert. Die Möglichkeit zum Umschalten zwischen den Modi PROG / SPD / DRV ist im **RUN**-Zustand deaktiviert. Im **STOP**-Zustand kann die Umschaltung des Betriebsmodus mit der Taste **MODE** erfolgen. **ERR** zeigt das Vorliegen von Fehlern an. **BAT** zeigt einen niedrigen Batteriestand im Gerät an. **USB** und **RS485** zeigen den Prozess eines über USB übertragenen Modbus-Frames an und
RS-485. Im **PROG**- und **SPD**-Modus zeigen die LEDs **IN0 ...IN7** das Vorhandensein eines hohen Pegels eines Logiksignals am entsprechenden Eingang an. Die aktiven Zustände der **OUT0 ...OUT11**-Anzeigen zeigen den offenen Zustand des Transistorausgangs an (siehe Abb. 5– Abb. 8). Im **DRV**-Modus (Step/Dir-Modus) zeigen die LEDs der Eingänge **IN0 ...IN7** die Mikroschritteinstellung an, die Ausgänge **OUT0 ...OUT3** zeigen die Einstellung des Betriebsstroms an, **OUT4 ...OUT7** zeigen die Einstellung des Haltestroms an.

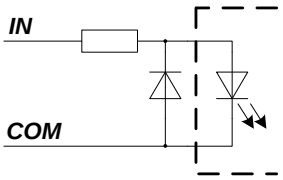


Abb. 5. Eingänge IN0 und IN1

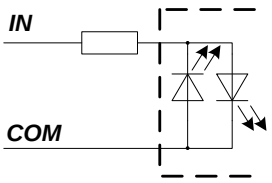


Abb. 6. Eingänge IN2 – IN7

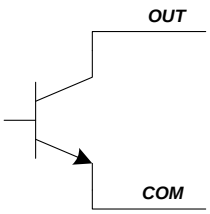


Abb. 7. Ausgänge OUT0 – OUT7

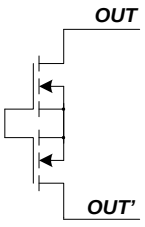


Abb. 8. Ausgänge OUT10 und OUT11

Die Position der Potentiometer **0, 1, 2 (SPEED)** wird in 12-Bit-Werte umgewandelt, die zur weiteren Verwendung in einem Ausführungsprogramm zugänglich sind und über einen Modbus-Befehl gelesen werden können. Im **SPD** (Drehzahlregelungsmodus) wird mit dem Potentiometer **2 (SPEED)** die Drehzahl eingestellt, mit Potentiometer 0 die Beschleunigungs- und Verzögerungsrate

und mit 1 der Motorarbeitsstrom. Im **DRV** (Pulse/Dir)-Modus wird mit Potentiometer 2 die Mikroschrittweite eingestellt, mit 0 der Motorarbeitsstrom und mit 1 der Haltestrom.

Der Schalter **RUN/STOP** wird verwendet, um die Programmausführung in den Steuerungsmodi **PROG** und **SPD** zu starten und zu stoppen. Im **DRV**-Steuerungsmodus wird er zum Umschalten zwischen Parametereinstellungen und Betrieb verwendet.

Abb. 2 (a–c) zeigt die Anschlussklemmen des Controllers und deren Zweck abhängig vom Betriebsmodus.

2. Technische Daten

Charakteristik		Wert	
		min	max
Versorgungsspannung, VDC	RS 434540	12	36
	RS 434542 und RS 434543	12	49
Max. Ausgangsstrom pro Phase, A	RS 434540	0.15	1.5
	RS 434542	1.0	4.2
	RS 434543	2.8	8.0
Hoher Pegel der Logikeingänge, VDC			2.4
Niedriger Pegel der Logikeingänge, VDC		0,7	
Empfohlene Spannung (beste Ansprechzeit) der Logikeingänge, VDC			5
Max. Spannungspegel der Logikeingänge, VDC			24(1)
Spannung des Logik-Transistor-Ausgangs, VDC			80
Maximaler Strom des Logik-Transistor-Ausgangs, mA			50
Spannung des Logikrelais-Ausgangs, VAC/VDC			350
Maximaler Strom des Logikrelais-Ausgangs DC (AC/DC), mA			250 (~120)
Maximaler Strom des zusätzlichen Ausgangs +5VDC, mA			200

Dauer des hohen/niedrigen Spannungspegels der Signale	STEP-Signal im Schritt-/Richtungs-Steuermodus (DRV), ns	250(2)	
	Signale an den Eingängen IN0 und IN1, ns	70(2)	
	Signale an den Eingängen IN2.. .IN7, µs	5(2)	
PWM-Signal-Generierungsfrequenz, Hz		0,3	5000
Basisbefehlsdauer, µs(3)		20	

(1) – Bei Verwendung einer Spannung von mehr als 8V für die Eingänge IN2.. .IN7 und mehr als 12V für die Eingänge IN0.. .IN1 müssen Strombegrenzungswiderstände installiert werden.
– Empfohlene Widerstände für die Eingänge IN2.. .IN7: mindestens 270 Ohm bei Verwendung einer 12V-Signalquelle und mindestens 1 kOhm bei Verwendung von 24V.
– Der empfohlene Widerstand für die Eingänge IN0 und IN1 beträgt 9,1 kOhm bei Verwendung einer 24V-Logiksignalquelle.

(2) unter der Bedingung eines 5VDC Hochspannungspegels

(3) - ohne Berücksichtigung der Rückkehr zur Nulllinie, der Einstellung der Ausgänge und des Lesens der Eingänge

Zusätzliche Informationen

№	Charakteristik	Wert
1	Mögliche Baudraten für die RS-485 Datenübertragung	1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000
2	Mögliche Mikroschritteinstellungen	1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/128, 1/256
3	Kommunikationsprotokoll	Modbus RTU, Modbus ASCII
4	Programmiersprache	IL (Anweisungsliste), LD* (Kontaktplan)

* *Spezielle Software wird benötigt, um LD in IL zu konvertieren, bevor das Programm in die Steuerung geschrieben wird.*

3. BETRIEBSABLAUF

Der Betriebsablauf des Controllers ist wie folgt:

- Lesen externer Geräte (Logikeingänge, Modbus-Coils);
- Verarbeitung des Benutzerprogramms;
- Einstellung neuer Zustände der Ausgabegeräte (Logikausgänge, Modbus-Discrete-Inputs, Ausführung der Bewegung).

Das Benutzerprogramm besteht aus einer Folge von Steueranweisungen (Befehlen), die die endgültige Funktionalität bestimmen. Der Controller führt die Befehle nacheinander aus. Der gesamte Programmdurchlauf wird kontinuierlich wiederholt. Die Zeit, die für einen Programmdurchlauf benötigt wird, wird als Zykluszeit bezeichnet, und die Programmdurchläufe werden als zyklisches Scannen bezeichnet.

Die Controller können im Echtzeitmodus betrieben werden und sowohl für den Aufbau lokaler Automatisierungsknoten als auch für verteilte E/A-Systeme mit Organisation des Datenaustauschs über die RS-485-Schnittstelle mit Modbus-Protokoll verwendet werden.

Wir bieten spezielle Software zum Erstellen und Debuggen von Benutzerprogrammen an, die keine erheblichen Computerressourcen benötigt und ein einfaches Werkzeug für Benutzer darstellt. Es werden zwei Programmiersprachen verwendet: LD (Kontaktplanlogik oder Leiterdiagramme) und IL (Liste der Anweisungen).

3.1. Das Funktionsprinzip von Relais-Schaltkreisen und Leiterdiagrammen im Controller

Die Sprache der Kontaktplandiagramme ist eine Ableitung des Relais-Kontakt-Schaltplans in einer vereinfachten Darstellung. Die Relais-Kontakt-Schaltungen im Controller verfügen über eine Reihe von Grundkomponenten, wie z. B.: Schließer, Öffner, Spule (Ausgang), Zeitgeber, Zähler usw., sowie angewandte Anweisungen: mathematische Funktionen, Motorsteuerbefehle, Datenverarbeitung und eine große Anzahl von Spezialfunktionen und Befehlen. Wir können davon ausgehen, dass der Controller aus Dutzenden oder Hunderten von separaten Relais, Zählern, Zeitgebern und Speicher besteht. Alle diese Zähler, Zeitgeber usw. existieren physisch nicht, sondern werden vom Prozessor modelliert und sind für den Datenaustausch zwischen eingebauten Funktionen, Zählern, Zeitgebern usw. ausgelegt.

Die Relais-Kontakt-Logiksprache im Controller ist den grundlegenden Relais-Kontakt-Schaltkreisen sehr ähnlich, wenn wir die verwendeten Grafiksymbole vergleichen. Es gibt zwei Arten von Logik in Relais-Kontakt-Schaltungen: kombinierte Logik, d. h. Schaltungen, die aus voneinander unabhängigen Fragmenten bestehen, und sequenzielle Logik, bei der alle Schritte des Programms miteinander verbunden sind und die Schaltung nicht parallelisiert werden kann.

Kombinierte Logik

Der erste Abschnitt der Schaltung besteht aus einem Schließer X0 und einer Spule Y0, die den Zustand des Ausgangs Y0 bestimmt. Wenn der Zustand des Kontakts X0 offen ist (logisch "0"), ist auch der Zustand des Ausgangs Y0 offen (logisch "0"). Wenn der Kontakt X0 geschlossen ist, ändert auch der Ausgang Y0 seinen Zustand in geschlossen (logisch "1").

Der zweite Abschnitt der Schaltung besteht aus einem Öffner X1 und einer Spule Y1, die den Zustand des Ausgangs Y1 bestimmt. Im Normalzustand des Kontakts X1 ist der Ausgang Y1 geschlossen (logisch "1"). Wenn sich der Zustand des Kontakts X1 in offen ändert, ändert auch der Ausgang Y1 seinen Zustand in offen.

Im dritten Abschnitt der Schaltung hängt der Zustand des Ausgangs Y2 von einer Kombination der Zustände der drei Eingangskontakte X2, X3 und X4 ab. Ausgang Y2 ist geschlossen, wenn X2 ausgeschaltet und X4 eingeschaltet ist oder wenn X3 und X4 eingeschaltet sind.
Das allgemeine Schema ist eine Kombination aus drei Segmenten, die unabhängig voneinander arbeiten.

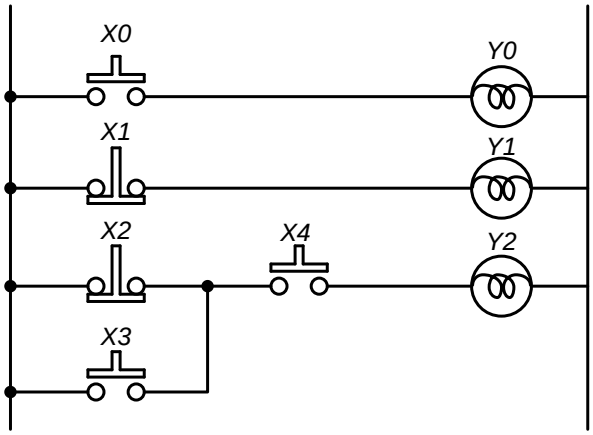


Abb. 9. Relais-Schaltkreis

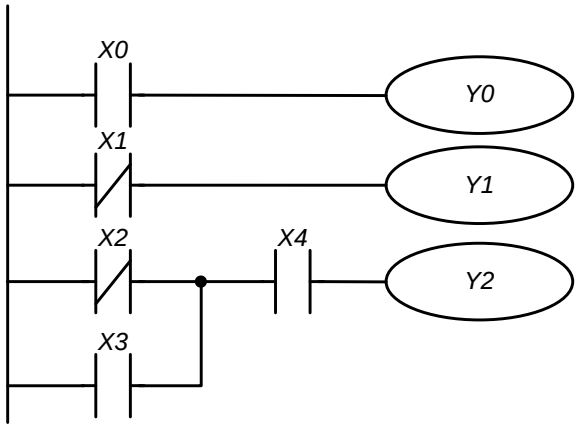


Abb. 10. Kontaktplan im Controller

Sequenzielle Logik

In den sequenziellen Logikschaltungen ist das Ergebnis der Ausführung in einem vorherigen Schritt eine Eingangsbedingung für den nächsten Schritt. Mit anderen Worten: Ein Ausgang im vorherigen Schritt ist ein Eingang im folgenden Schritt.

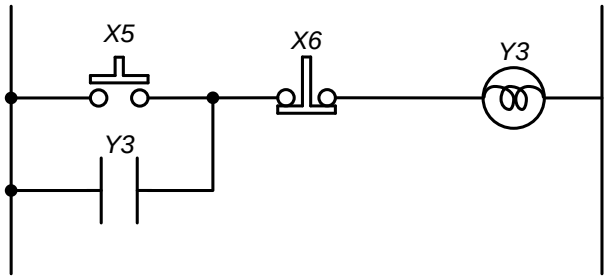


Abb. 11. Relais-Schaltkreis

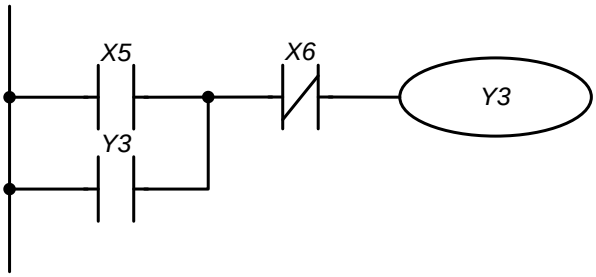


Abb. 12. Kontaktplan im Controller

Wenn der Kontakt X5 geschlossen ist, ändert der Ausgang Y3 seinen Zustand in geschlossen. Wenn X5 jedoch wieder geöffnet wird, behält Y3 seinen geschlossenen Zustand bis zu dem Zeitpunkt bei, an dem X6 geöffnet wird. In dieser Schaltung ist der Ausgang Y3 selbsthaltend.

3.2. Unterschiede zwischen der Logik realer Relais-Kontakt-Schaltungen und Kontaktplänen im Controller

Alle angegebenen Steuerprozesse werden in herkömmlichen Relais-Kontakt-Schaltkreisen gleichzeitig (parallel) ausgeführt. Jede Änderung des Zustands der Eingangssignale wirkt sich sofort auf den Zustand der Ausgangssignale aus.

Eine Änderung des Zustands der Eingangssignale, die während des aktuellen Programmdurchlaufs im Controller aufgetreten ist, wird erst im

nächsten Programmzyklus erkannt. Dieses Verhalten wird aufgrund der kurzen Zykluszeit geglättet.

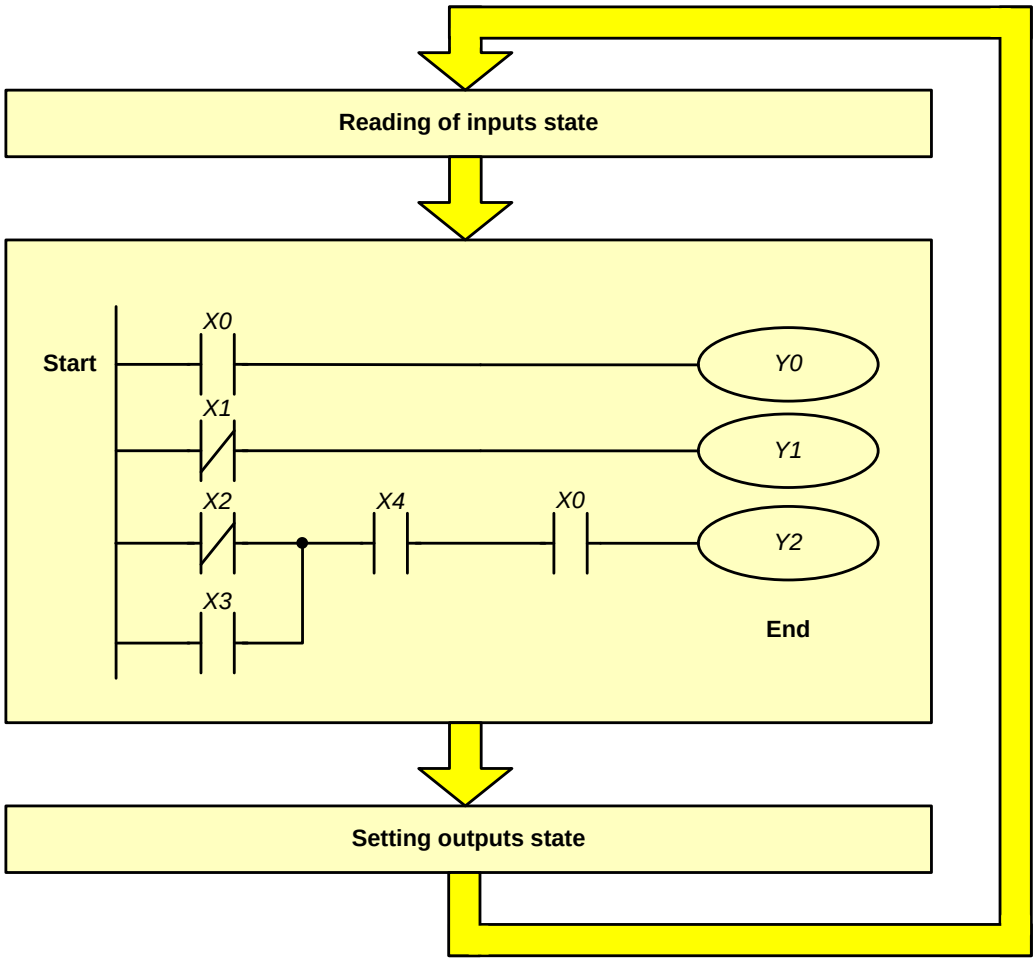


Abb. 13. Ablauf im Controller

Während des Betriebs liest der Controller kontinuierlich den aktuellen Zustand der Eingänge und ändert den Zustand der Ausgänge (ein/aus) abhängig vom Benutzerprogramm.

Abb. 13 zeigt ein Flussdiagramm eines Programmzyklus.

In der ersten Stufe liest der Controller den Zustand der physischen und virtuellen Eingänge (deren Zustände über Modbus-Coils eingestellt werden) und puffert sie im internen Speicher des Controllers.

In der zweiten Stufe wird der Zustand der gepufferten Eingänge verarbeitet und der Zustand der Ausgänge im Controllerspeicher ändert sich gemäß einem gegebenen Benutzerprogramm. Somit erfolgen alle Änderungen der Ausgänge, ohne ihren physischen Zustand zu ändern. Während dieser Phase kann sich der Zustand der physischen und virtuellen Eingänge ändern, aber die

nächste Pufferung des aktualisierten Zustands erfolgt in der ersten Stufe des nächsten Zyklus des Benutzerprogramms.

In der dritten Stufe ändert der Controller den Zustand der physischen und virtuellen Ausgänge.

Ein weiterer Unterschied zwischen der Relais-Kontakt-Logik des Controllers und herkömmlichen Relais-Kontakt-Schaltkreisen besteht darin, dass die Benutzerprogramme nur zeilenweise von links nach rechts und von oben nach unten ausgeführt werden. Beispielsweise führt eine Schaltung mit umgekehrter Stromrichtung (Abschnitt a-b in) zu einem Fehler während der Kompilierung im Controller.

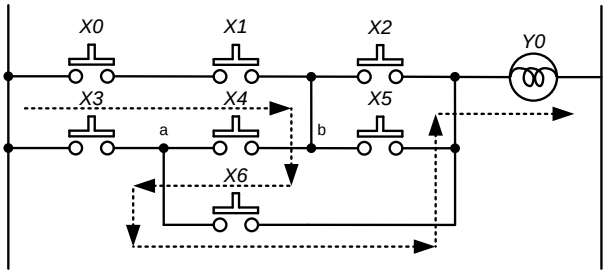


Abb. 14. Relais-Schaltkreis

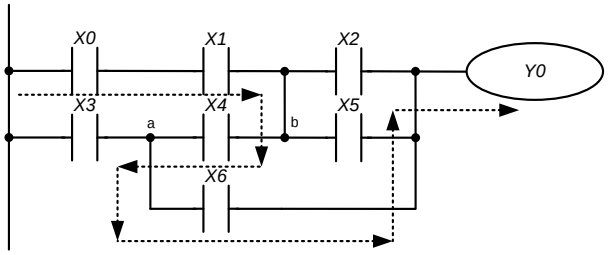


Abb. 15. Relais-Schaltkreis des Controllers

Ein Fehler in der 3. Zeile

3.3. Operanden

Alle internen Objekte (Geräte) des Controllers – Operanden – sind in einige verschiedene Typen unterteilt und haben Adressen. Jeder Typ hat seine Bezeichnung und sein Format, das bestimmt, welchen Platz er im Speicher des Controllers einnimmt. So werden beispielsweise Eingangsrelais mit „X“ bezeichnet und haben ein 1-Bit-Format, und allgemeine Datenregister werden mit „D“ bezeichnet und haben ein 16-Bit- (1 Wort) oder 32-Bit- (2 Wörter) Format.

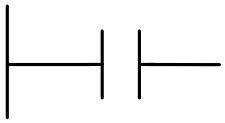
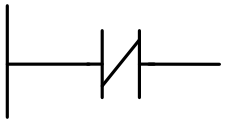
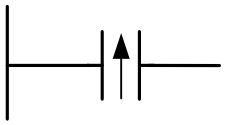
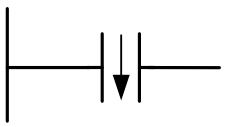
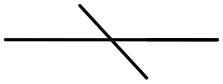
Typ und Name des Operanden		Beschreibung
Eingang	X	Eingangsrelais. Bestimmen den Zustand externer Bit-Geräte, die an die Eingangsklemmen des Controllers angeschlossen sind, und den Zustand virtueller Eingänge, deren Status über das Modbus-Protokoll eingestellt werden kann. Diese Operanden können einen von zwei möglichen Werten annehmen: 0 oder 1. Die Adressierung ist oktal: X0, X1.. . X7, X10, X11, ...
Ausgang	Y	Ausgangsrelais. Bestimmen den Zustand der Ausgangsklemmen des Controllers, an die die Last angeschlossen ist, und den Zustand der virtuellen Ausgänge, deren Status über das Modbus-Protokoll gelesen werden kann. Im Programm können entweder Kontakte oder Spulen sein. Diese Operanden können einen von zwei möglichen Werten annehmen: 0 oder 1. Die Adressierung ist oktal: Y0, Y1.. .Y7, Y10, Y11, ...
Merker	M	Hilfsrelais. Es ist ein Speicher für binäre Zwischenergebnisse. Im Anwenderprogramm können entweder Kontakte oder Spulen verwendet werden. Diese Operanden können einen von zwei möglichen Werten annehmen: 0 oder 1. Die Adressierung ist dezimal: M0, M1.. .M7, M8, M9,...
Zeitgeber	T	Zeitrelais. Das Programm kann zum Speichern des aktuellen Timerwerts verwendet werden und hat ein 16-Bit-Format. Außerdem können diese Operanden als Kontakte verwendet werden und nehmen einen von zwei Zuständen an: 0 oder 1. Die Adressierung ist dezimal: T0, T1 ...T64

Zähler	C	Ein Zähler wird zur Implementierung der Zählfunktion verwendet. Das Programm kann zum Speichern des aktuellen Zählerwerts verwendet werden und hat ein 16-Bit- oder 32-Bit-Format. Es kann auch als Kontakt verwendet werden und eine von zwei möglichen Bedeutungen annehmen: 0 oder 1. Die Adressierung ist dezimal: C0, C1.. .C66
Dezimalzahlkonstante	K	Bestimmt eine Zahl in Dezimal
Hexadezimal-Konstante	H	Bestimmt eine Zahl hexadezimal
Gleitkommakonstante	F	Bestimmt eine Gleitkommazahl
Datenregister	D	Datenspeicher. 16-Bit- oder 32-Bit-Format. Die Adressierung ist dezimal: D0, D1,..., D391. Bei 32-Bit-Daten belegt ein Element zwei Register. Beispielsweise werden beim Lesen von 32-Bit-Daten aus dem Register D10 die Daten aus den Registern D10 und D11 gelesen.
Indexregister	A	Datenspeicher für Zwischenergebnisse und Indexidentifikation. 16-Bit-Format. Adressierung: A0 – A7, B0 – B7 dezimal
	B	
Zeiger	P	Eine Adresse für den Unterprogrammaufruf. Dezimal.
Interruptzeiger	I	Adresse der Interrupt-Bearbeitung. Dezimal.

3.4. Grafische Symbole der Steueranweisungen im Kontaktplan

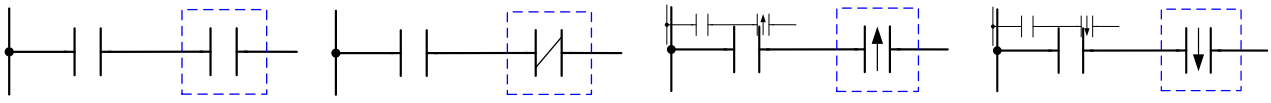
Ein Relais-Kontakt-Schaltkreis besteht aus einer vertikalen Linie links und horizontalen Linien, die sich nach rechts erstrecken. Die vertikale Linie links ist eine Busleitung, horizontale Linien sind Befehlszeilen = Schritte. Auf den Befehlszeilen, die zu den rechts befindlichen Befehlen (Anweisungen) führen, befinden sich Symbole für die Eingangsbedingungen. Die logischen Kombinationen dieser Eingangsbedingungen bestimmen, wann und wie rechtsgerichtete Befehle ausgeführt werden.

Die folgenden Symbole werden in Relais-Kontakt-Schaltkreisen verwendet:

Symbol	Beschreibung	Befehl	Operanden
	Eingangskontakt – Schließer	LD	X, Y, M, T, C
	Eingangskontakt – Schließer	LDI	X, Y, M, T, C
	Eingangsimpulskontakt – steigende Flanke	LDP	X, Y, M, T, C
	Eingangsimpulskontakt – fallende Flanke	LDF	X, Y, M, T, C
	Ausgangssignal (Spule)	OUT	Y, M
	Grundlegende Anweisungen und Anwendungshinweise	Siehe Kapitel 7. Anwendungshinweise	Siehe Kapitel 7. Anwendungshinweise
	Logikinvertierung	INV	-

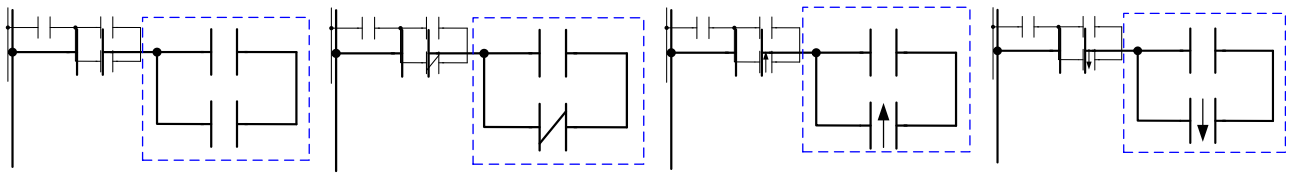
Eingangskontakte können zu Serien- und Parallelblöcken zusammengefasst werden:

Serielle Anschlüsse:



AND *ANI* *ANDP* *ANDF*

Parallelschaltungen:



OR *ORI* *ORP* *ORF*

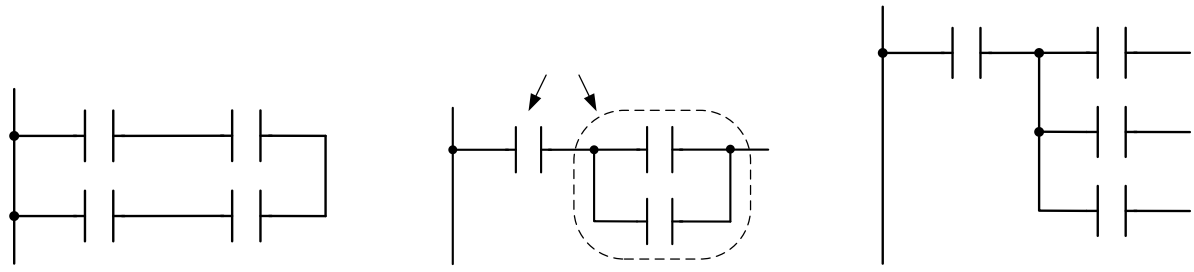


Abb. 16. Grafische Symbole von Steueranweisungen

Der Programmscan beginnt in der oberen linken Ecke des Diagramms und endet in der unteren rechten Ecke. Das folgende Beispiel veranschaulicht den Ablauf eines Programms:

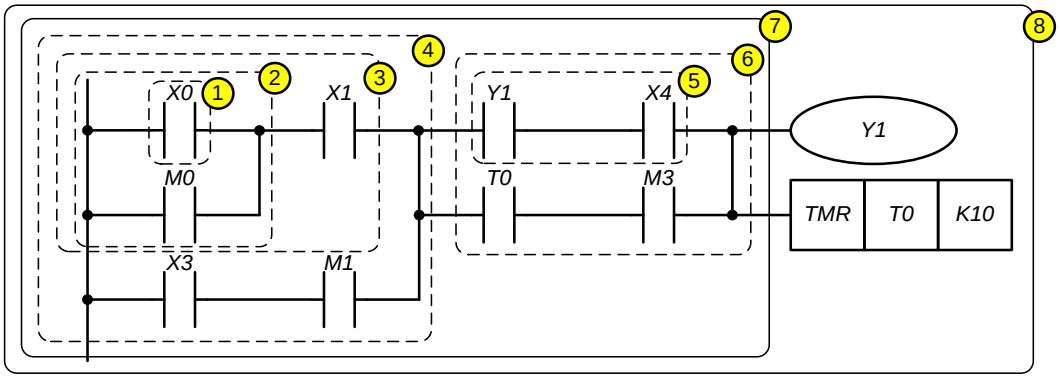
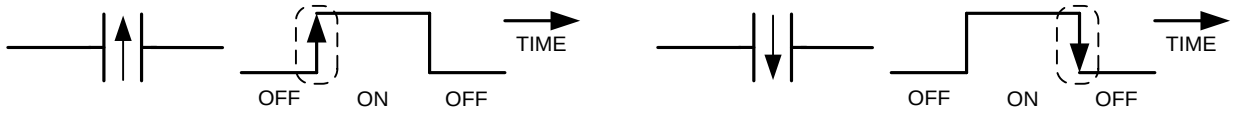


Abb. 17. Ablauf eines Programms

1	LD	X0	
2	OR	M0	
3	AND	X1	
4	LD	X3	
	AND	M3	
	ORB		
5	LD	Y1	
	AND	X4	
6	LD	T0	
	AND	M3	
	ORB		
7	ANB		
8	OUT	Y1	
	TMR	T0	K10

Symbole von Eingangssignalen mit einer steigenden Flanke (wenn ein Signal von 0 auf 1 geschaltet wird) und mit einer fallenden Flanke (wenn ein Signal von 1 auf 0 geschaltet wird) werden im Folgenden erläutert:



Ansteigende Flanke Kontakt Fallender Flankenkontakt

Abb. 18. Flankenfilterung

Die logischen Blockbefehle ANB und ORB entsprechen keinen bestimmten Bedingungen im Relais-Kontakt-Schaltkreis, sondern beschreiben die Beziehung zwischen den Blöcken. Der ANB-Befehl führt die Operation **AND-Verknüpfung** mit den Ausführungsbedingungen zweier Logikblöcke durch.

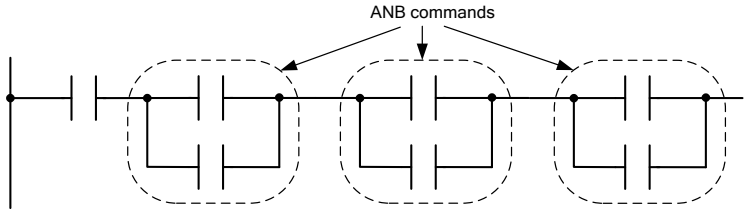


Abb. 19. ANB-Anweisung

Der ORB-Befehl führt eine **OR-Verknüpfung** mit den Ausführungsbedingungen zweier Logikblöcke durch.

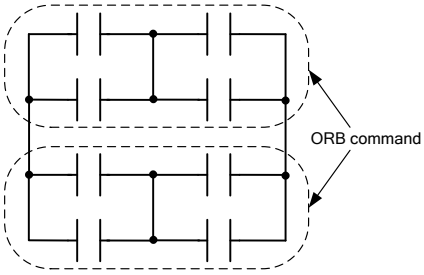


Abb. 20. ORB-Anweisung

3.5. Relaiskontakt-Schaltpläne (LD) in mnemonischen Code (IL) umwandeln

Die folgende Abbildung zeigt ein Programm, das in Form von Relaiskontaktsymbolen (LD) und einer Anweisungsliste - mnemonischer Code (IL) - dargestellt ist. Die Abbildung zeigt die Reihenfolge der Umwandlung des Kontaktplans (LD) in den vom Controller ausgeführten Code (IL).

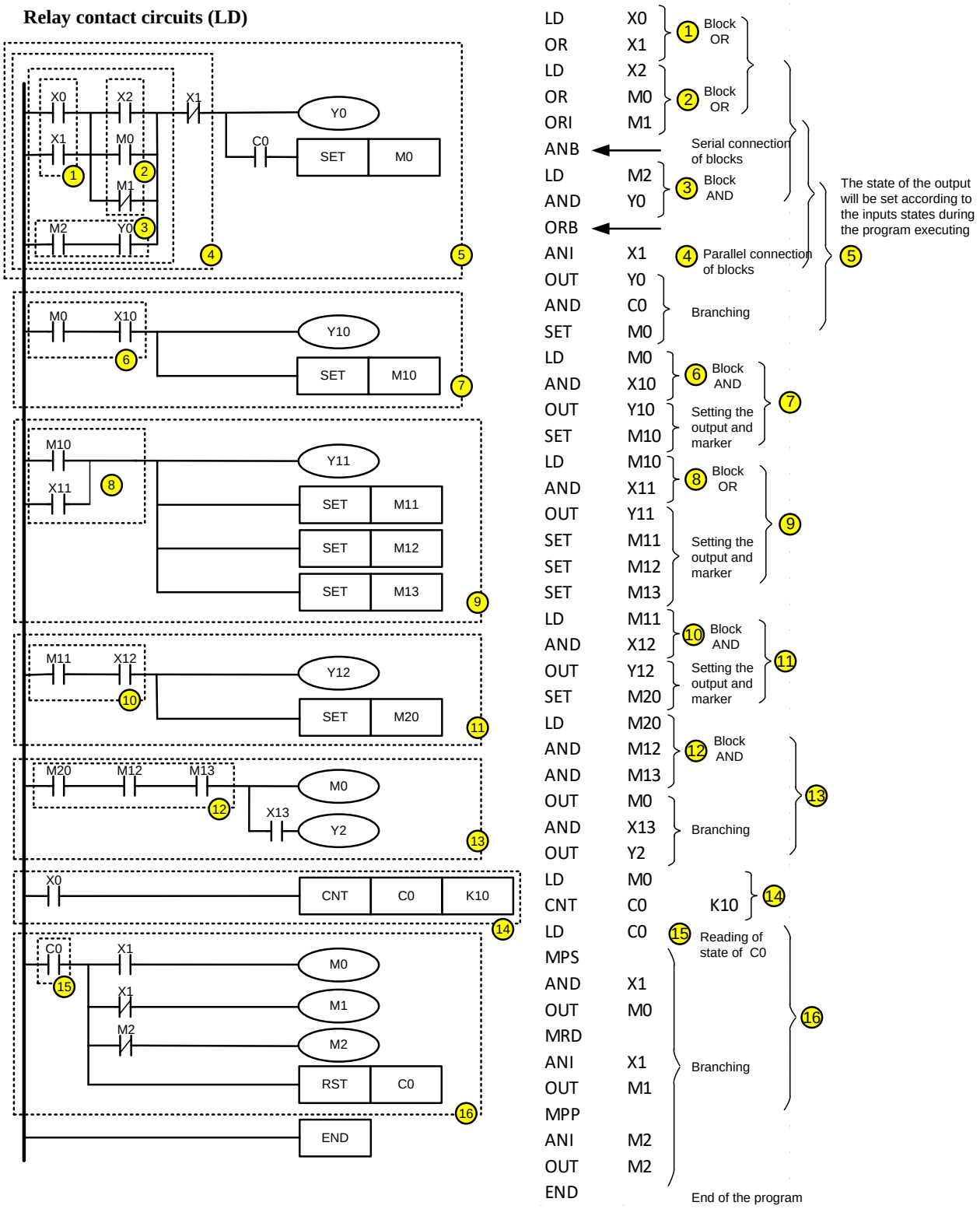


Abb. 21. Konvertierung von LD in IL

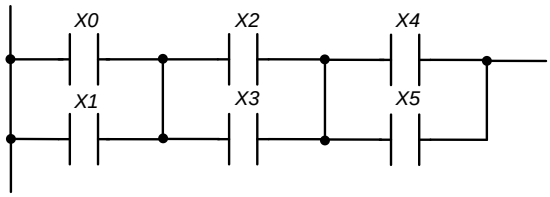
Die Verarbeitung des Relais-Kontakt-Schaltkreises beginnt in der oberen linken Ecke und endet in der unteren rechten Ecke. Es kann jedoch Ausnahmen und verschiedene Möglichkeiten der Umwandlung in mnemonischen Code geben, wie in den folgenden Beispielen gezeigt:

Beispiel 1

Der unten stehende Kontaktplan kann auf zwei verschiedene Arten in eine Anweisungsliste umgewandelt werden, das Ergebnis ist jedoch identisch (Abb. 22).

Die erste Codierungsmethode ist am besten geeignet, da die Anzahl der Logikblöcke unbegrenzt ist.

Die zweite Methode ist durch die maximale Anzahl von Logikblöcken begrenzt (maximale Anzahl von Blöcken ist 8).

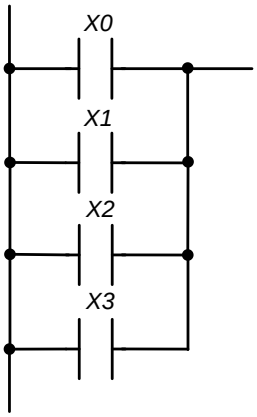


Methode 1		Methode 2	
LD	X0	LD	X0
OR	X1	OR	X1
LD	X2	LD	X2
OR	X3	OR	X3
ANB		LD	X4
LD	X4	OR	X5
OR	X5	ANB	
ANB		ANB	

Abb. 22. Verschiedene Methoden der Verwendung von ANB-Anweisungen

Beispiel 2

Unterschiedliche Codierungsmethoden für parallel geschaltete Kontakte sind unten dargestellt (Abb. 23).



Methode 1		Methode 2	
LD	X0	LD	X0
OR	X1	LD	X1
OR	X2	LD	X2
OR	X3	LD	X3
		ORB	
		ORB	
		ORB	

Abb. 23. Verschiedene Methoden der Verwendung von ORB-Anweisungen

Die erste Methode der Umwandlung eines Kontaktplans in eine Anweisungsliste ist aus Sicht der Nutzung des Controller-RAMs am besten geeignet.

4. Controller-Funktionalität

4.1. Operandenübersicht

Typ	Operand			Adressbereich		Funktion
Relais (1-Bit-Speicher)	X	Externe Eingangsrelais	Physische Eingänge	X0...X7	Max. 128 Punkte	Controllereingänge
			Virtuelle Eingänge (Modbus-Spule)	X10...X177		
	Y	Externe Ausgangsrelais	Physische Ausgänge	Y0...Y7	Max. 128 Punkte	Controllerausgänge
			Virtuelle Ausgänge (Modbus-Digitaleingänge)	Y10...Y177		
	M	Interne Relais (Merker)	Allgemeiner Zweck	M0...M99, M111...M127	Max. 128 Punkte	Zwischenspeicher. Entspricht den Hilfsrelais in elektrischen Schaltkreisen.
			Sonderzweck	M100...M110		
	T	Zeitschaltuhren	Auflösung 100 ms	T0...T47 (T46, T47 – kumulativ)	Max. 64 Punkte	Verwendet als Kontakte (T), die schließen, wenn der entsprechende Timer seinen eingestellten Wert erreicht (TMR-Befehl)
			Auflösung 10 ms	T48...T63 (T62, T63 – kumulativ)		
	C	Zähler	Inkrementeller Universalzweck	C0...C63	Max. 66 Punkte	Verwendung als Kontakte (C), die schließen, wenn der entsprechende Zähler seinen eingestellten
			Externe Impulse	C64, C65		

						Wert erreicht (CNT-Befehl)
Register (16-Bit-Speicher)	T	Aktueller Timerwert		64 Punkte (T0...T63)		Register zur Speicherung der aktuellen Timerwerte
	C	Aktueller Zählerstand		66 – 32-Bit-Zähler		Register zur Speicherung der aktuellen Zählerwerte
	D	Datenregister	Allgemeiner Zweck	D0...D319, D385...D391	Max. 384 Punkte	Dient zum Speichern von Daten. Spezielle Register konfigurieren den Controller und zeigen seinen Status an.
			Nichtflüchtig(1)	D320...D351		
			Spezial	D352...D384		
	A	Indexregister	Nebenwerte 16 Bit	A0...A7	Max. 16 Punkte	Kann zur Indexanzeige verwendet werden
	B		Wichtigste 16 Bits	B0...B7		
Zeiger	P	Zeiger für die Anweisungen CALL, CJ		32 Punkte (P0...P31)		Bezeichnungen für Anweisungen von Übergängen und Unterprogrammen
	I	Unterbrechungen	Kommunikation	I0	Max. 15 Punkte	Bezeichnungen für das Unterprogramm zur Verarbeitung von Interrupts
			Zeitgesteuert	I1...I100 (Max. 4 Punkte)		
			Extern	I1000...I1007		
			Treibers	I2000, I2001		
Konstanten	K	Dezimalkonstanten		K-32768 ...K32767 (16-Bit-Funktionen) K-2147483648 ...K2147483647 (32-Bit-Funktionen)		
	H	Hexadezimalkonstanten		H0000...HFFFF (16-Bit-Funktionen)		

			H00000000...HFFFFFFFF (32-Bit-Funktionen)
	F	Gleitkommakonstante	F±1,175494351 E-38... 3,402823466 E+38 (Nur 32-Bit-Funktionen)

(1) – Die Datenspeicherung erfolgt über die interne Stromversorgung CR2032.

4.2. Adressierung und Funktionen der Eingänge [X] und Ausgänge [Y]

Die Ein- und Ausgänge im Anwenderprogramm werden durch Operanden dargestellt. Durch Angabe der Adresse des Operanden kann während der Programmierung auf die physischen und virtuellen Ein- und Ausgänge des Controllers Bezug genommen werden.

Diskrete Ein-/Ausgänge werden im Oktalsystem adressiert, d. h. die Zahlen 8 und 9 werden nicht für Ein- und Ausgänge verwendet.

Funktion der Eingangsrelais X

Eingangsrelais X lesen den Zustand externer physischer Geräte (Taster, Schalter, Relaiskontakte usw.), die direkt an die Eingangsklemmen des Controllers angeschlossen sind. Jeder Eingang X kann im Programm beliebig oft verwendet werden.

Funktion der Ausgangsrelais Y

Ausgangsrelais Y steuern den Zustand der physischen Ausgangskontakte des Controllers und damit die Lastgeräte (Lampen, Relaisspulen usw.), die direkt an die Ausgangsklemmen des Controllers angeschlossen sind.

Jeder Ausgang Y kann im Programm beliebig oft verwendet werden, es wird jedoch empfohlen, die Ausgangsspule Y im Programm nicht mehr als einmal zu verwenden, da bei mehrmaliger Verwendung der Spule Y der Ausgangszustand durch das letzte Y im Scan bestimmt wird.

Der Zustand der E/A-Signale kann im Programm durch verschiedene Anweisungen gelesen werden.

Der Prozess der Behandlung von E/A-Signalen im Controller:

Eingänge:

1. Der Controller liest den Zustand der externen Eingabegeräte und speichert ihn am Anfang jedes Scan-Zyklus.
2. Änderungen des Eingangszustands während des Zyklus werden nicht übernommen, wenn der Eingangsimpuls sehr kurz ist (kürzer als die Zeit eines Scans).

Programm:

3. Der Controller führt das Programm ab Zeile 0 aus und speichert den Zustand aller Operanden im Speicher des Objekts.

Ausgänge:

4. Nach der Ausführung der END-Anweisung wird der Zustand des Ausgangsrelais Y in den Speicher der Ausgänge geschrieben und die Zustände der Ausgangskontakte werden geändert.

4.3. Adressierung und Funktion der internen Relais [M]

Um die binären Ergebnisse von logischen Verknüpfungen (Signalzustände "0" oder "1") zu speichern, wird innerhalb des Programms ein Zwischenspeicher (internes Relais) verwendet. Sie entsprechen den Zwischenrelais in Steuerungssystemen, die auf Relaislogik basieren.

Im Controller werden zwei Arten von internen Relais verwendet:

1. Allzweckrelais, die beim Ausschalten nicht gespeichert werden;
2. Spezialrelais, die dem Benutzer zusätzliche Funktionen bieten.

Interne Relais werden als Ausgänge programmiert. Sie können im Programm unbegrenzt oft verwendet werden. Die Adressierung der internen Relais erfolgt im Dezimalformat.

Verwendung von speziellen Merkern:

Merker	Funktion
M100...M107	Diese Hilfsrelais werden nur in Verbindung mit Interrupts der externen Eingänge I1000 ... I1007 verwendet. Der Wert der Merker entspricht dem Zustand des physikalischen Eingangs (IN0 ... IN7) zum Zeitpunkt der Interrupt-Bearbeitung (I1000 ... I1007). Die Werte X0 ... X7 werden erst zu Beginn des nächsten Zyklus des Anwenderprogramms aktualisiert. Zum Beispiel kann nach dem Eintritt in den Interrupt-Handler I1004 der Zustand des Eingangs IN4 durch Abfragen des Zustands von M104 (LD M104) ermittelt werden (der Wert von X4 ist in diesem Fall nicht relevant).

M108	Die steigende Flanke dieses Hilfsrelais zeigt den Abschluss der Initialisierung der Controller-Peripherie an. Während des nachfolgenden Betriebs behält der Merker einen hohen Pegelwert bei. Das Zurücksetzen des Merkers initialisiert den Controller neu. Zum Beispiel ist nach der Neudefinition der Ausgänge durch die PWM-Signalgeneratoren und der Eingänge durch die Impulszähler eine erneute Initialisierung erforderlich. In diesem Fall ist es notwendig, M108 zurückzusetzen.
M109	Das Setzen des Merkers aktiviert die Anzeige „ERR“ an der Vorderseite des Controllers, das Zurücksetzen deaktiviert sie.
M110	Das Setzen dieses Merkers und das anschließende Rücksetzen von M108 führt zu einem vollständigen Neustart des Controllers.

4.4. Adressierung und Funktion der Timer [T]

Einige Steuerprozesse erfordern ein Zeitrelais. Viele relaisgesteuerte Systeme verwenden Zeitrelais, die verzögert einschalten. Der Controller verwendet für diese Zwecke interne Speicherelemente, sogenannte Timer. Die Eigenschaften der Timer können im Programm festgelegt werden.

Die Adressierung der Timer erfolgt dezimal.

T	Zeitschaltuhren	Auflösung 100 ms	T0...T47 (T46, T47 – kumulativ)	Max. 64 Punkte
		Auflösung 10 ms	T48...T63 (T62, T63 – kumulativ)	

Die erforderliche Zeiteinstellung wird durch eine dezimale Konstante K bestimmt, die die Anzahl der gezählten Zeitschritte (diskret) angibt.

Beispiel: Ein Timer mit einer Auflösung von 100 ms, der als K5 eingestellt ist, hat einen tatsächlichen Einstellwert von 5 x 100 = 500 ms.

Der Timer arbeitet mit einer Einschaltverzögerung. Er wird mit dem Kontaktzustand = 1 aktiviert. Nach dem Zählen des eingestellten Zeitwerts setzt der Timer den entsprechenden Eingangskontakt T auf den Zustand „1“. Der

Timer kehrt in den Aus-Zustand zurück und setzt seinen aktuellen Wert zurück, wenn sein Eingangskontakt auf „0“ gesetzt wird.

Die Einstellung der Zeit kann auch indirekt mittels einer zuvor im Datenregister D gespeicherten Dezimalzahl erfolgen.

In den Controllern beginnt der Timer sofort mit dem Zählen, wenn der TMR-Befehl ausgeführt wird.

Erläuterung der Funktionsweise von zwei Timertypen:

Allzweck-Timer

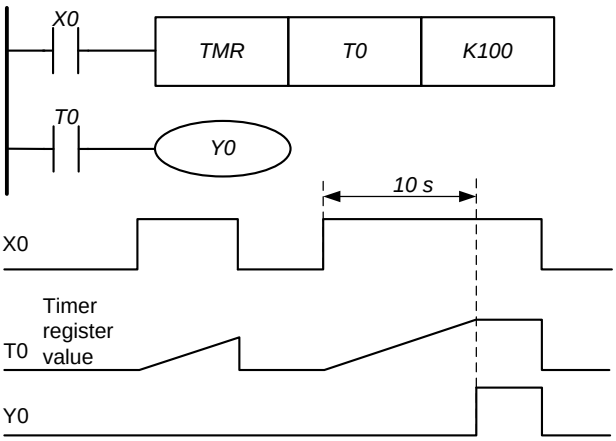


Abb. 24. Funktionsprinzip des Universaltimers

Wenn der Eingang X0 den Zustand "1" annimmt, beginnt die eingestellte Zeit zu zählen. Nach Ablauf der programmierten 10 Sekunden nimmt der Ausgang Y0 den Zustand "1" an. Der Timer schaltet sich aus und das Register T0 wird auf Null zurückgesetzt, sobald der Eingang X0 den Zustand "0" annimmt.

Akkumulations-Timer

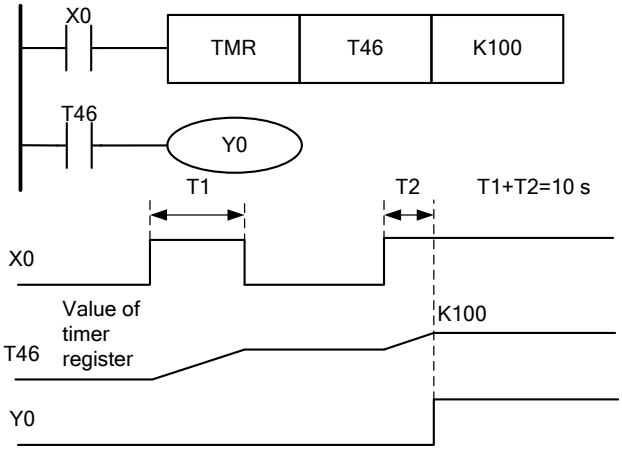


Abb. 25. Prinzip des akkumulierenden Timers

Zusätzlich zu den Universalzeitgebern verfügt der Controller über akkumulierende Zeitgeber, die nach dem Deaktivieren der steuernden logischen Verbindung den akkumulierten Zeitwert speichern.

4.5. Adressierung und Funktion der Zähler [C]

Bei einigen Steuerprozessen ist es notwendig, Impulse zu zählen (addieren oder subtrahieren). Viele relaisgesteuerte Systeme verwenden hierfür Impulszähler. Der Controller verwendet zwei Arten von internen Speicherelementen (Zähler).

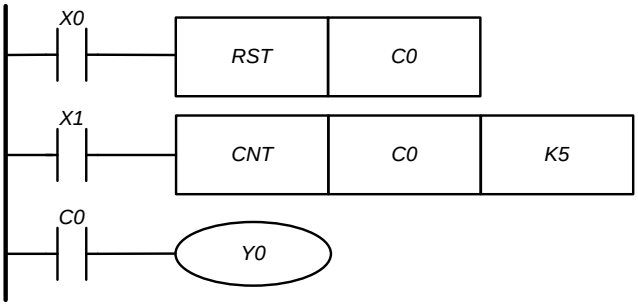
Die Adressierung der Timer erfolgt dezimal.

C	Zähler	Inkrementeller Universal-	C0...C63	Max. 66 Punkte
		Externe Impulse (Hardware)	C64, C65	

Funktion der Zähler:

Wenn das Eingangssignal des Zählers seinen Zustand von 0 auf 1 ändert, erhöht sich der aktuelle Wert des Zählers C um eins. Wenn er gleich dem eingestellten Wert (Sollwert) wird, schaltet der Arbeitskontakt des Zählers ein.

```
LD      X0
RST     C0
LD      X1
CNT     C0      K5
LD      C0
```



OUT Y0

Der Zähler wird zurückgesetzt, wenn X0=1 ist: der aktuelle Wert des Registers C0 = 0, der Kontakt C0 ist geöffnet.

Nach dem Umschalten von X1 von 0 auf 1 erhöht sich der Wert von C) um eins.

Wenn der Registerwert C0 = 5 ist, sind die Kontakte C0 und Y0 geschlossen und alle nachfolgenden Impulse am Eingang werden nicht gezählt.

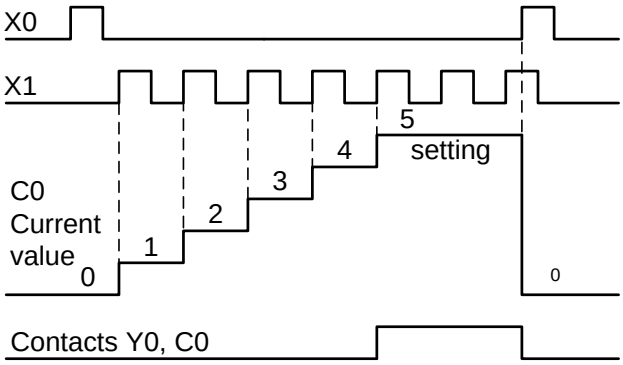
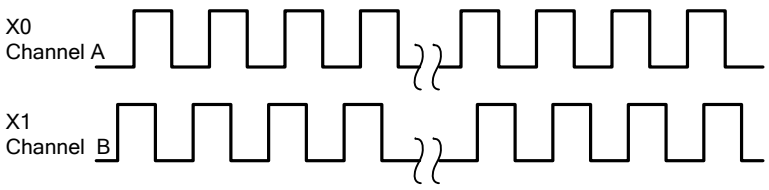


Abb. 26. Zählerfunktionsprinzip

Allzweckzähler zählen nicht über einen Schwellenwert hinaus, im Gegensatz zu Hardwarezählern, die nicht auf das Eingangssignal, sondern nur auf den physikalischen Zustand des diskreten Eingangs zählen, auf den sie sich beziehen. Abhängig vom Wert des Konfigurationsregisters D355 können die Zähler folgendermaßen konfiguriert werden:

Wert des Registers D355	Konfiguration
0	Der Standardwert. X0 und X1 (IN0 und IN1) arbeiten als diskrete Eingänge.
1	Der diskrete Eingang X0 bezieht sich auf den Zähler C64, welcher die steigenden Flanken von Impulsen zählt. X1 arbeitet als ein diskreter Eingang.
2	Der diskrete Eingang X0 bezieht sich auf den Zähler C64, welcher die fallenden Flanken von Impulsen zählt. X1 arbeitet als ein diskreter Eingang.
3	Der diskrete Eingang X0 bezieht sich auf den Zähler C64, der sowohl steigende als auch fallende Flanken von Impulsen zählt. X1 arbeitet als ein diskreter Eingang.

4	X0 arbeitet als ein diskreter Eingang. Der diskrete Eingang X1 bezieht sich auf den Zähler C65, welcher die steigenden Flanken von Impulsen zählt.
5	X0 arbeitet als ein diskreter Eingang. Der diskrete Eingang X1 bezieht sich auf den Zähler C65, welcher die fallenden Flanken von Impulsen zählt.
6	Die diskreten Eingänge X0 und X1 beziehen sich auf die Zähler C64 bzw. C65. Die Zähler zählen die steigenden Flanken der Impulse.
7	Die diskreten Eingänge X0 und X1 beziehen sich auf die Zähler C64 bzw. C65. C64 zählt fallende Flanken von Impulsen, C65 zählt steigende Flanken von Impulsen.
8	Die diskreten Eingänge X0 und X1 beziehen sich auf die Zähler C64 bzw. C65. C64 zählt sowohl steigende als auch fallende Flanken von Impulsen, C65 zählt steigende Flanken von Impulsen.
9	Die diskreten Eingänge X0 und X1 beziehen sich auf die Zähler C64 und C65. C64 zählt steigende Flanken von Impulsen, C65 zählt fallende Flanken von Impulsen.
10	Die diskreten Eingänge X0 und X1 beziehen sich auf die Zähler C64 bzw. C65. Die Zähler zählen die fallenden Flanken der Impulse.
11	Die diskreten Eingänge X0 und X1 beziehen sich auf die Zähler C64 bzw. C65. C64 zählt sowohl steigende als auch fallende Flanken von Impulsen, C65 zählt fallende Flanken von Impulsen.
12	Die diskreten Eingänge X0 und X1 beziehen sich auf den Zähler C64 und arbeiten als Encoder. Ein Quadratursignal wird an die Eingänge angelegt. 

4.6. Adressierung und Funktion der Register [D], [A], [B]

Datenregister [D]

Register stellen den Datenspeicher innerhalb des Controllers dar. Die Register können numerische Werte und binäre Informationen nacheinander speichern.

Die Daten werden in einem 16-Bit-Register (D0 usw.) gespeichert, das eine Zahl von -32768 bis +32767 speichern kann. Die Verbindung zweier 16-Bit-Register ergibt ein 32-Bit-"Doppelregister" (D0, D1 usw.), das eine Zahl von -2147483648 bis +2147483647 speichern kann.

Die Adressierung der Datenregister erfolgt dezimal. Bei Doppelregistern (32-Bit) beginnt die Adressierung mit dem unteren 16-Bit-Register.

D	Datenregister	Allgemeiner Zweck	D0...D319, D385...D391	Max. 384 Punkte
		Nichtflüchtig	D320...D351	
		Spezial	D352...D384	

Es gibt folgende Datenregistertypen:

Allzweck-Datenregister:

Diese Register werden während der Ausführung des Benutzerprogramms verwendet, die Daten werden beim Ausschalten nicht gespeichert.

Nichtflüchtige Datenregister:

Die Daten in diesen Registern werden beim Ausschalten im Speicher des Controllers gespeichert. Die Speicherstromversorgung erfolgt über eine interne Quelle, CR2032.

Indexregister:

Dieses Register wird verwendet, um Zwischenergebnisse zu speichern und Operanden anzuzeigen.

Spezialregister:

Diese Register dienen zur Konfiguration des Controllers und zum Zugriff auf spezielle Funktionen. Die Nummern der Spezialregister sind in der folgenden Tabelle aufgeführt:

Register	Funktion	Werte
D352	Das Register enthält Daten zur Position des Potentiometers „0“ auf der Vorderseite des Controllers.	0...4095
D353	Das Register enthält Daten zur Position des Potentiometers „1“ auf der Vorderseite des Controllers.	0...4095
D354	Das Register enthält Daten zur Position des Potentiometers „2“, „Geschwindigkeit“.	0...4095
D355	Das Register konfiguriert die Eingabetypen IN0 und IN1. Weitere Informationen finden Sie in Abschnitt 4.5.	0...12

<table>
<tr>
<td>D356</td>
<td>Das Register konfiguriert die Typen der Ausgänge OUT6 und OUT7 für den PWM-Befehl (siehe 7. Anwendungsbefehle, PWM-Befehl).
<table>
<tr><th rowspan="2">Wert</th><th rowspan="2">Diskretisierungszeit, µs</th><th colspan="2">Funktion des Ausgangs</th></tr>
<tr><th>OUT6</th><th>OUT7</th></tr>
<tr><td>0</td><td>–</td><td>Ausgang</td><td>Ausgang</td></tr>
<tr><td>1</td><td>100</td><td>PWM-Generator</td><td>Ausgang</td></tr>
<tr><td>2</td><td>10</td><td>PWM-Generator</td><td>Ausgang</td></tr>
<tr><td>3</td><td>100</td><td>Ausgang</td><td>PWM-Generator</td></tr>
<tr><td>4</td><td>10</td><td>Ausgang</td><td>PWM-Generator</td></tr>
<tr><td>5</td><td>100</td><td>PWM-Generator</td><td>PWM-Generator</td></tr>
<tr><td>6</td><td>10</td><td>PWM-Generator</td><td>PWM-Generator</td></tr>
</table>
Siehe Abschnitt 7. Anwendungshinweise (PWM-Anweisung) für detaillierte Informationen zur Erzeugung von PWM-Signalen.</td>
<td>0...6</td>
</tr>
<tr>
<td>D357...D384</td>
<td>Modus-Einstellungs- und Statusregister des Schrittmotortreibers. Siehe Abschnitt 8. «Anweisungen zur Steuerung des Schrittmotortreibers» für weitere Details.</td>
<td>–</td>
</tr>
</table>

4.7. Indexregister [A], [B]

Indexregister werden verwendet, um Operandenadressen zu indizieren und konstante Werte zu ändern.

Die Indexregister sind 16-Bit-Register.

Bei 32-Bit-Befehlen werden die Indexregister A und B in Kombination verwendet. A enthält 16 niederwertige Bits und B enthält 16 höherwertige Bits. Das Indexregister A wird als Zieladresse verwendet.

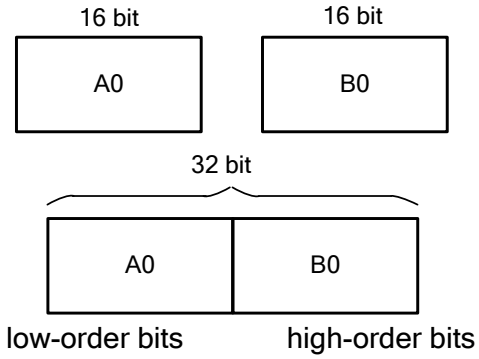


Abb. 27. Struktur des Indexregisters

Beispiel für eine Datenübertragung vom Datenregister D5A0 zum Datenregister D10B0:

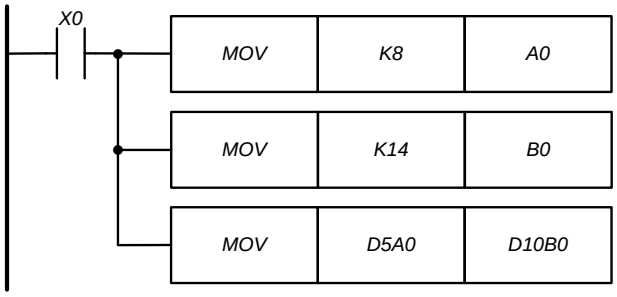


Abb. 28. Datenübertragung mit Indexregistern

Wenn X0 = 1: A0 = 8, B = 14

- Adresse der Transferquelle ist D5A0 = 5 + 8 = D13
- Zieladresse ist D10B0 = 10 + 14 = 24.
- Auf diese Weise werden Daten vom Register D13 in das Datenregister D24 übertragen

Indexregister können für Datenübertragungs- und Vergleichsoperationen in Verbindung mit Byte-Operanden und Bit-Operanden verwendet werden.

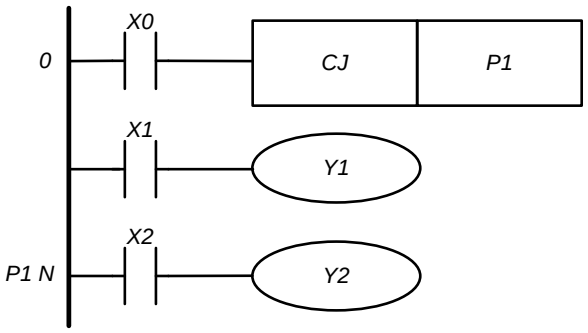
Es ist auch möglich, Konstanten auf die gleiche Weise zu indizieren. Bei der Indizierung von Konstanten muss das Symbol „@“ verwendet werden. Zum Beispiel: MOV K10 @ A0 D0B0.

4.8. Zeiger [P], [I].

P	Anweisungszeiger CALL, CJ		32 Punkte (P0...P31)		Markierungen oder Beschriftungen für Übergangsbefehle oder Unterprogrammaufrufe
I	Unterbrechungen	Kommunikation	I0	Max. 15 Punkte	Bezeichnungen für das Unterprogramm zur Verarbeitung von Interrupts
		Zeitgesteuert	I1...I100 (Max. 4 Punkte)		
		Extern	I1000...I1007		
		Treiber	I2000, I2001		

Zeiger (P) werden in Kombination mit den Anweisungen CJ (Sprünge) oder CALL (Unterprogramme) verwendet. Diese Zeiger sind Adressen von Speicherorten oder Unterprogrammen, die markiert wurden.

Ein Beispiel für die Ausführung eines CJ-Sprungbefehls:



Wenn X0 = 1 ist, springt das Programm nach der Ausführung von Zeile 0 sofort zur Zeile mit dem Zeiger P1, und die dazwischen liegenden Zeilen werden nicht ausgeführt.

Wenn X0 = 0 ist, wird das Programm Schritt für Schritt normal ausgeführt.

Abb. 29. Implementierung des CJ-Befehls

Ein Beispiel für die Verwendung von Unterprogrammen:

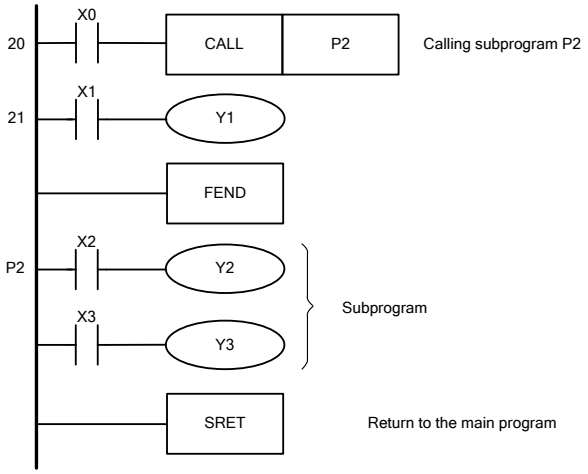


Abb. 30. Implementierung des CALL-Befehls

Wenn X0 = 1 in Zeile 20 ist, springt die Programmausführung direkt zur mit P2 markierten Zeile, das Unterprogramm wird ausgeführt, und nach dem SRET-Befehl kehrt die Programmausführung zu Zeile 21 zurück.

Interrupt-Zeiger (I) werden mit den Anweisungen EI, DI und IRET verwendet, um die Ausführung des Hauptprogramms zu unterbrechen. Es gibt die folgenden Arten von Interrupts:

1. Kommunikations-Interrupt: Wenn der Controller einen Broadcast-Frame über das Modbus-Protokoll empfängt, geht er sofort (unabhängig vom Scan-Zyklus) zum Interrupt-Bearbeitungsunterprogramm, das mit dem Zeiger I0 markiert ist. Er kehrt zum Hauptprogramm zurück, nachdem der IRET-Befehl ausgeführt wurde.
2. Zeitgesteuerte Unterbrechung: Das Unterbrechungs-Unterprogramm wird automatisch in festgelegten Zeitintervallen von 10 bis 1000 ms in Schritten von 10 ms ausgeführt. Insgesamt sind bis zu 4 zeitgesteuerte Unterbrechungen möglich. Beispielsweise werden Unterbrechungen mit den Zeigern I10, I50, I80 und I100 einmal pro 100 ms, 500 ms, 800 ms bzw. 1 s ausgeführt. Nach dem Befehl IRET wird zum Hauptprogramm zurückgekehrt.
3. Externe Unterbrechungen: Wenn das Signal am Eingang IN0 ... IN7 von 0 auf 1 oder von 1 auf 0 wechselt, wechselt die Steuerung sofort zur Ausführung des Unterbrechungs-Unterprogramms mit dem entsprechenden Zeiger I (IN0 → I1000, IN1 → I1001 usw.). Die Rückkehr zum Hauptprogramm erfolgt nach Ausführung des Befehls IRET.

4. Treiberunterbrechung: Wenn während der Kommutierung der Motorphasen ein Fehler auftritt, der durch das spezielle Register D381 (ERROR_CODE, weitere Details in Abschnitt 8. “Anweisungen für die Schrittmotortreiber-Steuerung”) dargestellt wird, wechselt die Steuerung zur Ausführung des Unterprogramms mit dem Zeiger I2000.Wenn sich der Status des Schrittmotortreibers ändert, Register D371 (MOTOR_STATUS, weitere Details finden Sie in Abschnitt 8. “Anweisungen für die Schrittmotortreiber-Steuerung”), wechselt die Steuerung zur Ausführung des Unterprogramms mit dem Zeiger I2001.Die Steuerung kehrt nach Ausführung des Befehls IRET zum Hauptprogramm zurück.

5. Fehlercodes

Wenn die LED "ERR" nach dem Laden und Ausführen des Benutzerprogramms leuchtet, bedeutet dies, dass das Benutzerprogramm einen Fehler enthält: einen Grammatikfehler oder einen falschen Operandenfehler. Jeder im Controller auftretende Fehler wird in einem speziellen Register aufgezeichnet (Schrittnummer und Fehlercode werden aufgezeichnet). Diese Informationen können mit einem PC oder einer SPS gelesen werden. Die folgende Tabelle enthält eine Liste der Fehlercodes und Beschreibungen.

Adresse	Typ	Größe	Beschreibung
E004	Eingangsregister	16-Bit	Fehlercode während der Ausführung eines Benutzerprogramms.
E084	Eingangsregister	16-Bit	Zeile des Anwenderprogramms, in der der Fehler erkannt wurde.
E004	Spulen		Fehlerflagge während der Ausführung des Benutzerprogramms.

Fehlercode	Beschreibung
2012h	Unbekannter Befehl
1007h	Interner Fehler, die Art der Signalkollision wurde nicht identifiziert.
1005h	Interner Fehler, Signaltyp im Falle einer Pegelkollision ist nicht identifiziert.
1006h	Interner Fehler, Signaltyp bei invertiertem Pegelkollision wird nicht erkannt.
2002h	LD-Anweisung, Stack-Überlauf.
2001h	Der Typ des Hauptsignals wurde nicht identifiziert.
2000h	Verarbeitung von LD-Befehlen, der Befehlscode hat sich geändert.
1001h	Interner Fehler, unbekannter Typ einer einzelnen Kollision.
1000h	Interner Fehler, einzelne Kollision beim aktuellen Wert. Der Signaltyp des Operanden ist unbekannt.

Fehlercode	Beschreibung
200Bh	Verarbeitung des AND-Befehls beim Entfernen des Signals vom Ausgabestapel. Unbekannter Kollisionstyp.
1004h	Gruppenkollisions-interner Fehler. Unbekannter Kollisionstyp.
1002h	Gruppenkollisions-interner Fehler. Der Typ des Operandensignals für Pegelkollisionen ist nicht definiert.
1003h	Gruppenkollisions-interner Fehler. Der Operandsignaltyp für Kollision mit invertiertem Pegel ist nicht definiert.
200Ch	Verarbeitung des AND-Befehls, der Befehlscode hat sich geändert.
200Dh	Verarbeitung des OR-Typ-Befehls, der Befehlscode hat sich geändert.
2010h	Wenn der ANB-Befehl angewendet wird, sind keine Einträge im Hauptstapel vorhanden.
200Fh	Fehler beim Anwenden der ANB-Anweisung. Es gibt keine Einträge im Ausgabestapel und nur einen Eintrag im Hauptstapel.
200Eh	Unbekanntes Signal im Ausgabestapel mit dem ANB-Befehl.
2011h	Das Fehlen von mindestens zwei Elementen im Hauptstapel für die Anwendung des ORB-Befehls.
2013h	Brunching Stack-Überlauf, Befehl MPS.
2016h	Verzweigungstack ist leer, Anweisungen MRD, MPP.
2015h	Hauptstapelüberlauf, Anweisungen MRD, MPP.
2014h	Der Signaltyp wird bei der Zuweisung des Selektors, der Anweisungen MRD und MPP nicht erkannt.
2017h	Stapelüberlauf, Anweisung NEXT.
3012h	Vorschau – Index P ist außerhalb des gültigen Bereichs.
3014h	Vorschau – Index I ist außerhalb des gültigen Bereichs.
3013h	Vorschau. Es konnte keine neue zeitgesteuerte Unterbrechung erstellt werden, da die maximale Anzahl überschritten wurde.
201Dh	Falscher Operandentyp, Anweisungen CJ/CJP.

Fehlercode	Beschreibung
201Ch	Operand außerhalb des gültigen Bereichs, Anweisungen CJ/CJP.
2024h	Falscher Operandentyp, Anweisungen CALL/CALLP.
2023h	Operand außerhalb des gültigen Bereichs, Anweisungen CALL/CALLP.
2025h	Es gibt keine Rücksprungpunkte im Stack, Befehl SRET.
2004h	Während der Interrupt-Bearbeitung wurde ein END/FEND-Befehl empfangen.
202Ah	Im Hauptprogramm wurde ein IRET-Befehl empfangen.
2056h	Anweisung ENDE, der Hauptstapel ist nicht leer.
2057h	Anweisung ENDE, Verzweigung-Stack ist nicht leer.
2058h	Anweisung ENDE, der Zyklen-Stack ist nicht leer.
2059h	Anweisung ENDE, der Unterprogramm-Stack ist nicht leer.
2026h	Anweisung IRET, der Hauptstapel ist nicht leer.
2027h	Anweisung IRET, Verzweigungsstapel ist nicht leer.
2028h	Anweisung IRET, der Zyklusstapel ist nicht leer.
2029h	Anweisung IRET, Unterprogrammstapel ist nicht leer.
2020h	Anweisung CALL/CALLP, unbekannter Index-Operand.
201Eh	Anweisung CALL/CALLP, Index-Operand A ist außerhalb des gültigen Bereichs.
201Fh	Anweisung CALL/CALLP, Index-Operand B ist außerhalb des gültigen Bereichs.
201Ah	Anweisung CJ/CJP, unbekannter Index-Operand.
2018h	Anweisung CJ/CJP, Index-Operand A ist außerhalb des zulässigen Bereichs.
2019h	Anweisung CJ/CJP, Index-Operand B ist außerhalb des gültigen Bereichs.
201Bh	Anweisung CJ/CJP, der angeforderte Zeiger existiert nicht.
2022h	Anweisung CALL/CALLP, die angeforderte Marke existiert nicht.
2021h	Anweisung CALL/CALLP, Stapelüberlauf.

Fehlercode	Beschreibung
2003h	Falscher Operand, Anweisung OUT.
200Ah	Falscher Operand, Anweisung SET/RST.
2005h	Der Befehlssatz kann nicht auf den Operanden C angewendet werden.
2006h	Der Befehlssatz kann nicht auf den Operanden T angewendet werden.
2007h	Der Befehlssatz kann nicht auf den Operanden D angewendet werden.
2008h	Der Befehlssatz kann nicht auf den Operanden A angewendet werden.
2009h	Der Befehlssatz kann nicht auf den Operanden B angewendet werden.
202Dh	Anweisung INV, unbekannter Signaltyp.
202Bh	Im Befehl TMR ist das erste Argument untypisch.
202Ch	Bei der Anweisung CNT ist das erste Argument untypisch.
202Eh	Anweisung INC/DEC, fehlerhafter Operand.
2037h	Anweisung ADD/SUB/MUL/DIV/WAND/WOR/WXOR, Typ des dritten Operanden ist falsch.
2038h	Anweisung NEG/ABS, falscher Operandentyp.
2030h	Anweisung CMP, Typ des dritten Operanden ist falsch.
2031h	Anweisung ZCP, Typ des dritten Operanden ist falsch.
202Fh	Anweisung MOV/BMOV/FMOV, falscher Typ des Zieloperanden.
2039h	Anweisung XCH, der Datentyp des 1stenOperanden ist falsch.
203Ah	Anweisung XCH, der Datentyp des zweiten Operanden ist falsch. .
203Bh	Befehl ROR/ROL, der Datentyp des 1.^{sten}Operanden ist falsch.
2033h	Anweisung ZRST, Operanden sind nicht vom gleichen Typ.
2032h	Anweisung ZRST, Operandentyp ist falsch.
2036h	Anweisung DIV, Division durch Null einer Ganzzahl.
2046h	Anweisung DECO, Typ des zweiten Operanden ist falsch.
2047h	Anweisung ENCO, Typ des zweiten Operanden ist falsch.
2048h	Anweisung SUM, Typ des zweiten Operanden ist falsch.

Fehlercode	Beschreibung
2049h	Anweisung BON, Typ des zweiten Operanden ist falsch.
204Bh	Anweisung SQR, Typ des 2d-Operanden ist falsch.
204Ah	Anweisung SQR, negativer Wert.
204Ch	Anweisung POW, Typ des dritten Operanden ist falsch.
203Ch	Anweisung FLT, Typ des zweiten Operanden ist falsch.
203Dh	Anweisung INT, Typ des 2d-Operanden ist falsch.
203Eh	Bei der Anweisung PWM ist der dritte Operand für die PWM-Signalausgabe nicht anwendbar.
203Fh	Anweisung PWM, Typ des dritten Operanden ist falsch.
2041h	Anweisung DECMP, Typ des 1. Operanden ist falsch.
2040h	Anweisung DECMP, Typ des zweiten Operanden ist falsch.
2042h	Anweisung DECMP, Typ des dritten Operanden ist falsch.
2045h	Anweisung DEZCP, Typ des dritten Operanden ist falsch.
2044h	Anweisung DEZCP, Typ des 1. Operanden ist falsch.
2043h	Anweisung DEZCP, Typ des zweiten Operanden ist falsch.
2050h	Anweisung DEADD/DESUB/DEMUL/DEDIV/DEPOW, Typ des dritten Operanden ist falsch.
204Fh	Anweisung DEADD/DESUB/DEMUL/DEDIV/DEPOW, Typ des 1. Operanden ist falsch.
204Eh	Anweisung DEADD/DESUB/DEMUL/DEDIV/DEPOW, Typ des zweiten Operanden ist falsch.
204Dh	Anweisung DEDIV, Division durch Null.
3015h	Vorschaufehler, unbekannter Befehl erkannt.
2053h	Anweisung DESQR, Typ des 1. Operanden ist falsch.
2052h	Anweisung DESQR, Typ des 2D-Operanden ist falsch.
2051h	Anweisung DESQR negativer Wert.

Fehlercode	Beschreibung
2035h	Befehl LD#, Stapelüberlauf.
2034h	Anweisung LD#, Hauptsignaltyp nicht erkannt.
4000h	Überlauf des Warteschlangenpuffers für Schalterunterbrechungen.
4001h	Überlauf der Warteschlange für zeitgesteuerte Interrupts TIM0.
4002h	Überlauf der Warteschlange für zeitgesteuerte Interrupts TIM1.
4003h	Überlauf der zeitgesteuerten Interrupt-Warteschlange von TIM2.
4004h	Überlauf der Warteschlange für zeitgesteuerte Interrupts TIM3.
4005h	Überlauf der Warteschlange für externe Interrupts IN0.
4006h	Überlauf der Warteschlange für externe Interrupts IN1.
4007h	Überlauf der Warteschlange für externe Interrupts IN2.
4008h	Überlauf der Warteschlange für externe Interrupts IN3.
4009h	Überlauf des externen Interrupt-Warteschlangenpuffers IN4.
400Ah	Überlauf des externen Interrupt-Warteschlangenpuffers IN5.
400Bh	Überlauf des externen Interrupt-Warteschlangenpuffers IN6.
400Ch	Überlauf der Warteschlange für externe Interrupts IN7.
400Dh	Überlauf des Warteschlangenpuffers für Treiberunterbrechungen.
400Eh	Überlauf des Warteschlangenpuffers für Motorstatusänderungs-Interrupts.
2054h	Anweisung TRD, falscher Operandentyp.
2055h	Anweisung TWR, falscher Operandentyp.
3000h	Stapel sind leer, kein Signalwert.
3001h	Der Hauptstapel ist leer, kein Signalwert.
3003h	Der Wert des Indexregisters liegt außerhalb des zulässigen Bereichs.
3004h	Index des Operanden X liegt außerhalb des gültigen Bereichs.
3005h	Index des Operanden Y liegt außerhalb des gültigen Bereichs.
3006h	Index des Operanden M liegt außerhalb des gültigen Bereichs.

Fehlercode	Beschreibung
3007h	Index des Operanden C liegt außerhalb des gültigen Bereichs.
3008h	Index des Operanden T liegt außerhalb des gültigen Bereichs.
3009h	Index des Operanden A/B liegt außerhalb des gültigen Bereichs.
300Ah	Index des Operanden D liegt außerhalb des gültigen Bereichs.
300Bh	Index des Operanden P liegt außerhalb des gültigen Bereichs.
300Ch	Index des Operanden I liegt außerhalb des gültigen Bereichs.
300Dh	Unbekannter Operandentyp
300Fh	Der Wert des Operanden kann nicht abgerufen werden.
300Eh	Die Gleitkommazahl (FLOAT) wird mit einem 16-Bit-Befehl verwendet.
3010h	Operandenwert abrufen - falscher Operandentyp.
3011h	Abrufen des Tokens des Operanden - falscher Operandentyp.
5000h	+5V-Stromversorgung - Kurzschluss
205Ah	Anweisung TWR, falsches Zeitformat.
205Bh	MOD-Anweisung, Division durch Null.
205Ch	DWR-Befehl, ungültiges Datumsformat.
205Dh	Kontakttyp-Anweisungsstapel übergelaufen (LD#*)
205Eh	Kontakttyp-Anweisungsstapel übergelaufen (AND#*)
205Fh	Kontakttyp-Anweisungsstapel übergelaufen (ODER#*)

6. Grundlegende Anweisungen

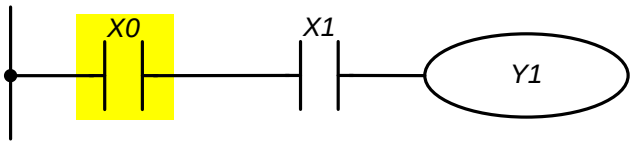
Anleitung	Funktion
LD	Schließer

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

Der Befehl LD wird als Schließer für die Programmierung des Starts von logischen Ketten verwendet. Er befindet sich links im Kontaktschema und ist direkt mit der Stromversorgungsleitung verbunden.

Verwendung:




```
LD    X0
AND   X1
OUT   Y1
```

Der Befehl LD X0 „Schließer X0“ startet die Ablaufkette. Wenn an den Eingängen X0 und X1 gleichzeitig ein Signal „1“ anliegt, wird der Ausgang Y1 auf den Zustand „1“ gesetzt.

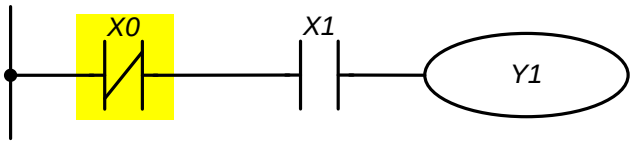
Anleitung	Funktion
LDI	Schließer

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

Der Befehl LDI wird als Öffner für die Programmierung des Starts von logischen Ketten verwendet. Er befindet sich links im Kontaktschema und ist direkt mit der Stromversorgungsleitung verbunden.

Verwendung:




```
LDI   X0
AND   X1
OUT   Y1
```

Der Befehl LDI X0 "Schließer X0" startet die sequentielle Logikverbindung. Wenn an den Eingängen X0 und X1 gleichzeitig ein Signal „1“ anliegt, wird der Ausgang Y1 auf den Zustand „1“ gesetzt.

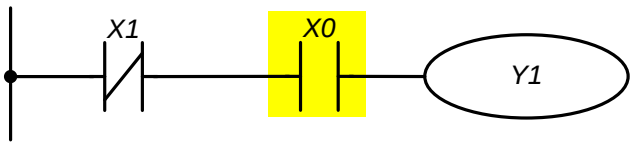
Anleitung	Funktion
AND	Reihenschaltung - Öffnerkontakt (logisches AND)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

Die Anweisung AND wird als in Reihe geschalteter Öffner für die Programmierung der logischen Multiplikationsoperation (AND) verwendet. Die Anweisung stellt eine logische Operation dar und kann daher nicht am Anfang der Sequenz programmiert werden. Für Befehle zum Sequenzbeginn muss LD oder LDI verwendet werden.

Verwendung:




```
LDI   X1
AND   X0
OUT   Y1
```

Der Befehl AND X0 "Serienverbindung - Schließer X0" erzeugt eine serielle logische Verbindung mit Kontakt X1 und wird verwendet, um die logische Multiplikationsoperation durchzuführen. Wenn am Eingang X1 "0" und am Eingang X0 "1" anliegt, wechselt der Ausgang Y1 in den Zustand "1".

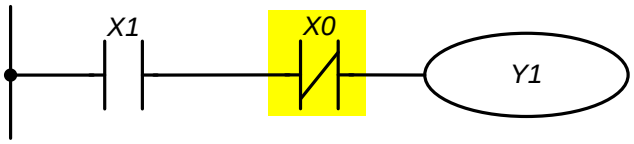
Anleitung	Funktion
ANI	Reihenschaltung - Öffner (logische NAND-Verknüpfung)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

Die Anweisung ANI wird als ein in Reihe geschalteter Öffner für die Programmierung der logischen Operation NAND (AND NOT) verwendet. Die Anweisung stellt eine logische Operation dar und kann daher nicht am Anfang der Sequenz programmiert werden. Für Befehle zum Starten einer Sequenz muss LD oder LDI verwendet werden.

Verwendung:




```
LD    X1
ANI   X0
OUT   Y1
```

Der Befehl "Serienverbindung - Öffner X0" erzeugt eine serielle logische Verbindung mit Kontakt X1 und wird verwendet, um die logische Operation NAND durchzuführen. Wenn am Eingang X1 "1" und am Eingang X0 "0" anliegt, wechselt der Ausgang Y1 in den Zustand "1".

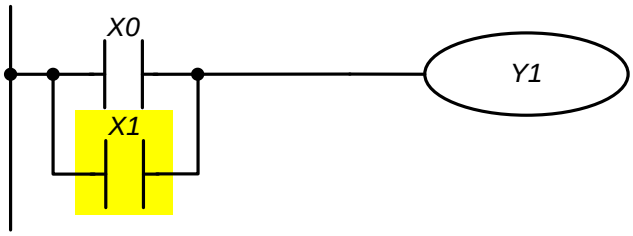
Anleitung	Funktion
OR	Parallelschaltung – Öffner (logisches OR)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

Der Befehl OR wird als parallel geschalteter Öffner für die Programmierung der logischen Addition (OR) verwendet. Die Anweisung stellt eine logische Operation dar und kann daher nicht am Anfang der Sequenz programmiert werden. Für Befehle zum Starten einer Sequenz muss LD oder LDI verwendet werden.

Verwendung:




```
LD    X0
OR    X1
OUT   Y1
```

Der Befehl "Parallelverbindung - Schließer X1" erzeugt eine parallele logische Verbindung mit Kontakt X0 und wird verwendet, um die Operation der logischen Addition durchzuführen. Wenn mindestens einer der Eingänge X0 oder X1 "1" ist, wechselt der Ausgang Y1 in den Zustand "1".

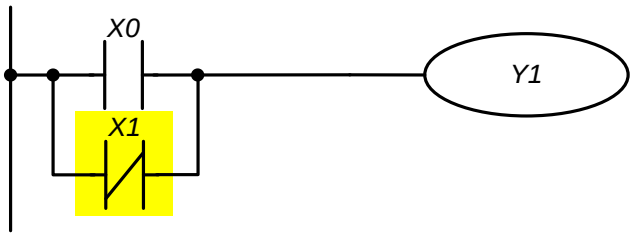
Anleitung	Funktion
ORI	Parallelschaltung – Öffnerkontakt (logische NOR-Verknüpfung)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

Der Befehl ORI wird als parallel geschalteter Schließer für die Programmierung der logischen Operation NOR (OR NOT) verwendet. Die Anweisung stellt eine logische Operation dar und kann daher nicht am Anfang der Sequenz programmiert werden. Für Befehle zum Starten einer Sequenz muss LD oder LDI verwendet werden.

Verwendung:




```
LD    X0
ORI   X1
OUT   Y1
```

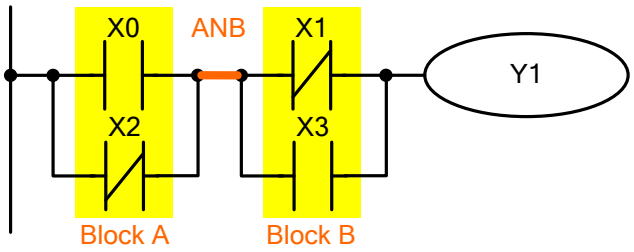
Der Befehl "Parallelverbindung - Öffner X1" erzeugt eine parallele logische Verbindung mit Kontakt X0 und wird verwendet, um die Operation der logischen Instruktion NOR (OR NOT) durchzuführen. Wenn der Eingang X0 "1" oder der Eingang X1 "0" ist (eine oder beide Bedingungen gleichzeitig), wechselt der Ausgang Y1 in den Zustand "1".

Anleitung	Funktion
ANB	«AND»-Block: Reihenschaltung von Blöcken

Beschreibung:

- Der Befehl ANB wird für die Reihenschaltung von zwei logischen Ketten (Blöcken) verwendet. Separate Blöcke von parallel geschalteten Elementen werden separat in das Programm eingegeben. Um diese Blöcke in Reihe zu schalten, wird nach jedem Block ein ANB-Befehl programmiert.
- Startverzweigung programmiert mit LD- oder LDI-Befehlen.
- Der ANB-Befehl ist unabhängig und benötigt keine Operanden.
- Der ANB-Befehl kann innerhalb des gesamten Benutzerprogramms unbegrenzt oft verwendet werden.
- Der ANB-Befehl wird als Reihenschaltung in einem Kontaktplan dargestellt. Der ANB-Befehl in einer Liste von IL-Sprachbefehlen kann in einer Kontaktschaltung als Brücke dargestellt werden.
- Müssen mehrere separate Blöcke nacheinander verbunden werden, so ist die Anzahl der LD/LDI-Anweisungen sowie der ANB-Anweisungen auf 8 zu begrenzen.

Verwendung:




```
LD    X0
ORI   X2
LDI   X1
OR    X3
ANB
OUT   Y1
```

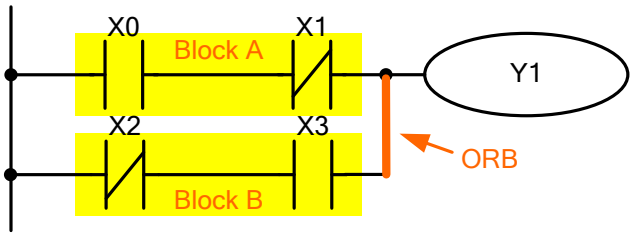
Die ANB-Anweisung erstellt eine Reihe von logischen Verbindungen zwischen zwei Logikblöcken (Block A und Block B).

Anleitung	Funktion
ORB	«OR»-Block: Parallelschaltung von Blöcken

Beschreibung:

- Die ORB-Anweisung dient zur Parallelschaltung von zwei oder mehr in Reihe geschalteten Kontakten oder Blöcken. Werden mehrere in Reihe geschaltete Blöcke parallel geschaltet, muss nach jedem Block eine ORB-Anweisung hinzugefügt werden.
- Der Verzweigungsbeginn wird mit den Anweisungen LD oder LDI programmiert.
- Die ORB-Anweisung ist eigenständig und benötigt keine Operanden.
- Die ORB-Anweisung kann innerhalb des Anwenderprogramms unbegrenzt oft verwendet werden.
- Werden mehrere separate Blöcke direkt nacheinander programmiert, ist die Anzahl der LD- und LDI-Anweisungen sowie der ORB-Anweisungen auf 8 zu begrenzen.
- Die ORB-Anweisung wird im Kontaktplan als Parallelschaltung dargestellt. Die ORB-Anweisung in einer Liste von AWL-Anweisungen kann in einem Kontaktplan als Brücke dargestellt werden.

Verwendung:




```
LD    X0
ANI   X1
LDI   X2
AND   X3
OR
OUT   Y1
```

Die ORB-Anweisung erstellt eine Reihe von logischen Verbindungen zwischen zwei Logikblöcken (Block A und Block B).

Anleitung	Funktion
MPS	Offset im Stack nach unten

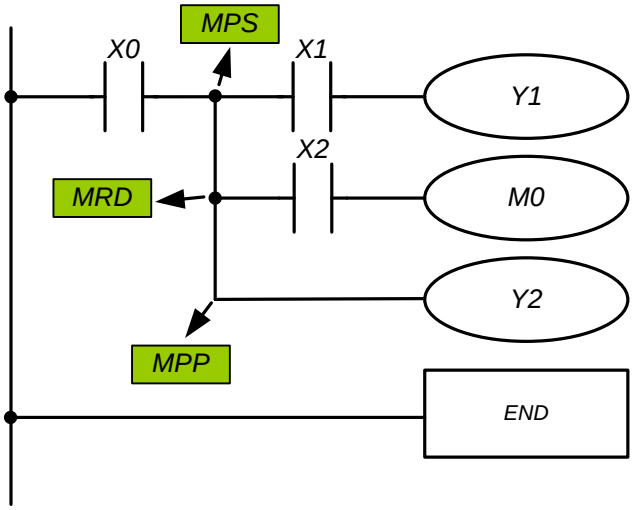
Anleitung	Funktion
MRD	Den Wert vom Stapel lesen

Anleitung	Funktion
MPP	Verlassen des Stacks

Beschreibung:

- Die Anweisungen MPS, MRD und MPP dienen zum Erstellen von Ebenen logischer Verbindungen (z. B. nach einem anfänglichen logischen Ausdruck mehrere logische Ausdrücke am Ausgang erstellen, d. h. mehrere Ausgangsspulen einschalten).
- Mit der MPS-Anweisung wird das vorherige Ergebnis logischer Verbindungen (Verarbeitung eines logischen Ausdrucks) gespeichert.
- Mit der MRD-Anweisung können mehrere unabhängige Verzweigungen zwischen dem Anfang (MPS) und dem Ende (MPP) der Verzweigung erstellt werden. Das Ergebnis der Verarbeitung eines logischen Ausdrucks am MPS-Punkt wird in jedem Zweig berücksichtigt.
- Der letzte Zweig wird durch die MPP-Anweisung erstellt.
- Die mit der MPS-Anweisung geöffnete Verzweigung muss immer mit der MPP-Anweisung geschlossen werden.
- Die Anweisungen MPS, MRD und MPP benötigen keine Operanden.
- Diese Anweisungen werden im Kontaktplan nicht angezeigt. Bei der Programmierung in einem Kontaktplan werden die Verzweigungen wie üblich verwendet. Bei der Konvertierung eines Anwenderprogramms von einem Kontaktplan (KOP) in eine Anweisungsliste (AWL) sollten MPS-, MRD- und MPP-Anweisungen zur AWL hinzugefügt werden.

Verwendung:




```
LD    X0
MPS
AND   X1
OUT   Y1
MPD
AND   X2
OUT   M0
MPP
OUT   Y2
END
```

MPS

Ein Zwischenergebnis (X0-Wert) auf der 1. Ebene der logischen Verbindungen wird an erster Stelle im Stack-Speicher der Zwischenverbindungen aufgeführt. Die logische Multiplikation von X1 mit X0 wird durchgeführt und der Ausgang Y1 wird gesetzt.

MRD

Vor der Ausführung des nächsten Befehls wird ein Zwischenergebnis am 1. Platz des Speichers der logischen Verknüpfungen gelesen. Eine logische AND-Verknüpfung von X2 mit X0 wird durchgeführt, und der Ausgang von M0 wird gesetzt.

MPP

Vor der Ausführung des nächsten Befehls wird ein Zwischenergebnis am 1. Platz des Speichers der logischen Verknüpfungen gelesen. Der Ausgang Y2 ist gesetzt. Die Operation auf der 1. Ebene der Zwischenergebnisse ist abgeschlossen, und der Speicher der logischen Verknüpfungen wird gelöscht.

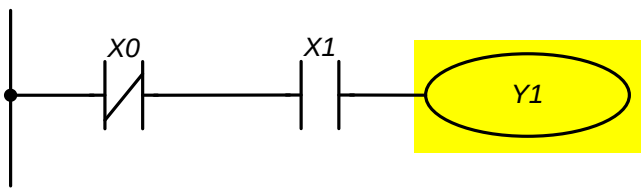
Anleitung	Funktion
OUT	Ausgangsspule

Operand	X	Y	M	T	C	A	B	D
		•	•					

Beschreibung:

- Die OUT-Anweisung dient zum Setzen einer Ausgangsspule abhängig vom Ergebnis der logischen Verbindungen (dem Ergebnis der Verarbeitung des logischen Ausdrucks durch die Steuerung).
- Mit der OUT-Anweisung ist es möglich, die Programmierung einer Zeile (logischer Ausdruck) zu beenden.
- Die Programmierung mehrerer OUT-Anweisungen als Ergebnis der Verarbeitung eines logischen Ausdrucks ist ebenfalls möglich.
- Das durch die OUT-Anweisung dargestellte Ergebnis logischer Verbindungen kann in den nächsten Programmschritten als Zustand des Eingangssignals verwendet werden, d. h. es kann in vielen logischen Ausdrücken mehrfach gelesen werden.
- Das durch die OUT-Anweisung dargestellte Ergebnis logischer Verbindungen ist aktiv (ein), solange die Bedingungen für sein Einschalten gültig sind.
- Bei der Programmierung der doppelten Aufzeichnung der gleichen Ausgänge (ihrer Adressen) können Probleme bei der Programmausführung auftreten. Vermeiden Sie die doppelte Aufzeichnung der Ausgänge, da dies zu Störungen beim Programmablauf führen kann.

Verwendung:




```
LDI   X0
AND   X1
OUT   Y1
```

Wenn X0 = 0 und X1 = 1 ist, setzt die Anweisung OUT Y1 den Zustand des Ausgangs Y1 = "1".

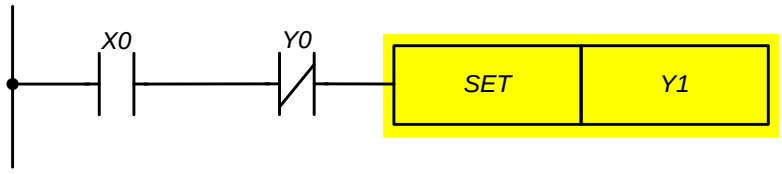
Anleitung	Funktion
SET	Aktivieren des verriegelten Ausgangs

Operand	X	Y	M	T	C	A	B	D
		•	•					

Beschreibung:

- Der Zustand des Operanden kann direkt mit der SET-Anweisung gesetzt werden.
- Die Operanden Y und M können mit der SET-Anweisung eingeschaltet werden.
- Sobald die Eingangsbedingung für die SET-Anweisung (Signal "1") erfüllt ist, schaltet sich der entsprechende Operand ein.
- Sind die Eingangsbedingungen für die SET-Anweisung nicht mehr erfüllt, bleibt der entsprechende Operand eingeschaltet.

Verwendung:




```
LD    X0
ANI   Y0
SET   Y1
```

Der Ausgang Y1 schaltet ein, wenn die Eingangsbedingungen (X0, Y0) erfüllt sind. Danach ist der Ausgang Y1 unabhängig von den Eingangsbedingungen. Die einzige Möglichkeit, den Ausgang Y1 auszuschalten, besteht darin, die RST-Anweisung zu verwenden oder die Stromversorgung der Steuerung auszuschalten.

Anleitung	Funktion
RST	Zurücksetzen des Operandenstatus

Operand	X	Y	M	T	C	A	B	D
		•	•	•	•	•	•	•

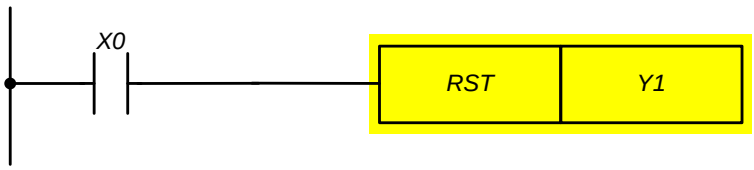
Beschreibung:

Der Zustand eines Operanden kann direkt zurückgesetzt werden.

Die RST-Anweisung schaltet entsprechende Operanden aus. Das bedeutet:

- Ausgänge Y, Kontakte M werden ausgeschaltet (Signalzustand "0").
- Aktuelle Werte von Timern und Zählern, Werte der Register D, A und B werden auf "0" zurückgesetzt.
- Sobald die Eingangsbedingung für den RST-Befehl (Signal "1") erfüllt ist, wird der entsprechende Operand ausgeschaltet.
- Wenn die Eingangsbedingungen für den RST-Befehl nicht mehr erfüllt sind, bleibt der entsprechende Operand ausgeschaltet.

Verwendung:




```
LD    X0
RST   Y1
```

Der Ausgang Y1 schaltet ab, wenn die Bedingung X1 erfüllt ist, und bleibt auch dann ausgeschaltet, wenn die Bedingung X0 nicht erfüllt ist.

Anleitung	Funktion
TMR	Zeitgeber (16-Bit)

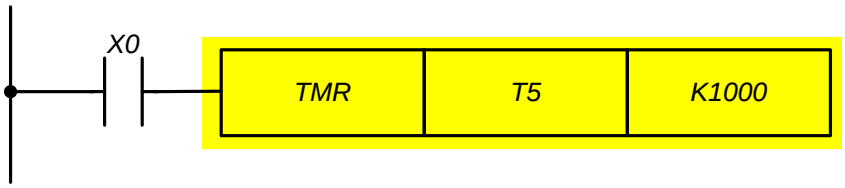
Operanden	K	H	F	X	Y	M	T	C	A	B	D
S1							•				
S2	•	•					•	•	•	•	•

Beschreibung:

- Der Befehl TMR wird verwendet, um einen Signalzustand (Ein/Aus) abhängig vom Ergebnis logischer Verknüpfungen nach einer im Befehl angegebenen Zeitspanne zu setzen.
- Mit dem TMR-Befehl ist es möglich, die Programmierung einer Zeile (logischer Ausdruck) zu beenden.
- Das Ergebnis der durch den TMR-Befehl dargestellten logischen Verknüpfungen kann in den nächsten Programmschritten als Zustand des Eingangssignals verwendet werden, d.h. es kann viele Male in vielen logischen Ausdrücken gelesen werden.

— Das Ergebnis der durch den TMR-Befehl dargestellten logischen Verknüpfungen ist aktiv (eingeschaltet), solange die Eingangsbedingungen gültig sind.

Verwendung:




```
LD    X0
TMR   T5   K1000
```

Bei der Bedingung X0 = 1 zählt der Befehl TMR T5, bis der Wert im Register T5 den Wert von K1000 (100 s) erreicht. Wenn X0 = 0 ist, wird die Ausführung des TMR-Befehls gestoppt und T5 wird auf "0" zurückgesetzt.

Anleitung		Funktion
CNT	S1 S2	Zähler (16-Bit)
DCNT		Zähler (32-Bit)

Operanden	K	H	F	X	Y	M	T	C	A	B	D
S1								•			
S2	•	•					•	•	•	•	•

Informationen

D Normalerweise wird zur Verwendung von 32-Bit-Anweisungen das Präfix „D“ dem Namen der Anweisung hinzugefügt. D – es existiert nur eine 32-Bit-Version der Anweisung.

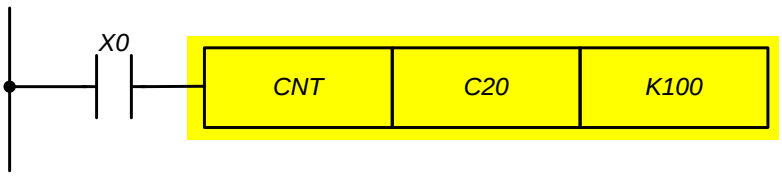
P Für Impulsanweisungen mit einer "Lebensdauer" von einem Scan wird das Postfix "**P**" hinzugefügt. Das Konzept eines *einzelnen Scans* sollte dem verwendeten Operanden zugeschrieben werden. Beispielsweise wurde der Operand **M0** in Zeile 7 des Hauptprogramms von "0" auf "1" gesetzt. Nun hat der Operand **M0** für alle folgenden Anweisungen vor FEND oder END eine Impulskomponente an der Vorderflanke (das Ergebnis für LDP M0 ist „1“), sowie für Anweisungen, die von der 0. bis zur 6. Zeile beginnen, während des nächsten Scans der Operand **M0** eine Impulskomponente hat. Beim Übergang zu Zeile 7 (oder

niedriger, wenn ein CJ-Befehl verwendet wird) hat **M0** einen hohen Signalpegel ohne Impulskomponenten. Somit wurde ein Kreis entlang des Programmkörpers ausgeführt - ein Scan, verschoben zur Operand-Änderungszeile. Im Falle von Unterbrechungen, die vor Erreichen von Zeile 7 auftreten, behält der Operand **M0** die Impulskomponente bis zur Rückkehr zum Hauptprogramm. Wenn der Operand **M0** in einer Unterbrechung oder einem Unterprogramm geändert wurde, wird als Ort, an dem der Operand geändert wird, die Zeile betrachtet, von der aus der Übergang zum Unterprogramm oder zur Hauptprogrammzeile durchgeführt wurde, bevor deren Bearbeitung der Interrupt-Handler aufgerufen wurde.

Beschreibung:

- Der Befehl CNT wird verwendet, um die Anzahl der Schließungen des Eingangskontakts zu summieren und den Signalzustand zuzuweisen (Ausgang einschalten), wenn der aktuelle Zählerwert den eingestellten Wert erreicht.
- Mit dem CNT-Befehl ist es möglich, die Programmierung einer Zeile (logischer Ausdruck) zu beenden.
- - Das Ergebnis der durch den CNT-Befehl dargestellten logischen Verknüpfungen kann in den nächsten Programmschritten als Zustand des Eingangssignals verwendet werden, d.h. es kann viele Male in vielen logischen Ausdrücken gelesen werden.
- Um den aktuellen Wert eines Zählers zurückzusetzen, verwenden Sie den RST-Befehl.
- **Achtung:** Hardwarezähler zählen über den Schwellenwert hinaus und arbeiten unabhängig vom Vorhandensein eines Eingangssignals

Verwendung:




```
LD    X0
CNT   C20   K100
```

Wenn X0 von „0“ auf „1“ wechselt, erhöht sich der Wert des Registers C20 um 1. Dies wiederholt sich, bis der Wert des Registers C20 K100 (100 Impulse)

erreicht. Danach stoppt der Zählvorgang und der Kontakt C20 schaltet ein. Um den Wert des Registers C20 zurückzusetzen, verwenden Sie den Befehl RST C20.

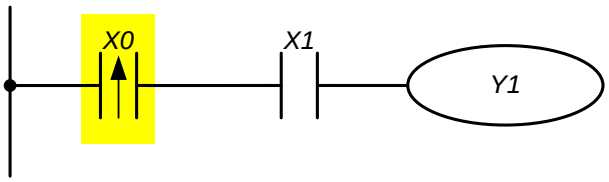
Anleitung	Funktion
LDP	Beginn der logischen Verknüpfung mit einer steigenden Flankenabfrage (Impuls)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

- Der Befehl LDP wird verwendet, um den Pulsstart einer logischen Verbindung zu programmieren.
- Der Befehl LDP muss am Anfang der Schaltung programmiert werden.
- Der LDP-Befehl wird auch in Verbindung mit den Befehlen ANB und ORB verwendet, um die Verzweigung zu starten.
- Der LDP-Befehl wird nach einer positiven Flanke für die Dauer des Programmzyklus (Scan) gespeichert.

Verwendung:




```
LDP   X0
AND   X1
OUT   Y1
```

Der Befehl "LDP X0" startet die serielle logische Verbindung. Wenn der Eingang X0 von "0" auf "1" wechselt (und X1 = 1), behält der Ausgang Y1 den Zustand "1" während eines Scans bei.

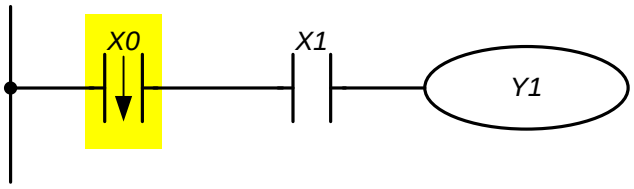
Anleitung	Funktion
LDF	Beginn einer logischen Verknüpfung mit einer fallenden Flankenabfrage (Impuls)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

- Der Befehl LDF wird verwendet, um den Pulsstart einer logischen Verbindung zu programmieren.
- Der Befehl LDF muss am Anfang der Schaltung programmiert werden.
- Der LDF-Befehl wird auch in Verbindung mit den Befehlen ANB und ORB verwendet, um die Verzweigung zu starten.
- Der LDF-Befehl wird nach einer negativen Flanke für die Dauer des Programmzyklus (Scan) gespeichert.

Verwendung:



LDF	X0	3
AND	X1	
OUT	Y1	

Der Befehl "LDF X0" startet die serielle logische Verbindung. Wenn der Eingang X0 von "1" auf "0" wechselt (und X1 = 1), behält der Ausgang Y1 den Zustand "1" während eines Scans bei.

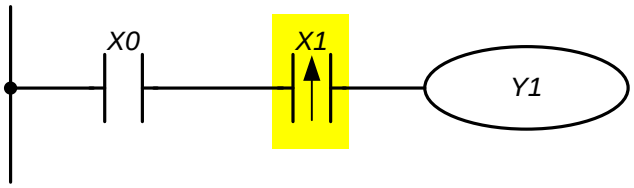
Anleitung	Funktion
ANDP	«AND» mit Flankenauswertung (Impuls)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

- Der Befehl ANDP wird zur Programmierung eines in Reihe geschalteten Pulskontakts mit positiver Flankenabfrage (Impuls) verwendet.

Verwendung:



LD	X0
ANDP	**X1**
OUT	Y1

Der Befehl "ANDP X1" erzeugt eine serielle logische Verbindung. Wenn der Eingang X1 von "0" auf "1" wechselt (und X0 = 1), behält der Ausgang Y1 den Zustand "1" während eines Scans bei.

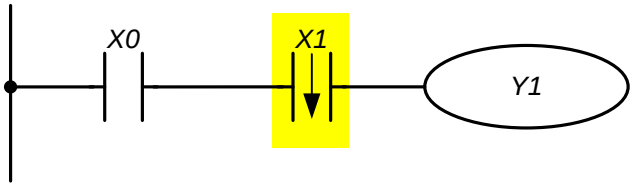
Anleitung	Funktion
ANDF	«AND» mit Abfrage bei fallender Flanke (Impuls)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

— Der Befehl ANDF wird zur Programmierung eines in Reihe geschalteten Pulskontakts mit negativer Flankenabfrage (Impuls) verwendet.

Verwendung:




```
LD    X0
ANDF  X1
OUT   Y1
```

Der Befehl "ANDF X1" erzeugt eine serielle logische Verbindung. Wenn der Eingang X1 von "1" auf "0" wechselt (und X0 = 1), behält der Ausgang Y1 den Zustand "1" während eines Scans bei.

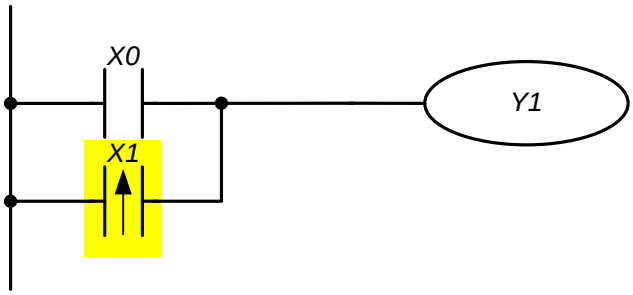
Anleitung	Funktion
ORP	«OR» mit Flankenauswertung (Impuls)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

— Der Befehl ORP wird zur Programmierung eines parallel geschalteten Pulskontakts mit positiver Flankenabfrage (Impuls) verwendet.

Verwendung:




```
LD    X0
ORP   X1
OUT   Y1
```

Der Befehl "ORP X1" erzeugt eine parallele logische Verbindung. Der Ausgang Y1 behält den Zustand "1" während eines Zyklus, wenn der Eingang X1 von "0" auf "1" wechselt oder X0 = 1 ist.

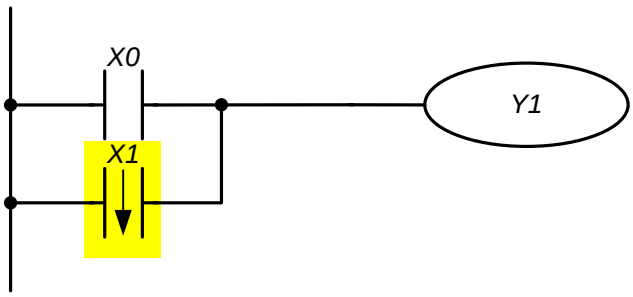
Anleitung	Funktion
ORF	«OR» mit Flankenabfrage (Impuls)

Operand	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Beschreibung:

— Der Befehl ORF wird zur Programmierung eines parallel geschalteten Impulskontakts mit Flankenabfrage (Impuls) verwendet.

Verwendung:




```
LD    X0
ORF   X1
OUT   Y1
```

Der Befehl "ORF X1" erzeugt eine parallele logische Verbindung. Der Ausgang Y1 behält den Zustand "1" während eines Zyklus, wenn der Eingang X1 von "1" auf "0" wechselt oder X0 = 1 ist.

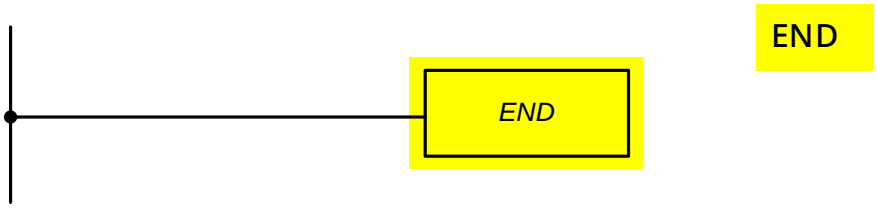
Anleitung	Funktion
END	Programmende

Beschreibung:

Das Ende eines Benutzerprogramms und der Übergang zum Programmanfang (Schritt 0).

- Jedes Steuerungsprogramm muss mit einem END-Befehl enden.
- Wenn ein END-Befehl programmiert wird, endet an dieser Stelle die Bearbeitung des Programms. Nachfolgende Programmbereiche werden nicht mehr berücksichtigt. Nach der Bearbeitung des END-Befehls werden die Ausgänge gesetzt und das Programm startet neu (Schritt 0).

Verwendung:



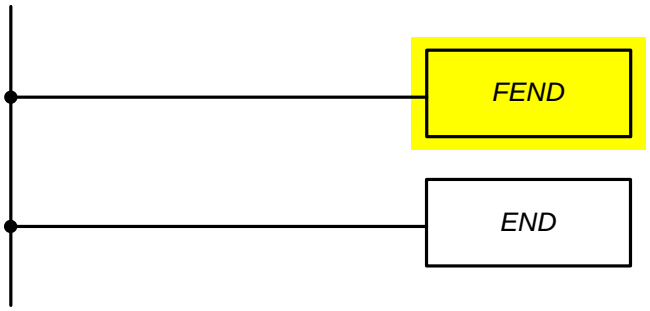
Anleitung	Funktion
FEND	Ende des Hauptprogramms

Beschreibung:

Das Ende des Hauptbenutzerprogramms und der Übergang zum Programmanfang (Schritt 0). Die Hauptunterschiede zum END-Befehl sind folgende:

- Die Bearbeitung endet nicht mit dem FEND-Befehl. Der Befehl FEND trennt das Hauptprogramm von Unterprogrammen und Interrupt-Handlern, die sich im Bereich zwischen den Befehlen FEND und END befinden und von P und SRET, I und IRET eingerahmt werden.
- Wenn in einem Benutzerprogramm keine Unterprogramme und Interrupts verwendet werden, ist der Befehl FEND nicht erforderlich.
- Der Befehl FEND kann nur einmal verwendet werden.

Verwendung:



FEND

END

Anleitung	Funktion
NOP	Leere Zeile im Programm

Beschreibung:

Eine leere Zeile ohne logische Funktionen kann später für beliebige Anweisungen verwendet werden, z. B. beim Assemblieren eines Programms oder zum Debuggen.

- Nach erfolgreichem Assemblieren eines Programms sollten NOP-Anweisungen gelöscht werden, da sie sonst die Zykluszeit des Programms unnötig verlängern.
- Die Anzahl der NOP-Anweisungen in einem Programm ist nicht begrenzt.

Verwendung:

```
LD    X0
NOP
OUT   Y0
```

NOP-Anweisungen werden in Kontaktplänen nicht angezeigt.

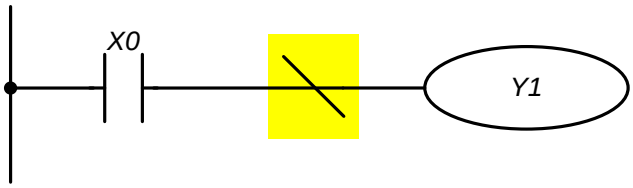
Anleitung	Funktion
INV	Invertierung - Ersetzen des Ergebnisses logischer Verknüpfungen durch das Gegenteil

Beschreibung:

- Der Befehl INV invertiert den Zustand des Ergebnissignals der davor platzierten Anweisungen.
- Das Ergebnis der logischen Verknüpfung „1“ vor dem INV-Befehl wird danach zu „0“.

— Das Ergebnis der logischen Verknüpfung „0“ vor dem INV-Befehl wird danach zu „1“.
— Der INV-Befehl kann als AND- oder ANI-Befehl angewendet werden.
— Der INV-Befehl kann verwendet werden, um das Ergebnissignal einer komplexen Schaltung umzukehren.
— Der INV-Befehl kann verwendet werden, um das Signalergebnis der Impulsanweisungen LDP, LDF, ANP usw. umzukehren.

Verwendung:




```
LD    X0
INV
OUT   Y1
```

Wenn der Eingang X0 = 0 ist, ist der Ausgang Y1 = 1. Wenn der Eingang X0 = 1 ist, ist der Ausgang Y1 = 0.

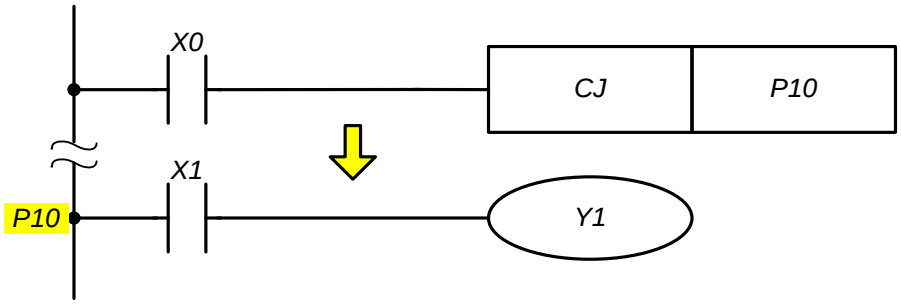
Anleitung	Funktion
P	Anspringen eines Sprungpunktes in einem Programm oder Unterprogramm

Operand	0...31

Beschreibung:

— Der P-Befehl wird verwendet, um einen Übergangspunkt für die Anweisungen CJ, CALL anzuzeigen.
— Die Punktnummer im Programm sollte nicht wiederholt werden.

Verwendung:




```
LD    X0
CJ    P10
.
.
P     10
LD    X1
OUT   Y1
```

Der Punkt P10 gibt die Übergangsadresse für die Ausführung des Befehls CJ P10 an.

Anleitung	Funktion
I	Adressierung eines Unterbrechungspunkts

Operand	0...100, 1000...1007, 2000, 2001

Beschreibung:

Der Befehl I wird verwendet, um den Übergangspunkt zum Interrupt-Handler anzuzeigen. Global werden Interrupts durch den Befehl EN aktiviert und durch den Befehl DS deaktiviert.

Insgesamt kann die Steuerung 15 Interrupts haben.

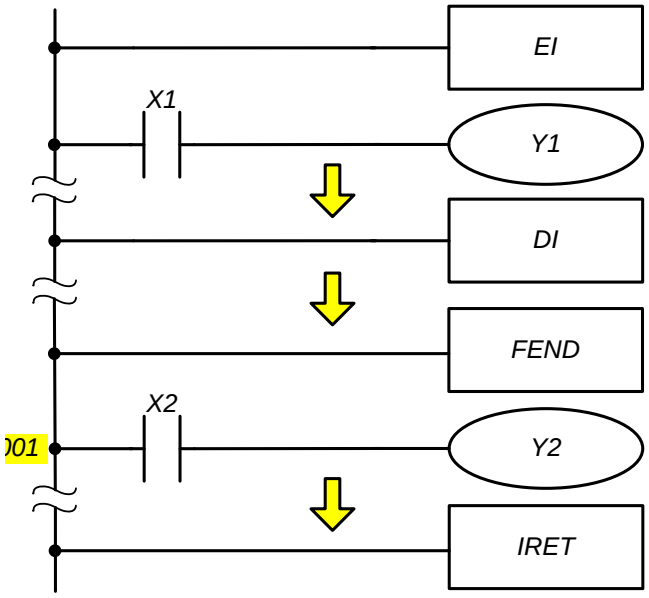
Ein Interrupt, der auftritt, wenn ein Modbus-Frame (Broadcast oder an die Steuerung adressiert) über RS-485 empfangen wird, wird als I0 markiert.

Es ist möglich, bis zu 4 zeitgesteuerte Interrupts in der Steuerung In zu organisieren, wobei n die Aufrufperiode des Interrupt-Handlers von 10 ms ist und einen Wert von 1 bis 100 haben kann. Also, $n = \frac{T}{10} ms$, wobei T eine gewünschte Periode des Aufrufs des Handlers ist (gemessen in ms).

8 externe Interrupts**I1000**...**I1007**entsprechen den diskreten Eingängen 0..7. Der Interrupt tritt auf, wenn sich der Pegel des Eingangssignals ändert.

2 Treiber-Interrupts: **I2000**– tritt im Fehlerfall auf und **I2001**– tritt auf, wenn sich der Motorstatus ändert (weitere Informationen finden Sie in Abschnitt 8. “Anweisungen zur Schrittmotortreibersteuerung”).

Verwendung:



EI		Aktivierung der Unterbrechungen
LD	X1	Öffner X1
OUT	Y1	Ausgang Y1
...		
DI		Unterbrechungssperre
...		
FEND		Ende des Hauptprogramms
I	**1001**	Einsprungpunkt für den Interrupt-Handler.
LD	X2	Öffner X2
OUT	Y2	Ausgang Y2
...		
IRET		Ende der Interrupt-Bearbeitungsroutine

7. Anwendungshinweise

Anleitung	Operanden	Zugehörige Varianten	Funktion
CJ	S	P	Bedingter Sprung

S	Zeiger P werden als Operanden verwendet. Die Operanden können indiziert werden (A, B)

Beschreibung:

Mit dem Befehl CJ kann ein Teil des Programms übersprungen werden. Durch die Anwendung dieses Befehls kann die Ausführungszeit des Programms reduziert werden. Beispielsweise das Überspringen eines Programmabschnitts, der für die Initialisierung der Peripheriegeräte der Steuerung, das Aktivieren von Interrupts usw. vorgesehen ist (weitere Informationen finden Sie in Abschnitt 4.8).

CALL	S	P	Aufruf eines Unterprogramms

S	Zeiger P werden als Operanden verwendet. Die Operanden können indiziert werden (A, B)

Beschreibung:

Der Befehl CALL wird verwendet, um ein Unterprogramm aufzurufen.

- Ein Unterprogramm ist mit den Punkten P markiert und kann durch einen CALL-Befehl aufgerufen werden.
- Der SRET-Befehl muss am Ende des Unterprogramms stehen.
- Ein Unterprogramm muss nach dem Befehl FEND und vor dem Befehl END platziert werden.
- Wenn der CALL-Befehl ausgeführt wird, springt die Steuerung zum markierten Punkt. Nach der Ausführung des SRET-Befehls kehrt die Steuerung zum Hauptprogramm zurück, zu dem Befehl, der auf CALL folgt.
- Die Punkte können mit einer unbegrenzten Anzahl von CALL-Befehlen verwendet werden.

— Unterprogramme können von anderen Unterprogrammen aufgerufen werden. Max. 8 Verschachtelungsebenen möglich.

SRET			Ende des Unterprogramms

Beschreibung:

Der Befehl SRET definiert das Ende eines Unterprogramms (weitere Details siehe Abschnitt 4.8).

— Jedes Unterprogramm muss mit dem SRET-Befehl abgeschlossen werden.
— Das Programm kehrt nach der Verarbeitung des SRET zu dem Befehl zurück, der dem CALL-Befehl folgt.
— Der SRET-Befehl kann nur zusammen mit dem CALL-Befehl verwendet werden.

Hinweis: Dieser Befehl benötigt keine Eingangsbedingung (Kontakte sind nicht erforderlich).

IRET			Ende der Interrupt-Bearbeitungsroutine

Beschreibung:

Der Befehl IRET definiert das Ende der Interrupt-Bearbeitung (weitere Details siehe Abschnitt 4.8).

Hinweis: Dieser Befehl benötigt keine Eingangsbedingung (Kontakte sind nicht erforderlich).

EI			Aktivierung der globalen Interrupts

Beschreibung:

Der Befehl EI aktiviert die Interrupt-Bearbeitung (weitere Details siehe Abschnitt 4.8).

Hinweis: Dieser Befehl benötigt keine Eingangsbedingung (Kontakte sind nicht erforderlich).

DI			Globale Interruptsperre

Beschreibung:
Der Befehl DI deaktiviert die Interrupt-Bearbeitung (weitere Details siehe Abschnitt 4.8).
Hinweis: Dieser Befehl benötigt keine Eingangsbedingung (Kontakte sind nicht erforderlich).

Aufruf eines Interrupt-Handler-Unterprogramms

- Bei der Bearbeitung einer Unterbrechung erfolgt ein Übergang vom Hauptprogramm zum Interrupt-Handler.
- Nachdem die Interrupt-Bearbeitung abgeschlossen ist, kehrt die Steuerung zum Hauptprogramm zurück.
- Der Beginn des Interrupt-Unterprogramms wird durch Setzen der Markierung (Interrupt-Punkt) festgelegt.
- Das Ende des Interrupt-Unterprogramms wird durch den IRET-Befehl bestimmt.
- Das Interrupt-Unterprogramm muss am Ende des Benutzerprogramms nach dem FEND-Befehl und vor dem END-Befehl programmiert werden.

Hinweis: Wenn keiner der beiden Befehle EI oder DI programmiert ist, ist der Interrupt-Modus nicht aktiviert, d.h. keines der Interrupt-Signale wird verarbeitet.

Ausführung eines Interrupt-Unterprogramms

Mehrere Interrupt-Unterprogramme, die nacheinander ablaufen, werden in der Reihenfolge ihres Aufrufs bearbeitet.
Wenn mehrere Interrupt-Unterprogramme gleichzeitig aufgerufen werden, wird das Interrupt-Programm mit der niedrigsten Punktadresse zuerst bearbeitet.

FOR	S		Schleifenanfang FÜR-NÄCHSTES

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•

Hinweis: Dieser Befehl benötigt keine Eingangsbedingung (Kontakte sind nicht erforderlich).

NEXT			Ende einer FOR-NEXT-Schleife

Hinweis: Dieser Befehl benötigt keine Eingangsbedingung (Kontakte sind nicht erforderlich).

Zyklen

Die Befehle FOR/NEXT werden zur Programmierung zyklischer Wiederholungen von Programmteilen (Programmschleife) verwendet.

Beschreibung:

- Der Programmteil zwischen den Befehlen FOR und NEXT wird "n" mal wiederholt. Nach Abschluss des FOR-Befehls fährt das Programm mit dem Programmschritt nach dem NEXT-Befehl fort.
- Der Wert "n" kann im Bereich von +1 bis +32 767 liegen. Wenn ein Wert aus dem Bereich von 0 bis -32 768 eingestellt wird, wird die Schleife FOR-NEXT nur einmal ausgeführt.
- Bis zu 8 Verschachtelungsebenen von FOR-NEXT-Schleifen sind möglich.
- Die Befehle FOR und NEXT können nur paarweise verwendet werden. Jedem FOR-Befehl muss ein NEXT-Befehl zugeordnet sein.

Fehlerquellen

In folgenden Fällen treten Fehler im Programm auf:

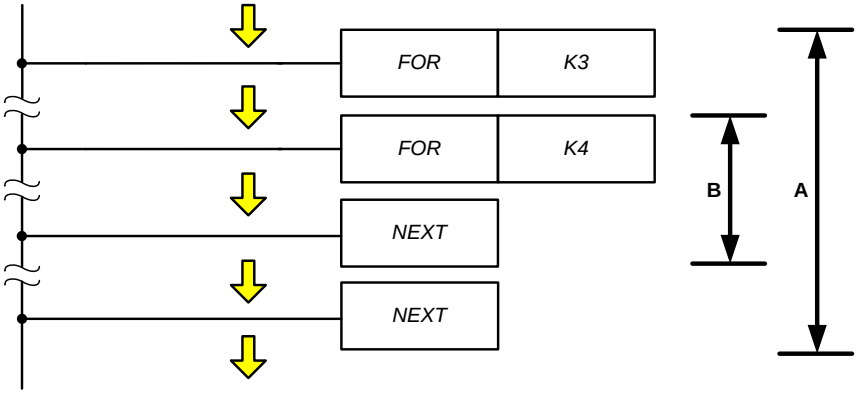
- Der nächste Befehl steht vor dem FOR-Befehl.
- Die Anzahl der NEXT-Befehle unterscheidet sich von der Anzahl der FOR-Befehle.
- Eine große Anzahl von Wiederholungen von "n" kann die Ausführungszeit eines Programms erheblich verlängern.

Ein Beispiel für die Verwendung von FOR/NEXT-Befehlen:

Das folgende Beispiel zeigt zwei FOR-NEXT-Schleifen, eine innerhalb der anderen.

Der Programmteil A wird 3-mal ausgeführt (K3 bedeutet Dezimalzahl 3).

Der Programmteil B wird innerhalb jeder Wiederholung von Teil A 4-mal ausgeführt (K4 bedeutet Dezimalzahl 4).



CMP	S1 S2 D	D P	Vergleich der numerischen Daten

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D					•	•					

Hinweis: Operand D belegt 3 Adressen.

Beschreibung:

Vergleich zweier numerischer Datenwerte (größer, kleiner, gleich).

— Die Daten in beiden Quellen (S1) und (S2) werden verglichen.

— Das Ergebnis des Vergleichs (größer, kleiner, gleich) wird durch Aktivieren des Relais M oder des Ausgangs Y angezeigt. Welcher der Kontakte am Zieloperanden (D) aktiv ist, wird durch das Vergleichsergebnis bestimmt:

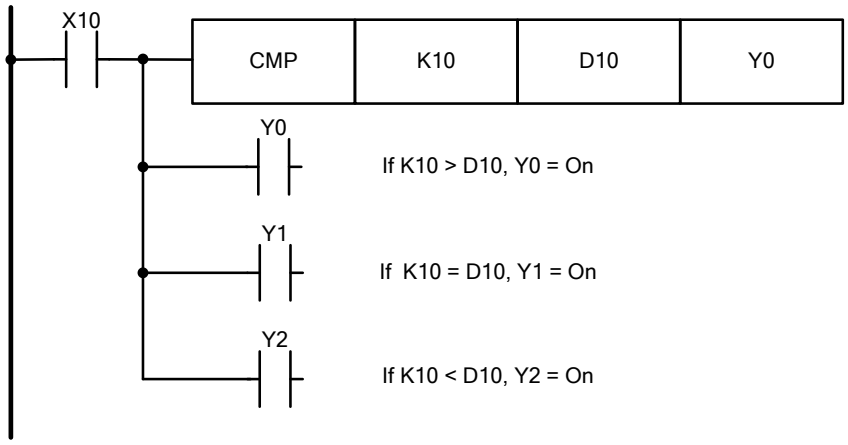
(S1) > (S2) → (D)

(S1) = (S2) → (D+1)

(S1) < (S2) → (D+2)

— Die Daten in S1 und S2 werden als vorzeichenbehaftete Integer-Daten verarbeitet.

Beispiel:



Y0: wird eingeschaltet, wenn K10 > Datenregister D10 ist, Y1 und Y2 sind ausgeschaltet.

Y1: wird eingeschaltet, wenn K10 = Datenregister D10 ist, Y0 und Y2 sind ausgeschaltet.

Y2: wird eingeschaltet, wenn K10 < Datenregister D10 ist, Y0 und Y1 sind ausgeschaltet.

Y0, Y1, Y2 werden nicht verändert, wenn die Eingangsbedingung X10=0 ist.

Um die Vergleichsergebnisse zurückzusetzen, verwenden Sie die Anweisungen RST, ZRST.

ZCP	S1 S2 S D	D P	Vergleich von numerischen Daten nach Zonen

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
S	•	•					•	•	•	•	•
D					•	•					

Hinweis:

— Operand D belegt 3 Adressen.
— Operand S1 muss kleiner als S2 sein.

Beschreibung:

- Vergleich von numerischen Datenwerten mit numerischen Datenbereichen (größer, kleiner, gleich)
- Daten in der Quelle (S) werden mit Daten in beiden Quellen (S1) und (S2) verglichen.
- Das Ergebnis des Vergleichs (größer, kleiner, gleich) wird durch Aktivieren des Relais M oder Ausgangs Y angezeigt. Welcher der Kontakte am Zieloperanden (D) aktiv ist, wird durch das Vergleichsergebnis bestimmt.

(S) < (S1) → (D)
(S1) ≤ (S) ≤ (S2) → (D + 1)
(S) > (S2) → (D + 2)

- Wenn der Wert in (S1) größer als der Wert in (S2) ist, werden alle Kontakte im Operanden (D) zurückgesetzt.

Um die Vergleichsergebnisse zurückzusetzen, verwenden Sie die Anweisungen RST, ZRST.

MOV	S D	D P	Datenübertragung

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•

Beschreibung:

- Der MOV-Befehl wird verwendet, um Daten von einer Datenquelle (S) zu einem Ziel (D) zu übertragen. Der Wert der Quelle (S) ändert sich nicht.
- Daten in der Datenquelle (S) werden bei der Ausführung des MOV-Befehls als Binärwerte gelesen.
- Bit-Operanden belegen die Anzahl der Adressen entsprechend dem Befehlstyp - 16 oder 32 Adressen. In diesem Fall ist es möglich, die Typen der Operanden für Quelle und Ziel zu kombinieren. Beispielsweise zeigen die Relais M0 ... M15 als Ergebnis der

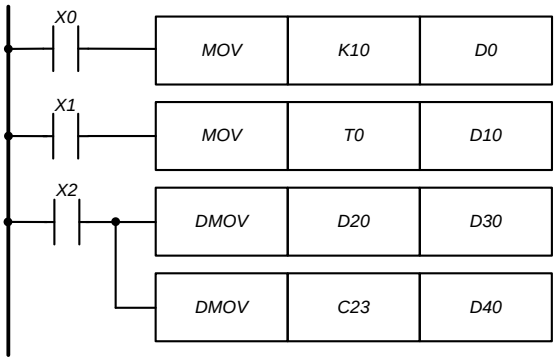
Ausführung des Befehls MOV D3 M0 den Wert des Registers D3 in binärer Form an.

Beispiel:

Wenn die Eingangsbedingung X0 eingeschaltet ist, ist der Wert des Registers D0 = 10. Wenn X0 ausgeschaltet ist, wird der Wert des Registers D0 nicht geändert.

Wenn die Eingangsbedingung X1 eingeschaltet ist, wird der aktuelle Wert des Timers T0 in das Datenregister D10 übertragen. Wenn X1 ausgeschaltet ist, wird der Wert des Registers D10 nicht geändert.

Wenn die Eingangsbedingung X2 eingeschaltet ist, wird der Wert der Register D20 und D21 in die Datenregister D30 und D31 übertragen; der aktuelle Wert des Zählers C23 wird in die Datenregister D40 und D41 übertragen.



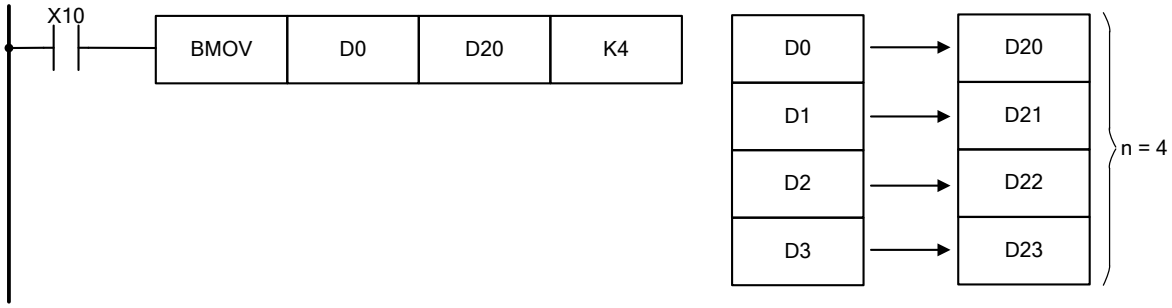
BMOV	S D n	D P	Blockdatenübertragung

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Beschreibung:

Kopieren Sie das Datenpaket. Die Verschiebung während des Vorgangs wird sowohl für den Quelloperanden (S) als auch für den Zieloperanden (D) um (n) Blockelemente durchgeführt, abhängig vom Befehl (16-Bit oder 32-Bit).

Beispiel:



Wenn X10 eingeschaltet ist, werden die Werte der Register D0 – D3 in die Register D20 – D23 übertragen.

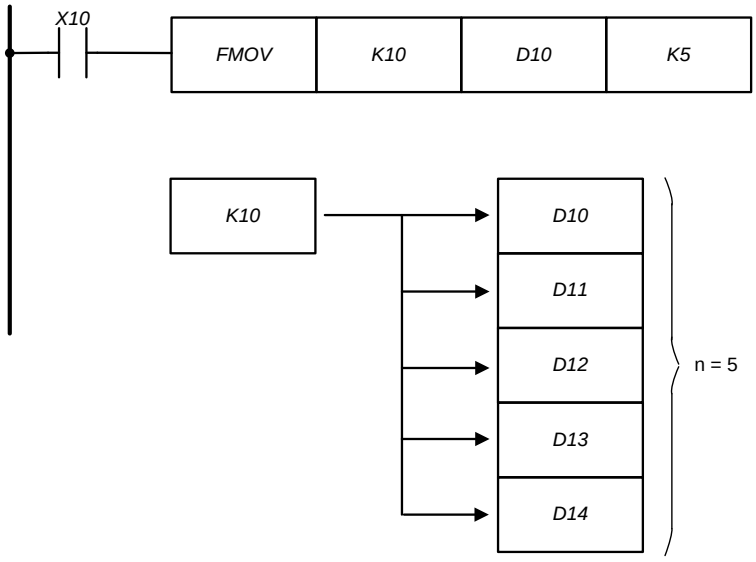
FMOV	S D n	D P	Übertragung von Daten an mehrere Adressen

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Beschreibung:

Der Wert des Quelloperanden (S) wird in (n) Zieloperanden (D) desselben Typs kopiert.

Beispiel:



Der Befehl FMOV kopiert den Wert "10" in die Datenregister D10.. .D14.

XCH	D1 D2	D P	Datenaustausch

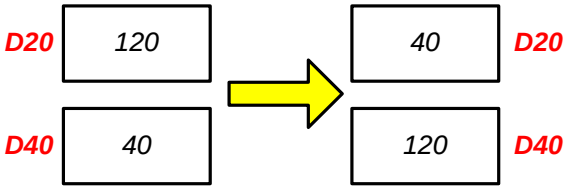
	K	H	F	X	Y	M	T	C	A	B	D
D1							•	•	•	•	•
D2							•	•	•	•	•

Beschreibung:

Die Werte der Operanden (D1) und (D2) werden getauscht.

Beispiel:

Wenn X0 = 1 ist, wird der Datenaustausch durchgeführt:



ADD	S1 S2 D	D P	Addition von numerischen Daten

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Beschreibung:

— Binärdaten in den Quelloperanden (S1) und (S2) werden addiert. Das Ergebnis der Addition wird im Zieloperanden (D) gespeichert. Die

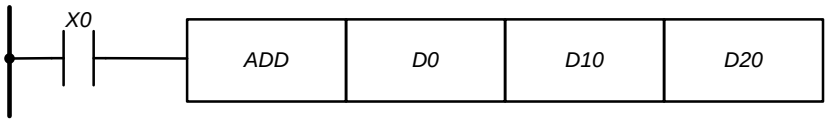
Operation wird mit vorzeichenbehafteten Integer-Datentypen durchgeführt.

(S1) + (S2) = (D)

— Das höchstwertige Bit enthält das Vorzeichen des Ergebnisses: 0 – Vorzeichen einer positiven Zahl, 1 – Vorzeichen einer negativen Zahl.

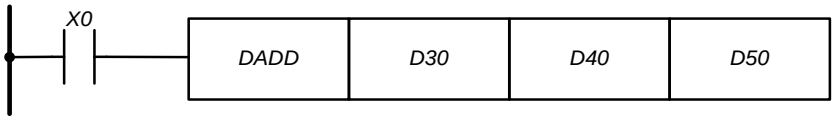
— Bei der Ausführung eines 32-Bit-Befehls sollten die unteren 16 Bit im Operanden angegeben werden. Das folgende Datenregister enthält die oberen 16 Bits.

Beispiele:



(D0) + (D10) = (D20)

Wenn X0 eingeschaltet ist, werden die Werte der Datenregister D0 und D10 addiert, das Ergebnis wird im Datenregister D20 gespeichert.



(D31, D30) + (D41, D40) = (D51, D50)

Wenn X0 eingeschaltet ist, wird das Ergebnis der Addition der Werte der Register (D31, D30) und (D41, D40) in den Datenregistern (D51, D50) gespeichert.

SUB	S1 S2 D	D P	Subtraktion numerischer Daten

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

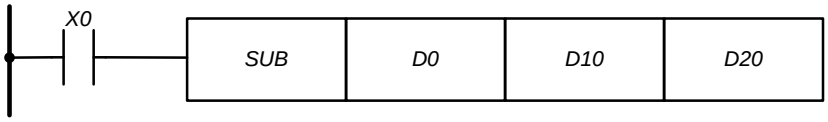
Beschreibung:

— Der Datenwert in (S2) wird vom Datenwert (S1) subtrahiert. Das Ergebnis der Subtraktion wird im Zieloperanden (D) gespeichert. Die

Operation wird mit vorzeichenbehafteten Integer-Datentypen durchgeführt.
(S1) – (S2) = (D)

- Das höchstwertige Bit enthält das Vorzeichen des Ergebnisses: 0 – Vorzeichen einer positiven Zahl, 1 – Vorzeichen einer negativen Zahl.
- Bei der Ausführung eines 32-Bit-Befehls sollten die unteren 16 Bit im Operanden angegeben werden. Das folgende Datenregister enthält die oberen 16 Bits.

Beispiele:



(D0) - (D10) = (D20)

Wenn X0 eingeschaltet ist, wird die Differenz zwischen den Datenwerten in den Registern D0 und D10 berechnet. Das Ergebnis wird im Datenregister D20 gespeichert.

(D31, D30) - (D41, D40) = (D51, D50)

Wenn X0 eingeschaltet ist, wird die Differenz zwischen den Datenwerten in den Registern (D31, D30) und (D41, D40) berechnet. Das Ergebnis wird in den Datenregistern (D51, D50) gespeichert.

MUL	S1 S2 D	D P	Multiplikation numerischer Daten

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

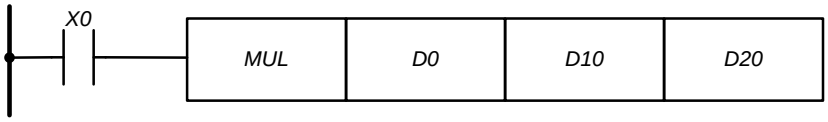
Beschreibung:

- Die Daten in den Operanden (S1) und (S2) werden miteinander multipliziert.Das Ergebnis wird im Zieloperanden (D) gespeichert. Die

Operation wird mit vorzeichenbehafteten Integer-Datentypen durchgeführt.
(S1) x (S2) = (D)

- Das höchstwertige Bit enthält das Vorzeichen des Ergebnisses: 0 – Vorzeichen einer positiven Zahl, 1 – Vorzeichen einer negativen Zahl.
- Bei der Ausführung eines 32-Bit-Befehls sollten die unteren 16 Bit im Operanden angegeben werden. Das folgende Datenregister enthält die oberen 16 Bits.

Beispiele:



(D0) * (D10) = (D20)

Wenn X0 eingeschaltet ist, werden die Werte in den Datenregistern D0 und D10 miteinander multipliziert. Das Ergebnis wird im Datenregister D20 gespeichert.

(D31, D30) * (D41, D40) = (D51, D50)

Wenn X0 eingeschaltet ist, werden die Werte in den Registern (D31, D30) und (D41, D40) miteinander multipliziert. Das Ergebnis wird in den Datenregistern (D51, D50) gespeichert.

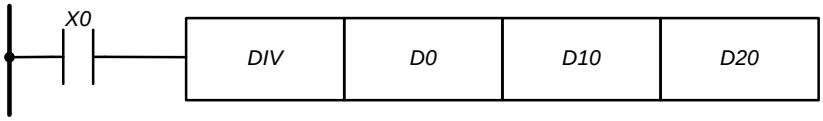
DIV	S1 S2 D	D P	Division numerischer Daten

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Beschreibung:

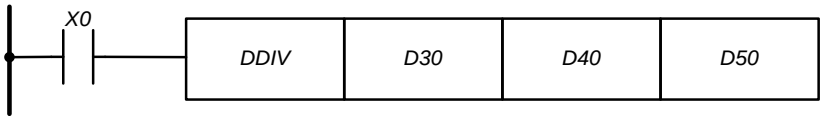
- Der Wert des Quelloperanden (S1) wird durch den Datenwert des Quelloperanden (S2) geteilt. Der ganze Teil des Divisionsergebnisses wird im Zieloperanden (D) gespeichert. Die Operation wird mit vorzeichenbehafteten Integer-Datentypen durchgeführt.
  (S1) / (S2) = (D)
- Das höchstwertige Bit enthält das Vorzeichen des Ergebnisses: 0 – Vorzeichen einer positiven Zahl, 1 – Vorzeichen einer negativen Zahl.
- Bei der Ausführung eines 32-Bit-Befehls sollten die unteren 16 Bit im Operanden angegeben werden. Das folgende Datenregister enthält die oberen 16 Bits.
- Division durch Null führt zu einem Fehler.

Beispiele:



(D0) / (D10) = (D20)

Wenn X0 eingeschaltet ist, wird eine Division der Datenwerte in den Registern D0 und D10 durchgeführt. Das Ergebnis wird im Datenregister D20 gespeichert.



(D31, D30) / (D41, D40) = (D51, D50)

Wenn X0 aktiviert ist, wird eine Division der Datenwerte in den Registern (D31, D30) und (D41, D40) durchgeführt. Das Ergebnis wird in den Datenregistern (D51, D50) gespeichert.

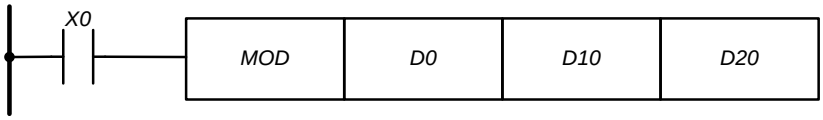
MOD	S1 S2 D	D P	Berechnung des Restes der Division

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Beschreibung:

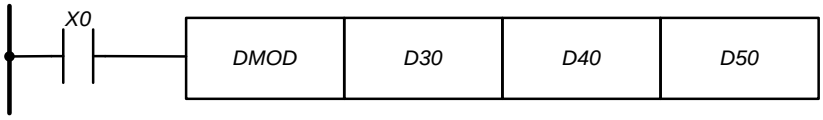
- Der Wert des Quelloperanden (S1) wird durch den Datenwert des Quelloperanden (S2) geteilt. Der Rest der Division wird im Zieloperanden (D) gespeichert. Die Operation wird mit vorzeichenbehafteten Integer-Datentypen durchgeführt.
  (S1) % (S2) = (D)
- Das höchstwertige Bit enthält das Vorzeichen des Ergebnisses: 0 – Vorzeichen einer positiven Zahl, 1 – Vorzeichen einer negativen Zahl.
- Bei der Ausführung eines 32-Bit-Befehls sollten die unteren 16 Bit im Operanden angegeben werden. Das folgende Datenregister enthält die oberen 16 Bits.
- Division durch Null führt zu einem Fehler.

Beispiele:



(D0) % (D10) = (D20)

Wenn X0 eingeschaltet ist, wird eine Division der Datenwerte in den Registern D0 und D10 durchgeführt. Das Ergebnis (Rest bei der Division) wird im Datenregister D20 gespeichert.



(D31, D30) % (D41, D40) = (D51, D50)

Wenn X0 aktiviert ist, wird eine Division der Datenwerte in den Registern (D31, D30) und (D41, D40) durchgeführt. Das Ergebnis (Rest bei der Division) wird in den Datenregistern (D51, D50) gespeichert.

INC	D	D P	Numerische Daten erhöhen

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Beschreibung:

Der Wert im Operanden (D) wird um 1 erhöht.

Beispiel:

Der Wert in den Datenregistern (D1, D0) wird um 1 erhöht, wenn die Eingangsbedingung X0 eingeschaltet ist. Der Befehl wird aufgrund der angeschlossenen Pulsfunktion aktiviert, so dass der Summierungsvorgang nicht in jedem Programmzyklus durchgeführt wird.

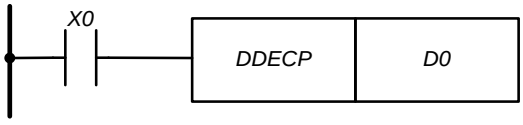
DEC	D	D P	Numerische Daten dekrementieren

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Beschreibung:

Der Wert im Operanden (D) wird um 1 verringert.

Beispiel:



Der Wert in den Datenregistern (D1, D0) wird um 1 verringert, wenn die Eingangsbedingung X0 eingeschaltet ist. Der Befehl wird aufgrund der

angeschlossenen Pulsfunktion aktiviert, so dass die Dekrementierung nicht in jedem Programmzyklus durchgeführt wird.

WAND	S1 S2 D	D P	Logische Multiplikation numerischer Daten (Operation „AND“)

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

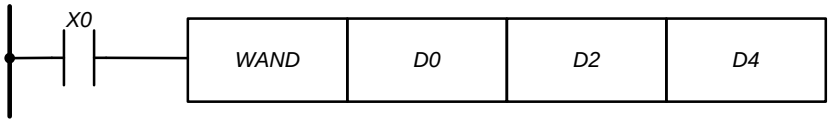
Hinweis: WAND ist ein 16-Bit-Befehl, DAND ist ein 32-Bit-Befehl.

Beschreibung:

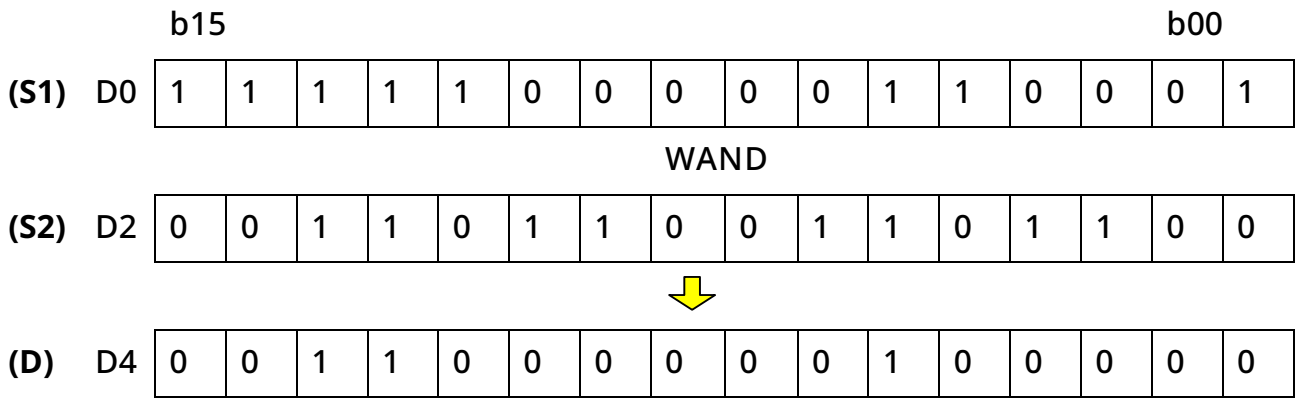
- Die Operation "logisches AND" für numerische Daten ist eine Bitoperation (wird bitweise ausgeführt).
- Die Werte der Quelloperanden (S1) und (S2) werden bitweise multipliziert. Das Ergebnis wird im Zieloperanden (D) gespeichert.
- Die Wahrheitstabelle der logischen Multiplikation:

(S1)	(S2)	(D)
1	1	1
1	0	0
0	1	0
0	0	0

Beispiel:



Wenn X0 = 1 ist, wird die logische Multiplikation der Werte aus den Datenregistern D0 und D2 durchgeführt. Das Ergebnis wird im Datenregister D4 gespeichert.



WOR	S1 S2 D	D P	Logische Addition numerischer Daten (OR-Verknüpfung)

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

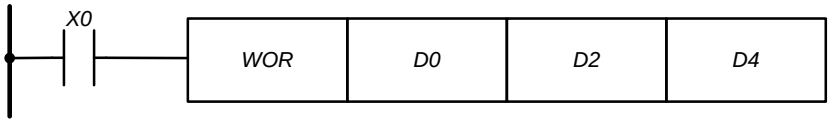
Hinweis: WOR ist ein 16-Bit-Befehl, DOR ist ein 32-Bit-Befehl.

Beschreibung:

- Die Operation "logisches OR" für numerische Daten ist eine Bitoperation (wird bitweise ausgeführt).
- Die Werte der Quelloperanden (S1) und (S2) werden bitweise addiert. Das Ergebnis wird im Zieloperanden (D) gespeichert.
- Die Wahrheitstabelle der logischen Addition:

(S1)	(S2)	(D)
1	1	1
1	0	1
0	1	1
0	0	0

Beispiel:



Wenn X0 = 1 ist, wird die logische Addition der Werte aus den Datenregistern D0 und D2 durchgeführt. Das Ergebnis wird im Datenregister D4 gespeichert.

		b15														b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
								WOR								
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
		⇩														
(D)	D4	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1

WXOR	S1 S2 D	D P	Logische Operation „Exklusives OR“

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

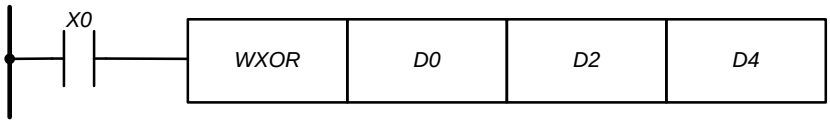
Hinweis:WXOR ist ein 16-Bit-Befehl, DXOR ist ein 32-Bit-Befehl.

Beschreibung:

- Die Operation "logisches exklusives OR" für numerische Daten ist eine Bitoperation (wird bitweise ausgeführt).
- Die Werte der Quelloperanden (S1) und (S2) werden bitweise verarbeitet. Das Ergebnis wird im Zieloperanden (D) gespeichert.
- Die Wahrheitstabelle des logischen exklusiven OR:

(S1)	(S2)	(D)
1	1	0
1	0	1
0	1	1
0	0	0

Beispiel:



Wenn X0 = 1 ist, wird die Operation "exklusives OR" mit den Werten in den Datenregistern D0 und D2 durchgeführt. Das Ergebnis der Operation wird im Datenregister D4 gespeichert.

		b15														b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
									WXOR							
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
									⇩							
(D)	D4	1	1	0	0	1	1	1	0	0	1	0	1	1	1	0	1

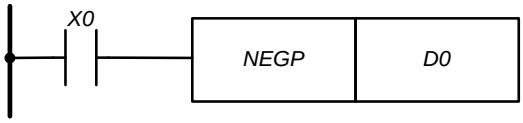
NEG	D	D P	Logische Negation

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Beschreibung:

Logische Negationsoperation (Inversion aller Bits in binärer Form und Addition mit 1) für numerische Daten.

Beispiel:



Wenn X0 = 1 ist, wird die Operation der logischen Negation und Modifikation mit dem Wert im Operanden D0 durchgeführt.

13931 → -13931 + 1 = -13931

		b15														b00
(D)	D0	0	0	1	1	0	1	1	0	0	1	1	0	1	0	1	1
		⇩														
(D)	D0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	1

-13931 → 13930 + 1 = 13931

		b15														b00
(D)	D0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	1
		⇩														
(D)	D0	0	0	1	1	0	1	1	0	0	1	1	0	1	0	1	1

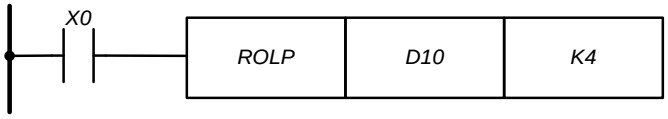
ROL	D n	D P	Zyklische Verschiebung nach links

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•
n	•	•					•	•	•	•	•

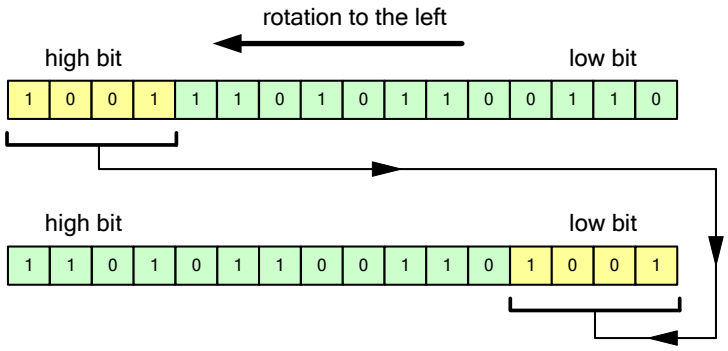
Beschreibung:

Bits Rotation um (n) Stellen nach links.

Beispiel:



Wenn X0 = 1 ist, rotieren die Bits des Wertes im Datenregister D10 um 4 Bits nach links und der Wert wird geändert.



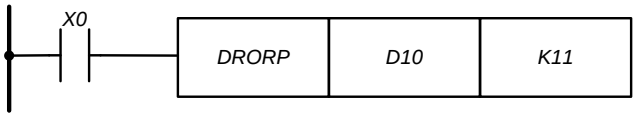
ROR	D n	P	Zyklische Verschiebung nach rechts

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•
n	•	•					•	•	•	•	•

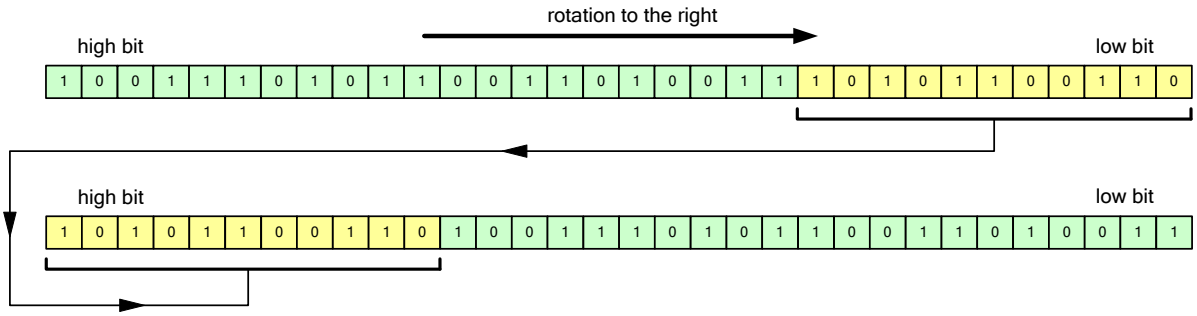
Beschreibung:

Bits Rotation um (n) Stellen nach rechts.

Beispiel:



Wenn X0 = 1 ist, rotieren die Bits des Wertes im Datenregister D10 um 11 Bits nach rechts und der Wert wird geändert.



ZRST	D1 D2	D P	Gruppenrücksetzung von Operanden

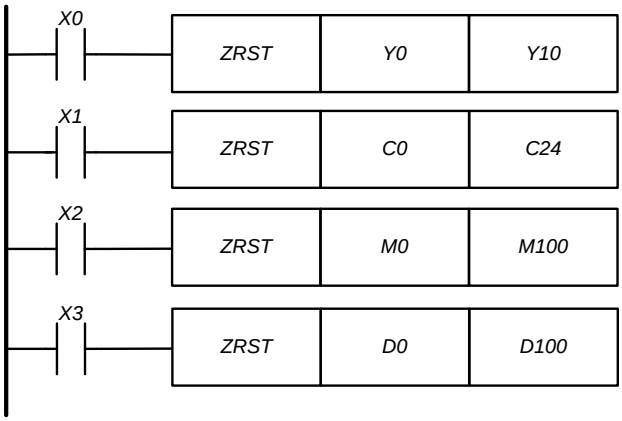
	K	H	F	X	Y	M	T	C	A	B	D
D1					•	•	•	•	•	•	•
D2					•	•	•	•	•	•	•

Beschreibung:

Die Werte mehrerer nacheinanderfolgender Operanden (Operandenbereich) können durch den Befehl ZRST zurückgesetzt werden. Bitkontakte werden ausgeschaltet, die Register werden auf den Wert "0" gesetzt.

- Die Operanden (D1) und (D2) bestimmen den zurückzusetzenden Bereich.
- Die Operanden in (D1) und (D2) müssen vom gleichen Typ sein.
- (D1) – die erste Adresse des Bereichs, (D2) – die letzte Adresse des Bereichs.
- (D1) muss kleiner als (D2) sein.

Beispiel:



Wenn die Eingangsbedingungen erfüllt sind, werden die Bitoperanden Y0 ... Y10, M0 ... M100 ausgeschaltet (gehen in den Zustand "0" über). Die numerischen Operanden C0 ... C24, D0 ... D100 werden auf den tatsächlichen Wert "0" gesetzt. Die entsprechenden Spulen und Kontakte werden ausgeschaltet.

DECO	S D n	P	Decoder 8 → 256 Bit

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•		•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Hinweis: Wenn (D) ein Bit-Operand ist: (n) = 1...8. Wenn (D) ein numerischer Operand ist: (n) = 1...4. Wenn (n) außerhalb des möglichen Bereichs liegt, wird der Befehl mit dem maximal möglichen (n) abhängig von (D) ausgeführt.

Beschreibung:

Dekodierung der Daten. Die Daten in (n) Operanden werden beginnend mit der Startadresse dekodiert, die in (S) angegeben ist. Der Operand (D) bestimmt die Startadresse des Ziels (wohin das Ergebnis der Entschlüsselung geschrieben wird).

(n) ist die Anzahl der Operanden, deren Daten dekodiert werden sollen. Bei der Angabe des Bitoperanden in (D) ist folgendes zu beachten: 1 ≤ (n) ≤ 8. Bei der Angabe des numerischen Operanden in (D) ist folgendes zu beachten: 1 ≤ (n) ≤ 4.

(S) ist die Startadresse der Operanden, deren Daten dekodiert werden sollen.

2^n– Anzahl der zu dekodierenden Operanden.

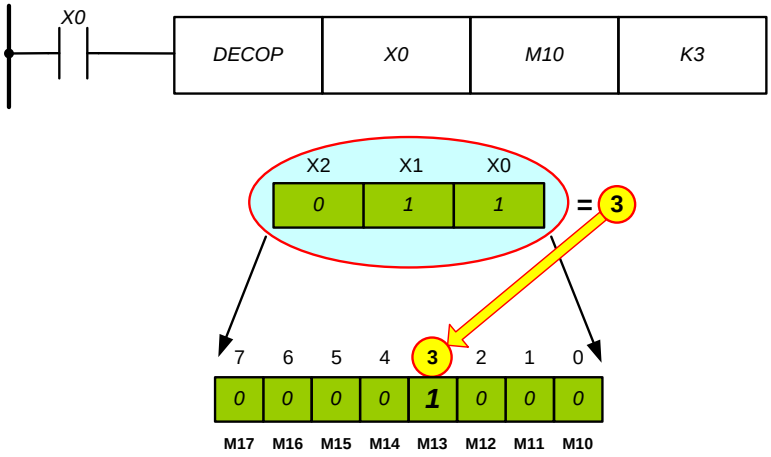
(D) ist eine Startadresse für den Zieloperanden.

Achtung! DieAnweisung wird nicht ausgeführt, wenn (n) = 0 ist.

Dementsprechend bleibt der Ausgang aktiv, wenn die Eingangsbedingungen am Ende der Aktion wieder abgeschaltet werden.

Beispiel:

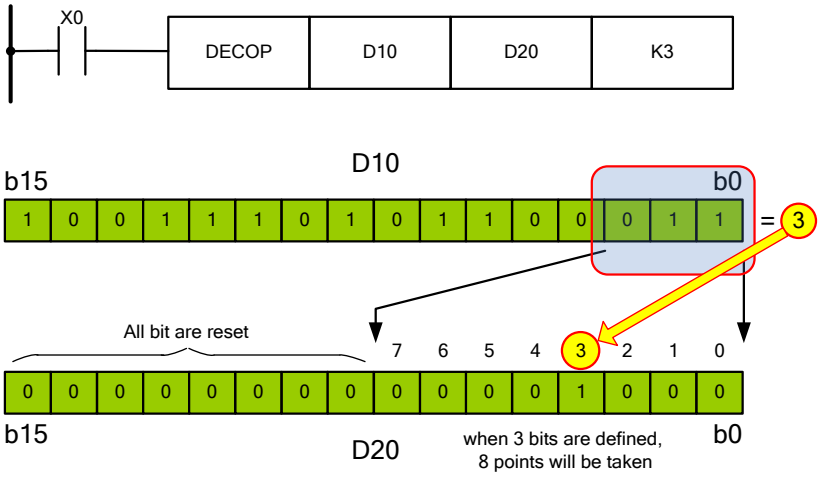
Verwendung eines DECO-Befehls mit Bitoperanden in (D).



Wenn (n) = 3 ist, werden die Eingangsoperanden X0, X1 und X2 verarbeitet. $2^n = 2^3 = 8$ Adressen werden als Ziel M10...M17 verwendet.

Der Wert der Eingangsoperanden ist 1 + 2 = 3. Daher wird die 3. Adresse des Ziels, d.h. Relais M13, eingeschaltet. Wenn der Wert des Eingangsoperanden "0" verarbeitet wird, dann wird das Relais M10 aktiviert.

Verwendung einer DECO-Anweisung mit numerischen Operanden in (D).



Die unteren 3 Bits des Datenregisters D10 werden dekodiert. Das Dekodierungsergebnis 1 + 2 = 3 wird in das Datenregister D20 übertragen. Das 3. Bit wird in diesem Datenregister eingeschaltet.

Wenn der Wert für (n) <3 ist, dann werden alle unnötigen Bits einer höheren Zahl in den Zieladressen auf Null gesetzt.

ENCO	S D n	P	Kodierer 256 → 8 Bit

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•		•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Hinweis: Wenn (D) ein Bit-Operand ist: (n) = 1...8. Wenn (D) ein numerischer Operand ist: (n) = 1...4. Wenn (n) außerhalb des möglichen Bereichs liegt, wird der Befehl mit dem maximal möglichen (n) abhängig von (D) ausgeführt.

Beschreibung:

Kodierung von Daten. Daten in 2^n Operanden werden ab der Startadresse kodiert, die in (S) angegeben ist. Der Operand (D) bestimmt die Startadresse des Ziels (wohin das Ergebnis der Entschlüsselung geschrieben wird).

2^nist die Anzahl der Operanden, deren Daten kodiert werden sollen.

(n) – die Anzahl der Zieloperanden.

Bei der Angabe des Bit-Operanden in (S) ist folgendes zu beachten: 1 ≤ (n) ≤ 8. Bei der Angabe des numerischen Operanden in (S) ist folgendes zu beachten: 1 ≤ (n) ≤ 4.

(S) ist die Startadresse der Operanden, deren Daten kodiert werden sollen.

(D) ist die Startadresse des Zieloperanden.

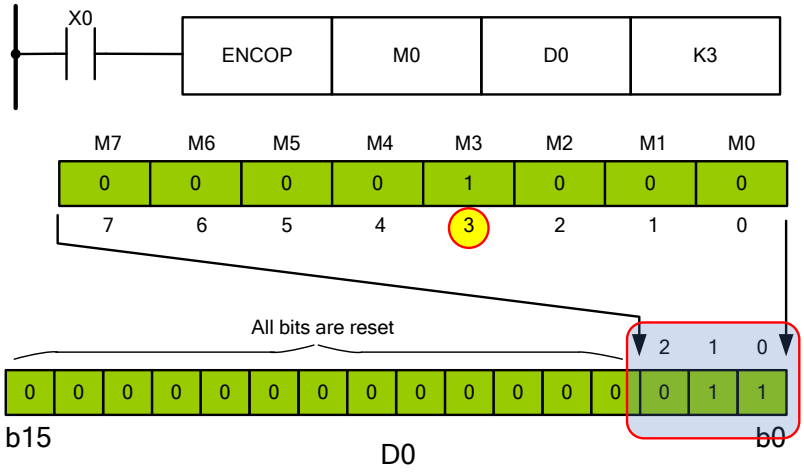
Wenn mehrere in (S) angegebene Operanden den Wert „1“ haben, wird nur das niedrigste Bit verarbeitet.

Achtung! Die Anweisung wird nicht ausgeführt, wenn (n) = 0 ist.

Dementsprechend bleibt der Ausgang aktiv, wenn die Eingangsbedingungen am Ende der Aktion wieder abgeschaltet werden.

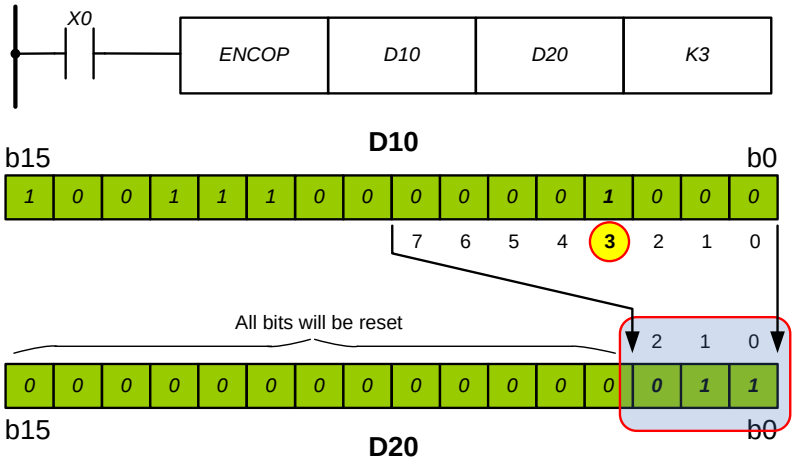
Beispiel:

Verwendung einer ENCO-Anweisung mit Bit-Operanden in (S).



Wenn $2^n = 2^3 = 8$ ist, dann sind die Ausgangsrelaisadressen M0.. .M7. Sobald der 3D-Ausgang aktiviert wird (d. h. M3 = 1), wird der Wert 3 in das Datenregister D0 geschrieben.

Verwendung einer ENCO-Anweisung mit numerischen Operanden in (S).



Das 3. Bit ist im Datenregister D10 eingeschaltet. Daher wird der Wert 3 kodiert und im Datenregister D20 gespeichert.

SUM	S D	D P	Summe einzelner Bits

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D							•	•	•	•	•

Beschreibung:

- Ermittlung der Anzahl der aktiven Bits in einem Datenwort. Die Anweisung zählt die eingeschalteten Bits in (S).
- Der Ergebniswert wird in (D) geschrieben.

Wenn eine 32-Bit-Anweisung verarbeitet wird, werden die oberen 16 Bits (D + 1) der Zieloperanden (D) auf Null gesetzt, da die maximale Anzahl der eingeschalteten Bits in (S) 32 beträgt.

BON	S D n	D P	Bitstatus prüfen

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D					•	•					
n	•	•					•	•	•	•	•

Hinweis: Die notwendige Bedingung: (n) = 0...15 (16 Bit), (n) = 0...31 (32 Bit).

Beschreibung:

Ein einzelnes Bit wird innerhalb des Datenworts geprüft. Wenn das Bit (n) in (S) eingeschaltet ist, wird das entsprechende Bit in (D) eingeschaltet.

SQR	S D	D P	Berechnung der Quadratwurzel

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D											•

Beschreibung:

Berechnung der Quadratwurzel $(D) = \sqrt{(S)}$

Die Quadratwurzel des Werts im Operanden (S) wird berechnet, das Ergebnis wird auf eine ganze Zahl gerundet und in den Operanden (D) geschrieben. Die Operation wird mit vorzeichenbehafteten Integer-Datentypen durchgeführt.

Achtung! Die Quadratwurzel einer negativen Zahl führt immer zu einem Fehler.

FLT	S D	D P	Konvertierung einer Ganzzahl in eine Gleitkommazahl

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D											•

Beschreibung:

Die Anweisung FLT konvertiert eine ganze Zahl mit Vorzeichen in das Gleitkommaformat.

- Die Ganzzahl im Operanden (S) wird in eine Gleitkommazahl umgewandelt. Das Ergebnis wird im Operanden (D) gespeichert.
- Das Ergebnis der Konvertierung ist immer eine 32-Bit-Zahl.

PWM	S1 S2 D		PWM-Pulsausgabe

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D					•						

Hinweis: Der Wert des Operanden (S1) muss kleiner oder gleich dem Wert des Operanden (S2) sein.

Beschreibung:

(S1) – Impulsbreite, t.

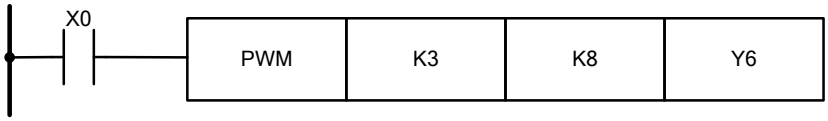
(S2) – Periodendauer, T.

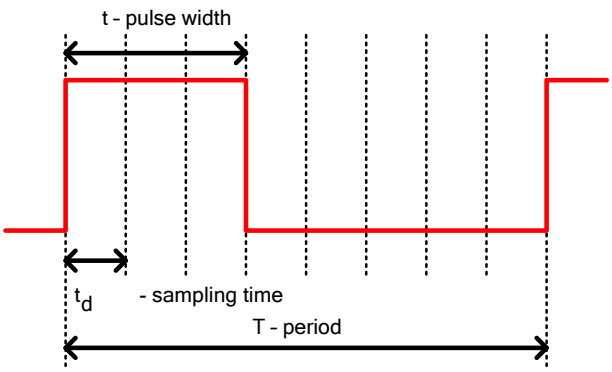
(D) – Ausgabeadresse, Y6 oder Y7.

Gültige Werte für (S1) und (S2) sind von 1 bis 32767. (S1) und (S2) sind die Anzahl der Abtastintervalle. Die Abtastperiode wird gleichzeitig für beide Kanäle, 100 µs oder 10 µs, durch das Sonderregister D356 eingestellt (siehe Abschnitt 4.6für weitere Details).

Ein PWM-Signal liegt am Ausgang an, solange das Signal am Eingang der PWM-Anweisung aktiv ist.

Beispiel:





Die Abtastperiode sei 100 µs. Wenn die Eingangsbedingung X0 = 1 erfüllt ist, erscheint am Ausgang Y6 das PWM-Signal mit einer Periode von 8 x 100 µs = 0.8 ms und einer Impulsdauer von 3 x 100 µs = 0.3 ms.

ABS	D	D P	Absolutwert

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Beschreibung:

Die Anweisung ABS schreibt den Absolutwert einer Zahl in den Operanden (D). Wenn der Wert in (D) negativ ist, wird nach der Ausführung der ABS-Anweisung das Vorzeichen „-“ verworfen und die Zahl wird positiv. Wenn (D) einen positiven Wert hat, treten keine Änderungen auf.

Beispiel:

Wenn die Eingangsbedingung erfüllt ist, wird der Betrag der Zahl im Register (D1, D0) ermittelt.

POW	S1 S2 D	D P	Potenzieren

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D											•

Beschreibung:

Potenzieren: $(D) = (S1)^{(S2)}$.

Der Befehl POW potenziert den Wert im Operanden (S1) mit (S2). Das Ergebnis wird im Operanden (D) gespeichert. Die Operation wird mit vorzeichenbehafteten Integer-Datentypen durchgeführt.

DECMP	S1 S2 D	D P	Vergleich von Gleitkommazahlen

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D					•	•					

Hinweis:

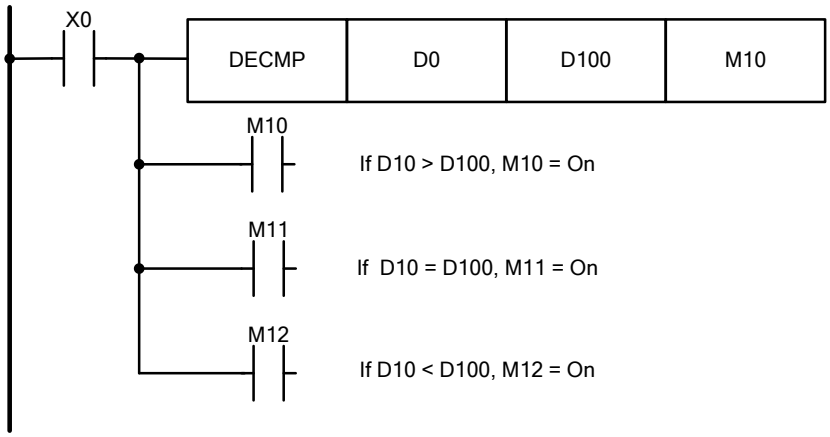
- Nur 32-Bit-Befehl.
- Der Operand (D) belegt 3 aufeinanderfolgende Adressen.
- K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.

Beschreibung:

Die Anweisung DECMP führt einen Vergleich zweier binärer Gleitkommazahlen durch und gibt das Ergebnis des Vergleichs aus.

- Die DECMP-Anweisung vergleicht die Gleitkommazahl in (S1) mit der Gleitkommazahl in (S2).
- Das Vergleichsergebnis wird in 3 aufeinanderfolgenden Adressen gespeichert.
- Wenn die Zahl in (S2) kleiner als die Zahl in (S1) ist, wird der Bit-Operand (D) eingeschaltet.
- Wenn die Zahl in (S2) gleich der Zahl in (S1) ist, wird der Bit-Operand ((D) +1) eingeschaltet.
- Wenn die Zahl in (S2) größer als die Zahl in (S1) ist, wird der Bit-Operand ((D) +2) eingeschaltet.
- Die abgefragten Ausgabeoperanden bleiben nach dem Deaktivieren der Eingangsbedingungen der Anweisung DECMP eingeschaltet.

Beispiel:



Wenn der Kontakt X0 eingeschaltet wird, wird die in D100 (S2) angegebene Gleitkommazahl mit der in D0 (S1) angegebenen Gleitkommazahl verglichen. Wenn die Zahl in D100 kleiner als die Zahl D0 ist, wird das Relais M10 aktiviert. Wenn die Zahl in D100 gleich der Zahl D0 ist, wird das Relais M11 eingeschaltet. Wenn die Zahl in D100 größer als die Zahl D0 ist, wird das Relais M12 aktiviert.

Um die Vergleichsergebnisse in der Form: ≤ ≥ ≠ zu erhalten, können Sie die parallelen Kontaktkombinationen M10 - M12 verwenden. Um das Ergebnis zurückzusetzen, können Sie die Anweisungen RST, ZRST verwenden.

DEZCP	S1 S2 S D	D P	Zonen-Gleitkommavergleich

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
S	•	•	•								•
D					•	•					

Hinweis:

- Nur 32-Bit-Befehl.
- Der Operand (D) belegt 3 aufeinanderfolgende Adressen.
- K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.

Beschreibung:

Vergleich der Gleitkommazahl mit dem ausgewählten (angegebenen) Bereich und Ausgabe des Vergleichsergebnisses.

- Der Befehl DEZCP vergleicht die Gleitkommazahl im Operanden (S1) mit dem Bereich zwischen (S1) und (S2).
- Das Vergleichsergebnis wird in 3 aufeinanderfolgenden Adressen gespeichert.
- Wenn die Zahl im Operanden (S) kleiner als die Zahlen in den Operanden (S1) und (S2) ist, wird der Bit-Operand (D) eingeschaltet.
-
- Wenn die Zahl im Operanden (S) gleich den Zahlen zwischen (S1) und (S2) ist, wird der Bit-Operand ((D) +1) eingeschaltet.
- Wenn die Zahl im Operanden (S) größer als die Zahlen zwischen (S1) und (S2) ist, wird der Bit-Operand ((D) +2) eingeschaltet.
- Die abgefragten Ausgabeoperanden bleiben eingeschaltet, nachdem die DEZCP-Befehlseingabebedingungen deaktiviert wurden.
- Wenn die Zahl im Operanden (S1) größer als die Zahl im Operanden (S2) ist, werden alle Bits in (D), (D+1) und (D+2) zurückgesetzt.

DEADD	S1 S2 D	D P	Addition von Gleitkommazahlen

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Hinweis:

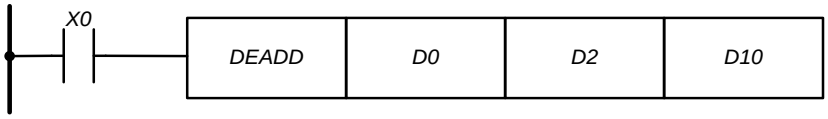
- Nur 32-Bit-Befehl.
- K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.

Beschreibung:

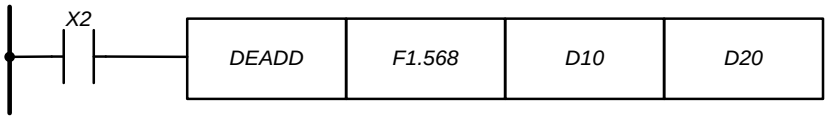
Berechnung der Summe zweier Zahlen im binären Gleitkommaformat.

— Die in (S1) und (S2) angegebenen Gleitkommazahlen werden addiert. Das Ergebnis der Addition wird im Zieloperanden (D) gespeichert.
— Für jeden Operanden werden zwei aufeinanderfolgende Register verwendet.
— Derselbe Operand kann als Quelle und als Ziel verwendet werden. In diesem Fall wird das berechnete Ergebnis wieder im Quelloperanden gespeichert und kann für die nächste Berechnung verwendet werden. Dieser Vorgang wird in jedem Programmzyklus wiederholt.

Beispiele:



Wenn der Eingang X0 eingeschaltet ist, wird die Gleitkommazahl in (D3, D2) zu der Gleitkommazahl in (D1, D0) addiert. Das Ergebnis wird in (D11, D10) gespeichert.



Wenn der Eingang X2 eingeschaltet ist, wird die Gleitkommazahl in (D11, D10) zur Konstanten F1.568 addiert. Das Ergebnis wird in (D21, D20) gespeichert.

DESUB	S1 S2 D	D P	Subtraktion von Gleitkommazahlen

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Hinweis:

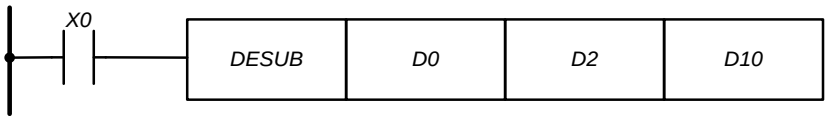
— Nur 32-Bit-Befehl.
— K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.

Beschreibung:

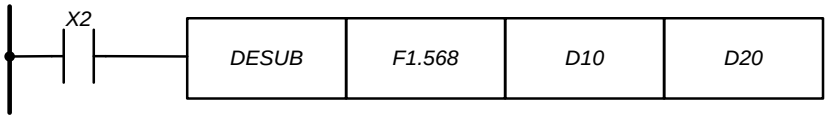
Der Befehl DESUB berechnet die Differenz zweier Zahlen im binären Gleitkommaformat.

- Die in (S2) angegebene Gleitkommazahl wird von der in (S1) angegebenen Gleitkommazahl subtrahiert. Das Ergebnis wird in (D) gespeichert.
- Für jeden Operanden werden zwei aufeinanderfolgende Register verwendet.
- Derselbe Operand kann als Quelle und als Ziel verwendet werden. In diesem Fall wird das berechnete Ergebnis wieder im Quelloperanden gespeichert und kann für die nächste Berechnung verwendet werden. Dieser Vorgang wird in jedem Programmzyklus wiederholt.

Beispiele:



Wenn Eingang X0 eingeschaltet ist, wird die Gleitkommazahl in (D3, D2) von der Gleitkommazahl in (D1, D0) subtrahiert. Das Ergebnis wird in (D11, D10) gespeichert.



Wenn der Eingang X2 eingeschaltet ist, wird die Gleitkommazahl in (D11, D10) von der Konstanten F1.568 subtrahiert. Das Ergebnis wird in (D21, D20) gespeichert.

DEMUL	S1 S2 D	D P	Multiplikation von Gleitkommazahlen

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Hinweis:

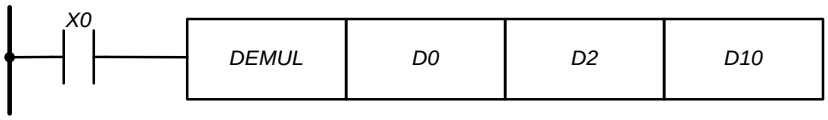
- Nur 32-Bit-Befehl.
- K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.

Beschreibung:

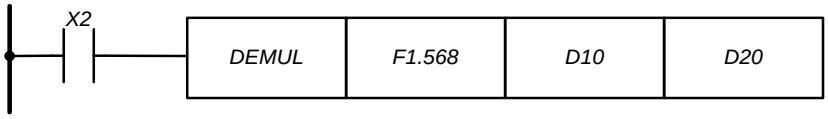
Multiplikation zweier Zahlen im binären Gleitkommaformat.

- Die im Operanden (S1) angegebene Gleitkommazahl wird mit der Gleitkommazahl im Operanden (S2) multipliziert. Das Ergebnis wird im Operanden (D) gespeichert.
- Für jeden Operanden werden zwei aufeinanderfolgende Register verwendet.
- Derselbe Operand kann als Quelle und als Ziel verwendet werden. In diesem Fall wird das berechnete Ergebnis wieder im Quelloperanden gespeichert und kann für die nächste Berechnung verwendet werden. Dieser Vorgang wird in jedem Programmzyklus wiederholt.

Beispiele:



Wenn Eingang X0 eingeschaltet ist, wird die Gleitkommazahl in (D1, D0) mit der Gleitkommazahl in (D3, D2) multipliziert. Das Ergebnis wird in (D11, D10) gespeichert.



Wenn Eingang X2 eingeschaltet ist, wird die Konstante F1.568 mit der Gleitkommazahl in (D11, D10) multipliziert. Das Ergebnis wird in (D21, D20) gespeichert.

DEDIV	S1 S2 D	D P	Gleitkommazahlendivision

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Hinweis:

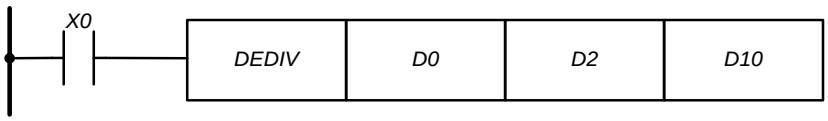
- Nur 32-Bit-Befehl.
- K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.

Beschreibung:

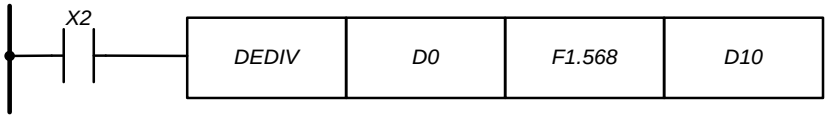
Berechnung des Quotienten der Division zweier Zahlen im binären Gleitkommaformat.

- Die im Operanden (S1) angegebene Gleitkommazahl wird durch die im Operanden (S2) angegebene Gleitkommazahl dividiert. Das Ergebnis wird in (D) gespeichert.
- Für jeden Operanden werden zwei aufeinanderfolgende Register verwendet.
- Derselbe Operand kann als Quelle und als Ziel verwendet werden. In diesem Fall wird das berechnete Ergebnis wieder im Quelloperanden gespeichert und kann für die nächste Berechnung verwendet werden. Dieser Vorgang wird in jedem Programmzyklus wiederholt.
- Der Operand (S2) darf nicht Null sein, da eine Division durch Null nicht zulässig ist.

Beispiele:



Wenn der Eingang X0 eingeschaltet ist, wird die Gleitkommazahl in (D1, D0) durch die Gleitkommazahl in (D3, D2) geteilt. Das Ergebnis wird in (D11, D10) gespeichert.



Wenn der Eingang X2 eingeschaltet ist, wird die Gleitkommazahl in (D1, D0) durch die Konstante F1.568 geteilt. Das Ergebnis wird in (D11, D10) gespeichert.

DESQR	S D	D P	Quadratwurzel im Gleitkommaformat

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•								•
D											•

Hinweis:

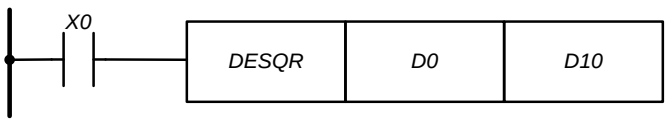
- Nur 32-Bit-Befehl.
- K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.
- Notwendige Bedingung: (S) ≥ 0

Beschreibung:

Berechnung der Quadratwurzel einer binären Gleitkommazahl.

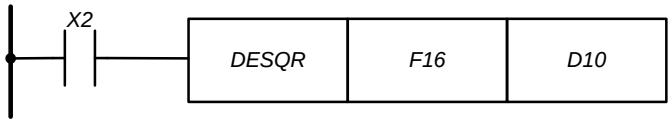
- Die Quadratwurzel wird aus der im Operanden (S) angegebenen Gleitkommazahl berechnet. Das Ergebnis wird in (D) gespeichert.
- Für jeden Operanden werden zwei aufeinanderfolgende Register verwendet.
- Derselbe Operand kann als Quelle und als Ziel verwendet werden. In diesem Fall wird das berechnete Ergebnis wieder im Quelloperanden gespeichert und kann für die nächste Berechnung verwendet werden. Dieser Vorgang wird in jedem Programmzyklus wiederholt.

Beispiele:



$$\sqrt{(D1, D0)} \rightarrow (D11, D10)$$

Wenn der Eingang X0 eingeschaltet ist, wird die Quadratwurzel der Gleitkommazahl in (D1, D0) berechnet. Das Ergebnis wird in (D11, D10) gespeichert.



Wenn der Eingang X0 eingeschaltet ist, wird die Quadratwurzel der Konstante F16 berechnet. Das Ergebnis wird in (D11, D10) gespeichert.

DEPOW	S1 S2 D	D P	Potenzieren im Gleitkommaformat

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Hinweis:

- Nur 32-Bit-Befehl.
- K- und H-Typen werden nicht in F konvertiert, sondern auf einen Speicherbereich projiziert. Um Integer-Datentypen in Gleitkommadaten zu konvertieren, verwenden Sie den Befehl **FLT**.

Beschreibung:

Potenzieren einer Zahl im binären Gleitkommaformat.

- Die in (S1) angegebene Zahl wird mit (S2) potenziert. Das Ergebnis wird im Operanden (D) gespeichert.
  $(S1)^{(S2)} = (D)$.
- Für jeden Operanden werden zwei aufeinanderfolgende Register verwendet.
- Derselbe Operand kann als Quelle und als Ziel verwendet werden. In diesem Fall wird das berechnete Ergebnis wieder im Quelloperanden gespeichert und kann für die nächste Berechnung verwendet werden. Dieser Vorgang wird in jedem Programmzyklus wiederholt.

INT	S D	D P	Konvertierung einer Gleitkommazahl in eine Ganzzahl

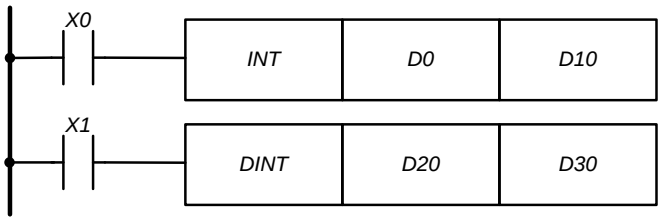
	K	H	F	X	Y	M	T	C	A	B	D
S	.										•
D							•	•	•	•	•

Beschreibung:

Der Befehl INT konvertiert Gleitkommazahlen in ganze Zahlen, gerundet auf die nächste ganze Zahl.

- Die in (S) angegebene Gleitkommazahl wird auf den nächsten ganzzahligen Wert gerundet und in (D) gespeichert.
- Der Quelloperand ist immer ein Doppelwort.
- Bei Verwendung des Befehls INT ist der Wortoperand der Operand des Ziels.
- Bei Verwendung des Befehls DINT ist der Zieloperand ein Doppelwortoperand.
- Der Befehl INT ist die Umkehrfunktion des Befehls FLT.

Beispiel:



Wenn der Eingang X0 eingeschaltet wird, wird die Gleitkommazahl in (D0, D1) auf den nächsten niedrigeren ganzzahligen Wert abgerundet. Das Ergebnis wird in D10 gespeichert.

Wenn der Eingang X1 eingeschaltet wird, wird die Gleitkommazahl in (D20, D21) auf den nächsten niedrigeren ganzzahligen Wert abgerundet. Das Ergebnis wird in (D30, D31) gespeichert.

TRD	D	P	Lesezeitdaten

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Hinweis: Der Operand D belegt 3 aufeinanderfolgende Adressen.

Beschreibung:

Lesen des aktuellen Werts der Echtzeituhr.

- Mit dem Befehl TRD werden Echtzeitdaten gelesen (Stunden, Minuten, Sekunden).
- Diese Daten werden in 3 aufeinanderfolgenden Operandenadressen (D) gespeichert.

Beispiel:

Wenn Eingang X0 eingeschaltet ist, werden Echtzeitdaten gelesen und in den Registern D0 ... gespeichert. D2

Register	Funktion	Wert	Beispiel	
D0	Sekunden	0...59	20	12:36:20
D1	Protokoll	0...59	36	
D2	Stunden	0...23	12	

TWR	S	P	Aufzeichnen von Zeitdaten

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Hinweis: Der Operand (S) belegt 3 aufeinanderfolgende Adressen.

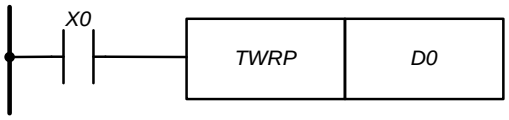
Beschreibung:

Der Befehl TWR wird verwendet, um die Echtzeitdaten (Stunden, Minuten, Sekunden) zu ändern.

Die Daten werden aus 3 aufeinanderfolgenden Adressen entnommen, die in (S) angegeben sind.

Wenn die Werte in (S) den zulässigen Wertebereich überschreiten, tritt ein Fehler auf.

Beispiel:



Wenn die Eingangsbedingung erfüllt ist, wird die Echtzeituhr des Controllers auf die in den Registern D0 ... angegebenen Werte gesetzt. D2.

Register	Funktion	Wert	Beispiel	
D0	Sekunden	0...59	42	03:11:42
D1	Protokoll	0...59	11	
D2	Stunden	0...23	3	

DRD	D	P	Datum lesen

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Hinweis: Der Operand D belegt 3 aufeinanderfolgende Adressen.

Beschreibung:

Lesen des aktuellen Datumswerts.

- Mit dem Befehl DRD wird das aktuelle Datum gelesen (Tag, Monat, Jahr).
- Diese Daten werden in 3 aufeinanderfolgenden Operandenadressen (D) gespeichert.

Beispiel:

Wenn Eingang X0 eingeschaltet ist, werden Echtzeitdaten gelesen und in den Registern D0 ... gespeichert. D2.

Register	Funktion	Wert	Beispiel	
D0	Tag	1...31	26	26.01.22
D1	Monat	1...12	1	
D2	Jahr	21...99	22	

DWR	S	P	Aufzeichnungsdatum Daten

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Hinweis: Der Operand (S) belegt 3 aufeinanderfolgende Adressen.

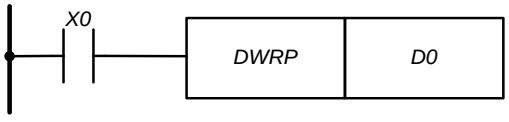
Beschreibung:

Der Befehl DWR wird verwendet, um die Datumsdaten (Tag, Monat, Jahr) zu ändern.

Die Daten werden aus 3 aufeinanderfolgenden Adressen entnommen, die in (S) angegeben sind.

Wenn die Werte in (S) den zulässigen Wertebereich überschreiten, tritt ein Fehler auf.

Beispiel:



Wenn die Eingangsbedingung erfüllt ist, wird die Echtzeituhr des Controllers auf die in den Registern D0 ... angegebenen Werte gesetzt. D2.

Register	Funktion	Wert	Beispiel	
D0	Tag	1...31	4	04.02.22
D1	Monat	1...12	2	
D2	Jahr	21...99	22	

LD#	S1 S2	D	Kontakttypische logische Operationen

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Hinweis:

- Das Symbol # ist &, |, ^.
- Bit-Operanden werden abhängig vom Befehlstyp als 16- oder 32-Bit-Werte genommen und für die weitere Verarbeitung in einen Integer-Datentyp konvertiert.

Beschreibung:

Durchführung der logischen Operation "AND", "OR" und " EXCLUSIVE OR" an den Operanden (S1) und (S2) und Einschalten des LD-Kontakts, abhängig vom Ergebnis der Operation.

Die LD#-Anweisungen im Programm befinden sich links und öffnen eine logische Verbindung oder sind Bedingungen für die Ausführung von Befehlen rechts.

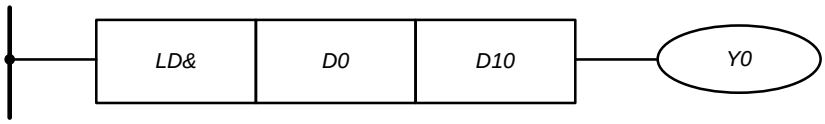
16-Bit-Befehle	32-Bit-Anweisungen	Kontakt geschlossen, wenn:	Kontakt offen, wenn:
LD&	DLD&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
LD|	DLD|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
LD^	DLD	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: logische Multiplikation (AND)

|: logische Addition (OR)

^: Exklusives ODER (XOR)

Beispiel:



AND#	S1 S2	D	Kontakttypische logische Operationen Serielle Verbindung

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Hinweis:

- Das Symbol # ist &, |, ^.
- Bit-Operanden werden abhängig vom Befehlstyp als 16- oder 32-Bit-Werte genommen und für die weitere Verarbeitung in einen Integer-Datentyp konvertiert.

Beschreibung:

Durchführung der logischen Operation "AND", "OR", "Exclusive OR " an den Operanden (S1) und (S2) und Einschalten des AND-Kontakts abhängig vom Ergebnis der Operation.

Die AND#-Anweisungen im Programm befinden sich nach den LD-Befehlen und erstellen eine logische AND-Verbindung.

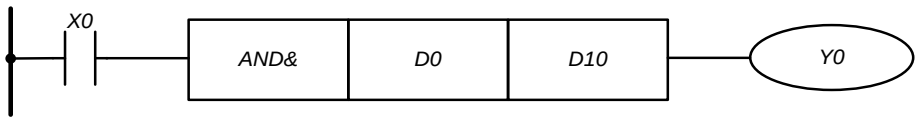
16-Bit-Befehle	32-Bit-Anweisungen	Kontakt geschlossen, wenn:	Kontakt offen, wenn:
AND	DAND&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
AND|	DAND|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
AND^	DAND^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: logische Multiplikation (AND)

|: logische Addition (OR)

^: Exklusives OR (XOR)

Beispiel:



OR#	S1 S2	D	Kontakttypische logische Operationen Parallelschaltung

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Hinweis:

- Das Symbol **#** ist &, |, ^.
- Bit-Operanden werden abhängig vom Befehlstyp als 16- oder 32-Bit-Werte genommen und für die weitere Verarbeitung in einen Integer-Datentyp konvertiert.

Beschreibung:

Durchführung der logischen Operation "AND", "OR", " Exclusive OR" an den Operanden (S1) und (S2) und Einschalten des OR-Kontakts abhängig vom Ergebnis der Operation.

Die OR#-Anweisungen im Programm befinden sich links parallel zum LD-Befehl und erstellen eine logische OR-Verbindung.

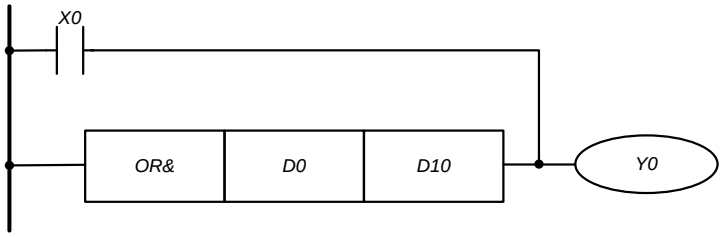
16-Bit-Befehle	32-Bit-Anweisungen	Kontakt geschlossen, wenn:	Kontakt offen, wenn:
OR&	DOR&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
OR|	DOR|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
OR^	DOR^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: logische Multiplikation (AND)

|: logische Addition (OR)

^: Exklusives ODER (XOR)

Beispiel:



LD*	S1 S2	D	Vergleichsoperationen für Kontakttypen

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Hinweis:

- Das Symbol # ist =, >, <, <>, ≤, ≥.
- Bit-Operanden werden abhängig vom Befehlstyp als 16- oder 32-Bit-Werte genommen und für die weitere Verarbeitung in einen Integer-Datentyp konvertiert.

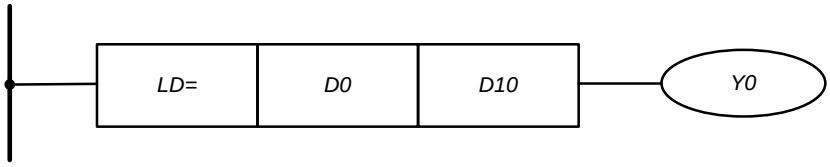
Beschreibung:

Vergleich der Werte der Operanden (S1) und (S2) und Einschalten eines LD-Kontakts, abhängig vom Ergebnis der Operation.

— Die LD *-Anweisungen im Programm befinden sich links und beginnen eine logische Verbindung oder sind Bedingungen für die Ausführung von Anweisungen rechts.
— Wenn das Vergleichsergebnis wahr ist, wird der LD-Kontakt eingeschaltet.
— Wenn das Ergebnis des Vergleichs falsch ist, wird der LD-Kontakt ausgeschaltet.

16-Bit-Befehle	32-Bit-Anweisungen	Kontakt geschlossen, wenn:	Kontakt offen, wenn:
LD=	DLD=	(S1) = (S2)	(S1) ≠ (S2)
LD>	DLD>	(S1) > (S2)	(S1) ≤ (S2)
LD<	DLD<	(S1) < (S2)	(S1) ≥ (S2)
LD<>	DLD<>	(S1) ≠ (S2)	(S1) = (S2)
LD<=	DLD<=	(S1) ≤ (S2)	(S1) > (S2)
LD>=	DLD>=	(S1) ≥ (S2)	(S1) < (S2)

Beispiel:



AND*	S1 S2	D	Vergleichsoperationen für Kontakttypen Serielle Verbindung

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Hinweis:

— Das Symbol # ist =, >, <, <>, ≤, ≥.

— Bit-Operanden werden abhängig vom Befehlstyp als 16- oder 32-Bit-Werte genommen und für die weitere Verarbeitung in einen Integer-Datentyp konvertiert.

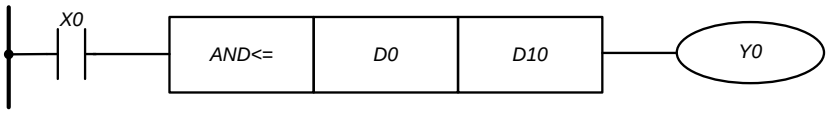
Beschreibung:

Vergleich der Werte der Operanden S1 und S2 und Einschalten des AND-Kontakts, abhängig vom Ergebnis der Operation.

— Die AND*-Befehle im Programm befinden sich nach den LD-Befehlen und stellen eine logische AND-Verbindung her. .
— Wenn das Vergleichsergebnis wahr ist, wird der AND-Kontakt eingeschaltet.
— Wenn das Ergebnis des Vergleichs falsch ist, wird der AND-Kontakt ausgeschaltet.

16-Bit-Befehle	32-Bit-Anweisungen	Kontakt geschlossen, wenn:	Kontakt offen, wenn:
AND=	DAND=	(S1) = (S2)	(S1) ≠ (S2)
AND>	DAND>	(S1) > (S2)	(S1) ≤ (S2)
AND<	DAND<	(S1) < (S2)	(S1) ≥ (S2)
AND<>	DAND<>	(S1) ≠ (S2)	(S1) = (S2)
AND<=	DAND<=	(S1) ≤ (S2)	(S1) > (S2)
AND>=	DAND>=	(S1) ≥ (S2)	(S1) < (S2)

Beispiel:



OR*	S1 S2	D	Vergleichsoperationen für Kontakttypen Parallelschaltung

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Hinweis:

- Das Symbol # ist =, >, <, <>, ≤, ≥.
- Bit-Operanden werden abhängig vom Befehlstyp als 16- oder 32-Bit-Werte genommen und für die weitere Verarbeitung in einen Integer-Datentyp konvertiert.

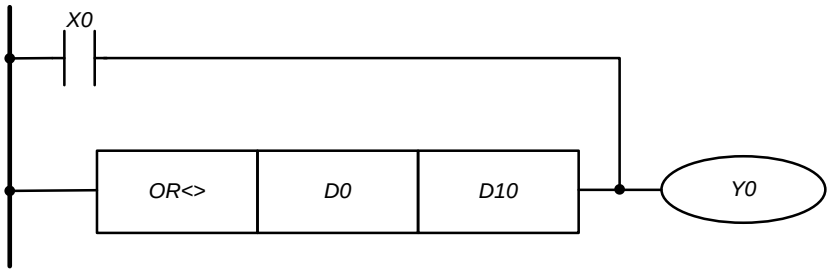
Beschreibung:

Vergleich der Werte der Operanden S1 und S2 und Einschalten des OR-Kontakts, abhängig vom Ergebnis der Operation.

- Die OR*-Anweisungen im Programm befinden sich links parallel zur LD-Anweisung und erstellen eine logische OR-Verbindung.
- Wenn das Vergleichsergebnis wahr ist, wird der OR-Kontakt eingeschaltet.
- Wenn das Ergebnis des Vergleichs falsch ist, wird der OR-Kontakt ausgeschaltet.

16-Bit-Befehle	32-Bit-Anweisungen	Kontakt geschlossen, wenn:	Kontakt offen, wenn:
OR=	DOR=	(S1) = (S2)	(S1) ≠ (S2)
OR>	DOR>	(S1) > (S2)	(S1) ≤ (S2)
OR<	DOR<	(S1) < (S2)	(S1) ≥ (S2)
OR<>	DOR<>	(S1) ≠ (S2)	(S1) = (S2)
OR<=	DOR<=	(S1) ≤ (S2)	(S1) > (S2)
OR>=	DOR>=	(S1) ≥ (S2)	(S1) < (S2)

Beispiel:



8. Anweisungen zur Steuerung des Schrittmotortreibers

Der Schrittmotortreiber wird durch Befehle gesteuert, die die Parameter für Drehung oder Bewegung festlegen. Alle Befehle sind in zwei Gruppen unterteilt: **RUN** und **MOVE**. Die Gruppe RUN dient zur Steuerung der aktuellen Geschwindigkeit des Antriebs und MOVE zur Steuerung der Bewegung. Um eine Drehung zu starten, nachdem ein Befehl ausgewählt und die Treiberparameter eingestellt wurden, wird der Befehl **SPIN** aufgerufen.

SPIN		P	Führen Sie die angegebene Bewegung aus
Adresse	**Objekttyp**	**Beschreibung des Modbus-Objekts**	
5100h	**Spulen**	Das Schreiben von „1“ in dieses Objekt (auch wenn es nicht zurückgesetzt ist) löst eine parametrisierte Bewegung aus, das Zurücksetzen wird ignoriert.	

Beschreibung:

Die Anweisung startet eine parametrisierte Drehung. Die Bewegungsparameter werden in den Serviceregistern eingestellt, die wie die Anweisungen des Schrittmotortreibers über das Modbus-Protokoll im RUN-Modus zugänglich sind. Die Anweisung SPIN hat eine niedrigere Priorität als xSTOP und xHIZ. Um Fehler zu vermeiden, wird empfohlen, die Flags BUSY_MOVE und BUSY_RUN zu überprüfen, bevor die Anweisung SPIN aufgerufen wird. Nachfolgend finden Sie eine detaillierte Beschreibung der Serviceregister des Treibers.

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
5000h	Halteregister	D357	SPEED	32	Die eingestellte (Ziel-)Motordrehzahl (in Impulsen pro Sekunde, pps) für die Befehle der RUN-Gruppe und die maximale Drehzahl für die Befehle der MOVE-Gruppe. Die untere Schwelle liegt bei 8 pps, die obere Grenze bei 120000 pps unter der Bedingung, dass FS_SW_EN zurückgesetzt ist. Wenn FS_SW_EN gesetzt ist, liegt die obere Grenze bei $SPEED_{max}=120000*2^{U\_STEP}$.
5002h	Halteregister	D359	MIN_SPEED	32	Die minimale Drehzahl für die Befehle der RUN-Gruppe. Für die MOVE-Gruppe ist dies ebenfalls die minimale Drehzahl, wenn das CMIN_SPD_EN-Flag nicht gesetzt ist. Wenn CMIN_SPD_EN gesetzt ist, wird die optimale minimale Drehzahl automatisch berechnet.
5004h	Halteregister	D361	ACC	16	Beschleunigung, pps^2.
5005h	Halteregister	D362	DEC	16	Verzögerung, pps^2.
5006h	Halteregister	D363	ABS	32	Aktuelle Position. Die Werteinheit entspricht der Verschiebung um einen Mikroschritt.

<table>
<tr><th rowspan="2">Adresse</th><th rowspan="2">Objekttyp</th><th colspan="2">Serviceregister</th><th rowspan="2">Größe (Bit)</th><th rowspan="2">Beschreibung</th></tr>
<tr><th>Zahl</th><th>Name</th></tr>
<tr><td>5008h</td><td>Eingaberegister</td><td>D365</td><td>U_POS</td><td>16</td><td>Die aktuelle Mikroschrittposition ist in vier vollen Schritten. Gibt die Position des Motorrotors relativ zu den Statorpolen an. Das Register ist schreibgeschützt.</td></tr>
<tr><td>5009h</td><td>Halteregister</td><td>D366</td><td>U_STEP</td><td>16</td><td>Mikroschrittbetrieb<table><tr><th>Registerwert</th><th>Mikroschrittbetrieb</th></tr><tr><td>0</td><td>1/1</td></tr><tr><td>1</td><td>1/2</td></tr><tr><td>2</td><td>1/4</td></tr><tr><td>3</td><td>1/8</td></tr><tr><td>4</td><td>1/16</td></tr><tr><td>5</td><td>1/32</td></tr><tr><td>7</td><td>1/128</td></tr><tr><td>8</td><td>1/256</td></tr></table>Der Wert „6“ ist ungültig.</td></tr>
<tr><td>500Ah</td><td>Halteregister</td><td>D374</td><td>DIR</td><td>16</td><td rowspan="2">Rotationsrichtung:
„1“ – vorwärts,
„0“ – rückwärts.</td></tr>
<tr><td>500Ah</td><td>Spulen</td><td colspan="3">DIR</td></tr>
<tr><td>500Bh</td><td>Halteregister</td><td>D377</td><td>FS_SPD_THR</td><td>32</td><td>Der Schwellenwert für den Übergang vom Mikroschritt- zum Vollschrittmodus, gemessen in Vollschritten pro Sekunde.</td></tr>
<tr><td>500Dh</td><td>Halteregister</td><td>D379</td><td>FS_SW_EN</td><td>16</td><td>Das Setzen des Objekts auf „1“ aktiviert die Morphing-Funktion –</td></tr>
</table>

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
500Dh	Spulen	FS_SW_EN			der Controller wechselt nach Erreichen der in FS_SPD_THR festgelegten Geschwindigkeit in den Vollschrittmodus. Diese Funktion ermöglicht ein höheres Drehmoment bei hohen Geschwindigkeiten. Das Zurücksetzen des Objekts auf „0“ deaktiviert diese Funktion (Morphing/Drehmomentverstärkung).
500Eh	Halteregister	D372	TARGET_POS	32	Die zu erreichende Zielposition. Der Wert entspricht der Verschiebung um einen Mikroschritt.
5010h	Halteregister	D376	Befehl	16	Ein Bewegungsbefehl an den Treiber (siehe Tabelle unten).

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
5011h	Halteregister	D375	SW_INPUT	16	Die Eingangsnummer für den Sensoranschluss – für die Befehle GOUNTIL_... und ...RELEASE. Werte 0...7. Achtung: Die Deklaration der Unterbrechung im Hauptprogramm ist für die verwendeten Eingänge notwendig. Die Behandlung der Unterbrechung kann leer bleiben. Beispiel: Ein Sensor ist mit dem Eingang X3 (IN3) verbunden, das Benutzerprogramm muss den folgenden Teil enthalten: *FEND* *I 1003* *IRET* *END*
5012h	Halteregister	D367	ACC_CUR	16	Beschleunigungsstrom, mA. Gültiger Wertebereich: RS 434540 - 150...1500; RS 434542 – 1000...5000 RS 434543 – 2800...10000.

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
5013h	Halteregister	D368	DEC_CUR	16	Bremsstrom, mA. Gültiger Wertebereich: RS 434540 - 150...1500; RS 434542 – 1000...5000 RS 434543 – 2800...10000.
5014h	Halteregister	D369	STEADY_CUR	16	Konstanter Strom, mA. Gültiger Wertebereich: RS 434540 - 150...1500; RS 434542 – 1000...4200 RS 434543 – 2800...8000
5015h	Halteregister	D370	HOLD_CUR	16	Haltestrom, mA. Gültiger Wertebereich: RS 434540 - 150...1500; RS 434542 – 1000...4200 RS 434543 – 2800...8000
5016h	Halteregister	D382	CMIN_SPD_EN	16	„1“ – Automatische Berechnung der Anfangs- und Endgeschwindigkeit für die Bewegungsbefehle der MOVE-Gruppe verwenden. „0“ – MIN_SPEED als Start- und Endgeschwindigkeit verwenden.
5017h	Halteregister	D380	ERROR_SET_HIZ	16	Die Bits dieses Registers bestimmen, welche Treiberfehler zum Abschalten des Motors (die Welle dreht sich frei) – dem HiZ-Zustand – führen müssen.

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
5017h	Spulen	TERMAL_OVER_CURRENT_ ERROR_SET_HIZ			0-Bit des Registers D380. Wenn das Bit gesetzt ist, führt ein Fehler TERMAL_ERROR (Überhitzung der Treiberschaltung – das Register D381) zum Abschalten des Motors (HiZ-Zustand).
5018h	Spulen	SOFTWARE_ERROR_SET_ HIZ			1-Bit des Registers D380. Wenn das Bit gesetzt ist, führt ein Fehler SOFTWARE_ERROR (der interne Fehler des Controllers – das Register D381) zum Abschalten des Motors (HiZ-Zustand).
5019h	Spulen	CMD_ERROR_SET_HIZ			2-Bit des Registers D380. Wenn das Bit gesetzt ist, kann ein Fehler CMD_ERROR einen eingehenden Befehl nicht verarbeiten – das Register D381) bewirkt das Abschalten des Motors (HiZ-Zustand).
501Ah	Spulen	DATA_ERROR_SET_HIZ			3-Bit des Registers D380. Ist das Bit gesetzt, führt ein Fehler DATA_ERROR (fehlerhafte Dateneingabe in ACC, DEC, U_STEP – das Register D381) zum Abschalten des Motors (HiZ-Zustand).

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
501Bh	Spulen	OUT_OF_LIM_MIN_SPD_ERROR_SET_HIZ			4-Bit des Registers D380. Wenn das Bit gesetzt ist, führt ein Fehler OUT_OF_LIM_MIN_SPD_ERROR_SET_HIZ (eingestellte Geschwindigkeit kleiner als Minimum – das Register D381) zum Deaktivieren des Motors (HiZ-Zustand).
501Ch	Spulen	OUT_OF_LIM_MAX_SPD_ERROR_SET_HIZ			5-Bit des Registers D380. Ist das Bit gesetzt, führt ein Fehler OUT_OF_LIM_MAX_SPD_ERROR_SET_HIZ (Überschreitung der maximal möglichen Geschwindigkeit – das Register D381) zum Abschalten des Motors (HiZ-Zustand).
501Dh	Spulen	UNREACHABLE_FS_SPD_ERROR_SET_HIZ			6-Bit des Registers D380. Wenn das Bit gesetzt ist, führt ein Fehler UNREACHABLE_FS_SPD_ERROR_SET_HIZ (Schwellwert für die volle Schrittgeschwindigkeit im Drehmoment-Boost-Modus nicht erreichbar – das Register D381) zum Abschalten des Motors (HiZ-Zustand).

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
501Eh	Spulen	NOT_APP_FS_PARAM_ ERROR_SET_HIZ			7-Bit des Registers D380. Wenn das Bit gesetzt ist, führt ein Fehler NOT_APP_FS_PARAM_ERROR_SET_ HIZ (Übergang von Drehmomentverstärkung ist nicht möglich, während der Motor dreht – das Register D381) zum Abschalten des Motors (HiZ-Zustand).
5027h	Halteregister	D381	ERROR_ CODE	16	Fehlercode. Siehe unten die Beschreibung der Registerbits (Fehler).
5027h	Spulen	TERMAL_OVER_CURRENT_ ERROR			0-Bit des Registers D381 TERMAL_OVER_ CURRENT_ERROR – Überhitzung der Treiberschaltung oder Überstrom in den Motorwicklungen.
5028h	Spulen	SOFT_ERROR			1-Bit des Registers D381 SOFTWARE_ERROR – interner Controller-Fehler.
5029h	Spulen	CMD_FEHLER			2-Bit des Registers D381 CMD_ERROR – Befehl konnte nicht verarbeitet werden.
502Ah	Spulen	DATA_ERROR			3-Bit des Registers D381 DATA_ERROR – Falsche Dateneingabe in ACC, DEC, U_STEP

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
502Bh	Spulen	OUT_OF_LIM_MIN_SPD_ERROR			4-Bit des Registers D381 OUT_OF_LIM_ MIN_SPD_ERROR – Dieeingestellte Geschwindigkeit ist kleiner als das Minimum.
502Ch	Spulen	OUT_OF_LIM_MAX_SPD_ERROR			5-Bit des Registers D381 OUT_OF_LIM_ MAX_SPD_ERROR - Überschreitung der maximal möglichen Geschwindigkeit.
502Dh	Spulen	UNREACHABLE_FS_SPD_ERROR			6-Bit des Registers D381 UNREACHABLE_FS_SPD_ERROR – Die volle Schrittfrequenzschwelle im Drehmoment-Boost-Modus konnte nicht erreicht werden.
502Eh	Spulen	NOT_APP_FS_PARAM_ERROR			7-Bit des Registers D381 . NOT_APP_FS_ PARAM_ERROR – Übergang von Drehmomentverstärkung ist nicht möglich, während der Motor sich dreht.
502F	Spulen	OVLO/UVLO_INTERNAL_ PROTECTION_ERROR			Fehler - die Spannung der internen Stromkreise liegt außerhalb des Standardbereichs.
5030	Spulen	VS_OUT_OF_RANGE_ERROR			Fehler - die Versorgungsspannung liegt außerhalb des zulässigen Bereichs.

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
5037h	Eingaberegister	D371	MOTOR_STATUS	16	Das Register zeigt den aktuellen Zustand des Motors und des Steuerungssystems. Die Beschreibung der Registerbits befindet sich unten.
5037h	Diskrete Eingänge	HIZ			0-Bit des Registers D371. HiZ-Zustand des Motors (Phasen sind stromlos).
5038h	Diskrete Eingänge	STOP			1-Bit des Registers D371. Haltemodus.
5039h	Diskrete Eingänge	ACCELERATING			2-Bit des Registers D371. Beschleunigung.
503Ah	Diskrete Eingänge	DECELERATING			3-Bit des Registers D371. Verzögerung.
503Bh	Diskrete Eingänge	STEADY			4-Bit des Registers D371. Drehung mit konstanter Geschwindigkeit.
503Ch	Diskrete Eingänge	BUSY_MOVE			5-Bit des Registers D371. Das Flag der Unmöglichkeit, die Befehle der MOVE-Gruppe anzuwenden.
503Dh	Diskrete Eingänge	BUSY_RUN			6-Bit des Registers D371. Die Flagge der Unmöglichkeit, die Befehle der RUN-Gruppe zu verwenden.

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
5047h	Eingaberegister	D383	CURRENT_SPD	32	Aktuelle Geschwindigkeit, pps. (Es wird empfohlen, das STEADY-Flag als Ereignis zum Erreichen einer gegebenen Geschwindigkeit zu verwenden.)
5048h	Halteregister	D385	EMERGENCY_DEC	32	Notbremsung, pps^2.
5100h	Spulen	SPIN (APPLY_CMD)			Das Setzen des Objekts auf „1“ oder die Anwendung des SPIN-Befehls aktiviert die Ausführung des im CMD-Register (D376) festgelegten Befehls mit den angegebenen Parametern.
5101h	Spulen	TORQUE (APPLY_CURRENT)			Das Setzen eines Objekts auf „1“ oder die Anwendung einer TORQUE-Anweisung wendet die aktuellen Werte ACC_CUR, DEC_CUR, RUN_CUR und HOLD_CUR für den Motor an.
5102h	Spulen	HSTOP (HARD_STOP)			Das Setzen eines Objekts auf „1“ oder die Anwendung des HSTOP-Befehls stoppt den Motor sofort und versetzt ihn in den Haltemodus.
5103h	Spulen	HHIZ (HARD_HIZ)			Das Setzen eines Objekts auf „1“ oder die sofortige Anwendung des HHIZ-Befehls versetzt den Motor in den HiZ-Zustand.

Adresse	Objekttyp	Serviceregister		Größe (Bit)	Beschreibung
		Zahl	Name		
5104h	Spulen	SSTOP (SLOW_STOP)			Das Setzen eines Objekts auf „1“ oder die Anwendung des SSTOP-Befehls stoppt den Motor gemäß der DEC und schaltet ihn dann in den Haltemodus.
5105h	Spulen	SHIZ (SLOW_HIZ)			Das Setzen eines Objekts auf „1“ oder die Anwendung des SHIZ-Befehls stoppt den Motor gemäß DEC und wechselt dann in den HiZ-Zustand.

Bewegungsbefehl (CMD-Register)

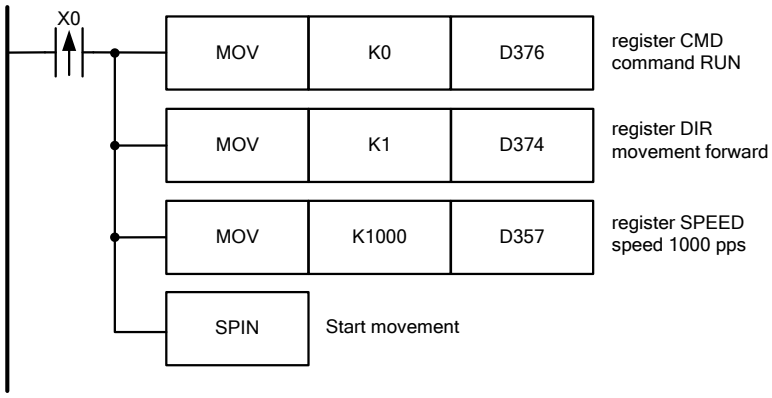
Wert	Gruppe	Name	Beschreibung
0	RUN	RUN	Rotation gemäß eingestellter Geschwindigkeit SPEED, Beschleunigung ACC, Verzögerung DEC, Richtung DIR.
1	MOVE	MOVE	Verschiebe um die angegebene Anzahl von Schritten TARGET_POS mit den gegebenen Bewegungsparametern SPEED, ACC, DEC und DIR.

2	MOVE	GOTO	Bewegen Sie sich mit den gegebenen Bewegungsparametern SPEED, ACC, DEC zur angegebenen Position TARGET_POS. DIR hängt von der aktuellen Position ab, der angegebene Wert wird nicht berücksichtigt.
3	MOVE	GOTO_DIR	Bewegen Sie sich mit den gegebenen Bewegungsparametern SPEED, ACC, DEC und DIR zur angegebenen Position TARGET_POS.
4	MOVE	GOHOME	Mit den gegebenen Bewegungsparametern SPEED, ACC, DEC zur Position "0" fahren. Der Befehl entspricht GOTO "0".
5	RUN	GOUNTIL_SLOWSTOP	Bewegung mit einer festgelegten Geschwindigkeit SPEED, Beschleunigung ACC, Richtung DIR bis der Sensor SW_INPUT an einer steigenden Flanke ausgelöst wird, mit einer anfänglichen Überprüfung des Eingangspegels, gefolgt von Abbremsen und Stoppen gemäß einem festgelegten DEC.

6	RUN	GOUNTIL_FRONT_SLOWSTOP	Bewegung mit einer eingestellten Geschwindigkeit SPEED, Beschleunigung ACC, Richtung DIR bis der Sensor SW_INPUT an einer steigenden Flanke auslöst, mit anschließendem Abbremsen und Stoppen gemäß einem eingestellten DEC.
7	RUN	GOUNTIL_HARDSTOP	Bewegung mit einer festgelegten Geschwindigkeit SPEED, Beschleunigung ACC, Richtung DIR bis der Sensor SW_INPUT an einer steigenden Flanke mit einer anfänglichen Überprüfung des Eingangspegels ausgelöst wird und dann in den Haltemodus wechselt.
8	RUN	GOUNTIL_FRONT_HARDSTOP	Bewegung mit einer festgelegten Geschwindigkeit SPEED, Beschleunigung ACC, Richtung DIR bis der Sensor SW_INPUT an einer steigenden Flanke ausgelöst wird und dann in den Haltemodus wechselt.

9	RUN	RELEASE	Bewegung mit einer festgelegten Geschwindigkeit SPEED, Beschleunigung ACC und Richtung DIR, bis der Sensor SW_INPUT an der fallenden Flanke auslöst, mit einer anfänglichen Überprüfung des Eingangspegels, und anschließend in den Haltemodus wechselt.
10	RUN	FRONT_RELEASE	Bewegung mit einer festgelegten Geschwindigkeit SPEED, Beschleunigung ACC, Richtung DIR bis der Sensor SW_INPUT an der fallenden Flanke auslöst und dann in den Haltemodus wechselt.

Beispiel:



Wenn die Eingangsbedingung erfüllt ist, werden der Bewegungsbefehl, die Richtung und die Drehgeschwindigkeit in den Serviceregistern eingestellt. Der folgende Befehl SPIN startet die Bewegung.

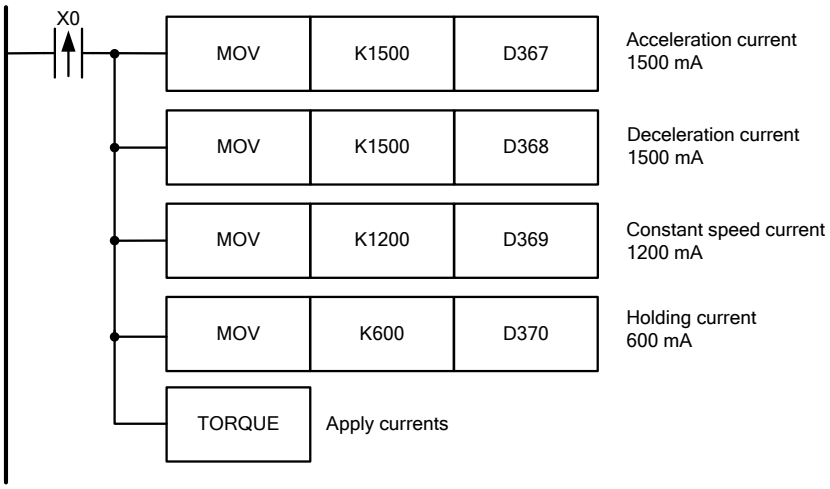
TORQUE		P	Die eingestellten Ströme an den Motor anlegen

5101h	Spulen	Das Schreiben von „1“ in ein Objekt (auch wenn es nicht zurückgesetzt wird) wendet die in den Serviceregistern eingestellten Stromwerte auf den Motor an. Zurücksetzen (Wert = „0“) wird ignoriert.

Beschreibung:

Durch Anwenden dieser Anweisung werden die Betriebsströme des Motors eingestellt, die in den Registern ACC_CUR (D367), DEC_CUR (D368), RUN_CUR (D369) und HOLD_CUR (D370) angegeben sind.

Beispiel:

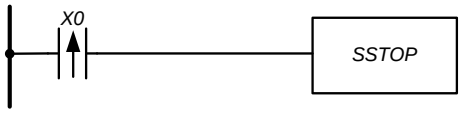


SSTOP		P	Der Motor stoppt gemäß dem DEC-Parameter und geht dann in den Haltemodus.
5104h	Spulen		Das Schreiben von „1“ in ein Objekt (auch wenn es nicht zurückgesetzt wird) bewirkt ein Motorbremsen gemäß DEC und geht dann in den Haltemodus über, ein Zurücksetzen wird ignoriert.

Beschreibung:

Anwenden der Bremsung gemäß DEC und anschließendes Umschalten in den Haltemodus. Diese Anweisung überschreibt den SPIN-Vorgang, hat die gleiche Priorität wie SHIZ, kann aber durch die Anweisungen HSTOP und HHIZ überschrieben werden.

Beispiel:

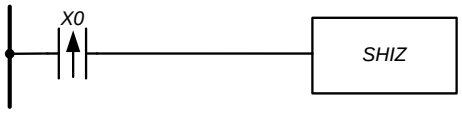


SHIZ		P	Der Motor stoppt gemäß dem DEC-Parameter und wechselt dann in den HiZ-Modus.
5105h	**Spulen**	Das Schreiben von „1“ in ein Objekt (auch wenn es nicht zurückgesetzt wird) bewirkt ein Motorbremsen gemäß DEC und geht dann in den HiZ-Modus, Reset wird ignoriert.	

Beschreibung:

Anwenden der Bremsung gemäß DEC, gefolgt von der Abschaltung der Wicklungen. Diese Anweisung überschreibt den SPIN-Vorgang, hat die gleiche Priorität wie SSTOP, kann aber durch die Anweisungen HSTOP und HHIZ überschrieben werden.

Beispiel:

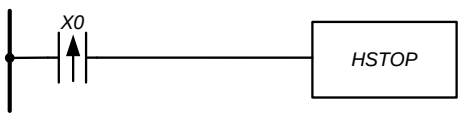


HSTOP		P	Hält den Motor sofort an und wechselt dann in den Haltemodus.
5102h	**Spulen**	Das Schreiben von „1“ in ein Objekt (auch wenn es nicht zurückgesetzt wird) stoppt den Motor sofort und wechselt dann in den Haltemodus, das Zurücksetzen wird ignoriert.	

Beschreibung:

Hält den Motor sofort an und wechselt dann in den Haltemodus. Diese Anweisung überschreibt SPIN, SSTOP, SHIZ und hat die gleiche Priorität wie HHIZ.

Beispiel:

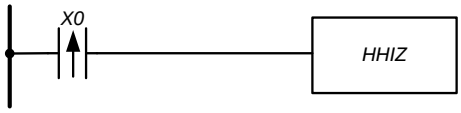


HHIZ		P	Hält den Motor sofort an und wechselt dann in den HiZ-Modus.
5103h	Spulen	Das Schreiben von „1“ in ein Objekt (auch wenn es nicht zurückgesetzt wird) stoppt den Motor sofort und wechselt dann in den HiZ-Modus, das Zurücksetzen wird ignoriert.	

Beschreibung:

Hält den Motor sofort an und wechselt dann in den HiZ-Modus (der Motor ist stromlos, die Welle dreht sich frei). Diese Anweisung überschreibt SPIN, SSTOP, SHIZ und hat die gleiche Priorität wie HSTOP.

Beispiel:



9. Kommunikationsparameter

Die Steuerung verfügt über eine USB- und eine RS-485-Schnittstelle, beide haben den gleichen Zugriff auf Register und Bit-Operanden. Die USB-Schnittstelle ist ein virtueller COM-Port (VCP), sie ist hauptsächlich für die Konfiguration der Steuerung und die Aufzeichnung von Benutzerprogrammen vorgesehen, daher hat sie feste Kommunikationsparameter: Modbus ASCII, ID 1, 115200 Baud, 7, Gerade, 1. Parametervariationen für RS-485 sind in Anhang Aim Abschnitt „RS-485 Schnittstellen-Kommunikationsparameter“ angegeben. Werkseitige Kommunikationsparameter für RS-485: Modbus RTU, ID 1, 9600 Baud, 8, gerade, 1.

9.1. Kommunikationseinstellungen für RS-485 ändern

Stellen Sie die erforderlichen Kommunikationsparameter gemäß dem Abschnitt „RS-485 Schnittstellen-Kommunikationsparameter“ von Anhang Aein. Damit die Änderungen wirksam werden, starten Sie das Gerät neu. Dies kann durch Aus- und Einschalten der Stromversorgung oder durch Einstellen des Coils 8101h (Reset) Objekts erfolgen.

Beispiel:

Es ist notwendig, die Kommunikationsparameter wie folgt zu ändern: Modbus RTU, ID 100, 128000 Baud, 8, Ungerade, 1. Alle möglichen Kombinationen von Kommunikationsparametern befinden sich in Anhang A.

Aktionsablauf:

1) Schreiben des Werts 100d in die Holding Registers 8103h – Geräteadresse (ID) auf 100 ändern.
2) Setzen von Coils 8100h – Protokollauswahl RTU.
3) Schreiben des Werts 128000d in die Holding Registers 8100h – Einstellen der Datenübertragungsgeschwindigkeit 128000 Baud.
4) Schreiben des Werts 1d in die Holding Registers 8102h – Auswahl des Paritätstyps Ungerade.
5) Setzen von Coils 8101h – Neustart der Steuerung.

9.2. Modbus-Protokoll

Es wird dringend empfohlen, die Protokollspezifikation auf der Website zu lesen. http://modbus.org/. Unterstützte Protokollfunktionen sind in der folgenden Tabelle aufgeführt:

			Funktion	Code
Datenzugriff	Diskrete Eingänge	1-Bit	Diskrete Eingänge lesen	02(02h)
	Spulen		Spulen lesen	01(01h)
			Einzelne Spule schreiben	05(05h)
			Mehrere Spulen schreiben	15(0Fh)
	Eingaberegister	16-Bit	Eingaberegister lesen	04(04h)
	Halteregister		Halte-Register lesen	03(03h)
			Schreiben Einzelnes Register	06(06h)
			Mehrere Register schreiben	16 (10h)
			Mehrere Register lesen/schreiben	23(17h)
			Masken-Schreibregister	22(16h)

Protokollfehlercodes sind in der folgenden Tabelle aufgeführt:

Fehlercode	Beschreibung
01(01h)	*UNGÜLTIGE FUNKTION* Der Funktionscode kann nicht verarbeitet werden.
02(02h)	*UNGÜLTIGE DATENADRESSE* Die Adresse des in der Anfrage angegebenen Registers ist nicht verfügbar.
03(03h)	*UNGÜLTIGER DATENWERT* Der Wert im Anforderungsdatenfeld ist ungültig.
04(04h)	*SERVER-GERÄTEAUSFALL* Bei der Durchführung der angeforderten Aktion ist ein nicht behebbarer Fehler aufgetreten.

Fehlercodes, die während der Verarbeitung von Protokollpaketen aufgezeichnet wurden, sind in den folgenden Tabellen aufgeführt.

Adresse	Typ	Größe	Beschreibung
E003h	Eingaberegister	16-Bit	Fehlercode bei der Verarbeitung des Modbus-Frames.
E003h	Spulen	1-Bit	Fehlerkennzeichen während des Austauschs über das Modbus-Protokoll.

Fehlercode	Beschreibung
0001h	Speicherzuordnungsfehler.
0002h	Prüfsummenfehler.
0003h	Beim Empfangen und Verarbeiten eines Broadcast-Pakets ist ein Fehler aufgetreten.
0004h	Rahmengrößenkonflikt.
0005h	Funktionsfehler (0Fh). Nicht alle Bits wurden überschrieben.
0006h	Funktionsfehler (10h). Nicht alle Register wurden überschrieben.
0007h	Funktionsfehler (17h). Nicht alle Register wurden überschrieben.
0008h	Verlorener Frame aufgrund eines DMA-Fehlers.
0009h	Frameverlust aufgrund eines überlaufenden Frame-Verarbeitungsstapels.

Wenn sich das Gerät am Ende der RS-485-Kommunikationsleitung befindet, schließen Sie einen Abschlusswiderstand an, indem Sie den Kippschalter neben dem RS-485-Anschluss einschalten, wie in Abb. 31 dargestellt.

Abb. 31. Anschluss des Abschlusswiderstands.

10. Einstellen der Echtzeituhr

Die Steuerung verfügt über eine Echtzeituhr, die von einer internen Quelle (CR2032-Batterie) gespeist wird, wodurch der Betrieb der Uhr bei ausgeschalteter Hauptstromversorgung gewährleistet ist. Dieselbe Batterie wird für den Betrieb von nichtflüchtigen Registern und den Sicherheitseinstellungen der Steuerungskommunikationsparameter verwendet. Die Anzeige **BAT** leuchtet auf, wenn die interne CR2032-Batterie fehlt oder bald ausfällt. Die Echtzeituhr kann über das Benutzerprogramm mit dem TWR-Befehl oder über das Modbus-Protokoll in der folgenden Reihenfolge eingestellt werden:

1) Deaktivieren Sie das automatische Überschreiben der Holding-Register 8110h...8112h, indem Sie die Coils 8110h zurücksetzen.
2) Aufzeichnen des aktuellen Zeitwerts in die Holding-Register 8110h, 8111h, 8112h Sekunden, Minuten bzw. Stunden (siehe Abschnitt «Uhrzeiteinstellung» im «Anhang A. Register der Steuerung»).
3) Stellen Sie einen neuen Zeitwert ein, indem Sie die Coils 8111h setzen.
4) Aktivieren Sie das automatische Überschreiben der Holding-Register 8110h...8112h, indem Sie die Coils 8110h setzen.

11. Ein Benutzerprogramm - Laden in die Steuerung und Lesen aus der Steuerung

11.1. Verfahren zum Hochladen/Herunterladen von Benutzerprogrammen

Die Steuerung verfügt über zwei Bereiche zum Herunterladen von Programmen: einen allgemeinen und einen speziellen Bereich.

Der allgemeine Bereich ist zum Laden eines Benutzerprogramms mit einer maximalen Länge von bis zu 59752 Zeilen vorgesehen (der Bereich ist standardmäßig leer). Die maximale Länge des Spezialbereichs beträgt 1926 Zeilen. Dieser Bereich enthält ein Programm zur Steuerung der Drehzahl eines Schrittmotors mithilfe eines Potentiometers, von Tasten und eines Encoders. Bei Bedarf kann dieser Bereich überschrieben werden.

Nachfolgend finden Sie ein Beispiel für ein Benutzerprogramm. Die Liste der an diesen Vorgängen beteiligten Register finden Sie in «Anhang A. Register der SteuerungAnhang A» im Abschnitt «Arbeiten mit ROM».

Benutzerprogramm in LD-Form:

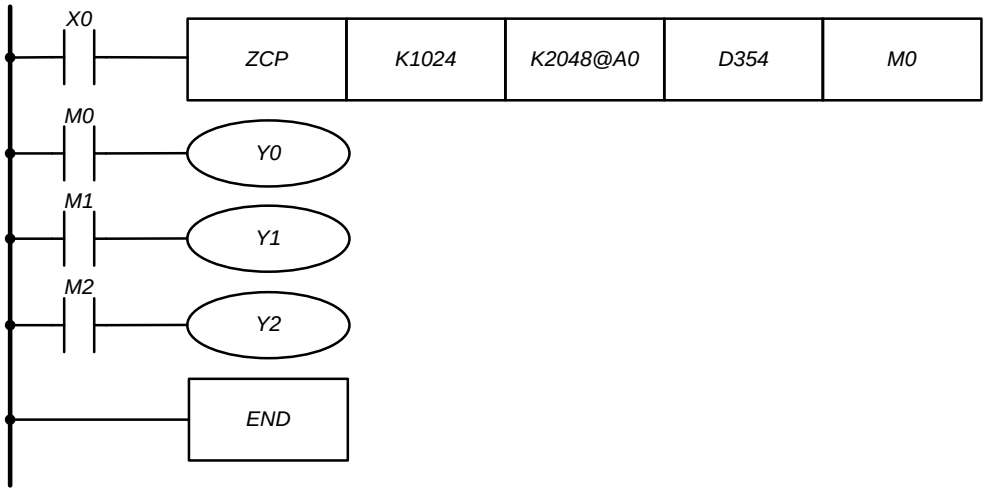


Abb. 32. Benutzerprogramm

Das in IL konvertierte Benutzerprogramm:

```
LD    X0                          ;Eingangszustandfür
                                  Zonenvergleichsoperation
ZCP   K1024  K2048@A0  D354  M0   ;Zonenvergleich, Bestimmung der Position
                                  des; Potentiometers SPEED (2)
LD    M0                          ;wenn der Wert im Register D354 kleiner als
                                  1024 ist, dann wird Y0 eingeschaltet,
                                  andernfalls – ausgeschaltet
OUT   Y0
LD    M1                          ;wenn der Wert im Register D354 größer oder
                                  gleich 1024 und kleiner oder gleich der
                                  Summe von 2048 und dem Wert von A0 ist,
                                  dann wird Y1 eingeschaltet, andernfalls –
                                  ausgeschaltet.
OUT   Y1
LD    M2                          Wenn der Wert im Register D354 größer ist
                                  als die Summe von 2048 und dem Wert von
                                  A0, dann wird Y2 eingeschaltet.
OUT   Y2
END                               ;Ende des Programms
```

Alle von der Steuerung unterstützten Befehlscodes sind im «Anhang B. Liste der Befehle» aufgeführt. Verwenden Sie diese beim Erstellen eines Benutzerprogramms oder verwenden Sie die mitgelieferte PC-Software zur Programmierung der Steuerung.

1) Stellen Sie sicher, dass sich die Steuerung im STOP-Modus befindet. Das Ändern des Benutzerprogramms im RUN-Modus ist nicht möglich. Um den RUN/STOP-Status der Steuerung zu überprüfen, lesen Sie den Wert der diskreten Eingänge F001h. Im STOP-Modus ist er zurückgesetzt, im RUN-Modus ist er gesetzt.
2) So lesen Sie das Programm aus der Steuerung: Prüfen Sie, ob der Lese-Schutz nicht aktiviert ist, bevor Sie ein Programm aus der Steuerung lesen. Es gibt zwei Lese-Schutzobjekte in der Steuerung: Coils F001h (zum Schutz eines Benutzerprogramms) und F002h (zum Schutz eines Serviceprogramms). Es ist nicht möglich, das Programm zu lesen, wenn der Schutz für das Programm gesetzt ist. Wenn der Schutz nicht gesetzt

ist, fahren Sie mit Schritt 5 fort, um das Programm aus der Steuerung zu lesen.

3) Bevor Sie ein neues Programm in die Steuerung schreiben, müssen Sie das vorherige löschen. Setzen Sie die Coils F003h, um das Benutzerprogramm zu löschen, oder die Coils F004h, um ein Serviceprogramm zu löschen. In diesem Beispiel wird das Hauptprogramm geschrieben, daher müssen die Coils F003h gesetzt werden.
4) Warten Sie nach dem Setzen der Coils F003h (oder F004h), bis die diskreten Eingänge F000h zurückgesetzt sind. Dies zeigt den Abschluss des Löschvorgangs und die Bereitschaft des ROMs für die weitere Arbeit an.
5) Nach dem Löschen des vorherigen Programms muss der Operationstyp festgelegt werden – Lesen oder Schreiben. Um ein neues Programm zu schreiben, setzen Sie die Coils F005h, um das Programm aus der Steuerung zu lesen – setzen Sie die Coils F005h zurück.
6) Wählen Sie den Programmtyp aus. Setzen Sie für das Benutzerprogramm die Coils F006h zurück, für das Serviceprogramm setzen Sie die Coils F006h.
7) Stellen Sie die Zeilennummer zum Schreiben/Lesen des Programms mithilfe der Holding-Register F100h ein. Die Nummerierung beginnt bei 0.Für den Lesevorgang fahren Sie mit Schritt 10 fort. Um ein neues Programm zu schreiben, setzen Sie seinen Wert auf 0.
8) Füllen Sie den Download-Sektor F300h ...F314 gemäß dem folgenden Beispiel:

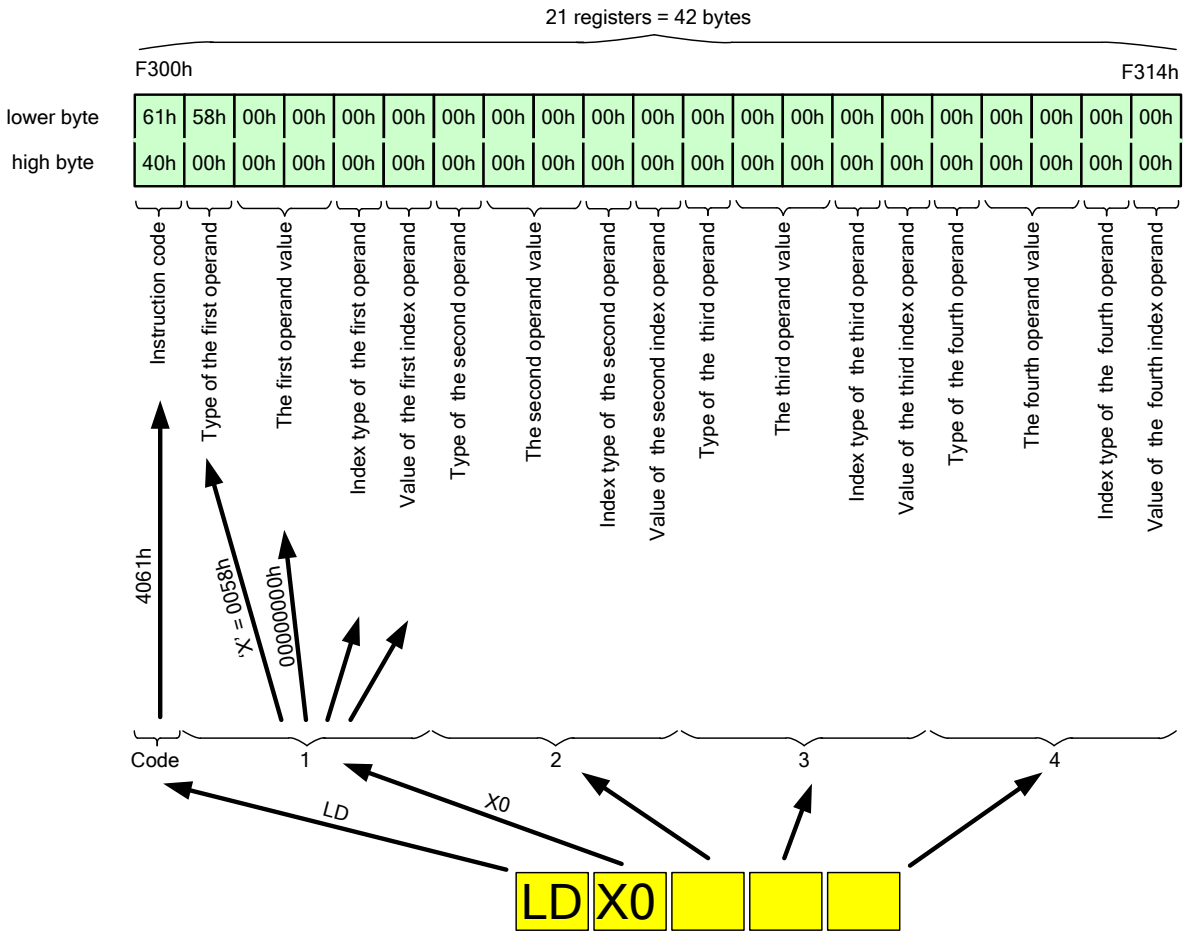


Abb. 33. Projizieren eines Befehls mit Operanden in den Adressraum des Modbus-Protokolls.

9) Das Setzen der Coils F000h startet den in den Coils F005h und F006h parametrierten Vorgang. In diesem Beispiel - Schreiben der ersten Zeile in den Controller-ROM. Durch Wiederholen der Schritte 7 - 9, Verschieben des Programms nach unten bis zum Ende und Inkrementieren der Holding-Register F100h wird das Programm im Controller aufgezeichnet.
   Als Beispiel ist unten die Bildung des Download-Sektors ab der zweiten Zeile des Programms dargestellt.

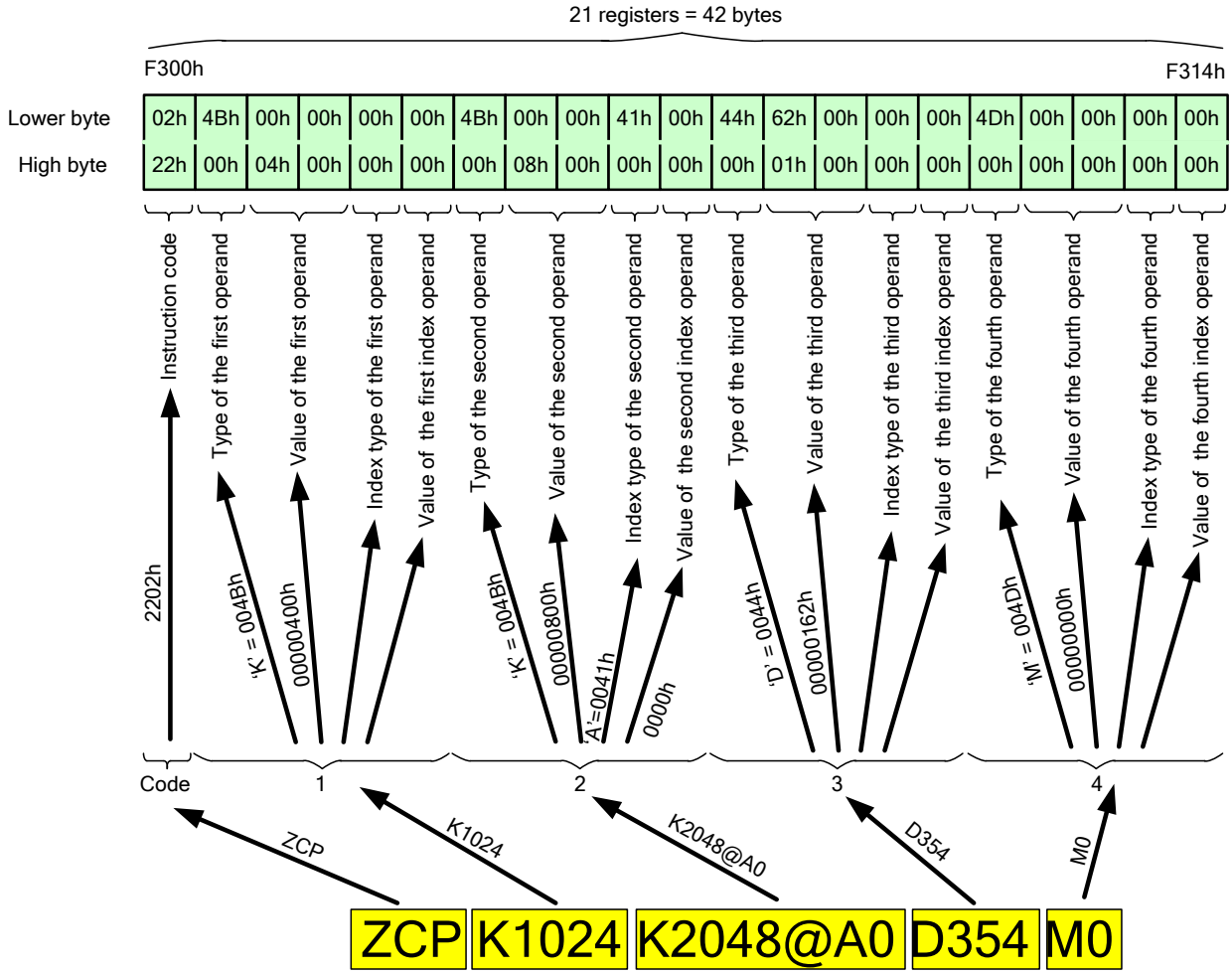


Abb. 34. Projizieren eines Befehls mit Operanden in den Adressraum des Modbus-Protokolls.

10) Zum Lesen des Programms ist der umgekehrte Vorgang erforderlich. Der Unterschied besteht darin, dass der Upload-Sektor die Adresse der Eingangsregister F200h ...F214h hat und der parametrierte Vorgang zuerst durch Setzen der Coils F000h, dann durch Lesen des Sektors und anschließend durch Inkrementieren der Zeilennummer durchgeführt wird, bis der END-Befehl eintrifft.

Die Operandentypcodes sind in der folgenden Tabelle aufgeführt:

Operand	Code
K	4Bh
H	48h
F	46h
X	58h
Y	59h
M	4Dh
T	54h
C	43h
A	41h
B	42h
D	44h
P	50h
I	49h

11.2. Blockweises Hochladen/Herunterladen eines Benutzerprogramms

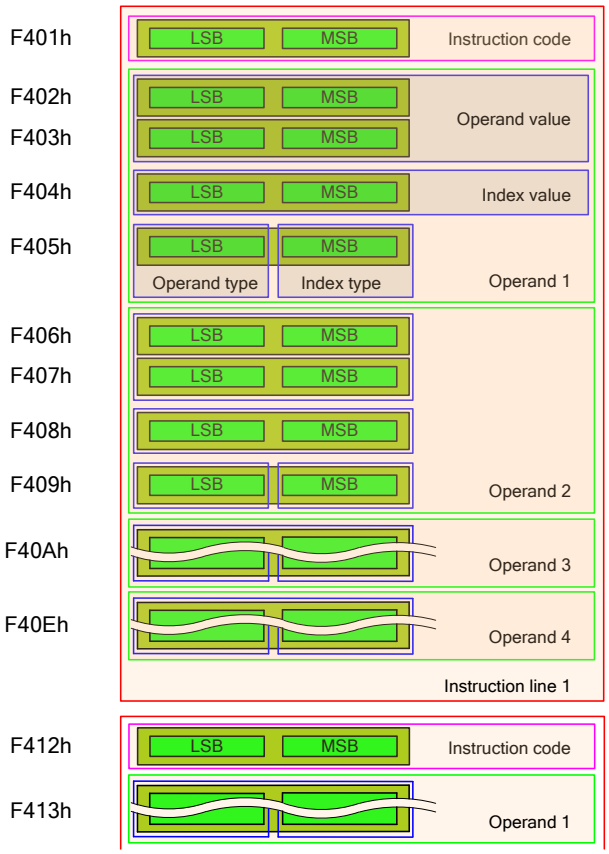
Ein Benutzerprogramm kann schneller gelesen und geschrieben werden, wenn das blockweise Hochladen/Herunterladen verwendet wird. Die Vorgehensweise beim blockweisen Hochladen/Herunterladen eines Benutzerprogramms ähnelt dem Abschnitt «Benutzerprogramm Hochladen/Herunterladen Vorgehensweise», jedoch mit den folgenden Unterschieden:

Ein Benutzerprogramm schreiben

Beginn des Verfahrens Verwenden Sie die Spulen F007h anstelle von F000h.

Sektor herunterladen	Download-Sektoradressen sind die Holding-Register F401h...F4FFh, der Sektor kann 1 bis 15 Befehlszeilen (Anweisungen) enthalten. Die Anzahl der zu schreibenden Zeilen wird in den Holding-Registern F400h angegeben.
Um ein Benutzerprogramm zu lesen	
Beginn des Verfahrens	Verwenden Sie die Spulen F007h anstelle von F000h.
Sektor hochladen	Adressen der Upload-Sektoren sind die Holding-Register F401h...F4FFh. Der Sektor kann 1 bis 15 Befehlszeilen (Anweisungen) enthalten. Die Anzahl der zu lesenden Zeilen wird in den Holding-Registern F400h angegeben. Nach Abschluss des Vorgangs zeigt F400h die tatsächlich gelesene Zeilenanzahl an. .
Kommandozeile	Für jede Befehlszeile im Download-/Upload-Sektor sind 34 Bytes reserviert.

Die Struktur des Block-Upload/Download-Sektors ist unten dargestellt:



. .
. .
. .

11.3. Fehlercodes, die bei der Arbeit mit ROM auftreten

Adresse	Typ	Größe	Beschreibung
E002h	Eingaberegister	16-Bit	Fehlercode beim Arbeiten mit ROM
E002h	Spulen		Kennzeichen eines Fehlers bei der Arbeit mit dem ROM.

Fehlercode	Beschreibung
0001h	Der Schreibschutz für das Hauptprogramm wurde nicht gesetzt.
0002h	Der Schreibschutz für das Serviceprogramm wurde nicht gesetzt.
0003h	Löschen des Hauptprogrammsektors fehlgeschlagen.
0004h	Löschen des Serviceprogramm-Sektors fehlgeschlagen.
0005h	Das Schreiben des Befehls in das Hauptprogramm ist fehlgeschlagen.
0006h	Das Schreiben des Befehls an das Serviceprogramm ist fehlgeschlagen.

12. Drehzahlregelungsmodus

Dieser Modus ist für die Steuerung der Drehzahl eines Schrittmotors mithilfe des eingebauten Potentiometers „SPEED“ (2), Tasten oder eines Encoders vorgesehen.

Um in den Drehzahlregelungsmodus zu gelangen, versetzen Sie den Controller in den STOP-Zustand und stellen Sie dann mit der Modusauswahltaste den SPD-Modus ein. Bauen Sie die in Abb. 35. – Anschluss der Bedienelementedargestellte Schaltung zusammen und schließen Sie sie an den Controller an.

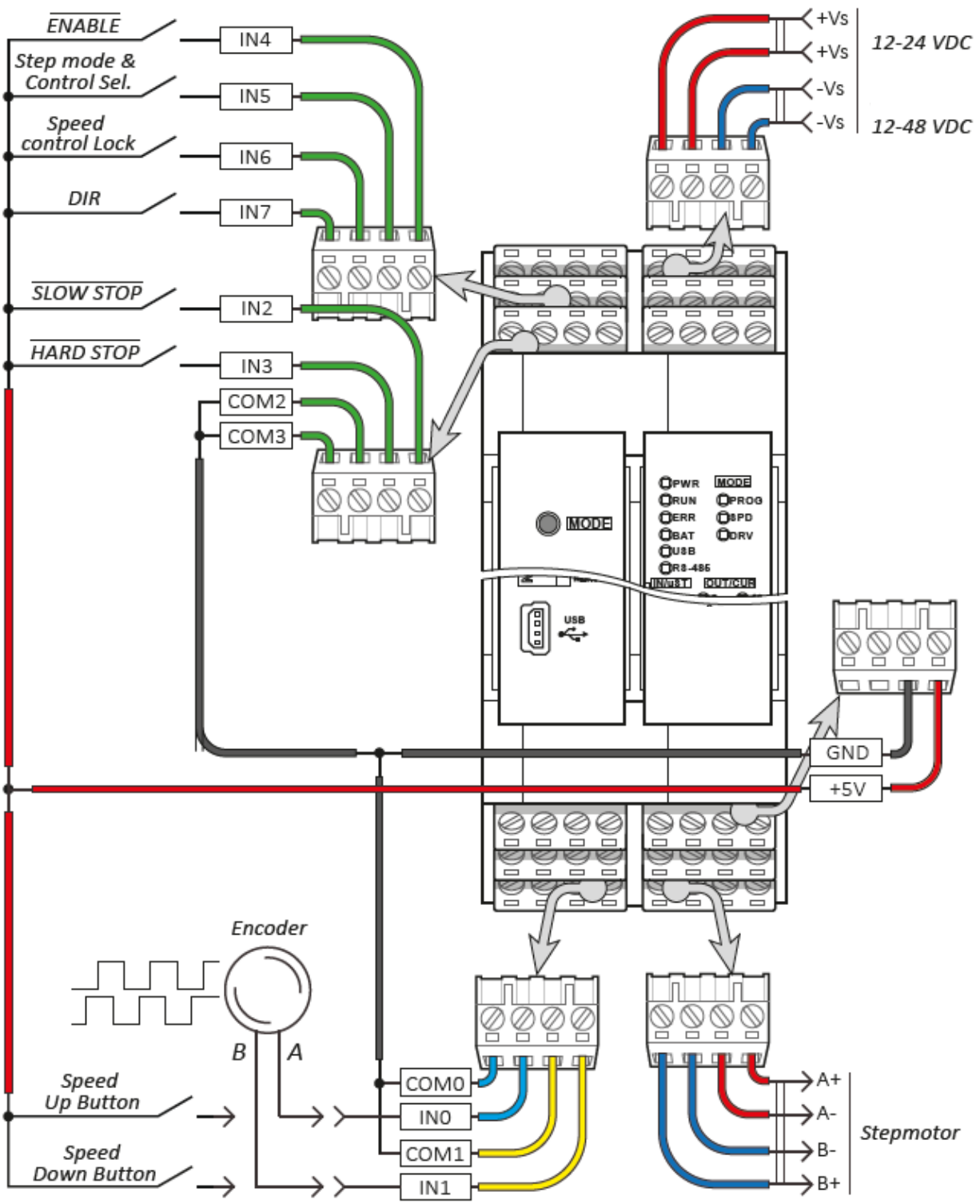


Abb. 35. Anschluss der Bedienelemente

Achtung

Wird der Controller im **RUN**-Modus eingeschaltet, während die Schalter SLOW STOP und HARD STOP geschlossen und der Schalter ENABLE geöffnet sind, **führt dies zur Motorrotation**. Um eine unkontrollierte Rotation zu vermeiden, drehen Sie das Potentiometer „SPEED“ auf die minimale Position oder ändern Sie die Position eines der oben genannten Schalter auf die im Diagramm angegebene entgegengesetzte Position.

Wählen Sie im Zustand **RUN** die gewünschte Mikroschrittweite durch Drücken der entsprechenden Taste. Die Methode der Drehzahlregelung wird über den Eingang IN5 ausgewählt, weitere Details in der folgenden Tabelle.

LED-Anzeige OUT0...7	Mikroschrittbetrieb
OUT0	1/1
OUT1	1/2
OUT2	1/4
OUT3	1/8
OUT4	1/16
OUT5	1/32
OUT6	1/128
OUT7	1/256

LED-Anzeige OUT10...11	Drehzahlregelungsquelle
Beide sind ausgeschaltet	Die Geschwindigkeit wird durch das Potentiometer «SPEED» eingestellt.
OUT10	Die Geschwindigkeit wird durch die Tasten „Erhöhen“ und „Verringern“ (IN0 und IN1) eingestellt. Der Geschwindigkeitsänderungs-Schritt wird durch das Potentiometer «SPEED» festgelegt.
OUT11	Die Geschwindigkeit wird durch einen Encoder eingestellt, der an die Eingänge IN0 und IN1 angeschlossen ist. Die Geschwindigkeitsänderung für ein Encoder-Ereignis wird durch das Potentiometer «SPEED» festgelegt.

Wenn die Drehzahländerungssperre aktiviert ist, reagiert der Controller nicht mehr auf die Drehzahlregler. Diese Option soll verhindern, dass das Potentiometer, der Encoder und die Tasten versehentlich mechanisch beeinflusst werden.

DIR– ändert die Drehrichtung des Motors.

ENABLE– steuert die Bestromung der Schrittmotorphasen.

$\overline{\textbf{HARD STOP}}$– die Öffnung des Stromkreises stoppt den Motor sofort und versetzt ihn in den Haltemodus. Der Haltestrom beträgt 50 % des Arbeitsstroms. Der Wert des Arbeitsstroms wird durch das Potentiometer (**1**) vom Minimal- bis zum Maximalwert für das Modell eingestellt.

$\overline{\textbf{SLOW STOP}}$– Öffnen des Stromkreises bewirkt, dass der Motor gemäß der durch das Potentiometer (0) eingestellten Verzögerung stoppt (der Beschleunigungswert wird ebenfalls durch das Potentiometer (0) eingestellt).

Der Code des Serviceprogramms ist in Anhang D. Code des Serviceprogramms „Schrittmotordrehzahlregelung“ angegeben. Der Code kann an spezifische Anforderungen angepasst werden.

13. Schritt/Dir-Impulspositions-Steuerungsmodus

Der Controller bietet einen Impulspositions-Steuerungsmodus durch Schrittimpulssignale STEP (die Eingänge STEP +, STEP-) und Richtungssignal DIR (die Eingänge DIR +, DIR-). Der Eingang ENABLE + steuert die Bestromung der Motorphasen. INVERT_ENABLE + invertiert das ENABLE-Signal. Der Ausgang FAULT zeigt Alarmzustände an: Überstrom und Überhitzung oder fehlende Schritte aufgrund dieser beiden Gründe (Abb. 2).

Um den Controller in den Impulspositions-Steuerungsmodus zu schalten, schalten Sie ihn zuerst in den Zustand **STOP** und wechseln Sie dann mit der Modusauswahltaste in den Modus **DRV**. In diesem Zustand sind die Motorphasen stromlos. Wählen Sie die erforderliche Mikroschrittweite, den Arbeitsstrom und den Haltestrom. (Der Übergang in den Modus erfolgt eine Sekunde nach der Erkennung des letzten Schrittsignals an der Vorderflanke des Impulses).

Die Mikroschrittweite wird durch das Potentiometer SPEED eingestellt, die Werte des Arbeitsstroms – durch das Potentiometer (0), der Haltestrom – durch

das Potentiometer (1). Die eingestellten Parameter werden auf der LED-Anzeige angezeigt, weitere Details in den folgenden Tabellen:

OUT0	OUT1	OUT2	OUT3	Betriebsstrom		
OUT4	OUT5	OUT6	OUT7	Haltestrom		
				RS 434540	RS 434542	RS 434543
•				150 мА	1000 мА	2800 мА
	•			245 мА	1285 мА	3310 мА
•	•			340 мА	1570 мА	3830 мА
		•		440 мА	1860 мА	4340 мА
•		•		535 мА	2140 мА	4860 мА
	•	•		630 мА	2430 мА	5370 мА
•	•	•		730 мА	2710 мА	5885 мА
			•	825 мА	3000 мА	6400 мА
•			•	920 мА	3285 мА	6910 мА
	•		•	1020 мА	3570 мА	7430 мА
•	•		•	1115 мА	3860 мА	7940 мА
		•	•	1210 мА	4140 мА	8460 мА
•		•	•	1310 мА	4430 мА	8970 мА
	•	•	•	1400 мА	4710 мА	9485 мА
•	•	•	•	1500 мА	5000 мА	10000 мА

LED-Anzeige IN0...7	Mikroschrittbetrieb
IN0	1/1
IN1	1/2
IN2	1/4
IN3	1/8
IN4	1/16
IN5	1/32
IN6	1/128
IN7	1/256

Wenn sich der Controller im Zustand **RUN** befindet, sind die oben genannten Parameter festgelegt und werden nach dem Ausschalten gespeichert. Verwenden Sie die Ein- und Ausgänge des Controllers gemäß der Pinbelegungstabelle (Abb. 2).

14. Benutzerprogramm-Steuerungsmodus

Der Controller bietet einen Steuerungsmodus gemäß einem Benutzerprogramm und wird über Modbus-Befehle gesteuert. Der Controller zeigt diesen Steuerungsmodus durch die LED-Anzeige **PROG** an. Ein Benutzerprogramm kann in den Controllerspeicher gesendet werden, wenn sich der Controller im Zustand **STOP** befindet. Nach dem Umschalten in den Zustand **RUN** beginnt der Controller mit der Ausführung des Benutzerprogramms. Es ist auch möglich, den Zustand des Controllers, des Benutzerprogramms, der physischen Ausgänge und des Schrittmotortreibers über die RS-485-Schnittstelle mithilfe des Modbus-Protokolls zu steuern und den Status der physischen Eingänge zu überwachen.

Beispiele von Benutzerprogrammen, die die grundlegende Funktionalität der Steuerung demonstrieren, sind im Anhang C. Beispiele von Benutzerprogrammenbeschrieben.

Anhang A. Register der Steuerung

Adresse	Typ	Größe	Beschreibung
RS-485-Schnittstellen-Kommunikationsparameter			
0x8100	Spulen	-	Kommunikationsprotokoll-Auswahl Zurücksetzen — Modbus ASCII. Einstellung — Modbus RTU. Änderungen werden nach dem Neustart des Controllers wirksam.
0x8100	Halteregister	32-Bit	Baudrate. Zulässige Werte: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Änderungen werden nach dem Neustart des Controllers wirksam.
0x8102	Halteregister	16-Bit	Parität: 0 – NONE 1 – ODD 2 – EVEN
			*Die folgenden Datenübertragungsparameter sind verfügbar.:* *Modbus RTU:* ■ *8-bit / EVEN / 1 STOP* ■ *8-bit / ODD / 1 STOP* ■ *8-bit / NONE / 2 STOP* *Modbus ASCII:* ■ *7bit / EVEN / 1 STOP* ■ *7bit / ODD / 1 STOP* *Stopp-Bit-Parameter werden automatisch eingestellt.*
0x8103	Halteregister	16-Bit	ID des Controllers (Geräteadresse). Zulässige Werte: 1.. .247.
Takteinstellung			
0x8110	Halteregister	16-Bit	Sekunden. Zulässige Werte: 0.. .59.
0x8111	Halteregister	16-Bit	Minuten. Gültige Werte: 0.. .59.
0x8112	Halteregister	16-Bit	Stunden. Gültige Werte: 0.. .23.

Adresse	Typ	Größe	Beschreibung
0x8110	Spulen	-	Automatische Registeraktualisierung. Setzen — in den Registern 0x8110 - 0x8112 den tatsächlichen Wert der Zeit. Zurücksetzen — die automatische Aktualisierung der Daten ist deaktiviert, die Aufzeichnung von Benutzerwerten ist erlaubt.
0x8111	Spulen	-	Das Einstellen des Objekts setzt die Zeit aus den Registern 0x8110 - 0x8112. Nach dem Setzen des Objekts ist es erlaubt, die automatische Aktualisierung wieder einzuschalten.
Datumseinstellung			
0x8113	Bestände Register	16-Bit	Tag. Gültige Werte: 1...31
0x8114	Bestände Register	16-Bit	Monat Gültige Werte: 1...12
0x8115	Bestände Register	16-Bit	Jahr Gültige Werte: 21...99
		Das Einstellen des Datums erfolgt analog zum Einstellen der Uhrzeit mit einem vorherigen Zurücksetzen der Coils 8110h (siehe Abschnitt 10).	
0x8112	Spulen	-	Das Setzen des Objekts setzt das Datum aus den Registern 0x8110 - 0x8112. Nach dem Setzen des Objekts ist es erlaubt, die automatische Aktualisierung wieder einzuschalten.
Zusätzlich			
0x8101	Spulen	-	Das Setzen des Objekts verursacht einen Neustart des Controllers.

Adresse	Typ	Größe	Beschreibung
0xF001	Halteregister	16-Bit	Controller-Betriebsmodus: Benutzerprogramm (PROG), Serviceprogramm (SPD), Treibermodus (DRV). Das Ändern des Betriebsmodus des Controllers ist nur im STOP-Zustand möglich. 0 - Benutzerprogramm. 1 - Serviceprogramm. 2 - Treibermodus.
Arbeiten mit ROM			
0xF001	Diskrete Eingänge	-	RUN/STOP-Kippschalterzustand. ROM-Operationen sind nicht möglich, wenn sich die Steuerung im RUN-Zustand befindet. Zurücksetzen —STOP-Zustand. —RUN-Status setzen.
0xF000	Diskrete Eingänge	-	Anzeige des ROM-Status. Zurücksetzen — ROM ist betriebsbereit. Setzen — ROM ist belegt.
0xF000	Spulen	-	Steuerelement für das zeilenweise Lesen/Schreiben eines Benutzerprogramms. Das Setzen des Objekts startet die in den Objekten 0xF005 und 0xF006 parametrierte Operation.
0xF001	Spulen	-	Leseschutz des Haupt (benutzer) programms. Setzen – nicht geschützt. Zurücksetzen — Schreibschutz ist gesetzt. Der Versuch, das Objekt zu setzen, führt zum Löschen des Hauptprogramms.
0xF002	Spulen	-	Leseschutz des Serviceprogramms. Setzen – nicht geschützt. Zurücksetzen — Schreibschutz ist gesetzt. Der Versuch, das Objekt zu setzen, führt zum Löschen des Serviceprogramms.
0xF003	Spulen	-	Löschen des Hauptprogramms. Das Setzen des Objekts startet den Löschvorgang des Hauptprogramms. Das Zurücksetzen des Objekts wird ignoriert.

Adresse	Typ	Größe	Beschreibung
0xF004	Spulen	-	Löschen des Serviceprogramms. Das Setzen des Objekts startet den Löschvorgang des Serviceprogramms. Das Zurücksetzen des Objekts wird ignoriert.
0xF005	Spulen	-	Auswahl des Operationstyps. Zurücksetzen — lesen. Setzen — schreiben.
0xF006	Spulen	-	Programmauswahl (Hauptprogramm/Dienstprogramm) Zurücksetzen — Hauptprogramm. Einstellen — Service.
0xF007	Spulen	-	Steuerelement für blockweises Lesen/Schreiben eines Anwenderprogramms (siehe Abschnitt 11.2). Das Setzen des Objekts startet die in den Objekten 0xF005 und 0xF006 parametrierte Operation.
0xF100	Halteregister	16-Bit	Die Zeilennummer im Programm, die gelesen oder überschrieben werden soll (0 ... 59753 - für das Hauptprogramm, 0 ... 1927 - für den Dienst).
Zeilenweises ROM-Lesesektor			
0xF200	Eingaberegister	16-Bit	Befehlscodes
0xF201	Eingaberegister	16-Bit	Typ des ersten Operanden
0xF202	Eingaberegister	32-Bit	Wert des ersten Operanden
0xF204	Eingaberegister	16-Bit	Indextyp des ersten Operanden
0xF205	Eingaberegister	16-Bit	Wert des ersten Index-Operanden
0xF206	Eingaberegister	16-Bit	Typ des zweiten Operanden
0xF207	Eingaberegister	32-Bit	Wert des zweiten Operanden

Adresse	Typ	Größe	Beschreibung
0xF209	Eingaberegister	16-Bit	Indextyp des zweiten Operanden
0xF20A	Eingaberegister	16-Bit	Wert des zweiten Index-Operanden
0xF20B	Eingaberegister	16-Bit	Typ des dritten Operanden
0xF20C	Eingaberegister	32-Bit	Wert des dritten Operanden
0xF20E	Eingaberegister	16-Bit	Indextyp des dritten Operanden
0xF20F	Eingaberegister	16-Bit	Wert des dritten Index-Operanden
0xF210	Eingaberegister	16-Bit	Typ des vierten Operanden
0xF211	Eingaberegister	32-Bit	Wert des vierten Operanden
0xF213	Eingaberegister	16-Bit	Indextyp des vierten Operanden
0xF214	Eingaberegister	16-Bit	Wert des vierten Index-Operanden
Zeilenweiser ROM-Schreibsektor			
0xF300	Halteregister	16-Bit	Befehlscodes
0xF301	Halteregister	16-Bit	Typ des ersten Operanden
0xF302	Halteregister	32-Bit	Wert des ersten Operanden
0xF304	Halteregister	16-Bit	Indextyp des ersten Operanden
0xF305	Halteregister	16-Bit	Wert des ersten Index-Operanden
0xF306	Halteregister	16-Bit	Typ des zweiten Operanden

Adresse	Typ	Größe	Beschreibung
0xF307	Halteregister	32-Bit	Wert des zweiten Operanden
0xF309	Halteregister	16-Bit	Indextyp des zweiten Operanden
0xF30A	Halteregister	16-Bit	Wert des zweiten Index-Operanden
0xF30B	Halteregister	16-Bit	Typ des dritten Operanden
0xF30C	Halteregister	32-Bit	Wert des dritten Operanden
0xF30E	Halteregister	16-Bit	Indextyp des dritten Operanden
0xF30F	Halteregister	16-Bit	Wert des dritten Index-Operanden
0xF310	Halteregister	16-Bit	Typ des vierten Operanden
0xF311	Halteregister	32-Bit	Wert des vierten Operanden
0xF313	Halteregister	16-Bit	Indextyp des vierten Operanden
0xF314	Halteregister	16-Bit	Wert des vierten Index-Operanden
Lesesektor/Schreibsektor für das Hochladen/Herunterladen von Blöcken eines Benutzerprogramms ***(15 Befehlszeilen, 17 Register pro Zeile)***			
Zeile 1 des Lese-/Schreibsektors			
0xF401	Halteregister	16-Bit	Befehlscodes der Zeile 1 des Lese-/Schreibsektors
Operand 1 der Zeile 1 des Lese-/Schreibsektors			
0xF402	Halteregister	32-Bit	Wert des Operanden 1 der Zeile 1 des Lese-/Schreibsektors
0xF404	Halteregister	16-Bit	Wert des Operanden 1 Index der Zeile 1 des Lese-/Schreibsektors

Adresse	Typ	Größe	Beschreibung
0xF405	Halteregister	16-Bit	8 Bits LSB – Operand 1 Typ – Zeile 1 des Lese-/Schreibsektors 8 Bit MSB – Indextyp des Operanden 1 des Lese-/Schreibsektors
. . .	. . .	. . .	. . .
Operand 4 der Zeile 1 des Lese-/Schreibsektors			
0xF40E	Halteregister	32-Bit	Wert des Operanden 4 der Zeile 1 des Lese-/Schreibsektors
0xF404	Halteregister	16-Bit	Wert des Operanden 4 Index der Zeile 1 des Lese-/Schreibsektors
0xF405	Halteregister	16-Bit	8 Bits LSB – Operand 4 Typ – Zeile 1 des Lese-/Schreibsektors 8 Bits MSB – Indextyp des Operanden 4 des Lese-/Schreibsektors
. . .	. . .	. . .	. . .
Zeile 15 des Lese-/Schreibsektors			
0xF4EF	Halteregister	16-Bit	Befehlscodes der Zeile 15 des Lese-/Schreibsektors
Operand 1 der Zeile 15 des Lese-/Schreibsektors			
0xF4F0	Halteregister	32-Bit	Wert des Operanden 1 der Zeile 15 des Lese-/Schreibsektors
0xF4F2	Halteregister	16-Bit	Wert des Operanden 1 Index der Zeile 15 des Lese-/Schreibsektors
0xF4F3	Halteregister	16-Bit	8 Bits LSB – Operand 1 Typ – Zeile 15 des Lese-/Schreibsektors 8 Bit MSB – Indextyp des Operanden 1 des Lese-/Schreibsektors
. . .	. . .	. . .	. . .
Operand 4 der Zeile 15 des Lese-/Schreibsektors			
0xF4FC	Halteregister	32-Bit	Wert des Operanden 4 der Zeile 15 des Lese-/Schreibsektors
0xF4FE	Halteregister	16-Bit	Wert des Operanden 4 Index der Zeile 15 des Lese-/Schreibsektors

Adresse	Typ	Größe	Beschreibung
0xF4FF	Halteregister	16-Bit	8 Bits LSB – Operand 4 Typ – Zeile 15 des Lese-/Schreibsektors 8 Bit MSB – Indextyp des Operanden 4 des Lese-/Schreibsektors.
Fehler			
0xE000	Spulen	-	Das Setzen des Objekts setzt alle Fehlertypen zurück, die für den aktuellen Zustand des Controllers gültig sind.
0xE000	Diskrete Eingänge	-	Allgemeiner Fehler. Er wird immer gesetzt, wenn mindestens einer der Fehlertypen von 0xE001 bis 0xE004 auftritt.
0xE001	Diskrete Eingänge	-	Der gesetzte Zustand des Objekts zeigt eine entladene CR2032-Batterie im Controller an. Ein Austausch ist erforderlich.
0xE002	Diskrete Eingänge	-	Der gesetzte Zustand des Objekts zeigt einen Fehler während des ROM-Betriebs an. Der Fehlercode ist in den Eingangsregistern 0xE002 angegeben.
0xE003	Diskrete Eingänge	-	Der gesetzte Zustand des Objekts zeigt einen Fehler während des Austauschprozesses unter Verwendung des Modbus-Protokolls an. Der Fehlercode ist in den Eingangsregistern 0xE003 angegeben.
0xE004	Diskrete Eingänge	-	Der gesetzte Zustand des Objekts zeigt einen Fehler während der Ausführung des Benutzerprogramms an. Der Fehlercode ist in den Eingangsregistern 0xE004 angegeben. Die fehlerverursachende Zeile des Programms wird in 0xE084 angezeigt.
0xE002	Eingaberegister	16-Bit	ROM-Betriebsfehlercode.
0xE003	Eingaberegister	16-Bit	Modbus-Protokoll Fehlercode.
0xE004	Eingaberegister	16-Bit	Benutzerprogramm-Fehlercode.

Adresse	Typ	Größe	Beschreibung
0xE084	Eingaberegister	16-Bit	Die Nummer der Zeile, die den Fehler im Benutzerprogramm verursacht hat (Nummerierung beginnt bei 0, siehe Beschreibung der Coils 0xF100 oben).
Zugriff auf Programmoperanden			
Diskrete Ausgänge			
0x1000	Diskrete Eingänge	-	Diskretes Ausgangssignal Y0
0x1001	Diskrete Eingänge	-	Diskretes Ausgangssignal Y1
. . .	. . .	. . .	. . .
0x107F	Diskrete Eingänge	-	Diskretes Ausgang Y177
Zustand der diskreten physischen Eingänge			
0x2000	Diskrete Eingänge	-	Diskreter Eingang X0
. . .	. . .	. . .	. . .
0x2007	Diskrete Eingänge	-	Diskreter Eingang X7
Diskrete Eingänge			
0x2008	Spulen	-	Diskretes Eingang X10
0x2009	Spulen	-	Diskreter Eingang X11
. . .	. . .	. . .	. . .
0x207F	Spulen	-	Diskreter Eingang X177

Adresse	Typ	Größe	Beschreibung
Allzweck-Datenregister D192.. .D255			
0x3000	Eingaberegister	16-Bit	Register D192
0x3001	Eingaberegister	16-Bit	Register D193
. . .	. . .	. . .	. . .
0x303F	Eingaberegister	16-Bit	Register D255
Allzweck-Datenregister D256.. .D319			
0x4000	Halteregister	16-Bit	Register D256
0x4001	Halteregister	16-Bit	Register D257
. . .	. . .	. . .	. . .
0x403F	Halteregister	16-Bit	Register D319
Nichtflüchtige Datenregister D320.. .D327			
0x3100	Eingaberegister	16-Bit	Register D320
. . .	. . .	. . .	. . .
0x3107	Eingaberegister	16-Bit	Register D327
Nichtflüchtige Datenregister D328.. .335			
0x4100	Halteregister	16-Bit	Register D328
. . .	. . .	. . .	. . .
0x4107	Halteregister	16-Bit	Register D335

Adresse	Typ	Größe	Beschreibung
Hardware-Zähler			
0x4200	Halteregister	32-Bit	Zähler C64
0x4202	Halteregister	32-Bit	Zähler C65
Analog-Digital-Wandler			
0x3200	Eingaberegister	16-Bit	Register D352, Daten vom Potentiometer «0»
0x3201	Eingaberegister	16-Bit	Register D353, Daten vom Potentiometer «1»
0x3202	Eingaberegister	16-Bit	Register D354, Daten vom Potentiometer «SPEED»
Hardware- und Softwareversionen			
0x8001	Eingaberegister	16-Bit	Haupt-Hardwareversion
0x8002	Eingaberegister	16-Bit	Nebenhardwareversion
0x8003	Eingaberegister	16-Bit	Hauptversion der Software
0x8004	Eingaberegister	16-Bit	Nebenversionsnummer der Software
0x8005	Eingaberegister	16-Bit	Hauptversion des Bootloaders
0x8006	Eingaberegister	16-Bit	Nebenversion des Bootloaders
Schrittmotorsteuerung			
0x5000	Halteregister	32-Bit	Register D357 – SPEED.
0x5002	Halteregister	32-Bit	Register D359 – MIN_SPEED.
0x5004	Halteregister	16-Bit	Register D361 – ACC.

Adresse	Typ	Größe	Beschreibung
0x5005	Halteregister	16-Bit	Register D362 – DEC.
0x5006	Halteregister	32-Bit	Register D363 – ABS.
0x5008	Eingaberegister	16-Bit	Register D365 – U_POS.
0x5009	Halteregister	16-Bit	Register D366 – U_STEP.
0x500A	Halteregister	16-Bit	Register D374 – DIR.
0x500A	Spulen	-	Register D374 – DIR.
0x500B	Halteregister	32-Bit	Register D377 – FS_SPD_THR.
0x500D	Halteregister	16-Bit	Register D379 – FS_SW_EN.
0x500D	Spulen	-	Register D379 – FS_SW_EN.
0x500E	Halteregister	32-Bit	Register D372 – ZIEL_POS.
0x5010	Halteregister	16-Bit	Register D376 – CMD.
0x5011	Halteregister	16-Bit	Register D375 – SW_INPUT.
0x5012	Halteregister	16-Bit	Register D367 – ACC_CUR.
0x5013	Halteregister	16-Bit	Register D368 – DEC_CUR.
0x5014	Halteregister	16-Bit	Register D369 – RUN_CUR.
0x5015	Halteregister	16-Bit	Register D370 – HOLD_CUR.
0x5016	Halteregister	16-Bit	Register D382 – CMIN_SPD_EN.
0x5017	Halteregister	16-Bit	Register D380 – ERROR_SET_HIZ.

Adresse	Typ	Größe	Beschreibung
0x5017	Spulen	-	TERMAL_ERROR_SET_HIZ
0x5018	Spulen	-	SOFTWARE_ERROR_SET_HIZ
0x5019	Spulen	-	CMD_ERROR_SET_HIZ
0x501A	Spulen	-	DATA_ERROR_SET_HIZ
0x5027	Halteregister	16-Bit	Register D381 – ERROR_CODE.
0x5027	Spulen	-	TERMAL_ERROR_OVER_CURRENT
0x5028	Spulen	-	SOFT_ERROR
0x5029	Spulen	-	CMD_ERROR
0x502A	Spulen	-	DATA_ERROR
0x502F	Spulen	-	OVLO/UVLO_INTERNAL_PROTECTION_ERROR (RS 434542 and RS 434543)
0x5030	Spulen	-	VS_OUT_OF_RANGE_ERROR
0x5037	Eingaberegister	16-Bit	Register D371 – MOTOR_STATUS.
0x5037	Diskrete Eingänge	-	HIZ
0x5038	Diskrete Eingänge	-	STOP
0x5039	Diskrete Eingänge	-	ACCELERATING
0x503A	Diskrete Eingänge	-	DECELERATING
0x503B	Diskrete Eingänge	-	STEADY
0x503C	Diskrete Eingänge	-	BUSY_MOVE
0x503D	Diskrete Eingänge	-	BUSY_RUN
0x5048	Bestände Register	32-Bit	Register D385 – EMERGENCY_DEC.
0x5100	Spulen	-	Instruction SPIN – APPLY_CMD.
0x5101	Spulen	-	Instruction TORQUE – APPLY_CURRENT.

Adresse	Typ	Größe	Beschreibung
0x5102	Spulen	-	Instruction HSTOP – HARD_STOP.
0x5103	Spulen	-	Instruction HHIZ – HARD_HIZ.
0x5104	Spulen	-	Instruction SSTOP – SLOW_STOP.
0x5105	Spulen	-	Instruction SHIZ – SLOW_HIZ.

Anhang B. Liste der Anweisungen

Befehl		Beschreibung
API	Code	
Grundlegende Anweisungen		
LD	0x4061	Schließer
LDI	0x4001	Schließer
AND	0x4065	Reihenschaltung - Öffner (logisches AND)
ANI	0x4005	Reihenschaltung - Öffner (logische NAND-Verknüpfung)
OR	0x4066	Parallelschaltung – Schließer (logisches OR)
ORI	0x4046	Parallelschaltung – Öffner (logische NOR-Verknüpfung)
LDP	0x4821	Beginn des logischen Ausdrucks mit Flankenabfrage (Impuls)
LDF	0x4841	Beginn eines logischen Ausdrucks mit Abfrage an einer fallenden Flanke (Impuls)
ANDP	0x4825	«AND» mit einer steigenden Flankenabfrage (Impuls)
ANDF	0x4845	«AND» mit einer fallenden Flankenabfrage (Impuls)
ORP	0x4806	«OR» mit Flankenabfrage (Impuls)
ORF	0x4826	«OR» mit Flankenabfrage (Impuls)
TMR	0x2014	Zeitgeber (16-Bit)
CNT	0x2015	Zähler (16-Bit)
DCNT	0x3015	Zähler (32-Bit)
INV	0x4016	Invertierung - Ersetzen des Ergebnisses logischer Verknüpfungen durch das Gegenteil
ANB	0x4007	«AND»-Block: Reihenschaltung von Blöcken
ORB	0x4008	«OR»-Block: Parallelschaltung von Blöcken
MPS	0x4009	Offset im Stack nach unten
MRD	0x402A	Wert vom Stapel lesen
MPP	0x400A	Verlassen des Stacks

SET	0x2024	Aktivieren des verriegelten Ausgangs (Setzen der logischen „1“)
RST	0x2004	Zurücksetzen des Operandenstatus
OUT	0x2002	Ausgangsspule - Zuweisung des Ergebnisses eines logischen Ausdrucks an den Ausgang
FEND	0x6003	Ende des Hauptprogramms
NOP	0x8011	Leere Zeile im Programm
P	0x6051	Adressierung eines Sprungpunkts in einem Programm oder Unterprogramm
I	0x6031	Adressierung eines Unterbrechungspunkts
END	0x6023	Programmende
Anweisungen für Schleifen, Übergänge, Unterprogramme		
CJ	0x200D	Bedingter Sprung - gehe zur angegebenen Programmzeile
CJP	0x280D	Bedingter Sprung - gehe zur angegebenen Programmzeile mit Flankenabfrage (Impuls)
CALL	0x200E	Unterprogrammaufruf
CALLP	0x280E	Aufruf eines Unterprogramms mit Flankenabfrage (Impuls)
SRET	0x600F	Ende des Unterprogramms
FOR	0x400B	Beginn einer FOR-NEXT-Schleife
NEXT	0x400C	Ende einer FOR-NEXT-Schleife
Unterbrechungen		
IRET	0x6010	Ende des Interrupt-Handlers
EI	0x8012	Aktivierung globaler Interrupts
DI	0x8013	Globale Interruptsperre
Datenübertragung und -vergleich		
CMP	0x2201	Vergleich von numerischen Daten
CMPP	0x2A01	Vergleich von numerischen Daten mit Flankenabfrage (Impuls)

DCMP	0x3201	Vergleich von numerischen Daten, 32-Bit-Befehl
DCMPP	0x3A01	Vergleich von numerischen Daten, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
ZCP	0x2202	Zonenvergleich von numerischen Daten
ZCPP	0x2A02	Zonenvergleich von numerischen Daten mit Flankenabfrage (Impuls)
DZCP	0x3202	Zonenvergleich von numerischen Daten, 32-Bit-Befehl
DZCPP	0x3A02	Zonenvergleich von numerischen Daten, 32-Bit-Befehl mit Flankenabfrage (Impuls)
MOV	0x2018	Datenübertragung
MOVP	0x2818	Datenübertragung mit Flankenabfrage (Impuls)
DMOV	0x3018	Datenübertragung, 32-Bit-Befehl
DMOVP	0x3818	Datenübertragung, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
BMOV	0x2038	Blockdatenübertragung
BMOVP	0x2838	Blockdatenübertragung mit Flankenabfrage (Impuls)
DBMOV	0x3038	Blockdatenübertragung, 32-Bit-Befehl
DBMOVP	0x3838	Blockdatenübertragung, 32-Bit-Befehl mit Flankenabfrage (Impuls)
FMOV	0x2058	Datenübertragung an mehrere Adressen
FMOVP	0x2858	Datenübertragung an mehrere Adressen mit Flankenabfrage (Impuls)
DFMOV	0x3058	Übertragung von Daten an mehrere Adressen, 32-Bit-Instruktion
DFMOVP	0x3858	Datenübertragung an mehrere Adressen, 32-Bit-Befehl mit Flankenabfrage (Impuls)
XCH	0x220A	Datenaustausch
XCHP	0x2A0A	Datenaustausch mit Flankenabfrage (Impuls)
DXCH	0x320A	Datenaustausch, 32-Bit-Befehl

DXCHP	0x3A0A	Datenaustausch, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
Arithmetische Operationen (ganzzahlig)		
ADD	0x2208	Addition numerischer Daten
ADDP	0x2A08	Addition numerischer Daten mit Flankenabfrage (Impuls)
DADD	0x3208	Addition numerischer Daten, 32-Bit-Instruktion
DADDP	0x3A08	Addition numerischer Daten, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
SUB	0x2228	Subtraktion numerischer Daten
SUBP	0x2A28	Subtraktion numerischer Daten mit Flankenabfrage (Impuls)
DSUB	0x3228	Subtraktion numerischer Daten, 32-Bit-Instruktion
DSUBP	0x3A28	Subtraktion numerischer Daten, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
MUL	0x2248	Multiplikation numerischer Daten
MULP	0x2A48	Multiplikation von numerischen Daten mit Flankenabfrage (Impuls)
DMUL	0x3248	Multiplikation numerischer Daten, 32-Bit-Instruktion
DMULP	0x3A48	Multiplikation numerischer Daten, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
DIV	0x2268	Division numerischer Daten
DIVP	0x2A68	Division numerischer Daten mit Flankenabfrage (Impuls)
DDIV	0x3268	Division numerischer Daten, 32-Bit-Instruktion
DDIVP	0x3A68	Division numerischer Daten, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
MOD	0x22E8	Rest der Division
MODP	0x2AE8	Rest der Division mit Flankenabfrage (Impuls)
DMOD	0x32E8	Rest der Division, 32-Bit-Instruktion
DMODP	0x3AE8	Rest der Division, 32-Bit-Instruktion mit Flankenabfrage (Impuls)

INC	0x2037	Numerische Daten inkrementieren (um 1 erhöhen)
INCP	0x2837	Numerische Daten inkrementieren (um 1 erhöhen) mit Flankenabfrage (Impuls)
DINC	0x3037	Numerische Daten inkrementieren (um 1 erhöhen), 32-Bit-Instruktion
DINCP	0x3837	Numerische Daten inkrementieren (um 1 erhöhen), 32-Bit-Befehl mit Flankenabfrage (Impuls)
DEC	0x2017	Numerische Daten dekrementieren (um 1 verringern)
DECP	0x2817	Numerische Daten dekrementieren (um 1 verringern) mit Flankenabfrage (Impuls)
DDEC	0x3017	Numerische Daten dekrementieren (um 1 verringern), 32-Bit-Instruktion
DDECP	0x3817	Numerische Daten dekrementieren (um 1 verringern), 32-Bit-Instruktion mit Flankenabfrage (Impuls)
WAND	0x2288	Logische Multiplikation numerischer Daten (Operation „AND“)
WANDP	0x2A88	Logische Multiplikation von numerischen Daten (Operation „AND“) mit Flankenabfrage (Impuls)
DAND	0x3288	Logische Multiplikation von numerischen Daten (Operation „AND“), 32-Bit-Instruktion
DANDP	0x3A88	Logische Multiplikation von numerischen Daten (Operation „AND“), 32-Bit-Instruktion mit Flankenabfrage (Impuls)
WOR	0x22A8	Logische Addition numerischer Daten (ODER-Verknüpfung)
WORP	0x2AA8	Logische Addition von numerischen Daten (OR-Verknüpfung) mit Flankenabfrage (Impuls)
DOR	0x32A8	Logische Addition von numerischen Daten (OR-Verknüpfung), 32-Bit-Instruktion
DORP	0x3AA8	Logische Addition von numerischen Daten (OR-Verknüpfung), 32-Bit-Instruktion mit Flankenabfrage (Impuls)
WXOR	0x22C8	Logische Operation „Exklusives OR“

WXORP	0x2AC8	Logische Operation „Exklusiv-OR“ mit Flankenabfrage (Impuls)
DXOR	0x32C8	Logische Operation „Exklusives OR“, 32-Bit-Instruktion
DXORP	0x3AC8	Logische Operation „Exklusiv-OR“, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
NEG	0x2209	Logische Negation
NEGP	0x2A09	Logische Negation mit Flankenabfrage (Impuls)
DNEG	0x3209	Logische Negation, 32-Bit-Instruktion
DNEGP	0x3A09	Logische Negation, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
ABS	0x2229	Absolutwert
ABSP	0x2A29	Absolutwert mit Flankenabfrage (Impuls)
DABS	0x3229	Absolutwert, 32-Bit-Befehl
DABSP	0x3A29	Absolutwert, 32-Bit-Befehl mit Flankenabfrage (Impuls)
SQR	0x2215	Berechnung der Quadratwurzel
SQRP	0x2A15	Berechnung der Quadratwurzel mit Flankenabfrage (Impuls)
DSQR	0x3215	Quadratwurzelberechnung, 32-Bit-Instruktion
DSQRP	0x3A15	Berechnung der Quadratwurzel, 32-Bit-Befehl mit Flankenabfrage (Impuls)
POW	0x2216	Potenzieren
POWP	0x2A16	Potenzieren mit Flankenabfrage (Impuls)
DPOW	0x3216	Potenzieren, 32-Bit-Befehl
DPOWP	0x3A16	Potenzieren, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
Schiebeoperationen		
ROR	0x220B	Zyklische Verschiebung nach rechts
RORP	0x2A0B	Zyklische Verschiebung nach rechts mit Flankenabfrage (Impuls)
DROR	0x320B	Zyklische Rechtsverschiebung, 32-Bit-Befehl

DRORP	0x3A0B	Zyklische Rechtsverschiebung, 32-Bit-Befehl mit Flankenabfrage (Impuls)
ROL	0x222B	Zyklische Verschiebung nach links
ROLP	0x2A2B	Zyklische Verschiebung nach links mit Flankenabfrage (Impuls)
DROL	0x322B	Zyklische Verschiebung nach links, 32-Bit-Befehl
DROLP	0x3A2B	Zyklische Verschiebung nach links, 32-Bit-Befehl mit Flankenabfrage (Impuls)
Datenverarbeitung		
ZRST	0x2203	Gruppenrücksetzung von Operanden in einem gegebenen Bereich
ZRSTP	0x2A03	Gruppenrücksetzung von Operanden in einem gegebenen Bereich mit Flankenabfrage (Impuls)
DECO	0x2211	Decoder 8 → 256-Bit
DECOP	0x2A11	Decoder 8 → 256-Bit mit Flankenabfrage (Impuls)
ENCO	0x2212	Kodierer 256 → 8 Bit
ENCOP	0x2A12	Encoder 256 → 8-Bit mit Abfrage der steigenden Flanke (Impuls)
SUM	0x2213	Summe der einzelnen Bits im Register
SUMP	0x2A13	Summe der einzelnen Bits im Register mit Flankenabfrage (Impuls)
DSUM	0x3213	Summe der einzelnen Bits im Register, 32-Bit-Instruktion
DSUMP	0x3A13	Summe der einzelnen Bits im Register, 32-Bit-Instruktion mit Flankenabfrage (Impuls)
BON	0x2214	Überprüfung eines Bitstatus mit Setzen eines Ausgangssignals
BONP	0x2A14	Überprüfung eines Bitstatus mit Setzen eines Ausgangs durch Flankenabfrage (Impuls)
DBON	0x3214	Überprüfung eines Bitstatus mit Setzen eines Ausgangs, 32-Bit-Befehl

DBONP	0x3A14	Überprüfung eines Bitstatus mit Setzen eines Ausgangs, 32-Bit-Befehl mit Flankenabfrage (Impuls)
FLT	0x220C	Konvertierung einer Ganzzahl in eine Gleitkommazahl
FLTP	0x2A0C	Konvertiere Integer in Gleitkommazahl mit Flankenabfrage (Impuls)
DFLT	0x320C	Konvertierung einer Ganzzahl in eine Gleitkommazahl, 32-Bit-Befehl
DFLTP	0x3A0C	Konvertiere Integer zu Gleitkommazahl, 32-Bit-Befehl mit Flankenabfrage (Impuls)
Gleitkommaoperationen		
DECMP	0x220F	Vergleich von Gleitkommazahlen
DECMPP	0x2A0F	Vergleich von Gleitkommazahlen mit Flankenabfrage (Impuls)
DEZCP	0x2210	Zonen-Gleitkommavergleich
DEZCPP	0x2A10	Zonen-Gleitkommavergleich mit Flankenabfrage (Impuls)
DEADD	0x2217	Addition von Gleitkommazahlen
DEADDP	0x2A17	Addition von Gleitkommazahlen mit Flankenabfrage (Impuls)
DESUB	0x2237	Subtraktion von Gleitkommazahlen
DESUBP	0x2A37	Subtraktion von Gleitkommazahlen mit Flankenabfrage (Impuls)
DEMUL	0x2257	Multiplikation von Gleitkommazahlen
DEMULP	0x2A57	Multiplikation von Gleitkommazahlen mit Flankenabfrage (Impuls)
DEDIV	0x2277	Gleitkommazahlendivision
DEDIVP	0x2A77	Gleitkommazahlendivision mit Flankenabfrage (Impuls)
DESQR	0x2218	Quadratwurzel im Gleitkommaformat
DESQRP	0x2A18	Quadratwurzel im Gleitkommaformat mit Flankenabfrage (Impuls)
DEPOW	0x2297	Potenzieren im Gleitkommaformat

DEPOWP	0x2A97	Potenzieren im Gleitkommaformat mit Flankenabfrage (Impuls)
INT	0x220D	Konvertierung einer Gleitkommazahl in eine Ganzzahl
INTP	0x2A0D	Konvertierung einer Gleitkommazahl in eine Ganzzahl mit Flankenabfrage (Impuls)
DINT	0x320D	Konvertierung einer Gleitkommazahl in eine Ganzzahl, 32-Bit-Anweisung
DINTP	0x3A0D	Konvertierung einer Gleitkommazahl in eine Ganzzahl, 32-Bit-Befehl mit Flankenabfrage (Impuls)
Zeit und PWM		
TRD	0x2219	Auslesen des aktuellen Werts der Echtzeituhr
TRDP	0x2A19	Auslesen des aktuellen Werts der Echtzeituhr mit Flankenabfrage (Impuls)
TWR	0x221A	Ändern des Werts einer Echtzeituhr
TWRP	0x2A1A	Ändern des Werts einer Echtzeituhr mit Flankenabfrage (Impuls)
PWM	0x220E	Pulsweitenmodulation (PWM) Ausgang
Datum		
DRD	0x2239	Das Lesen des aktuellen Datumswerts
DRDP	0x2A39	Auslesen des aktuellen Datumswerts mit Flankenabfrage (Impuls)
DWR	0x223A	Den Datumswert ändern
DWRP	0x2A3A	Ändern Sie den Datumswert mit Flankenabfrage (Impuls)
Kontakttypische logische Operationen		
LD&	0x4204	Kontakt ist geschlossen, wenn S1 & S2 ≠ 0
DLD&	0x5204	Kontakt ist geschlossen, wenn S1 & S2 ≠ 0, 32-Bit-Instruktion
LD|	0x4224	Kontakt ist geschlossen, wenn S1 | S2 ≠ 0
DLD|	0x5224	Kontakt ist geschlossen, wenn S1 | S2 ≠ 0, 32-Bit-Instruktion

LD^	0x4244	Kontakt ist geschlossen, wenn S1 ^ S2 ≠ 0
DLD	0x5244	Kontakt ist geschlossen, wenn S1 ^ S2 ≠ 0, 32-Bit-Instruktion
AND	0x4205	Serienkontakt geschlossen, wenn S1 & S2 ≠ 0
DAND&	0x5205	Serienkontakt geschlossen, wenn S1 & S2 ≠ 0, 32-Bit-Instruktion
UND|	0x4225	Serienkontakt geschlossen, wenn S1 | S2 ≠ 0
DAND|	0x5225	Serienkontakt geschlossen, wenn S1 | S2 ≠ 0, 32-Bit-Instruktion
AND^	0x4245	Serienkontakt geschlossen, wenn S1 ^ S2 ≠ 0
DAND^	0x5245	Serienkontakt geschlossen, wenn S1 ^ S2 ≠ 0, 32-Bit-Instruktion
OR&	0x4206	Parallelkontakt geschlossen, wenn S1 & S2 ≠ 0
DOR&	0x5206	Parallelkontakt geschlossen, wenn S1 & S2 ≠ 0, 32-Bit-Instruktion
OR|	0x4226	Parallelkontakt geschlossen, wenn S1 | S2 ≠ 0
DOR|	0x5226	Parallelkontakt geschlossen, wenn S1 | S2 ≠ 0, 32-Bit-Instruktion
OR^	0x4246	Parallelkontakt geschlossen, wenn S1 ^ S2 ≠ 0
DOR^	0x5246	Parallelkontakt geschlossen, wenn S1 ^ S2 ≠ 0, 32-Bit-Instruktion
Vergleichsoperationen für Kontakttypen		
LD=	0x4264	Kontakt ist geschlossen, wenn S1 = S2
DLD=	0x5264	Kontakt ist geschlossen, wenn S1 = S2 ist, 32-Bit-Instruktion
LD	0x4284	Kontakt ist geschlossen, wenn S1 > S2
DLD	0x5284	Kontakt ist geschlossen, wenn S1 > S2, 32-Bit-Instruktion
LD	0x42A4	Kontakt ist geschlossen, wenn S1 < S2
DLD	0x52A4	Kontakt ist geschlossen, wenn S1 < S2, 32-Bit-Instruktion
LD	0x42C4	Kontakt ist geschlossen, wenn S1 ≠ S2
DLD	0x52C4	Kontakt ist geschlossen, wenn S1 ≠ S2, 32-Bit-Instruktion

LD<=	0x42E4	Kontakt ist geschlossen, wenn S1 ≤ S2
DLD<=	0x52E4	Kontakt ist geschlossen, wenn S1 ≤ S2 ist, 32-Bit-Instruktion
LD=	0x4304	Kontakt ist geschlossen, wenn S1 ≥ S2
DLD=	0x5304	Kontakt ist geschlossen, wenn S1 ≥ S2, 32-Bit-Instruktion
AND=	0x4265	Serienkontakt geschlossen, wenn S1 = S2
DAND=	0x5265	Serienkontakt geschlossen, wenn S1 = S2, 32-Bit-Instruktion
UND	0x4285	Serienkontakt geschlossen, wenn S1 > S2
DAND	0x5285	Serienkontakt geschlossen, wenn S1 > S2, 32-Bit-Instruktion
AND	0x42A5	Serienkontakt geschlossen, wenn S1 < S2
DAND<	0x52A5	Serienkontakt geschlossen, wenn S1 < S2, 32-Bit-Instruktion
AND<>	0x42C5	Serienkontakt geschlossen, wenn S1 ≠ S2
DAND<>	0x52C5	Serienkontakt geschlossen, wenn S1 ≠ S2, 32-Bit-Instruktion
AND<=	0x42E5	Serieller Kontakt geschlossen, wenn S1 ≤ S2
DAND<=	0x52E5	Serienkontakt geschlossen, wenn S1 ≤ S2, 32-Bit-Instruktion
AND>=	0x4305	Serienkontakt geschlossen, wenn S1 ≥ S2
DAND>=	0x5305	Serienkontakt geschlossen, wenn S1 ≥ S2, 32-Bit-Instruktion
OR=	0x4266	Parallelkontakt geschlossen, wenn S1 = S2
DOR=	0x5266	Parallelkontakt geschlossen, wenn S1 = S2, 32-Bit-Instruktion
OR>	0x4286	Parallelkontakt geschlossen, wenn S1 > S2
DOR>	0x5286	Parallelkontakt geschlossen, wenn S1 > S2, 32-Bit-Instruktion
OR<	0x42A6	Parallelkontakt geschlossen, wenn S1 < S2
DOR<	0x52A6	Parallelkontakt geschlossen, wenn Parallelkontakt geschlossen, wenn S1 < S2, 32-Bit-Instruktion
OR<>	0x42C6	Parallelkontakt geschlossen, wenn S1 ≠ S2
DOR<>	0x52C6	Parallelkontakt geschlossen, wenn S1 ≠ S2, 32-Bit-Instruktion
OR<=	0x42E6	Parallelkontakt geschlossen, wenn S1 ≤ S2
DOR<=	0x52E6	Parallelkontakt geschlossen, wenn S1 ≤ S2, 32-Bit-Instruktion
OR>=	0x4306	Parallelkontakt geschlossen, wenn S1 ≥ S2

DOR>=	0x5306	Parallelkontakt geschlossen, wenn S1 ≥ S2, 32-Bit-Instruktion
Schrittmotorsteuerung		
SPIN	0x2207	Bewegungsvorwahl starten
SPINP	0x2A07	Starten Sie die voreingestellte Bewegung mit einer Abfrage der steigenden Flanke (Impuls)
TORQUE	0x2227	Die eingestellten Ströme an den Motor anlegen
TORQUEP	0x2A27	Die eingestellten Ströme mit Flankenabfrage (Impuls) auf den Motor geben.
HSTOP	0x2247	Sofort in den Haltemodus wechseln
HSTOPP	0x2A47	Sofortiger Wechsel in den Haltemodus mit Flankenauswertung (Impuls)
HHIZ	0x2267	Motorphasen sofort deaktivieren (die Welle dreht sich frei)
HHIZP	0x2A67	Motorphasen sofort (Welle dreht sich frei) mit Flankenabfrage (Impuls) stromlos schalten
SSTOP	0x2287	Bis zum vollständigen Stillstand abbremsen und in den Haltemodus wechseln
SSTOPP	0x2A87	Bis zum vollständigen Stillstand abbremsen und mit Flankenauswertung (Impuls) in den Haltemodus wechseln
SHIZ	0x22A7	Bis zum vollständigen Stillstand abbremsen und Motorphasen stromlos schalten (die Welle dreht sich frei)
SHIZP	0x2AA7	Abbremsen bis zum vollständigen Stillstand und Spannungsfreischalten der Motorphasen (die Welle dreht sich frei) mit Flankenabfrage (Impuls)

Anhang C. Beispiele von Benutzerprogrammen

Beispiel 1. Verwendung des RUN-Befehls

LDP	X0			***;erfasse den vorderen Teil des Impulses am Eingang X0 (Taste)***
DMOV	K8	D359		*;minimale Geschwindigkeit auf 8 pps setzen*
DMOV	K120000	D357		*;Maximale Geschwindigkeit auf 120000 pps setzen*
FMOV	K30000	D361	K2	*;Beschleunigung und Verzögerung auf 30000 pps^2 setzen*
MOV	K3	D366		*;Mikroschrittbetrieb 1/8 (siehe Beschreibung des Befehls SPIN)*
MOV	K1	D374		*;Richtung – vorwärts*
DMOV	K6000	D377		*;volle Schrittgeschwindigkeit 6000 pps/Sek. einstellen*
MOV	K1	D379		*;Aktivierung, um in den Vollschrittmodus zu wechseln, wenn die Vollschrittgeschwindigkeit erreicht ist*
MOV	K0	D376		*;Befehl AUSFÜHREN*
FMOV	K1500	D367	K2	*;Beschleunigungs- und Verzögerungsströme 1500 mA*
MOV	K1200	D369		*;konstante Geschwindigkeit Strom 1200 mA*
MOV	K600	D370		*;hält aktuell 600 mA*
TORQUE				*;aktuelle Werte anwenden*
FMOV	K0	D380	K3	*;keine Fehlermeldung, Fehler zurückgesetzt, verwenden ;MINDESTGESCHWINDIGKEIT*
SPIN				*;Bewegung starten*
LDP	X1			*;erfasse den vorderen Teil des Impulses am Eingang X1 (Taste)*
SSTOP				*;gemäß dem voreingestellten DEC anhalten und in den Haltemodus wechseln*
LDP	X2			*;erfasse den vorderen Teil des Impulses am Eingang X2 (Taste)*
SHIZ				*;gemäß dem voreingestellten DEC anhalten und in den HiZ-Modus wechseln*
LDP	X3			*;den vorderen Teil des Impulses am Eingang X3 (Taste) erfassen*
HSTOP				*; sofort in den Haltemodus wechseln*

LDP	X4			*;erfasse den vorderen Teil des Impulses am Eingang X4 (Taste)*
HHIZ				*;sofort in den HiZ-Modus wechseln*
END				*;Ende des Programms*

Beispiel 2. Verwendung der Befehle MOVE, GOTO, GOHOME

LD	M0			***;um den Initialisierungsabschnitt zu überspringen, überprüfen Sie die Bedingung M0***
CJ	P1			*;und springe zur mit P1 markierten Zeile*
LDP	M108			*;M108 Vorderflanke nur nach der Initialisierung*
DMOV	K120000	D357		*;maximale Geschwindigkeit auf 120000 pps setzen*
FMOV	K30000	D361	K2	*;Beschleunigung und Verzögerung auf 30000pps²setzen*
MOV	K3	D366		*;Mikroschrittbetrieb 1/8 (siehe Beschreibung des Befehls SPIN)*
DMOV	K6000	D377		*;volle Schrittgeschwindigkeit 6000 pps/Sek. einstellen*
MOV	K1	D379		*;Aktivierung, um in den Vollschrittmodus zu wechseln, wenn die Vollschrittgeschwindigkeit erreicht ist*
FMOV	K1500	D367	K2	*;Beschleunigungs- und Verzögerungsströme 1500 mA*
MOV	K1200	D369		*;konstante Geschwindigkeit Strom 1200 mA*
MOV	K600	D370		*;hält aktuell 600 mA*
TORQUE				*;aktuelle Werte anwenden*
FMOV	K0	D380	K2	*;keine Fehlermeldung, Fehler werden zurückgesetzt*
MOV	K1	D382		*;automatische Berechnung der Anfangs- und Endgeschwindigkeit verwenden*
DMOV	K0	D363		*;aktuelle Position auf null setzen*
SET	M0			*;Initialisierung des Treibers umgehen aktivieren*
P	1			*;Übergangsmarkierung*
LDP	X0			*;erfasse den vorderen Teil des Impulses am Eingang X0 (Taste)*
AND&	D371	K3		*;nur wenn der Motor im HiZ- oder Hold-Modus ist*
DMOV	K10000	D372		*;bewege10000 Mikroschritte*
MOV	K1	D374		*;in Vorwärtsrichtung*

MOV	K1	D376	,wird*durch den Befehl MOVE ausgeführt*
SPIN			*;Bewegung starten*
LDP	X1		*;erfasse den vorderen Teil des Impulses am Eingang X1 (Taste)*
AND&	D371	K3	*;nur wenn der Motor im HiZ- oder Hold-Modus ist*
DMOV	K100000	D372	*;Wechsel zu einer Position mit der Koordinate 100000*
MOV	K2	D376	,wird*durch den Befehl GOTO ausgeführt*
SPIN			*;Bewegung starten*
LDP	X2		*;erfasse den vorderen Teil des Impulses am Eingang X2 (Taste)*
AND&	D371	K3	*;nur wenn der Motor im HiZ- oder Hold-Modus ist*
MOV	K0	D374	*;Bewegung in die "0"-Position in Rückwärtsrichtung*
MOV	K4	D376	,wird*durch den Befehl GOHOME ausgeführt*
SPIN			*;Bewegung starten*
LDP	X3		*;den vorderen Teil des Impulses am Eingang X3 (Taste) erfassen*
SSTOP			*;gemäß dem voreingestellten DEC anhalten und in den Haltemodus wechseln*
LDP	X4		*;erfasse den vorderen Teil des Impulses am Eingang X4 (Taste)*
SHIZ			*;gemäß dem voreingestellten DEC anhalten und in den HiZ-Modus wechseln*
LDP	X5		*;erfasse den vorderen Teil des Impulses am Eingang X5 (Taste)*
HSTOP			*; sofort in den Haltemodus wechseln*
LDP	X6		*;erfasse den vorderen Teil des Impulses am Eingang X6 (Taste)*
HHIZ			*;sofort in den HiZ-Modus wechseln*
END			*;Ende des Programms*

Beispiel 3. Verwendung der Befehle GOUNTIL_SLOWSTOP und RELEASE

Verwendung der Befehle GOUNTIL_SLOWSTOP und RELEASE als Beispiel für die Bewegung zur Ausgangsposition entlang des positiven Endschalters (siehe Abb. 36).

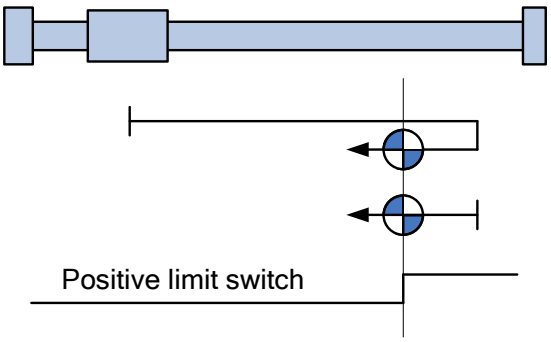


Abb. 36. Bewegung zum Ursprung

LD	M0			***;um den Initialisierungsabschnitt zu überspringen, überprüfen Sie die Bedingung M0***
CJ	P1			*;und springe zur mit P1 markierten Zeile*
LDP	M108			*;M108 Vorderflanke nur nach der Initialisierung*
FMOV	K30000	D361	K2	*;Beschleunigung und Verzögerung auf 30000pps^2 setzen*
MOV	K3	D366		*;Mikroschrittbetrieb 1/8 (siehe Beschreibung des Befehls SPIN)*
DMOV	K6000	D377		*;volle Schrittgeschwindigkeit 6000 pps/Sek. einstellen*
MOV	K1	D379		*;Aktivierung, um in den Vollschrittmodus zu wechseln, wenn die Vollschrittgeschwindigkeit erreicht ist*
FMOV	K1500	D367	K2	*;Beschleunigungs- und Verzögerungsströme 1500 mA*
MOV	K1200	D369		*;konstante Geschwindigkeit Strom 1200 mA*
MOV	K600	D370		*;hält aktuell 600 mA*
TORQUE				*;aktuelle Werte anwenden*
FMOV	K0	D380	K2	*;keine Fehlermeldung, Fehler werden zurückgesetzt*
MOV	K1	D382		*;automatische Berechnung der Anfangs- und Endgeschwindigkeit verwenden*
MOV	K7	D375		*;der Schaltergrenzwert ist mit dem Eingang IN7 verbunden*
SET	M0			*;Initialisierung des Treibers umgehen aktivieren*
P	1			*;Übergangsmarkierung*

LDP	X0		*;erfasse den vorderen Teil des Impulses am Eingang X0 (Taste)*
AND&	D371	K3	*;nur wenn der Motor im HiZ- oder Hold-Modus ist*
DMOV	K20000	D357	*;Maximale Geschwindigkeit auf 20000 pps setzen*
MOV	K5	D376	*;Befehl GOUNTIL_SLOWSTOP*
MOV	K1	D374	*;Richtung – vorwärts*
SPIN			*;Bewegung starten*
SET	M1		*;Setze das Flag, um die erste Stufe zu starten*
LD	M1		*;warte, wenn der Endschalter in der ersten Stufe aktiviert wird*
AND&	D371	K2	*;und der Motor stoppt*
RST	M1		*;Setze das Flag der ersten Stufe zurück*
DMOV	K1000	D357	*;Geschwindigkeit verringern*
MOV	K9	D376	*;Bewegung in die entgegengesetzte Richtung bis zum Endschalter*
MOV	K0	D374	*;öffnet*
SPIN			*;Bewegung starten*
SET	M2		*;und weiter zum zweiten Schritt*
LD	M2		*;Warten auf das Öffnen des Endschalters und das Anhalten des Motors*
AND&	D371	K2	*;in der zweiten Phase*
RST	M2		*;Setze das Flag der zweiten Stufe zurück*
DMOV	K0	D363	*;und setzt die aktuelle Position zurück, sie wird nun zum Ursprung*
LDP	X1		*;erfasse den vorderen Teil des Impulses am Eingang X1 (Taste)*
SSTOP			*;gemäß dem voreingestellten DEC anhalten und in den Haltemodus wechseln*
LDP	X2		*;erfasse den vorderen Teil des Impulses am Eingang X2 (Taste)*
SHIZ			*;gemäß dem voreingestellten DEC anhalten und in den HiZ-Modus wechseln*
LDP	X3		*;den vorderen Teil des Impulses am Eingang X3 (Taste) erfassen*
HSTOP			*; sofort in den Haltemodus wechseln*
LDP	X4		*;erfasse den vorderen Teil des Impulses am Eingang X4 (Taste)*
HHIZ			*;sofort in den HiZ-Modus wechseln*

FEND		*;Ende des Hauptprogramms*
I	1007	*;Interrupt-Handler für den Eingang IN7 (erforderlich für GOUNTIL _... und ... RELEASE Befehle, kann leer gelassen werden)*
IRET		*;zurück zum Hauptprogramm*
END		*;Ende des Programms*

Anhang D. Code des Dienstprogramms „Schrittmotordrehzahlregelung“

LD	M0			***;Initialisierungs-Bypass-Bedingung***
CJ	P1			
LDP	M108			*;Initialisierungsteil*
MPS				
LD=	D320	K6		*;D320 speichert Mikroschrittwert*
OR>	D320	K8		
OR<	D320	K0		
ANB				
MOV	K0	D320		
MRD				
MOV	D320	D366		
MOV	D320	D0		*;D0 – das Serviceregister für die Visualisierung von Mikroschritten*
MOV	K0	D2		
MRD				
AND>	D0	K6		*;da der Controller kein Mikroschrittverfahren 1/64 unterstützt,*
DEC	D0			*;überspringe diesen Wert*
MRD				
DECO	D0	Y0	K3	*;Visualisierung des Mikroschritts auf der Ausgangsskala*
MOV	K0	D379		*;Einrichtung des Schrittmotortreibers*
MOV	K0	D376		
MOV	K250	D359		
MOV	K0	D380		
MRD				
LD<	D321	K0		*;D321 speichert Steuerungsmethodendaten:*
OR>	D321	K2		*;Potentiometer/Tasten/Drehgeber*
ANB				
MOV	K0	D321		
MRD				
SET	M0			
MOV	D321	Y10		*;Visualisierung der Steuerungsmethode*
MRD				
AND<>	A0	D321		
MPS				
AND=	D321	K2		*;wenn ein**Encoder**ausgewählt ist, dann die Peripheriegeräte*

MOV	K12	D355		*;des Controllers müssen entsprechend eingestellt werden*
MOV	D321	A0		
RST	M108			*;und das Programm neu starten*
MPP				
AND<>	D321	K2		
MOV	K0	D355		
MOV	D321	A0		
RST	M108			
MRD				
MUL	D353	K10	D4	*;Daten des Potentiometers1*
DIV	D4	K27	D4	
FMOV	D4	D367	K3	*;Beschleunigungs-, Verzögerungs- und Konstantgeschwindigkeitsstrom einstellen*
DIV	D4	K2	D370	*;Halten des aktuellen Werts – 50% des Arbeitsstroms*
TORQUE				
MRD				
AND	X7			
MOV	K1	D374		
MRD				
ANI	X7			
MOV	K0	D374		
MRD				
DLD<	D322	K250		*;D322, D323 Geschwindigkeitseinstellung über Tasten*
DOR>	D322	K120000		
ANB				
DMOV	K250	D322		
MRD				
DMOV	D322	D5		
MRD				
DLD<	D324	K250		*;D324, D325 Geschwindigkeitseinstellung durch den Encoder*
DOR>	D324	K120000		
ANB				
DMOV	K250	D324		
MRD				
DMOV	D324	D15		
MRD				

DMOV	K0	C64	
DMOV	C64	D7	
MPP			
ANI	X4		
HSTOP			
LD	X2		
OUT	M102		
EI			;Aktivieren*Sie Interrupts, das Ende des Initialisierungsblocks*
P	1		
LD	X4		
AND&	D371	HFE	
HHIZ			
LDI	X4		
MPS			
AND	M102		
AND	X3		
AND&	D371	K3	
CALL	P10A0		
SPIN			
MPP			
LDI	X3		
ANB			
AND&	D371	HFD	
HSTOP			
LDI	M102		
AND	X3		
ANI	X4		
AND&	D371	H15	
SSTOP			
LD	M109		*;falls ein Fehler aufgetreten ist und die ERR-Anzeige leuchtete -*
TMR	T0	K10	*;Timer zum Ausschalten starten*
AND	T0		
RST	M109		
LD<>	A0	K1	
CJ	P19		
LD=	A0	K1	
MPS			
ANDP	X0		

DADD	D5	D354	D5
MPS			
DAND>	D5	K120000	
DMOV	K120000	D5	
MPP			
DMOV	D5	D322	
MPP			
ANDP	X1		
DSUB	D5	D354	D5
MPS			
DAND<	D5	K250	
DMOV	K250	D5	
MPP			
DMOV	D5	D322	
P	19		
LD<>	A0	K2	
CJ	P20		
LD=	A0	K2	
MPS			
DSUB	C64	D7	D9
DADD	D7	D9	D7
DCMP	D9	K0	M1
MRD			
AND	M1		
MOV	D354	D1	
DMUL	D9	D1	D11
DADD	D11	D15	D15
MPP			
AND	M3		
MOV	D354	D1	
ABS	D9		
DMUL	D9	D1	D11
DADD	D15	D11	D15
LD	M1		
OR	M3		
MPS			
DAND<	D15	K250	
DMOV	K250	D15	
MRD			
DAND>	D15	K120000	

```
DMOV    K120000   D15
MPP
DMOV    D15       D324
P       20
FEND                                  ;Ende des Hauptprogramms
P       10                            ;Unterprogramm zum Einstellen der
                                      Geschwindigkeit über das Potentiometer
LD      M108
MOV     K0        D2
MOV     D354      D1
MPS
AND=    D1        K0
MOV     K1        D1
MRD
DMUL    D1        K29       D13
MRD
DAND<   D13       K250
DMOV    K250      D13
MPP
DMOV    D13       D357
CALL    P0
SRET
P       11                            ; Unterprogramm zum Einstellen der
                                      Geschwindigkeit über Tasten
LD      M108
DMOV    D322      D357
CALL    P0
SRET
P       12                            ; Unterprogramm zum Einstellen der
                                      Geschwindigkeit durch den Encoder
LD      M108
DMOV    D324      D357
CALL    P0
SRET
P       0                             ;Unterprogramm zur Aktualisierung von
                                      Beschleunigung und Verzögerung
LD      M108
MOV     D352      D3
MPS
AND=    D3        K0
```

MOV	K1	D3		
MPP				
MUL	D3	K14	D361	
MOV	D361	D362		
SRET				
I	50			*;500 ms Unterbrechung zur Aktualisierung der Ströme*
LD	M108			
MUL	D353	K10	D4	
DIV	D4	K27	D4	
FMOV	D4	D367	K3	
DIV	D4	K2	D370	
TORQUE				
IRET				
I	10			*;Unterbrechung mit einer Periode von 100 ms zur Aktualisierung der Geschwindigkeit*
LDI	X6			
AND	X2			
AND	M102			
AND	X3			
ANI	X4			
BON	D371	M60	K6	
ANI	M60			
CALL	P10A0			
SPIN				
IRET				
I	1005			*;Unterbrechung von der Eingabe IN5*
LD	M105			
INC	D320			
MPS				
AND=	D320	K6		
INC	D320			
MRD				
AND=	D320	K9		
MOV	K0	D320		
INC	D321			
MPS				
AND=	D321	K3		
MOV	K0	D321		
MRD				

```
MOV     D321    Y10
MOV     D321    A0
MRD
AND=    D321    K2
MOV     K12     D355
RST     M108
MPP
AND<>   D321    K2
MOV     K0      D355
RST     M108
MRD
MOV     D320    D0
MOV     D320    D366
MRD
AND>    D0      K6
DEC     D0
MPP
DECO    D0      Y0      K3
IRET
I       1004                    ;Unterbrechung von der Eingabe IN4
LD      M104
HHIZ
LDI     M104
MPS
AND     X2
AND     X3
AND&    D371    K3
CALL    P10A0
SPIN
MPP
LDI     X2
ORI     X3
ANB
HSTOP
IRET
I       1003                    ;Unterbrechung von der Eingabe IN3
LDI     M103
ANI     X4
HSTOP
LD      M103
```

```
AND     X2
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       1002                    ;Unterbrechung von der Eingabe IN2
LDI     M102
AND     X3
ANI     X4
SSTOP
LD      M102
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       1007                    ;Unterbrechung von der Eingabe IN7
LD&     D371    H1C
MPS
AND     M107
AND=    D374    K0
SSTOP
MPP
ANI     M107
AND=    D374    K1
SSTOP
LD      M107
MOV     K1      D374
AND     X2
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
LDI     M107
MOV     K0      D374
AND     X2
AND     X3
```

ANI	X4		
AND&	D371	K3	
CALL	P10A0		
SPIN			
IRET			
I	2000		*;Unterbrechung bei Auftreten eines Treiberfehlers*
LD&	D381	K1	
MOV	K0	D381	
SET	M109		
MOV	K0	T0	
IRET			
END			

Anhang E. Die Lebensdauer der Flanken der Operanden M und Y

Alle folgenden Informationen gelten für die Operanden M und Y, sowohl für die Vorder- als auch für die Rückflanke.

Beispiel 1

Betrachten Sie die Lebensdauer der Vorderflanke des Operanden M0, mit garantierter Passage des Startpunkts der Lebensdauer beim nächsten Scan.

Zeile	Befehl	Operanden, Auftragsnummer		M0-Front-Lebensdauer, Scan		Erläuterung
		1	2	1	2	
1	LDP	M108				Die Vorderkante des M108 existiert nur beim ersten Scan des Programms
2	ZRST	D0	D2			Zurücksetzen von D0...D2 auf Null
3	LD	M108				
4	OUT	M0				Der Beginn der Lebensdauer der Vorderflanke des

						Operanden M0 liegt im ersten Scan, das Ende im zweiten.
5	P	1				
6	LD	M0				
7	CALL	P0				Der aktuelle Zustand M0 wird für jeden Sprung zum Unterprogramm P0 gespeichert.
8	LDP	M0				Es funktioniert nur einmal, da die Bedingung für einen erneuten Durchlauf des Anfangspunkts der Vorderseite des Operanden M0 im nächsten Programmscan erfüllt ist.
9	INC	D0				Das Ergebnis des Programms ist D0 = 1.
10	FEND					
11	I	0				Die erste Unterbrechung tritt nach Zeile 2 beim ersten Scan auf. Zu diesem Zeitpunkt ist die Vorderflanke von M0 nicht vorhanden. Die zweite Unterbrechung wird nach Zeile 6 verarbeitet. Das Vorhandensein einer Vorderflanke an M0 wird an die Unterbrechungsverarbeitung übertragen, sodass das Ergebnis der Arbeit D2 = 1 ist.
13	LDP	M0				
14	INC	D2				
15	IRET					
16	P	0				
17	LDP	M0				Die Bedingung ist beim ersten Scan erfüllt. Die Bedingung ist beim zweiten Scan nicht erfüllt.

18	INC	D1				Das Ergebnis ist D1 = 1.
19	SRET					
20	END					

- Unterbrechung

- Existenz einer Vorderseite des Operanden

Beispiel 2

Betrachten Sie die Lebensdauer der Vorderflanke des Operanden M0, ohne den Startpunkt der Lebensdauer beim nächsten Scan zu passieren.

Zeile	Befehl	Operanden, Auftragsnummer		M0-Front-Lebensdauer, Scan			Erläuterung
		1	2	1	2	3	
1	LDP	M0					
2	CJ	P1					Umgehung des Lebensbeginns der Vorderseite.
3	LDP	M108					Die Vorderflanke von M108 existiert nur beim ersten Scan des Programms.
4	ZRST	D0	D2				D0...D2 auf null setzen.
5	LD	M108					
6	OUT	M0					Der Beginn der Lebensdauer der Vorderflanke des Operanden M0 im ersten Scan. Der nächste Durchlauf dieses Punktes erfolgt erst im dritten Scan.
7	P	1					
8	LDP	M0					Es wird zweimal funktionieren, da der Startpunkt der Lebensdauer

							der Vorderseite vom Befehlshandler im zweiten Scan übersprungen wurde und die Lebensdauer bis zum Empfang des END / FEND-Befehls erhöht wurde.
9	INC	D0					Das Ergebnis des Programms ist D0 = 2.
10	END						Die Lebensdauer der Vorderflanke des Operanden M0 wird aufgrund des Fehlens der Passage des Startpunkts der Lebensdauer der Vorderflanke M0 bis zum Ende des Scans verlängert.

↓ - Existenz einer Vorderseite des Operanden

Wenn ein einzelner Durchlauf zwischen den Punkten 7 ... 9 erforderlich ist, kann beispielsweise der optionale Relaiskontakt M1 verwendet werden. Das Programm wird wie folgt geändert:

```
1   LDP     M0
2   CJ      P1
3   LDP     M108
4   ZRST    D0      D2
5   ZRST    M1              ;Initialisierung M1
6   LD      M108
7   OUT     M0
8   P       1
9   LDP     M0
10  ANI     M1              ;Zusätzliche Bedingung
11  INC     D0              ;Das Ergebnis des Programms ist D0 = 1
12  SET     M1              ;Sperrung
13  END
```

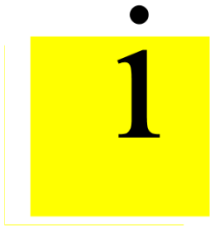
Anhang F. Debuggen des Benutzerprogramms

Der Debug-Modus ermöglicht dem Benutzer:

- vier Haltepunkte für die Ausführung des Benutzerprogramms zu setzen (Breakpoint),
- Operanden anzuzeigen und zu bearbeiten,
- die Ausführung des Benutzerprogramms anzuhalten und fortzusetzen.

Unten finden Sie die Liste der Debugger-Register:

Adresse	Typ	Größe	Beschreibung
Steuerung der Ausführung des Benutzerprogramms			
0x6100	Eingaberegister	16-Bit	Aktueller Index (Befehlszeilennummer) des Benutzerprogramms
0x6100	Spulen	-	Das Setzen des Objekts aktiviert den Debugging-Modus, das Zurücksetzen deaktiviert ihn. Der Debugging-Modus wird auch deaktiviert, wenn der RUN/STOP-Kippschalter in den STOP-Zustand geschaltet wird.
0x6100	Diskrete Eingänge	-	Anzeige des Debugging-Modus. Setzen – der Controller befindet sich im Debugging-Modus Zurücksetzen - der Controller befindet sich nicht im Debugging-Modus
0x6101	Spulen	-	Das Setzen des Objekts hält die Ausführung des Benutzerprogramms an. Das Setzen des Registers hält die Ausführung des Benutzerprogramms am aktuellen Index an. Zurücksetzen - setzt die Ausführung fort.
0x6101	Diskrete Eingänge	-	Anzeige des Anhaltens eines Benutzerprogramms. Angehalten – das Benutzerprogramm ist angehalten Zurücksetzen - das Benutzerprogramm läuft

Adresse	Typ	Größe	Beschreibung
0x6102	Spulen	-	Das Setzen des Objekts aktiviert das Einzelschritt-Debugging. Beim Versuch, das Benutzerprogramm durch Zurücksetzen der 6101h-Spulen wiederaufzunehmen, wird die Ausführung beim nächsten Index automatisch abgebrochen.
Haltepunkte			
			*Zusätzlich zum Einzelschritt-Debugging, wenn die Ausführung eines Benutzerprogramms bei jeder nächsten Befehlszeile anhält, ist es möglich, vier Haltepunkte festzulegen, an denen die Programmausführung angehalten wird.*
Haltepunkt 1			
0x6200	Spulen	-	Das Setzen des Objekts aktiviert den Haltepunkt 1. Die Ausführung des Benutzerprogramms wird an dem Index angehalten, der in den Holding Registern 6200h angegeben ist.
0x6200	Halteregister	16-Bit	Index (Anzahl der Befehlszeile) des Haltepunkts 1.
Haltepunkt 2			
0x6201	Spulen	-	Das Setzen des Objekts aktiviert den Haltepunkt 2. Die Ausführung des Benutzerprogramms wird an dem Index angehalten, der in den Holding Registern 6201h angegeben ist.
0x6201	Halteregister	16-Bit	Index (Anzahl der Befehlszeile) des Haltepunkts 2.

Adresse	Typ	Größe	Beschreibung
Haltepunkt 3			
0x6202	Spulen	-	Das Setzen des Objekts aktiviert den Haltepunkt 3. Die Ausführung des Benutzerprogramms wird an dem Index angehalten, der in den Holding Registern 6202h angegeben ist.
0x6202	Halteregister	16-Bit	Index (Anzahl der Befehlszeile) des Haltepunkts 3.
Haltepunkt 4			
0x6203	Spulen	-	Das Setzen des Objekts aktiviert den Haltepunkt 4. Die Ausführung des Benutzerprogramms wird an dem Index angehalten, der in den Holding Registern 6203h angegeben ist.
0x6203	Bestände Register	16-Bit	Index (Anzahl der Befehlszeile) des Haltepunkts 4.
Überwachung und Bearbeitung von Operanden			
0x6000	Spulen	-	Anforderung zum Lesen von Daten durch Setzen des Registers. Der Reset erfolgt automatisch. Die Antwort auf die Anfrage zeigt die Bereitschaft der angeforderten Daten über den Operanden an.
0x6001	Spulen	-	Anforderung zum Schreiben von Daten durch Setzen des Registers. Der Reset erfolgt automatisch. Die Antwort auf die Anfrage zeigt an, dass Daten in den Operanden geschrieben wurden.
0x6002	Spulen	-	Größe der Registerdaten beim Lesen/Schreiben Zurücksetzen – 16-Bit Setzen – 32-Bit.

Adresse	Typ	Größe	Beschreibung
0x6003	Spulen	-	Das Setzen des Registers bricht die Bearbeitung des Operandenwerts ab, wenn das Coil 6001h gesetzt ist – für die Operanden „C“ und „T“.
0x6004	Spulen	-	Das Setzen des Registers bricht die Bearbeitung des Operandensignals ab, wenn das Coil 6001h gesetzt ist – für die Operanden „C“ und „T“.
0x6000	Diskrete Eingänge	-	Das Objekt wird gesetzt, wenn ein Lese- oder Schreibvorgang eines Operanden fehlschlägt. Die Rücksetzung erfolgt automatisch, wenn ein Lese- oder Schreibvorgang angefordert wird.
Operandenparametrisierung			
0x6000	Halteregister	16-Bit	Operandentyp. X (0x58), Y (0x59), M (0x4D), T (0x54), C (0x43), A (0x41), B (0x42), D (0x44).
0x6001	Halteregister	16-Bit	Operandenindex.
Operandenüberwachung			
0x6002	Eingaberegister	32-Bit	Dieses Register enthält den Wert des Operanden (falls verfügbar) parametrisiert zum Lesen. Die Größe wird durch Coils 6002h eingestellt.
0x6004	Eingaberegister	16-Bit	Dieses Register enthält das Signal des Operanden (falls verfügbar), parametriert zum Lesen. Mögliche Werte: 0x00 – niedriges Niveau, 0x03 – hoher Pegel, mit einer Vorderflanke 0x02 – hohe Ebene, 0x04 – niedriger Pegel, mit einer fallenden Flanke.

Adresse	Typ	Größe	Beschreibung
Operandenbearbeitung			
0x6002	Halteregister	32-Bit	Dieses Register enthält den Wert des Operanden (falls verfügbar), der zum Schreiben parametriert ist. Die Größe wird durch Coils 6002h eingestellt.
0x6004	Halteregister	16-Bit	Dieses Register enthält das Signal des Operanden (falls verfügbar), das zum Schreiben parametriert ist. Mögliche Werte: 0x00 – niedriges Niveau, 0x03 – hoher Pegel, mit einer Vorderflanke 0x02 – hohe Ebene, 0x04 – niedriger Pegel, mit einer fallenden Flanke.

Lieferung in kompletten Sets

Schrittmotortreiber RS 434540 / RS 434542 / RS 434543 1 Stück

Herstellerinformationen

RS Components verfolgt die Linie der kontinuierlichen Entwicklung und behält sich das Recht vor, Änderungen und Verbesserungen am Design und der Software des Produkts ohne vorherige Ankündigung vorzunehmen.

Die in diesem Handbuch enthaltenen Informationen können jederzeit und ohne vorherige Ankündigung geändert werden.

Garantie

Jegliche Reparaturen oder Modifikationen werden vom Hersteller oder einer autorisierten Firma durchgeführt.

Der Hersteller garantiert den fehlerfreien Betrieb des Controllers für 12 Monate ab Verkaufsdatum, sofern die Betriebsbedingungen erfüllt sind.

Die Adresse der Verkaufsabteilung des Herstellers:

RS Components Ltd, Birchington Rd, Corby, NN17 9RS, United Kingdom, rs-online.com
RS Components GmbH, Mainzer Landstrasse 180, 60327 Frankfurt/Main, Germany, rs-online.com

Zuletzt geändert: 07.2025