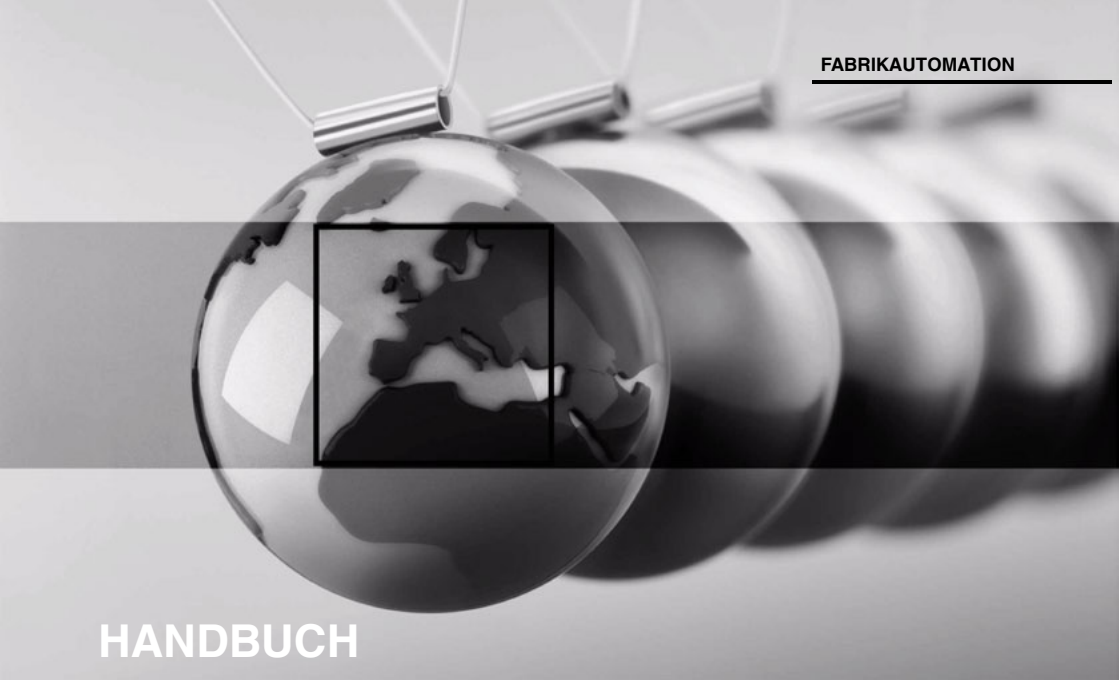


HANDBUCH

IC-KP-B12-V45

Inbetriebnahme an einer SIMATIC S-7 400 SPS

PEPPERL+FUCHS
SENSING YOUR NEEDS

Es gelten die Allgemeinen Lieferbedingungen für Erzeugnisse und Leistungen der Elektroindustrie, herausgegeben vom Zentralverband Elektroindustrie (ZVEI) e.V. in ihrer neusten Fassung sowie die Ergänzungsklausel: "Erweiterter Eigentumsvorbehalt".

2011-01

1 Sicherheit

1.1 Sicherheitsrelevante Symbole

Gefahr!
Dieses Symbol kennzeichnet eine unmittelbar drohende Gefahr.

Bei Nichtbeachten drohen Personenschäden bis hin zum Tod.

Warnung!
Dieses Zeichen warnt vor einer möglichen Störung oder Gefahr.

Bei Nichtbeachten drohen Personenschäden oder schwerste Sachschäden.

Vorsicht!
Dieses Zeichen warnt vor einer möglichen Störung.

Bei Nichtbeachten können Geräte oder daran angeschlossene Systeme und Anlagen bis hin zur völligen Fehlfunktion gestört werden.

1.2 Bestimmungsgemäße Verwendung

Die IDENTControl IC-KP-B12-V45 stellt eine Auswerteeinheit inklusive Ethernet-Schnittstelle für Identifikationssysteme dar. Das Gerät kann als Schaltschrankmodul oder für Feldanwendungen eingesetzt werden. Neben der Ethernet-Anbindung können geeignete induktive Schreib-/Leseköpfe, Mikrowellenantennen oder Triggersensoren angeschlossen werden. Dabei ist eine für das Systemkonzept geeignete Verkabelung zu verwenden.

1.3 Allgemeine Sicherheitshinweise

Das Gerät darf nur von eingewiesenem Fachpersonal entsprechend der vorliegenden Betriebsanleitung betrieben werden.

Eigene Eingriffe und Veränderungen sind gefährlich und es erlischt jegliche Garantie und Herstellerverantwortung. Falls schwerwiegende Störungen an dem Gerät auftreten, setzen Sie das Gerät außer Betrieb. Schützen Sie das Gerät gegen versehentliche Inbetriebnahme. Schicken Sie das Gerät zur Reparatur an Pepperl+Fuchs.

Der Anschluss des Gerätes und Wartungsarbeiten unter Spannung dürfen nur durch eine elektrotechnische Fachkraft erfolgen.

Die Verantwortung für das Einhalten der örtlich geltenden Sicherheitsbestimmungen liegt beim Betreiber.

Verwahren Sie das Gerät bei Nichtbenutzung in der Originalverpackung auf. Diese bietet dem Gerät einen optimalen Schutz gegen Stöße und Feuchtigkeit.

Halten Sie die zulässigen Umgebungsbedingungen ein.

2011-01

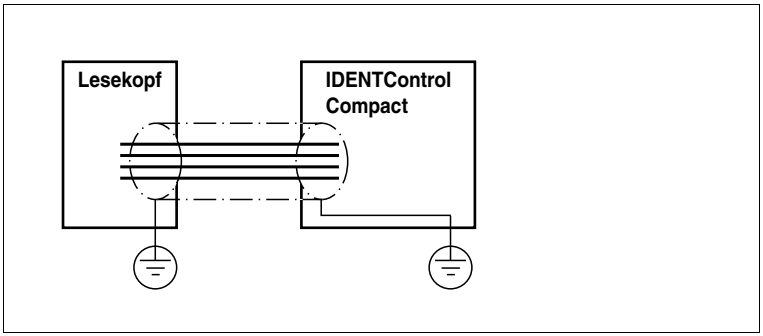
Hinweis!

Entsorgung

Elektronikschrott ist Sondermüll. Beachten Sie zu dessen Entsorgung die einschlägigen Gesetze im jeweiligen Land sowie die örtlichen Vorschriften.

1.4 Berührungsschutz

Zur Verbesserung der Störfestigkeit bestehen die Gehäuse unserer Komponenten teilweise oder ganz aus Metall.



Gefahr!

Stromschlag

Zum Schutz vor gefährlichen Spannungen im Störungsfall des SELV-Netzteils müssen die metallischen Gehäuseteile mit der Schutzerde verbunden werden!

2 Installation

2.1 Eingesetzte Geräte

In der nachfolgenden Tabelle sind alle Komponenten aufgelistet, die Sie zum Anschluss einer IDENTControl IC-KP-B12-V45 an eine SIMATIC S7-400 Steuerung mit TCP/IP Anbindung benötigen.

Komponente	Herstellerbezeichnung
Simatic S-7 400 Spannungsversorgung	PS 407 4A
Simatic S-7 400 CPU	CPU-412-2
Simatic S-7 400 Kommunikationsprozessor	CP 443-1
Auswerteeinheit IDENTControl	IC-KP-B12-V45
4 x Schreib-/Leseköpfe	IQH-18GM-V1
4 x Anschlusskabel Schreib-/Leseköpfe	V1-G-0,6M-PUR-V1-W
Netzwerkkabel	V45-G-10M-V45-G
Datenträger	IQC21-58
1 x Anschlusskabel IDENTControl	V1-G-2M-PUR

Tabelle 2.1: Auflistung der verwendeten Hardwarekomponenten

Diese Geräte sind dem in diesem Handbuch beschriebenen Testaufbau entnommen. Sie können die Inbetriebnahme des Identifikationssystems IDENTControl IC-KP-B12-V45 auch mit anderen, funktionsgleichen Steuerungseinheiten durchführen. Die dazu benötigten Informationen entnehmen Sie den zugehörigen Handbüchern der SIMATIC-Steuerung.

2.2 Aufbau und Anschluss

Der Aufbau der einzelnen Komponenten ist schematisch in der nachfolgenden Grafik dargestellt.

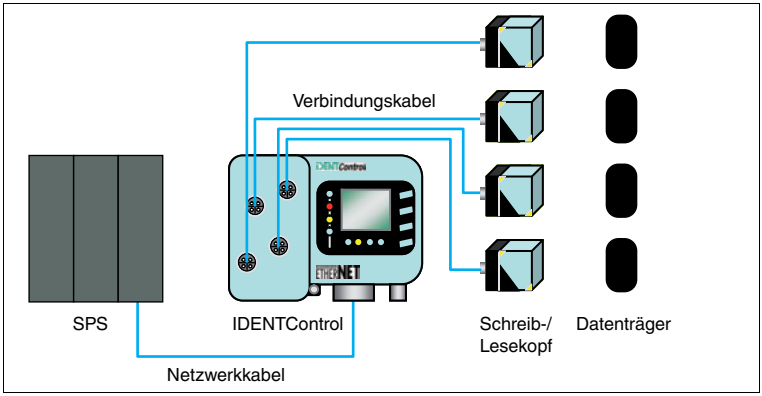


Abbildung 2.1: Aufbau der Hardwarekomponenten

Anschluss

1. Verbinden Sie die Schreib-/Leseköpfe über die Verbindungskabel mit den zugehörigen Anschlussbuchsen der IDENTControl.
2. Beachten Sie hierbei, dass Sie den richtigen Abstand zwischen zwei Schreib-/Leseköpfen einhalten, da sich die Schreib-/Leseköpfe gegenseitig beeinflussen.
3. Entnehmen Sie den korrekten Abstand dem Datenblatt des jeweiligen Schreib-/Lesekopfes.
4. Beachten Sie, dass die Datenträger sich im Erfassungsbereich der Schreib-/Leseköpfe befinden.
5. Beachten Sie zudem bei der Positionierung der Datenträger, dass sich immer nur **ein** Datenträger im Erfassungsbereich eines Schreib-/Lesekopfes befindet.
6. Verbinden Sie anschließend die IDENTControl über das zugehörige Anschlusskabel mit einer Versorgungsspannung 20 ... 30 V_{DC}
7. Stellen Sie die Kommunikationsverbindung zwischen IDENTControl und der SPS über das Netzwerkkabel her.

Hinweis!

Den genauen Aufbau der Komponenten der SPS können Sie dem Buch "Automatisierungssystem S7-400 Installationshandbuch" entnehmen. Weitere Informationen bezüglich der Inbetriebnahme finden Sie im "Handbuch IC-KPB12-V45“. Sie können das Handbuch auf www.pepperl-fuchs.com herunterladen.

2.3 Konfiguration in der Steuerung

In diesem Abschnitt erfolgt die Beschreibung der Konfiguration der Hardware innerhalb der Programmiersoftware “SIMATIC Manager“. Die Konfiguration wird anhand eines Beispiels durchgeführt.

Konfiguration

1. Passen Sie die Parameter der Einsatzumgebung für ihre spezifische Konfiguration entsprechend an.
2. Stellen Sie die Parameter direkt an der IDENTControl über die Bedienfeldtasten ein.
3. Für die Durchführung der Parametereinstellung nehmen Sie das "Handbuch IC-KPB12- V45“. zur Hand. Sie können das Handbuch auf http://www.pepperl-fuchs.com herunterladen.
4. Beachten Sie, dass im Auslieferungszustand der IDENTControl bereits Default-Parameter eingestellt sind. Die Default-Einstellung können Sie ebenfalls dem Handbuch entnehmen.

Nachfolgende Tabelle schlüsselt die im Beispiel eingestellten Parameter auf.

Parameter	Wert
IP-Adresse IDENTControl	172.16.11.25
IP-Adresse S7-400 Steuerung	172.16.11.26
Netzmaske	255.255.0.0
IP-Adresse Router	172.16.11.222

Tabelle 2.2: Parameter zur Einstellung der Hardware

Konfiguration SPS

1. Starten Sie ein neues Projekt, um die IDENTControl in Betrieb zu nehmen.
2. Binden Sie die Komponenten des Steuerungssystems über die Hardware-Konfiguration in das Projekt ein.
3. Implementieren Sie die SIMATIC400-Station in das Projekt.

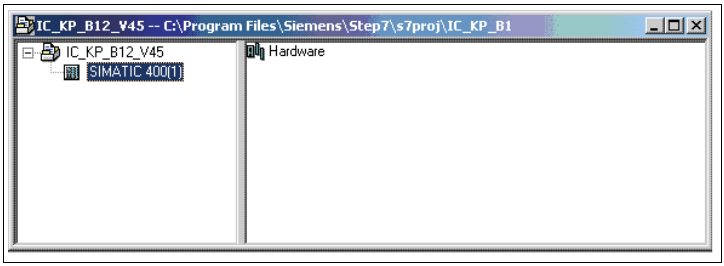


Abbildung 2.2: Einbindung SIMATIC400-Station

4. Rufen Sie die Hardware-Konfiguration durch Doppelklick auf das Symbol "Hardware“ auf.
5. Fügen Sie die benötigten Komponenten Rack, Spannungsversorgung CPU und Kommunikations-CP in die Hardware-Konfiguration ein.

   ↳ Die folgende Abbildung zeigt die Oberfläche nach der Zuweisung der Hardware-Komponenten.

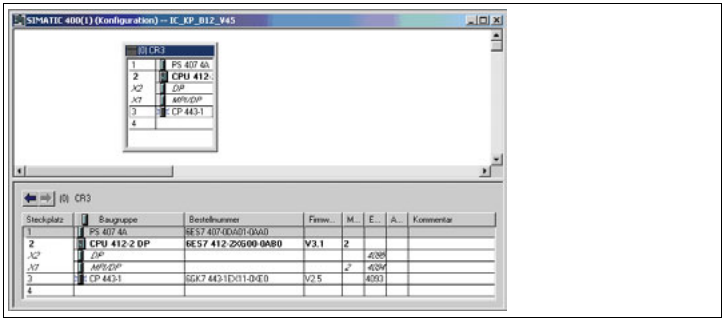


Abbildung 2.3: Hardware-Konfiguration

2011-01

6. Legen Sie die Eigenschaften der Kommunikationsschnittstelle des CP443-1 fest. Öffnen Sie dazu das Eigenschaftenfenster des CP443-1 durch Doppelklick auf das zugehörige Symbol im oberen Bild. In diesem Fenster werden die aktuellen Eigenschaften angezeigt.
7. Klicken Sie auf die Taste "Eigenschaften", um die aktuelle Einstellung zu verändern.

   ↳ Ein Fenster öffnet sich. In diesem Fenster können Sie die Parameter für die Ethernet-Schnittstelle des Kommunikations-CP einstellen.

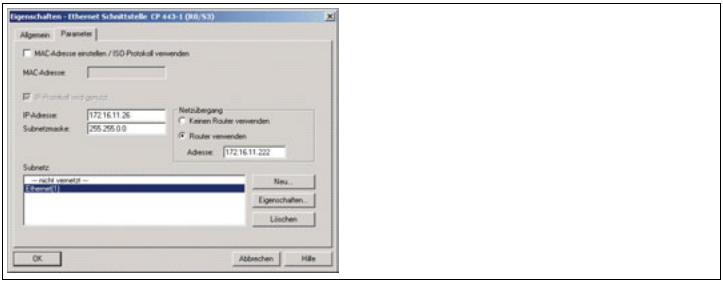


Abbildung 2.4: Einstellung Ethernet-Schnittstelle SPS

8. Geben Sie entsprechende IP-Adresse sowie die Subnetzmaske an.
9. Legen Sie den Netzübergang fest. Der Netzübergang erfolgt über einen Router. Wählen Sie die entsprechende Markierung an.
10. Tragen Sie die Adresse des Routers in das dafür vorgesehene Feld ein.
11. Verbinden Sie das Subnetz neu, indem Sie die festgelegten Parameter über die Taste "OK“ übernehmen.
12. Binden Sie die IDENTControl an das Kommunikationsnetz an. Wählen Sie dazu "Andere Station“ aus. Fügen Sie eine "Andere Station“ entweder im SIMATIC-Manager oder in der Netzkonfiguration NetPro ein.
13. In der Netzkonfiguration NetPro fügen Sie das Element "Andere Station“ in das Feld der Kommunikationspartner ein. Bezeichnen Sie das Element entsprechend der IDENTControl mit "IC-KP-B12-V45“.

    ↳ Folgende Ansicht entsteht.

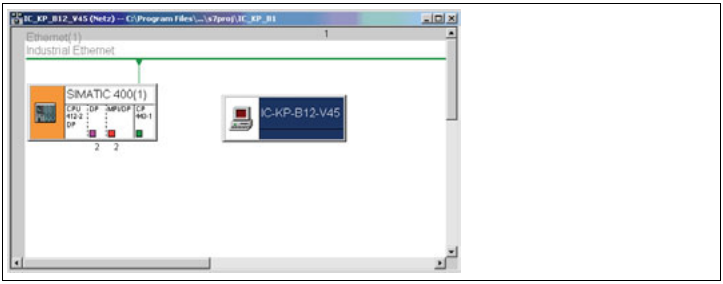


Abbildung 2.5: Einbindung IDENTControl in Netzkonfiguration NetPro

14. Um in ein Menü zur Einstellung der Kommunikation zu gelangen, doppelklicken Sie das Symbol "IC-KP-B12-V45“.
15. Stellen Sie die Schnittstelle der IDENTControl über den Reiter "Schnittstellen“ ein. Wählen Sie dazu über die Taste "Neu“ der Schnittstellentyp "Industrial Ethernet“ aus.

    ↳ Es erscheint ein Fenster, in dem Sie die IP-Adresse für die IDENTControl und den Router einstellen können. Beachten Sie bei der Einstellung, dass Sie das Ethernet-Subnetz über das Feld "Neu“ einbinden.

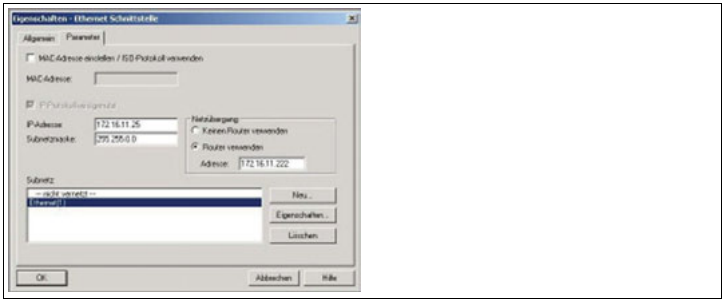


Abbildung 2.6: Einstellung Ethernet-Schnittstelle IDENTControl

16. Übernehmen Sie die Eingaben der Parameter über die Taste “OK“.
17. Fügen Sie in der Netzkonfiguration die neue Verbindung in die Verbindungstabelle der CPU ein. Die Verbindungstabelle befindet sich in der Netzkonfiguration unter dem Feld für die Kommunikationsteilnehmer.

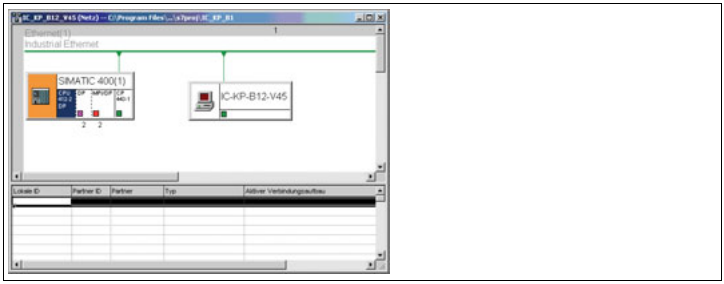


Abbildung 2.7: Netzkonfiguration mit Verbindungstabelle

18. Klicken Sie die rechte Maustaste in der oberste Zeile der Verbindungstabelle, um eine Menüauswahl zu öffnen. Wählen Sie den Unterpunkt "Neue Verbindung einfügen“ aus.

    ↳ Ein Auswahlfenster öffnet sich. Stellen Sie in diesem Fenster die Verbindungspartner ein.

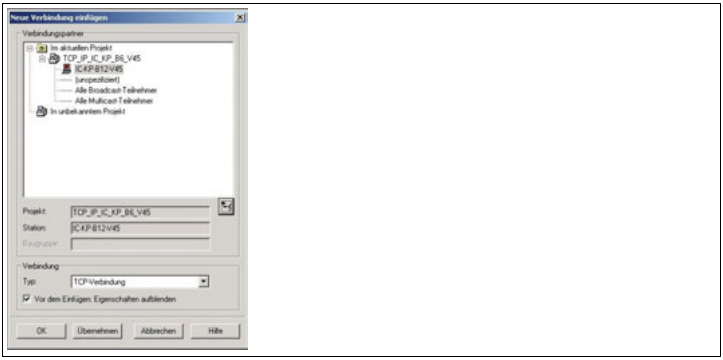


Abbildung 2.8: Verbindungspartner

19. Stellen Sie in diesem Fenster die Verbindungspartner ein.
20. Als Verbindungspartner wählen Sie die "Andere Station“ bzw. IC-KP-B12-V45.
21. Wählen Sie als Verbindungstyp die "TCP-Verbindung" aus.
22. Bestätigen Sie die eingestellten Parameter mit der Taste "Übernehmen“.

    ↳ Das Fenster "Eigenschaften TCP-Verbindung“ öffnet sich. Über die Karteikarte "Allgemein“ wird die aktuelle Verbindungs-ID angezeigt.
23. Öffnen Sie die Karteikarte "Adressen“.

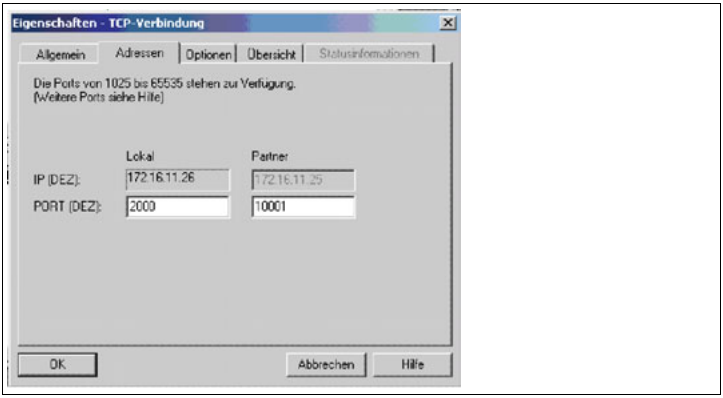


Abbildung 2.9: Auswahl Portnummer

24. Legen Sie die Portnummern der Kommunikationsteilnehmer fest. Für die IDENTControl ist als Portnummer sowohl 10000 als auch 10001 zulässig. Wenn Sie den Port 10000 auswählen, werden 34 Byte große Datenpakete zwischen SPS und IDENTControl ausgetauscht. Wenn Sie den Port 10001 auswählen, haben die Datenpakete eine Größe von 66 Bytes.

    ↳ Die Hardware der IDENTControl ist an die Steuerung angebunden.

3 Befehle

3.1 Befehlsarten

Die Befehle der IDENTControl, die mit Hilfe des Steuerungsprogramms ausgeführt werden können, sind in 2 Arten zu untergliedern. Man unterscheidet zwischen einem Single- und einem Enhanced-Befehl.

Single-Befehl

Bei einem Single-Befehl erfolgt die Befehlsausführung einmalig, sofern sich ein Code-/Datenträger im Erfassungsbereich des Schreib-/Lesekopfes befunden hat. Wenn sich bei der Befehlsausführung kein Code-/Datenträger erfassen lässt, wird die Befehlsausführung durch die Steuerung so lange wiederholt, bis der Befehl vollständig ausgeführt wurde oder aber die maximale Anzahl der Wiederholungen überschritten wurde.

Hinweis!

Die Anzahl der Wiederholungen ist vom Anwender selbst festzuglegen, um eine Blockierung des Schreib-/Lesekopfes durch die dauerhafte Befehlsausführung zu verhindern.

Enhanced-Befehl

Bei einem Enhanced-Befehl erfolgt die Befehlsausführung dauerhaft, die bei verschiedenen Anwendungen erfoderlich ist. Der Befehl wird so lange ausgeführt, bis die Bearbeitung durch einen Quit-Befehl abgebrochen wird.

3.2 Befehlsablauf

Das Steuerungsprogramm überträgt die Befehlsparameter, die notwendig für die Ausführung eines Befehls sind, an die IDENTControl.

Die IDENTControl sendet bei der Annahme des Befehls die Statusrückmeldung (FF)h an die Steuerung zurück.

Diese Statusrückmeldungen signalisiert, dass die IDENTControl den Befehl angenommen hat und und ihn bearbeitet.

Die IDENTControl leitet den Befehl an die Schreib-/Leseköpfe weiter und wartet auf eine Rückantwort von den Köpfen.

Die IDENTControl leitet die Rückantwort von den Schreib-/Leseköpfen an die Steuerung weiter.

Die empfangenen Dateien werden innerhalb der Steuerung ausgewertet und dem Anwender zur Verfügung gestellt.

Die IDENTControl leitet Rückantworten der Schreib-/Leseköpfe bei der Durchführung eines Enhanced-Befehls so lange an die Steuerung weiter, bis der Befehl abgebrochen wird.

2011-01

3.3 Befehlsstruktur

Um einen Befehl mit der IDENTControl ausführen zu können, müssen vorher die zugehörigen Befehlsparameter an die IDENTControl übertragen werden.

Die Befehlsparameter sind byteweise zusammengefasst und bilden ein Befehlstelegramm.

Die Anzahl der zu übertragenden Parameter sowie der Befehlscode sind unterschiedlich, jedoch sind die Strukturen der verschiedenen Befehle identisch.

Die nachfolgende Tabelle beschreibt den Aufbau eines Befehlstelegramms für einen Single-Read-Data-Befehl.

Dieses Telegramm wird von der Steuerung zur IDENTControl gesendet und bewirkt das Einlesen von Nutzdaten.

Byte	Inhalt / Variable	Bitbelegung							
0	#Head_X.OutData.TelegrammLengthHigh	0	0	0	0	0	0	0	0
1	#Head_X.OutData.TelegrammLengthLow	0	1	0	0	0	0	1	0
2	#Head_X.OutData.CommandCode	0	0	0	1	0	0	0	0
3	#Head_X.OutData.Channel	Datenwortanzahl				Kanalnummer			
4	#Head_X.OutData.Wordadr_High	Anfangsadresse high Byte							
5	#Head_X.OutData.Wordadr_Low	Anfangsadresse low Byte							
6..65	#Head_X.OutData.DW1 ... DW15	unbenutzt							

Abbildung 3.1: Aufbau Befehlstelegramm Single-Read-Data-Befehl

- In den ersten beiden Bytes des Befehlstelegramms wird die Telegrammlänge übermittelt.
  Dies wird über die Parameter **#Head_X.OutData.TelegrammLengthHigh** und **#Head_X.OutData.TelegrammLengthLow** realisiert.
- Der Befehlsparameter **#Head_X.OutData.TelegrammCommandCode** enthält den Befehlscode des zu übertragenden Befehls.
  Für ein Single-Read-Data-Befehl hat der Befehlscode die binäre Codierung (10000)b.

  Eine ausführliche Auflistung der verschiedenen Befehlscodes für die IDENTControl ist dem "Handbuch IDENTControl IC-KP-B12-V45" (www.pepperl-fuchs.com) oder der Befehlsliste im Anhang zu entnehmen.
- Mit Hilfe des Befehlsparameters **#Head_X.OutData.TelegrammChannel** werden zwei Befehlseigenschaften übertragen.

- Durch die oberen Bits wird die Anzahl der zu lesenden / schreibenden Datenblöcke übertragen.
  Die Anzahl ist in der Variable **#Head_X.WordNum** enthalten und wird während der Befehlszuweisung in den Befehlsparameter übertragen.
- Die Datenblöcke weisen ein Doppeltwort-Format auf und haben somit eine Größe von vier Byte.
  Die maximale Anzahl der zu übertragenden Datenblöcke ist abhängig von der Hardware-Konfiguration.

  Bei der Auswahl von Port 10000 können maximal sieben Datenblöcke übertragen werden.

  Wenn mehr Daten übertragen werden sollen, so muss Port10001 in der Hardware-Konfiguration definiert werden. Mit dieser Einstellung werden maximal 15 Datenblöcke übertragen.
- Die unteren vier Bytes enthalten die Information über des Kanal des Schreib-/Lesekopfes, an welchem der Befehl ausgeführt werden soll.

Die nachfolgende Tabelle listet die verschiedenen Kanalcodierungen auf, über die der zugehörige Schreib-/Lesekopf angesprochen wird.

Schreib-/Lesekopf	Codierung	
	dez	dual
1	2	(0010)
2	4	(0100)
3	6	(0110)
4	8	(1000)

Abbildung 3.2: Kanalcodierung der Schreib-/Leseköpfe

Vorsicht!

Beachten Sie bei der Codierung des Kanals, dass das niedrigste BIT (LSB) nicht verwendet wird.

Bei der Kommunikation über Profibus wird es durch toggeln dazu genutzt, um bei nacheinander folgenden Befehlen eine Unterscheidung treffen zu können.

Wenn die Kommunikation allerdings über TCP/IP erfolgt, so ist das Bit nicht erforderlich. Sie brauchen es nicht weiter zu beachten.

- Mit Hilfe der Parameter **#Head_X.OutData.Wordadr_High** und **#Head_X.OutData.Wordadr_Low** kann eine Startadresse festgelegt werden, ab welcher die Datenblöcke gelesen bzw. geschrieben werden.

- Die Befehlsparameter **#Head_X.OutData.DW 1 ... DW 15** enthalten die Nutzdatenwörter.
  Allerdings bleiben diese Befehlsparameter beim Senden des Befehlstelegramms an die IDENTControl unbenutzt.

  Die IDENTControl antwortet auf ein Befehlstelegramm mit einem Antworttelegramm. Der Aufbau eines Antworttelegramms ist in der nachfolgenden Tabelle dargestellt.

Byte	Inhalt / Variable	Bitbelegung							
0	#Head_X.InData.TelegrammLengthHigh	0	0	0	0	0	0	0	0
1	#Head_X. InData.TelegrammLengthLow	0	1	0	0	0	0	1	0
2	#Head_X. InData.CommandCode	0	0	0	1	0	0	0	0
3	#Head_X. InData.Channel	Datenwortanzahl				Kanalnummer			
4	#Head_X. InData. Status	Status							
5	#Head_X. InData. ExecutionCounter	Ausführungszähler							
6..(33)65	#Head_X. InData.DW1 ... .DW(7)15	00 .. FF							

Abbildung 3.3: Aufbau Antworttelegramm Single-Read-Data-Befehl

- In dem Antworttelegramm werden die ersten vier Byte des zuvor gesendeten Befehlstelegramms zurückgesendet.
  Außerdem enthält das Antworttelegramm eine Statusmeldung, die Aufschluss über den Ausführungszustand des Befehls gibt.

  Mit Hilfe des Ereigniszählers kann die Anzahl der Befehlsereignisse ermittelt werden.

  Bei jedem Befehlsereignis wird der Zähler um eins erhöht. Ein Befehlsereignis ist zum Beispiel ein Statuswechsel von (00)h auf (FF)h.

- Die weiteren Parameter des Antworttelegramms enthalten die eingelesenen Nutzdaten.
  Die Anzahl der eingelesenen Nutzdaten ist dabei wiederum abhängig von der in der Hardware-Konfiguration getroffenen Portzuweisung.

3.4 Durchführung der Initialisierung

Die Initialisierung wird durchgeführt, weil man mit Hilfe dieser feststellen kann, an welchem IDENT-Kanal ein Schreib-/Lesekopf angeschlossen ist.

Durchführung der Initialisierung

Initialisieren Sie zu Beginn der Bearbeitung des Funktionsbausteins die Schreib- und Leseköpfe.

↳ Die Initialisierung wird für jeden Kopf nacheinander durchgeführt.

Bei der Initialisierungs-Routine wird ein Change-Tag-Befehl an die IDENTControl gesendet. Durch diesen Befehl wird dem Schreib-/Lesekopf mitgeteilt, mit welchem Datenträger er kommuniziert. Anhand der Statusrückmeldung erkennt der Funktionsbaustein, ob ein Schreib-/Lesekopf am entsprechenden Kanal angeschlossen ist.

Wenn die Initialisierung für einen Kopf abgeschlossen ist, wird entweder das Bit **#Head_X.ExistTC** oder **#Head_X.NotExist** gesetzt.

Das Bit **#Head_X.ExistTC** ist gesetzt, wenn ein Schreib-/Lesekopf an dem zugehörigen Kanal angeschlossen ist. Andernfalls ist das Bit **#Head_X.NotExist** gesetzt.

1. Beachten Sie bei der Durchführung der Initialisierung, dass die richtige Datenträgerkennung zugewiesen wird.

   ↳ Die Kennung für den Datenträger wird der Variablen **#Head_X.TagType** zugewiesen.
2. Legen Sie die Kennung selbst fest.

   ↳ Die Angabe der Datenerkennung erfolgt innerhalb des Programms in hexadezimaler Form.
3. Entnehmen Sie die Kennung der datenträger aus dem "Handbuch IDENTControl IC-KP-B12-V45" unter der Beschreibung des Change-Tag-Befehls.
4. Beachten Sie, dass die Angabe innerhalb des Handbuchs im ASCII-Zeichnformat erfolgt. Eine Umrechnung kann mit Hilfe einer, ebenfalls im Handbuch aufgeführten, ASCII-Tabelle durchgeführt werden.
5. Sie können die Datenträgerkennung ebenso aus der Auflistung für die Code-/Datenträger im Anhang entnehmen.

   ↳ Nach erfolgreich durchgeführter Initialisierung aller Köpfe kann ein Befehl durch die IDENTControl durchgeführt werden.

3.5 Durchführung eines Single-Befehls

Ein Single-Befehl wird von der IDENTControl nur einmalig ausgeführt. Zur Ausführung des Befehls ist es erforderlich, dass die zugehörigen Befehlsparameter vom Ausgangsdatenfeld des zugehörigen Kopfes innerhalb der SPS zur IDENTControl gesendet werden. Um das Ausgangsdatenfeld an die IDENTControl senden zu können, muss es mit den zugehörigen Befehlsparametern und Datenwörtern belegt werden. Auf den Befehlsaufbau wird an dieser Stelle nicht näher eingegangen. Nähere Informationen befinden sich in Abschnitt 3.3 Befehlsstruktur.

Bevor Sie den Single-Befehl ausführen, definieren Sie unterschiedliche IN-Variablen.

↳ Durch den Parameter **#HeadXDataFixcode** wird festgelegt, ob ein Zugriff auf den Datenbereich (**#HeadXDataFixcode = 0**) oder auf den Fixcode (**#HeadXDataFixcode = 1**) durchgeführt wird. Ein Single-Befehl wird ausgeführt, wenn die Variable **#HeadXSingleEnhanced** nicht gesetzt ist. Der Befehl wird angestoßen, wenn die IN-Variablen **#HeadXRead** bzw. **#HeadXWrite** gesetzt sind.

Durch das Setzen einer dieser beiden Parameter beginnt die Befehlszuweisung innerhalb des Funktionsbausteins.

Anschließend wird das Ausgangsdatenfeld mit den Befehlsparametern an die IDENTControl übertragen und ausgeführt. Die IDENTControl sendet nach Erhalt der Befehlsparameter Statusmeldungen an die SPS zurück. Das Senden der Statusrückmeldungen wird solange durchgeführt, bis der Befehl vollständig abgearbeitet ist.

Die Statusmeldung wird innerhalb des Funktionsbausteins dazu benutzt, um eine Fehlerauswertung durchzuführen. Die genaue Bedeutung der einzelnen Statuswerte ist dem "Handbuch IDENTControl IC-KP-B12-V45" (www.pepperl-fuchs.com) zu entnehmen.

Durch die Auswertung der Statusinformationen können kopfbezogene Statusbits generiert werden. Anhand dieser Statusbits kann eine Aussage über den Ausführungszustand des Befehls getroffen werden. Das Statusbit **#Head_X.Busy** signalisiert, dass der Befehl noch von der IDENTControl ausgeführt wird. Wenn das Bit gesetzt ist, kann kein Befehl an diesem Kopf gestartet werden. Das Bit wird vom Funktionsbaustein zurückgesetzt, wenn die Abarbeitung des Befehls durch die IDENTControl beendet wurde.

Das Ende der Befehlsbearbeitung wird durch das Statusbit **#Head_X.Done** gekennzeichnet. Nach Beendigung der Befehlsausführung durch die IDENTControl wird dieses Bit selbständig durch den Funktionsbaustein gesetzt. Eine erneute Befehlsausführung ist an einen Kopf erst dann möglich, wenn das Bit **#Head_X.Busy** nicht gesetzt und das Bit **#Head_X.Done** gesetzt ist. Befindet sich zum Zeitpunkt der Befehlsausführung kein Datenträger im Erfassungsbereich des Schreib-/Lesekopfes, so wird das Statusbit **#Head_X.NoDataCarrier** gesetzt.

Daraufhin wird die Befehlsausführung automatisch von der IDENTControl wiederholt. Die maximale Anzahl der Wiederholungen wird durch die IN-Variable **#RetrySingleCommand** vom Anwender vorgegeben. Die Anzahl der maximalen Wiederholungen gilt für alle Schreib-/Leseköpfe.

Wenn bei der Befehlsbearbeitung ein Timeout aufgetreten ist, so wird das Bit **#Head_X.TimeoutOccured** gesetzt. Ein Timeout tritt dann auf, wenn das Ende der Befehlsbearbeitung nicht innerhalb der durch die IN-Variable **#Timeout** vorgegebenen Zeitspanne erfolgte. Dies wird als Fehler betrachtet und das Bit **#Head_X.Error** gesetzt.

Das Bit **#Head_X.Error** signalisiert das Auftreten eines Fehlers in der Befehlsausführung der IDENTControl. Sobald das Bit **#Head_X.Error** gesetzt ist, ist der zugehörige Kopf für weitere Befehlsausführungen gesperrt. Die Sperre kann durch die IN-Variable **#QuitErrorHeadX** aufgehoben werden.

Die eingelesenen Nutzdatenblöcke befinden sich innerhalb des Funktionsbausteins in den Variablen **#Head_X.InData.DW1...DW15**. Nach Beendigung der Befehlsbearbeitung können die eingelesenen Daten weiterverarbeitet werden.

Die Datenblöcke, welche auf einen Datenträger geschrieben werden sollen, befinden sich in den Variablen **#Head_X.OutData.DW1...DW15**. Die Datenblöcke sind frei durch den Anwender festzulegen und müssen vor dem Start der Befehlsausführung zugewiesen werden. Die für die Ausführung eines Single-Befehls notwendigen Einstellungen sind der Befehlsliste im Anhang zu entnehmen.

3.6 Durchführung eines Enhanced-Befehls

Ein Enhanced-Befehl wird von der IDENTControl so lange ausgeführt, bis er durch einen Quit-Befehl abgebrochen wird. Bei einem Enhanced-Befehl wird der Schreib-/Lesevorgang so lange durchgeführt, bis die Daten auf den Datenträger geschrieben bzw. vom Datenträger gelesen werden. Im Gegensatz zum Single-Befehl bleibt der Enhanced-Befehl auch nach der Durchführung des Schreib-/Lesevorgangs aktiv. Verlässt ein alter Datenträger den Erfassungsbereich der Schreib-/Leseköpfe, so kann ein neuer Datenträger innerhalb des Erfassungsbereiches ausgelesen oder beschrieben werden. Bei der Ausführung des Enhanced-Befehls ist zu beachten, dass sich immer nur ein Datenträger im Erfassungsbereich des Schreib-/Lesekopfes befindet.

Durchführung eines Enhanced-Befehls

1. Legen Sie für die Ausführung von Enhanced-Befehlen analog zum Single-Befehl verschiedene IN-Parameter fest.
2. Wie bei einem Single-Befehl wird durch die Variable **#HeadXDataFixcode** festgelegt, ob auf den Datenbereich oder Fixcode zugegriffen werden soll.
3. Setzen Sie vor Beginn der Befehlsausführung die Variable **#HeadXSingleEnhanced**.

   ↳ Mit Hilfe von dieser erfolgt die Ausführung eines Enhanced-Befehls. Durch die Parameter **#HeadXRead** und **#HeadXWrite** wird die Befehlsausführung analog zum Single-Befehl angestoßen.

Der Befehl wird nun von der IDENTControl bearbeitet, was durch das Bit **#Head_X.Busy** angezeigt wird. Das Bit bleibt gesetzt, bis der Enhanced-Befehl durch einen Fehler oder einen Quit-Befehl abgebrochen wurde.

Das Bit **#Head_X.Done** signalisiert bei einem Enhanced-Befehl, dass ein Code-/Datenträger gelesen oder beschrieben wurde. Im Gegensatz zum Single-Befehl wird hier nicht die Beendigung eines Befehls signalisiert. Entfernt sich der Code-/Datenträger wieder aus dem Erfassungsbereich des Schreib-/Lesekopfes, wird das Bit **#Head_X.Done** automatisch zurückgesetzt, aber der Befehl ist noch aktiv.

Eine Ausführung eines neuen Befehls ist erst dann möglich, wenn das Bit **#Head_X.Busy** zurückgesetzt und das Bit **#Head_X.Done** gesetzt ist. Diese Bedingung wird nach der Ausführung eines Quit-Befehls erreicht.

2011-01

Wie bei der Ausführung eines Single-Befehls befinden sich die Nutzdaten, die auf einen Datenträger geschrieben werden sollen, in den Variablen **#Head_X.OutDataDW1...DW15**. Die Ausgangsdaten müssen vor der Ausführung eines Enhanced-Befehls parametriert werden. Eine Änderung der Ausgangsdaten im laufenden Betrieb hat keine Asuwirkung auf den laufenden Enhanced-Befehl. Die Eingangsdaten werden in den Variablen **#Head_X.InData.DW1...DW15** innerhalb des Instanz-DB abgelegt. Die notwendigen Einstellungen für die Ausführung eines Enhanced-Befehls sind der Befehlsliste im Anhang zu entnehmen.

3.7 Durchführung eines Special-Befehls

Mit Hilfe eines Special-Befehls kann der Anwender einen Befehl selbständig parametrieren. Der Special-Befehl kann in erster Linie für die Durchführung von Befehlen genutzt werden, die nicht in der Befehlsliste des Funktionsbausteins enthalten sind. Die Ausführung von Standardbefehlen z.B. Single-Read/-Write ist über den Special-Befehl allerdings ebenfalls möglich.

Durchführung eines Special-Befehls

1. Bevor Sie einen Special-Befehl durchführen, übergeben Sie die Befehlsparameter in ein eigens dafür vorgesehenes Datenfeld **#Head_X.SpecialCommand** innerhalb des Instanz-Datenbausteins.
2. Weisen Sie den Befehlscode des auszuführenden Befehls dabei der Variable **#Head_X.SpecialCommand.Code** zu.

Eine Kanalszuweisung ist für die Ausführung eines Special-Befehls am entsprechenden Schreib-/Lesekopf nicht notwendig. Die entsprechende Kanalzuweisung wird innerhalb des Funktionsbausteins automatisch durchgeführt.

Geben Sie die Anzahl der zu übertragenden Nutzdatenwörter bei der Ausführung eines Read-/Write-Befehls mit Hilfe eines Special-Befehls in der Variablen **#Head_X.SpecialCommand.Channel** an. Dazu werden die oberen vier Bit der Variablen genutzt.

Für die Parametrierung eines Special-Befehls stehen weitere Variablen zur Verfügung. Durch die Variablen **#Head_X.SpecialCommand.Parameter1...6** können weitere Befehlsparameter zugewiesen werden. Nähere Informationen über die Befehlsparametrierung sind dem "Handbuch IDENTControl IC-KP-B12-V45" in der Befehlsübersicht zu entnehmen.

Die Ausführung eines Special-Befehls wird durch das Setzen der IN-Variable **#HeadXSpecialCommand** gestartet. Daraufhin wird das Datenfeld **#Head_X.SpecialCommand** dem Ausgangsdatenfeld **#Head_X.OutData** des zugehörigen Kopfes übergeben und an die IDENTControl gesendet. Der übermittelte Befehl wird anschließend von der IDENTControl ausgeführt.

Wenn durch den Special-Befehl ein Enhanced-Befehl gestartet wurde, muss Sie ihn wieder über ein Quit-Befehl am zugehörigen Kopf abbrechen.

Mit Hilfe des Special-Befehls können auch Befehle an die IDENTControl gesendet werden, die nicht von den Schreib-/Leseköpfen, sondern nur von der IDENTControl bearbeitet werden. Diese Befehle werden als IDENTControl-Befehle bezeichnet. Ein Beispiel für einen IDENTControl-Befehl ist der Befehl Set-Multiplexed-Mode. Wie bei einem Special-Befehl erfolgt die Parametrierung über das Datenfeld **#Head_1.SpecialCommand**. Da die Befehle nicht von den Schreib-/Leseköpfen ausgeführt werden, ist keine Kanalzuweisung erforderlich. Die Ausführung des Befehls wird durch das Setzen der IN-Variablen **#Head1SpecialCommand** und **#IC_Command_on_Head1** gestartet. Daraufhin wird das Datenfeld **#Head_1.SpecialCommand** an das Ausgangsdatenfeld **#Head_1.OutData** übergeben und an die IDENTControl übertragen und ausgeführt.

3.8 Durchführung einer Fehlerauswertung

Bei der Ausführung eines Befehls kann ein Fehler auftreten. Der Funktionsbaustein bietet die Möglichkeit eine Fehlerauswertung durchzuführen. Wenn bei der Ausführung eines Befehls ein Fehler auftritt, wird das Bit **#Head_X.Error** gesetzt. Der Kopf ist daraufhin für eine weitere Befehlsausführung gesperrt. Für die Fehlerauswertung stehen verschiedene Parameter zur Verfügung.

In der nachfolgenden Tabelle sind die verschiedenen Parameter zur Fehlerauswertung aufgeführt.

Parameter	Bedeutung
#Head_X.Error	Fehler in der Befehlsausführung
#Head_X.InvalidResponse	Ungültige Antwort der IDENTControl
#Head_X.TimeoutOccured	Timeout abgelaufen
#Head_X.Error_FC_Recv	Fehler bei der Ausführung FC 60
#Head_X.Error_FC_Send	Fehler bei der Ausführung FC 50
#Head_X.RetVal_FC_Recv	Fehlercode FC 60
#Head_X.RetVal_FC_Send	Fehlercode FC 50
#Memory_Error_FC_Recv	Fehler bei der Ausführung FC 60
#Memory_RetVal_FC_Recv	Fehlercode FC 60
#DynErrorTCPConnection	Unterbrechung der Kommunikation
#Error_FC_Recv	Fehler bei der Ausführung FC 60
#Ret_Val_SFC20	Fehlercode SFC 20

Tabelle 3.1: Fehlerauswertung

Die Fehlerverriegelung eines Schreib-/Lesekopfes kann über den Quit-Error-Befehl aufgehoben werden. Der Quit-Error-Befehl wird durch das Setzen der IN-Variable **#QuitErrorHeadX** gestartet. Durch diesen Befehl werden alle Bits zurückgesetzt, die eine Befehlsausführung anderer Befehle unterbinden. Dadurch können Befehle am entsprechenden Schreib-/Lesekopf ausgeführt werden, sofern kein weiterer Fehler eine Verriegelung verursacht.

2011-01

4 Verwendete Bausteine und ihre Funktionalität

Die Anbindung einer IDENTControl an eine SPS wird über einen Funktionsbaustein realisiert. Ein Funktionsbaustein ist frei programmierbar und kann somit optimal auf die Anwendung zugeschnitten werden. Die für die Bearbeitung eines Funktionsbaustein anfallenden Daten werden in einem Instanz-Datenbaustein abgespeichert. Ebenso sind noch weitere verschiedene Systemfunktionen unterschiedlicher Funktionalität für die Realisierung der Kommunikation zwischen IDENTControl und SPS erfoderlich. Nachfolgende Tabelle gibt einen Überblick über die verwendeten Bausteine und ihre Funktion.

Baustein	Typ	Funktion
OB1	Organisationsbaustein	Wird vom Betriebssystem zyklisch durchlaufen
FB 10 "IDENTControl"	Funktionsbaustein	Steuerung des Kommunikations-*ablaufs mit der IDENTControl
DB 10 "InstDB"	Instanzen-Datenbaustein	Speicherung der anfallenden Lokaldaten
FC 50 "AG_LSEND"	Funktion	Senden von Daten über eine projektierte Ethernet-Verbindung
FC 60 "AG_LRECV"	Funktion	Empfangen von Daten über eine projektierte Ethernet-Verbindung
SFC 20 "BLKMOV"	Systemfunktion	Umkopieren eines Speicher-bereiches
SFB 5 "TOF"	Systemfunktionsbaustein	Erzeugung einer Ausschaltverzö-gerung
SFC 58 "WR_REC"	Systemfunktion	Übertragen eines Datensatzes an eine Baugruppe
SFC 59 "RD_REC"	Systemfunktion	Einlesen eines Datensatzes von einer Baugruppe

Tabelle 4.1: Verwendete Bausteine und ihre Funktion

Der Zusammenhang zwischen den einzelnen Bausteinen wird in der nachfolgenden Grafik dargestellt.

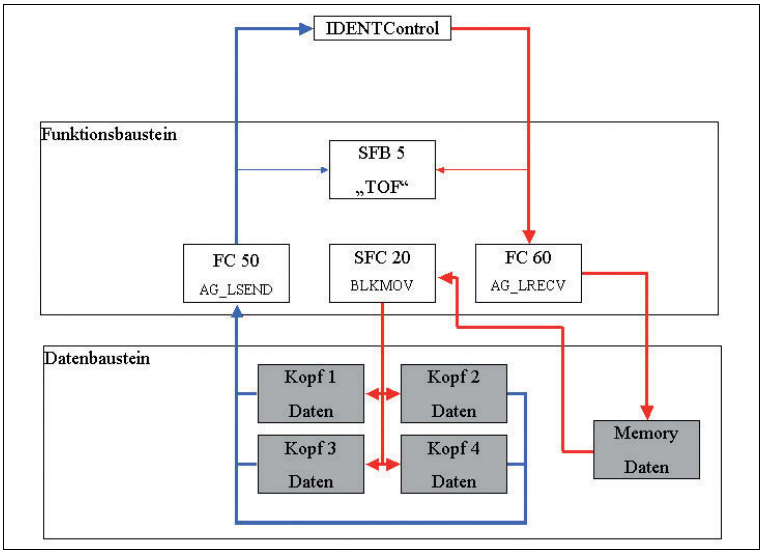


Abbildung 4.1: Zusammenhang der Bausteine

In dieser Grafik sehen Sie neben den verwendeten Bausteinen auch die Datenströme, welche zwischen der SPS und der IDENTControl ausgetauscht werden. Die jedem Kopf zugeordneten Datenfelder sind in ein Eingangs- und ein Ausgangsdatenfeld untergliedert.

Die Funktion FC 50 führt das Senden der Daten von der SPS zur IDENTControl durch. Die Funktion sendet Daten über eine projektierte Ethernet-Verbindung. Innerhalb der Funktion FC 50 wird die Systemfunktion SFC58 aufgerufen.

Durch die Systemfunktion SFC 58 werden Daten an eine projektierte Baugruppe übertragen. Beim Senden der Daten wird gleichzeitig die Ausschaltverzögerung (SFB 5) aktiviert, um somit die Antwortzeit der IDENTControl zu überwachsen.

Die Funktion FC 60 realisiert den Empfang der Daten. Sie ruft dabei selbständig die Systemfunktion SFC 59 auf.

Durch die Systemfunktion SFC 90 werden Daten von einer projektierten Baugruppe eingelesen.

Die Ausschaltverzögerung wird deaktiviert, sobald Daten von der IDENTControl empfangen werden. Die eingelesenen Daten werden in einem Memory-Datenfeld innerhalb des Instanz-DB abgespeichert. Die Daten innerhalb des Memory-Datenfeldes werden analysiert und danach durch die Systemfunktion SFC 20 in das Eingangsdatenfeld des zugehörigen Kopfes kopiert.

2011-01

5 Organisationsbaustein OB 1

Der Organisationsbaustein OB 1 wird von dem Betriebssystem der CPU zyklisch durchlaufen. Der OB 1 bildet die Schnittstelle zwischen dem Betriebssystem der CPU und dem Anwenderprogramm. Zu Beginn des OB 1 wird zunächst die Datenträgerkennung in den Instanz-Datenbaustein geladen.

Beachten Sie, dass die Zuweisung der Datenträgerkennung für die Ausführung der Initialisierung erforderlich ist.

Führen Sie diese vor der Ausführung der Initialisierung durch.

Zuweisung der Datenträgerkennung

1. Achten Sie bei der Zuweisung der Datenträgerkennung darauf, dass die Angabe der Kennung in hexadezimaler Form erfolgt.

   ↳ Die Datenträgerkennung wird der Variablen **#Head_X.TagType** übergeben.

2. Führen Sie die Zuweisung der Datenträgerkennung für jeden angeschlossenen Schreib-/Lesekopf durch.

Anschließend erfolgt die Zuweisung der Anzahl der zu übertragenden Nutzdatenwörter. Die Anzahl der maximal übertragbaren Nutzdatenwörter ist abhängig von der in der Hardware-Konfiguration getroffenen Porteinstellung. Der Anwender kann aber die Anzahl in den vorgegebenen Grenzen frei bestimmen. Die Zuweisung wird über den Parameter **#Head_X.WordNum** durchgeführt.

1. Beachten Sie dabei, dass Sie die Zuweisung immer vor dem Start eines Befehls durchführen.
2. Beachten Sie außerdem, dass Sie während der Befehlsausführung keine Zuweisungen verändern.
3. Rufen Sie den Bausteinaufruf "Call" auf.

   ↳ Der Funktionsbaustein "IDENTControl" und der zugehörige Instanz-Datenbaustein "InstDB" werden gestartet.

Beispiel: Call "IDENTControl“ , "InstDB“.

Der Funktionsbaustein ist multiinstanzfähig. Darunter ist zu verstehen, dass einem Funktionsbaustein mehrere Instanzen zugewiesen werden können. Das ermöglicht die Anbindung mehrerer IDENTControl in eine Steuerung.

Rufen Sie dazu den Baustein mehrmals auf.

Beispiel: Call "IDENTControl“ , "InstDB1“ Call "IDENTControl“ , "InstDB2“

6 Funktionsbaustein "IDENTControl"

Der Funktionsbaustein "IDENTControl“ hat die Aufgabe, das Identifikationssystem IDENTControl in die Steuerung einzubinden. Ein Funktionsbaustein ist durch den Anwender frei programmierbar. Dadurch kann die Funktionalität der IDENTControl auf die Anwendung angepasst werden.

Der Funktionsbaustein ist in verschiedene Netzwerke untergliedert. Nachfolgende Tabelle stellt die verschiedenen Netzwerke sowie deren Aufgabe innerhalb des FBs dar.

Netzwerk	Aufgabe
1	Start-Up-Sequence
2	Initialisierung Kopf 1
3	Initialisierung Kopf 2
4	Initialisierung Kopf 3
5	Initialisierung Kopf 4
6	Timeout-Überwachung
7	Umsetzung von IN- auf STAT-Variablen
8	Einlesen der Rückantwort
9	Befehlszuweisung Kopf 1
10	Befehlszuweisung Kopf 2
11	Befehlszuweisung Kopf 3
12	Befehlszuweisung Kopf 4
13	Befehlsausführung Kopf 1
14	Befehlsausführung Kopf 2
15	Befehlsausführung Kopf 3
16	Befehlsausführung Kopf 4
17	Restart-Routine
18	Quittieren
19	Error-Routine
20	Analyse FC 50
21	Analyse FC 60
22	Analyse der Eingangsdatenfelder

Tabelle 6.1: Netzwerke des Funktionsbausteins

6.1 Ablauf der Start-UP-Sequence

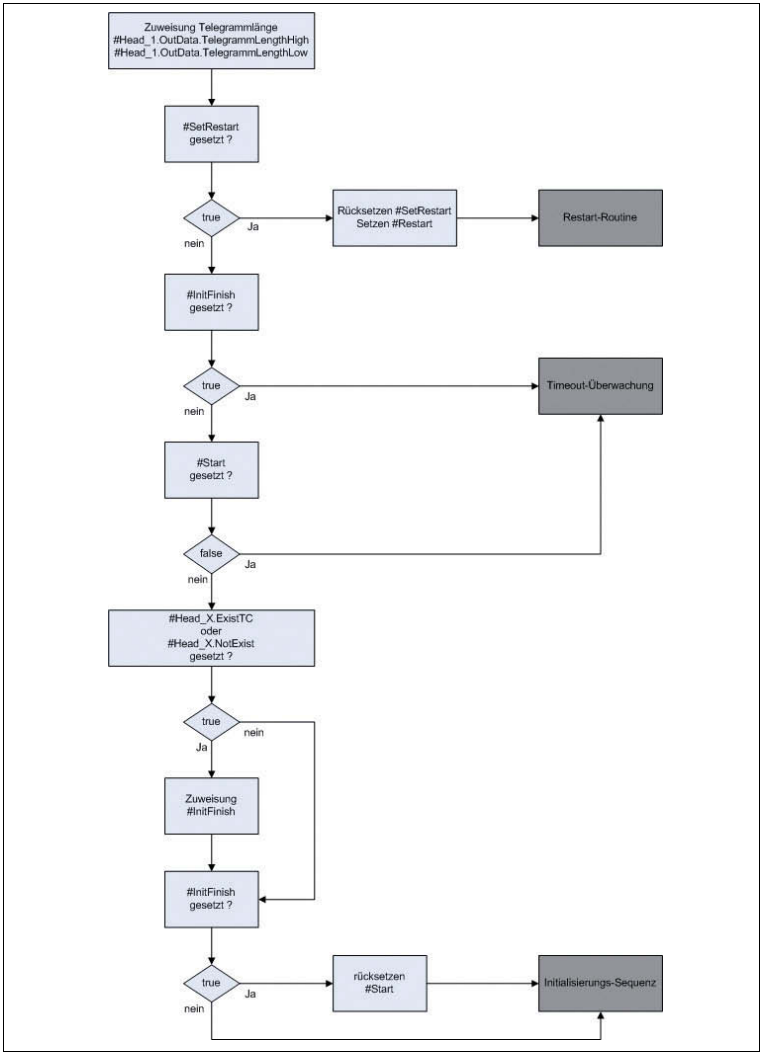


Abbildung 6.1: Ablauf der Start-UP-Sequence

Die Start-UP-Sequence im Netzwerk 1 hat die Aufgabe festzustellen, ob die Schreib-/Leseköpfe der IDENTControl initialisiert worden sind. Mit Hilfe der Initialisierung wird überprüft, an welchen Kanälen die Schreib-/Leseköpfe an die IDENTControl angeschlossen sind.

Zu Beginn des Programmteils wird die in der Hardwarekonfiguration festgelegte Telegrammlänge der Ausgangsdaten über die IN-Variable **#TCP_TelegrammLength** der Variablen **#Head_X.OutData**.TelegrammLengthLow des Ausgangsdatenfeldes übergeben.

Im Anschluss wird überprüft, ob das Bit #SetRestart gesetzt ist. Dadurch ist es möglich, einen Neustart des Programms durchzuführen. Ist die Bedingung für einen Neustart erfüllt, erfolgt ein Sprung zur Sprungmarke res in das Netzwerk 19.

Ist ein Neustart des Programms nicht vorgesehen, wird das Bit **#InitFinish** überprüft. Ist es gesetzt, so ist die Initialisierung aller Köpfe bereits abgeschlossen und es erfolgt ein Sprung zur Sprungmarke end in das Netzwerk 6 zur Timeoutüberwachung. Ebenso erfolgt ein Sprung zur Timeoutüberwachung, wenn das Bit **#Start** nicht gesetzt ist. Mit Hilfe des **#Start** Bits wird signalisiert, dass ein Neustart des Programms durchgeführt worden ist.

Im nachfolgenden Abschnitt erfolgt die Überprüfung, inwieweit die Initialisierung für alle Kanäle erfolgreich abgeschlossen ist. Dazu wird geprüft, ob das Bit **#Head_X.Exist** oder **#Head_X.NotExist** gesetzt ist. Sobald eines der beiden Bits gesetzt ist, wurde die Initialisierung für den Kanal erfolgreich durchgeführt. Unterscheiden Sie dabei, ob ein Schreib-/Lesekopf an den entsprechenden IDENT-Kanal angeschlossen (**#Head_X.Exist**) oder nicht angeschlossen (**#Head_X.NotExist**) ist. Ist die Überprüfung erfolgt für alle Köpfe erfolgreich abgeschlossen, wird das Bit **#InitFinish** gesetzt. Dieses Bit signalisiert die erfolgreiche Beendigung der Initialisierung für alle Köpfe. Abschließend erfolgt das Rücksetzen des **#Start** Bits.

Die Start-UP-Sequence ist damit beendet und es erfolgt nun die Initialisierung der einzelnen Schreib-/Leseköpfe, sofern dieses noch nicht in den vorhergehenden Zyklen durchgeführt wurde.

6.2 Ablauf der Initialisierung

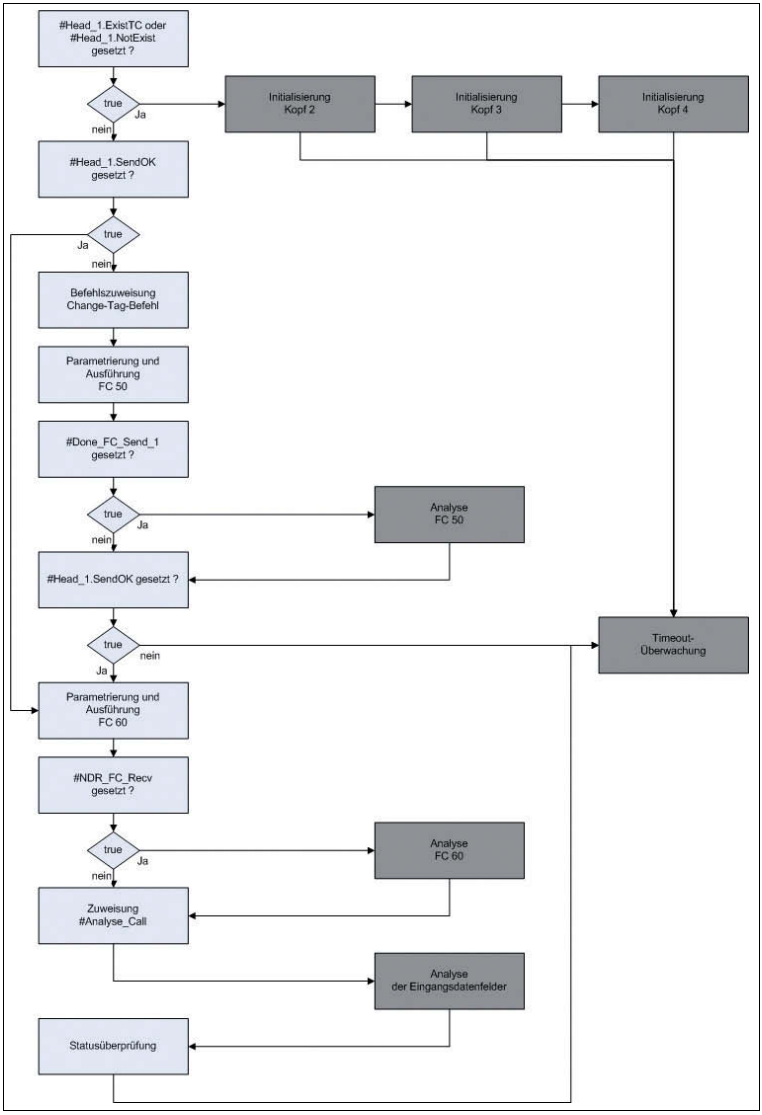


Abbildung 6.2: Ablauf der Initialisierung von Kopf 1

Der Programmabschnitt Initialisierung ist erforderlich um festzustellen, ob ein Schreib-/Lesekopf an einen IDENT-Kanal angeschlossen ist. Bei der Initialisierung wird ein Change-Tag-Befehl an den entsprechenden IDENT-Kanal der Auswerteeinheit geschickt. Mit Hilfe dieses Befehls wird dem Schreib-/Lesekopf mitgeteilt, welcher Code-/Datenträger beschrieben bzw. ausgelesen werden soll. Nachfolgend ist die Initialisierung des IDENT-Kanal 1 beschrieben. Für die anderen IDENT-Kanäle erfolgt die Initialisierung analog zu diesem Teil.

Zu Beginn des Programmteils wird überprüft, ob der IDENT-Kanal bereits initialisiert wurde. Dazu werden die Bits **#Head_1.Exist** und **#Head_1.NotExist** überprüft. Ist eines dieser beiden Statusbits gesetzt, so ist die Initialisierung für diesen Kanal bereits erfolgt und es wird zur Sprungmarke ini2 gesprungen. An dieser Sprungmarke beginnt die Initialisierung für den IDENT-Kanal 2.

Ist die Initialisierung für diesen Kopf noch nicht erfolgreich abgeschlossen, wird zunächst überprüft, ob das Bit **#Head_1.SendOK** gesetzt ist. Mit Hilfe dieses Bits wird an dieser Stelle signalisiert, dass der Change-Tag-Befehl bereits an die IDENTControl geschickt wurde. Damit der Befehl nicht erneut gesendet wird, erfolgt ein Sprung zur Sprungmarke rec1 im selben Netzwerk. Durch diesen Sprung wird das Laden der Befehlsparameter für den Change-Tag-Befehl in das Ausgangsdatenfeld sowie das erneute Senden dieses Befehls an die IDENTControl unterbunden.

Im Anschluss an die Abfrage der Statusbits werden die Befehlsparameter für den Change-Tag-Befehl in das Ausgangsdatenfeld der IDENTControl übergeben. Nach der Übergabe der Befehlsparameter werden verschiedene Statusbits gesetzt bzw. zurückgesetzt. Durch das Setzen von **#Head_1.Busy** wird gekennzeichnet, dass am Kanal 1 gerade ein Befehl bearbeitet wird. Mit **#Head_1.TimeoutActive** wird die Timeoutüberwachung zur Ausführung des in das Ausgangsdatenfeld geladenen Befehls vorbereitet.

Nachdem die Befehlsparameter in das Ausgangsdatenfeld übertragen und die Statusbits gesetzt wurden, erfolgt das Senden des Befehls an die IDENTControl. Das Senden der Daten an die IDENTControl über TCP/IP erfolgt durch die SIMATIC Funktion FC 50. Die für die Ausführung der Funktion erforderlichen Parameter werden mit Hilfe eines ANY-Parameters der Funktion zugeführt. Anschließend erfolgt der Aufruf der Funktion FC 50. Nach erfolgreicher Ausführung der Funktion wird das Bit **#Done_FC_Send_1** gesetzt und es erfolgt ein Sprung zur Sprungmarke aus1 in das Netzwerk 20. Dort erfolgt eine Fehlerauswertung für die Funktion.

Nachdem die Fehlerauswertung durchgeführt wurde und kein Fehler bei der Ausführung der Funktion aufgetreten ist, erfolgt der Rücksprung zur Sprungmarke F501 in das Netzwerk 2.

Befindet sich die Funktion noch in Bearbeitung, so ist das Bit **#Done_FC_Send_1** nicht gesetzt und es wird nicht zur Fehlerauswertung gesprungen. Stattdessen wird direkt an der Sprungmarke F501 das Programm weiter bearbeitet. An der Sprungmarke F501 wird überprüft, ob das Bit **#Head_1.SendOK** gesetzt ist. Dieses Bit ist gesetzt, wenn die Funktion FC 50 erfolgreich und ohne Fehler ausgeführt wurde. In diesem Fall erfolgt die Weiterverarbeitung des Programms an der Sprungmarke rec1. Ist das Bit **#Head_1.SendOK** nicht gesetzt, erfolgt ein Sprung zur Timeoutüberwachung in das Netzwerk 6.

Nachdem das Senden der Daten an die IDENTControl erfolgreich durchgeführt worden ist, erfolgt im Anschluss das Empfangen der Rückantwort von der IDENTControl. Das Empfangen von Daten über TCP/IP wird durch die SPS über die SIMATIC Funktion FC 60 durchgeführt.

Zu Beginn wird der Wert 0 der Variablen **#FC_Recv_Call** übergeben. Durch diese Variable erfolgt mit Hilfe einer Sprungauswahlliste ein Rücksprung an die Sprungmarke F601 in diesem Netzwerk. Die für die Ausführung der Funktion erforderlichen Parameter werden mit Hilfe eines ANY-Parameters der Funktion zugeführt. Anschließend erfolgt der Aufruf der Funktion FC 60. Nach erfolgreicher Ausführung der Funktion wird das Bit **#NDR_FC_Recv** gesetzt und es erfolgt ein Sprung zur Sprungmarke **aus5** in das Netzwerk 21. Dort findet eine Fehlerauswertung für die Funktion statt. Nachdem die Fehlerauswertung durchgeführt wurde und kein Fehler bei der Ausführung aufgetreten ist, erfolgt der Rücksprung zur Sprungmarke F601 in das Netzwerk 2.

Befindet sich die Funktion noch in Bearbeitung, ist das Bit **#NDR_FC_Recv** nicht gesetzt und es wird nicht zur Fehlerauswertung gesprungen. Stattdessen wird direkt an der Sprungmarke F601 das Programm weiter bearbeitet.

An der Sprungmarke F601 wird die Variable **#Head_1.TimeoutActiv** zurückgesetzt. Anschließend wird der Variablen **#Analyse_Call** der Wert 0 zugewiesen. Durch diese Variable erfolgt mit Hilfe einer Sprungauswahlliste der Rücksprung an die Sprungmarke ana1 in diesem Netzwerk. Danach erfolgt ein Sprung zur Sprungmarke lyse in das Netzwerk 22. Dort werden die von der IDENTControl empfangenen Daten analysiert. Nach Abschluss der Analyse findet der Rücksprung über eine Sprungauswahlliste in das Netzwerk 2 zur Sprungmarke ana1 statt.

In diesem Programmabschnitt erfolgt die Auswertung der Statusrückmeldung. Die Antwort der IDENTControl auf einen Befehl enthält eine Statusmeldung. Durch deren Auswertung kann man Rückschlüsse auf die Ausführung des Befehls ziehen. Die Statusrückmeldung ist ein ein Byte großer Code, welcher in der Variablen **#Head_1.InData.Status** enthalten ist. Hat der Status den Wert (06)h, so ist kein Schreib-/Lesekopf an den IDENT-Kanal 1 angeschlossen und somit wird das Bit **#Head_1.NotExist** gesetzt. Auf Grund dessen, dass kein Schreib-/Lesekopf angeschlossen ist, werden die Statusbits **#Head_1.ExistTC** und **#Head_1.Error** zurückgesetzt. Anschließend wird überprüft, ob das Statusbit **#Head_1.Done** gesetzt ist. Durch dieses Bit wird signalisiert, dass neue Daten zur Analyse am Kopf 1 bereitstehen. Ebenso muss die Überprüfung erfolgen, dass das Bit **#Head_1.NotExist** nicht gesetzt ist. Sind beide Bedingungen erfüllt, ist ein Schreib-/Lesekopf an den IDENTKanal 1 angeschlossen und es wird das

Statusbit **#Head_1.ExistTC** gesetzt. Abschließend erfolgt ein absoluter Sprung zur Sprungmarke end in das Netzwerk 6 zur Timeoutüberwachung. Nach Bearbeitung dieses Programmabschnitts ist die Initialisierung für den Kopf 1 abgeschlossen und es erfolgt im nächsten Zyklus die Initialisierung für Kopf 2, sofern sie noch nicht durchgeführt wurde.

6.3 Ablauf der Timeoutüberwachung

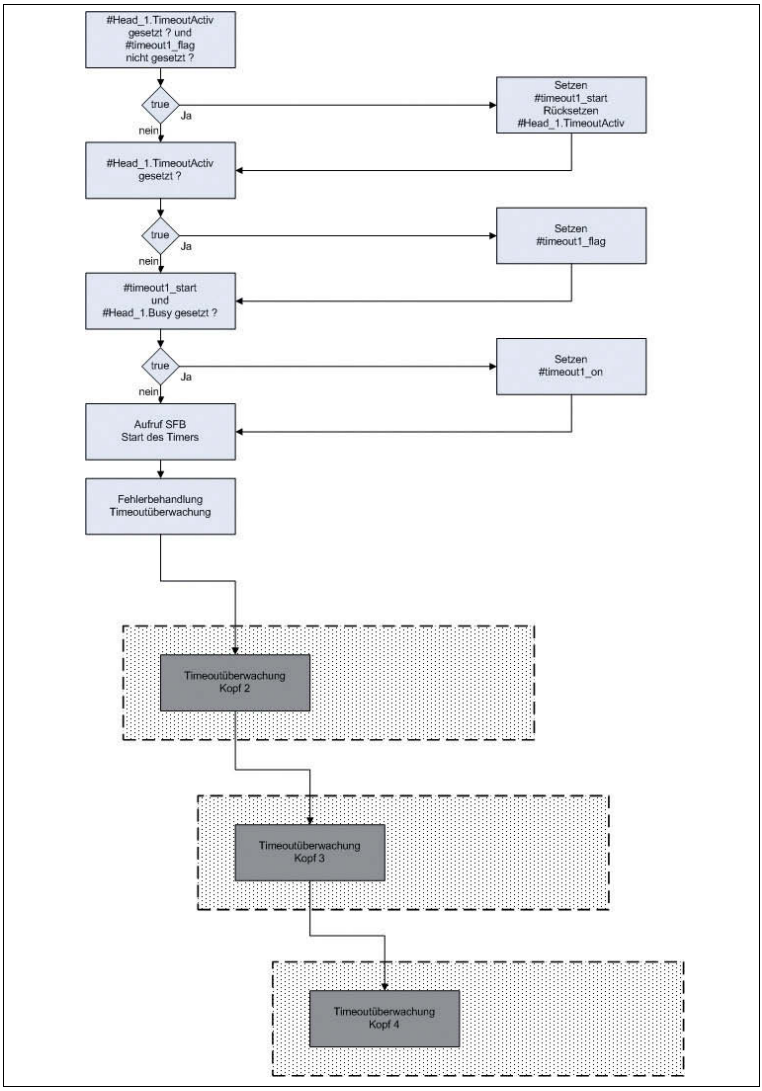


Abbildung 6.3: Ablaufplan Timeoutüberwachung

2011-01

Die Timeoutüberwachung hat die Aufgabe, die maximale Antwortzeit der IDENTControl auf einen Befehl zu überwachen und bei einer zeitlichen Überschreitung eine Fehlermeldung auszugeben. Nachfolgend wird die Timeoutüberwachung für Kopf 1 erläutert. Sie gilt analog auch für die anderen Köpfe.

Zu Beginn wird überprüft, ob das Bit **#Head_1.TimeoutActiv** gesetzt und das Bit **#timeout1_ flag** nicht gesetzt ist. Das Verknüpfungsergebnis dieser Abfrage wird der Variablen **#timeout1_start** zugewiesen. Ist die Bedingung erfüllt, so wird die Variable **#Head_1.TimeoutActiv** zurückgesetzt. Andernfalls wird überprüft, ob nur die Variable **#Head_1.TimeoutActiv** gesetzt ist. Das Verknüpfungsergebnis wird der Variablen **#timeout1_flag** zugewiesen. Anschließend wird überprüft ob die Variablen #timeout1_ start und #Head_1.Busy gesetzt sind. Das Verknüpfungsergebnis wird der Variablen **#timeout1_on** zugewiesen.

Im Anschluss erfolgt der Aufruf des SFB 5. Dieser erzeugt eine Ausschaltverzögerung am Ausgang Q. Die Variable bewirkt eine steigende Flanke am Eingang IN, was zu einer steigende Flanke am Ausgang Q bei der Variablen **#timeout1_running** führt. Liegt eine fallende Flanke am Eingang Q an, so entsteht eine fallende Flanke am Ausgang Q erst nach Ablauf der Zeit PT **#Timeout**.

Im Anschluss wird überprüft, ob das Bit **#Head_1.Busy** gesetzt sowie die Bits **#Head_1.ReceivedOK**, **#timeout1_running** und **#timeout1_start** nicht gesetzt sind. Ist diese Bedingung erfüllt, ist die Überwachungszeit überschritten und es erfolgt Fehlermeldung. Dabei werden die Bits **#Head_1.TimeoutOccured**, **#Head_1.Error** sowie **#Head_1.Done** gesetzt. Des Weiteren werden die Bits **#Head_1.SglCommandActiv**, **#TransfToHead1**, **#Head_1.Busy** sowie **#Head_1.TimeoutActiv** zurückgesetzt.

Dieses Netzwerk bewirkt eine Ausschaltverzögerung. Die Ausschaltbedingung wird realisiert, wenn **#Head_1.TimeoutActiv** und **#timeout1_flag** nicht gesetzt sind. Des Weiteren muss das Bit **#Head_1.Busy** gesetzt sein. Beide **Head_1** Bits werden beim Aufruf des SFC15 "Schreiben an Slave“ gesetzt. Durch die Variable **#timeout1_flag** wird signalisiert, dass ein Timeout noch aktiv ist und nicht verlängert werden kann. Die Timeoutüberwachung für die anderen Köpfe erfolgt analog. An die Timeoutüberwachung schließt sich die Variabelenumsetzung von IN- auf STAT-Variablen an.

6.4 Ablauf der Variablenumsetzung von IN- auf STAT-Variablen

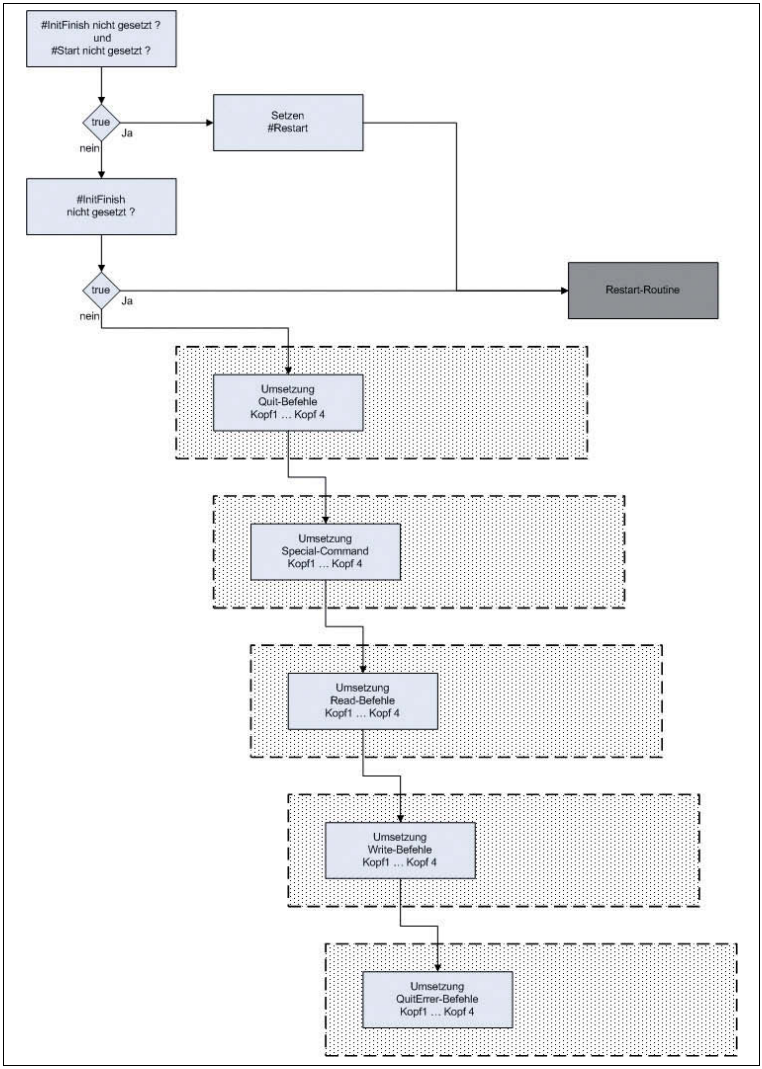


Abbildung 6.4: Ablaufplan Variablenumsetzung

Dieser Abschnitt hat die Aufgabe, die Eingangsvariablen auf statische Variablen umzusetzen. Die Umsetzung ist erforderlich, weil sich während der Bearbeitung des Funktionsbausteins der Zustand der Eingangsvariablen ändern kann. Dadurch kann ein Befehl angestoßen werden, obwohl noch ein anderer Befehl in Bearbeitung ist.

2011-01

Die IN-Variablen haben den Nachteil, dass sie durch den Funktionsbaustein nicht verändert werden können. Dadurch würde der Befehl, welcher durch eine IN-Variable angestoßen wird, solange ausgeführt, bis die IN-Variable durch einen Programmteil außerhalb des Funktionsbausteins zurückgesetzt wird. Eine Flankensteuerung verschafft an dieser Stelle eine Abhilfe.

Bei einer positiven Flanke im Signalverlauf einer IN-Variablen erfolgt die Umsetzung auf eine entsprechende STAT-Variable. Die STAT-Variable ist eine Lokalvariable innerhalb des Funktionsbausteins. Die STAT-Variablen haben den Vorteil, dass sie durch den Funktionsbaustein verändert werden können. Nach erfolgter Befehlsausführung wird die STAT-Variable zurückgesetzt und es kann wieder ein neuer Befehl angestoßen werden.

Zu Beginn des Netzwerks wird überprüft, dass die Variablen **#InitFinish** und **#Start** nicht gesetzt sind. Die Bedingung ist erfüllt, wenn die Initialisierung der Köpfe noch nicht abgeschlossen ist und kein Restart erfolgte. Somit wird das Bit **#Restart** gesetzt und es erfolgt ein Sprung zur Sprungmarke res in das Netzwerk 17. Dort wird die Restart-Routine durchgeführt.

Andernfalls erfolgt die alleinige Abfrage des Bits **#InitFinish**. Der Anwender kann durch das Rücksetzen dieses Bits die automatische Neuinitialisierung des Funktionsbausteins einleiten. Wurde die Initialisierung noch nicht durchgeführt, erfolgt ein Sprung zur Restart-Routinen, an welche sich die Initialisierung bei einem neuerlichen Programmdurchlauf anschließt.

Wurde die Initialisierung zuvor bereits erfolgreich abgeschlossen, erfolgt nun die eigentliche Befehlsumsetzung.

Anschließend werden die extern gegebenen Quit-Befehle auf interne Signale umgewandelt. Es wird der vorige Signalzustand in der Variablen **#SaveQuitHead1** gespeichert. Diese Variable wird in jedem Zyklus mit der Verknüpfungsvariablen **#Head1Quit** verglichen. Geht die Variable **#Head1Quit** auf "1“ und war der Vorzustand "0“ (in **#SaveQuitHead1** gespeichert), so wird ein positiver Flankenwechsel registriert und das Bit **#QuitHead1** gesetzt. Die Umsetzung der anderen Befehle erfolgt nach dem gleichen Muster.

6.5 Ablauf Programmteil Einlesen der Datei

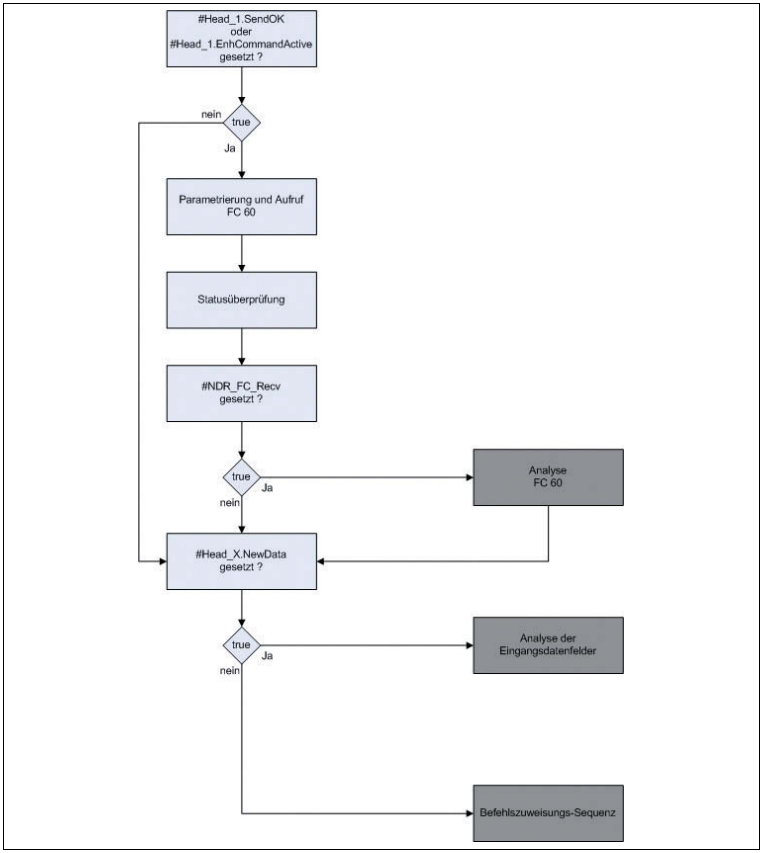


Abbildung 6.5: Ablaufplan Einlesen von Daten

In diesem Netzwerk erfolgt das Einlesen der Daten durch die Funktion FC 60. Nach einer Überprüfung, ob Daten eingelesen werden können, erfolgt die Ausführung der Funktion. Nach dem Einlesen werden die Daten analysiert.

Zu Beginn des Netzwerkes wird überprüft, ob das Senden eines Befehls an die IDENTControl erfolgreich durchgeführt werden konnte. Sobald ein Befehl an die IDENTControl übersandt wurde, ist es möglich die Rückantwort auf diesen Befehl einzulesen. Es erfolgt ebenso die Prüfung, ob ein Enhanced-Befehl an einen der Köpfe aktiv ist. Ist dies der Fall, können ständig Daten von der IDENTControl eingelesen werden. Das Einlesen von Daten wird nicht durchgeführt, wenn weder ein Enhanced- Befehl an den Köpfen aktiv ist, noch das Senden eines Befehles an einem Kopf erfolgreich durchgeführt wurde. Somit erfolgt schließlich ein Sprung zur Sprungmarke F605 an das Ende dieses Netzwerkes.

2011-01

Nachdem die Überprüfung, ob neue Daten eingelesen werden müssen, erfolgreich abgeschlossen ist, erfolgt nun die Ausführung des FC 60.

Zu Beginn wird die Variable **#FC_Recv_Call** mit den Wert 4 belegt. Dadurch wird mit Hilfe einer Sprungleiste ein Rücksprung aus der Auswertung in diesen Programmteil realisiert. Anschließend erfolgt das Laden des ANY-Parameters. Die eingelesenen Daten werden im Memory-Datenfeld abgespeichert. Das Datenfeld beginnt an der Adresse 642.0 und muss bei der Parametrierung des ANY-Parameters berücksichtigt werden. Eine vollständige Beschreibung des ANY-Parameters ist in einem separaten Abschnitt enthalten. Nach den Laden des ANY-Parameters erfolgt der eigentliche Aufruf der Funktion FC 60.

Die Variable **#Memory_RetVal_FC_Recv** enthält eine Statusmeldung. Zunächst erfolgt die Überprüfung auf eine Verbindungsunterbrechung. Sobald dieser Fehler auftritt, enthält die Variable **#Memory_RetVal_FC_Recv** den Fehlercode (8304)h und die Variable **#DynErrorTCPConnection** wird gesetzt.

Anschließend erfolgt die Überprüfung auf den Wert 0. Die Variable hat den Wert 0, wenn die Daten durch die Funktion eingelesen wurden. Somit ist kein Fehler bei der Ausführung aufgetreten und es wird die Variable **#DynErrorTCPConnection** zurückgesetzt.

Bei der Ausführung der Funktion können keine Daten an der Quelladresse vorhanden sein. Dies wird durch den Fehlercode (8180)h signalisiert. Es können zwar keine neuen Daten eingelesen werden, aber die Kommunikation zwischen SPS und IDENTControl ist vorhanden. Deshalb erfolgt ein Rücksetzen des Bits **#DynErrorTCPConnection**.

Das Ende der Bearbeitung der Funktion FC 60 wird durch das Bit **#NDR_FC_Recv** signalisiert. Das Bit ist gesetzt, wenn die Bearbeitung der Funktion abgeschlossenist. Daraufhin erfolgt ein Sprung zur Sprungmarke aus 5 in das Netzwerk 21. Dort erfolgt die Analyse der durch die FC 60 eingelesenen Daten. Nach erfolgreich durchgeführter Analyse findet ein Rücksprung mit Hilfe einer Sprungleiste zur Sprungmarke F605 statt.

Ist die Funktion FC 60 noch in Bearbeitung, so ist das Bit **#NDR_FC_Recv** nicht gesetzt. Dadurch sind noch keine Daten eingelesen, wodurch ein Sprung zur Analyse überflüssig ist. Die Programmbearbeitung erfolgt sogleich an der Sprungmarke F605. An der Sprungmarke F605 wird überprüft, ob an einem der Köpfe neue Daten zur Auswertung bereitstehen. Sobald neue Daten an Kopf X zur Auswertung bereitstehen, ist das Bit **#Head_X.NewData** gesetzt. Das Bit wird bei der Analyse des FC 60 gesetzt, wenn das Umkopieren des Memory-Datenfeldes in das Eingangsdatenfeld des entsprechenden Kopfes erfolgreich durchgeführt wurde. Die Überprüfung der Köpfe erfolgt durch eine Oder-Verknüpfung. Wenn an mindestens einem Kopf neue Daten vorhanden sind, erfolgt ein Sprung zur Sprungmarke lyse in das Netzwerk 22 zur Analyse der kopfeigenen Eingangsdatenfelder.

Eine Analyse ist erforderlich, wenn neue Daten in den Eingangdatenfeldern der Köpfe vorliegen. Für die Realisierung des Rücksprunges aus der Analyse, wird der Variablen **#Analyse_Call** der Wert 4 zugewiesen. Somit kann über eine Sprungleiste nach durchgeführter Analyse aus der Analysesequenz in Netzwerk 22 zur Befehlsbearbeitung an die Sprungmarke ana5 in das Netzwerk 9 gesprungen werden.

Wenn bei der Abfrage keines der Bits **#Head_X.NewData** gesetzt ist, befinden sich keine neuen Daten in den Eingangsdatenfeldern der Schreib-/Leseköpfe. Eine Analyse der Eingangsdatenfelder ist nicht erforderlich. Deshalb erfolgt kein Sprung zur Sprungmarke lyse in das Netzwerk 22, sondern das Programm wird an der Sprungmarke ana5 im Netzwerk 9 fortgesetzt. Hier erfolgt die Befehlszuweisung für die verschiedenen Köpfe.

6.6 Ablauf der Befehlszuweisungssequenz von Kopf 1

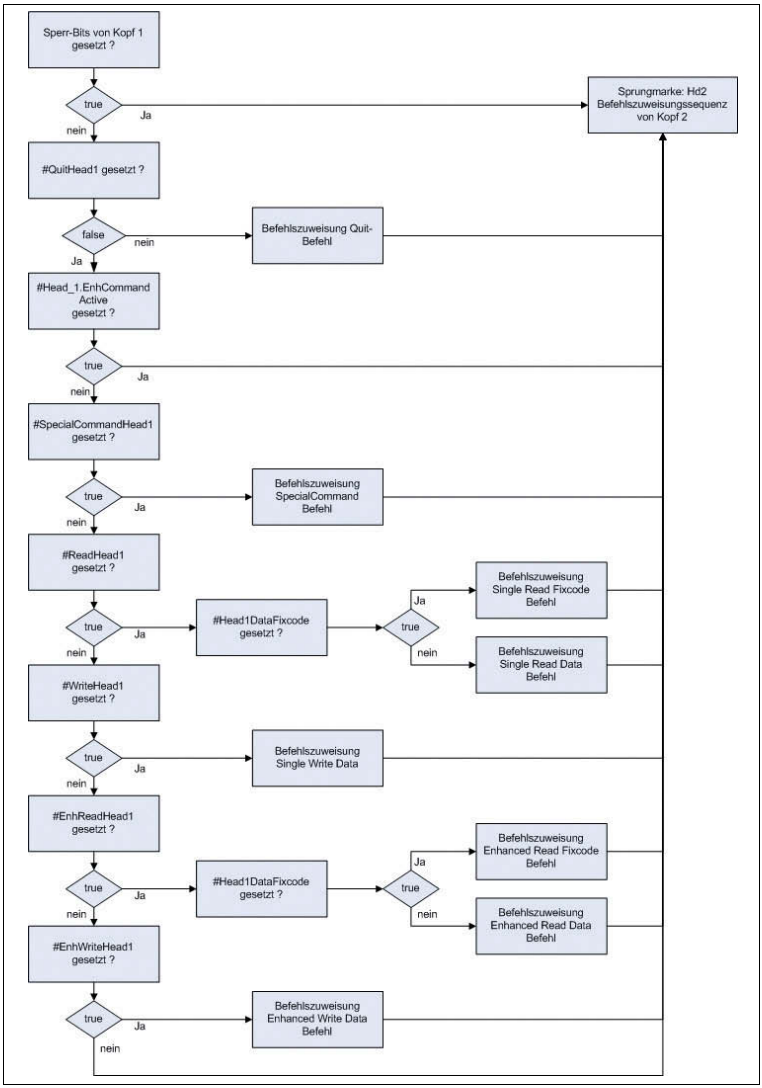


Abbildung 6.6: Ablaufplan der Befehlszuweisungssequenz

In diesem Abschnitt wird die Befehlszuweisung beispielhaft für Kopf 1 erläutert. Die Befehlszuweisungen der anderen Köpfe verhalten sich analog zu der von Kopf 1. Es wird an dieser Stelle darauf verzichtet, näher auf die Befehlszuweisungssequenzen der Köpfe 2, 3 und 4 einzugehen.

Zu Beginn der Befehlszuweisung wird überprüft, ob der Kopf für eine Befehlsausführung blockiert ist. Es gibt verschiedene Möglichkeiten, die zur Blockierung der Befehlszuweisung führen. Ist bei vorangegangenen Befehlsausführungen an Kopf 1 ein Fehler aufgetreten, ist dieser Kopf für weitere Befehle gesperrt, bis er durch Setzen des Bits **#Head_1.QuittError** wieder in den Grundzustand versetzt wird. Eine Befehlszuweisung wird ebenfalls unterbunden, wenn kein Schreib-/Lesekopf am entsprechenden IDENT-Kanal angeschlossen ist. Die Befehlszuweisung ist aber ebenfalls blockiert, wenn sich schon Daten in dem Ausgangsdatenfeld befinden. Diese Daten stammen aus einer vorhergehenden Befehlszuweisung und wurden noch nicht an den entsprechenden Kopf versandt. Damit die Ausführung des in diesen Daten enthaltenen Befehls durchgeführt werden kann, ist die Befehlszuweisung für das entsprechende Ausgangsdatenfeld gesperrt bis der Befehl ausgeführt wurde. Die Abfrage wird mit einer Oder-Bedingung realisiert. Sobald eine der Bedingungen erfüllt ist, wird zur Befehlszuweisung von Kopf 2 an die Sprungmarke Hd2 gesprungen.

Im Anschluss an die Überprüfung auf eine Blockierung erfolgt die Zuweisung der Befehlsparameter in das Ausgangsdatenfeld. Dazu muss vorher ermittelt werden, welcher Befehl an Kopf 1 ausgeführt werden soll.

Als erstes wird überprüft, ob ein Quit-Befehl an Kopf 1 ausgeführt werden soll. Dazu wird das Bit **#QuitHead1** abgefragt. Wenn kein Quit-Befehl gesetzt ist, so ist das Bit nicht gesetzt. Es erfolgt ein Sprung zur Sprungmarke SCH1. An dieser Sprungmarke erfolgt die Abfrage, ob ein Enhanced-Befehl aktiv ist.

Wenn ein Quit-Befehl ausgeführt werden soll, so ist das Bit **#QuitHead1** gesetzt und es müssen nun die Befehlsparameter in das Ausgangsdatenfeld für Kopf 1 geladen werden. Nach der Zuweisung der Befehlsparameter in das Ausgangsdatenfeld werden verschiedene Sperrbits gesetzt. Durch das Bit **#TransfToHeadX** wird signalisiert, dass an Kopf X Befehlsparameter in das Ausgangsdatenfeld geladen wurden. Das Senden des Befehles an die IDENTControl hat aber noch nicht stattgefunden. Damit die Befehlsparameter des Quit-Befehls nach der ersten Befehlsausführung nicht nochmalig in das Ausgangsdatenfeld übergeben werden, setzt man das Bit **#QuitHead1** zurück. Dadurch können andere Befehle an Kopf 1 wieder gestartet werden, sobald der Quit-Befehl ausgeführt wurde. Mit Hilfe des Rücksetzens der Bits **#Head_1.EnhCommandActive** und **#Head_1.Busy** wird deutlich, dass der Quit-Befehl nur einmalig und nicht dauerhaft ausgeführt wird. Im Anschluss an die Parameterzuweisung für den Quit-Befehl in das Ausgangsdatenfeld erfolgt ein Sprung zur Befehlszuweisung von Kopf 2. Der Sprung zu Kopf 2 erfolgt, weil weitere Befehlszuweisungen in das Ausgangsdatenfeld nicht mehr möglich sind, bis der Befehl vollständig abgearbeitet wurde.

Wenn an Kopf 1 kein Quit-Befehl ausgeführt werden soll, so erfolgt ein Sprung zur Sprungmarke SCH1. Hier wird zunächst überprüft, ob ein Enhanced-Befehl an Kopf 1 ausgeführt wird. Sobald ein Enhanced-Befehl aktiv ist, wird der Kopf für die Ausführung weiterer Befehle gesperrt. Bei der Steuerung des Programms ist es während der Ausführung eines Enhanced-Befehles nicht möglich, einen anderen Befehl an Kopf 1 zu starten. Deshalb werden alle Startbit für die anderen Befehle zurückgesetzt. Eine Ausnahme bildet hier der Quit-Befehl. Dieser Befehl kann an

2011-01

dieser Stelle trotz aktiven Enhanced-Befehls zugewiesen und ausgeführt werden, weil er dadurch zum Abbruch des laufenden Enhanced-Befehles führt. Wenn ein Enhanced-Befehl aktiv ist, erfolgt im Anschluss an das Rücksetzen der Startbits ein Sprung zur Sprungmarke Hd2 zur Befehlszuweisung für Kopf 2. Danach wird überprüft, welcher Befehl an Kopf 1durchgeführt werden soll.

Wenn die Ausführung eines Special-Commands an Kopf 1 beabsichtigt ist, wird dies durch das gesetzte Bit **#SpecialCommandHead1** signalisiert. Daraufhin wird an die Sprungmarke SH1 gesprungen. An der Sprungmarke SH1 werden die durch den Anwender festgelegten Befehlsparameter in das Ausgangsdatenfeld von Kopf 1 kopiert.

Soll an Kopf 1 ein Read-Befehl gestartet werden, ist das Bit **#ReadHead1** gesetzt und es wird zur Sprungmarke SRH1 gesprungen. An der Sprungmarke werden die Befehlsparameter zur Ausführung eines Read-Befehls an Kopf 1 in das zugehörige Ausgangsdatenfeld kopiert.

Bei einer beabsichtigten Ausführung eines Write-Befehls an Kopf 1 ist das Bit **#WriteHead1** gesetzt. In diesem Fall erfolgt ein Sprung zur Sprungmarke SWH1. Hier wird analog zu den anderen Befehlen die Parameterübergabe in das Ausgangsdatenfeld für einen Single-Write-Befehl durchgeführt.

Bei der Ausführung eines Enhanced-Befehls ist zwischen einem Enhanced-Read und einem Enhanced-Write-Befehl zu unterscheiden. Zur Ausführung eines Enhanced- Read-Befehls wird das Bit **#EnhReadHead1** überprüft und es erfolgt ein Sprung zur Sprungmarke ERH1, sofern das Bit gesetzt ist. Andernfalls wird überprüft, ob ein Enhanced-Write-Befehl ausgeführt werden soll. Ist die Ausführung dieses Befehls beabsichtigt, so ist das Bit **#EnhWriteHead1** gesetzt und es erfolgt ein Sprung zur Sprungmarke EWH1. Durch diese gestaffelte Abfrage werden alle möglichen, durch den Anwender startbaren, Befehle überprüft und in die entsprechende Befehlsparameterzuweisung gesprungen. Es besteht aber auch die Möglichkeit, keinen Befehl an Kopf 1 auszuführen. Dann sind alle der zu überprüfenden Bits nicht gesetzt und es erfolgt ein Sprung zu der Befehlszuweisung von Kopf 2 an die Sprungmarke Hd2.

6.7 Ablauf der Befehlsausführung für Kopf 1

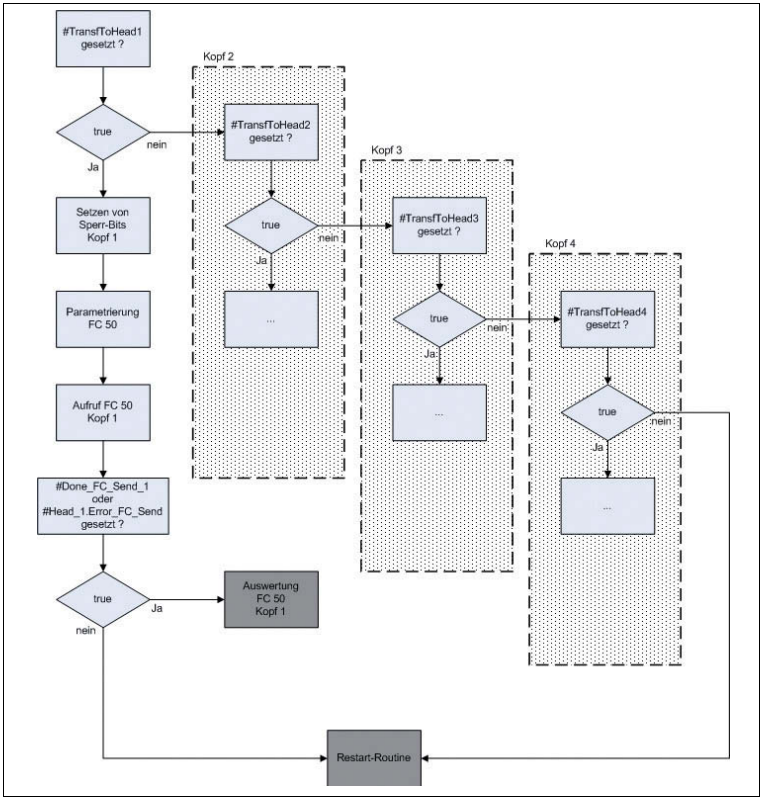


Abbildung 6.7: Ablauf der Befehlsausführung für Kopf 1

In diesem Abschnitt wird beispielhaft der Ablauf der Befehlsausführung für Kopf 1 erläutert. Die Bearbeitung der Befehlsausführung für die Köpfe 2, 3 und 4 verhält sich analog. Deshalb wird an dieser Stelle darauf verzichtet, näher auf die Befehlsausführung dieser Köpfe einzugehen.

Unter der Befehlsausführung von Kopf 1 versteht man das Senden der in dem zugehörigen Ausgangsdatenfeld abgelegten Befehlsparameter an die IDENTControl. Die Zuweisung der Befehlsparameter in die zugehörigen Ausgangsdatenfelder wurde im vorhergehenden Abschnitt erläutert. Das Senden der Ausgangsdatenfelder wird durch die Funktion FC 50 durchgeführt. Nach erfolgreicher Übermittelung des Ausgangsdatenfeldes wird der Befehl über eine Statusrückmeldung von der IDENTControl quittiert und ausgeführt.

Am Anfang der Befehlsausführung wird zunächst geprüft, ob die Ausgangsdatenfelder der einzelnen Köpfe mit neuen Befehlsparametern belegt sind. Wenn ein Ausgangsdatenfeld mit Parametern belegt ist, die noch nicht an die IDENTControl übermittelt wurden, ist das Bit **#TransfToHeadX** gesetzt. Deshalb erfolgt zunächst eine Abfrage, welche dieser Bits gesetzt sind. Ist das Bit **#TransfToHead1** für Kopf 1 gesetzt, erfolgt ein Sprung zur Sprungmarke exe1. Hier erfolgt die Befehlsausführung für Kopf 1. Ansonsten wird das Bit **#TransfToHead2** geprüft. Das gleiche gilt für die Abfrage der übrigen **#TransfToHeadX** Bit. Sollte keines der Bits gesetzt sein, wird zur Sprungmarke res gesprungen. Dies ist der Fall, wenn kein neuer Befehl an einen der Köpfe gestartet werden soll. An der Sprungmarke res wird eine Restart-Routine durchgeführt.

An der Sprungmarke exe1 werden die für Kopf 1 vorgesehenen Befehle an die IDENTControl gesendet. Zunächst werden verschiedene Sperrbits gesetzt. Durch das Bit **#Head_1.Busy** wird signalisiert, dass im Moment ein Befehl an Kopf 1 gesendet und bearbeitet wird. Mit Hilfe des Bit **#Head_1.TimeoutActiv** wird die Timeoutüberwachung der Befehlsübermittelung initialisiert. Die Variable **#Head1_exe** dient zur Unterscheidung des Aufrufes der Funktion FC 50. In dem Funktionsbaustein sind zwei mögliche Stellen vorhanden, an denen die Funktion für Kopf 1 aufgerufen wird. Dies ist zum einen die Initialisierung des ersten Kopfes und die andere Stelle ist die Befehlsausführung von Kopf 1. Beim Aufruf der Funktion in der Initialisierungsphase wird das Bit **#Head1_exe** zurückgesetzt. Hingegen wird das Bit gesetzt, wenn der Aufruf bei der Befehlsausführung erfolgt. Somit kann abhängig vom Ort des Funktionsaufrufes ein zielgerichteter Sprung realisiert werden. Des Weiteren werden verschiedene Statusbits an der Sprungmarke exe1 zurückgesetzt. Dadurch wird das Senden des Ausgangsdatenfeldes für Kopf 1 ermöglicht.

Anschließend erfolgt die Parametrierung der Funktion FC 50. Die Parametrierung wird über einen ANY-Parametertyp realisiert. Bei der Parametrierung wird das Quelldatenfeld festgelegt. Als Quelldatenfeld für den Aufruf der Funktion FC 50 dient das Ausgangsdatenfeld für Kopf 1 **#Head_1.OutData**. Das Datenfeld beginnt im Instanz- Datenbaustein ab der Adresse 92.0. Die Zuweisung des richtigen Datenfeldes ist wichtig, weil sonst nicht das Ausgangsdatenfeld mit den vorher zugewiesenen Befehlsparametern an die IDENTControl gesendet wird.

An die Parametrierung schließt sich der Aufruf der Funktion FC 50 an. Nach dem Funktionsaufruf wird überprüft, ob das Senden der Daten abgeschlossen ist. Dazu werden die Bits **#Done_FC_Send_1** und **#Head_1.Error_FC_Send** abgefragt. Wenn das Bit **#Done_FC_Send_1** gesetzt ist, so konnten die Daten erfolgreich an die IDENTControl übermittelt werden. Wenn bei der Ausführung der Funktion ein Fehler aufgetreten ist, so ist das Bit **#Head_1.Error_FC_Send** gesetzt. Wurde eines der beiden Bits gesetzt, ist die Bearbeitung der Funktion abgeschlossen und es wird zur Sprungmarke aus1 gesprungen. Dort erfolgt eine Auswertung des aufgetretenen Fehlers oder gegebenenfalls ein Sprung zur Routine zum Einlesen der Rückmeldung der IDENTControl auf den gerade versendeten Befehl. Sollte keines der beiden Statusbits gesetzt sein, ist die Funktion noch in Bearbeitung, d.h. die Ausgangsdatenfelder werden noch an die IDENTControl gesendet. In diesen Fall ist eine Auswertung nicht erforderlich.

Stattdessen wird zur Restart-Routine gesprungen. Dort erfolgt eine Überprüfung, ob ein Restart erfolgen soll und der Funktionsbaustein wird beendet. Im nächsten Durchlauf des Funktionsbausteins wird nochmals die Ausführung der Funktion geprüft und an die entsprechende Stelle zur Weiterverarbeitung innerhalb des Bausteins gesprungen.

Somit ist die Befehlsausführung für Kopf 1 beendet. Nach erfolgreichem Abschluss der Befehlsausführung und der Analyse der Rückmeldung erfolgt die Befehlsausführung für Kopf 2.

6.8 Ablauf der Restart-Routine

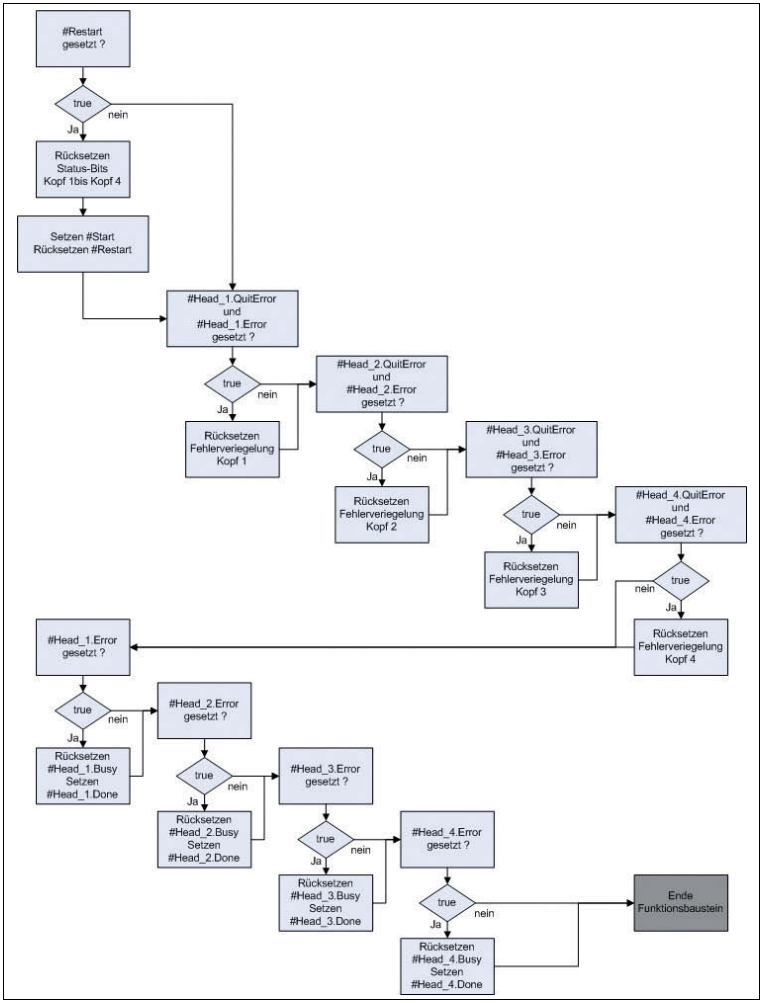


Abbildung 6.8: Ablauf der Restart-Routine

2011-01

Die Restart-Routine hat die Aufgabe den Funktionsbaustein in einen definierten Grundzustand zurückzusetzen und zu beenden. Nachfolgend ist die Restart-Routine beispielhaft für den Kopf 1 beschrieben. Die Köpfe 2, 3 und 4 verhalten sich analog zu diesem Beispiel.

Zu Beginn der Restart-Routine wird überprüft, ob das Bit **#Restart** gesetzt ist. Das Bit ist gesetzt, wenn der Funktionsbaustein einen Restart vornehmen soll. Sollte das Bit aber nicht gesetzt sein, erfolgt ein Sprung zur Sprungmarke end2. Die Sprungmarke end2 befindet sich am Anfang des Netzwerkes 19. Dort erfolgt das Rücksetzen der Bits, die einen Fehlerzustand melden.

Wenn das Bit **#Restart** gesetzt ist, werden die für den Kopf 1 erforderlichen Zustandsbits zurückgesetzt.

Das Rücksetzen der Zustandsbits für die Köpfe 2, 3 und 4 schließt sich direkt an den beispielhaft gezeigten Ausschnitt der Restart-Routine an. Nach dem Rücksetzen der Zustandsbits für alle Köpfe wird das Bit **#Start** gesetzt. Durch dieses Bit wird im nächsten Durchlauf des Funktionsbausteins die Initialisierung aller Köpfe innerhalb der Start-Up-Sequence in Netzwerk 1 überprüft. Durch das Rücksetzen des Bits **#Restart** wird signalisiert, dass der Funktionsbaustein in einen definierten Grundzustand zurückversetzt wurde.

Ein Bestandteil der Restart-Routine ist die Quittierung von Fehlermeldebits. Dieser Pogrammteil wird durchlaufen, wenn zu Beginn der Restart-Routine das Bit #Restart nicht gesetzt ist. Dann wird direkt zu der Quittierung der Fehlermeldungen gesprungen. Die Quittierung wird ebenfalls durchlaufen, wenn ein Restart erfolgt ist. Sobald bei der Ausführung des Funktionsbausteins an einem Kopf ein Fehler aufgetreten ist, wird das entsprechende Fehlermeldebit sowie das Bit **#Head_X.Error** gesetzt. Dadurch wird die Ausführung weiterer Befehle an diesem Kopf gesperrt. Der Funktionsbaustein bietet dem Anwender nun die Möglichkeit die Fehlerverriegelung des entsprechenden Kopfes aufzuheben. Dazu muss die IN-Variable **#QuitErrorHeadX** durch den Anwender gesetzt werden. In der Variablenumsetzung wird diese INVariable auf die STAT-Variable **#Head_X.QuitError** umgesetzt.

Am Anfang der Quittierungssequenz wird zunächst überprüft, ob die Bits **#Head_1.QuitError** und **#Head_1.Error** gesetzt sind. Ist die Bedingung erfüllt, so ist bei der Bearbeitung des Funktionsbausteins ein Fehler an Kopf 1 aufgetreten und der Anwender möchte die Fehlerverriegelung aufheben. Daraufhin werden die entsprechenden Fehlermelde- und Statusbits zurückgesetzt.

Die Quittierung der Fehlerverriegelung für die übrigen Köpfe schließt sich direkt an die beispielhaft gezeigte Quittierungssequenz an und erfolgt analog. An dieser Stelle wird darauf verzichtet, näher auf die Quittierungssequenzen der übrigen Köpfe einzugehen.

Nach der Quittierungssequenz erfolgt ein Programmteil zur Bearbeitung von Fehlermeldungen. Die Quittierungssequenz wird nur durchlaufen, wenn der Anwender die Fehlerverriegelung des entsprechenden Kopfes aufheben will. Wenn die Fehlerverriegelung nicht aufgehoben wurde, müssen Statusbits gesetzt werden. Diese Statusbits haben die Aufgabe zu signalisieren, dass der Befehl wegen eines Fehlers beendet wurde.

Zu Beginn wird überprüft, ob ein Fehler aufgetreten ist. Dazu wird das Bit **#Head_1.Error** geprüft. Wenn das Bit gesetzt ist, wird das Bit **#Head_1.Busy** zurückgesetzt. Dieses Bit signalisiert die laufende Ausführung eines Befehls. Anschließend wird das Bit **#Head_1.Done** gesetzt. Dadurch wird signalisiert, dass die Bearbeitung des Befehls beendet ist.

Das Rücksetzen der Statusbits wird nacheinander für alle weiteren Köpfe durchgeführt. Danach erfolgt ein Sprung zur Sprungmarke endG. Die Sprungmarke endG stellt das Ende des Funktionsbausteins dar.

6.9 Ablauf der Analyse der Funktion FC 50

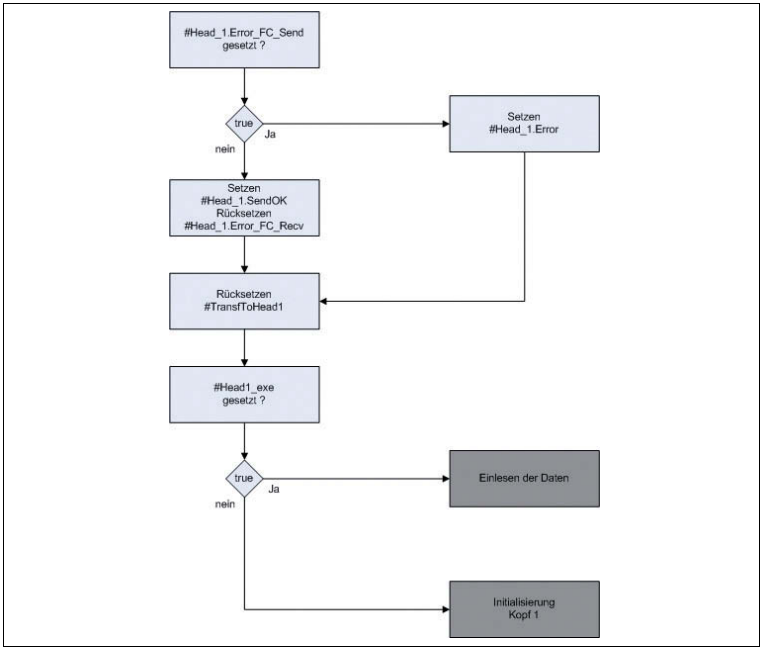


Abbildung 6.9: Ablauf der Analyse der Funktion FC 50 Kopf 1

Die Analyse der Funktion FC 50 hat die Aufgabe, einen Fehler bei der Ausführung der Funktion FC 50 festzustellen. Die Funktion FC 50 wird innerhalb des Funktionsbausteins bei der Initialisierung und dem Ausführungsteil der Befehle aufgerufen. Nach der Ausführung der Funktion in den entsprechenden Programmteilen, erfolgt ein Sprung zur Analyse des FC 50. Mit Hilfe der Analyse wird festgestellt, ob bei der Ausführung der Funktion ein Fehler aufgetreten ist und die Daten der Ausgangsdatenfelder nicht an die IDENTControl übermittelt werden konnten.

Nachfolgend ist der Ablauf der Analyse der Funktion FC 50 beispielhaft für den Kopf 1 erläutert. Für die Köpfe 2, 3 und 4 erfolgt der Ablauf der Analyse analog. Deshalb wird an dieser Stelle darauf verzichtet, näher auf die übrigen Analysen einzugehen.

Zu Beginn der Analyse wird überprüft, ob das Bit **#Head_1.Error_FC_Send** gesetzt ist. Wenn ein Fehler bei der Ausführung der FC 50 aufgetreten ist, erfolgt ein Sprung zur Sprungmarke err1. Dort wird das Fehlerbit **#Head_1.Error** gesetzt. Bei erfolgreicher Ausführung der Funktion FC 50 konnte das Ausgangsdatenfeld von Kopf 1 an die IDENTControl übermittelt werden. Deshalb wird das Bit **#Head_1.SendOK** gesetzt. Anschließend erfolgt das Rücksetzen des Bits **#Head_1.Error_FC_Recv**. Das Bit wird an dieser Stelle zurückgesetzt, weil sich das Einlesen der Rückantwort der IDENTControl direkt an das Senden des Ausgangsdatenfeldes anschließt. Somit wird verhindert, dass die Ausführung der Funktion FC 60 von vornherein als inkorrekt aufgefasst wird. Danach erfolgt ein Sprung zur Sprungmarke BE1. Die Sprungmarke BE1 wird unabhängig davon, ob die Funktion FC 50 erfolgreich oder nicht erfolgreich ausgeführt wurde, durchlaufen. In diesen Programmabschnitt wird zunächst das Bit **#TransfToHead1** zurückgesetzt. Dadurch ist es möglich, sofern keine weiteren Sperr- Bits gesetzt sind, das Ausgangsdatenfeld mit neuen Befehlsparametern zu belegen. Anschließend wird das Bit **#Head1_exe** überprüft. Mit Hilfe dieses Bits wird das Ziel des Rücksprunges aus der Analyse festgelegt. Das Bit ist gesetzt, wenn die Analyse infolge der Befehlsausführung durchgeführt wurde. Es erfolgt ein Sprung zum Einlesen der Rückantwort an die Sprungmarke ReDa in das Netzwerk 8. Dort wird die Einlese- Routine von Daten durchlaufen. Wenn die Analyse der Funktion FC 50 innerhalb der Initialisierung durchgeführt wurde, ist das Bit **#Head1_exe** nicht gesetzt. Somit erfolgt ein Rücksprung zur Initialisierungs-Routine zu der Sprungmarke F501.

Die Analyse für die übrigen Köpfe schließt sich an den hier erläuterten Programmabschnitt an. Der Aufruf der Analyse erfolgt analog, entweder aus der Initialisierung oder aus der Befehlsausführung des entsprechenden Kopfes. Der Rücksprung aus der Analyse des sich anschließenden Programmteils wird ebenfalls über die Variable #HeadX_exe realisiert. Dadurch erfolgt der Rücksprung in die Initialisierung des entsprechenden Kopfes oder zu der Ausführung der Funktion FC 60 an die Sprungmarke ReDa.

6.10 Ablauf der Analyse der Funktion FC 60

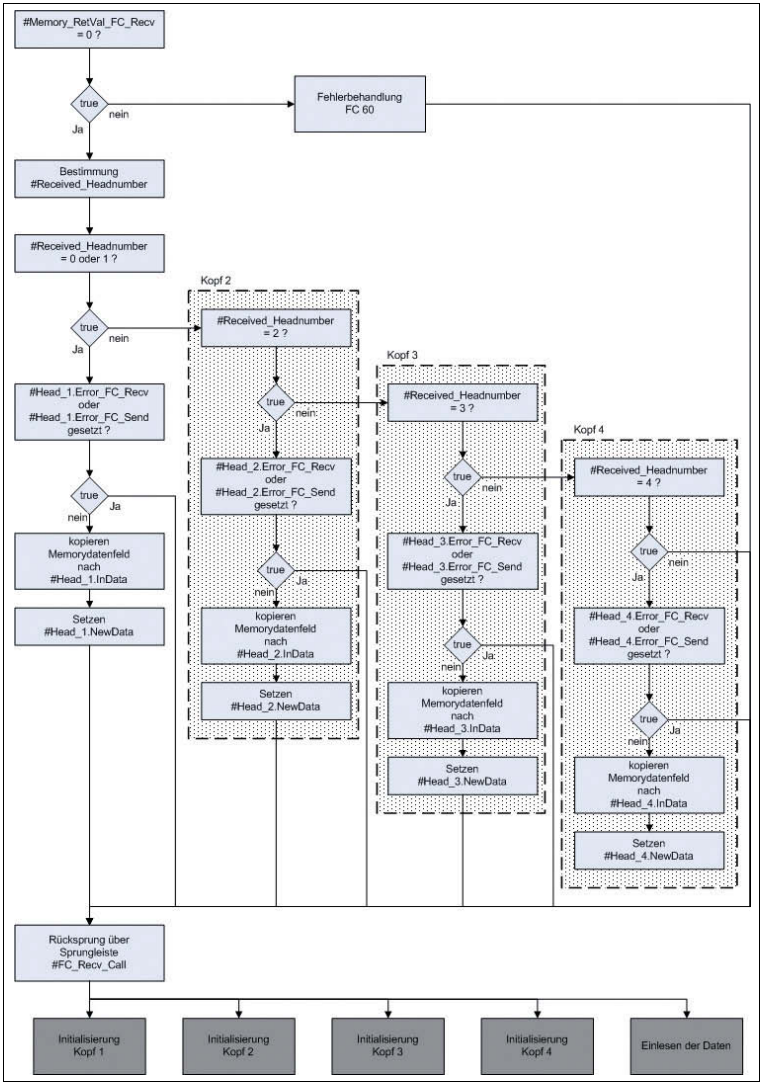


Abbildung 6.10: Ablauf der Analyse der Funktion FC 60

Die Analyse der Funktion FC 60 hat die Aufgabe, einen Fehler bei der Ausführung der Funktion FC 60 festzustellen. Die Funktion FC 60 wird innerhalb des Funktionsbausteins bei der Initialisierung und bei der Einlese-Routine aufgerufen. Nach der Ausführung der Funktion FC 60 in den entsprechenden Programmabschnitten erfolgtein Sprung zur Analyse der Funktion FC 60 in das Netzwerk 21. Zunächst wird überprüft, ob bei der Ausführung der Funktion ein Fehler aufgetreten ist. Dazu wird überprüft, ob die Variable **#Memory_RetVal_FC_Recv** mit einem Wert belegt ist. In diesem Fall erfolgt eine entsprechende Fehlerbehandlung an der Sprungmarke er60. Wurde die Funktion FC 60 fehlerfrei ausgeführt, wird zunächst die Kopfnummer der eingelesenen Daten aus den Memorydatenfeld isoliert. Die Kopfnummer der eingelesenen Daten wird der Variablen **#Received_Headnumber** zugewiesen.

Anschließend wird anhand der Kopfnummer das Memorydatenfeld in das zugehörige Eingangsdatenfeld kopiert. Daraufhin wird das Bit **#Head_X.NewData** gesetzt. Dieses Bit signalisiert, dass neue Daten im Eingangsdatenfeld des zugehörigen Kopfes zur Auswertung bereitstehen.

Im letzten Schritt erfolgt mit Hilfe einer Sprungleiste der Rücksprung in das Programmteil, von dem aus die Analyse aufgerufen wurde.

6.11 Analyse der Eingangsdatenfelder

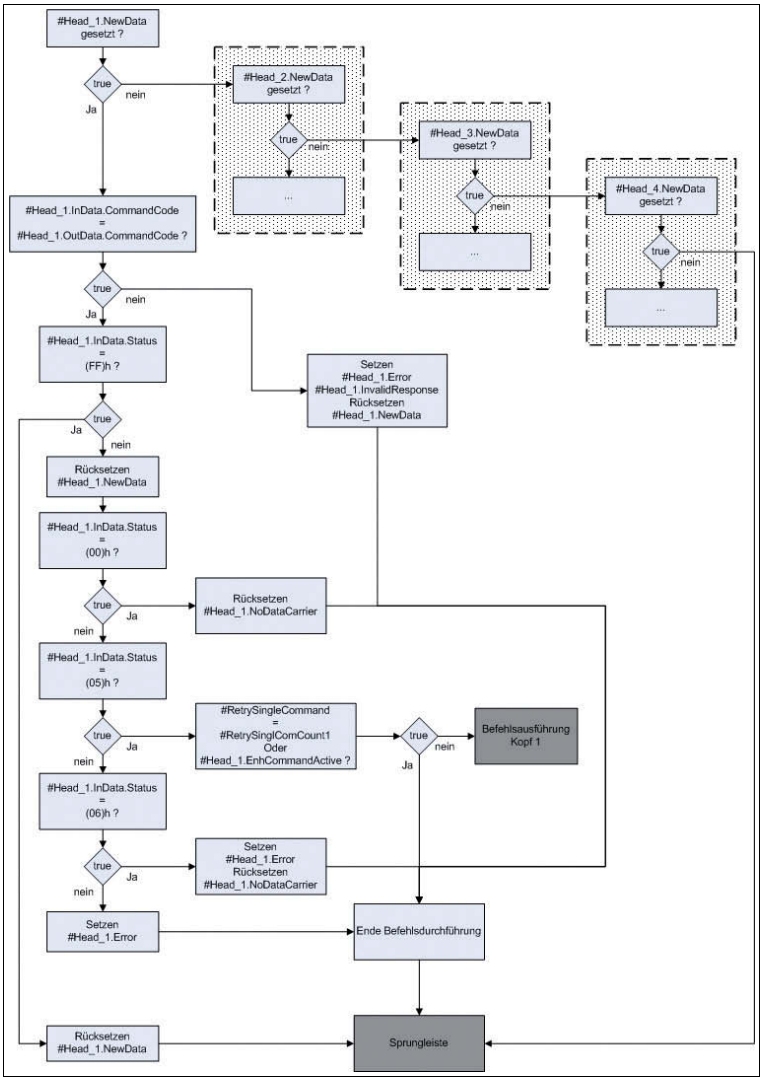


Abbildung 6.11: Analyse der Eingangsdatenfelder

Dieser Programmteil des Funktionsbausteins hat die Aufgabe, die neu eingelesenen Daten der Eingangsdatenfelder zu analysieren. Die Analyse der Eingangsdatenfelder ist erforderlich, sobald Daten aus dem Memorydatenfeld in die jeweiligen Eingangsdatenfelder kopiert wurden.

Zu Beginn der Analyse wird festgestellt, in welchem der Eingangsdatenfelder neue Daten vorhanden sind. Wenn in einem Eingangsdatenfeld neue Daten zur Auswertung bereitstehen, ist das Bit **#Head_1.NewData** gesetzt und es wird zur Auswertung an die entsprechende Sprungmarke gesprungen.

Dort erfolgt die eigentliche Analyse der Eingangsdaten. Dazu wird zunächst der Befehlscode des Ausgangsdatenfeldes mit dem Befehlscode des Eingangsdatenfeldes verglichen. Sind die beiden Befehlscodes nicht identisch, ist bei der Befehlsausführung der IDENTControl ein Fehler aufgetreten und es erfolgt ein Sprung zur Sprungmarke era1. Dort werden die Fehlermeldebit **#Head_1.Error** und **#Head_1.InvalidResponse** gesetzt.

Wenn beide Befehlscodes gleich sind, wird im nächsten Schritt der Status der eingelesenen Daten überprüft.

Hat der Status den Wert (FF)h, wurde der an die IDENTControl gesendete Befehl zwar "verstanden“, aber der Befehl wird zurzeit noch ausgeführt, so dass die eingelesenen Daten verworfen werden müssen. Deshalb wird an der Sprungmarke sff1 das Bit **#Head_1.NewData** zurückgesetzt.

Wenn die Befehlsdurchführung der IDENTControl abgeschlossen ist, hat der Status der Wert (00)h.

Der Status der eingelesenen Daten hat den Wert (05)h, wenn sich bei der Befehlsausführung kein Datenträger im Erfassungsbereich des Schreiblesekopfes befunden hat. Dadurch wird der Ausführungszähler inkrementiert und der Befehl nochmals ausgeführt. Wenn die maximale Anzahl der Befehlswiederholungen erreicht ist, wird das Bit **#Head_1.NoDataCarrier** gesetzt und die Befehlsausführung nicht wiederholt.

Besitzen die Eingangsdaten einen anderen Statuswert, ist ein Fehler bei der Befehlsdurchführung aufgetreten und es wird das Fehlermeldebit **#Head_1.Error** gesetzt.

7 Anhang

7.1 Auflistung der Parameter

7.1.1 Eingangsparameter (IN-Parameter)

- **IDENT_Control_Address**

  Adresse der IDENTControl innerhalb des Kommunikationsnetzes

- **ID**

  Verbindungsnummer

- **TCP_Telegramlength**

  Telegrammlänge (Port 10000 -> 34 Byte; Port 10001 -> 66 Byte)

- **Timeout**

  Zeitfenster zur Timeoutüberwachung

- **RetrySingleCommand**

  Anzahl der maximalen Befehlswiderholungen

- **HeadXDataFixcode**

  HeadXDataFixcode = 0 -> Zugriff auf Fixcode
  HeadXDataFixcode = 1 -> Zugriff auf den Datenbereich

- **HeadXSingleEnhanced**

  HeadXSingleEnhanced = 0 -> Ausführung eines Single-Befehls
  HeadXSingleEnhanced = 1 -> Ausführung eines Enhanced-Befehls

- **HeadXQuit**

  Start eines Quit-Befehls zum Abbruch eines Enhanced-Befehls

- **HeadXRead**

  Start eines Read-Befehls

- **HeadXWrite**

  Start eines Write-Befehls

- **QuitErrorHeadX**

  Start eines QuitError-Befehls zur Aufhebung der Fehlerverriegelung

- **HeadXSpecialCommand**

  Start eines Special-Commands

- **IC_Command_on_Head1**

  Start eines Befehls an die IDENTControl. Dabei erfolgt keine Angabe des Kanals und somit keine Befehlsausführung durch die Schreib-/Leseköpfe.

7.1.2 Durchgangsparameter

- **InitFinish**

  Ende der Initialisierung aller Köpfe

- **SetRestart**

  Start eines Restarts

7.1.3 Static Parameter (STAT-Parameter)

- **Head_X.InData**

  Head_X.InData ist eine Datenstruktur, welche die Rückantwort der IDENTControl auf einen Befehl enthält. Die Datenstruktur setzt sich aus verschiedenen Elementen unterschiedlichen Datentyps zusammen.

  - **Head_X.InData.TelegrammLengthHigh**

    oberes Byte von der Telegrammlänge

  - **Head_X.InData.TelegrammLengthLow**

    unteres Byte von der Telegrammlänge (enthält #TCP_Telegramlength)

  - **Head_X.InData.CommandCode**

    Befehlscode der eingelesenen Antwort

  - **Head_X.InData.Channel**

    Obere 4 Bits enthalten mögliche Wortanzahl; untere 4 Bits enthalten die Kanalkennung

  - **Head_X.InData.Status**

    Ausführungszustand

  - **Head_X.InData.ExecutionCounter**

    Ausführungszähler

  - **Head_X.InData.DW1 … DW15**

    Datenfeld für die eingelesenen Nutzdatenwörter

- **Head_X.OutData**

  Head_X.OutData ist ebenfalls eine Datenstruktur. Diese Struktur beinhaltet die Daten, welche zur Ausführung eines Befehls an die IDENTControl gesendet werden. Das Datenfeld ist in mehrere Elemente unterschiedlichen Datentyps gegliedert.

  - **Head_X.OutData.TelegrammLengthHigh**

    Oberes Byte der Telegrammlänge

  - **Head_X.OutData.TelegrammLengthLow**

    Unteres Byte der Telegrammlänge

  - **Head_X.OutData.CommandCode**

    Befehlscode des auszuführenden Befehls

  - **Head_X.OutData.Channel**

    Kanalkennung

  - **Head_X.OutData.Wordadr_High**

    Oberes Byte der Adresse, ab der die Daten gelesen/geschrieben werden; bei Change- Tag Befehl -> oberes Byte der Datenträgerkennung

  - **Head_X.OutData.Wordadr_Low**

    unteres Byte der Adresse, ab der die Daten gelesen/geschrieben werden; bei Change- Tag Befehl -> unteres Byte der Datenträgerkennung

  - **Head_X.OutData.DW1 … DW15**

    Ausgangsdatenfeld der Nutzdatenwörter

- **Head_X.WordAddress**

  Startadresse für Datenträgerzugriff

- **Head_X.TimeoutActiv**
  Timeoutüberwachung ist aktiv
- **Head_X.InvalidResponse**
  Ungültige Antwort der IDENTControl
- **Head_X.QuitError**
  Aufheben der Fehlerverriegelung
- **Head_X.NewData**
  Neue Daten stehen zur Analyse bereit
- **Head_X.NotExist**
  Kein Schreib-/Lesekopf angeschlossen
- **Head_X.ExistTC**
  Schreib-/Lesekopf angeschlossen
- **Head_X.Error**
  Fehler in der Befehlsausführung der IDENTControl
- **Head_X.TimeoutOccured**
  Zeitfenster der Timeoutüberwachung abgelaufen
- **Head_X.ReceiveOK**
  Antwort der IDENTControl empfangen
- **Head_X.SendOK**
  Daten an die IDENTControl gesendet
- **Head_X.NoDataCarrier**
  Kein Datenträger im Erfassungsbereich
- **Head_X.Done**
  Enhanced-Befehl -> Daten gelesen bzw. geschrieben
  Single-Befehl -> Befehlsausführung beendet
- **Head_X.Busy**
  Befehl in Bearbeitung
- **Head_X.Error_FC_Recv**
  Fehler bei der Ausführung der FC 60
- **Head_X.Error_FC_Send**
  Fehler bei der Ausführung der FC 50
- **Head_X.EnhCommandActive**
  Enhanced-Befehl wird ausgeführt
- **Head_X.SglCommandActive**
  Single-Befehl wird ausgeführt
- **Head_X.WordNum**
  Anzahl der zu übertragenden Nutzdatenwörter
- **Head_X.RetVal_FC_Recv**
  Enthält einen Fehlercode bei der Ausführung der FC 60
- **Head_X.RetVal_FC_Send**
  Enthält einen Fehlercode bei der Ausführung der FC 50

2011-01

- **Head_X.SpecialCommand**

  Das Datenfeld enthält die Parameter zur Ausführung eines Special-Commands. Mit Hilfe eines Special-Commands kann man Befehle ausführen, die nicht durch die Befehlsliste des Funktionsbausteins abgedeckt sind. Zur Ausführung eines Special-Commands muss der Anwender die Befehlsparameter des auszuführenden Befehls in dieses Datenfeld eingeben. Die hier eingetragenen Parameter werden dann innerhalb des Funktionsbausteins in das Ausgangsdatenfeld überführt und zur IDENTControl übertragen.

  - **Head_X.SpecialCommand.Code**

    Befehlscode

  - **Head_X.SpecialCommand.Channel**

    Kanalkennung und evtl. Anzahl zu übertragender Nutzdatenwörter

  - **Head_X.SpecialCommand.Parameter1 …5**

    Befehlsparameter

- **Head_X.Memory**

  Das Memory-Datenfeld enthält die Daten, welche von der IDENTControl an die SPS gesendet werden. Alle eingehenden Daten werden unabhängig vom IDENT-Kanal in diesem Datenfeld abgelegt. Nach der Überprüfung des IDENT-Kanals der eingehenden Daten, erfolgt die Umkopierung des Memory-Datenfeldes in das Eingangsdatenfeld des entsprechenden IDENTKanals. Der Aufbau des Memory-Datenfeldes ist identisch zu den Eingangsdatenfeldern der IDENT-Kanäle und wird deshalb hier nicht näher beschrieben.

7.2 Befehlsliste

Befehl	Code	Parametrierung	Ausführung
Single-Read-Fixcode	(01)h	Keine	#HeadDataFixcode = 1 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 0
Enhanced-Read-Fixcode	(1D)h	Keine	#HeadXDataFixcode = 1 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 1
Single-Read-Data	(10)h	#HeadX.WordAddress #HeadX.WordNum	#HeadXDataFixcode = 0 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 0
Enhanced-Read-Data	(19)h	#HeadX.WordAddress #HeadX.WordNum	#HeadXDataFixcode = 0 #HeadXRead = 1 #HeadXWrite = 0 #HeadXSingleEnhanced = 1
Single-Write-Data	(10)h	#HeadX.WordAddress #HeadX.WordNum #HeadX.OutData.DW	#HeadXDataFixcode = 0 #HeadXRead = 0 #HeadXWrite = 1 #HeadXSingleEnhanced = 0
Enhanced-Write-Data	(19)h	#HeadXWordAddress #HeadXWordNum #HeadX.OutData.DW	#HeadXDataFixcode = 0 #HeadXRead = 0 #HeadXWrite = 1 #HeadXSingleEnhanced = 0
Special-Command	(??)h	#Head_X.SpecialCommand.Code #Head_X.SpecialCommand.Channel #Head_X.SpecialCommand. Parameter	#HeadXSpecialCommand = 1 #IC_Command_on_Head1 = 0

Befehl	Code	Parametrierung	Ausführung
IDENT-Control-Command	(??)h	#Head_1.SpecialCommand. Code #Head_1.SpecialCommand. Parameter	#Head1SpecialCommand = 1 #IC_Command_on_Head1 = 1
QuitError-Befehl	-	-	#QuitErrorHeadX
Quit-Befehl	-	-	#HeadXQuit

7.3 Code- / Datenträger

Typ	#HeadXTagType	Zugriff	Datenbereich	Fixcodelänge
IPC02	W#16#3032	Read Fixcode	-	5 Byte
IPC03	W#16#3033	Read Fixcode Read Data Write Data	116 Byte	4 Byte
IPC10	W#16#3130	Read Data Write Data	12 Byte	-
IPC11	W#16#3131	Read Data Write Data	5 Byte	-
IPC12	W#16#3132	Read Fixcode Read Data Write Data	8 kByte	4 Byte
IPC14	W#16#3134	Read Data Write Data	5 Byte	-
IQC20	W#16#3230	Read Fixcode Read Data Write Data	-	8 Byte
IQC21	W#16#3231	Read Fixcode Read Data Write Data	112 Byte	8 Byte
IQC22	W#16#3232	Read Fixcode Read Data Write Data	256 Byte	8 Byte
IDC1k	W#16#3530	Read Fixcode Read Data Write Data	128 Byte	4 Byte
ICC	W#16#3532	Read Fixcode	-	7 Byte
MVC60	W#16#3630	Read Fixcode Read Data Write Data	-	8 kByte

FABRIKAUTOMATION – SENSING YOUR NEEDS

Zentrale weltweit
Pepperl+Fuchs GmbH
68307 Mannheim · Deutschland
Tel. +49 621 776-0
E-Mail: info@de.pepperl-fuchs.com

Zentrale USA
Pepperl+Fuchs Inc.
Twinsburg, Ohio 44087 · USA
Tel. +1 330 4253555
E-Mail: sales@us.pepperl-fuchs.com

Zentrale Asien
Pepperl+Fuchs Pte Ltd.
Singapur 139942
Tel. +65 67799091
E-Mail: sales@sg.pepperl-fuchs.com

www.pepperl-fuchs.com

PEPPERL+FUCHS
SENSING YOUR NEEDS

TDOCT1190A_GER
01/2011