Handbuch zur Inbetriebnahme des Pepperl + Fuchs Identifikationssystems IDENT-Control mit Profibus-DP-Anschluss an einer S7-300 Steuerung

PEPPERL+FUCHS
IDENT

Inhaltsverzeichnis

1 Eingesetzte Geräte

In der folgenden Übersicht sind alle zur Inbetriebnahme der IDENT-Control erforderlichen Komponenten in der benötigten Anzahl aufgelistet.

Anzahl	Bezeichnung	Hersteller	Beschreibung
1	PS 307 5A	Siemens	Stromversorgung für die SPS
1	CPU 315-2 DP	Siemens	CPU-Baugruppe der SPS mit integrierter Profibus-Master-Schnittstelle
1	PG 740 P II	Siemens	Programmiergerät (alternativ kann auch ein Laptop mit der erforderlichen MPI-Schnittstelle verwendet werden)
1	Step7 V5.1	Siemens	Programmiersoftware
1	IC-KP-B6	Pepperl + Fuchs	IDENT-Control (Schreib-/Lesestation zum Anschluss von max. 4 Schreib-/Leseköpfen)
1 .. 4	IPH-FP-V1	Pepperl + Fuchs	Schreib-/Lesekopf

Tabelle 1: Übersicht der eingesetzten Geräte

Hinzu kommen noch:

- 1 x MPI-Verbindungskabel zum Verbinden des Programmiergeräts mit der SPS
- 1 x Profibuskabel zum Anschluss der IDENT-Control an die SPS
- 1-4 Anschlusskabel mit V1-Stecker zum Anschluss der Schreib-/Leseköpfe an die IDENT-Control

2 Aufbauen und Anschließen

Wie die S7-Komponenten aufzubauen und anzuschließen sind, entnehmen Sie bitte dem Handbuch „Automatisierungssystem S7-300, Aufbauen, CPU-Daten“ der Fa. Siemens.
Die IDENT-Control wird an eine Spannung von 20 .. 30 V DC angeschlossen und die Schreib-/Leseköpfe sind mit der Station zu verbinden. Anschließend stellen Sie noch die Verbindung der IDENT-Control zur SPS über das Profibuskabel her.

Genauere Ausführungen zum Anschluss und zur Installation der IDENT-Control sind im zugehörigen Handbuch von Pepperl + Fuchs zu finden.
Weder die SPS noch die IDENT-Control haben integrierte Abschlusswiderstände an ihren Profibus-Anschlüssen. Deshalb sind die Abschlusswiderstände am Profibus-Kabel einzuschalten, wenn die IDENT-Control bzw. die SPS das Ende des Profibuszweigs darstellen.

Im folgenden wird eine IDENT-Control-Station mit der Profibusadresse 3 an die SPS angeschlossen. Für weitere Stationen erfolgt der Anschluss und die Konfiguration analog.

3 Konfiguration der Hardware

3.1 Installieren der GSD-Datei

Zunächst ist die beiliegende GSD-Datei in die Simatic-Step7-Software zu installieren. Dazu verfahren Sie wie folgt:

Gehen Sie in den Hardware-Konfigurator der Step7-Software und schließen Sie dort alle ggf. geöffneten Projektfenster. Anschließend klicken Sie im Menü Extras auf „Neue GSD installieren“ (siehe Abb. 1).

Abb. 1: Installieren der GSD-Datei (Schritt 1)

Im sich daraufhin öffnenden Menü wählen Sie den Ort, an dem sich die GSD-Datei befindet (hier: Diskettenlaufwerk A:), markieren Sie die zu installierende Datei und klicken Sie auf Öffnen (Abb. 2). Die GSD-Datei wird nun installiert.

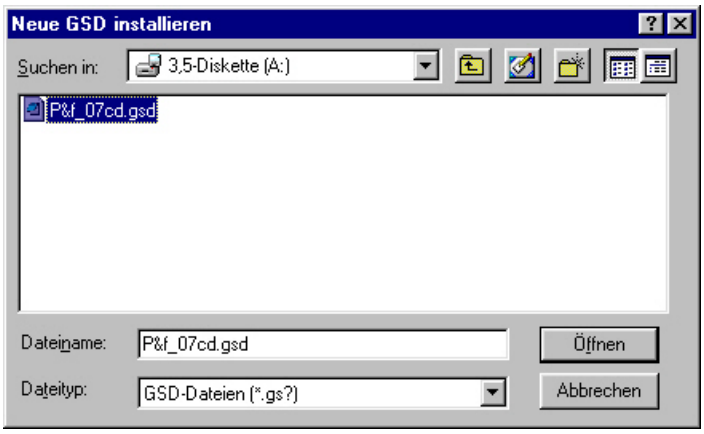

Abb. 2: Installieren der GSD-Datei (Schritt 2)

Dann klicken Sie im Menü Extras auf den Unterpunkt „Katalog aktualisieren“ (Abb. 3).

Abb. 3: Installieren der GSD-Datei (Schritt 3)

Damit ist die Installation der GSD-Datei abgeschlossen und das Gerät IDENT-Control ist unter dem Namen IC-KP-B6 im Hardware-Katalog in folgendem Pfad zu finden:

Profibus-DP → Weitere Feldgeräte → Identsysteme

3.2 Profibus-Konfiguration

Zuerst sind die Siemens S7-Komponenten in gewohnter Weise in die Hardware-Konfiguration einzubinden. Die Profibus-Schnittstelle der SPS ist defaultmäßig als Profibus-Master eingestellt und es kann nun ein Profibus-Mastersystem erzeugt werden.

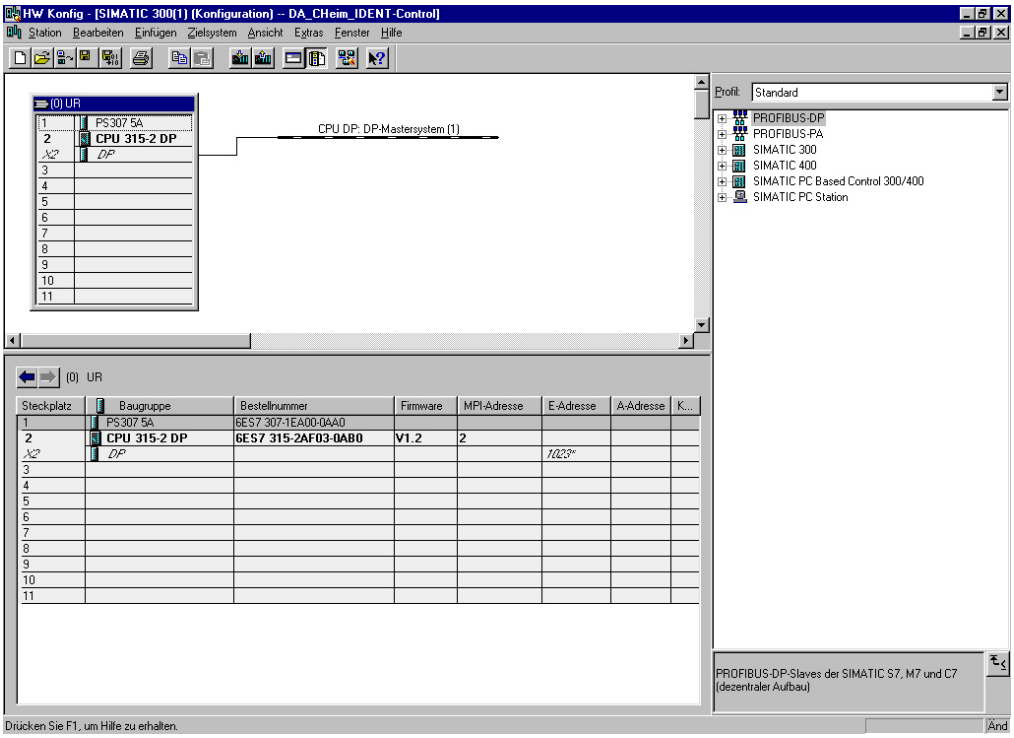

Abb. 4: Hardwarekonfiguration (Schritt 1)

Anschließend wählen Sie aus dem Hardwarekatalog in dem zuvor genannten Pfad die IDENT-Control unter dem Namen IC-KP-B6 aus und ziehen sie bei gedrückter Maustaste auf den Profibusstrang. Dort lassen Sie die Maustaste los.

Im sich nun öffnenden Fenster (Abb. 5) wird die Profibus-Adresse der IDENT-Control-Einheit eingestellt, in diesem Fall Adresse 3.

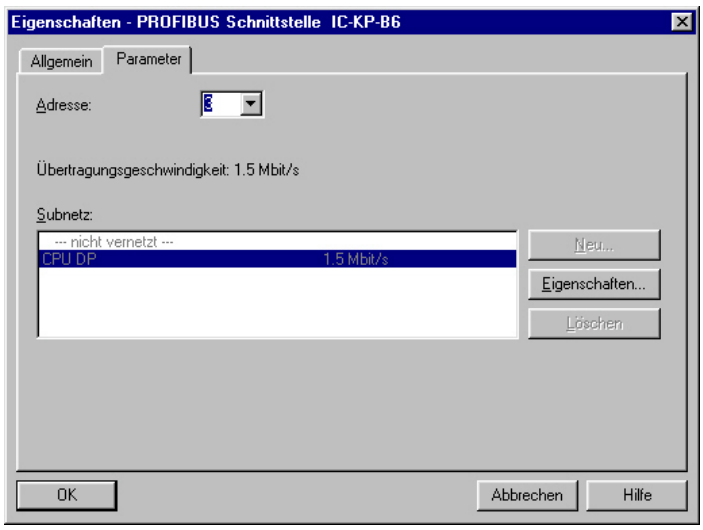

Abb. 5: Hardwarekonfiguration (Schritt 2)

Die Hardwarekonfiguration sieht dann aus wie in Abb. 6.

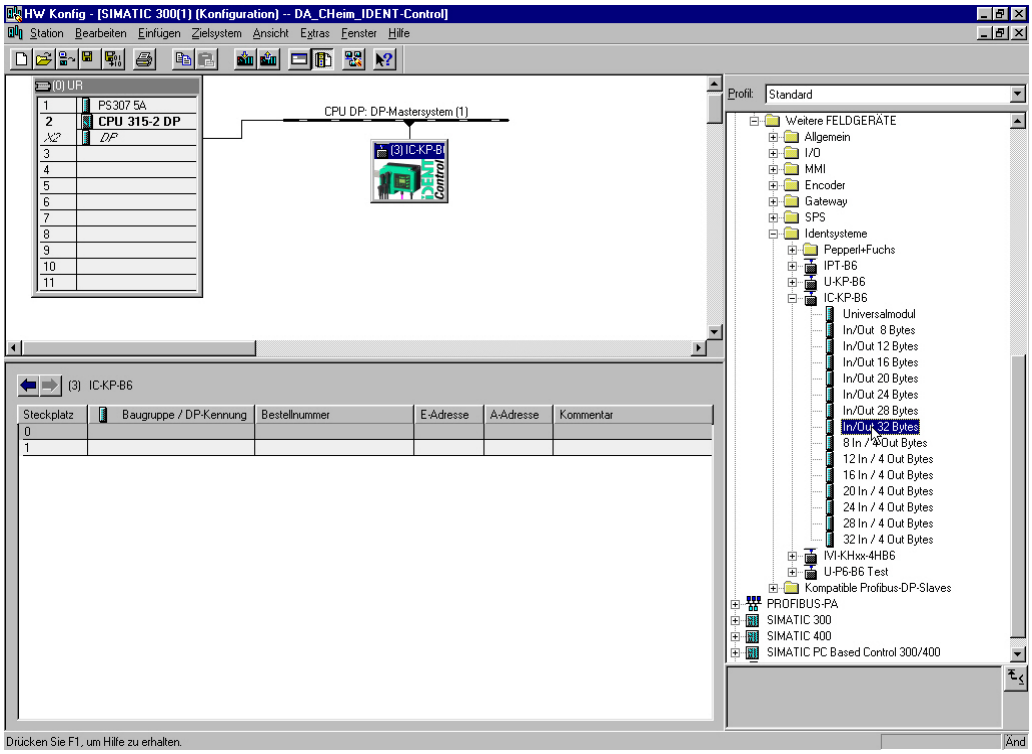

Abb. 6: Hardwarekonfiguration (Schritt 3)

Nun müssen Sie durch Hinzufügen eines der im Hardwarekatalog unter IC-KP-B6 angezeigten Module die Länge der Ein-/Ausgangsdaten der IDENT-Control festlegen. Dazu wählen Sie das gewünschte Modul aus und ziehen es in die Modulfelder der IDENT-Control. Im Beispiel wird das Modul „In/Out 32 Bytes“ gewählt.

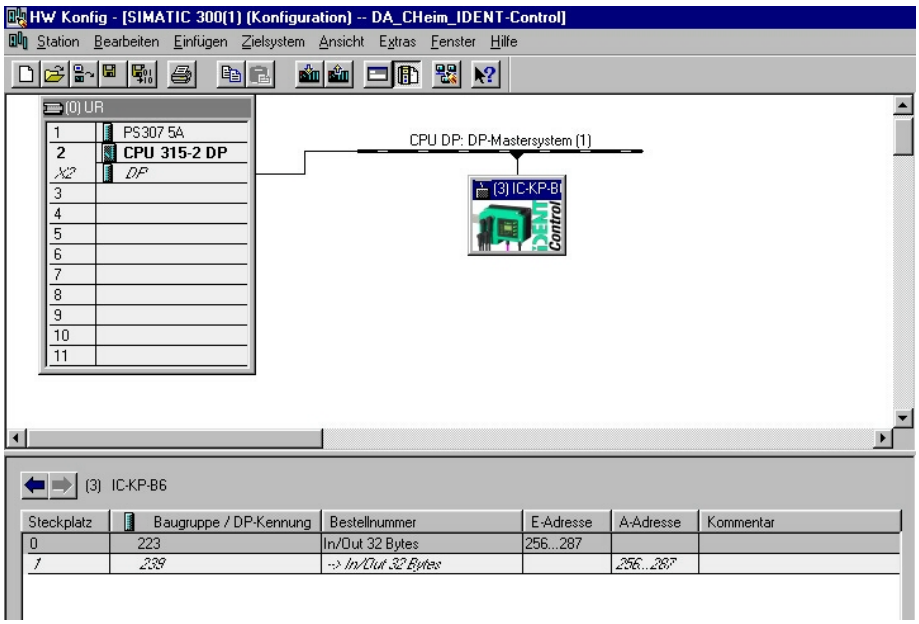

Abb. 7: Hardwarekonfiguration (Schritt 4)

Wie in Abb. 7 zu erkennen ist, werden beim Einfügen des Moduls die Ein-/Ausgangsadressen von der SPS automatisch auf die nächsten freien Adressen, hier ab E 256 bzw. A 256, gelegt. Diese automatisch vorgenommene Einstellung kann vom Anwender durch Doppelklicken auf das E- bzw. A-Adresse-Feld und Eingeben der gewünschten Adresse verändert werden.
Die erste Ein-/Ausgangsadresse ist gleichzeitig die Adresse, unter der die IDENT-Control im späteren Programm angesprochen wird (hier also Adresse 256_{dez} bzw. 100_{hex}).
Damit ist die Hardwarekonfiguration abgeschlossen. Sie muss nun noch in die SPS geladen werden.

Bevor das System in Betrieb genommen werden kann, muss die Profibus-Adresse der IDENT-Control auf die in der Hardwarekonfiguration gewählte Adresse (hier: 3) eingestellt werden. Danach ist die Versorgungsspannung der IDENT-Control kurz ab- und wieder zuzuschalten, damit die neue Konfiguration übernommen wird. Nähere Erläuterungen dazu gibt das Handbuch der IDENT-Control.

4 Anmerkungen zur IDENT-Control

4.1 Unterscheidung Single- und Enhanced-Befehle

Generell wird bei den Identsystemen von Pepperl + Fuchs zwischen den Single- und Enhanced-Befehlen unterschieden. Single-Befehle sind alle Befehle die nach erfolgter Befehlsvergabe nur einmalig ausgeführt werden. Für eine weitere Ausführung ist die erneute Übertragung des Befehlscodes und der Parameter notwendig.
Enhanced-Befehle sind „Dauerbefehle“, d.h. der gegebene Befehl wird solange ausgeführt, bis ein neuer Befehl gegeben wird. Der Befehlscode für einen Enhanced-Befehl muss dabei nur einmal an das Identsystem gesendet werden. Das Identsystem schickt automatisch Antworten, wenn neue Daten gelesen oder geschrieben wurden. Auch eine Änderung des Befehlsstatus wird durch Senden einer entsprechenden Antwort des Identsystems mitgeteilt.

4.2 Genereller Befehlsablauf

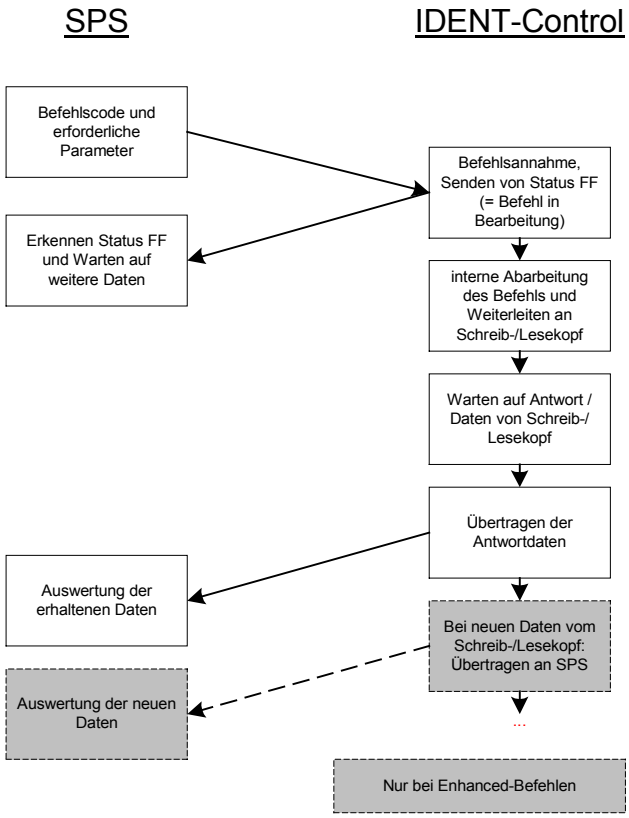

Abb. 8: Ablauf der Bearbeitung eines Befehls bei der IDENT-Control

Um einen Befehl von der IDENT-Control ausführen zu lassen, müssen der Befehlscode und die erforderlichen Parameter von der SPS an die IDENT-Control übertragen werden. Diese antwortet zunächst mit einer Rückmeldung, die den Status FF_h enthält. Das bedeutet, dass sie den Befehl angenommen hat und ihn bearbeitet. Intern leitet sie den Befehl an den oder die betreffenden Schreib-/Leseköpfe weiter und wartet dann auf eine Antwort von diesen. Diese Antwort wird anschließend an die SPS übertragen und kann dort ausgewertet werden. Bei Enhanced-Befehlen folgen ggf. weitere Antworten, wenn neue Daten oder ein neuer Befehlsstatus vorliegen.

4.3 Aufbau eines Befehls und der dazugehörigen Antwort

Der Aufbau eines Befehls und der dazugehörigen Antwort wird am Beispiel des Single-Read-Befehls dargestellt. Mit diesem Befehl wird einmal versucht Daten von einem Datenträger zu lesen. Alle weiteren Befehle und die erforderlichen Parameter sind im Handbuch der IDENT-Control aufgeführt.

single read words (SR):

Byte	Inhalt	Bit Nr. 7	6	5	4	3	2	1	0
Byte 0	Befehlscode	0	0	0	1	0	0	0	0
Byte 1	Wortanzahl/Kopfnummer/Togglebit	<WordNum>				<HeadNo>			<T>
Byte 2	Wortadresse	<WordAddr> (High Byte)							
Byte 3	Wortadresse	<WordAddr> (Low Byte)							
Byte 4	unbenutzt	-	-	-	-	-	-	-	-
Byte 5	unbenutzt	-	-	-	-	-	-	-	-
Byte 6	unbenutzt	-	-	-	-	-	-	-	-
Byte 7	unbenutzt	-	-	-	-	-	-	-	-

Antwort:

Byte	Inhalt	Bit Nr. 7	6	5	4	3	2	1	0
Byte 0	Befehlscode	0	0	0	1	0	0	0	0
Byte 1	Wortanzahl/Kopfnummer/Togglebit	<WordNum>				<HeadNo>			<T>
Byte 2	Status	<Status>							
Byte 3	Ausführungszähler	<ExecCounter>							
Byte 4	Daten 00 ... FFh	<Data>							
Byte 5	Daten 00 ... FFh	<Data>							
Byte 6	Daten 00 ... FFh	<Data>							
Byte 7	Daten 00 ... FFh	<Data>							
...	Daten 00 ... FFh	<Data>							
Byte N[a)]	Daten 00 ... FFh	<Data>							

a) N = 4 x <WordNum>

Es wird einmal versucht, <WordNum> 32-Bit-Worte ab Adresse <WordAddr> zu lesen.

Abb. 9: Parametertabelle für den Single-Read-Befehl

Der Befehlscode ist jeweils abhängig vom verwendeten Befehl (in diesem Fall 10_h) und steht in Byte 0.

Byte 1 enthält die Anzahl der zu lesenden 32-Bit-Wörter, die Kopfnummer und das Togglebit. Bei Verwendung des Profibus-DP können gleichzeitig maximal 32 Byte Daten übertragen werden. Da die ersten 4 Byte vom Befehlscode und den Parametern belegt sind, bleiben noch 28 Byte, also 7 mal 32 Bit für Nutzdaten übrig. Somit kann die Anzahl der zu lesenden 32 Bit Wörter maximal 7 sein. Durch die Kopfnummer wird bestimmt, an welchen Kopf der Befehl übertragen wird (001_b z.B. bedeutet Kopf 1, 010_b Kopf 2 etc.). Erläuterungen zur Verwendung des Togglebits sind unter dem folgendem Punkt „Die Bedeutung des Togglebits“ zu finden.

Als weiterer Parameter muss beim Single-Read-Befehl die Startadresse angegeben werden, ab der Daten vom Datenträger gelesen werden sollen (Byte 2 und Byte 3). Die restlichen Bytes sind beim Single-Read-Befehl unbenutzt.

Bei jeder Antwort der IDENT-Control auf einen Befehl werden die Bytes 0 und 1 vom eingegangenen Befehl gespiegelt, d.h. einfach in den Ausgangsdaten der IDENT-Control als Byte 0 und 1 wieder eingefügt. Damit kann in der SPS die eindeutige Zuordnung zum vorher gesen-

deten Befehl erfolgen. Desweiteren werden in der Antwort Informationen über den Status des Befehls und den Stand des Ausführungszählers in Byte 2 und 3 übermittelt. Der Status des Befehls gibt an, ob dieser derzeit noch bearbeitet wird (Status FF_h), fehlerfrei abgearbeitet wurde (Status 00_h) oder ob Fehler aufgetreten sind (Status entspricht dann einem Fehlercode). Der Ausführungszähler gibt an, wie oft der Befehl bearbeitet wurde, bei Single-Befehlen normalerweise einmal. Die weiteren Bytes der Antwort enthalten beim Single-Read-Befehl die gelesenen Daten.

4.4 Die Bedeutung des Togglebits

Die Bedeutung des Togglebits wird deutlich, wenn ein und derselbe Befehl direkt nacheinander mehrfach an die IDENT-Control zur Ausführung übertragen werden soll. Da sich bei Nichtverwendung des Togglebits die erhaltenen Daten im Befehlseingangspuffer der IDENT-Control nicht ändern, sondern in allen Bits identisch sind, ist die IDENT-Control nicht in der Lage diese Daten als einen neuen Befehl zu interpretieren. Der Befehl wird somit nicht nochmals ausgeführt. Invertiert man aber das Togglebit, so unterscheiden sich die beiden aufeinanderfolgenden Befehle in einem Bit und die IDENT-Control erkennt, dass es sich um einen neuen Befehl handelt und führt diesen aus. Wird zwischendurch ein anderer Befehl oder der gleiche Befehl an einen anderen Kopf geschickt, so unterscheiden sich diese Befehle in mehreren Bits und dem Togglebit kommt keine Bedeutung zu, da die IDENT-Control auch ohne ein invertiertes Togglebit den neuen Befehl erkennen kann.

Um in der SPS mit Sicherheit sagen zu können, dass der gesendete Befehl von der IDENT-Control angenommen und bearbeitet wird, muss man die erste Antwort der IDENT-Control auf den Befehl abwarten. Bei jeder Antwort der IDENT-Control werden die ersten beiden Bytes des Befehls wie vorher erklärt gespiegelt. Somit ist die Zuordnung der Daten zum vorher gesendeten Befehl in der SPS durch einfaches Vergleichen der ersten beiden Bytes eindeutig möglich. Etwas Vorsicht ist dabei jedoch geboten was den Stand des Togglebits betrifft, da das Ausgangs-Togglebit der IDENT-Control für alle Antworten immer den Zustand des Togglebits im zuletzt eingegangenen Befehl erhält. Das bedeutet, so lange kein neuer Befehl an die IDENT-Control übertragen wird, ist das Togglebit in den Antworten der IDENT-Control genau wie im zuletzt an die IDENT-Control gesendeten Befehlstelegramm. Wird jedoch ein neuer Befehl mit invertiertem Togglebit gesendet, so hat das Togglebit aller Ant-

worten der IDENT-Control ab diesem Zeitpunkt diesen invertierten Zustand. Dies trifft also auch bei Antworten auf vorher gegebene und noch aktive Befehle zu, bei denen ursprünglich der Zustand des gesendeten Togglebits ein anderer war.

Daraus ergeben sich zwei Möglichkeiten für den Anwender was die Verwendung des Togglebits betrifft:

Wenn klar und sichergestellt ist, dass ein und derselbe Befehl nicht zweimal direkt hintereinander an den gleichen Kopf geschickt wird, kann der Anwender das Togglebit ständig auf 0 belassen, d.h. er braucht es gar nicht zu verwenden. Damit ist dann auch bei allen Antworten das Togglebit 0 und in der SPS können die Ein- und Ausgangsdaten direkt auf Gleichheit miteinander verglichen werden und den betreffenden Befehlen zugeordnet werden.

Wenn der vorerwähnte Fall nicht gewährleistet werden kann, besteht eine weitere Möglichkeit darin, die Invertierung des Togglebits bei jedem Befehl vorzunehmen, unabhängig davon, ob es sich dabei um einen neuen oder den gleichen Befehl nochmals handelt. Es wird damit sichergestellt, dass der Befehl auf jeden Fall von der IDENT-Control als neuer Befehl erkannt und ausgeführt wird.

Desweiteren muss in beiden Fällen aber gewährleistet werden, dass bis zur ersten Antwort auf einen Befehl kein weiterer Befehl an die IDENT-Control gesendet wird. Wird dies nicht getan, wäre folgendes Szenario denkbar:

- SPS sendet Befehl x für Kopf 1 an IDENT-Control
- SPS sendet Befehl y für Kopf 2 (im Fall 2: Togglebit invertiert) an IDENT-Control
- IDENT-Control sendet Antwort auf Befehl x von Kopf 1 an SPS (=> Nachweis, Befehl wurde angenommen und wird bearbeitet)
- SPS sendet Befehl x für Kopf 1 erneut an IDENT-Control (im Fall 2: Togglebit wieder invertiert, also wieder Zustand wie beim ersten Senden)
- Zwischenzeitlich kam keine Antwort auf Befehl y von Kopf 2, d.h. es steht noch die erste Antwort auf den Befehl x von Kopf 1 im Ausgangsspeicher der IDENT-Control
- Diese wird nun wieder an die SPS übertragen

- Die SPS interpretiert diese Daten als Antwort auf den erneut gesendeten Befehl x von Kopf 1 und damit als Bestätigung der erneuten Annahme und Ausführung des Befehls. Diese ist aber tatsächlich noch gar nicht erfolgt...

Im Prozess könnte eine solche Fehlinterpretation schwerwiegende Folgen nach sich ziehen. Es muss also auf jeden Fall sichergestellt werden, dass ein neuer Befehl erst nach Erhalt der ersten Antwort auf den letzten Befehl gesendet werden darf. Nach der ersten Antwort ist sichergestellt, dass der vorhergehende Befehl angenommen wurde. Im zweiten Fall kann dann das Togglebit in der SPS bei allen weiteren Antworten auf diesen Befehl einfach ausmaskiert werden. Damit können auch im zweiten Fall alle weiteren Antworten wieder direkt mit den gesendeten Befehlen, bei denen dann natürlich ebenso das Togglebit ausmaskiert wurde, verglichen und diesen zugeordnet werden.

Im nachfolgend dargestellten Programm wurde die zweite Möglichkeit verwirklicht.

4.5 Die Bedeutung des Ausführungszählers

Der Ausführungszähler gibt an, wie oft der betreffende Befehl abgearbeitet wurde. Bei Single-Befehlen spielt er keine große Rolle, da diese eh nur einmal ausgeführt werden. Interessant wird der Stand des Ausführungszählers bei den Enhanced-Befehlen. Bei jeder erfolgreichen Durchführung des Befehls werden die neuen Daten in den Ausgangspuffer der IDENT-Control gestellt und der Ausführungszähler um eins erhöht. Dadurch kann von der SPS her überwacht werden, ob alle Daten empfangen wurden. Dies ist der Fall wenn der Ausführungszähler sich in den von der SPS empfangenem Datensätzen immer nur um 1 erhöht hat. Beträgt die Differenz mehr als 1 zwischen zwei empfangenen Datensätzen, so wurde die Abholung eines oder entsprechend mehrerer Datensätze dazwischen verpasst. Abhilfe schafft hier meist eine entsprechende Parametrierung der Data Hold Time - mehr dazu im Kapitel „Die Data Hold Time“ (s. Ziff. 7.1.1).

5 Software

5.1 Die verwendeten Bausteine und ihre Funktion

Um eine konsistente Datenübertragung über den Profibus bei einer Datenlänge von 3 Byte oder mehr als 4 Byte gewährleisten zu können, müssen die der Step7-Software beigefügten Systemfunktionen SFC 14 und SFC 15 verwendet werden.
Die folgende Tabelle gibt Aufschluss über die verwendeten Bausteine und deren Bedeutung bzw. Funktion.

Baustein	**Typ**	**Funktion**
SFC 14	Systemfunktion	Lesen konsistenter Daten eines Profibus-DP-Slaves
SFC 15	Systemfunktion	Schreiben konsistenter Daten an einen Profibus-DP-Slave
FB 10 („Ident-Control-System“)	Funktionsbaustein	Steuert den gesamten Kommunikationsablauf mit der IDENT-Control
FC 100 („COMR“)	Funktion	Dient zum Empfangen der Daten der IDENT-Control unter Verwendung der SFC 14
FC 101 („COMS“)	Funktion	Dient zum Senden von Daten an die IDENT-Control unter Verwendung der SFC 15
FC 102 („ANALYSE“)	Funktion	Wertet die von der IDENT-Control empfangenen Daten aus
DB xy (Nr. kann vom Anwender selbst bestimmt werden)	Instanzen-Datenbaustein	Erforderlich einmal je IDENT-Control-Einheit zur Speicherung der Daten dieser Station
VAT 1 .. VAT 4	Variablentabelle	Wird zur Onlinesteuerung der einzelnen Köpfe des Identsystems verwendet

Tabelle 2: Die verwendeten Bausteine und ihre Funktion

Der Zusammenhang der verwendeten Bausteine wird in Abb. 10 nochmals verdeutlicht.

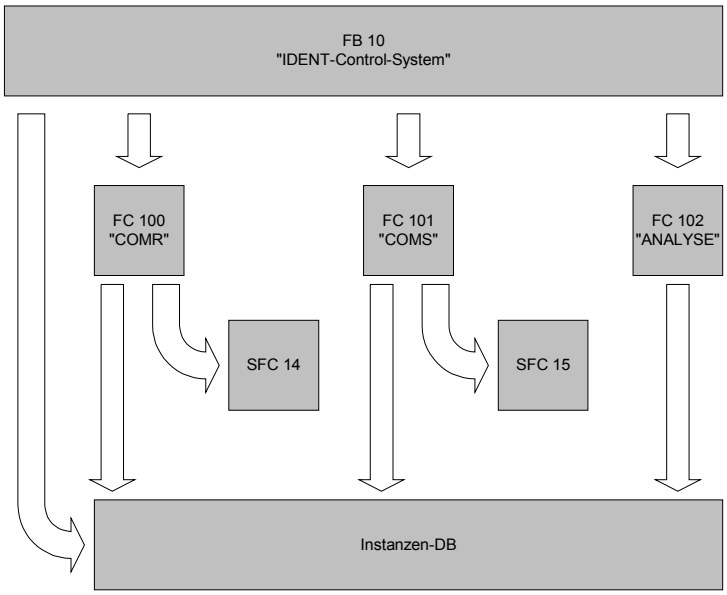

Abb. 10: Struktur des Anwenderprogramms

Der gesamte Ablauf wird vom FB 10 gesteuert und kontrolliert. Über die FC 100 werden die Daten von der IDENT-Control unter Verwendung der SFC 14 abgeholt und dann von der FC 102 ausgewertet. Zum Senden eines Befehls wird die Funktion FC 101 verwendet, die die Daten mittels der SFC 15 an die IDENT-Control überträgt. Alle Daten, die eine IDENT-Control-Einheit betreffen, werden im Instanzen-DB dieser Einheit abgespeichert. Pro IDENT-Control-Einheit ist ein Instanzen-DB erforderlich. Die restlichen Bausteine des Programms sind multiinstanzfähig, d.h. sie können für alle in diesem Projekt benutzten IDENT-Control-Einheiten verwendet werden und brauchen nicht vervielfältigt zu werden.

5.2 Die verwendeten Bits und deren Bedeutung

In der folgenden Übersicht werden alle im Beispielprogramm verwendeten Bits und deren Bedeutung aufgezeigt.

- *TimeoutÜberwAktiv*

  Dieses Bit zeigt an, dass bei den Single-Befehlen der Timer zur Timeoutüberwachung gestartet wurde und dieser auch überwacht wird.

- *BefehlNeuSenden*

  Wenn der gleiche Befehl nochmals an den gleichen Kopf gesendet werden soll, so stehen der Befehlscode und die erforderlichen Parameter noch im Ausgangsdatenfeld dieses Kopfes im Datenbaustein. Der Befehl braucht also nicht komplett neu geladen zu werden, sondern kann direkt nochmal an den Kopf übertragen werden. Durch Setzen des BefehlNeuSenden-Bits wird die Befehlladeroutine übersprungen und direkt der Programmteil aufgerufen, der den Befehl an den Kopf überträgt (exe-Teil des betreffenden Kopfes). Die BefehlNeuSenden-Funktionalität wird im Beispielprogramm nur für Single-Befehle verwendet.

- *InvalidResponse*

  Dieses Bit wird gesetzt, wenn von der IDENT-Control eine ungültige, d.h. in Befehlscode oder Parametern nicht zum gesendeten Befehl passende Antwort des entsprechenden Kopfes empfangen wurde.

- *QuitError*

  Dieses interne Bit dient zum Quittieren eines Fehlers. Ein Fehlerzustand ist am gesetzten Error-Bit erkennbar.

- *NeueDatenVorhanden*

  Hierbei handelt es sich um ein Programm-intern verwendetes Bit, das gesetzt wird, wenn neue Daten von der IDENT-Control empfangen wurden. Damit erkennt der Baustein ANALYSE bei seinem Aufruf, dass neue Daten vorliegen und beginnt mit deren Auswertung.

- *NichtVorhanden*

  Auch dieses Bit ist ein rein Programm-intern verwendetes Bit. Es wird bei der Initialisierung gesetzt, wenn der Kopf nicht vorhanden ist.

- *VorhandenTC**

  Dieses Bit erhält den Wert „true“, wenn die Initialisierung des Kopfes erfolgreich durchgeführt wurde, d.h. der Kopf erkannt (=“Vorhanden“) und der ChangeTag-Befehl von dem Kopf angenommen und ausgeführt wurde („TC“ = Tag Changed).

- *Error**

  Wie der Name schon vermuten lässt, handelt es sich hierbei um ein Fehlermeldebit. Es dient ebenfalls als Verriegelung, d.h. solange dieses Bit gesetzt ist, wird kein weiterer Befehl an dem betreffenden Kopf ausgeführt. Rückgesetzt wird es mittels der QuitError-Routine, die durch Setzen des entsprechenden QuitError-Bits gestartet wird.

- *TimeoutOccured**

  Das TimeoutOccured-Bit wird gesetzt, wenn ein Timeout-Timer abgelaufen ist, ohne dass eine Antwort auf den gesendeten Befehl von der IDENT-Control empfangen wurde.

- *EmpfangenOK*

  Dieses Bit zeigt an, dass eine Antwort der IDENT-Control korrekt empfangen wurde.

- *SendenOK*

  Das SendenOK-Bit signalisiert, dass der Befehl korrekt und vollständig zur IDENT-Control übertragen wurde.

- *KeinDatenträger**

  Wird ein Schreib- oder Lesebefehl ausgeführt und es befindet sich kein Datenträger im Erfassungsbereich des Kopfes, so wird dies dem Anwender mit diesem Bit gemeldet.

- *Done**

  Bei dem Done-Bit handelt es sich um ein Signalisierungsbit, das immer dann gesetzt ist, wenn neue, von der Funktion ANALYSE ausgewertete Daten für den Anwender bereitstehen.

- *Busy**[*]

  Wird ein Befehl gerade von einem Kopf bearbeitet, so wird dieses Bit in der Steuerung gesetzt. Es verhindert, dass während der Bearbeitung dieses Befehls bereits wieder ein neuer an den Kopf geschickt wird und somit die Bearbeitung des ersten Befehls abgebrochen würde.

- *Error_SFC_14*

  Wenn bei der Ausführung der Systemfunktion SFC 14 ein Fehler auftritt (der Rückgabewert also ungleich 0 ist), wird dieses Bit gesetzt.

- *Error_SFC_15*

  Analog zum Error_SFC_14-Bit wird dieses Bit gesetzt, wenn bei der Ausführung der Systemfunktion SFC 15 ein Fehler aufgetreten ist.

- *EnhBefehlAktiv*

  Dieses Bit zeigt an, dass an dem betreffenden Kopf gerade ein Enhanced-Befehl aktiv ist, der im Beispielprogramm nur mit dem Quit-Befehl beendet werden kann.

- *SglBefehlAktiv*

  Bei diesem Bit handelt es sich ebenfalls um ein Signalisierungs- und Sperr-Bit, das gesetzt ist, wenn ein Single-Befehl an diesem Kopf aktiv ist.

- *ErstAntwortErhalten*

  Wie im Kapitel „Die Bedeutung des Togglebits“ erläutert, soll das Togglebit nach Erhalt der ersten Antwort auf einen Befehl ausmaskiert werden. Um dies zu erreichen wird dieses Bit verwendet. Bei der ersten Antwort auf einen Befehl wird es gesetzt, beim Senden eines neuen Befehl an diesen Kopf zurückgesetzt.

- *Restart*

  Dieses Bit wird intern verwendet und dient dazu bei einem Neustart, also noch vor der Neuinitialisierung, die entsprechenden Zustandssignalisierungsbits der einzelnen Köpfe zurücksetzen zu können.

- *Freigegeben*

  Dieses Bit ist gesetzt, wenn die Initialisierung erfolgreich durchgeführt wurde. Wird es vom Anwender zurückgesetzt, so wird automatisch das Restart-Bit gesetzt und eine Neuinitialisierung eingeleitet.

[*] Alle mit einem Sternchen (*) versehenen Bits werden dem Anwender mittels den kopfspezifischen Rückgabebytes des Beispielprogramms zugänglich gemacht.

- *TransfAnKopf1..TransfAnKopf4*

  Diese Bits dienen zur Anzeige, dass ein neuer Befehlscode mit Parametern in das Ausgangsdatenfeld des Kopfes geladen und übertragen wurde, jedoch noch keine Antwort der IDENT-Control erhalten wurde.

Auf alle diese Bits (mit Ausnahme des Freigegeben-Bits) hat der Anwender von außen keinen Steuerungszugriff, d.h. diese Bits werden ausschließlich Programm-intern verwaltet, können jedoch angezeigt werden. Zur Entgegennahme von Befehlen des Anwenders dienen die folgenden Bits:

- *DatenFixcode*

  Mit diesem Bit wird ausgewählt, ob Daten oder Fixcode gelesen werden soll. Dabei bedeutet „0“ Daten und „1“ Fixcode.

- *SingleEnhanced*

  Ebenso verhält es sich mit dem SingleEnhanced-Bit. Hier wird eine Auswahl getroffen, ob ein Single- oder ein Enhanced-Befehl ausgeführt werden soll. „0“ bedeutet Single-Befehl, „1“ Enhanced-Befehl.

- *Kopf1Lesen .. Kopf4Lesen*

  Diese Bits dienen zum Starten eines Lesebefehls an dem betreffenden Kopf.

- *Kopf1Schreiben .. Kopf4Schreiben*

  Diese Bits dienen zum Starten eines Schreibbefehls an dem betreffenden Kopf.

- *Kopf1Quit .. Kopf4Quit*

  Durch Setzen eines dieser Bits wird der Quit-Befehl an den betreffenden Kopf gesendet.

- *QuitErrorKopf1 .. QuitErrorKopf4*

  Mit diesen Bits wird die Routine zur kopfbezogenen Quittierung von Fehlern aufgerufen.

5.3 Erforderliche Deklarationen und Definitionen

Damit alle diese Bits nun auch mit möglichst geringem Aufwand für jeden Kopf in den Datenbaustein implementiert werden können, besteht die Möglichkeit, die Deklaration mittels eines UDTs[1] vorzunehmen. Bei einem UDT-Baustein handelt es sich um einen Struktur-Deklarationsbaustein, der nach seiner Erstellung im Deklarationsteil der FBs und FCs einfach aufgerufen werden kann und die komplette Struktur damit eingebaut bzw. deklariert ist.

Wie die Erstellung neuer Bausteine in Step7 funktioniert, wird an dieser Stelle als bekannt vorausgesetzt. Nachzulesen ist dies im Handbuch „AWL für S7-300/400 – Bausteine programmieren“ der Fa. Siemens.

5.3.1 Der UDT 100 („Kopf_Datenstruktur“)

Mit dem UDT 100 werden alle für einen Schreib-/Lesekopf erforderlichen Bits und Datenfelder in der erforderlichen Struktur deklariert. Die folgende Tabelle zeigt die Struktur des UDT 100. Als symbolischen Namen erhält der UDT 100 die Benennung „Kopf_Datenstruktur“, die dann bei der späteren Verwendung des UDTs in den Deklarationsteilen der Bausteine angezeigt wird.

Adresse	Name	Typ	Anfangswert	Kommentar
0.0	STRUCT			
+0.0	Eingangsdaten	STRUCT		
+0.0	Befehlscode_Eing	BYTE	B#16#0	
+1.0	WortAnzTog_Eing	BYTE	B#16#0	
+2.0	Status	BYTE	B#16#0	
+3.0	Ausfzaehler	BYTE	B#16#0	
+4.0	Eing_DW1	DWORD	DW#16#0	
+8.0	Eing_DW2	DWORD	DW#16#0	
+12.0	Eing_DW3	DWORD	DW#16#0	
+16.0	Eing_DW4	DWORD	DW#16#0	
+20.0	Eing_DW5	DWORD	DW#16#0	
+24.0	Eing_DW6	DWORD	DW#16#0	
+28.0	Eing_DW7	DWORD	DW#16#0	
=32.0	END_STRUCT			
+32.0	Ausgangsdaten	STRUCT		
+0.0	Befehlscode_Ausg	BYTE	B#16#0	
+1.0	WortAnzTog_Ausg	BYTE	B#16#0	
+2.0	Wortadr_High	BYTE	B#16#0	
+3.0	Wortadr_Low	BYTE	B#16#0	
+4.0	Ausg_DW1	DWORD	DW#16#0	
+8.0	Ausg_DW2	DWORD	DW#16#0	
+12.0	Ausg_DW3	DWORD	DW#16#0	
+16.0	Ausg_DW4	DWORD	DW#16#0	
+20.0	Ausg_DW5	DWORD	DW#16#0	
+24.0	Ausg_DW6	DWORD	DW#16#0	

[1] UDT = user defined data type, Anwenderdefinierter Datentyp

+28.0	Ausg_DW7	DWORD	DW#16#0	
=32.0	END_STRUCT			
+64.0	AktAnzWdh	INT	0	
+66.0	MaxAnzWdh	INT	0	
+68.0	TimerFuerTimeout	S5TIME	S5T#0MS	
+70.0	TimeoutUeberwAktiv	BOOL	FALSE	
+70.1	BefehlNeuSenden	BOOL	FALSE	
+70.2	ExitTimeout	BOOL	FALSE	
+70.3	RestartTimeout	BOOL	FALSE	
+70.4	InvalidResponse	BOOL	FALSE	
+70.5	QuitError	BOOL	FALSE	
+70.6	NeueDatenVorhanden	BOOL	FALSE	
+70.7	NichtVorhanden	BOOL	FALSE	
+71.0	reserviert1	BYTE	B#16#0	
+72.0	VorhandenTC	BOOL	FALSE	
+72.1	Error	BOOL	FALSE	
+72.2	TimeoutOccured	BOOL	FALSE	
+72.3	EmpfangenOK	BOOL	FALSE	
+72.4	SendenOK	BOOL	FALSE	
+72.5	KeinDatentraeger	BOOL	FALSE	
+72.6	Done	BOOL	FALSE	
+72.7	Busy	BOOL	FALSE	
+73.0	Error_SFC_14	BOOL	FALSE	
+73.1	Error_SFC_15	BOOL	FALSE	
+73.2	EnhBefehlAktiv	BOOL	FALSE	
+73.3	SglBefehlAktiv	BOOL	FALSE	
+73.4	ErstAntwortErhalten	BOOL	FALSE	
+74.0	reserviert3	BYTE	B#16#0	
+75.0	reserviert4	BYTE	B#16#0	
+76.0	Ret_Val_SFC14	WORD	W#16#0	
+78.0	Ret_Val_SFC15	WORD	W#16#0	
+80.0	reserviert5	BYTE	B#16#0	
+81.0	Alt_Befehl_Eing	BYTE	B#16#0	
+82.0	Alt_WortAnzTog_Eing	BYTE	B#16#0	
+83.0	Alt_Status	BYTE	B#16#0	
+84.0	Alt_Ausfzaehler	BYTE	B#16#0	
+85.0	reserviert6	BYTE	B#16#0	
+86.0	reserviert7	BYTE	B#16#0	
+87.0	reserviert8	BYTE	B#16#0	
+88.0	reserviert9	BYTE	B#16#0	
+89.0	reserviert10	BYTE	B#16#0	
=90.0	END_STRUCT			

Tabelle 3: Struktur des UDT 100

5.3.2 Der UDT 101 („Byte_Puffer“)

Im UDT 101 wird eine Byte-Struktur definiert, die für den Pufferspeicher bei der Ausführung des SFC 14 benötigt wird. Der symbolische Name des UDT 101 ist „Byte_Puffer“. In Tabelle 4 ist die Struktur des Bausteins dargestellt.

Adresse	Name	Typ	Anfangswert	Kommentar
0.0	STRUCT			
+0.0	Befehlscode	BYTE	B#16#0	
+1.0	WortAnzTog	BYTE	B#16#0	
+2.0	Status	BYTE	B#16#0	
+3.0	Ausfzaehler	BYTE	B#16#0	
+4.0	Datenbyte1	BYTE	B#16#0	
+5.0	Datenbyte2	BYTE	B#16#0	
+6.0	Datenbyte3	BYTE	B#16#0	
+7.0	Datenbyte4	BYTE	B#16#0	
+8.0	Datenbyte5	BYTE	B#16#0	
+9.0	Datenbyte6	BYTE	B#16#0	
+10.0	Datenbyte7	BYTE	B#16#0	
+11.0	Datenbyte8	BYTE	B#16#0	
+12.0	Datenbyte9	BYTE	B#16#0	
+13.0	Datenbyte10	BYTE	B#16#0	
+14.0	Datenbyte11	BYTE	B#16#0	
+15.0	Datenbyte12	BYTE	B#16#0	
+16.0	Datenbyte13	BYTE	B#16#0	
+17.0	Datenbyte14	BYTE	B#16#0	
+18.0	Datenbyte15	BYTE	B#16#0	
+19.0	Datenbyte16	BYTE	B#16#0	
+20.0	Datenbyte17	BYTE	B#16#0	
+21.0	Datenbyte18	BYTE	B#16#0	
+22.0	Datenbyte19	BYTE	B#16#0	
+23.0	Datenbyte20	BYTE	B#16#0	
+24.0	Datenbyte21	BYTE	B#16#0	
+25.0	Datenbyte22	BYTE	B#16#0	
+26.0	Datenbyte23	BYTE	B#16#0	
+27.0	Datenbyte24	BYTE	B#16#0	
+28.0	Datenbyte25	BYTE	B#16#0	
+29.0	Datenbyte26	BYTE	B#16#0	
+30.0	Datenbyte27	BYTE	B#16#0	
+31.0	Datenbyte28	BYTE	B#16#0	
=32.0	END_STRUCT			

Tabelle 4: Struktur des UDT 101

5.3.3 Der Deklarationsteil des Bausteins „Ident-Control-System“ (FB10)

Beim späteren Aufruf des FB10 muss ein Instanzen-DB mitangegeben werden, in dem die zu dieser IDENT-Control-Einheit gehörenden Daten abgespeichert werden. Die Erstellung des DB erfolgt dabei automatisch aus dem Deklarationsteil des FB10. In Tabelle 5 ist die zu erstellende Struktur im Deklarationsteil des FB10 aufgezeigt.

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Adresse_Ident_Control	WORD	W#16#0	
2.0	in	Datenbaustein	INT	0	
4.0	in	MaxAnzWdh	INT	0	
6.0	in	TimeoutDauer	S5TIME	S5T#0MS	
8.0	in	TimeoutTimerKopf1	TIMER		
10.0	in	TimeoutTimerKopf2	TIMER		
12.0	in	TimeoutTimerKopf3	TIMER		
14.0	in	TimeoutTimerKopf4	TIMER		
16.0	in	Datentraegertyp	WORD	W#16#0	
18.0	in	DatenFixcode	BOOL	FALSE	
18.1	in	SingleEnhanced	BOOL	FALSE	
18.2	in	Kopf1Lesen	BOOL	FALSE	
18.3	in	Kopf1Schreiben	BOOL	FALSE	
18.4	in	Kopf2Lesen	BOOL	FALSE	
18.5	in	Kopf2Schreiben	BOOL	FALSE	
18.6	in	Kopf3Lesen	BOOL	FALSE	
18.7	in	Kopf3Schreiben	BOOL	FALSE	
19.0	in	Kopf4Lesen	BOOL	FALSE	
19.1	in	Kopf4Schreiben	BOOL	FALSE	
19.2	in	Kopf1Quit	BOOL	FALSE	
19.3	in	Kopf2Quit	BOOL	FALSE	
19.4	in	Kopf3Quit	BOOL	FALSE	
19.5	in	Kopf4Quit	BOOL	FALSE	
19.6	in	QuitErrorKopf1	BOOL	FALSE	
19.7	in	QuitErrorKopf2	BOOL	FALSE	
20.0	in	QuitErrorKopf3	BOOL	FALSE	
20.1	in	QuitErrorKopf4	BOOL	FALSE	
22.0	out	RueckgabewertKopf1	BYTE	B#16#0	
23.0	out	RueckgabewertKopf2	BYTE	B#16#0	
24.0	out	RueckgabewertKopf3	BYTE	B#16#0	
25.0	out	RueckgabewertKopf4	BYTE	B#16#0	
26.0	in_out	Freigegeben	BOOL	FALSE	
28.0	stat	reserviert100	DWORD	DW#16#0	
32.0	stat	reserviert101	DWORD	DW#16#0	
36.0	stat	reserviert102	DWORD	DW#16#0	
40.0	stat	reserviert103	DWORD	DW#16#0	
44.0	stat	reserviert104	DWORD	DW#16#0	
48.0	stat	reserviert105	DWORD	DW#16#0	
52.0	stat	reserviert106	DWORD	DW#16#0	
56.0	stat	reserviert107	DWORD	DW#16#0	
60.0	stat	Kopf_1	"Kopf_Datenstruktur"		
150.0	stat	Kopf_2	"Kopf_Datenstruktur"		
240.0	stat	Kopf_3	"Kopf_Datenstruktur"		
330.0	stat	Kopf_4	"Kopf_Datenstruktur"		
420.0	stat	Zwischenspeicher	"Kopf_Datenstruktur"		
510.0	stat	Empf_Kopfnummer	INT	0	
512.0	stat	Fehler_Aufgetreten	BOOL	FALSE	
512.1	stat	Quit	BOOL	FALSE	
514.0	stat	TimerFuerTimeout	S5TIME	S5T#0MS	
516.0	stat	Timeout	BOOL	FALSE	
516.1	stat	NeueDatenVorhanden	BOOL	FALSE	
518.0	stat	Ret_Val_SFC14	WORD	W#16#0	
520.0	stat	Ret_Val_SFC15	WORD	W#16#0	
522.0	stat	Reset	BOOL	FALSE	
522.1	stat	reserviert108	BOOL	FALSE	
524.0	stat	reserviert109	WORD	W#16#0	
526.0	stat	LesenKopf1	BOOL	FALSE	
526.1	stat	LesenKopf2	BOOL	FALSE	
526.2	stat	LesenKopf3	BOOL	FALSE	
526.3	stat	LesenKopf4	BOOL	FALSE	
526.4	stat	SchreibenKopf1	BOOL	FALSE	

526.5	stat	SchreibenKopf2	BOOL	FALSE	
526.6	stat	SchreibenKopf3	BOOL	FALSE	
526.7	stat	SchreibenKopf4	BOOL	FALSE	
527.0	stat	TransfAnKopf1	BOOL	FALSE	
527.1	stat	TransfAnKopf2	BOOL	FALSE	
527.2	stat	TransfAnKopf3	BOOL	FALSE	
527.3	stat	TransfAnKopf4	BOOL	FALSE	
527.4	stat	reserviert110	BOOL	FALSE	
527.5	stat	reserviert111	BOOL	FALSE	
527.6	stat	reserviert112	BOOL	FALSE	
527.7	stat	reserviert113	BOOL	FALSE	
528.0	stat	EnhLesenKopf1	BOOL	FALSE	
528.1	stat	EnhLesenKopf2	BOOL	FALSE	
528.2	stat	EnhLesenKopf3	BOOL	FALSE	
528.3	stat	EnhLesenKopf4	BOOL	FALSE	
528.4	stat	EnhSchreibenKopf1	BOOL	FALSE	
528.5	stat	EnhSchreibenKopf2	BOOL	FALSE	
528.6	stat	EnhSchreibenKopf3	BOOL	FALSE	
528.7	stat	EnhSchreibenKopf4	BOOL	FALSE	
529.0	stat	Empf Togglebit	BYTE	B#16#0	
530.0	stat	Start	BOOL	FALSE	
530.1	stat	reserviert114	BOOL	FALSE	
530.2	stat	Restart	BOOL	FALSE	
530.3	stat	reserviert115	BOOL	FALSE	
530.4	stat	reserviert116	BOOL	FALSE	
530.5	stat	reserviert117	BOOL	FALSE	
530.6	stat	reserviert118	BOOL	FALSE	
530.7	stat	reserviert119	BOOL	FALSE	
531.0	stat	reserviert120	BOOL	FALSE	
531.1	stat	reserviert121	BOOL	FALSE	
531.2	stat	reserviert122	BOOL	FALSE	
531.3	stat	reserviert123	BOOL	FALSE	
531.4	stat	reserviert124	BOOL	FALSE	
531.5	stat	reserviert125	BOOL	FALSE	
531.6	stat	reserviert126	BOOL	FALSE	
531.7	stat	reserviert127	BOOL	FALSE	
532.0	stat	QuitKopf1	BOOL	FALSE	
532.1	stat	QuitKopf2	BOOL	FALSE	
532.2	stat	QuitKopf3	BOOL	FALSE	
532.3	stat	QuitKopf4	BOOL	FALSE	
534.0	stat	reserviert128	"Kopf Datenstruktur"		
624.0	stat	MerkerLesenKopf1	BOOL	FALSE	
624.1	stat	MerkerLesenKopf2	BOOL	FALSE	
624.2	stat	MerkerLesenKopf3	BOOL	FALSE	
624.3	stat	MerkerLesenKopf4	BOOL	FALSE	
624.4	stat	MerkerSchreibenKopf1	BOOL	FALSE	
624.5	stat	MerkerSchreibenKopf2	BOOL	FALSE	
624.6	stat	MerkerSchreibenKopf3	BOOL	FALSE	
624.7	stat	MerkerSchreibenKopf4	BOOL	FALSE	
625.0	stat	MerkerEnhLesenKopf1	BOOL	FALSE	
625.1	stat	MerkerEnhLesenKopf2	BOOL	FALSE	
625.2	stat	MerkerEnhLesenKopf3	BOOL	FALSE	
625.3	stat	MerkerEnhLesenKopf4	BOOL	FALSE	
625.4	stat	MerkerEnhSchreibKopf1	BOOL	FALSE	
625.5	stat	MerkerEnhSchreibKopf2	BOOL	FALSE	
625.6	stat	MerkerEnhSchreibKopf3	BOOL	FALSE	
625.7	stat	MerkerEnhSchreibKopf4	BOOL	FALSE	
626.0	stat	MerkerQuitKopf1	BOOL	FALSE	
626.1	stat	MerkerQuitKopf2	BOOL	FALSE	
626.2	stat	MerkerQuitKopf3	BOOL	FALSE	
626.3	stat	MerkerQuitKopf4	BOOL	FALSE	
626.4	stat	MerkerQuitErrorKopf1	BOOL	FALSE	
626.5	stat	MerkerQuitErrorKopf2	BOOL	FALSE	
626.6	stat	MerkerQuitErrorKopf3	BOOL	FALSE	
626.7	stat	MerkerQuitErrorKopf4	BOOL	FALSE	
0.0	temp	zugeord_DB	INT		

Tabelle 5: Der Deklarationsteil des FB10

Nachfolgend wird nun der Programmablauf anhand der Ablaufpläne der einzelnen Bausteine detailliert dargestellt und erläutert. Beim Erklären des Ablaufs wird auch gleich der dazugehö-

rige Programmcode angegeben. Dabei werden auch die Schnittstellen und die zu übergebenden Parameter zwischen den Bausteinen aufgezeigt.

Alle unterstrichenen Begriffe in den folgenden Ablaufplänen deuten darauf hin, dass es sich dabei um ein Unterprogramm bzw. um einen Programmteil handelt, auf das bzw. auf den in einem späteren Kapitel detailliert eingegangen wird. Ein entsprechender Verweis auf das betreffende Kapitel ist jeweils angegeben.

5.4 Programmablauf

5.4.1 Ablauf des Bausteins „Ident-Control-System“ (FB 10)

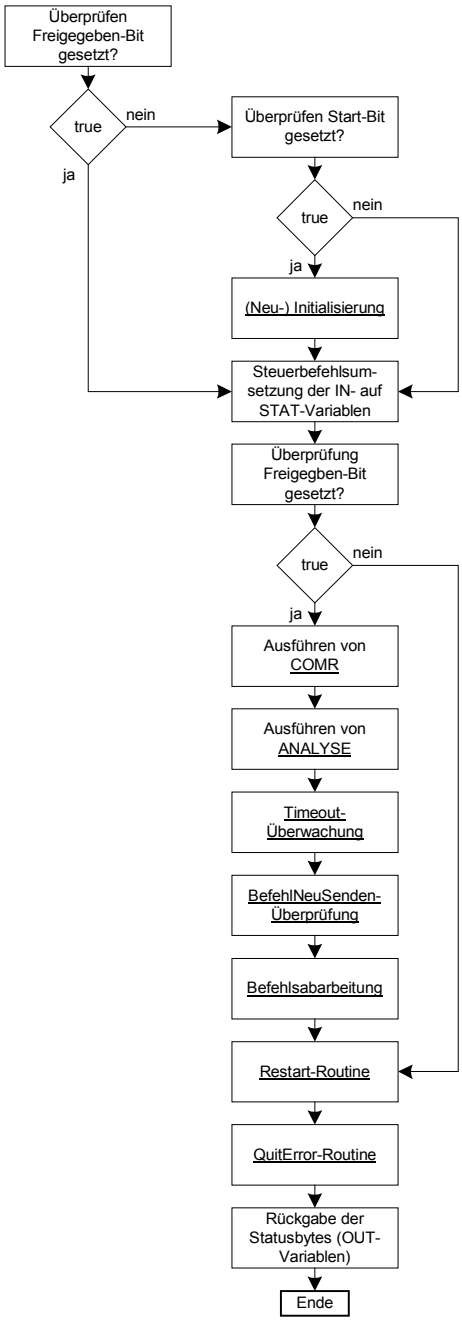

Abb. 11: Ablaufplan des Bausteins „Ident-Control-System“ (FB10)

Der Baustein FB10 ist wie bereits vorher angedeutet das Herzstück des Programms. Er steuert den gesamten Programmablauf.

Im Programmcode wird zuerst die vom Anwender als Parameter übergebene Nummer des zugeordneten DBs auf eine Temp-Variable umkopiert und anschließend der betreffende Datenbaustein geöffnet. Damit muss bei allen Zugriffen auf Bits, Bytes, etc. dieses Datenbausteins nicht mehr der Baustein selbst mitangegeben werden, sondern es erfolgt automatisch der Zugriff auf den jetzt geöffneten Datenbaustein.
Dann wird überprüft, ob das Freigegeben-Bit bereits gesetzt ist. Ist dies der Fall, so wurde die Initialisierung schon erfolgreich durchgeführt und es geht direkt weiter mit dem Programmteil „Steuerbefehlsumsetzung der IN- auf STAT-Variablen“.
Ist das Freigegeben-Bit nicht gesetzt, so muss überprüft werden, ob das Start-Bit, welches die Initialisierung einleitet, gesetzt ist. Ist dieses nicht gesetzt, so erfolgt ebenfalls ein Sprung zum Programmteil „Steuerbefehlsumsetzung“.

```
FB10, Netzwerk 1: Start-Up-Sequence

      L     #Datenbaustein
      T     #zugeord_DB
      AUF   DB [#zugeord_DB]

      SET
      U     #Freigegeben
      SPB   end                    // Sprung zu „Steuerbefehlsumsetzung“ Netzwerk 6

      UN    #Start
      SPB   end                    // Sprung zu „Steuerbefehlsumsetzung“ Netzwerk 6
```

Hat das Start-Bit den Wert „true“, so wird geprüft, ob bereits von jedem Kopf eine Antwort auf den Initialisierungsbefehl erhalten wurde. Erkennbar ist dies, wenn entweder das VorhandenTC- oder das NichtVorhanden-Bit gesetzt ist. Wurde von jedem Kopf bereits eine Antwort erhalten, wird das Freigegeben-Bit gesetzt, das Start-Bit zurückgesetzt und an das Ende der Initialisierung gesprungen. Andernfalls werden das Freigegeben- und das Start-Bit nicht verändert und es geht mit der Initialisierungsroutine weiter.

```
FB10, Netzwerk 1, Fortsetzung

      SET
      U(
      U     #Kopf_1.VorhandenTC
      O     #Kopf_1.NichtVorhanden
      )
      U(
      U     #Kopf_2.VorhandenTC
      O     #Kopf_2.NichtVorhanden
      )
      U(
      U     #Kopf_3.VorhandenTC
      O     #Kopf_3.NichtVorhanden
      )
      U(
      U     #Kopf_4.VorhandenTC
      O     #Kopf_4.NichtVorhanden
      )
      =     #Freigegeben

      SET
      U     #Freigegeben
      R     #Start
      SPB end
```

Der genaue Ablauf der Initialisierung wird im Kapitel „Ablauf des Programmteils (Neu)-Initialisierung“ detailliert erläutert (s. Ziff. 5.4.2).

Im darauffolgenden Programmteil „Steuerbefehlsumsetzung“ wird zunächst überprüft, ob das Freigegeben-Bit gesetzt und das Start-Bit nicht gesetzt ist. Bei dieser Konstellation wird das Restart-Bit gesetzt und zur Restart-Routine gesprungen. Dieser Vorgang ist die automatische Einleitung der Initialisierung beim Start des Programms und auf Befehl des Anwenders (durch explizites Rücksetzen des Freigegeben-Bits).

Ist diese Konstellation nicht gegeben, wird das Restart-Bit nicht gesetzt und es wird überprüft, ob das Freigegeben-Bit gesetzt ist. Wenn nicht, erfolgt ein Sprung zur Restart-Routine. Ist dieses gesetzt, geht es mit der eigentlichen Befehlsumsetzung weiter.

```
FB10, Netzwerk 6: Steuerbefehlsumsetzung

      SET
      UN    #Freigegeben
      UN    #Start
      S     #Restart
      SPB   end1                  // Sprung zur Restart-Routine

      UN    #Freigegeben
      SPB   end1                  // Sprung zur Restart-Routine
```

Das Starten oder Auslösen eines Befehls wird vom Anwender in dessen Anlagensteuerung meist durch Setzen eines Eingangs, Merkers, o.ä. an der SPS vorgenommen. Man könnte nun diese Signale direkt zur Befehlsausführung verwenden. Steuert der Anwender die Befehle über Merker, wäre dies auch problemlos möglich, da Merker nach erfolgter einmaliger Be-

fehlsausführung in der SPS zurücksetzbar sind. Bei der Steuerung über Sensoren o.ä., also über SPS-Eingänge, besteht jedoch die folgende Problematik: z.B. ein Sensor liefert so lange ein High-Signal an den Eingang der SPS wie er bedämpft ist, d.h. das zu lesende oder zu schreibende Objekt sich vor dem Sensor befindet. Man kann also einen Eingang in der SPS im Programmablauf nicht zurücksetzen. Der Befehl würde somit mehr als einmal ausgeführt - nämlich genau so lange wie das High-Signal am Eingang anliegt. Abhilfe schafft hier eine Flankensteuerung. Dabei ist es dann egal, ob die im Programm verwendete steigende Flanke von Merkern oder Eingängen kommt. Eine solche flankengesteuerte Umsetzung der von extern gegebenen Signale auf die internen Bits, die den betreffenden Befehl auslösen, erfolgt im weiteren Verlauf dieses Programmteils.

```
FB10, Netzwerk 6, Fortsetzung

      SET
      UN    #SingleEnhanced        // IN-Variable
      U     #Kopf1Lesen            // IN-Variable
      FP    #MerkerLesenKopf1      // Flankenmerker
      S     #LesenKopf1            // Internes befehlauslösendes Bit

      SET
      UN    #SingleEnhanced
      U     #Kopf2Lesen
      FP    #MerkerLesenKopf2
      S     #LesenKopf2

      SET
      UN    #SingleEnhanced
      U     #Kopf3Lesen
      FP    #MerkerLesenKopf3
      S     #LesenKopf3

      SET
      UN    #SingleEnhanced
      U     #Kopf4Lesen
      FP    #MerkerLesenKopf4
      S     #LesenKopf4

      SET
      UN    #SingleEnhanced
      U     #Kopf1Schreiben
      FP    #MerkerSchreibenKopf1
      S     #SchreibenKopf1

      SET
      UN    #SingleEnhanced
      U     #Kopf2Schreiben
      FP    #MerkerSchreibenKopf2
      S     #SchreibenKopf2

      SET
      UN    #SingleEnhanced
      U     #Kopf3Schreiben
      FP    #MerkerSchreibenKopf3
      S     #SchreibenKopf3

      SET
      UN    #SingleEnhanced
      U     #Kopf4Schreiben
      FP    #MerkerSchreibenKopf4
      S     #SchreibenKopf4
```

```
SET
U     #Kopf1Quit
FP    #MerkerQuitKopf1
S     #QuitKopf1

SET
U     #Kopf2Quit
FP    #MerkerQuitKopf2
S     #QuitKopf2

SET
U     #Kopf3Quit
FP    #MerkerQuitKopf3
S     #QuitKopf3

SET
U     #Kopf4Quit
FP    #MerkerQuitKopf4
S     #QuitKopf4

SET
U     #SingleEnhanced
U     #Kopf1Lesen
FP    #MerkerEnhLesenKopf1
S     #EnhLesenKopf1

SET
U     #SingleEnhanced
U     #Kopf2Lesen
FP    #MerkerEnhLesenKopf2
S     #EnhLesenKopf2

SET
U     #SingleEnhanced
U     #Kopf3Lesen
FP    #MerkerEnhLesenKopf3
S     #EnhLesenKopf3

SET
U     #SingleEnhanced
U     #Kopf4Lesen
FP    #MerkerEnhLesenKopf4
S     #EnhLesenKopf4

SET
U     #SingleEnhanced
U     #Kopf1Schreiben
FP    #MerkerEnhSchreibKopf1
S     #EnhSchreibenKopf1

SET
U     #SingleEnhanced
U     #Kopf2Schreiben
FP    #MerkerEnhSchreibKopf2
S     #EnhSchreibenKopf2

SET
U     #SingleEnhanced
U     #Kopf3Schreiben
FP    #MerkerEnhSchreibKopf3
S     #EnhSchreibenKopf3

SET
U     #SingleEnhanced
U     #Kopf4Schreiben
FP    #MerkerEnhSchreibKopf4
S     #EnhSchreibenKopf4

SET
U     #QuitErrorKopf1
FP    #MerkerQuitErrorKopf1
S     #Kopf_1.QuitError
```

```
SET
U     #QuitErrorKopf2
FP    #MerkerQuitErrorKopf2
S     #Kopf_2.QuitError

SET
U     #QuitErrorKopf3
FP    #MerkerQuitErrorKopf3
S     #Kopf_3.QuitError

SET
U     #QuitErrorKopf4
FP    #MerkerQuitErrorKopf4
S     #Kopf_4.QuitError
```

Anschließend werden durch Ausführen der Funktion COMR Daten von der IDENT-Control abgeholt. Danach wird die Funktion ANALYSE zur Auswertung der empfangenen Daten aufgerufen.

```
FB10, Netzwerk 7: Ausführen von COMR und ANALYSE

      CALL  "COMR"
       Adresse      :=#Adresse_Ident_Control
       Datenbaustein:=#Datenbaustein

      CALL  "ANALYSE"
       Datenbaustein:=#Datenbaustein
```

Dann wird der Befehlsteil Timeout-Überwachung durchlaufen. Genaueres zum Ablauf ist in Kapitel „Ablauf des Programmteils Timeout-Überwachung“ beschrieben (s. Ziff. 5.4.9).
Im folgenden Programmteil werden die BefehlNeuSenden-Bits auf Gesetztsein überprüft und ggf. ein Sprung direkt zur exe-Einheit des betreffenden Kopfes veranlasst. Dieser Programmpart wird später im Kapitel „Ablauf des Programmteils BefehlNeuSenden“ behandelt.
Der nächste Programmteil umfasst die eigentliche Befehlsbearbeitung. In diesem Teil werden die Befehle mit ihren Parametern in die Ausgangsdatenfelder der einzelnen Köpfe geladen und anschließend an diese übertragen. Der genaue Ablauf wird im Kapitel „Ablauf des Programmteils Befehlsbearbeitung“ erklärt (s. Ziff. 5.4.3).
Die Restart-Routine ist der nächste Part des Programms. Sie dient dazu, eine Neuinitialisierung und/oder eine Umstellung auf einen anderen Datenträgertyp jederzeit durchführen zu können. Details zum Ablauf sind im Kapitel „Ablauf der Restart-Routine“ zu finden (s. Ziff. 5.4.11).
In der QuitError-Routine werden die Quittierungen aufgetretener Fehler behandelt. Auch diese Routine wird später noch im Detail erläutert (Kapitel „Ablauf der QuitError-Routine“, s. Ziff. 5.4.12).

Damit der Anwender den aktuellen Zustand der einzelnen Köpfe sehen und seine Anlage mit den gelesenen/geschriebenen Daten steuern kann, muss eine Rückmeldung über den derzeitigen Status des Identsystems nach außen erfolgen. Dies geschieht über 4 Bytes. Jedes Byte spiegelt den Zustand eines Kopfes wider. Anhand der in diesem Byte enthaltenen Bits lässt sich der Zustand des betreffenden Kopfes eindeutig erkennen und es kann entsprechend gehandelt werden. Diese Bytes werden im letzten Teil des Programms in die Out-Variablen geschrieben und somit dem Anwender zugänglich gemacht.

```
FB10, Netzwerk 21: Rückgabe der Statusbytes

// Rückmeldebytes der Köpfe in OUT-Variablen speichern

      L     DBB  132
      L     B#16#E7                    // EmpfangenOK und SendenOK ausmaskieren
      UW
      T     #RueckgabewertKopf1

      L     DBB  222
      L     B#16#E7
      UW
      T     #RueckgabewertKopf2

      L     DBB  312
      L     B#16#E7
      UW
      T     #RueckgabewertKopf3

      L     DBB  402
      L     B#16#E7
      UW
      T     #RueckgabewertKopf4
```

Damit ist das Ende des Bausteins „Ident-Control-System“ erreicht und es erfolgt ein Rücksprung in das Programm des Anwenders, von dem der Baustein aufgerufen wurde.
Nun werden zunächst die erwähnten unterstrichenen Programmteile des FB 10 im Detail erläutert und anschließend die Abläufe der anderen Bausteine erklärt.

5.4.2 Ablauf des Programmteils (Neu)-Initialisierung

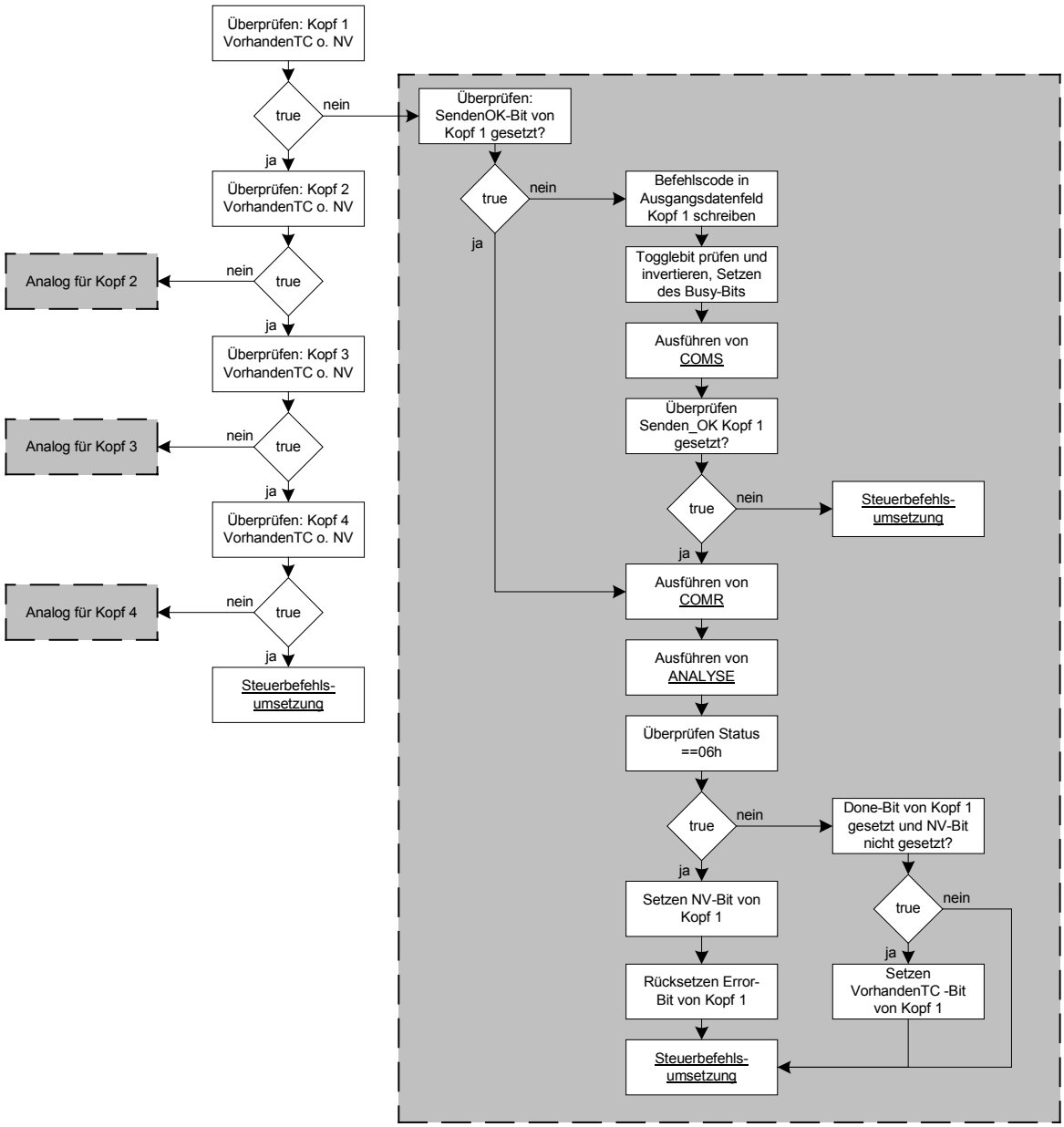

Abb. 12: Ablaufplan der Initialisierung

Im Folgenden wird der erforderliche Code für die Initialisierung von Kopf 1 erklärt. Für alle anderen Köpfe ist dieser analog zu übertragen. Im Programm ist die Initialisierung auf die Netzwerke 2-5 aufgeteilt.

Die Initialisierungsroutine wird nur aufgerufen, wenn noch nicht von jedem Kopf das VorhandenTC oder das NichtVorhanden-Bit gesetzt ist, also noch nicht alle Köpfe initialisiert sind. In diesem Fall muss überprüft werden, welcher Kopf als nächstes zu initialisieren ist. Ist beispielsweise Kopf 1 schon initialisiert, wird zur Initialisierung von Kopf 2 gesprungen (= Netzwerk 3). Dort erfolgt diese Überprüfung für Kopf 2 etc.

```
FB10, Netzwerk 2: Initialisierung von Kopf 1

      U     #Kopf_1.VorhandenTC
      O     #Kopf_1.NichtVorhanden
      SPB   ini2
```

Die eigentliche Initialisierungssequenz ist in Abb. 12 in dem hellgrau hinterlegten Kasten beispielhaft für Kopf 1 dargestellt. Für alle anderen Köpfe erfolgt sie analog.
Zunächst wird überprüft, ob das SendenOK-Bit gesetzt ist. Dieses signalisiert, dass der Befehl bereits an die IDENT-Control übertragen wurde und die SPS nun auf eine Antwort von der IDENT-Control wartet. Ist es gesetzt, geht es sofort mit dem Aufruf des Bausteins COMR zum Empfangen neuer Daten von der IDENT-Control weiter.

```
FB10, Netzwerk 2, Fortsetzung

      SET
      U     #Kopf_1.SendenOK
      SPB   rec1
```

Ist das SendenOK-Bit nicht gesetzt, so werden der Befehlscode für den ChangeTag-Befehl, sowie die erforderlichen Parameter in das Ausgangsdatenfeld von Kopf 1 geladen. Anschließend wird das zuletzt empfangene Togglebit überprüft und invertiert, damit die IDENT-Control den Befehl mit Sicherheit als einen neuen Befehl erkennt und diesen dann auch ausführt.

```
FB10, Netzwerk 2, Fortsetzung

      L     B#16#4
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg
      L     B#16#2
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg
      L     #Datentraegertyp
      L     W#16#FF00
      UW
      SRW   8
      T     #Kopf_1.Ausgangsdaten.Wortadr_High
      L     #Datentraegertyp
      T     #Kopf_1.Ausgangsdaten.Wortadr_Low

      L     #Empf_Togglebit             // Abfrage Status Togglebit
      L     B#16#1
      <>I
```

```
        SPB   cg1

        L     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg    // Falls gesetzt, wird es jetzt
                                                       // rückgesetzt
        UW    W#16#FFFE
        T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg
        SPA   sn1

cg1:    L     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg    // Falls nicht gesetzt, wird es
                                                       // jetzt gesetzt
        OW    W#16#1
        T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg
        SPA   sn1
```

Danach wird das Busy-Bit von Kopf 1 gesetzt und der Baustein COMS aufgerufen, der die Daten an die IDENT-Control überträgt. Bei erfolgreicher Übertragung wird dann das Bit SendenOK gesetzt. Dieses Bit wird anschließend überprüft. Ist es nicht gesetzt, so muss der Befehl nochmal gesendet und der Initialisierungsteil dazu nochmals durchlaufen werden. Deshalb erfolgt in diesem Fall ein Sprung aus der Initialisierungsroutine heraus zum Programmteil „Steuerbefehlsumsetzung“.

```
FB10, Netzwerk 2, Fortsetzung

sn1:  SET
      S     #Kopf_1.Busy
      CALL  "COMS"
       Adresse      :=#Adresse_Ident_Control
       Ausg_Daten   :=#Kopf_1.Ausgangsdaten
       Datenbaustein:=#Datenbaustein

      SET
      U     #Kopf_1.SendenOK
      SPBN  end                 // ggf. Sprung zu Steuerbefehlsumsetzung
```

Ist das SendenOK-Bit gesetzt, wurden die Daten erfolgreich an die IDENT-Control übertragen und es wird auf eine Antwort von der IDENT-Control bzw. von dem angesprochenen Schreib-/Lesekopf gewartet. Somit wird nun der Baustein COMR aufgerufen und die Ausgangsdaten der IDENT-Control abgeholt. Anschließend wird der Baustein ANALYSE aufgerufen und wenn neue Daten vorliegen diese ausgewertet. Sollte der Status in der Antwort der IDENT-Control den Status 06_h enthalten, so bedeutet das, dass zwar eine passende Antwort auf diesen Befehl erhalten wurde, dieser Kopf jedoch nicht vorhanden ist. Entsprechend wird das NichtVorhanden-Bit gesetzt. Das Error-Bit, das vom Baustein ANALYSE aus im Kapitel „Ablauf des Bausteins ANALYSE (FC 102)“ erläuterten Gründen gesetzt wurde, wird zurückgesetzt. Letzteres geschieht, weil in diesem Fall nur indirekt überprüft wird, welche Köpfe angeschlossen sind und welche nicht. Die Meldung „Status = 06_h“ stellt in diesem Fall somit keinen Fehler-Zustand dar.

```
FB10, Netzwerk 2, Fortsetzung

rec1: CALL  "COMR"
       Adresse      :=#Adresse_Ident_Control
       Datenbaustein:=#Datenbaustein

      CALL  "ANALYSE"
       Datenbaustein:=#Datenbaustein

      L     #Kopf_1.Eingangsdaten.Status    // Wenn Status = 06, dann ist kein Kopf
                                            // angeschlossen
      L     B#16#6
      ==I
      S     #Kopf_1.NichtVorhanden
      R     #Kopf_1.VorhandenTC
      R     #Kopf_1.Error
```

Ist der Status nicht 06_h, ist die Überprüfung notwendig, ob das Done-Bit von der Funktion ANALYSE gesetzt wurde und - zur Sicherheit - das NichtVorhanden-Bit nicht gesetzt ist. Das Done-Bit ist gesetzt, wenn neue Daten empfangen und ausgewertet wurden. Ist das Ergebnis dieser Abfrage wahr, so wurde eine zum gesendeten Befehl passende Antwort erhalten und der Kopf ist vorhanden. Das VorhandenTC-Bit für diesen Kopf wird gesetzt und der Kopf ist somit erkannt und erfolgreich initialisiert.
Ist das Ergebnis dieser Abfrage nicht wahr, so wird dieser Teil übersprungen.
Am Ende des Netzwerks 2 erfolgt immer ein Sprung zum Programmteil „Steuerbefehlsumsetzung“.

```
FB10, Netzwerk 2, Fortsetzung

      SET
      U     #Kopf_1.Done
      UN    #Kopf_1.NichtVorhanden
      S     #Kopf_1.VorhandenTC
      SPA   end
```

Die Initialisierung ist abgeschlossen, wenn von jedem Kopf das VorhandenTC- oder das NichtVorhanden-Bit gesetzt ist und somit im nächsten Durchlauf des FB 10 das Freigegeben-Bit gesetzt werden kann.

5.4.3 Ablauf des Programmteils Befehlsbearbeitung

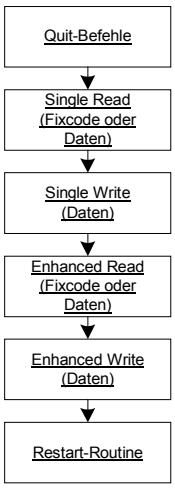

Abb. 13: Ablaufplan Befehlsbearbeitung

Zuerst werden die Quit-Befehle bearbeitet. Der Befehl Quit dient zum Abbrechen jedes anderen Befehls – insbesondere wird er in diesem Programm zum Beenden der Enhanced-Befehle verwendet. Danach werden die Single-Read- und Single-Write-Befehle bearbeitet. Anschließend die Enhanced-Read- und Enhanced-Write-Befehle.
Bei allen Lesebefehlen erfolgt zusätzlich noch einmal die Untergliederung in LeseFixcode und LeseDaten. Diese Unterscheidung entfällt bei den Schreibbefehlen, da Fixcodeträger im Allgemeinen nicht wiederbeschreibbar sind[2].
Abschließend erfolgt der Rücksprung ins Hauptprogramm des FB10, genauer zur Restart-Routine.

Im Folgenden wird die Bearbeitung der einzelnen Befehle jeweils exemplarisch für Kopf 1 aufgezeigt und erläutert. Für die übrigen Köpfe ist diese in separate Netzwerke analog zu übertragen.

[2] Die Ausnahme bildet hier der Datenträger IPC-11 von Pepperl + Fuchs, der mit speziellen Befehlen auch beschreibbar ist.

5.4.4 Ablauf der Bearbeitung der Quit-Befehle

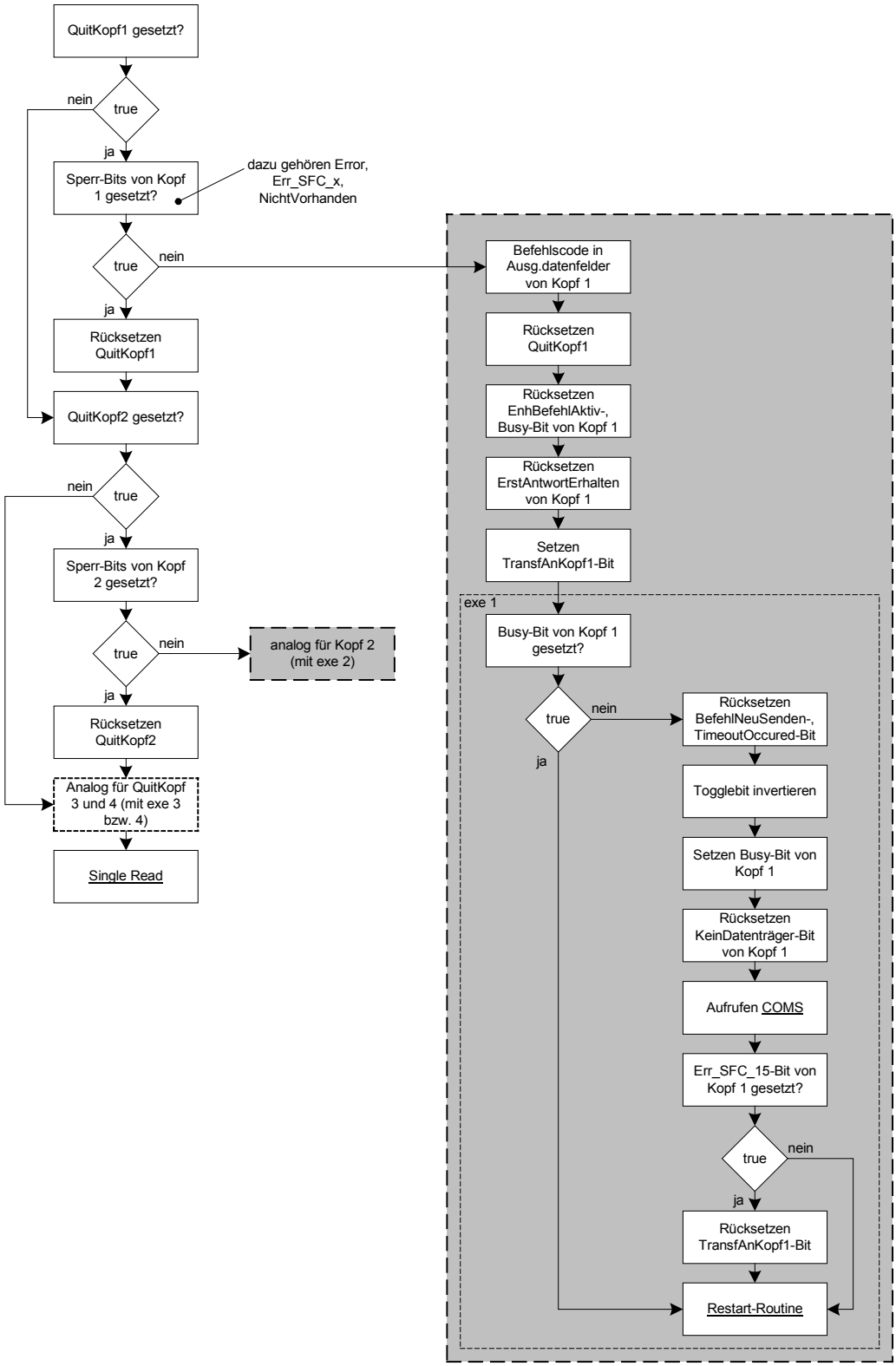

Abb. 14: Ablaufplan der Bearbeitung der Quit-Befehle

Zuerst wird überprüft, ob das QuitKopf1-Bit überhaupt gesetzt ist und die Befehlsausführung somit gewünscht ist. Zusätzlich wird eine Überprüfung der Sperr-Bits von Kopf 1 durchgeführt. Zu den Sperrbits gehören Fehlermeldebits wie Error, Err_SFC_14, Err_SFC_15 und das NichtVorhanden-Bit. Alle diese Bits sollen eine Ausführung des Befehls verhindern, wenn irgendwelche Störungen oder Fehler aufgetreten sind bzw. wenn der Kopf gar nicht vorhanden ist.
Ist das QuitKopf1-Bit nicht gesetzt oder eines der Sperr-Bits gesetzt, so wird das QuitKopf1-Bit ggf. zurückgesetzt und weitergesprungen zur Überprüfung, ob das QuitKopf2-Bit oder ein Sperrbit von Kopf 2 gesetzt ist usw.

```
FB10, Netzwerk 10: Ausführen der Quit-Befehle

qu1:  ON    #QuitKopf1
      O     #Kopf_1.Error
      O     #Kopf_1.Error_SFC_15
      O     #Kopf_1.Error_SFC_14
      ON    #Kopf_1.VorhandenTC
      O     #Kopf_1.NichtVorhanden
      O     #TransfAnKopf1
      R     #QuitKopf1
      SPB   qu2
```

Ist keines dieser Bits gesetzt, so wird der Quit-Befehl für Kopf 1 ausgeführt. Dazu werden der Befehlscode und die erforderlichen Parameter in das Ausgangsdatenfeld von Kopf 1 geladen. Anschließend wird das TransfAnKopf1-Bit gesetzt. Dann wird das QuitKopf1-Bit zurückgesetzt, ebenso wie das EnhBefehlAktiv-, das Busy- und das ErstAntwortErhalten-Bit von Kopf 1. Abschließend erfolgt ein unbedingter Sprung zur exe-Einheit von Kopf 1 (Netzwerk 15 des FB 10).

```
FB10, Netzwerk 10, Fortsetzung

      L     B#16#2                      // Laden des Befehls für Schreib-/Lesekopf 1
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg
      L     B#16#2
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg
      L     B#16#0
      T     #Kopf_1.Ausgangsdaten.Wortadr_High
      T     #Kopf_1.Ausgangsdaten.Wortadr_Low
      SET
      S     #TransfAnKopf1
      R     #QuitKopf1
      R     #Kopf_1.EnhBefehlAktiv
      R     #Kopf_1.Busy
      R     #Kopf_1.ErstAntwortErhalten
      SPA   exe1
```

Wenn die exe-Einheit aufgerufen wird, überprüft sie zuerst, ob für den Kopf gerade ein Befehl aktiv, d.h. ob das Busy-Bit des Kopfes gesetzt ist. Wenn ja, wird zur Restart-Routine gesprungen.
Ist das Busy-Bit nicht gesetzt, kann der Befehl an den Kopf übertragen werden. Zunächst werden dann die Bits BefehlNeuSenden und TimeoutOccured zurückgesetzt.

```
FB10, Netzwerk 15: Ausführungseinheit von Kopf 1 (exe1)

exe1: SET
      U     #Kopf_1.Busy
      SPB   end1

      SET
      R     #Kopf_1.BefehlNeuSenden
      R     #Kopf_1.TimeoutOccured
```

Anschließend wird das zuletzt empfangene Togglebit invertiert und das Busy-Bit von Kopf 1 gesetzt. Dann wird das KeinDatenträger-Bit zurückgesetzt und der Baustein COMS zum Übertragen des Befehls an die IDENT-Control aufgerufen.

```
FB10, Netzwerk 15, Fortsetzung

      L     #Empf_Togglebit                          // Abfrage Status Togglebit
      L     B#16#1
      <>I
      SPB   chg1

      L     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg    // Falls gesetzt, wird es jetzt
                                                     // rückgesetzt
      UW    W#16#FFFE
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg
      SPA   snd1

chg1: L     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg    // Falls nicht gesetzt, wird es
                                                     // jetzt gesetzt
      OW    W#16#1
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg
      SPA   snd1

snd1: S     #Kopf_1.Busy
      R     #Kopf_1.KeinDatentraeger
      CALL  "COMS"
       Adresse      :=#Adresse_Ident_Control
       Ausg_Daten   :=#Kopf_1.Ausgangsdaten
       Datenbaustein:=#Datenbaustein
```

Danach wird überprüft, ob das Err_SFC_15-Bit gesetzt ist, das zur Signalisierung von Übertragungsfehlern bei der Ausführung der Systemfunktion SFC 15 dient. Ist es gesetzt, so wird das TransfAnKopf1-Bit zurückgesetzt und es geht mit der Restart-Routine weiter.
Ist es nicht gesetzt, so geht es direkt mit der Restart-Routine weiter, ohne das TransfAnKopf1-Bit zurückzusetzen. In diesem Fall wird dies später nach der ersten Antwort auf diesen Befehl vom Baustein ANALYSE erledigt.

```
FB10, Netzwerk 15, Fortsetzung

      SET
      U     #Kopf_1.Error_SFC_15
      R     #TransfAnKopf1

      SPA   end1
```

Für die anderen Köpfe erfolgt die Befehlsbearbeitung der Quit-Befehle analog wie für Kopf 1 beschrieben. Auch die exe-Einheiten für die Köpfe 2-4 (Netzwerke 16-18) sind analog aufgebaut.

5.4.5 Ablauf der Bearbeitung der Single-Read-Befehle

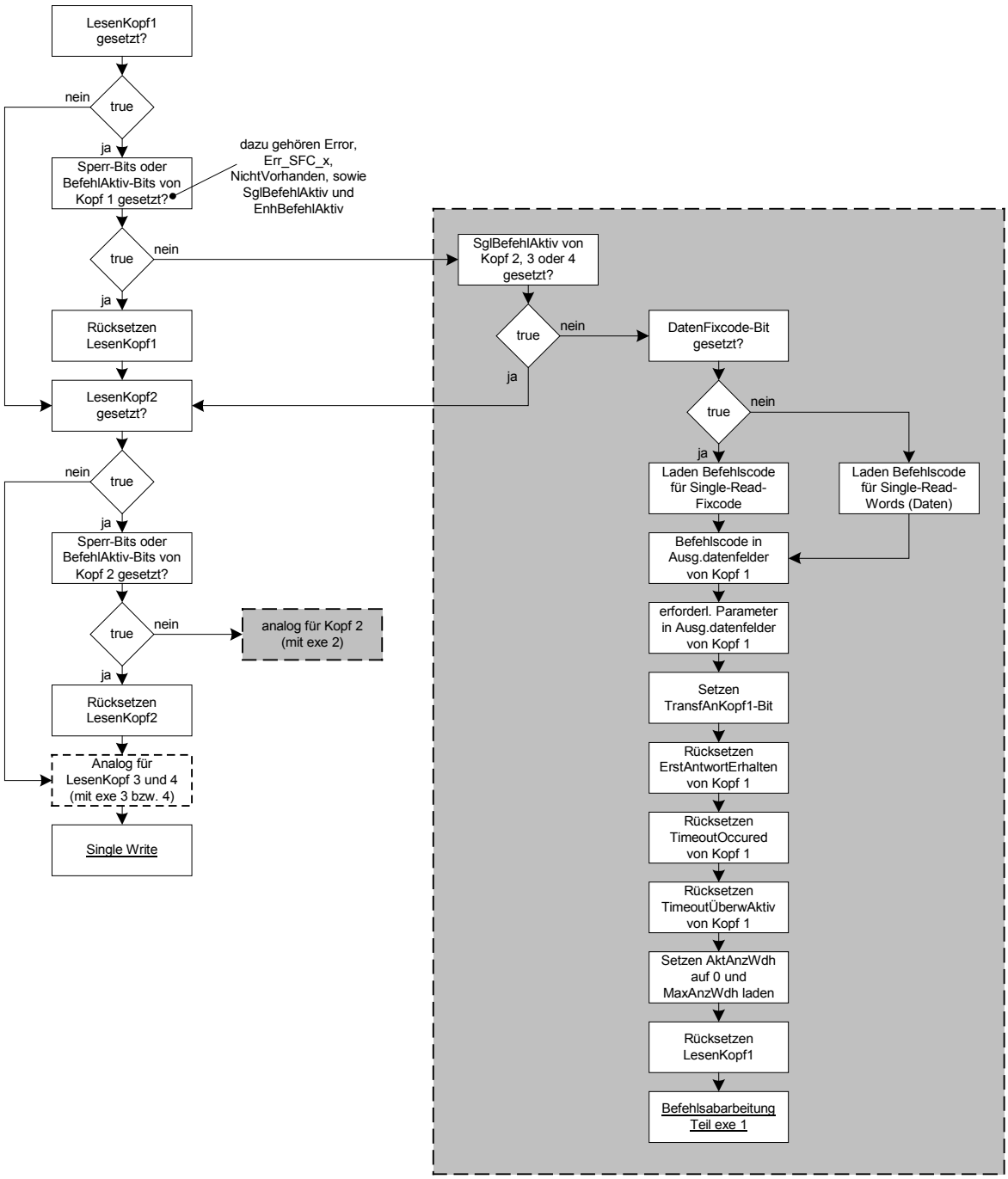

Abb. 15: Ablauf der Bearbeitung der Single-Read-Befehle

Die Überprüfung, ob der Lesebefehl für einen Kopf ausgeführt werden soll, erfolgt ähnlich zu der Überprüfung bei den Quitbefehlen, nur dass bei den Lesebefehlen noch zwei weitere

Sperrbits kontrolliert werden müssen. Dies sind das SglBefehlAktiv- und das EnhBefehl-Aktiv-Bit. Diese beiden Bits haben die Aufgabe die Befehlsausführung zu verhindern, wenn bereits ein Single- bzw. Enhanced-Befehl an diesem Kopf aktiv ist, um diesen nicht zu unterbrechen. Genau wie bei den Quitbefehlen wird bei Nichtgesetztsein des LesenKopf1-Bits bzw. Gesetztsein eines der Sperrbits das LesenKopf1-Bit ggf. zurückgesetzt und mit derselben Prüfung für Kopf 2 fortgefahren.

```
FB10, Netzwerk 11: Ausführen der Single-Read-Befehle

les1: O     #Kopf_1.Error_SFC_15
      O     #Kopf_1.Error_SFC_14
      O     #Kopf_1.Error
      ON    #LesenKopf1
      ON    #Kopf_1.VorhandenTC
      O     #Kopf_1.NichtVorhanden
      O     #TransfAnKopf1
      O     #Kopf_1.EnhBefehlAktiv
      O     #Kopf_1.Busy
      O     #Kopf_1.SglBefehlAktiv
      R     #LesenKopf1
      SPB   les2
```

Andernfalls erfolgt die Überprüfung, ob an den anderen Köpfen ein Single-Befehl aktiv ist.
Bei der Ausführung eines Single-Befehls kann es vorkommen, dass der Datenträger aufgrund einer ungünstigen Positionierung im Erfassungsbereich des Kopfes nicht aufs erste Mal erkannt wird. Deshalb bietet das Programm die Möglichkeit, die Anzahl der Wiederholungen eines Single-Befehls bei der Rückmeldung „Kein Datenträger im Erfassungsbereich“ einzustellen. Zur erneuten Ausführung eines Single-Befehls muss der Befehl allerdings erneut (natürlich mit invertiertem Togglebit) an den Kopf gesendet werden. Wird kein Datenträger erkannt, erfolgt diese Rückmeldung sehr schnell und der Befehl kann direkt wieder gesendet werden. Diese Abfolge geschieht dann mit einer solch hohen Geschwindigkeit, dass die Bearbeitung weiterer Single-Befehle mit der Wiederholungsfunktionalität an anderen Köpfen in diesem Zeitraum nicht möglich ist. Deshalb wird die Ausführung weiterer Single-Befehle an anderen Köpfen im Programm von vornherein zur Vermeidung eventueller Fehler mit Hilfe der SglBefehlAktiv-Bits blockiert, bis der aktive Single-Befehl beendet ist, dieser also ggf. bis zur eingestellten maximalen Häufigkeit wiederholt wurde. Aus diesem Grund sollte die Einstellung der maximalen Anzahl der Wiederholungen mit Bedacht gewählt werden und der Wert 10-15 nicht überschritten werden. Sinnvoll ist eine Einstellung zwischen 3 und 5 Wiederholungen, da dies im Normalfall ausreicht.
Ist an einem der anderen Köpfe nun ein Single-Befehl aktiv, erfolgt ein Sprung zu der Überprüfung von Kopf 2, ob das LesenKopf2-Bit nicht gesetzt bzw. eines der Sperrbits von Kopf 2

gesetzt ist. Das LesenKopf1-Bit wird dabei jedoch nicht zurückgesetzt. Somit wird der Lesebefehl für Kopf 1 direkt im Anschluss an den derzeit noch aktiven Singlebefehl automatisch ausgeführt.

```
FB10, Netzwerk 11, Fortsetzung

      O     #Kopf_2.SglBefehlAktiv
      O     #Kopf_3.SglBefehlAktiv
      O     #Kopf_4.SglBefehlAktiv
      SPB   les2
```

Ist an keinem der anderen Köpfe ein Singlebefehl aktiv, wird zunächst überprüft, ob das DatenFixcode-Bit gesetzt ist. Wenn ja, wird der Befehlscode für den Single-Read-Fixcode-Befehl geladen. Ist es nicht gesetzt, so wird der Befehlscode für den Single-Read-Words-Befehl, also zum Auslesen eines Datenträgers, geladen.
In beiden Fällen geht es danach mit dem Transfer des Befehlscodes und der dazugehörigen Parameter in das Ausgangsdatenfeld von Kopf 1 weiter. Der Anwender kann die Startadresse im Datenträger und die Länge der zu lesenden Daten beim Aufruf des Bausteins angeben. Deshalb müssen diese Informationen zuerst geladen, in das betreffende Byte implementiert und danach in das Ausgangsdatenfeld geschrieben werden. Beim Single-Read-Fixcode-Befehl werden diese Parameter ebenfalls ins Ausgangsdatenfeld geschrieben, sind jedoch für den Befehl nicht erforderlich und werden von der IDENT-Control bei der Befehlsausführung auch nicht beachtet.
Anschließend wird das TransfAnKopf1-Bit gesetzt.

```
FB10, Netzwerk 11, Fortsetzung

      SET                               // Laden SingleRead-Befehl
      UN    #DatenFixcode
      L     B#16#10
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg
      SPB   xx1

      SET                               // Laden SingleReadFixcode-Befehl
      U     #DatenFixcode
      L     B#16#1
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg

xx1:  L     #Anzahl32BitWorte
      SLW   4
      L     B#16#2
      OW
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg

      L     #Datentraegerstartadresse
      L     W#16#FF00
      UW
      SRW   8
      T     #Kopf_1.Ausgangsdaten.Wortadr_High
```

```
L     #Datentraegerstartadresse
T     #Kopf_1.Ausgangsdaten.Wortadr_Low

SET
S     #Kopf_1.SglBefehlAktiv
S     #TransfAnKopf1
```

Danach müssen noch die Bits ErstAntwortErhalten, TimeoutOccured und TimeoutÜberwAktiv zurückgesetzt werden. Die aktuelle Anzahl an Wiederholungen wird nun auf 0 und die maximale Anzahl an Wiederholungen auf den vom Anwender eingegebenen Wert der IN-Variable MaxAnzWdh gesetzt. Im Wiederholungsfalle des Befehls wird dieser Teil des Programms nicht mehr durchlaufen, da sich der Befehl noch im Ausgangsdatenfeld befindet und einfach nur nochmals übertragen werden muss. Deshalb ist es an dieser Stelle möglich die aktuelle Anzahl auf 0 zu setzen.
Schließlich wird noch das LesenKopf1-Bit zurückgesetzt und anschließend mit der Ausführung des exe-Teils von Kopf 1 fortgefahren.

```
FB10, Netzwerk 11, Fortsetzung

      R     #Kopf_1.ErstAntwortErhalten
      R     #Kopf_1.TimeoutOccured
      R     #Kopf_1.TimeoutUeberwAktiv

      L     W#16#0
      T     #Kopf_1.AktAnzWdh
      L     #MaxAnzWdh
      T     #Kopf_1.MaxAnzWdh

      R     #LesenKopf1

      SPA   exe1
```

Wie bereits im Kapitel „Ablauf der Bearbeitung der Quit-Befehle“ beschrieben, erfolgt am Ende der exe-Routine dann der Sprung zur Restart-Routine.
Für die anderen Köpfe erfolgt die Bearbeitung der Single-Read-Befehle analog wie hier für Kopf 1 beschrieben.

5.4.6 Ablauf der Bearbeitung der Single-Write-Befehle

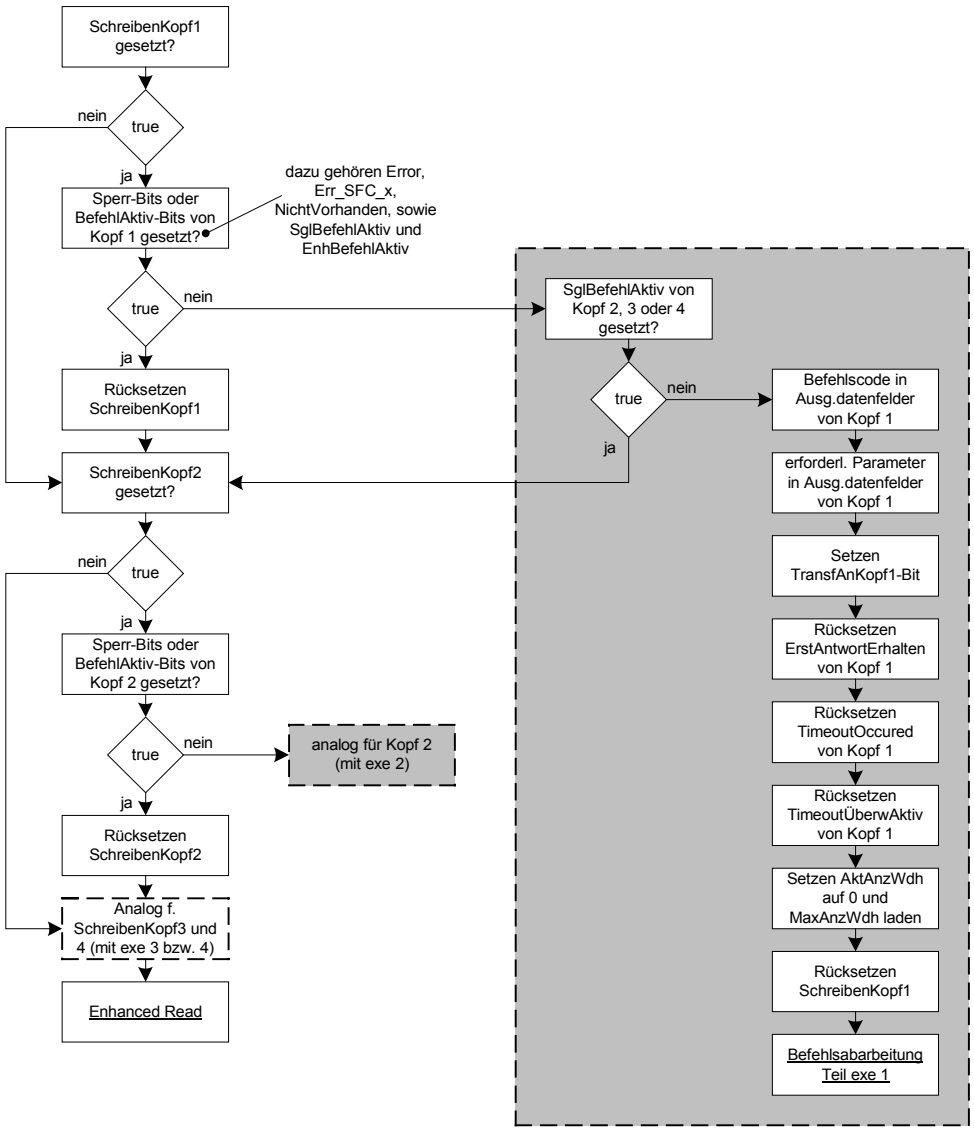

Abb. 16: Ablauf der Bearbeitung der Single-Write-Befehle

Die Bearbeitung der Single-Write-Befehle erfolgt ähnlich der der Single-Read-Befehle, mit dem einzigen Unterschied, dass die Überprüfung des DatenFixcode-Bits wegfällt, da wie bereits erwähnt i.A. ein Beschreiben der Fixcodeträger nicht möglich ist. Der restliche Ablauf ist genau wie bei den Single-Read-Befehlen.

Im Folgenden ist wiederum der Befehlscode exemplarisch für Kopf 1 aufgezeigt. Für die übrigen Köpfe muss dieser nur analog übertragen werden.

FB10, Netzwerk 12: Ausführen der Single-Write-Befehle

```
wrt1: O     #Kopf_1.Error_SFC_15
      O     #Kopf_1.Error_SFC_14
      O     #Kopf_1.Error
      ON    #SchreibenKopf1
      ON    #Kopf_1.VorhandenTC
      O     #Kopf_1.NichtVorhanden
      O     #TransfAnKopf1
      O     #Kopf_1.EnhBefehlAktiv
      O     #Kopf_1.Busy
      O     #Kopf_1.SglBefehlAktiv
      R     #SchreibenKopf1
      SPB   wrt2

      O     #Kopf_2.SglBefehlAktiv
      O     #Kopf_3.SglBefehlAktiv
      O     #Kopf_4.SglBefehlAktiv
      SPB   wrt2

      L     B#16#40                     // Laden des Befehls für Schreib-/Lesekopf 1
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg
      L     #Anzahl32BitWorte
      SLW   4
      L     B#16#2
      OW
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg

      L     #Datentraegerstartadresse
      L     W#16#FF00
      UW
      SRW   8
      T     #Kopf_1.Ausgangsdaten.Wortadr_High

      L     #Datentraegerstartadresse
      T     #Kopf_1.Ausgangsdaten.Wortadr_Low

      SET
      S     #Kopf_1.SglBefehlAktiv
      S     #TransfAnKopf1

      R     #Kopf_1.ErstAntwortErhalten
      R     #Kopf_1.TimeoutOccured
      R     #Kopf_1.TimeoutUeberwAktiv

      L     W#16#0
      T     #Kopf_1.AktAnzWdh
      L     #MaxAnzWdh
      T     #Kopf_1.MaxAnzWdh

      R     #SchreibenKopf1

      SPA   exe1
```

5.4.7 Ablauf der Bearbeitung der Enhanced-Read-Befehle

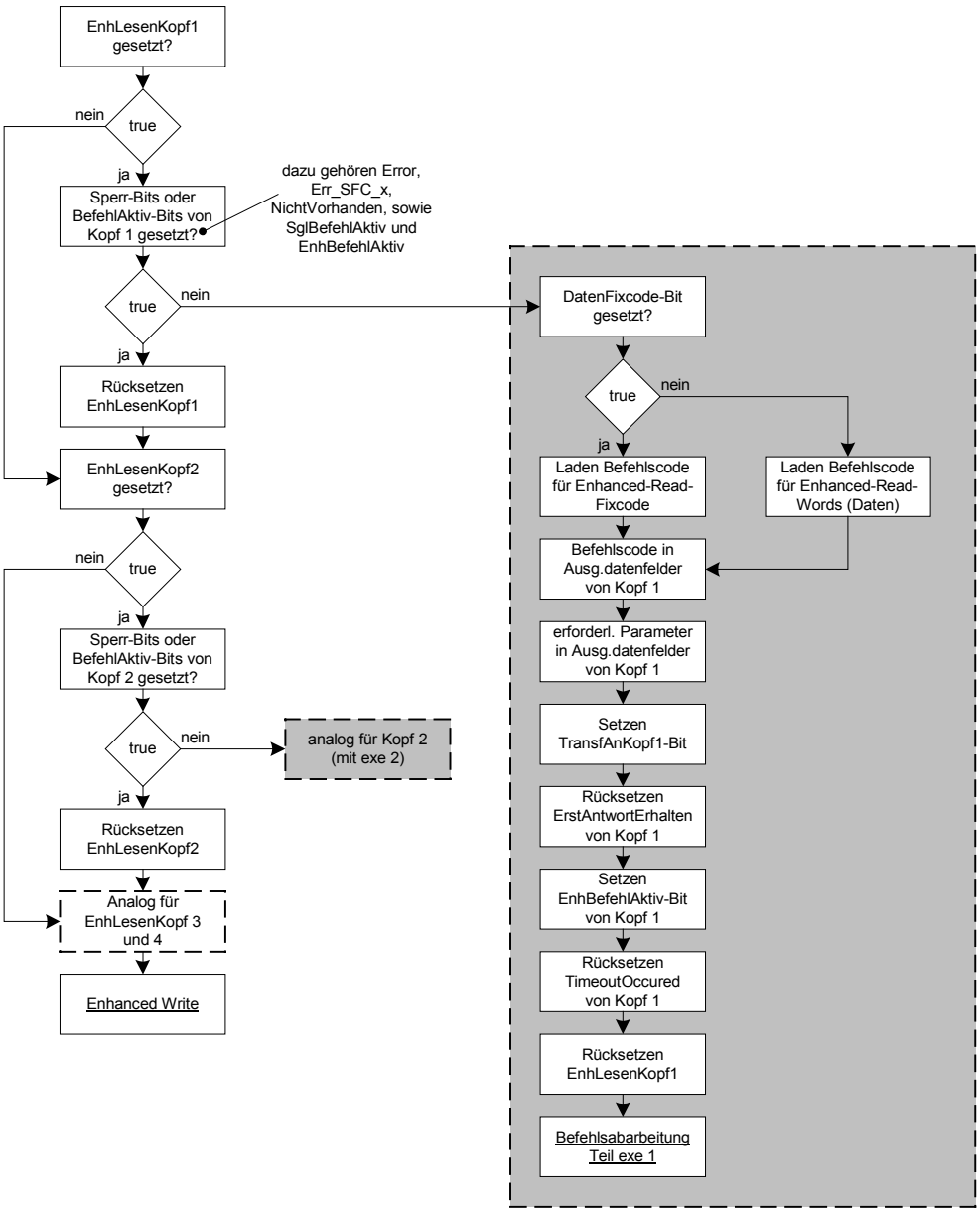

Abb. 17: Ablauf der Bearbeitung der Enhanced-Read-Befehle

Der im linken Teil von Abb. 17 aufgezeigte Ablauf entspricht den zuvor erwähnten Befehlsbearbeitungsabläufen.

```
FB10, Netzwerk 13: Ausführen der Enhanced-Read-Befehle

enL1: O     #Kopf_1.Error_SFC_15
      O     #Kopf_1.Error_SFC_14
      O     #Kopf_1.Error
      ON    #EnhLesenKopf1
      ON    #Kopf_1.VorhandenTC
      O     #Kopf_1.NichtVorhanden
      O     #TransfAnKopf1
      O     #Kopf_1.EnhBefehlAktiv
      O     #Kopf_1.Busy
      O     #Kopf_1.SglBefehlAktiv
      R     #EnhLesenKopf1
      SPB   enL2
```

Einige Unterschiede sind nur im hellgrauen Kasten des Plans enthalten.

So entfällt z.B. die Überprüfung, ob ein SglBefehlAktiv-Bit eines anderen Kopfes gesetzt ist, da die Bearbeitung und Übertragung eines Enhanced-Befehls trotz des aktiven Single-Befehls an einem anderen Kopf möglich ist.

Bei den Enhanced-Read-Befehlen erfolgt genau wie bei den Single-Read-Befehlen die Überprüfung des DatenFixcode-Bits und das Laden des entsprechenden Codes mit den dazugehörigen Parametern in das Ausgangsdatenfeld des entsprechenden Kopfes. Anschließend wird ebenfalls wieder das TransfAnKopf1-Bit gesetzt und das ErstAntwortErhalten-Bit zurückgesetzt. Danach wird das EnhBefehlAktiv-Bit gesetzt, das die Ausführung von weiteren Befehlen an diesen Kopf bis zum Senden des Quitbefehls oder der Durchführung eines Neustarts (durch Rücksetzen des Freigegeben-Bits) unterbindet. Es wird auch das Bit TimeoutOccured zurückgesetzt. Dies erfolgt deshalb, da auch wenn Enhanced-Befehle nicht timeoutüberwacht werden, dieses Bit doch noch von einem vorhergegangen Befehl gesetzt sein könnte. Abschließend wird das EnhLesenKopf1-Bit zurückgesetzt und die exe-Einheit des betreffenden Kopfes aufgerufen.

```
FB10, Netzwerk 13, Fortsetzung

      SET
      UN    #DatenFixcode
      L     B#16#19
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg
      SPB   xx5

      SET
      U     #DatenFixcode
      L     B#16#1D
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg

xx5:  L     #Anzahl32BitWorte
      SLW   4
      L     B#16#2
      OW
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg

      L     #Datentraegerstartadresse
      L     W#16#FF00
      UW
      SRW   8
      T     #Kopf_1.Ausgangsdaten.Wortadr_High

      L     #Datentraegerstartadresse
      T     #Kopf_1.Ausgangsdaten.Wortadr_Low

      SET
      S     #TransfAnKopf1
      S     #Kopf_1.EnhBefehlAktiv
      R     #Kopf_1.ErstAntwortErhalten
      R     #Kopf_1.TimeoutOccured
      R     #EnhLesenKopf1
      SPA   exe1
```

Der hier aufgezeigte Befehlscode für Kopf 1 braucht für die anderen Köpfe wieder nur analog übertragen zu werden.

5.4.8 Ablauf der Bearbeitung der Enhanced-Write-Befehle

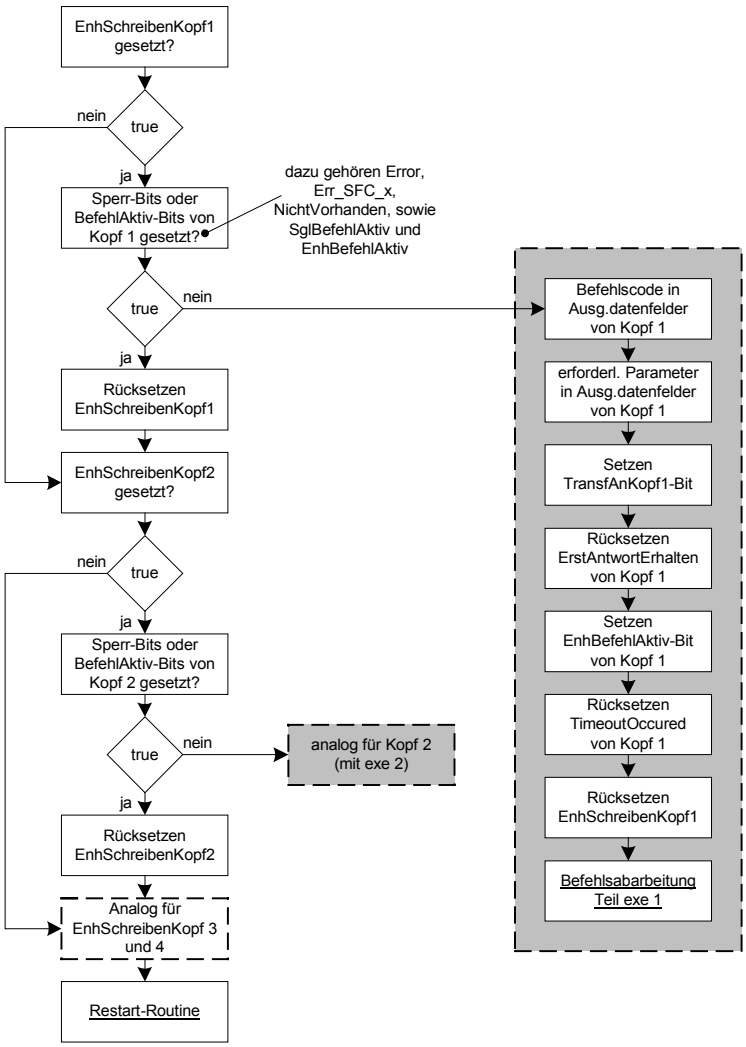

Abb. 18: Ablauf der Bearbeitung der Enhanced-Write-Befehle

Der Ablauf der Enhanced-Write-Befehle erfolgt genau wie der der Enhanced-Read-Befehle, nur dass aus bekannten Gründen die Fixcode/Daten-Überprüfung wegfällt.

Der Programmcode ist exemplarisch für Kopf 1 im Folgenden abgebildet – der Code für die anderen Köpfe muss wie gewohnt analog übertragen werden.

```
FB10, Netzwerk 14: Ausführen der Enhanced-Write-Befehle

enS1: O     #Kopf_1.Error_SFC_15
      O     #Kopf_1.Error_SFC_14
      O     #Kopf_1.Error
      ON    #EnhSchreibenKopf1
      ON    #Kopf_1.VorhandenTC
      O     #Kopf_1.NichtVorhanden
      O     #TransfAnKopf1
      O     #Kopf_1.EnhBefehlAktiv
      O     #Kopf_1.Busy
      O     #Kopf_1.SglBefehlAktiv
      R     #EnhSchreibenKopf1
      SPB   enS2

      L     B#16#1A                    // Laden des Befehls für Schreib-/Lesekopf 1
      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg

      L     #Anzahl32BitWorte
      SLW   4
      L     B#16#2
      OW
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg

      L     #Datentraegerstartadresse
      L     W#16#FF00
      UW
      SRW   8
      T     #Kopf_1.Ausgangsdaten.Wortadr_High

      L     #Datentraegerstartadresse
      T     #Kopf_1.Ausgangsdaten.Wortadr_Low

      SET
      S     #TransfAnKopf1
      S     #Kopf_1.EnhBefehlAktiv
      R     #Kopf_1.ErstAntwortErhalten
      R     #Kopf_1.TimeoutOccured
      R     #EnhSchreibenKopf1
      SPA   exe1
```

5.4.9 Ablauf des Programmteils Timeout-Überwachung

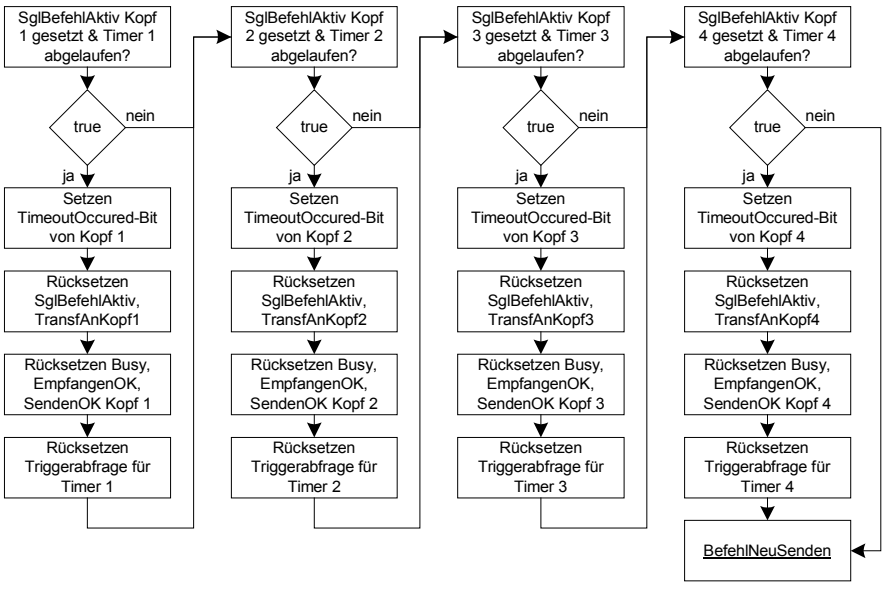

Abb. 19: Ablaufplan des Programmteils Timeout-Überwachung

Im Programm wird die Timeoutüberwachung nur bei den Single-Befehlen verwendet. Dementsprechend wird bei der Timeoutüberwachung überprüft, ob überhaupt ein Single-Befehl aktiv ist und ob der dazugehörige Timer abgelaufen ist. Wenn nein, erfolgt die Überprüfung für Kopf 2 usw.

Wenn ja, so wird das TimeoutOccured-Bit des entsprechenden Kopfes gesetzt. Anschließend werden die Bits SglBefehlAktiv, TransfAnKopf1, EmpfangenOK, SendenOK, Busy und TimeoutÜberwAktiv zurückgesetzt. Damit der Timer beim nächsten Single-Befehl wieder gestartet werden kann, muss die Abfrage, die den Timer startet, an dieser Stelle ebenfalls zurückgesetzt werden.

In jedem Fall geht es danach mit dem Programmteil „BefehlNeuSenden“ weiter.

```
FB10, Netzwerk 8: Timeoutüberwachung

      SET
      U     #Kopf_1.SglBefehlAktiv
      UN    #TimeoutTimerKopf1
      S     #Kopf_1.TimeoutOccured
      R     #Kopf_1.SglBefehlAktiv
      R     #TransfAnKopf1
      R     #Kopf_1.EmpfangenOK
      R     #Kopf_1.SendenOK
      R     #Kopf_1.Busy
      R     #Kopf_1.TimeoutUeberwAktiv

      SET
      U     #Kopf_1.TimeoutUeberwAktiv  // Rücksetzen der Triggerabfrage für Timer
                                        // von Kopf 1, da Bit hier definitiv 0 ist
      SI    #TimeoutTimerKopf1

      SET
      U     #Kopf_2.SglBefehlAktiv
      UN    #TimeoutTimerKopf2
      S     #Kopf_2.TimeoutOccured
      R     #Kopf_2.SglBefehlAktiv
      R     #TransfAnKopf2
      R     #Kopf_2.EmpfangenOK
      R     #Kopf_2.SendenOK
      R     #Kopf_2.Busy
      R     #Kopf_2.TimeoutUeberwAktiv

      SET
      U     #Kopf_2.TimeoutUeberwAktiv  // Rücksetzen der Triggerabfrage für Timer
                                        // von Kopf 2
      SI    #TimeoutTimerKopf2

      SET
      U     #Kopf_3.SglBefehlAktiv
      UN    #TimeoutTimerKopf3
      S     #Kopf_3.TimeoutOccured
      R     #Kopf_3.SglBefehlAktiv
      R     #TransfAnKopf3
      R     #Kopf_3.EmpfangenOK
      R     #Kopf_3.SendenOK
      R     #Kopf_3.Busy
      R     #Kopf_3.TimeoutUeberwAktiv


      SET
      U     #Kopf_3.TimeoutUeberwAktiv  // Rücksetzen der Triggerabfrage für Timer
                                        // von Kopf 3
      SI    #TimeoutTimerKopf3
```

```
SET
U     #Kopf_4.SglBefehlAktiv
UN    #TimeoutTimerKopf4
S     #Kopf_4.TimeoutOccured
R     #Kopf_4.SglBefehlAktiv
R     #TransfAnKopf4
R     #Kopf_4.EmpfangenOK
R     #Kopf_4.SendenOK
R     #Kopf_4.Busy
R     #Kopf_4.TimeoutUeberwAktiv

SET
U     #Kopf_4.TimeoutUeberwAktiv  // Rücksetzen der Triggerabfrage für Timer
                                  // von Kopf 4
SI    #TimeoutTimerKopf4
```

5.4.10 Ablauf des Programmteils BefehlNeuSenden

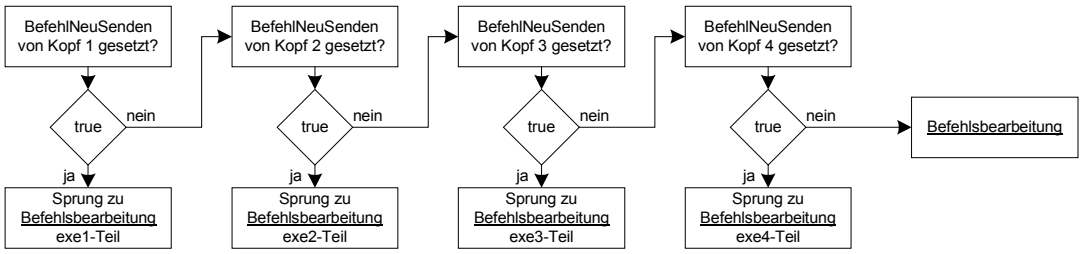

Abb. 20: Ablaufplan des Programmteils BefehlNeuSenden

In diesem Programmteil werden der Reihe nach die BefehlNeuSenden-Bits der einzelnen Köpfe überprüft. Ist ein Bit gesetzt, wird sofort in die exe-Einheit des entsprechenden Kopfes gesprungen und der Befehl somit direkt noch einmal an die IDENT-Control bzw. den angeschlossenen Kopf gesendet.

Ist keines der BefehlNeuSenden-Bits gesetzt, so geht es im Programm mit der Bearbeitung der Quit-Befehle weiter.

```
FB10, Netzwerk 9: Befehle neu senden?

      SET
      U     #Kopf_1.BefehlNeuSenden
      SPB   exe1

      SET
      U     #Kopf_2.BefehlNeuSenden
      SPB   exe2


      SET
      U     #Kopf_3.BefehlNeuSenden
      SPB   exe3

      SET
      U     #Kopf_4.BefehlNeuSenden
      SPB   exe4
      SPA   qu1
```

5.4.11 Ablauf der Restart-Routine

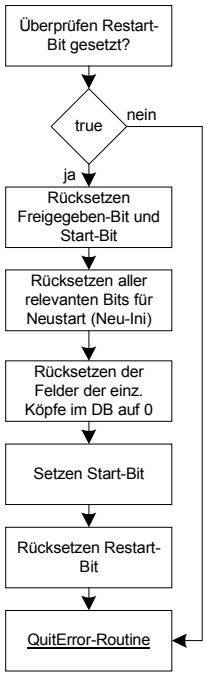

Abb. 21: Ablauf der Restart-Routine

Zunächst wird überprüft, ob das Restart-Bit gesetzt ist. Ist dies nicht der Fall, ist keine Neuinitialisierung von Seiten des Anwenders gewünscht, d.h. das Freigegeben-Bit wurde nicht zurückgesetzt. Die Restart-Routine wird damit übersprungen und gleich mit der QuitError-Routine weitergemacht (siehe nächstes Kapitel).
Ist das Restart-Bit gesetzt, so werden das Freigegeben- und das Start-Bit zurückgesetzt. Das Freigegeben-Bit wird an dieser Stelle zurückgesetzt, da theoretisch die Neuinitialisierung auch mittels direktem Setzen des internen Restart-Bits erfolgen kann.

```
FB10, Netzwerk 19: Restart-Routine

end1: AUF   DB [#zugeord_DB]       // Öffnet den zugeordneten DB

      SET
      U     #Restart
      SPBN  end2

      R     #Freigegeben
      R     #Start
```

Anschließend werden alle Signalisierungs-, Verriegelungs- und Freigabe-Bits aller Köpfe und die des Zwischenspeichers zurückgesetzt.

FB10, Netzwerk 19, Fortsetzung

```
// Rücksetzen aller relevanten Bits für Neustart/Neuinitialisierung
      R     #Kopf_1.VorhandenTC
      R     #Kopf_1.NichtVorhanden
      R     #Kopf_1.Done
      R     #Kopf_1.Error
      R     #Kopf_1.Error_SFC_14
      R     #Kopf_1.Error_SFC_15
      R     #Kopf_1.EmpfangenOK
      R     #Kopf_1.SendenOK
      R     #Kopf_1.Busy
      R     #Kopf_1.EnhBefehlAktiv
      R     #Kopf_1.NeueDatenVorhanden
      R     #Kopf_1.TimeoutUeberwAktiv
      R     #Kopf_1.TimeoutOccured
      R     #Kopf_1.SglBefehlAktiv
      R     #TransfAnKopf1

      R     #Kopf_2.VorhandenTC
      R     #Kopf_2.NichtVorhanden
      R     #Kopf_2.Done
      R     #Kopf_2.Error
      R     #Kopf_2.Error_SFC_14
      R     #Kopf_2.Error_SFC_15
      R     #Kopf_2.EmpfangenOK
      R     #Kopf_2.SendenOK
      R     #Kopf_2.Busy
      R     #Kopf_2.EnhBefehlAktiv
      R     #Kopf_2.NeueDatenVorhanden
      R     #Kopf_2.TimeoutUeberwAktiv
      R     #Kopf_2.TimeoutOccured
      R     #Kopf_2.SglBefehlAktiv
      R     #TransfAnKopf2

      R     #Kopf_3.VorhandenTC
      R     #Kopf_3.NichtVorhanden
      R     #Kopf_3.Done
      R     #Kopf_3.Error
      R     #Kopf_3.Error_SFC_14
      R     #Kopf_3.Error_SFC_15
      R     #Kopf_3.EmpfangenOK
      R     #Kopf_3.SendenOK
      R     #Kopf_3.Busy
      R     #Kopf_3.EnhBefehlAktiv
      R     #Kopf_3.NeueDatenVorhanden
      R     #Kopf_3.TimeoutUeberwAktiv
      R     #Kopf_3.TimeoutOccured
      R     #Kopf_3.SglBefehlAktiv
      R     #TransfAnKopf3


      R     #Kopf_4.VorhandenTC
      R     #Kopf_4.NichtVorhanden
      R     #Kopf_4.Done
      R     #Kopf_4.Error
      R     #Kopf_4.Error_SFC_14
      R     #Kopf_4.Error_SFC_15
      R     #Kopf_4.EmpfangenOK
      R     #Kopf_4.SendenOK
      R     #Kopf_4.Busy
      R     #Kopf_4.EnhBefehlAktiv
      R     #Kopf_4.NeueDatenVorhanden
      R     #Kopf_4.TimeoutUeberwAktiv
      R     #Kopf_4.TimeoutOccured
      R     #Kopf_4.SglBefehlAktiv
      R     #TransfAnKopf4
      R     #Zwischenspeicher.VorhandenTC
      R     #Zwischenspeicher.NichtVorhanden
      R     #Zwischenspeicher.Done
      R     #Zwischenspeicher.Error
      R     #Zwischenspeicher.Error_SFC_14
      R     #Zwischenspeicher.Error_SFC_15
```

```
R     #Zwischenspeicher.EmpfangenOK
R     #Zwischenspeicher.SendenOK
R     #Zwischenspeicher.Busy
R     #Zwischenspeicher.EnhBefehlAktiv
R     #Zwischenspeicher.NeueDatenVorhanden
R     #Zwischenspeicher.TimeoutUeberwAktiv
R     #Zwischenspeicher.TimeoutOccured
R     #Zwischenspeicher.SglBefehlAktiv
```

Danach werden die folgenden Felder im Datenbaustein auf ihren Anfangszustand (den Wert „0“) gebracht:

- Befehlscode_Ausg und Befehlscode_Eing
- WortAnzTog_Ausg und WortAnzTog_Eing
- Wortadr_High und Wortadr_Low
- Status und Ausfzähler

Dies erfolgt ebenfalls für alle Köpfe. Der Sinn dabei besteht darin, dass nun keine Überbleibsel an Parametern des vorhergegangenen Befehls mehr in diesen Feldern stehen, sondern diese Felder definiert auf den Anfangszustand gebracht werden.

```
FB10, Netzwerk 19, Fortsetzung

      L     B#16#0

      T     #Kopf_1.Ausgangsdaten.Befehlscode_Ausg
      T     #Kopf_1.Ausgangsdaten.WortAnzTog_Ausg
      T     #Kopf_1.Ausgangsdaten.Wortadr_High
      T     #Kopf_1.Ausgangsdaten.Wortadr_Low
      T     #Kopf_1.Eingangsdaten.Befehlscode_Eing
      T     #Kopf_1.Eingangsdaten.WortAnzTog_Eing
      T     #Kopf_1.Eingangsdaten.Status
      T     #Kopf_1.Eingangsdaten.Ausfzaehler

      T     #Kopf_2.Ausgangsdaten.Befehlscode_Ausg
      T     #Kopf_2.Ausgangsdaten.WortAnzTog_Ausg
      T     #Kopf_2.Ausgangsdaten.Wortadr_High
      T     #Kopf_2.Ausgangsdaten.Wortadr_Low
      T     #Kopf_2.Eingangsdaten.Befehlscode_Eing
      T     #Kopf_2.Eingangsdaten.WortAnzTog_Eing
      T     #Kopf_2.Eingangsdaten.Status
      T     #Kopf_2.Eingangsdaten.Ausfzaehler

      T     #Kopf_3.Ausgangsdaten.Befehlscode_Ausg
      T     #Kopf_3.Ausgangsdaten.WortAnzTog_Ausg
      T     #Kopf_3.Ausgangsdaten.Wortadr_High
      T     #Kopf_3.Ausgangsdaten.Wortadr_Low
      T     #Kopf_3.Eingangsdaten.Befehlscode_Eing
      T     #Kopf_3.Eingangsdaten.WortAnzTog_Eing
      T     #Kopf_3.Eingangsdaten.Status
      T     #Kopf_3.Eingangsdaten.Ausfzaehler

      T     #Kopf_4.Ausgangsdaten.Befehlscode_Ausg
      T     #Kopf_4.Ausgangsdaten.WortAnzTog_Ausg
      T     #Kopf_4.Ausgangsdaten.Wortadr_High
      T     #Kopf_4.Ausgangsdaten.Wortadr_Low
      T     #Kopf_4.Eingangsdaten.Befehlscode_Eing
      T     #Kopf_4.Eingangsdaten.WortAnzTog_Eing
      T     #Kopf_4.Eingangsdaten.Status
      T     #Kopf_4.Eingangsdaten.Ausfzaehler
```

Schließlich wird noch das Start-Bit zum Anstoßen der Initialisierungsroutine gesetzt und das Restart-Bit zurückgesetzt.

Danach geht es immer direkt weiter mit der Ausführung der QuitError-Routine.

```
FB10, Netzwerk 19, Fortsetzung

      S     #Start
      R     #Restart

end2: NOP   0
```

5.4.12 Ablauf der QuitError-Routine

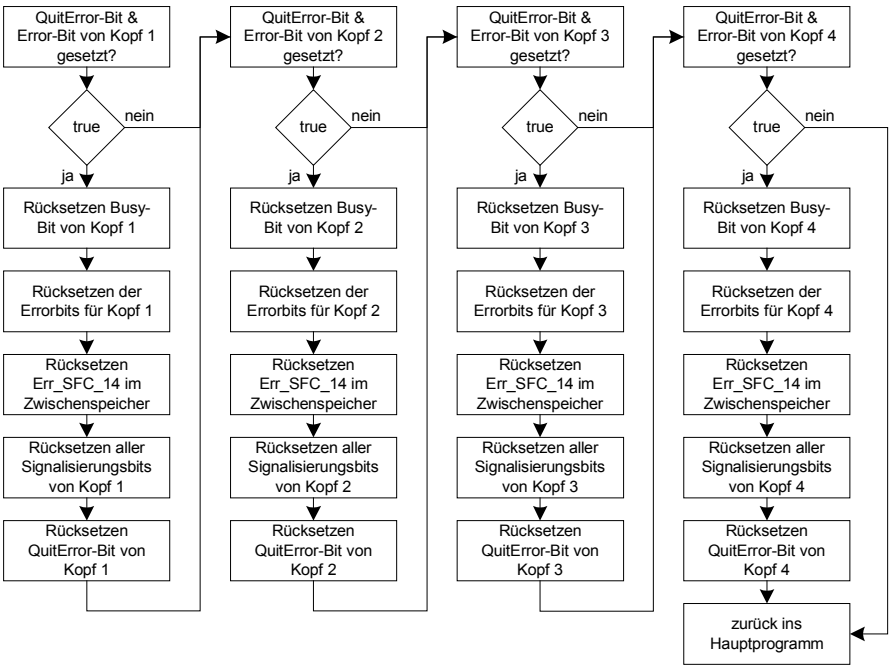

Abb. 22: Ablauf der QuitError-Routine

Es wird zunächst überprüft, ob das QuitError-Bit und das Error-Bit von Kopf 1 gesetzt sind. Ist dies nicht der Fall, so geht es mit der gleichen Überprüfung für Kopf 2 weiter etc.
Sind die beiden Bits jedoch gesetzt, werden das Error-, Err_SFC_14- und Err_SFC_15-Bit von Kopf 1, sowie das Err_SFC_14-Bit im Zwischenspeicher zurückgesetzt. Anschließend werden alle Signalisierungs- und Sperrbits, gefolgt vom QuitErrorBit von Kopf 1 zurückgesetzt und es geht weiter mit der Überprüfung auf gemeinsames Gesetztsein des QuitError- und des Error-Bits von Kopf 2 usw.

In einem SPS-Zyklus können also für alle Köpfe die Fehler quittiert und die Köpfe wieder in den normalen Betriebszustand gebracht werden.
Nach der QuitError-Routine geht es weiter mit dem Programmteil „Rückgabe der Statusbytes“ (s. S. 33).

FB10, Netzwerk 20: QuitError-Routine

```
SET
U     #Kopf_1.QuitError
U     #Kopf_1.Error
R     #Kopf_1.Error
R     #Kopf_1.Error_SFC_14
R     #Kopf_1.Error_SFC_15
R     #Zwischenspeicher.Error_SFC_14
R     #Kopf_1.Busy
R     #Kopf_1.EnhBefehlAktiv
R     #Kopf_1.SglBefehlAktiv
R     #Kopf_1.Done
R     #Kopf_1.EmpfangenOK
R     #Kopf_1.SendenOK
R     #Kopf_1.NeueDatenVorhanden
R     #Kopf_1.TimeoutUeberwAktiv
R     #TransfAnKopf1
R     #Kopf_1.KeinDatentraeger
R     #Kopf_1.TimeoutOccured
R     #Kopf_1.QuitError

SET
U     #Kopf_2.QuitError
U     #Kopf_2.Error
R     #Kopf_2.Error
R     #Kopf_2.Error_SFC_14
R     #Kopf_2.Error_SFC_15
R     #Zwischenspeicher.Error_SFC_14
R     #Kopf_2.Busy
R     #Kopf_2.EnhBefehlAktiv
R     #Kopf_2.SglBefehlAktiv
R     #Kopf_2.Done
R     #Kopf_2.EmpfangenOK
R     #Kopf_2.SendenOK
R     #Kopf_2.NeueDatenVorhanden
R     #Kopf_2.TimeoutUeberwAktiv
R     #TransfAnKopf2
R     #Kopf_2.KeinDatentraeger
R     #Kopf_2.TimeoutOccured
R     #Kopf_2.QuitError


SET
U     #Kopf_3.QuitError
U     #Kopf_3.Error
R     #Kopf_3.Error
R     #Kopf_3.Error_SFC_14
R     #Kopf_3.Error_SFC_15
R     #Zwischenspeicher.Error_SFC_14
R     #Kopf_3.Busy
R     #Kopf_3.EnhBefehlAktiv
R     #Kopf_3.SglBefehlAktiv
R     #Kopf_3.Done
R     #Kopf_3.EmpfangenOK
R     #Kopf_3.SendenOK
R     #Kopf_3.NeueDatenVorhanden
R     #Kopf_3.TimeoutUeberwAktiv
R     #TransfAnKopf3
R     #Kopf_3.KeinDatentraeger
R     #Kopf_3.TimeoutOccured
R     #Kopf_3.QuitError

SET
U     #Kopf_4.QuitError
U     #Kopf_4.Error
R     #Kopf_4.Error
R     #Kopf_4.Error_SFC_14
R     #Kopf_4.Error_SFC_15
R     #Zwischenspeicher.Error_SFC_14
R     #Kopf_4.Busy
R     #Kopf_4.EnhBefehlAktiv
R     #Kopf_4.SglBefehlAktiv
```

```
R     #Kopf_4.Done
R     #Kopf_4.EmpfangenOK
R     #Kopf_4.SendenOK
R     #Kopf_4.NeueDatenVorhanden
R     #Kopf_4.TimeoutUeberwAktiv
R     #TransfAnKopf4
R     #Kopf_4.KeinDatentraeger
R     #Kopf_4.TimeoutOccured
R     #Kopf_4.QuitError
```

Damit wäre der komplette Ablauf des FB 10 „Ident-Control-System“ erklärt und es werden jetzt die Abläufe der anderen Bausteine des Beispielprogramms COMR, COMS und ANALYSE erläutert.

5.4.13 Ablauf des Bausteins COMR (FC 100)

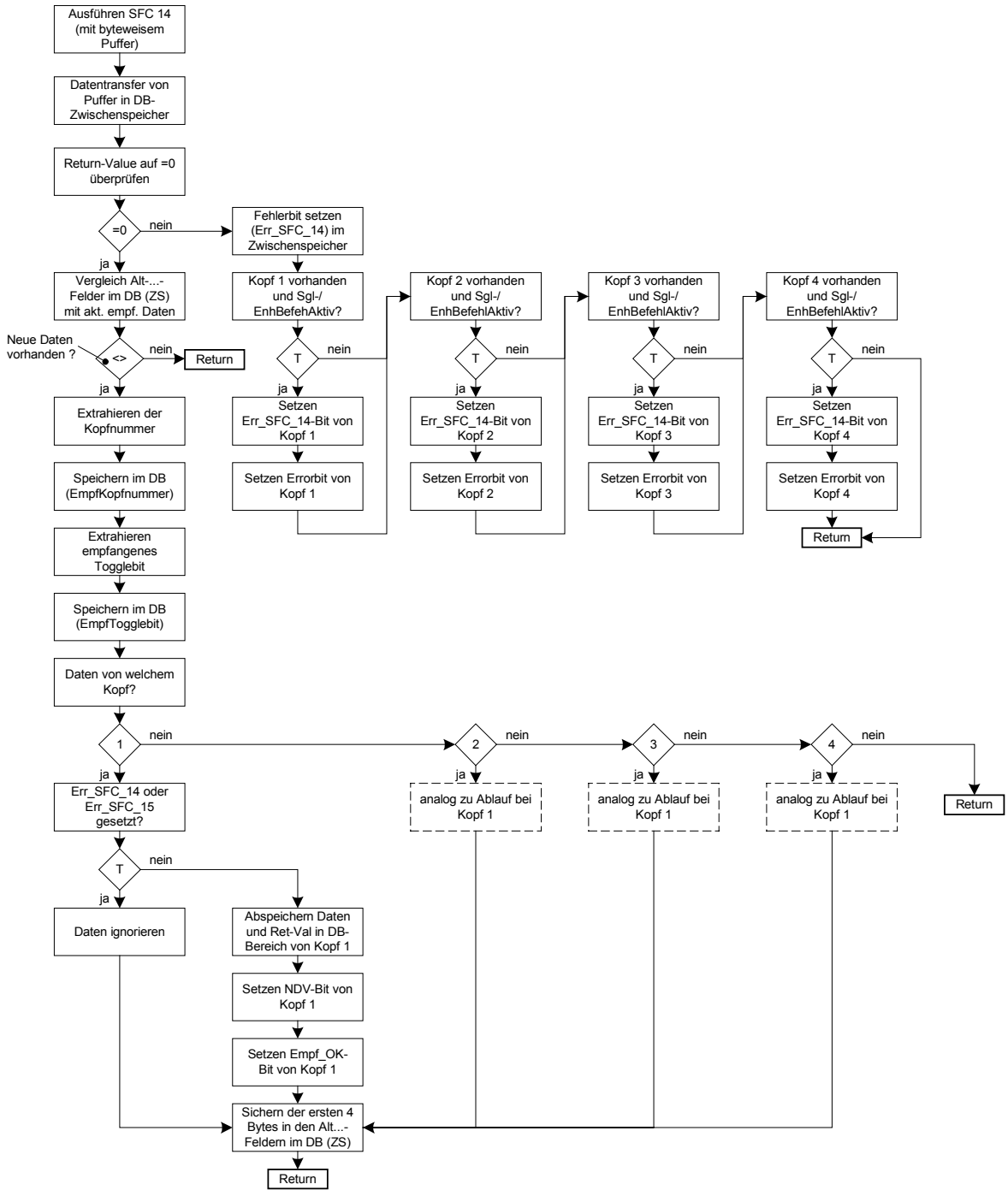

Abb. 23: Ablaufplan des Bausteins COMR (FC 100)

Der Deklarationsteil des Bausteins COMR ist in Abb. 23 aufgezeigt. Die Bedeutung und die Verwendung der deklarierten Variablen werden in den weiteren Ausführungen deutlich.

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Adresse	WORD		
2.0	in	Datenbaustein	INT		
out					
in_out					
0.0	temp	zugeord_DB	INT		
2.0	temp	Empf_Kopfnummer	BYTE		
4.0	temp	Puffer	"Byte-Puffer"		
36.0	temp	Zwischenpuffer	DWORD		

Tabelle 6: Deklarationsteil des Bausteins COMR (FC 100)

Zunächst wird der der IDENT-Control-Einheit zugeordnete Datenbaustein geöffnet. Anschließend wird die Systemfunktion SFC 14 mit den entsprechenden Parametern aufgerufen. Als Adresse muss die Adresse der IDENT-Control-Einheit in der SPS (hier also 100_{hex}) angegeben werden. Diese wird jedoch automatisch vom FB 10 weitergereicht, sodass sich der Anwender darum nicht zu kümmern braucht. Der Rückgabewert (Return Value) wird im zugeordneten Datenbaustein in das Feld Zwischenspeicher.Ret_Val_SFC14 gespeichert. Als Puffer, in den die empfangenen Daten geladen werden, muss an dieser Stelle ein Speicherbereich in reiner Byte-Struktur verwendet werden. D.h. die im übrigen Datenbaustein verwendete gemischte Struktur bestehend aus Bytes und Doppelworten kommt hier nicht in Frage. Aus diesem Grund wurde im Vorfeld die Datenstruktur UDT 101 „Byte-Puffer“ definiert, die hier nun ihre Anwendung findet.

```
FC100, Netzwerk 1, Abholen der Daten von der IDENT-Control

      L     #Datenbaustein
      T     #zugeord_DB

      AUF   DB [#zugeord_DB]

      CALL  "DPRD_DAT"
       LADDR  :=#Adresse
       RET_VAL:=DBW496                  // Abspeichern im zugeord_DB im
                                        // Zwischenspeicher.Ret_Val_SFC14
       RECORD :=#Puffer                 // Abspeichern in Puffer
```

Im weiteren Ablauf des Bausteins werden die nun im Empfangspuffer zur Verfügung stehenden Daten zur einfacheren Weiterverarbeitung in die im Datenbaustein übliche gemischte Struktur aus Byte und Doppelworten gebracht und im Zwischenspeicher des Datenbausteins abgespeichert.

FC100, Netzwerk 2, Umformen der Daten in DB-Struktur

```
      L     #Puffer.Befehlscode
      T     DBB  420
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Befehlscode_Eing
      L     #Puffer.WortAnzTog
      T     DBB  421
// Abspeichern nach Zwischenspeicher.Eingangsdaten.WortAnzTog_Eing
      L     #Puffer.Status
      T     DBB  422
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Status
      L     #Puffer.Ausfzaehler
      T     DBB  423
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Ausfzaehler
      L     #Puffer.Datenbyte1
      SLD   8
      L     #Puffer.Datenbyte2
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte3
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte4
      OD
      T     DBD  424
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Daten_DW1
      L     #Puffer.Datenbyte5
      SLD   8
      L     #Puffer.Datenbyte6
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte7
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte8
      OD
      T     DBD  428
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Daten_DW2
      L     #Puffer.Datenbyte9
      SLD   8
      L     #Puffer.Datenbyte10
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte11
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte12
      OD
      T     DBD  432
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Daten_DW3
      L     #Puffer.Datenbyte13
      SLD   8
      L     #Puffer.Datenbyte14
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte15
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
```

```
      SLD   8
      L     #Puffer.Datenbyte16
      OD
      T     DBD  436
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Daten_DW4
      L     #Puffer.Datenbyte17
      SLD   8
      L     #Puffer.Datenbyte18
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte19
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte20
      OD
      T     DBD  440
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Daten_DW5
      L     #Puffer.Datenbyte21
      SLD   8
      L     #Puffer.Datenbyte22
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte23
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte24
      OD
      T     DBD  444
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Daten_DW6
      L     #Puffer.Datenbyte25
      SLD   8
      L     #Puffer.Datenbyte26
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte27
      OD
      T     #Zwischenpuffer
      L     #Zwischenpuffer
      SLD   8
      L     #Puffer.Datenbyte28
      OD
      T     DBD  448
// Abspeichern nach Zwischenspeicher.Eingangsdaten.Daten_DW7
```

Im Anschluss daran wird die fehlerfreie Ausführung der SFC 14 anhand des Return Values überprüft. Ist der Wert des Return Value ungleich Null, so ist ein Fehler bei der Übertragung bzw. Abholung der Daten aufgetreten und es erfolgt ein Sprung in die Fehlerbehandlung dieses Bausteins (siehe letzter Absatz dieses Kapitels).

Steht im Return Value der Wert Null, so wurde der Empfang der Daten fehlerfrei durchgeführt und es wird nun überprüft, ob die empfangenen Daten auch wirklich neu für die SPS sind, oder ob es sich schlichtweg um das letzte sich noch im Ausgangsdatenfeld der IDENT-

Control befindliche Telegramm handelt[3]. Dazu werden die 4 ersten Byte der neuen Daten mit denen des letzten Datensatzes verglichen. Besteht ein Unterschied in mindestens einem Bit, so liegen neue Daten vor. Handelt es sich um neue Daten wird in den Programmteil ndv gesprungen, wenn nicht, erfolgt ein Sprung ans Ende des Bausteins.

```
FC100, Netzwerk 3, Überprüfen, ob neue Daten vorliegen

// Überprüfen des Ret_Val auf "0"
      L     DBW  496                    // Überprüfen des Ret_Val auf "0"
      L     W#16#0
      ==I
      SPBN  err

// Überprüfung, ob neue Daten vorliegen
      L     DBB  501                    // Lade Alt_Befehl_Eing
      L     DBB  420                    // Lade Befehl_Eing der akt. empf. Daten
      <>I
      SPB   ndv
      L     DBB  502                    // Lade Alt_WortAnzTog_Eing
      L     DBB  421                    // Lade WortAnzTog_Eing der akt. empf. Daten
      <>I
      SPB   ndv
      L     DBB  503                    // Lade Alt_Status
      L     DBB  422                    // Lade Status der akt. empf. Daten
      <>I
      SPB   ndv
      L     DBB  504                    // Lade Alt_Ausfzaehler
      L     DBB  423                    // Lade Ausfzaehler der akt. empf. Daten
      <>I
      SPB   ndv
      SPA   end                         // d.h. keine neue Daten vorhanden = Sprung
                                        // ans Ende von COMR
```

Im Programmteil ndv, was im Übrigen für „Neue Daten Vorhanden“ steht, wird zunächst die Kopfnummer aus den empfangenen Daten herausgelesen und im Datenbaustein gespeichert. Genauso wird mit dem empfangenen Togglebit verfahren, da dieses wichtig ist, wenn es darum geht, den nächsten Befehl mit invertiertem Togglebit an die IDENT-Control zu übertragen.

```
FC100, Netzwerk 3, Fortsetzung

ndv:  SET
      L     DBB  421                    // Laden WortAnzTog_Eing zur Extrahierung der
                                           Kopfnummer
      L     W#16#E
      UW
      SRW   1
      T     #Empf_Kopfnummer
```

[3] Bei der IDENT-Control bleiben die Daten im Ausgangsdatenfeld solange erhalten, bis sie von neuen Daten überschrieben werden, d.h. sie werden nicht nach erfolgter Abholung gelöscht. Bei der Abholung der Daten von Seiten der SPS werden jeweils die sich aktuell im Ausgangsdatenfeld befindlichen Daten übertragen – auch wenn diese dort schon einige Zeit stehen und auch schon mehrmals von der SPS abgeholt wurden.

```
L     DBB  421                    // Laden WortAnzTog_Eing zur Extrahierung des
                                  // empf. Togglebits
L     W#16#1
UW
T     DBB  529                    // Abspeichern des empfangenen Togglebits
```

Mit Hilfe der gespeicherten Kopfnummer wird nun kontrolliert, von welchem Kopf die empfangenen Daten stammen und in welchen Datenbereich sie somit verschoben werden sollen. Stammen sie z.B. von Kopf 1, so erfolgt ein Sprung in Netzwerk 4 „Schreiben der empfangenen Daten in den DB-Bereich von Kopf 1“.

FC100, Netzwerk 3, Fortsetzung

```
L     W#16#1
==I
SPB   trf1
L     #Empf_Kopfnummer
L     W#16#2
==I
SPB   trf2
L     #Empf_Kopfnummer
L     W#16#3
==I
SPB   trf3
L     #Empf_Kopfnummer
L     W#16#4
==I
SPB   trf4
SPA   end
```

Das Netzwerk 4, das den Transfer der empfangenen Daten in den Datenspeicherbereich für Kopf 1 erledigt, soll nun beispielhaft erklärt werden. Die Netzwerke 5-7 für den Transfer der Daten in die Datenbereiche der übrigen Köpfe sind analog zu programmieren.

Zunächst wird nun überprüft, ob das Err_SFC_14- oder das Err_SFC_15-Bit gesetzt ist. Ist dies der Fall, so ist ein Fehler bei der Kommunikation zwischen SPS und der IDENT-Control aufgetreten. Es konnten im Vorfeld keine Daten gelesen oder geschrieben werden. D.h. es kann nicht garantiert werden, dass die nun erhaltenen Daten fehlerfrei sind und zwischendurch keine Datensätze verpasst wurden. Deswegen werden die neu empfangenen Daten solange ignoriert bis die Fehlermeldebits durch Quittierung des Anwenders zurückgesetzt wurden. Es erfolgt ein Sprung zur Speicherung der ersten 4 Byte der empfangenen Daten.
Ist keines der beiden Fehlermeldebits gesetzt, so werden die Daten zusammen mit dem Rückgabewert in den DB-Speicherbereich von Kopf 1 gesichert. Anschließend werden das NeueDatenVorhanden- und das EmpfangenOK-Bit von Kopf 1 gesetzt. Danach erfolgt ein Sprung zu Netzwerk 9 „Sichern der ersten 4 Byte der Empfangsdaten“.

```
FC100, Netzwerk 4, Transfer der empfangenen Daten in DB-Bereich von Kopf 1

trf1: SET
      U(
      O     DBX  133.0                  // Err_SFC14 von Kopf 1?
      O     DBX  133.1                  // Err_SFC15 von Kopf 1?
      )
      SPB   res                         // Daten ignorieren und weiter

      L     DBB  420                    // Laden Befehlscode_Eing aus
                                        // Zwischenspeicher
      T     DBB   60
      L     DBB  421                    // Laden WortAnzTog_Eing aus Zwischenspeicher
      T     DBB   61
      L     DBB  422                    // Laden Status aus Zwischenspeicher
      T     DBB   62
      L     DBB  423                    // Laden Ausfzaehler aus Zwischenspeicher
      T     DBB   63
      L     DBD  424                    // Laden Eing_DW1 aus Zwischenspeicher
      T     DBD   64
      L     DBD  428                    // Laden Eing_DW2 aus Zwischenspeicher
      T     DBD   68
      L     DBD  432                    // Laden Eing_DW3 aus Zwischenspeicher
      T     DBD   72
      L     DBD  436                    // Laden Eing_DW4 aus Zwischenspeicher
      T     DBD   76
      L     DBD  440                    // Laden Eing_DW5 aus Zwischenspeicher
      T     DBD   80
      L     DBD  444                    // Laden Eing_DW6 aus Zwischenspeicher
      T     DBD   84
      L     DBD  448                    // Laden Eing_DW7 aus Zwischenspeicher
      T     DBD   88

      L     DBW  496                    // Laden des Ret_Val_SFC14 aus
                                        // Zwischenspeicher
      T     DBW  136

      SET
      S     DBX  130.6                  // Setzen NeueDatenVorhanden-Bit in DB-
                                        // Bereich Kopf 1
      S     DBX  132.3                  // Setzen EmpfangenOK von Kopf 1
      SPA   res                         // Sprung in Netzwerk 9
```

Wie bereits erwähnt, sind die Netzwerke 5-7 für den Transfer in den Datenbereich der Köpfe 2-4 analog zu erstellen.

War der Rückgabewert der SFC 14 nach deren Ausführung ungleich Null, so ist ein Fehler aufgetreten und es wird in den Fehlerbehandlungsteil gesprungen. Dort wird zuerst das Err_SFC_14-Bit im Zwischenspeicher gesetzt. Anschließend wird für jeden Kopf überprüft, ob dieser vorhanden und ob an diesem gerade ein Befehl (single oder enhanced) aktiv ist. Ist dies der Fall, so kann aufgrund des aufgetretenen Kommunikationsfehlers keine Garantie mehr übernommen werden, dass alle Daten korrekt empfangen wurden bzw. dass kein Datensatz verloren gegangen ist. Deshalb werden alle Köpfe mit aktivem Befehl in den Fehlerzustand versetzt, d.h. das Error- und das Err_SFC_14-Bit gesetzt. Diese Fehlermeldungen müssen vom Anwender nun zuerst quittiert werden, bevor der normale Betrieb wieder aufge-

nommen werden kann. Damit ist sichergestellt, dass der Anwender zum einen den Kommunikationsfehler in der Anlage bemerkt hat und zum anderen wird Folgefehlern aufgrund von verlorenen Datensätzen, falschen Daten o.ä. vorgebeugt.

```
FC100, Netzwerk 8, Fehlerbehandlung bei SFC_14-Fehler

err:  SET
      S     DBX  493.0                  // Wenn Fehler, setzen von Fehlermeldebit
                                        // Err_SFC14 im Zwischenspeicher

      SET
      U     DBX  132.0                  // VorhandenTC von Kopf 1 gesetzt
      U(
      O     DBX  133.2                  // EnhBefehlAktiv von Kopf 1?
      O     DBX  133.3                  // SglBefehlAktiv von Kopf 1?
      )
      S     DBX  133.0                  // Setzen Err_SFC14 von Kopf 1
      S     DBX  132.1                  // Setzen Error-Bit von Kopf 1

      SET
      U     DBX  222.0                  // VorhandenTC von Kopf 2 gesetzt
      U(
      O     DBX  223.2                  // EnhBefehlAktiv von Kopf 2?
      O     DBX  223.3                  // SglBefehlAktiv von Kopf 2?
      )
      S     DBX  223.0                  // Setzen Err_SFC14 von Kopf 2
      S     DBX  222.1                  // Setzen Error-Bit von Kopf 2

      SET
      U     DBX  312.0                  // VorhandenTC von Kopf 3 gesetzt
      U(
      O     DBX  313.2                  // EnhBefehlAktiv von Kopf 3?
      O     DBX  313.3                  // SglBefehlAktiv von Kopf 3?
      )
      S     DBX  313.0                  // Setzen Err_SFC14 von Kopf 3
      S     DBX  312.1                  // Setzen Error-Bit von Kopf 3

      SET
      U     DBX  402.0                  // VorhandenTC von Kopf 4 gesetzt
      U(
      O     DBX  403.2                  // EnhBefehlAktiv von Kopf 4?
      O     DBX  403.3                  // SglBefehlAktiv von Kopf 4?
      )
      S     DBX  403.0                  // Setzen Err_SFC14 von Kopf 4
      S     DBX  402.1                  // Setzen Error-Bit von Kopf 4
      SPA   end
```

In Netzwerk 9 werden die ersten 4 Bytes der empfangenen Daten in den Alt-...-Feldern im Zwischenspeicher des DB gespeichert. Dies ist erforderlich, damit erkannt werden kann, wenn neue Daten vorliegen. Die ersten 4 Bytes der nächsten empfangenen Daten werden dann mit den eben abgespeicherten verglichen. Bei Ungleichheit in mindestens einem Bit liegen neue Daten vor.

FC100, Netzwerk 9, Sichern der ersten 4 Bytes in DB

```
res:  L     DBB  420                  // Sichern von Befehlscode_Eing aus
                                      // Zwischenspeicher
      T     DBB  501                  // in Alt_Befehlscode_Eing in
                                      // Zwischenspeicher
      L     DBB  421                  // Sichern von WortAnzTog_Eing aus
                                      // Zwischenspeicher
      T     DBB  502                  // in Alt_WortAnzTog_Eing in Zwischenspeicher
      L     DBB  422                  // Sichern von Status aus Zwischenspeicher
      T     DBB  503                  // in Alt_Status in Zwischenspeicher
      L     DBB  423                  // Sichern von Ausfzaehler aus
                                      // Zwischenspeicher
      T     DBB  504                  // in Alt_Ausfzaehler in Zwischenspeicher

end:  NOP   0
```

5.4.14 Ablauf des Bausteins COMS (FC 101)

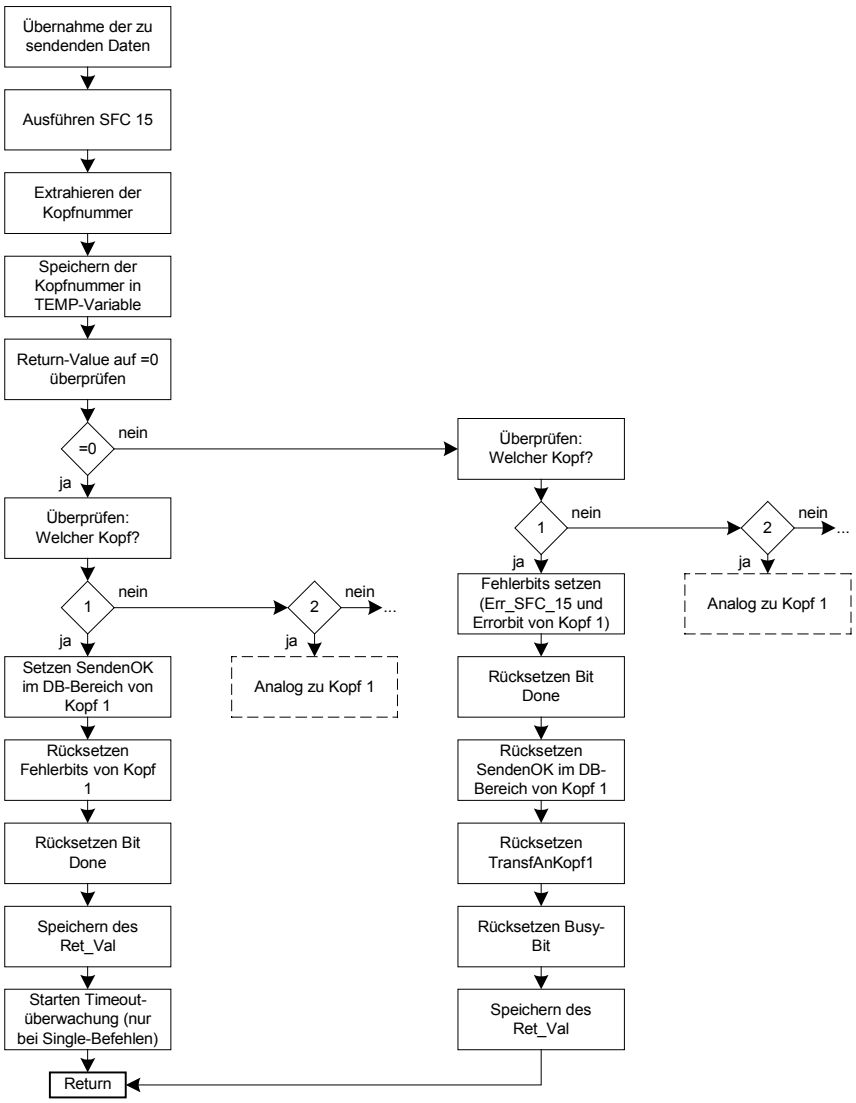

Abb. 24: Ablaufplan des Bausteins COMS

Auch in diesem Baustein sind zunächst noch einige Deklarationen nötig. Dazu gehört z.B. der Parameter „Adresse“, mit dem die Adresse der IDENT-Control-Einheit in der SPS (hier im Beispiel die 100_{hex}) wiederum vom FB 10 automatisch übergeben wird. Die darauffolgende Struktur „Ausg_Daten“ entspricht der im UDT 100 definierten Struktur der Ausgangsdaten, die bereits auch im gesamten Instanzen-DB verwendet wird. An dieser Stelle dient sie zur Entgegennahme der zu sendenden Daten.

Die zu sendenden Daten werden anschließend im Datenbereich „Puffer“ vom Typ UDT 100 zwischengespeichert, da die Datenversorgung der später verwendeten SFC 15 aus IN-Variablen nicht möglich ist und stattdessen Temp-Variablen erforderlich sind. Der Datentyp UDT 100 wurde an dieser Stelle aus Gründen der Einfachheit gewählt. Sonst hätte dieselbe Struktur wie im IN-Variablen-Deklarationsteil „Ausg_Daten“ noch einmal deklariert werden müssen. Dort wurde übrigens jedoch bewusst auf diese Vereinfachung verzichtet, da sonst immens viele Parameterdaten über- bzw. angegeben werden müssten, da der Datenbereich des UDT ja wesentlich größer ist als die zu sendenden Daten.
Die beiden Temp-Variablen Kopfnummer und Ret_Val_SFC15 dienen ebenfalls der Zwischenspeicherung der ihrem Namen entsprechenden Daten.

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Adresse	WORD		
2.0	in	Ausg_Daten	STRUCT		
+0.0	in	Befehlscode_Ausg	BYTE		
+1.0	in	WortAnzTog_Ausg	BYTE		
+2.0	in	Wortadr_High	BYTE		
+3.0	in	Wortadr_Low	BYTE		
+4.0	in	Ausg_DW1	DWORD		
+8.0	in	Ausg_DW2	DWORD		
+12.0	in	Ausg_DW3	DWORD		
+16.0	in	Ausg_DW4	DWORD		
+20.0	in	Ausg_DW5	DWORD		
+24.0	in	Ausg_DW6	DWORD		
+28.0	in	Ausg_DW7	DWORD		
=32.0	in	END_STRUCT			
34.0	in	Datenbaustein	INT		
out					
in_out					
0.0	temp	zugeord_DB	INT		
2.0	temp	puffer	"Kopf Datenstruktur"		
92.0	temp	Kopfnummer	BYTE		
94.0	temp	Ret_Val_SFC15	INT		

Tabelle 7: Deklarationsteil des Bausteins COMS

Damit auch von diesem Baustein aus der Zugriff auf den gemeinsam verwendeten Datenbaustein erfolgen kann, muss dieser zunächst geöffnet werden. Dann werden die zu sendenden Daten übernommen, d.h. die im Aufruf des Bausteins COMS angegebenen Daten werden in den Pufferspeicher des Bausteins COMS geladen und dort zwischengespeichert. Selbstverständlich müssen die Schreibdaten vorher auch im angegebenen Datenbereich enthalten sein. Die folgende Übersicht zeigt die Schreibdatenbereiche für die einzelnen Köpfe:

Kopf	Datenbereich im Instanzen-DB

1	DBD 96 bis incl. DBD 120
2	DBD 186 bis incl. DBD 210
3	DBD 276 bis incl. DBD 300
4	DBD 366 bis incl. 390

Tabelle 8: Die Schreibdatenbereiche für die einzelnen Köpfe

Anschließend wird die Systemfunktion SFC 15 zur Übertragung der Daten an die IDENT-Control ausgeführt. Die hierfür erforderlichen Parameter sind ähnlich wie bei Ausführung der SFC 14 die Adresse der IDENT-Control-Einheit, das betreffende Feld im Datenbaustein für den Rückgabewert und natürlich die zu sendenden Daten im Pufferspeicher.

```
FC101, Netzwerk 1: Übernahme der zu sendenden Daten

      L     #Datenbaustein
      T     #zugeord_DB

      L     #Ausg_Daten.Befehlscode_Ausg
      T     #puffer.Ausgangsdaten.Befehlscode_Ausg
      L     #Ausg_Daten.WortAnzTog_Ausg
      T     #puffer.Ausgangsdaten.WortAnzTog_Ausg
      L     #Ausg_Daten.Wortadr_High
      T     #puffer.Ausgangsdaten.Wortadr_High
      L     #Ausg_Daten.Wortadr_Low
      T     #puffer.Ausgangsdaten.Wortadr_Low
      L     #Ausg_Daten.Ausg_DW1
      T     #puffer.Ausgangsdaten.Ausg_DW1
      L     #Ausg_Daten.Ausg_DW2
      T     #puffer.Ausgangsdaten.Ausg_DW2
      L     #Ausg_Daten.Ausg_DW3
      T     #puffer.Ausgangsdaten.Ausg_DW3
      L     #Ausg_Daten.Ausg_DW4
      T     #puffer.Ausgangsdaten.Ausg_DW4
      L     #Ausg_Daten.Ausg_DW5
      T     #puffer.Ausgangsdaten.Ausg_DW5
      L     #Ausg_Daten.Ausg_DW6
      T     #puffer.Ausgangsdaten.Ausg_DW6
      L     #Ausg_Daten.Ausg_DW7
      T     #puffer.Ausgangsdaten.Ausg_DW7

      CALL  "DPWR_DAT"
       LADDR  :=#Adresse
       RECORD :=#puffer.Ausgangsdaten
       RET_VAL:=#Ret_Val_SFC15
```

Nun wird wiederum die Kopfnummer aus den Daten extrahiert und in der Temp-Variable „Kopfnummer“ abgespeichert. Wie im Baustein COMR wird auch hier überprüft, ob der Rückgabewert der SFC 15 gleich Null ist, und die Ausführung somit fehlerfrei erfolgt ist. Wenn nicht ist ein Fehler aufgetreten und es folgt ein Sprung in die Fehlerbehandlung des Bausteins (siehe vorletzter Absatz in diesem Kapitel).

Ist der Rückgabewert gleich Null, wird mit Hilfe der vorher in der Temp-Variable abgespeicherten Kopfnummer überprüft, an welchen Kopf die Daten gesendet wurden und in welchem Teil somit auch die Meldebits zu setzen sind. In diesen Teil in Netzwerk 3 wird dann auch gesprungen.

```
FC101, Netzwerk 2: Überprüfen Ret_Val und Kopfnummer

      L     #puffer.Ausgangsdaten.WortAnzTog_Ausg    // Dekodieren der Kopfnummer
      L     W#16#E
      UW
      SRW   1
      T     #Kopfnummer                 // Schreiben in Variable

      AUF   DB [#zugeord_DB]

      L     #Ret_Val_SFC15              // Ret_Val auf "0" überprüfen
      L     W#16#0
      <>I
      SPB   err

      L     #Kopfnummer                 // Überprüfen, welcher Kopf und Sprung an
                                        // entsprechende Stelle
      L     W#16#1
      ==I
      SPB   ok1
      L     #Kopfnummer
      L     W#16#2
      ==I
      SPB   ok2
      L     #Kopfnummer
      L     W#16#3
      ==I
      SPB   ok3
      L     #Kopfnummer
      L     W#16#4
      ==I
      SPB   ok4
```

Wurden die Daten z.B. an Kopf 1 übertragen, so wird zuerst das SendenOK-Bit für Kopf 1 gesetzt. Anschließend werden die Fehlermeldebits und das Done-Bit zurückgesetzt. Letzteres erfolgt deswegen, weil jetzt ein neuer Befehl an den Kopf gesendet wurde und dieser nun bearbeitet wird und nicht schon fertig bearbeitet sein kann, was ja mit dem Done-Bit signalisiert wird. Danach wird der Rückgabewert in das entsprechende Feld im Datenbaustein für Kopf 1 gesichert.

```
FC101, Netzwerk 3: Setzen der OK-Bits bei fehlerfreier Ausführung der SFC 15

ok1:  SET
      S     DBX  132.4                  // Setzen von Bit SendenOK von Kopf 1
      R     DBX  133.1                  // Rücksetzen von Err_SFC_15 von Kopf 1
      R     DBX  493.0                  // Rücksetzen Err_SFC_14 im Zwischenspeicher
      R     DBX  132.6                  // Rücksetzen Done von Kopf 1
      L     #Ret_Val_SFC15
      T     DBW  138                    // Schreiben des Ret_Val in DB-Bereich für
                                        //  Kopf 1
```

Bei einem Singlebefehl muss noch die Timeout-Überwachung gestartet werden. Dazu erfolgt zuerst die Kontrolle, ob es sich bei dem eben gesendeten Befehl überhaupt um einen Single-Befehl handelt. Ist dies der Fall, so wird das TimeoutÜberwAktiv-Bit gesetzt und der vom Anwender angegebene Wert für den Timeouttimer geladen. Dieses ist der Startwert für den Timeouttimer für Kopf 1. Damit an dieser Stelle nicht fest ein Timer belegt wird und der Baustein somit nicht mehr multiinstanzfähig wäre, ist in DBW 8 (Datenbausteinwort 8) der vom Anwender diesem Kopf zugeteilte Timer eingetragen und wird hier gestartet.

```
FC101, Netzwerk 3, Fortsetzung

// Starten TimeoutUeberwAktiv (bei Singlebefehlen)
      CLR
      U     DBX  133.3                  // SglBefehlAktiv von Kopf 1 gesetzt?
      S     DBX  130.0                  // Setzen TimeoutUeberwAktiv von Kopf 1

      U     DBX  130.0                  // Wenn TimeoutUeberwAktiv von Kopf 1
                                        // gesetzt, dann
      L     DBW    6                    // Laden TimeoutUeberwAktiv-Startwert (IN-
                                        // Variable)
      SI    T [DBW 8]                   // Timer für Kopf 1
      SPA   end
```

Für die übrigen Köpfe erfolgt dieser Ablauf genauso, nur müssen eben die diesen Köpfen zugeordneten Bereiche im Datenbaustein verwendet werden.

```
FC101, Netzwerk 3, Fortsetzung

ok2:  SET
      S     DBX  222.4                  // Setzen von Bit SendenOK von Kopf 2
      R     DBX  223.1                  // Rücksetzen des Err_SFC_15 von Kopf 2
      R     DBX  493.0                  // Rücksetzen Err_SFC_14 im Zwischenspeicher
      R     DBX  222.6                  // Rücksetzen Done von Kopf 2
      L     #Ret_Val_SFC15
      T     DBW  228                    // Schreiben des Ret_Val in DB-Bereich für
                                        // Kopf 2
// Starten TimeoutUeberwAktiv (bei Singlebefehlen)
      CLR
      U     DBX  223.3                  // SglBefehlAktiv von Kopf 2 gesetzt?
      S     DBX  220.0                  // Setzen TimeoutUeberwAktiv von Kopf 2

      U     DBX  220.0                  // Wenn TimeoutUeberwAktiv von Kopf 2
                                        // gesetzt, dann
      L     DBW    6                    // Laden TimeoutUeberwAktiv-Startwert (IN-
                                        // Variable)
      SI    T [DBW 10]                  // Timer für Kopf 2
      SPA   end

ok3:  SET
      S     DBX  312.4                  // Setzen von Bit SendenOK von Kopf 3
      R     DBX  313.1                  // Rücksetzen von Err_SFC_15 von Kopf 3
      R     DBX  493.0                  // Rücksetzen Err_SFC_14 im Zwischenspeicher
      R     DBX  312.6                  // Rücksetzen Done von Kopf 3
      L     #Ret_Val_SFC15

      T     DBW  318                    // Schreiben des Ret_Val in DB-Bereich für
                                        // Kopf 3
```

```
// Starten TimeoutUeberwAktiv (bei Singlebefehlen)
      CLR
      U     DBX  313.3                  // SglBefehlAktiv von Kopf 3 gesetzt?
      S     DBX  310.0                  // Setzen TimeoutUeberwAktiv von Kopf 3

      U     DBX  310.0                  // Wenn TimeoutUeberwAktiv von Kopf 3
                                        // gesetzt, dann
      L     DBW    6                    // Laden TimeoutUeberwAktiv-Startwert (IN-
                                        // Variable)
      SI    T [DBW 12]                  // Timer für Kopf 3
      SPA   end

ok4:  SET
      S     DBX  402.4                  // Setzen von Bit SendenOK von Kopf 4
      R     DBX  403.1                  // Rücksetzen von Err_SFC_15 von Kopf 4
      R     DBX  493.0                  // Rücksetzen Err_SFC_14 im Zwischenspeicher
      R     DBX  402.6                  // Rücksetzen Done von Kopf 4
      L     #Ret_Val_SFC15
      T     DBW  408                    // Schreiben des Ret_Val in DB-Bereich für
                                        // Kopf 4
// Starten TimeoutUeberwAktiv (bei Singlebefehlen)
      CLR
      U     DBX  403.3                  // SglBefehlAktiv von Kopf 4 gesetzt?
      S     DBX  400.0                  // Setzen TimeoutUeberwAktiv von Kopf 4

      U     DBX  400.0                  // Wenn TimeoutUeberwAktiv von Kopf 4
                                        // gesetzt, dann
      L     DBW    6                    // Laden TimeoutUeberwAktiv-Startwert (IN-
                                        // Variable)
      SI    T [DBW 14]                  // Timer für Kopf 4
      SPA   end
```

Ist der Rückgabewert der SFC 15 ungleich Null, so wird die folgende Fehlerbehandlungsroutine durchlaufen:

Zunächst wird wiederum mit Hilfe der Variablen „Kopfnummer“ festgestellt, an welchen Kopf die Daten hätten gesendet werden sollen. Für diesen Kopf werden anschließend alle erforderlichen Bits gesetzt bzw. rückgesetzt.

```
FC101, Netzwerk 4: Fehlerbehandlung

err:  L     #Kopfnummer               // Überprüfen, welcher Kopf und Sprung an
      L     W#16#1                    // entsprechende Stelle
      ==I
      SPB   err1
      L     #Kopfnummer
      L     W#16#2
      ==I
      SPB   err2
      L     #Kopfnummer
      L     W#16#3
      ==I
      SPB   err3
      L     #Kopfnummer
      L     W#16#4
      ==I
      SPB   err4
```

Zuerst werden die Fehlermeldebits Error und Err_SFC_15, sowie das Done-Bit gesetzt. Anschließend werden das SendenOK-, das TransfAnKopf1- und das Busy-Bit zurückgesetzt.

Danach erfolgt die Speicherung des Rückgabewerts im betreffenden Speicherbereich im Datenbaustein.

```
FC101, Netzwerk 4, Fortsetzung

err1: SET
      S     DBX  132.1                 // Setzen des Errorbits von Kopf 1
      S     DBX  133.1                 // Setzen von Err_SFC15 von Kopf 1
      S     DBX  132.6                 // Setzen von Done von Kopf 1
      R     DBX  132.4                 // Rücksetzen SendenOK von Kopf 1
      R     DBX  527.0                 // Rücksetzen TransfAnKopf1
      R     DBX  132.7                 // Rücksetzen Busy von Kopf 1
      L     #Ret_Val_SFC15
      T     DBW  138                   // Schreiben des Ret_Val in DB-Bereich für
                                       // Kopf 1
      SPA   end
```

Für die anderen Köpfe ist der Ablauf genau analog.

```
FC101, Netzwerk 4, Fortsetzung

err2: SET
      S     DBX  222.1                 // Setzen des Errorbits von Kopf 2
      S     DBX  223.1                 // Setzen von Err_SFC15 von Kopf 2
      S     DBX  222.6                 // Setzen von Done von Kopf 2
      R     DBX  222.4                 // Rücksetzen von Bit SendenOK von Kopf 2
      R     DBX  527.1                 // Rücksetzen TransfAnKopf2
      R     DBX  222.7                 // Rücksetzen Busy von Kopf 2
      L     #Ret_Val_SFC15
      T     DBW  228                   // Schreiben des Ret_Val in DB-Bereich für
                                       // Kopf 2
      SPA   end
err3: SET
      S     DBX  312.1                 // Setzen des Errorbits von Kopf 3
      S     DBX  313.1                 // Setzen von Err_SFC15 von Kopf 3
      S     DBX  312.6                 // Setzen von Done von Kopf
      R     DBX  312.4                 // Rücksetzen von Bit SendenOK von Kopf 3
      R     DBX  527.2                 // Rücksetzen TransfAnKopf3
      R     DBX  312.7                 // Rücksetzen Busy von Kopf 3
      L     #Ret_Val_SFC15
      T     DBW  318                   // Schreiben des Ret_Val in DB-Bereich für
                                       // Kopf 3
      SPA   end
err4: SET
      S     DBX  402.1                 // Setzen des Errorbits von Kopf 4
      S     DBX  403.1                 // Setzen von Err_SFC15 von Kopf 4
      S     DBX  402.6                 // Setzen von Done von Kopf 4
      R     DBX  402.4                 // Rücksetzen von Bit SendenOK von Kopf 4
      R     DBX  527.3                 // Rücksetzen TransfAnKopf4
      R     DBX  402.7                 // Rücksetzen Busy von Kopf 4
      L     #Ret_Val_SFC15
      T     DBW  408                   // Schreiben des Ret_Val in DB-Bereich für
                                       // Kopf 4
      SPA   end

end:  NOP 0
```

5.4.15 Ablauf des Bausteins ANALYSE (FC 102)

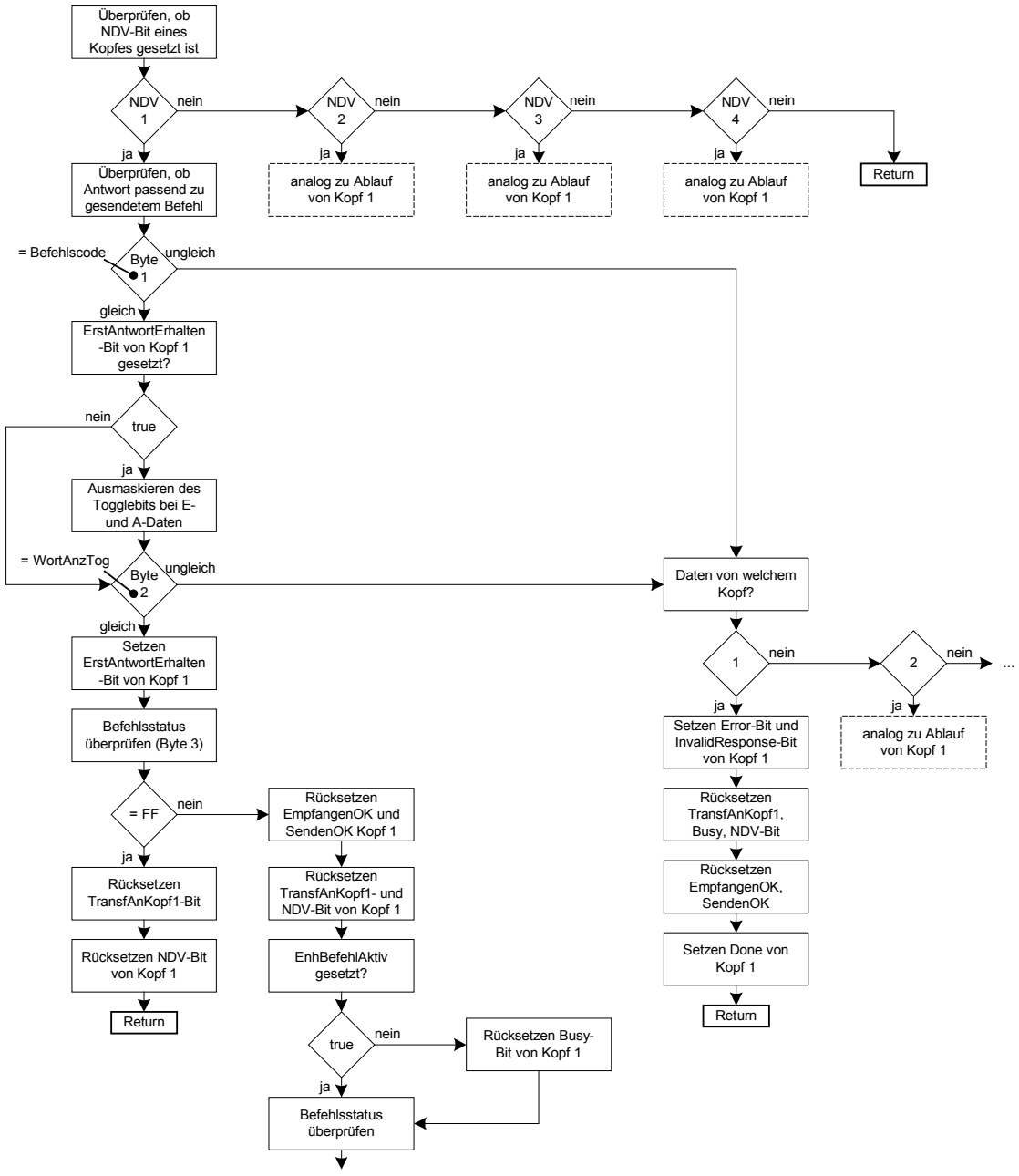

Abb. 25: Teil 1 des Ablaufplans des Bausteins ANALYSE

Der Deklarationsteil des Bausteins ANALYSE ist nur sehr klein. Alle darin enthaltenen Variablen erfüllen denselben Zweck, den sie in den anderen Bausteinen ebenfalls inne haben und werden an dieser Stelle nicht weiter erläutert.

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Datenbaustein	INT		
out					
in out					
0.0	temp	zugeord_DB	INT		
2.0	temp	Kopfnummer	BYTE		

Tabelle 9: Deklarationsteil des Bausteins ANALYSE

Auch dem Baustein ANALYSE muss Zugriff auf den gemeinsam genutzten Datenbaustein gewährt und dieser somit geöffnet werden.
Wurden von einem Kopf neue Daten mit dem Baustein COMR empfangen, so ist das NDV-Bit dieses Kopfes gesetzt und dient als Signal, die Analyse dieser Daten vom Baustein ANALYSE durchführen zu lassen. Die NDV-Bits der einzelnen Köpfe werden nun zu Bausteinbeginn überprüft. Ist eines der Bits gesetzt, erfolgt ein Sprung in den betreffenden Programmteil. Ist keines der NDV-Bits gesetzt, wird an das Ende des Bausteins gesprungen.

```
FC102, Netzwerk 1: Überprüfen der NDV-Bits der einzelnen Köpfe

      L     #Datenbaustein
      T     #zugeord_DB
      AUF   DB [#zugeord_DB]

      SET
      U     DBX  130.6                  // NDV-Bit von Kopf 1 gesetzt?
      SPB   an1
      SET
      U     DBX  220.6                  // NDV-Bit von Kopf 2 gesetzt?
      SPB   an2
      SET
      U     DBX  310.6                  // NDV-Bit von Kopf 3 gesetzt?
      SPB   an3
      SET
      U     DBX  400.6                  // NDV-Bit von Kopf 4 gesetzt?
      SPB   an4
      SPA   end
```

Bei der Analyse der Daten wird zuerst überprüft, ob die empfangenen Daten überhaupt zu dem gesendeten Befehl passen, d.h. ob die ersten beiden Bytes (Befehlscode und WortAnzTog) des gesendeten Befehls mit der eingegangenen Antwort übereinstimmen. Die Kontrolle des ersten Bytes ist dabei problemlos durch Vergleichen möglich; bei dem zweiten Byte kommt die im Kapitel „3.4.4 Die Bedeutung des Togglebits“ beschriebene Problematik zum Tragen. Wie bereits schon erwähnt, wird in diesem Programm das Togglebit nach der ersten Antwort auf einen Befehl ausmaskiert. D.h. wenn das ErstAntwortErhalten-Bit gesetzt ist, wird sowohl bei den Eingangs-, als auch bei den Ausgangsdaten das Togglebit ausmaskiert. Damit ist das zweite Byte ebenfalls durch einfaches Vergleichen kontrollierbar. Ist die

erste Antwort auf einen Befehl noch nicht erfolgt, so wird das Togglebit nicht ausmaskiert und der Vergleich erfolgt direkt.
Ist einer der beiden Vergleiche negativ, so wird in die Fehlerbehandlung des Bausteins ANALYSE gesprungen.
Sind beide positiv, wird das ErstAntwortErhalten-Bit gesetzt, da ja nun die erste Antwort auf den Befehl erhalten wurde.

```
FC102, Netzwerk 2: Analyse der Daten von Kopf 1

an1:  L     DBB   60                  // Laden Befehlscode_Eing von Kopf 1
      L     DBB   92                  // Laden Befehlscode_Ausg von Kopf 1
      <>I
      SPB   err

      SET
      UN    DBX  133.4                // ErstAntwortErhalten von Kopf 1
      SPB   tg1

      L     DBB   61                  // Wenn ErstAntwortErhalten gesetzt, dann
                                      // Togglebit ausmaskieren
      L     B#16#FE
      UW
      T     DBB   61

      L     DBB   93                  // Auch in AusgDatenfeld ausmaskieren, damit
                                      // die Überprüfung wieder funktioniert
      L     B#16#FE
      UW
      T     DBB   93

tg1:  L     DBB   61                  // Laden WortAnzTog_Eing von Kopf 1
      L     DBB   93                  // Laden WortAnzTog_Ausg von Kopf 1
      <>I
      SPB   err

      SET
      S     DBX  133.4                // Setzen ErstAntwortErhalten von Kopf 1
```

Als nächstes erfolgt die Überprüfung des Status, der im 3. Byte der empfangenen Daten enthalten ist. Die folgende Tabelle gibt einen kurzen Überblick über die wichtigsten Stati und welche Bedeutung mit ihnen einhergeht. Diese Tabelle ist in etwas längerer und ausführlicherer Form auch im Handbuch zur IDENT-Control ganz am Ende zu finden.

Status	Bedeutung
00_h	Befehl wurde fehlerfrei ausgeführt
FF_h	Befehl in Bearbeitung
02_h	Einschaltmeldung, Reset wurde ausgeführt
04_h	Falscher bzw. unvollständiger Befehl oder Parameter nicht im gültigen Bereich
05_h	Kein Datenträger im Erfassungsbereich
06_h	Hardwarefehler, z.B. bei Selbsttest oder Lesekopf defekt
07_h	Softwarefehler

Tabelle 10: Status-/Fehlermeldungen der IDENT-Control

Zuerst erfolgt die Kontrolle, ob der Status gleich FF_h, der Befehl somit also noch in Bearbeitung ist. Wenn ja, wird in den Programmteil sff des jeweiligen Kopfes gesprungen.
Wenn nicht, werden die Bits EmpfangenOK, SendenOK, TransfAnKopf1 und NeueDatenVorhanden zurückgesetzt. Handelt es sich bei den eingegangenen Daten nicht um eine Antwort von einem Enhanced-Befehl, so wird zusätzlich noch das Busy-Bit zurückgesetzt. Bei einem aktiven Enhanced-Befehl muss das Busy-Bit natürlich gesetzt bleiben, da der Befehl ja immer noch aktiv ist.

```
FC102, Netzwerk 2, Fortsetzung

      L     DBB   62                  // Laden Eingangsdaten.Status von Kopf 1
      L     B#16#FF
      ==I
      SPB   sff1                      // Wenn gleich FFh, Befehl noch in
                                      // Bearbeitung

      R     DBX  132.3                // Rücksetzen von EmpfangenOK von Kopf 1
      R     DBX  132.4                // Rücksetzen von SendenOK von Kopf 1
      R     DBX  527.0                // Rücksetzen TransfAnKopf1-Bit
      R     DBX  130.6                // Rücksetzen NDV-Bit von Kopf 1

      SET                             // Wenn kein Enh-Befehl, dann
      UN    DBX  133.2
      R     DBX  132.7                // Rücksetzen Busy von Kopf 1
```

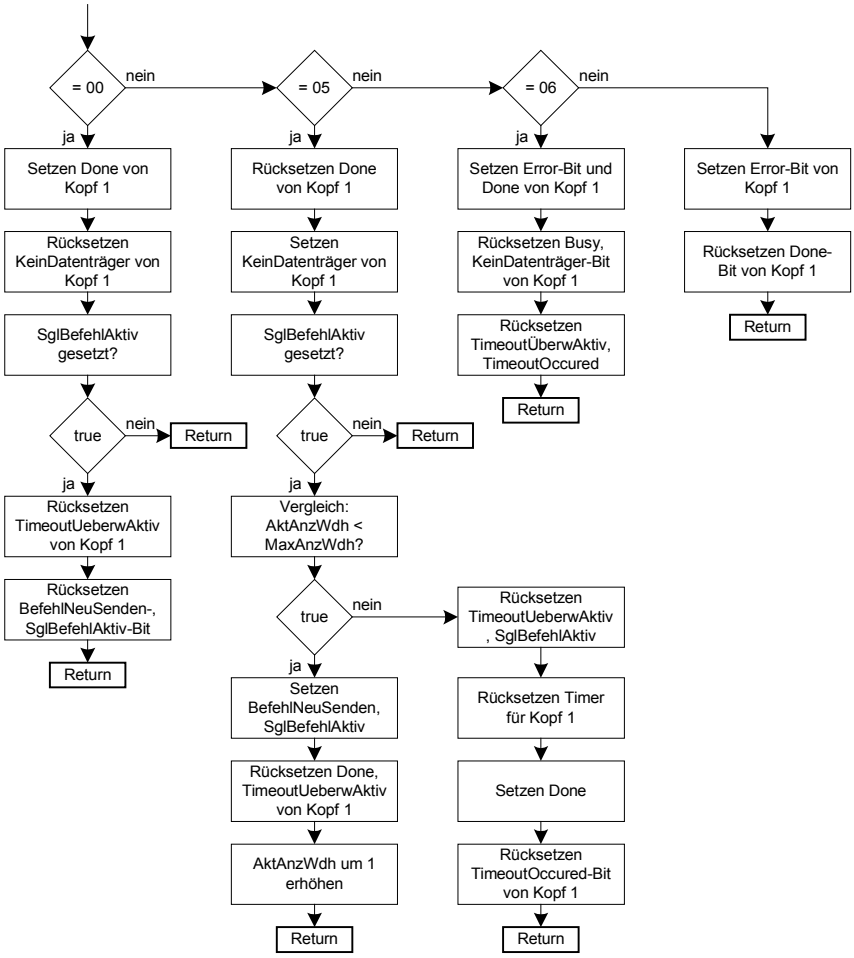

Abb. 26: Teil 2 des Ablaufplans des Bausteins ANALYSE

Anschließend wird überprüft, ob der Status 00_h, 05_h oder 06_h ist. Je nachdem erfolgt dann ein Sprung zur Markierung ok1 (Befehlsausführung war erfolgreich), kdv1 (kein Datenträger vorhanden) oder knv1 (Kopf nicht vorhanden bzw. Hardwarefehler). Handelt es sich um einen anderen Status, so kann es nur noch eine weitere Fehlermeldung sein und diese wird entsprechend durch Setzen des Error- und Done-Bits signalisiert.

```
FC102, Netzwerk 2, Fortsetzung

      L     DBB   62                    // Laden Eingangsdaten.Status von Kopf 1
      L     B#16#0
      ==I
      SPB   ok1                         // Wenn gleich 00h, Befehl fehlerfrei
                                        // ausgeführt

      L     DBB   62                    // Laden Eingangsdaten.Status von Kopf 1
      L     B#16#5
      ==I
      SPB   kdv1                        // Wenn gleich 05h, kein Datenträger
                                        // vorhanden

      L     DBB   62                    // Laden Eingangsdaten.Status von Kopf 1
      L     B#16#6
      ==I
      SPB   knv1                        // Wenn gleich 06h, Kopf nicht oder nicht
                                        // mehr vorhanden

// Wenn Status weder 0, 5, 6 oder FF ist, dann ist ein anderer Fehler aufgetreten
      SET
      S     DBX  132.1                  // Setzen Error-Bit von Kopf 1
      S     DBX  132.6                  // Setzen Done-Bit von Kopf 1
      SPA   end
```

Im Programmteil sff wird lediglich das TransfAnKopf1-Bit zurückgesetzt, da nun durch Einholen der Antwort mit Status FF_h bekannt ist, dass der Befehl beim Kopf angekommen ist und auch ausgeführt wird. Außerdem wird das NeueDatenVorhanden-Bit zurückgesetzt, da dieser Datensatz hiermit komplett ausgewertet wurde. Danach wird an das Ende des Bausteins ANALYSE gesprungen.

```
FC102, Netzwerk 2, Fortsetzung

sff1: SET
      R     DBX  527.0                  // Rücksetzen TransfAnKopf1-Bit
      R     DBX  130.6                  // Rücksetzen NDV-Bit von Kopf 1
      SPA   end
```

Wurde der Befehl fehlerfrei ausgeführt, so wird der Code im Teil ok1 bearbeitet. Dabei wird zunächst das Done-Bit gesetzt und das KeinDatenträger-Bit zurückgesetzt – letzteres weil ja nun ein Datenträger gelesen wurde. Dann wird kontrolliert, ob derzeit ein Single-Befehl aktiv ist. Wenn nein, wird ans Ende des Bausteins gesprungen, wenn ja, erfolgt ein Sprung zum Programmteil oto1.

Im Programmteil oto 1 wird die Timeout-Überwachung durch Rücksetzen des TimeoutÜberwAktiv-Bits abgeschaltet. Danach wird das BefehlNeuSenden- und das SingleBefehlAktiv-Bit zurückgesetzt, da ja nun der Single-Befehl einmal erfolgreich ausgeführt wurde und damit beendet ist. Abschließend erfolgt ein Sprung ans Bausteinende.

```
FC102, Netzwerk 2, Fortsetzung

ok1:  SET
      S     DBX  132.6                  // Setzen Done von Kopf 1
      R     DBX  132.5                  // Rücksetzen KeinDatentraeger von Kopf 1

      SET
      U     DBX  133.3                  // SglBefehlAktiv-Bit von Kopf 1
      SPB   oto1
      SPA   end

oto1: SET
      R     DBX  130.0                  // Rücksetzen TimeoutUeberwAktiv von Kopf 1
      R     DBX  130.1                  // Rücksetzen BefehlNeuSenden von Kopf 1
      R     DBX  133.3                  // Rücksetzen SglBefehlAktiv von Kopf 1
      SPA   end
```

Im Fall der Statusmeldung 05_h wird in den Programmteil kdv1 gesprungen. Dort wird das KeinDatenträgerVorhanden-Bit von Kopf 1 gesetzt. Das Done-Bit wird zurückgesetzt, da dieser Zustand weder bei laufenden Enhanced- noch bei laufenden Single-Befehlen einer Meldung an den Anwender bedarf. Danach erfolgt die Überprüfung auf einen aktiven Single-Befehl und ggf. der Sprung zum Programmteil to1. Handelt es sich nicht um einen Single-Befehl, wird ans Bausteinende gesprungen.

```
FC102, Netzwerk 2, Fortsetzung

kdv1: SET
      R     DBX  132.6                  // Rücksetzen Done von Kopf 1
      S     DBX  132.5                  // Setzen KeinDatentraeger von Kopf 1

      SET
      U     DBX  133.3                  // SglBefehlAktiv-Bit von Kopf 1
      SPB   to1
      SPA   end
```

Im Teil to1 muss geprüft werden, ob die maximale Anzahl an Wiederholungen bereits erreicht wurde. Wenn nein, werden die Bits BefehlNeuSenden und SglBefehlAktiv gesetzt, damit der Single-Befehl im nächsten Programmzyklus erneut an den Kopf gesendet wird. Außerdem werden das Done- und das TimeoutÜberwAktiv-Bit zurückgesetzt - letzteres deswegen, weil im Rahmen des Zeitlimits ja eine Antwort vom Kopf gekommen ist und die Timeoutüberwachung nach dem erneuten Senden neu gestartet wird. Abschließend muss noch die aktuelle Anzahl der Wiederholungen um 1 erhöht werden, bevor an das Ende des Bausteins gesprungen wird.

Ist die maximale Anzahl an Befehlswiederholungen erreicht, so wird im Programmteil ext1 zuerst das TimeoutÜberwAktiv-, das SglBefehlAktiv-, das TimeoutOccured-Bit und der dem Kopf 1 zugeordnete Timer zurückgesetzt. Schließlich wird noch das Done-Bit gesetzt, damit

der Anwender diese Daten nun auswerten bzw. das Vorhandensein der neuen Daten erkennen kann.

```
FC102, Netzwerk 2, Fortsetzung

to1:  SET
      L     DBW  124                  // Laden AktAnzWdh von Kopf 1
      L     DBW  126                  // Laden MaxAnzWdh
      <D
      SPBN  ext1

      SET
      S     DBX  130.1                // Setzen BefehlNeuSenden von Kopf 1
      S     DBX  133.3                // Setzen SglBefehlAktiv von Kopf 1
      R     DBX  130.0                // Rücksetzen TimeoutUeberwAktiv von Kopf 1
      R     DBX  132.6                // Rücksetzen Done von Kopf 1
      L     DBW  124                  // Laden AktAnzWdh
      INC   1                         // Wert um 1 erhöhen
      T     DBW  124
      SPA   end

ext1: SET
      R     T [DBW 8]                 // Timer für Kopf 1 zurücksetzen
      R     DBX  130.0                // Rücksetzen TimeoutUeberwAktiv von Kopf 1
      R     DBX  133.3                // Rücksetzen SglBefehlAktiv von Kopf 1
      S     DBX  132.6                // Setzen Done von Kopf 1
      R     DBX  132.2                // Rücksetzen TimeoutOccured von Kopf 1
      SPA   end
```

Wurde mit der Antwort der Status 06_h geliefert, geht es im Programmteil knv1 weiter. Dort werden das Fehlermeldebit Error und das Done-Bit gesetzt, damit der Fehlerzustand dem Anwender gemeldet wird. Dann werden die Bits KeinDatenträger, Busy, TimeoutÜberwAktiv und das TimeoutOccured-Bit zurückgesetzt. Anschließend erfolgt wieder der Sprung ans Bausteinende.

```
FC102, Netzwerk 2, Fortsetzung

knv1: SET
      S     DBX  132.1                // Setzen Error-Bit von Kopf 1
      S     DBX  132.6                // Setzen Done von Kopf 1
      R     DBX  132.5                // Rücksetzen KeinDatentraeger von Kopf 1
      R     DBX  132.7                // Rücksetzen Busy von Kopf 1
      R     DBX  130.0                // Rücksetzen TimeoutUeberwAktiv von Kopf 1
      R     DBX  132.2                // Rücksetzen TimeoutOccured von Kopf 1
      SPA   end
```

Für die übrigen Köpfe ist der Ablauf genau analog und in den Netzwerken 3-5 programmiert.

Ist bei der zu Beginn durchgeführten Prüfung herausgekommen, dass Antwortdaten und zuvor gesendeter Befehl gar nicht zusammenpassen, so ist das Durchlaufen der Fehlerbehandlung erforderlich. Diese wird nun im Folgenden dargestellt (Bezug auf Abb. 25, rechter Teil).

Bei der Fehlerbehandlung wird zuerst erkannt, von welchem Kopf die eingegangenen Daten stammen und anschließend in den betreffenden Programmbereich (err1 .. err4) gesprungen. Dort wird dann für diesen Kopf neben dem Error- und dem Done-Bit auch das InvalidResponse-Bit dieses Kopfes gesetzt. Danach werden noch die folgenden Bits zurückgesetzt: TransfAnKopf, Busy, NeueDatenVorhanden, EmpfangenOK und SendenOK.
Dieser Ablauf ist für alle Köpfe identisch – es müssen nur wie gewohnt die entsprechenden DB-Bereiche zugeordnet werden.

```
FC102, Netzwerk 6: Fehlerbehandlung

err:  SET
      U     DBX  130.6                  // NDV-Bit von Kopf 1 gesetzt?
      SPB   err1
      SET
      U     DBX  220.6                  // NDV-Bit von Kopf 2 gesetzt?
      SPB   err2
      SET
      U     DBX  310.6                  // NDV-Bit von Kopf 3 gesetzt?
      SPB   err3
      SET
      U     DBX  400.6                  // NDV-Bit von Kopf 4 gesetzt?
      SPB   err4
      SPA   end
err1: SET
      S     DBX  132.1                  // Setzen Error-Bit von Kopf 1
      S     DBX  130.4                  // Setzen InvalidResponse von Kopf 1
      R     DBX  527.0                  // Rücksetzen TransfAnKopf1-Bit
      R     DBX  132.7                  // Rücksetzen Busy-Bit von Kopf 1
      R     DBX  130.6                  // Rücksetzen NDV-Bit von Kopf 1
      R     DBX  132.3                  // Rücksetzen von EmpfangenOK von Kopf 1
      R     DBX  132.4                  // Rücksetzen von SendenOK von Kopf 1
      S     DBX  132.6                  // Setzen Done von Kopf 1
      SPA   end
err2: SET
      S     DBX  222.1                  // Setzen Error-Bit von Kopf 2
      S     DBX  220.4                  // Setzen InvalidResponse von Kopf 2
      R     DBX  527.1                  // Rücksetzen TransfAnKopf2-Bit
      R     DBX  222.7                  // Rücksetzen Busy-Bit von Kopf 2
      R     DBX  220.6                  // Rücksetzen NDV-Bit von Kopf 2
      R     DBX  222.3                  // Rücksetzen EmpfangenOK von Kopf 2
      R     DBX  222.4                  // Rücksetzen SendenOK von Kopf 2
      S     DBX  222.6                  // Setzen Done von Kopf 2
      SPA   end
err3: SET
      S     DBX  312.1                  // Setzen Error-Bit von Kopf 3
      S     DBX  310.4                  // Setzen InvalidResponse von Kopf 3
      R     DBX  527.2                  // Rücksetzen TransfAnKopf3-Bit
      R     DBX  312.7                  // Rücksetzen Busy-Bit von Kopf 3
      R     DBX  310.6                  // Rücksetzen NDV-Bit von Kopf 3
      R     DBX  312.3                  // Rücksetzen EmpfangenOK von Kopf 3
      R     DBX  312.4                  // Rücksetzen SendenOK von Kopf 3
      S     DBX  312.6                  // Setzen Done von Kopf 3
      SPA   end
err4: SET
      S     DBX  402.1                  // Setzen Error-Bit von Kopf 4
      S     DBX  400.4                  // Setzen InvalidResponse von Kopf 4
      R     DBX  527.3                  // Rücksetzen TransfAnKopf4-Bit
      R     DBX  402.7                  // Rücksetzen Busy-Bit von Kopf 4
      R     DBX  400.6                  // Rücksetzen NDV-Bit von Kopf 4
      R     DBX  402.3                  // Rücksetzen EmpfangenOK von Kopf 4
      R     DBX  402.4                  // Rücksetzen SendenOK von Kopf 4
      S     DBX  402.6                  // Setzen Done von Kopf 4
      SPA   end

end:  NOP   0
```

Damit ist nun der komplette Programmablauf und die Umsetzung in den Step7-Code erläutert.

5.5 Programmaufruf

Beim Aufruf des FB 10 z.B. im OB 1 wird die folgende Parameterliste angezeigt:

```
OB1, Netzwerk 1: Aufruf des FBs Ident-Control-System

CALL  "Ident-Control-System" , "DB für IC_1"
       Adresse_Ident_Control   :=
       Datenbaustein           :=
       MaxAnzWdh               :=
       TimeoutDauer            :=
       TimeoutTimerKopf1       :=
       TimeoutTimerKopf2       :=
       TimeoutTimerKopf3       :=
       TimeoutTimerKopf4       :=
       Datentraegertyp         :=
       DatenFixcode            :=
       SingleEnhanced          :=
       Kopf1Lesen              :=
       Kopf1Schreiben          :=
       Kopf2Lesen              :=
       Kopf2Schreiben          :=
       Kopf3Lesen              :=
       Kopf3Schreiben          :=
       Kopf4Lesen              :=
       Kopf4Schreiben          :=
       Kopf1Quit               :=
       Kopf2Quit               :=
       Kopf3Quit               :=
       Kopf4Quit               :=
       QuitErrorKopf1          :=
       QuitErrorKopf2          :=
       QuitErrorKopf3          :=
       QuitErrorKopf4          :=
       Datentraegerstartadresse:=
       Anzahl32BitWorte        :=
       RueckgabewertKopf1      :=
       RueckgabewertKopf2      :=
       RueckgabewertKopf3      :=
       RueckgabewertKopf4      :=
       Freigegeben             :=
```

Mit dem Aufruf des FB 10 „Ident-Control-System“ einhergehend ist die Angabe eines Datenbausteins, in dem die Daten dieser IDENT-Control-Einheit abgespeichert werden. Dieser Instanzen-DB wird dabei automatisch erstellt (im Beispiel wurde der DB 10 verwendet).

Die Parameter werden nun im Einzelnen erläutert.

5.5.1 Die Parameter beim Programmaufruf im Einzelnen

- *Adresse_Ident_Control*

  Hier ist die Adresse der IDENT-Control in der SPS im Wort-Format anzugeben.

  (Angabe im Beispiel: W#16#100)

- *Datenbaustein*

  Die Nummer des im Call-Aufruf angegebenen Instanzen-DBs ist hier anzugeben. Dies ist erforderlich, damit diese an die anderen untergeordneten Bausteine übergeben werden kann.

  (Angabe im Beispiel: 10)

- *MaxAnzWdh*

  Dieser Parameter enthält die Information, wie oft ein Single-Befehl bei der Status-Rückmeldung 05_h (kein Datenträger im Erfassungsbereich) wiederholt werden soll. Eine sinnvolle Einstellung liegt zwischen 3 und 5.

  (Angabe im Beispiel: 5)

- *TimeoutDauer*

  Der Startwert des Countdown-Timers zur Timeoutüberwachung bei Single-Befehlen wird auf den hier angegebenen Wert eingestellt. Soll nur die Funktion Lesen verwendet werden, ist ein Wert von 1 s ausreichend, bei Lesen und Schreiben muss die Zeit mindestens 2 s sein.

  (Angabe im Beispiel: S5T#2S)

- *TimeoutTimerKopf1 .. TimeoutTimerKopf4*

  Hier müssen die Timer angegeben werden, die für die Timeoutüberwachung des betreffenden Kopfes verwendet werden sollen.

  (Angabe im Beispiel: T1 .. T4)

- *Datenträgertyp*

  Der Datenträgertyp, der von den Köpfen gelesen bzw. geschrieben werden soll, muss bei diesem Parameter im Wort-Format angegeben werden. Dabei ist darauf zu achten, dass die im Handbuch der IDENT-Control genannte Auswahl mittels in Hexadezimal angegebenen ASCII-Zeichen getroffen werden muss (Bsp: Für die Auswahl des IPC-03 ist „03“$_{ASCII}$, also 3033_{hex} einzugeben. Näheres dazu siehe im Kapitel „Change Tag Befehl“ im Handbuch zur IDENT-Control.

  (Angabe im Beispiel: W#16#3033)

- *DatenFixcode*

  Hier wird die Auswahl getroffen, ob Daten oder Fixcode ausgelesen werden sollen. Dabei bedeutet „0“ Daten und „1“ Fixcode. Natürlich kann dies auch von einem Merker oder Eingang abhängig gemacht werden und muss hier nicht fest mit 0 oder 1 angegeben zu werden.

  (Angabe im Beispiel: M 1.0)
- *SingleEnhanced*

  Mit diesem Parameter verhält es sich ebenso wie mit DatenFixcode. Mit SingleEnhanced wird die Auswahl getroffen, ob ein Single- oder ein Enhanced-Befehl ausgeführt werden soll. („0“ = single, „1“ = enhanced)

  (Angabe im Beispiel: M1.1)
- *Kopf1Lesen .. Kopf4Lesen*

  Hier kann mittels Merkern oder Eingängen der Lesebefehl für den jeweiligen Kopf ausgelöst werden. Sollen ein oder mehrere Köpfe nicht verwendet werden, so kann auch FALSE eingegeben werden, damit keine Merker unnötig belegt werden.

  (Angabe im Beispiel: M 2.0 .. M 2.3)
- *Kopf1Schreiben .. Kopf4Schreiben*

  Analog zu Kopf1Lesen .. Kopf4Lesen – nur eben für die Schreibbefehle.

  (Angabe im Beispiel: M 3.0 .. M 3.3)
- *Kopf1Quit .. Kopf4Quit*

  Mit den hier angegebenen Merkern oder Eingängen wird der Quitbefehl für den betreffenden Kopf ausgelöst.

  (Angabe im Beispiel: M 4.0 .. M 4.3)
- *QuitErrorKopf1 .. QuitErrorKopf4*

  Mit Hilfe dieser Bits wird die Quittierung von aufgetretenen Fehlern vorgenommen, d.h. es werden beim Durchlaufen der QuitError-Routine die Fehlermeldungen des entsprechenden Kopfes quittiert.

  (Angabe im Beispiel: M 5.0 .. M 5.3)

- *Datenträgerstartadresse*

  An dieser Stelle ist die Startadresse anzugeben, ab der Daten aus dem Datenträger gelesen bzw. geschrieben werden sollen. Bei Fixcode-Befehlen ist zwar ein Wert beim Aufruf des FBs aus rein formalen Gründen einzutragen, dieser wird jedoch eigentlich nicht benötigt und somit bei der Befehlsausführung von der IDENT-Control ignoriert.
  (Angabe im Beispiel: W#16#0)

- *Anzahl32BitWorte*

  Hier wird die Anzahl der zu lesenden oder zu schreibenden 32 Bit Worte angegeben. Dabei ist zu beachten, dass der Wert nicht größer als 7 sein darf, da maximal 28 Byte Nutzdaten übertragen werden können (siehe Kapitel „Aufbau eines Befehls und der dazugehörigen Antwort“). Genau wie bei der Datenträgerstartadresse muss die Angabe der Anzahl32BitWorte aus rein bei Fixcode-Befehlen nicht benötigt und somit ignoriert.
  (Angabe im Beispiel: 7)

- *RückgabewertKopf1 .. RückgabewertKopf4*

  In diesen 4 Bytes werden die Statusbits der einzelnen Köpfe zurückgegeben. Die Bytes sind kopfbezogen, d.h. jeder Kopf hat sein eigenes Rückgabebyte. In Abhängigkeit dieser Rückgabebytes kann der Anwender den Rest der Anlage steuern bzw. den aktuellen Status der Köpfe ablesen. Mehr dazu im folgenden Kapitel.
  (Angabe im Beispiel: MB 6 .. MB 9)

- *Freigegeben*

  Das Bit Freigegeben dient zum einen als Signal, dass die Initialisierung erfolgreich durchgeführt, d.h. die Köpfe auf den parametrierten Datenträgertyp eingestellt wurden. Zum anderen kann durch Rücksetzen dieses Bits eine automatische Neuinitialisierung gestartet werden. Nach erfolgreichem Durchlaufen der Initialisierung ist das Bit dann wieder gesetzt.
  (Angabe im Beispiel: M 0.0)

Mit allen erforderlichen Angaben sieht der Funktionsaufruf dann folgendermaßen aus. (Die Merker wurden mit symbolischen Namen versehen, die ihre Funktion zum Ausdruck bringen.)

```
OB1, Netzwerk 1: Aufruf des FBs Ident-Control-System

CALL  "Ident-Control-System" , "DB für IC_1"
       Adresse_Ident_Control   :=W#16#100
       Datenbaustein           :=10
       MaxAnzWdh               :=20
       TimeoutDauer            :=S5T#2S
       TimeoutTimerKopf1       :=T1
       TimeoutTimerKopf2       :=T2
       TimeoutTimerKopf3       :=T3
       TimeoutTimerKopf4       :=T4
       Datentraegertyp         :=W#16#3033
       DatenFixcode            :="DatenFixcode"
       SingleEnhanced          :="SingleEnhanced"
       Kopf1Lesen              :="Kopf1Lesen"
       Kopf1Schreiben          :="Kopf1Schreiben"
       Kopf2Lesen              :="Kopf2Lesen"
       Kopf2Schreiben          :="Kopf2Schreiben"
       Kopf3Lesen              :="Kopf3Lesen"
       Kopf3Schreiben          :="Kopf3Schreiben"
       Kopf4Lesen              :="Kopf4Lesen"
       Kopf4Schreiben          :="Kopf4Schreiben"
       Kopf1Quit               :="Kopf1Quit"
       Kopf2Quit               :="Kopf2Quit"
       Kopf3Quit               :="Kopf3Quit"
       Kopf4Quit               :="Kopf4Quit"
       QuitErrorKopf1          :="QuitErrorKopf1"
       QuitErrorKopf2          :="QuitErrorKopf2"
       QuitErrorKopf3          :="QuitErrorKopf3"
       QuitErrorKopf4          :="QuitErrorKopf4"
       Datentraegerstartadresse:=W#16#0
       Anzahl32BitWorte        :=7
       RueckgabewertKopf1      :="RueckgabewertKopf1"
       RueckgabewertKopf2      :="RueckgabewertKopf2"
       RueckgabewertKopf3      :="RueckgabewertKopf3"
       RueckgabewertKopf4      :="RueckgabewertKopf4"
       Freigegeben             :="Freigegeben"
```

Um die Multiinstanzfähigkeit der vorliegenden Programmbausteine auszunutzen, d.h. eine zweite IDENT-Control-Einheit noch in die Konfiguration mit aufzunehmen, muss diese IDENT-Control-Einheit im Hardware-Manager der Step7-Entwicklungsumgebung wie in Kapitel „Profibus-Konfiguration“ beschrieben ergänzt werden. Anschließend ist der Baustein FB 10 einfach nur mit einem anderen Instanzen-DB, der Adresse der zweiten IDENT-Control-Einheit und den entsprechenden weiteren Parametern aufzurufen und schon kann diese angesprochen und bedient werden. Für weitere IDENT-Control-Einheiten ist genauso zu verfahren.

5.5.2 Datenbereiche der Schreib- und Lesedaten im Instanzen-DB

Durch die aus dem Deklarationsteil des FB 10 erzeugte Struktur der Instanzen-DBs liegen die Schreib- und Lesedaten der einzelnen Köpfe an der gleichen Stelle bzw. Adresse in allen Instanzen-DBs. Außerdem sind die ersten 4 Bytes (in der Tabelle als Header bezeichnet), wie bereits im Vorfeld erwähnt, für die Auswertung in der SPS von großer Bedeutung, da sie den Befehlscode, das Togglebit, den Status und den Ausführungszähler enthalten. Die nachfolgende Übersicht gibt die Adressen im DB an:

Kopf 1	Header Schreibdaten:	DBB 92 bis inkl. DBB 95
	Schreibdaten:	DBD 96 bis inkl. DBD 120
	Header Lesedaten:	DBB 60 bis inkl. DBB 63
	Lesedaten:	DBD 64 bis inkl. DBD 88
Kopf 2	Header Schreibdaten:	DBB 182 bis inkl. DBB 185
	Schreibdaten:	DBD 186 bis inkl. DBD 210
	Header Lesedaten:	DBB 150 bis inkl. DBB 153
	Lesedaten:	DBD 154 bis inkl. DBD 178
Kopf 3	Header Schreibdaten:	DBB 272 bis inkl. DBB 275
	Schreibdaten:	DBD 276 bis inkl. DBD 300
	Header Lesedaten:	DBB 240 bis inkl. DBB 243
	Lesedaten:	DBD 244 bis inkl. DBD 268
Kopf 4	Header Schreibdaten:	DBB 362 bis inkl. DBB 365
	Schreibdaten:	DBD 366 bis inkl. DBD 390
	Header Lesedaten:	DBB 330 bis inkl. DBB 333
	Lesedaten:	DBD 334 bis inkl. DBD 358

Tabelle 11: Datenbereiche der Schreib- und Lesedaten im Datenbaustein

5.5.3 Auswertung der Rückgabebytes

Anhand der zurückgemeldeten Statusbytes (Aufbau siehe Abb. 27) können Sie erkennen, in welchem Zustand sich derzeit welcher Kopf befindet und ob der gegebene Befehl ausgeführt wurde bzw. noch wird.

Abb. 27: Aufbau der Rückgabebytes

Um zu verdeutlichen wann welches Bit gesetzt wird und welche Schlüsse zur weiteren Anlagensteuerung daraus zu ziehen sind, werden hier die möglichen Bitkombinationen aufgezeigt und kurz erläutert.

- Ist das **VorhandenTC-Bit** (Bit 0 des Rückgabebytes) nicht gesetzt, so ist entweder die Initialisierung dieses Kopfes fehlgeschlagen oder der Kopf gar nicht vorhanden. Damit können dann auch keine Befehle an diesen Kopf gegeben werden.
- Solange das **Busy-Bit** (Bit 7) gesetzt ist, ist ein Befehl an diesem Kopf in Bearbeitung. Es kann kein weiterer Befehl zur Ausführung angenommen werden.
- Generell lässt sich sagen, dass immer wenn das **Done-Bit** (Bit 6 des Rückgabebytes) gesetzt wird, irgendwelche neuen Informationen von der Ausführung der gegebenen Befehle zur Verfügung stehen. Die folgenden Fälle können dabei auftreten (Es wird davon ausgegangen, dass das VorhandenTC-Bit in allen nachfolgend aufgezeigten Fällen gesetzt ist!)
    - Nur **Done** (Bit 6)

      Der gegebene Befehl wurde fehlerfrei ausgeführt und die Daten können aus den entsprechenden Bereichen im Datenbaustein ausgelesen werden.

- **Done** (Bit 6) **& Error** (Bit 1)

  Bei der Ausführung des Befehls ist ein Fehler aufgetreten. Das Error-Bit lässt sich nur durch Setzen des QuitErrorKopfx-Bits quittieren und somit zurücksetzen. Bevor das Error-Bit nicht zurückgesetzt wurde, werden keine weiteren Befehle mehr für diesen Kopf angenommen und ausgeführt.

  Will der Anwender genauer wissen, welcher Fehler aufgetreten ist, so kann er die im DB verwendeten internen Signalisierungsbits wie Err_SFC14, Err_SFC15, InvalidResponse etc. für eine weitergehende Analyse verwenden.

- **Done** (Bit 6) **& TimeoutOccured** (Bit 2)

  Ist diese Bitkombination gesetzt, so ist ein Timeoutfehler aufgetreten, d.h. der betreffende Kopf hat nicht innerhalb der eingestellten Zeit geantwortet.

- **Done** (Bit 6) **& KeinDatenträger** (Bit 5)

  Bei dieser Bitkombination wurde der Single- bzw. Enhanced-Befehl zwar ausgeführt, jedoch hat sich zu diesem Zeitpunkt kein Daten- oder Fixcodeträger im Erfassungsbereich des Schreib-/Lesekopfes befunden. Es konnten somit auch keine Daten gelesen oder geschrieben werden.

- Die Bits 3 und 4 der Rückgabebytes werden nicht verwendet.

6 Steuerung mittels Variablentabelle

Die dem Beispielprogramm beigefügten Variablentabellen AnwTabKopf1 .. AnwTabKopf4 können zur Steuerung der IDENT-Control verwendet werden.

Die Voraussetzungen für die Steuerungsmöglichkeit mittels dieser Tabellen sind:

- Die Hardwarekomponenten wurden richtig aufgebaut und angeschlossen
- Sämtliche Bausteine des Beispielprogramms wurden in die SPS übertragen
- Die Hardwarekonfiguration wurde in die Steuerung geladen
- Der Mode-Schalter an der S7 wurde auf RUN-P (!) gestellt
- Die Variablentabelle ist geöffnet und in den Online-Modus geschaltet

In der folgenden Abbildung ist der Aufbau der AnwTabKopf1 exemplarisch dargestellt.

	Operand	Symbol	Anzeigef	Statuswert	Steuerwert
1	M 0.0	"Freigegeben"	BOOL		
2	M 1.0	"DatenFixcode"	BOOL		
3	M 1.1	"SingleEnhanced"	BOOL		
4	M 2.0	"Kopf1Lesen"	BOOL		
5	M 3.0	"Kopf1Schreiben"	BOOL		
6	M 4.0	"Kopf1Quit"	BOOL		
7	M 5.0	"QuitErrorKopf1"	BOOL		
8	MB 6	"RueckgabewertKopf1"	BIN		
9					
10	DB10.DBX 132.0	"DB für IC_1".Kopf_1.VorhandenTC	BOOL		
11	DB10.DBX 132.1	"DB für IC_1".Kopf_1.Error	BOOL		
12	DB10.DBX 132.2	"DB für IC_1".Kopf_1.TimeoutOccured	BOOL		
13	DB10.DBX 132.3	"DB für IC_1".Kopf_1.EmpfangenOK	BOOL		
14	DB10.DBX 132.4	"DB für IC_1".Kopf_1.SendenOK	BOOL		
15	DB10.DBX 132.5	"DB für IC_1".Kopf_1.KeinDatentraeger	BOOL		
16	DB10.DBX 132.6	"DB für IC_1".Kopf_1.Done	BOOL		
17	DB10.DBX 132.7	"DB für IC_1".Kopf_1.Busy	BOOL		
18					
19	DB10.DBD 64	"DB für IC_1".Kopf_1.Eingangsdaten.Eing_DW1	ZEICHEN		
20	DB10.DBD 68	"DB für IC_1".Kopf_1.Eingangsdaten.Eing_DW2	ZEICHEN		
21	DB10.DBD 72	"DB für IC_1".Kopf_1.Eingangsdaten.Eing_DW3	ZEICHEN		
22	DB10.DBD 76	"DB für IC_1".Kopf_1.Eingangsdaten.Eing_DW4	ZEICHEN		
23	DB10.DBD 80	"DB für IC_1".Kopf_1.Eingangsdaten.Eing_DW5	ZEICHEN		
24	DB10.DBD 84	"DB für IC_1".Kopf_1.Eingangsdaten.Eing_DW6	ZEICHEN		
25	DB10.DBD 88	"DB für IC_1".Kopf_1.Eingangsdaten.Eing_DW7	ZEICHEN		
26					
27	DB10.DBD 96	"DB für IC_1".Kopf_1.Ausgangsdaten.Ausg_DW1	ZEICHEN		'KOPF'
28	DB10.DBD 100	"DB für IC_1".Kopf_1.Ausgangsdaten.Ausg_DW2	ZEICHEN		'12 '
29	DB10.DBD 104	"DB für IC_1".Kopf_1.Ausgangsdaten.Ausg_DW3	ZEICHEN		
30	DB10.DBD 108	"DB für IC_1".Kopf_1.Ausgangsdaten.Ausg_DW4	ZEICHEN		
31	DB10.DBD 112	"DB für IC_1".Kopf_1.Ausgangsdaten.Ausg_DW5	ZEICHEN		
32	DB10.DBD 116	"DB für IC_1".Kopf_1.Ausgangsdaten.Ausg_DW6	ZEICHEN		
33	DB10.DBD 120	"DB für IC_1".Kopf_1.Ausgangsdaten.Ausg_DW7	ZEICHEN		
34					

Abb. 28: Die Variablentabelle AnwTabKopf1 zur Steuerung des Kopfes Nr. 1

Das Freigegeben-Bit wird durch den Merker M 0.0 in Zeile 1 der Variablentabelle repräsentiert. Ist dieses Bit gesetzt, wurde die Initialisierung erfolgreich durchgeführt. Um eine Neuinitialisierung zu starten, setzen Sie wie in den vorherigen Ausführungen schon erwähnt das Freigegeben-Bit einfach zurück. Nach der Neuinitialisierung ist es dann wieder gesetzt.
Die in den Zeilen 2-7 eingetragenen Merker dienen zum Starten des gewünschten Befehls. Je nach Zustand der Merkerbits M 1.0 und M 1.1 wird ein Fixcode- oder Daten- bzw. ein Single- oder Enhanced-Befehl ausgeführt.
Im Merkerbyte MB 6 (Zeile 8 der Tabelle) ist der Rückgabewert des Kopfes zu finden. Die darin enthaltenen Bits sind in den Zeilen 10-17 nochmals einzeln aufgeschlüsselt um einen besseren Überblick zu gewährleisten.
Im darauffolgenden Block (Zeilen 19-25) stehen die von der IDENT-Control ausgelesenen Daten.
In den Zeilen 27-33 können die Daten eingetragen werden, die auf einen Datenträger geschrieben werden sollen. Dazu müssen die entsprechenden Daten, wie in Abb. 28 beispielhaft zu erkennen ist (‚KOPF 12'), in der Spalte Steuerwert eingetragen und anschließend durch Drücken der Taste  in den Datenbaustein übernommen werden. Erst danach darf der Schreibbefehl aufgerufen werden, da sonst die in der Spalte Statuswert noch enthaltenen vorherigen Daten geschrieben werden.

Die vorgestellte Struktur ist in allen Tabellen die gleiche und in den Tabellen 2-4 genauso zu verwenden.
Sollten Sie andere Merker oder Eingänge verwendet haben, sind die in der Tabelle stehenden Eintragungen natürlich nichtig und Sie müssen Ihre spezifische Konfiguration zur Steuerung der IDENT-Control eintragen.

7 Weiterführende Hinweise

7.1 Die gerätespezifischen Parameter der IDENT-Control

An jeder in die Hardwarekonfiguration der SPS eingefügten IDENT-Control-Einheit können gerätespezifische Einstellungen vorgenommen werden. Zum einen kann die Data Hold Time parametriert werden und zum anderen vom Anwender entschieden werden, ob der Diagnose-Interrupt (s. Ziff. 3.7.1.2) freigeschaltet sein soll oder nicht.

Vorgenommen werden diese Einstellungen im Hardware-Konfigurator der Step7-Software durch Rechtsklick auf die betreffende IDENT-Control-Einheit am Profibusstrang und Auswahl des Punkt „Objekteigenschaften“. Klicken Sie nun auf die Registerkarte Parametrieren und es öffnet sich daraufhin das in der folgenden Abbildung gezeigte Menü.

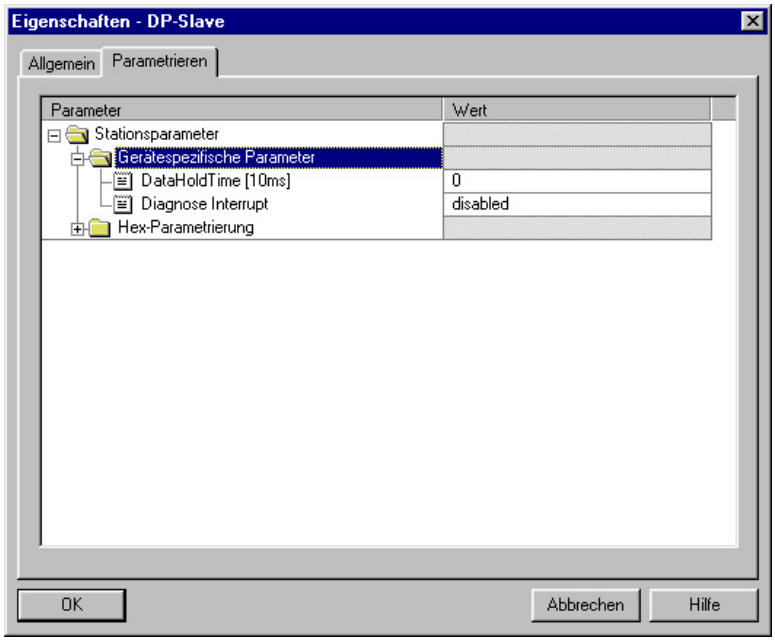

Abb. 29: Die gerätespezifischen Parameter der IDENT-Control

Die beiden Parameter werden nun im Einzelnen erläutert. Zum Abschließen der Parametrierung klicken Sie auf OK und laden die Hardwarekonfiguration in die SPS.

7.1.1 Die Data Hold Time

Mittels der Data Hold Time wird bestimmt, wie lange die Antwortdaten mindestens im Ausgangspuffer der IDENT-Control zur Abholung bereit stehen müssen, bevor sie von den nächsten Antwortdaten überschrieben werden dürfen. Gerade bei größeren Steuerungsprogrammen und der damit verbundenen längeren Zykluszeit ist diese Einstellmöglichkeit von großer Bedeutung. Normalerweise wird der Wert der Data Hold Time auf etwas mehr als den Wert der Zykluszeit der SPS eingestellt. Damit ist sichergestellt, dass alle Daten von der SPS empfangen werden und keine Datensätze auf dem Übertragungsweg zwischen SPS und IDENT-Control verloren gehen.
Die Default-Einstellung der Data Hold Time ist „0“. Der eingegebene Zahlenwert ist mit dem Faktor 10 ms zu multiplizieren, um die resultierende Data Hold Time in ms zu erhalten.

7.1.2 Der Diagnose Interrupt

Im Betrieb überprüft die IDENT-Control ständig, an welchen Anschlüssen Schreib-/Leseköpfe angeschlossen sind. Ändert sich an der Kopfkonfiguration etwas, d.h. wird ein Kopf abgeklemmt oder ein weiterer Kopf angeschlossen, so kann dies mit dem Diagnose Interrupt der SPS gemeldet werden. Die Entscheidung darüber fällt der Anwender, indem er den Wert des Diagnose Interrupts auf „Enabled“ oder „Disabled“ stellt.
Ist der Diagnose Interrupt eingeschaltet, wird bei jeder Kopfkonfigurationsänderung die Meldung „Baugruppe gestört“ und damit der OB 82 (s. Ziff. 7.2) sowohl bei kommender als auch bei gehender Meldung in der SPS ausgelöst. Ist der OB 82 nicht in das Step7-Projekt miteingebunden, so geht die CPU bei der Auslösung des Diagnose Interrupts in den STOP-Mode.
Wenn Sie in den OB 82 den Befehl zum Rücksetzen des Freigegeben-Bits hineinschreiben, so wird bei jeder Änderung der Kopfkonfiguration an der IDENT-Control eine automatische Neu-Initialisierung durchgeführt. Dabei werden allerdings dann auch alle derzeit aktiven Befehle an dieser IDENT-Control-Einheit abgebrochen.

7.2 Die Organisationsbausteine OB 82 und OB 86

Der OB 82, symbolischer Name „I/O_FLT_1“, wird beim Auftreten eines Diagnose Alarms einer Baugruppe von der CPU automatisch aufgerufen, also z.B. wenn ein Kopf an der IDENT-Control abgeklemmt wird oder ausfällt. Ist der Baustein OB 82 nicht in das Step7-Projekt eingefügt, so geht die CPU bei Auftritt eines Diagnose Alarms in den STOP-Zustand.

Der OB 86 („Rack_FLT“) wird bei Ausfall eines Erweiterungsgeräts, eines DP-Mastersystems oder eines am Profibus-DP befindlichen Gerätes (z.B. durch Ausfall der Versorgungsspannung) aufgerufen. Wird dieser Organisationsbaustein nicht in das Step7-Projekt eingefügt, erfolgt wie beim OB 82 im Fehlerfall ein Wechsel der CPU vom RUN- in den STOP-Zustand.

Ist dies nicht gewünscht, d.h. soll die SPS trotz aufgetretenem Fehler weiterlaufen um beispielsweise die Anlage kontrolliert herunterfahren zu können, so müssen die beiden OBs einfach nur in das Step7-Projekt eingefügt werden.

7.3 Stichwort: Konsistente Datenübertragung

Im Beispielprogramm wurde eine konsistente Datenübertragung zwischen IDENT-Control und der SPS durch Verwendung der Systemfunktionen SFC 14 und SFC 15 sichergestellt. Auch bei Systemen anderer Hersteller muss auf jeden Fall eine konsistente Übertragung der Daten mit geeigneten Mechanismen sichergestellt werden. Andernfalls sind Funktions- oder Programmfehler aufgrund fehlerhafter Befehle und/oder Parametern die Folge und die Sicherheit der Anlage kann nicht gewährleistet werden!

8 Kurzanleitung zur Schnellinbetriebnahme der IDENT-Control

1. Verbinden Sie die IDENT-Control über Profibus mit der SPS.
2. Schließen Sie die Schreib-/Leseköpfe an die IDENT-Control an.
3. Stellen Sie die Verbindung zwischen Programmiergerät und der SPS über ein MPI-Kabel her.
4. Versorgen Sie die SPS und die IDENT-Control (20-30 V DC) mit Spannung.
5. Installieren Sie die GSD-Datei im Hardware-Konfigurator der Step7-Software.
6. Dearchivieren Sie das File „AnwProg.zip“ und öffnen Sie das Projekt in Step7.
7. Ändern Sie die Parameter beim Aufruf des Bausteins „Ident-Control-System“, falls dies gewünscht ist. Standardmäßig ist der Datenträgertyp IPC-03 eingestellt. Alle Parameter werden im Inbetriebnahmehandbuch im Kapitel „Die Parameter beim Programmaufruf im Einzelnen“ ausführlich erläutert.
8. Übertragen Sie die Hardware-Konfiguration und alle Bausteine an die SPS.
9. Stellen Sie die im Programm gewählte Profibus-Adresse (im Beispielprogramm: 3) in der IDENT-Control ein. Dazu verfahren Sie folgendermaßen:
   - ↵, ↓, „IDENT-Gateway“, ↵, „Profibus-Einstellung“, ↵, „Busadresse“
   - Zuerst Zehnerziffer einstellen mit ↑↓, dann ↵, danach Einerziffer auch mit ↑↓
   - Schließlich 3 x ESC

   Nun müssen Sie die IDENT-Control kurz von der Versorgungsspannung trennen und wieder verbinden, damit diese die neuen Einstellungen übernimmt.
10. Stellen Sie den Mode-Schalter an der SPS auf „Run-P“
11. Öffnen Sie in der Step7-Software die Variablentabelle AnwTabKopf1 und wechseln Sie in den Online-Mode.
12. Jetzt können Sie durch Setzen der Merker im oberen Teil der Tabelle einen Befehl zur IDENT-Control senden.

    Im zweiten Teil sind Bits und deren Zustand zu erkennen, die ebenfalls im Rückgabewert zu finden sind (mit Ausnahme der beiden Bits SendenOK und EmpfangenOK)

    Die gelesenen Daten werden im dritten Teil der Variablentabelle dargestellt.

    Zum Schreiben von Daten auf einen Datenträger müssen die Daten zunächst im vierten Teil der Tabelle eingegeben und übertragen werden. Anschließend können Sie den gewünschten Schreibbefehl aufrufen und ausführen.