

Control III Programmierung in C (Klein SPS)

Beschreibung der Befehle

Änderungen vorbehalten.

Die Nennung von Waren erfolgt in diesem Werk in der Regel ohne Erwähnung bestehender Patente, Gebrauchsmuster oder Warenzeichen.

Das Fehlen eines solchen Hinweises begründet nicht die Annahme, eine Ware sei frei.

Inhaltsverzeichnis

Control III Programmierung in C (Klein SPS)

1	Allgemein	11
1.1	Die verwendeten Symbole	11
1.2	Sicherheit	11
1.2.1	Bestimmungsgemäße Verwendung	11
1.2.2	Allgemeine Sicherheitshinweise	11
1.2.3	Entsorgung	11
1.3	Produktinformation	12
2	Getting Started	13
2.1	Download von Eclipse Control III	13
2.2	Installation von Eclipse Control III	14
3	Verwenden von Eclipse	15
3.1	Programm öffnen	15
3.2	Einführung in die Oberflächen von Eclipse	15
3.2.1	Der Projekt Explorer	15
3.2.2	Toolbar	16
3.2.2.1	Compiler	16
3.2.2.2	Debuggen	16
3.2.2.3	Konfigurationstools	16
	- Unlock Control:	16
	- Download Control:	16
	- Start Control:	16
	- Download + Start Control:	17
	- Stop Control:	17
	- Set Auto Start:	17
	- Clear Auto Start:	17
	- Read Merker:	17
	- Read Merker zyklisch:	18
	- Zykluszeit:	18
	- Reset Cycle Time:	18
3.2.3	Editor	18
3.2.4	Konsole	18
3.3	Datei-Informationen	19
	- control_io.c	19
	- control_io.h	19
	- control.h	19
	- main.c	19
	- startup.c	19
	- *.ld	19
	- *.mak	19

- settings.mak	20
3.4 Einstellen der Schnittstelle	20
3.5 Anlegen eines neuen Projektes	20
3.6 Ein Beispielprojekt	21
3.6.1 Der C-Code	22
3.6.2 Kompilieren	24
3.6.3 Download	24
3.6.4 Starten von Control III	24
3.7 Der Debugger	24
3.7.1 Initialisierung	25
3.7.2 Übersicht Debugger	25
3.7.2.1 Das Control Panel	26
- Resume (F8):	26
- Terminate (Ctrl + F2):	26
- Step Into (F5):	26
- Step Over (F6):	26
3.7.2.2 Tasks	26
3.7.2.3 Debugger Übersicht	27
- Variablen:	27
- Breakpoints:	27
3.7.2.4 Codeübersicht	27
3.7.2.5 Disassembly	27
3.7.2.6 Konsole	27
3.7.3 Start des Debuggers	28
3.7.4 Beispiel	28
3.7.4.1 Start des Debuggers	32
3.7.4.2 Debugger benutzen	33
4 Programmieren von Control III	35
4.1 Übersicht der ASi-3 Befehle	35
4.1.1 ASi Data Exchange	37
4.1.2 ASi Read IDI	38
4.1.3 ASi Write ODI	38
4.1.4 ASi Read ODI	39
4.1.5 ASi Write Permanent Parameter	39
4.1.6 ASi Read Permanent Parameter	39
4.1.7 ASi Send Parameter	39
4.1.8 ASi Read PI	40
4.1.9 ASi Store PI	40
4.1.10 ASi Read Duplicate Address List	40
4.1.11 ASi Read Fault Detector	40
4.1.12 ASi Write PCD	41
4.1.13 ASi Read PCD	41
4.1.14 ASi Store CDI	41
4.1.15 ASi Read CDI	42
4.1.16 ASi Write Extended ID1	42
4.1.17 ASi Write LPS	43
4.1.18 ASi Read LPS	43
4.1.19 ASi Read LAS	44
4.1.20 ASi Read LDS	44
4.1.21 ASi Read LCS	45
4.1.22 ASi Write LOS	45
4.1.23 ASi Read LOS	45
4.1.24 ASi Read LPF	45
4.1.25 ASi Read Ec Flags	46
4.1.26 ASi set Config Mode	46

4.1.27	ASi Write Hi Flags	46
4.1.28	ASi Read Hi-Flags	46
4.1.29	ASi Address Slave	47
4.1.30	ASi Execute Command	47
4.1.31	ASi Read All Config	47
4.1.32	ASi Write All Config	47
4.1.33	ASi read Error Counter	48
4.1.34	ASi Mail Box	48
4.1.35	ASi Write 16Bit ODI	48
4.1.36	ASi Read 16Bit ODI	49
4.1.37	ASi Read 16Bit IDI	49
4.1.38	ASi Read Ctrl Acc ODI	49
4.1.39	ASi Write Ctrl Acc ODI	49
4.1.40	ASi Read Ctrl Acc AODI	50
4.1.41	ASi Write Ctrl Acc AODI	50
4.1.42	AASi Read PE	51
4.1.43	AASi Set CPRL	51
4.1.44	AASi Send Parameter Non Blocking	51
4.1.45	Get Systeime	51
4.2	Übersicht der ASi-5 Befehle	52
4.2.1	Asi5ReadCtrlAccODI	55
4.2.2	Asi5WriteCtrlAccODI	55
4.2.3	Asi5ClearCtrlAccODI	55
4.2.4	Asi5ReadEcFlags	56
4.2.5	Asi5OdcRequest	56
4.2.6	Asi5OdcRequestByAsiid	56
4.2.7	Asi5SetOperatingMode	56
4.2.8	Asi5GetOperatingMode (circuit, ...)	57
4.2.9	Asi5SetDataExchangeMode	57
4.2.10	Asi5GetDataExchangeMode	57
4.2.11	Asi5SetOfflineMode	57
4.2.12	Asi5GetOfflineMode	58
4.2.13	Asi5SetAutoAddressEnable	58
4.2.14	Asi5GetAutoAddressAssignMode	58
4.2.15	Asi5SetASi3CoexistingMode	58
4.2.16	Asi5GetASi3CoexistingMode	59
4.2.17	Asi5ChangeSlaveAddress	59
4.2.18	Asi5ChangeSlaveAddressByAsiid	59
4.2.19	Asi5InLds	60
4.2.20	Asi5InLps	60
4.2.21	Asi5InLpf	60
4.2.22	Asi5InLms	61
4.2.23	Asi5InLas	61
4.2.24	Asi5InLis	61
4.2.25	Asi5InLda	62
4.2.26	Asi5InLdd	62
4.2.27	Asi5SetCdi	62
4.2.28	Asi5GetCdi	63
4.2.29	Asi5SetPcd	63
4.2.30	Asi5GetPcd	63
4.2.31	Asi5StoreCdi	64
4.2.32	Asi5StoreSlaveCdi	64
4.2.33	Asi5ResetPcd	64
4.2.34	Asi5ResetAllPcd	65
4.2.35	Asi5GetLas	65
4.2.36	Asi5GetLps	65
4.2.37	Asi5GetLds	65
4.2.38	Asi5GetLms	66
4.2.39	Asi5GetLpf	66
4.2.40	Asi5GetLis	66

4.2.41	Asi5GetLda	66
4.2.42	Asi5GetLdd	67
4.2.43	Asi5GetSlaveAsiid	67
4.2.44	Asi5GetSlaveInfo	67
4.2.45	Asi5GetSlaveInfoAsiid	68
4.2.46	Asi5GetSystemResources	68
4.2.47	Asi5IdentifySlaveByLogicalAddress	68
4.2.48	Asi5IdentifySlaveByAsiid	68
4.2.49	Asi5SetDeviceDesignatorByLogicalAddress	69
4.2.50	Asi5SetDeviceDesignatorByAsiid	70
4.2.51	Asi5GetDeviceDesignatorByLogicalAddress	70
4.2.52	Asi5GetDeviceDesignatorByAsiid	71
4.2.53	Asi5AutoSetLogicalAddresses	71
4.2.54	Asi5SetEnergySavingState	71
4.2.55	Asi5GetEnergySavingState	72
4.2.56	Asi5WriteAsi5Odi	72
4.2.57	Asi5ReadAsi5Odi	72
4.2.58	Asi5ReadAsi5Idi	73
4.2.59	Asi5DiagGetMasterStatus	73
4.2.60	Asi5GetParameters	73
4.2.61	Asi5SetParameters	74
4.2.62	Asi5SetPsMode	74
4.2.63	Asi5GetPsMode	74
4.2.64	Asi5GetStoredPi	75
4.2.65	Asi5SetStoredPi	75
4.2.66	Asi5StorePi	75
4.2.67	Asi5RestorePi	76
4.2.68	Asi5StoreAllSlavePi	76
4.2.69	Asi5RestoreAllSlavePi	76
4.2.70	Asi5SlavenumberToLogAddr	76
4.2.71	Asi5LogAddrToSlavenumber	77
4.2.72	Asi5DeltaList	77
4.2.73	Asi5InDeltaList	77
4.2.74	Asi5GetIoLength	78
4.2.75	Asi5GetIoLengthByAsiid	78
4.2.76	Asi5ResetAllSlaves	78
4.3	Übersicht der Control Befehle	79
4.3.1	Ctrl Init Timer	80
4.3.2	Ctrl Delay	80
4.3.3	Ctrl Init wgd	80
4.3.4	Ctrl Trigger wdg	80
4.3.5	Ctrl Eval Cycle time	80
4.3.6	Ctrl Read Parameter	81
4.3.7	Ctrl Write Parameter	81
4.3.8	Ctrl Read Flags	81
4.3.9	Ctrl Write Flags	81
4.3.10	Ctrl Read Key	81
4.3.11	Ctrl printf	82
4.3.12	Ctrl Breakpoint	82
4.3.13	Ctrl Display	82
4.3.14	CCTRL Enet Set IP data	83
4.3.15	CCTRL Enet Socket	83
4.3.16	CCTRL Enet Set sockopt	84
4.3.17	CCTRL Enet Bind	84
4.3.18	CCTRL Enet Listen	84
4.3.19	CCTRL Enet Accept	84
4.3.20	CCTRL Enet Connect	85
4.3.21	CCTRL Enet Send	85
4.3.22	CCTRL Enet Send to	85
4.3.23	Ctrl Read Parameter	86

4.3.24	CCtrl Enet Recv from	86
4.3.25	CCtrl Enet Socket close	86
4.3.26	CCtrl Read Parameter Additional	86
4.3.27	CCtrl Write Parameter Additional	87
4.3.28	CCtrlFcntl	87
4.4	Übersicht der BACnet-Befehle	88
4.4.1	CCtrlBACnetWriteBinaryOutputPresentValue	89
4.4.2	CCtrlBACnetReadBinaryOutputPresentValue	89
4.4.3	CCtrlBACnetClearBinaryOutputPresentValue	89
4.4.4	CCtrlBACnetWriteAnalogOutputPresentValue	90
4.4.5	CCtrlBACnetReadAnalogOutputPresentValue	90
4.4.6	CCtrlBACnetClearAnalogOutputPresentValue	90
4.4.7	CCtrlBACnetWriteIntegerValuePresentValue	90
4.4.8	CCtrlBACnetReadIntegerValuePresentValue	91
4.4.9	CCtrlBACnetClearIntegerValuePresentValue	91
4.4.10	CCtrlBACnetReadBinaryInputPresentValue	91
4.4.11	CCtrlBACnetReadAnalogInputPresentValue	91
4.4.12	CCtrlBACnetWriteODI	92
4.4.13	CCtrlBACnetWrite16BitODI	92
4.4.14	CCtrlBACnetDataExchange	93
4.4.15	CCtrlBACnetReadInstance	93
4.4.16	CCtrlBACnetWriteInstance	94
5	Technische Daten	95
5.1	Übersicht	95
5.2	Merker	96
5.3	Nichtflüchtige Parameter	97
5.4	Zugriffsrechte auf den Ausgangsbereich	97
6	Fehlermeldungen	98
6.1	error: control not activated!	98
6.2	error: wrong control version	98
6.3	Launching problem	98
6.4	Es wird keine oder eine falsche Zykluszeit angezeigt.	99
6.5	Das Gateway geht in keinen Haltepunkt.	99
6.6	Das Programm geht immer in einen Haltepunkt.	99
6.7	Es lassen sich keine Ausgänge durch Control III beeinflussen.	99
7	Anzeigen der Ziffernanzeige	100

Abbildungsverzeichnis

Abb. 1. Bihl+Wiedemann Downloadbereich	13
Abb. 2. Softwarelizenz	14
Abb. 3. Startfenster von Eclipse Control III	15
Abb. 4. Eclipse settings.mak	20
Abb. 5. Neues Control III Projekt	21
Abb. 6. Debug Configuration... ..	28
Abb. 7. Markierung eines Breakpoints im Programmcode	32
Abb. 8. Erster Haltepunkt im Debug-mode	33
Abb. 9. Zweiter Haltepunkt im Debug-mode	33
Abb. 10. Darstellung Merkerbereich	96
Abb. 11. Darstellung nichtflüchtige Parameter	97
Abb. 12. Darstellung Zugriffsrechte Ausgangsdatenbereich	97
Abb. 13. error: control not activated	98
Abb. 14. error: wrong control version!	98
Abb. 15. Launching problem	99

1. Allgemein

1.1 Die verwendeten Symbole



Hinweis!

Dieses Zeichen macht auf eine wichtige Information aufmerksam.

1.2 Sicherheit

1.2.1 Bestimmungsgemäße Verwendung



Warnung!

Der Schutz von Betriebspersonal und Anlage ist nicht gewährleistet, wenn die Baugruppe nicht entsprechend ihrer bestimmungsgemäßen Verwendung eingesetzt wird.

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

1.2.2 Allgemeine Sicherheitshinweise



Warnung!

Ein anderer Betrieb, als der in dieser Anleitung beschriebene, stellt die Sicherheit und Funktion des Gerätes und angeschlossener Systeme in Frage.

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

Können Störungen nicht beseitigt werden, ist das Gerät außer Betrieb zu setzen und gegen versehentliche Inbetriebnahme zu schützen.

Reparaturen dürfen nur direkt beim Hersteller durchgeführt werden. Eingriffe und Veränderungen im Gerät sind nicht zulässig und machen jeden Anspruch auf Garantie nichtig.



Hinweis!

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

1.2.3 Entsorgung



Hinweis!

Verwendete Geräte und Bauelemente sachgerecht handhaben und entsorgen!

Unbrauchbar gewordene Geräte als Sondermüll entsorgen!

Die nationalen und örtlichen Richtlinien bei der Entsorgung einhalten!

1.3 Produktinformation

Control III erfüllt alle Aufgaben einer leistungsfähigen Klein-SPS und bietet Ihnen die Möglichkeit, alle Sensor- und Aktuatorinformationen vorzuverarbeiten.

Control III, die in die Bihl+Wiedemann ASi Master integrierte SPS-Funktionalität, bildet zusammen mit handelsüblichen ASi E/A-Bausteinen eine Klein-SPS. Die Programmierung der Klein-SPS erfolgt komplett in Standard C.

In Verbindung mit Bihl+Wiedemann ASi Mastern ab der Spezifikation ASi-3 beinhaltet Control III neben allen bisherigen Funktionen die Bedienung von bis zu 62 ASi Teilnehmern pro ASi Kreis, die zusätzliche Auswertung anfallender Peripheriefehler, wie z.B. Kurzschluss auf einer Sensorleitung sowie die direkte Bedienung von ASi Analogmodulen mit Hilfe standardisierter ASi Analogprofilen.

Somit können pro ASi Strang bis zu 248 digitale Eingänge und Ausgänge oder maximal 248 Analogwerte im reinen ASi-3 Betrieb bzw. bis zu 3 MB E/A-Daten von ASi-5 und ASi-3 Modulen im gemischten Betrieb verarbeitet werden.

Control III stellt in Verbindung mit ASi Mastern eine optimale Stand-alone-Steuerung für kleinere Maschinen und Anlagen dar.

Die Verwendung von **Control III** in ASi Gateways bietet die Möglichkeit der Vorverarbeitung von Sensoren und Aktuatoren.

Die übergeordnete SPS wird dadurch entlastet. **Control III** ermöglicht somit eine Dezentralisierung der Steuerungsaufgabe.

Ein typischer Anwendungsfall ist die schnelle Ausführung von zeitkritischen Operationen direkt im Gateway.

2 Getting Started

Eclipse ist ein weitverbreitetes und quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Eclipse wurde für die Verwendung von **Control III** so angepasst, dass es dem Anwender bei der Realisierung seines Software-Programms für **Control III** unterstützt.



2.1 Download von Eclipse Control III

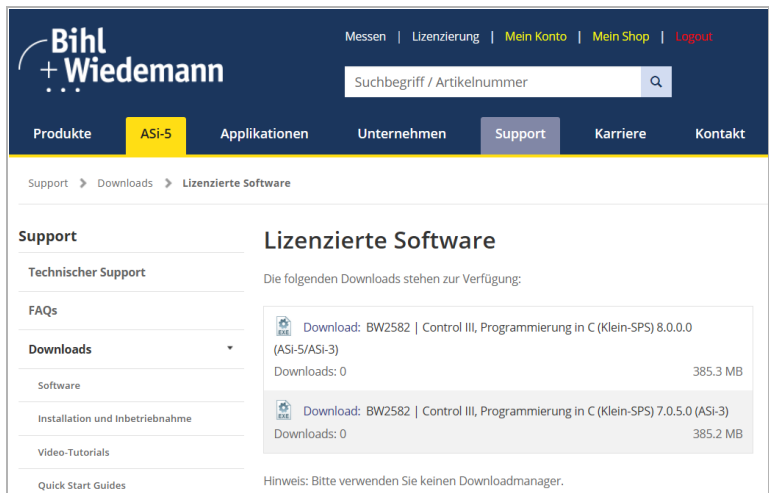


Bitte achten Sie auf die Auswahl der geeigneten Version!

- Die aktuellste Eclipse-Version ist für alle aktuellen Gateways die geeignete Programmierungsumgebung.
- Gateways mit ASI-3 Mastern (oder älter) sind ausschließlich mit der Eclipse-Version 7.0.5.0 kompatibel!

Sie finden die aktuelle Eclipse-Version zur Programmierung von **Control III** auf unserer Webseite unter:

<https://www.bihl-wiedemann.de/de/support/downloads/lizenzierte-software.html>



The screenshot shows the Bihl+Wiedemann website's 'Support' section. The navigation bar includes links for 'Messen', 'Lizenzierung', 'Mein Konto', 'Mein Shop', and 'Logout'. The main menu has 'Produkte', 'ASI-5', 'Applikationen', 'Unternehmen', 'Support' (highlighted), 'Karriere', and 'Kontakt'. The breadcrumb trail is 'Support > Downloads > Lizenzierte Software'. On the left, a 'Support' sidebar lists 'Technischer Support', 'FAQs', 'Downloads' (selected), 'Software', 'Installation und Inbetriebnahme', 'Video-Tutorials', and 'Quick Start Guides'. The main content area, titled 'Lizenzierte Software', states 'Die folgenden Downloads stehen zur Verfügung:' and lists two download items:

Download	Downloads	Size
Download: BW2582 Control III, Programmierung in C (Klein-SPS) 8.0.0.0 (ASI-5/ASI-3)	0	385.3 MB
Download: BW2582 Control III, Programmierung in C (Klein-SPS) 7.0.5.0 (ASI-3)	0	385.2 MB

A note at the bottom says: 'Hinweis: Bitte verwenden Sie keinen Downloadmanager.'

Abb. 2-1. Bihl+Wiedemann Downloadbereich

2.2 Installation von Eclipse Control III



Abb. 2-2. Softwarelizenz

Bitte entnehmen die Punkte zur Lizenzierung, Installation und Freischaltung von Control III aus dem PDF-Dokument "Softwarelizenz". Unter dem Link [„Hinweise zur Lizenzierung“](http://www.bihl-wiedemann.de/uploads/media/lizenzierung_control3_de.pdf) finden Sie eine Schritt für Schritt Anleitung zur Lizenzierung, Installation und Freischaltung.

3. Verwenden von Eclipse

Nach erfolgreicher Installation können Sie direkt mit dem Erstellen eines eigenen **Control III**-Programmes fortfahren.

3.1 Programm öffnen

Folgen Sie dem Installationspfad (standardmäßig C:/Programme) zum Ordner "eclipse_control" und führen dort die "eclipse.exe" aus.

3.2 Einführung in die Oberflächen von Eclipse

Der Eclipse Startbildschirm ist übersichtlich in die wichtigsten Bereiche unterteilt.

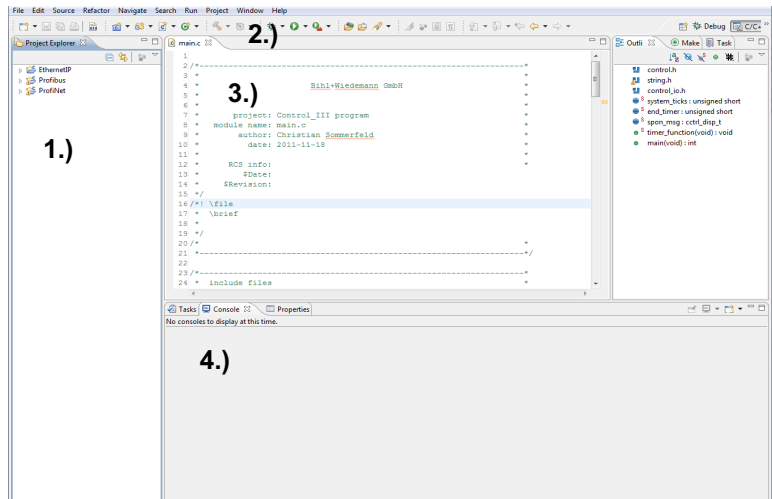


Abb. 3-3. Startfenster von Eclipse **Control III**

1. Projekt Explorer
2. Toolbar
3. Editor
4. Konsole

3.2.1 Der Projekt Explorer

Im Projekt Explorer finden Sie bei Neuinstallation drei Beispielprojekte, welche Sie direkt verwenden und anpassen können. Im Projekt Explorer können Sie alle Ihre Projekte verwalten, anpassen und testen. In jedem Projekt befinden sich nach Neuinstallation folgende Dateien:

- control_io.c
- control_io.h
- control.h
- main.c
- startup.c

- *.ld
- *.mak
- settings.mak

3.2.2 Toolbar

In der Toolbar befinden sich alle nötigen Tools, um mit **Control III** zu arbeiten.

3.2.2.1 Compiler



- **clean:**
Dient dazu den Projektordner zu säubern. Clean löscht alle durch das Kompilieren erstellte Dateien und Ordner.
- **debug:**
Kompiliert das aktuell gewählte Projekt ohne Optimierungsgrad. Die daraus resultierende control.bin wird zum Debuggen verwendet. (Siehe Kap. <Der Debugger>).
- **release:**
Kompiliert das aktuell gewählte Projekt. Die daraus resultierende control.bin ist auf Zeit optimiert, um schnellstmögliche Zykluszeiten ihres Programmcodes zu erreichen.

3.2.2.2 Debuggen



Mithilfe dieses Buttons wird unter anderem der Debugger gestartet. Sie müssen zum Debuggen noch den entsprechenden Port einstellen. Mehr Informationen finden Sie im Kap. <Start des Debuggers>.



Hinweis!

Steht das Control III Programm in einem 'Breakpoint', so steht das ganze Betriebssystem und die Feldbusschnittstelle wird nicht mehr bearbeitet.

3.2.2.3 Konfigurationstools

Der Button Konfigurationstools dient zur Kommunikation mit dem Gateway und hat folgende Funktionen:



Unlock Control:

Tool zum Freischalten von Control III im Gateway (siehe Kap. <Getting Started>).

Download Control:

Die Datei "control.bin" wird in das Gateway geschrieben.

Start Control:

Das **Control III**-Programm im Gateway wird gestartet.

**Hinweis!**

Um ein neues Programm zu starten, muss nach dem Download das laufende Programm zuerst gestoppt werden.

Download + Start Control:

Das **Control III**-Programm wird zuerst gestoppt, danach das neue Programm in das Gateway geladen und im Anschluss gestartet.

Stop Control:

Das **Control III**-Programm wird angehalten.

Set Auto Start:

Das Autostart-Flag wird gesetzt. Das **Control III**-Programm startet nach jedem Power-on automatisch.

Das Autostart-Flag wird zurückgesetzt wenn:

- Über Eclipse oder die Steuerung ein Kommandoschnittstellen-Kommando gesendet wird, um das Flag zu löschen.
 - ☐ *Abhilfe:* Setzen Sie das Auto Start Flag erneut per, Eclipse, Steuerung oder Display.
- Eine Funktion verwendet wird, welche vom Gateway nicht unterstützt wird (beispielsweise die Funktion 'setsocket' mit PROFIBUS).
 - ⇒ Spontanmeldung: 78 'FUNC NOT SUPP.'
 - ☐ *Abhilfe:* Bitte überprüfen Sie ihr **Control III**-Programm. Sie verwenden möglicherweise eine **Control III**-Funktion, die von ihrem AS-i Gateway nicht unterstützt wird.

Das Autostart-Flag wird zurückgesetzt und das Gerät neu gestartet wenn:

- Eine unerwartete Adresse im Speicher angesprungen wird. Bei einem Neustart würde der Fehler immer wieder auftreten und das Gateway wäre darauf hin nicht mehr bedienbar.
 - ⇒ Spontanmeldung: 78 'CONTROL EXEC ERR'
 - ☐ *Abhilfe:* Bitte überprüfen Sie ihr **Control III**-Programm auf fehlerhafte Speicherzugriffe und/oder verwenden Sie ggf. die Debug-Funktion des AS-i Gateways.
- Das Gateway bspw. durch äußere Einflüsse überlastet wird oder ein internen Defekt vorliegt.
 - ⇒ Spontanmeldung: 78 'CONTROL EXEC ERR'
 - ☐ *Abhilfe:* Bitte überprüfen Sie die Anschlüsse ihres AS-i Gateways. Sollte der Fehler weiterhin bestehen, senden Sie uns das defekte Gateway zur Überprüfung zu.

Clear Auto Start:

Das Autostart-Flag wird gelöscht.

Read Merker:

Die Control III Merker werden gelesen und in der Konsole dargestellt.

Read Merker zyklisch:

Die Control III Merker werden gelesen und die Darstellung in der Konsole zyklisch aktualisiert. Hiermit können z.B. Variablen zur Laufzeit beobachtet werden.

Zykluszeit:

Die aktuellen Zykluszeiten werden angezeigt.

Reset Cycle Time:

Die Zykluszeit des **Control III** wird zurückgesetzt und neu berechnet.

3.2.3 Editor

In diesem Fenster wird der gesamte C-Code geschrieben und angepasst. Der Editor unterstützt des Weiteren auch Fehler im Syntax von 'C'.

3.2.4 Konsole

Die Konsole dient als Informationsfenster. Sie gibt bspw. Fehlermeldungen oder Statusmeldungen aus und zeigt nach dem Kompilieren den verwendeten Speicherplatz des **Control III** Programms an.

3.3 Datei-Informationen

Wie bereits in Kap. <Der Projekt Explorer> beschrieben, liegen in jedem Projekt-ordner verschiedene Dateien. In diesem Kapitel werden die einzelnen Dateien genauer beschrieben.

control_io.c

Diese Datei dient als Beispiel, um ein C Programm in mehrere Teilmodule zu zerlegen, um es so übersichtlicher und besser lesbar zu gestalten.

Die beiden Funktionen 'read_bit' und 'write_bit' lesen bzw. schreiben ein entsprechendes Ausgangs- oder Eingangsbit.

```
int read_bit (AASiProcessData idi, int slave_addr, int bit)
    idi = Eingangsdaten der Slaves
    slave_addr = die Adresse des entsprechenden Slaves
    bit = Eingangsbit des Slaves (0-3)
```

```
void write_bit (AASiProcessData odi, int slave_addr, int bit, int value)
    odi = Ausgangsdaten der Slaves
    slave_addr = die Adresse des entsprechenden Slaves
    bit = Ausgangsbit des Slaves (0-3)
    value = Ausgangsbitwert (0 oder 1)
```

control_io.h

Header zu control_io.c. In dieser Datei befinden sich die Funktionsdefinitionen für 'read_bit' und 'write_bit'.

control.h

Die Header-Datei 'control.h' beinhaltet alle Datentypen und Bibliotheksfunktionen. Außerdem erläutert diese gleichzeitig deren Funktions- und Verwendungsweise (siehe Kap. <Programmieren von Control III>).

main.c

Die Funktion 'main' ist der „Startpunkt“ des eigentlichen Programmcodes. In ihr befindet sich auch die Hauptschleife des Programms (for ; ;).

startup.c

Diese Datei dient zur Initialisieren verschiedener Speicherbereiche. Die Datei ist für den Anwender ohne Bedeutung.

***.ld**

Ist das Linkerfile und die entsprechenden Speicherbereiche im Gateway fest. Die Datei ist für den Anwender ohne Bedeutung.

***.mak**

Diese Datei legt alle nötigen Informationen für den Compiler fest. Die Datei ist für den Anwender ohne Bedeutung.

settings.mak

In dieser Datei wird der Kommunikationsport für das Gateway festgelegt. Näher Informationen finden Sie im Kap. <Einstellen der Schnittstelle>.

3.4 Einstellen der Schnittstelle

In jedem Projektordner befindet sich eine Datei 'settings.mak'. In dieser Datei können Sie den Port für die Kommunikation zum Gateway einstellen. Wählen Sie hierzu die entsprechende Datei aus und tragen Sie die Verbindung zum Gateway ein. Verwenden Sie bspw. den COM Port 3 an ihrem PC dann schreiben Sie in Zeile 30 der settings.mak `PORT=COM3`. Verwenden Sie eine Ethernet-Schnittstelle, dann tragen Sie bspw. `PORT=UDP:192.168.42.149` ein.

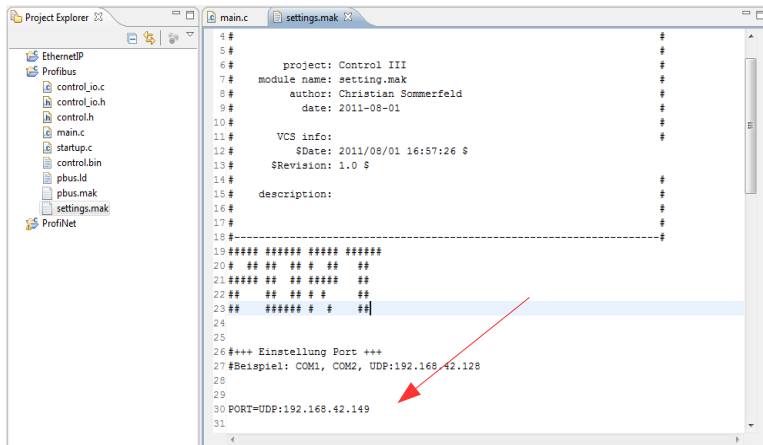


Abb. 3-4. Eclipse settings.mak

3.5 Anlegen eines neuen Projektes

Im Projektextplorer befinden sich nach einer Neuinstallation, für jedes Gateway mit **Control III** von Bihl+Wiedemann, Beispielpunkte. Es ist somit möglich, direkt nach der Installation von Eclipse **Control III** mit der Programmierung zu starten.

Sollten Sie dennoch ein neues Projekt anlegen wollen, fahren Sie wie folgt fort:

- ☐ Wählen Sie unter 'File' -> 'New' ein neues 'C Projekt'.
- ☐ Vergeben sie einen neuen Projektnamen und wählen Sie ein leeres 'Makefile Project'.
- ☐ Bestätigen den Dialog mit 'Finish'.

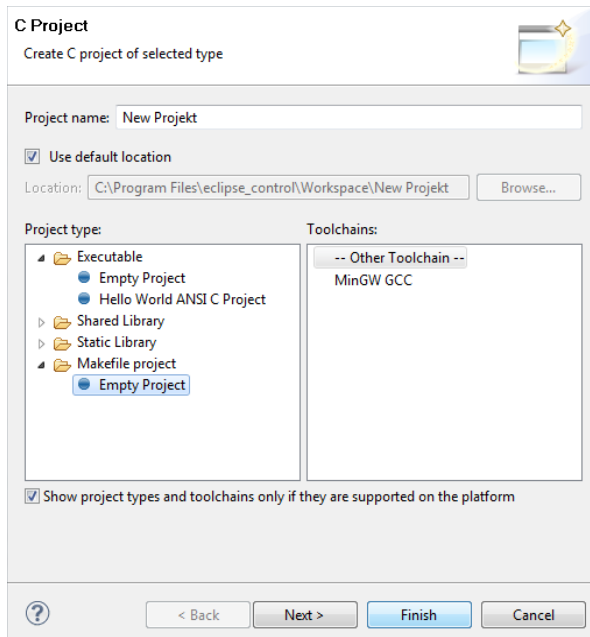


Abb. 3-5. Neues Control III Projekt

Sie finden nun im Projekt Explorer ihren neuen leeren Projektordner. Mit einem Rechtsklick auf das neue Projekt und 'Import...' können Sie alle nötigen Projektdateien für ihr Gateway hinzufügen.

- ☐ Wählen Sie 'File System' und klicken danach auf 'Next'.
- ☐ Wählen Sie nun 'Browse...' und navigieren zur Ihren Eclipse-Installation .../eclipse_control/Templates/
- ☐ Wählen Sie nun ihr verwendetes Gateway aus.
 - ☐ EthernetIP
 - ☐ Profibus
 - ☐ ProfiNET
- ☐ Wählen Sie 'Select All' und Finish.
- ☐ Um alle Einstellungen zu übernehmen, wählen Sie in folgendem Dialog 'Yes to All'.

3.6 Ein Beispielprojekt

In folgendem Kapitel wird die komplette Vorgehensweise vom Schreiben des Codes bis zum Starten im Gateway erläutert.

3.6.1 Der C-Code

Als Beispiel dient ein Programm, welches nacheinander jede Sekunde die Ausgänge eines 4E/4A Slave mit der Adresse 1 setzt und wieder löscht. Ändern Sie hierfür die main.c wie folgt:

```

/*-----*
 * include files                                *
 *-----*/
#include "control.h"
#include "string.h"
#include "control_io.h"

/*-----*
 * local definitions                            *
 *-----*/

/*-----*
 * external declarations                        *
 *-----*/

/*-----*
 * public data                                *
 *-----*/

/*-----*
 * private data                                *
 *-----*/

static unsigned short system_ticks;
static unsigned short end_timer;

/*-----*
 * private functions                            *
 *-----*/

static void timer_function ( void )
{
    /* timer interrupt every 10 ms */
    system_ticks++;
}

/*-----*
 * public functions                            *
 *-----*/
int main ( void )
{
    //initialization of the Debugger
    //cctrl_func.CCtrlBreakpoint();
    unsigned charctrl_flags;
    int i = 0;
    int x = 1;

    AASiProcessData odi[2];
    AASiProcessData idi[2];
    AASiCtrlAccODI acc_odi;
    AASiEcFlags ecflags;

```

Ausgabedatum: 16.05.2024

```

/* We want to access all odis */
for (i=0;i<32;i++)
{
    acc_odi[i] = 0xFF;
}
cctrl_func.AASiWriteCtrlAccODI ( 0, acc_odi, 0, 64 );

/* init timer function with 10ms ticks */
cctrl_func.CCtrlInitTimer ( 10, timer_function );

/* init watchdog */
//cctrl_func.CCtrlInitWdg( 10 );

    // clear outputs from slave 1
    odi[0][0] = 0x00;

for(;;)
{
    /* trigger watchdog */
    //cctrl_func.CCtrlTriggerWdg();

    /* Define data exchange for AS-i Circuit 1 and 2*/
    cctrl_func.AASiDataExchange(0, odi[0], idi[0],
&ecflags);
    cctrl_func.AASiDataExchange(1, odi[1], idi[1],
&ecflags);

    //Timer 1 100 * 10ms = 1sec.
    if ( ((unsigned short)(system_ticks - end_timer)) > 100)
    {
        // set and clear outputs circuit=1, slave=1, output=0-3
        if (x == 1) write_bit(odi[0], 1, 0, 1);
        else if (x == 2) write_bit(odi[0], 1, 1, 1);
        else if (x == 3) write_bit(odi[0], 1, 2, 1);
        else if (x == 4) write_bit(odi[0], 1, 3, 1);
        else if (x == 5) write_bit(odi[0], 1, 0, 0);
        else if (x == 6) write_bit(odi[0], 1, 1, 0);
        else if (x == 7) write_bit(odi[0], 1, 2, 0);
        else if (x == 8) write_bit(odi[0], 1, 3, 0);
        x++;

        if (x == 9) x = 1;

        end_timer = system_ticks;
    }

    /* to check Cycletime */
    cctrl_func.CCtrlEvalCycletime();

    /*read flags if we should stop control*/
    cctrl_func.CCtrlReadFlags( &ctrl_flags );
    if ( !( ctrl_flags & CCTRL_FLAG_RUN ) )
    {
        return 1;
    }
}
}

```

Ausgabedatum: 16.05.2024

3.6.2 Kompilieren

Der erstellte C-Code muss nun für den Prozessor übersetzt werden. Wählen Sie hierfür „release“ bei der Option „Compiler“ in der Toolbar. Es wird ein „release“-Ordner erstellt und in Ihrem Projektordner befindet sich nun die entsprechende Binär-Datei 'control.bin'.

3.6.3 Download

Um das neu erstellte Programm in das Gateway zu laden wählen Sie in Ihrem Projektordner die Datei settings.mak und stellen den entsprechenden Port ihres Gateways ein. Weitere Informationen hierfür finden Sie im Kap. <Einstellen der Schnittstelle>.

Als Nächstes wählen Sie in der Toolbar unter den Konfigurationstools: Download Control. Bei erfolgreichem Download erscheint in der Konsole von Eclipse folgende Meldung:

```
-----
++++ CONTROL III ++++
-----

communication port set to UDP:192.168.42.157.

writing C-Control control.bin to Master ...
.....
o.k.

closing control.bin ...

have a nice day.
```

3.6.4 Starten von Control III

Um das Programm zu starten und zu testen, wählen Sie nun unter der Toolbar / Konfigurationstools den Button Start Control. Im Display erscheint folgende Meldung:



3.7 Der Debugger

Ein Debugger ist ein Programmierwerkzeug und dient dem Diagnostizieren und Auffinden von Programmierfehlern. Wird die Debuggerfunktion verwendet, so steht bei der Diagnose das gesamte Betriebssystem.



Hinweis!

Die Debugg-Funktion dient lediglich dazu, ihr Programm in der Ausführung zu testen. Sie können mit dem Debugger keine Diagnose ihres Hardwareaufbaus durchführen.

3.7.1 Initialisierung

Um den Debugger verwenden zu können, muss dieser im C-Code initialisiert werden. Dies geschieht über folgende Codezeile:

```
//initialization of the Debugger
cctrl_func.CCtrlBreakpoint();
```

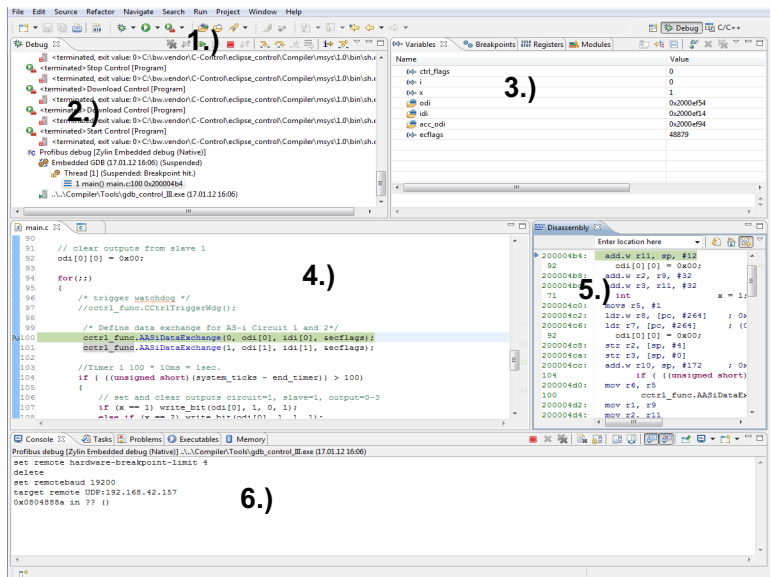
Durch das Verwenden dieser Zeile wird das Betriebssystem gestoppt und Eclipse kann mit dem Prozessor kommunizieren. Der Debugger bekommt zu diesem Zeitpunkt die Daten des Programms mitgeteilt und springt in den nächsten, vom Anwender definierten, Haltepunkt des **Control III** Programms.



Hinweis!

Die Initialisierung des Debuggers sollte nicht in der Hauptschleife des Programmcodes (for (;;)) erfolgen, da dieser Haltepunkt nicht über Eclipse gesteuert wird und daher nicht beendet werden kann.

3.7.2 Übersicht Debugger



1. Control Panel
2. Tasks
3. Debugger Übersicht
4. Codeübersicht
5. Disassembly
6. Konsole

3.7.2.1 Das Control Panel



Das Control Panel dient dazu, den Debugger zu steuern.

Resume (F8):

Mit dem Button 'Resume' läuft das **Control III** Programm weiter bis es auf einen neuen Haltepunkt stößt. Wird innerhalb von 10 Sekunden kein neuer Haltepunkt erreicht so beendet sich der Debugger in Eclipse automatisch und es wird eine entsprechende Meldung in der Konsole ausgegeben.



Hinweis!

Bei Benutzung des Debuggers sind keine Delays länger als 10 Sekunden möglich, da der Debugger sonst automatisch beendet wird. Dies dient dazu, bei fehlender oder falscher Kommunikation den Debugger zu beenden.

Terminate (Ctrl + F2):

Terminate beendet den Debugger und lässt das **Control III** Programm weiterlaufen, auch wenn noch Haltepunkte über Eclipse im Code vorhanden sind.

Step Into (F5):

Der Button 'Step Into' wird dazu benutzt, um eine Programmzeile im Code weiterzuspringen.



Hinweis!

Die StepInto-Funktion kann nur zuverlässig funktionieren, wenn der Programmcode als „debug“ übersetzt worden ist.

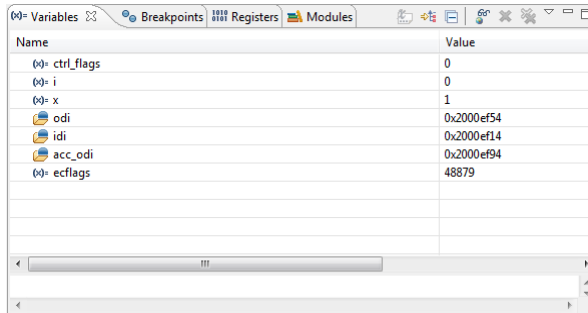
Step Over (F6):

'Step Over' wird verwendet, um bspw. einen Funktionsaufruf zu überspringen.

3.7.2.2 Tasks

In diesem Fenster werden alle verwendeten Programme, welche über Eclipse gesteuert werden, angezeigt. Dieses Fenster ist für den Anwender von keiner großer Bedeutung.

3.7.2.3 Debugger Übersicht



Name	Value
(0)= ctrl_flags	0
(0)= i	0
(0)= x	1
odi	0x2000ef54
idi	0x2000ef14
acc_odi	0x2000ef94
(0)= eflags	48879

Variablen:

Im Reiter Variablen in der „Debugger-Übersicht“ können alle vorhandenen Variablen und deren Werte angezeigt werden. Mit einem Rechtsklick in das Fenster können mit 'Add Global Variables...' weitere Variablen der Ansicht hinzugefügt werden. Der Wert der Variablen wird immer in Hex angezeigt. Die Ansicht kann mit einem Rechtsklick und dem Punkt 'Format' auf binär oder dezimal geändert werden.

Breakpoints:

Im Reiter Breakpoints befindet sich eine Übersicht über alle in Eclipse gesteuerten bzw. verwendeten Haltepunkte inklusive der Programmzeile. Einzelne Haltepunkte können mit einem Rechtsklick entfernt werden.

3.7.2.4 Codeübersicht

In diesem Fenster sehen Sie jederzeit, an welcher Stelle im Programmcode sich der Debugger gerade befindet. Hier können Sie auch mit einem Doppelklick neben der entsprechenden Codezeile, einen neuen Haltepunkt setzen oder löschen.



Hinweis!

Sobald der Debugger gestartet wird, sieht man zuerst ein leeres Fenster mit dem Inhalt: „Source not found“. Der Debugger befindet sich zu diesem Zeitpunkt im Betriebssystem und kennt den dazugehörigen C-Code nicht.

3.7.2.5 Disassembly

Das Disassembly-Fenster zeigt genau wie die Codeübersicht an, an welcher Stelle sich der Debugger gerade befindet. Hier werden sowohl die Speicheradresse als auch der dazugehörige assembler-Code angezeigt.

3.7.2.6 Konsole

Die Konsole dient als Ausgabe Fenster und informiert Sie über den Status des Debuggers.

3.7.3 Start des Debuggers

Der Debugger wird über das Control Panel gestartet.



Bevor Sie Verbindung mit dem Target aufnehmen können, müssen Sie unter 'Debug-Configuration' die Schnittstelle zum Gateway definieren. Klicken Sie hierzu auf den Reiter 'Commands' und geben unter target remote die Schnittstelle an. (bspw. target remote UDP:192.168.42.33 oder COM3).

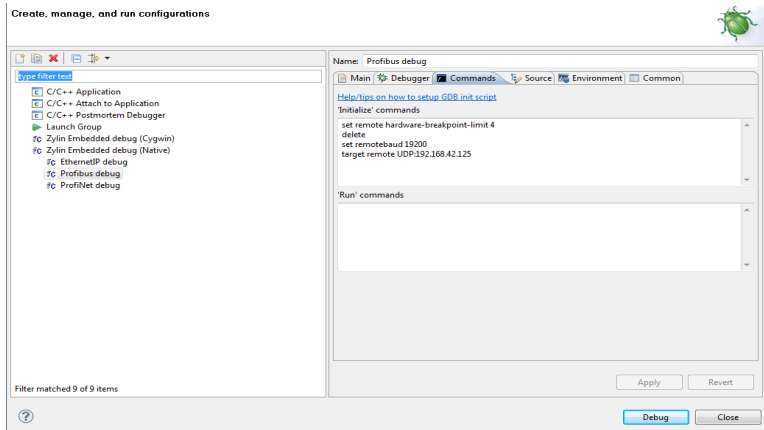


Abb. 3-6. Debug Configuration...

3.7.4 Beispiel

Als Beispiel für den Debugger dient ein Programm, welches in jedem Hauptschleifendurchlauf ein Ausgangsbit (0-3) des Slaves mit der Adresse 1 setzt. Verwenden Sie hierzu folgenden Programmcode.

Erstellen Sie nun zwei Haltepunkte, indem Sie am Rand des Programmcodes auf die entsprechende Code-Zeile doppelklicken. In unserem Beispiel verwenden wir die beiden Zeilen 103 und 109. Die Zeile, in der ein Haltepunkt eingefügt wurde, wird durch einen Punkt markiert. Diese Zeilen werden später in der Debugger Übersicht unter Breakpoint dargestellt und dem Prozessor bei der Initialisierung mitgeteilt.

```

*-----*
*   include files                               *
*-----*/
#include "control.h"
#include "string.h"
#include "control_io.h"

/*-----*
*   local definitions                           *
*-----*/

/*-----*
*   external declarations                       *
*-----*/

/*-----*
*   public data                                *
*-----*/

/*-----*
*   private data                              *
*-----*/

static unsigned short system_ticks;
static unsigned short end_timer;

/*-----*
*   private functions                           *
*-----*/

static void timer_function ( void )
{
    /* timer interrupt every 10 ms */
    system_ticks++;
}

/*-----*
*   public functions                            *
*-----*/

```

```

int main ( void )
{
    //initialization of the Debugger
    cctrl_func.CCtrlBreakpoint();

    unsigned char cctrl_flags;
    int          i = 0;
    int          x = 1;

    AASiProcessData odi[2];
    AASiProcessData idi[2];
    AASiCtrlAccODI acc_odi;
    AASiEcFlags ecflags;

    /* We want to access all odis */
    for (i=0;i<32;i++)
    {
        acc_odi[i] = 0xFF;
    }
    cctrl_func.AASiWriteCtrlAccODI ( 0, acc_odi, 0, 64 );

    /* init timer function with 10ms ticks */
    cctrl_func.CCtrlInitTimer ( 10, timer_function );

    /* init watchdog */
    //cctrl_func.CCtrlInitWdg( 10 );

    // clear outputs from slave 1
    odi[0][0] = 0x00;

```

```

for (;;)
{
    /* trigger watchdog */
    //cctrl_func.CCtrlTriggerWdg();

    /* Define data exchange for AS-i Circuit 1 and 2*/
    cctrl_func.AASiDataExchange(0, odi[0], idi[0],
&ecflags);
    cctrl_func.AASiDataExchange(1, odi[1], idi[1],
&ecflags);

    if (x == 1)
    {
        write_bit(odi[0], 1, 0, 1);
    }
    else if (x == 2)
    {
        write_bit(odi[0], 1, 1, 1);
    }
    else if (x == 3)
    {
        write_bit(odi[0], 1, 2, 1);
    }
    else if (x == 4)
    {
        write_bit(odi[0], 1, 3, 1);
        x = 1;
    }
    x++;

    /* check Cycletime */
    cctrl_func.CCtrlEvalCycletime();

    /*read flags if we should stop control*/
    cctrl_func.CCtrlReadFlags( &ctrl_flags );
    if ( !( ctrl_flags & CCTRL_FLAG_RUN ) )
    {
        return 1;
    }
}

}

/*-----*
* eof                                           *
*-----*/

```

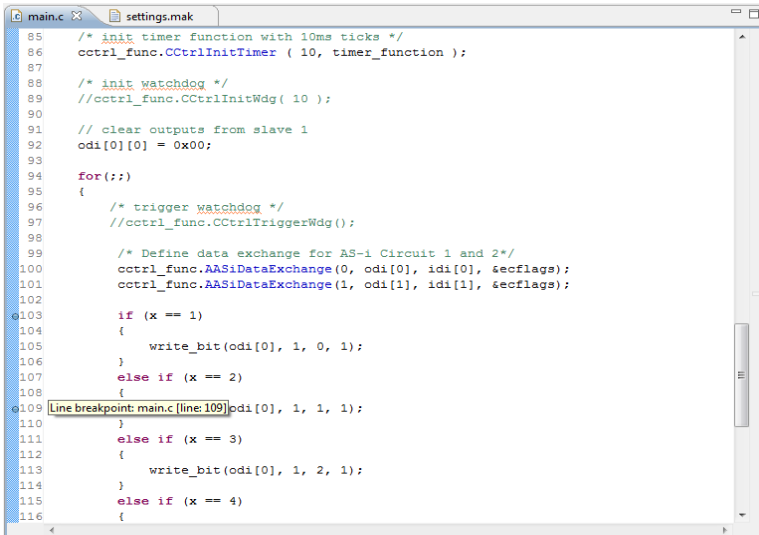
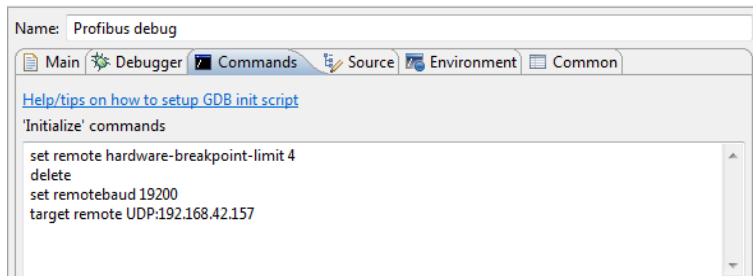


Abb. 3-7. Markierung eines Breakpoints im Programmcode

Kompilieren Sie nun das Programm, indem Sie in der Toolbar unter Compiler auf 'Debug' klicken. Es wird ein neues control.bin-File ohne Optimierungsgrad erzeugt. Laden Sie dieses File, wie schon im Kap. <Editor> beschrieben, in das Gateway.

3.7.4.1 Start des Debuggers

Bevor Sie mit dem Debuggen beginnen können, müssen Sie den Port einstellen. Klicken Sie in der Toolbar unter Debugger auf 'Debug Configurations...'. Öffnen Sie hier den Reiter 'Commands' und tragen unter 'target remote' ihre Schnittstelle (bspw. UDP:192.168.42.157 oder COM1) zum Gateway ein und wählen 'Apply'.



Sollte ihr **Control III** Programm bereits in einem Haltepunkt stehen, so können Sie den Debugger direkt über 'Debug' starten. Ein Haltepunkt wird ihnen über das Display im Gateway mit der Nummer '79' angezeigt.

79
breakpoint

Sollte dies nicht der Fall sein, beenden Sie die Eingabe mit 'Close' und starten ihr Programm. Sie können den Debugger jetzt direkt mit einem Klick auf bspw. 'Profilbus debug' in der Toolbar starten.

3.7.4.2 Debugger benutzen

Nachdem Sie den Debugger gestartet haben, wird dieser für die Verwendung konfiguriert. Das Debugging-Fenster wird automatisch von Eclipse geöffnet. Sie sehen jetzt unter Breakpoints den ersten Haltepunkt, der vom Programmcode ausgeführt wird. Dies ist die Initialisierung des Debuggers. Sie bekommen in diesem Fall ein leeres Fenster angezeigt. Klicken Sie hierfür auf 'Resume'. Der Programmcode wird bis zu der entsprechende Zeile mit dem ersten Haltepunkt ausgeführt und gestoppt.

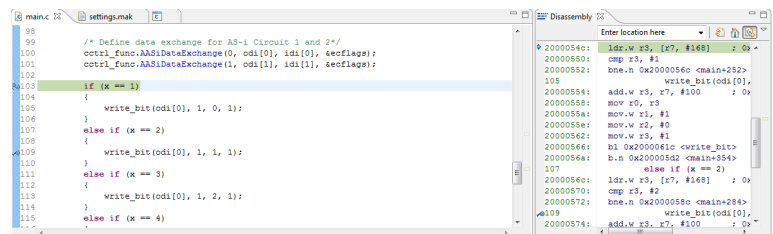


Abb. 3-8. Erster Haltepunkt im Debug-mode

Des Weiteren können Sie mit einem Klick auf 'Variables' in der Debugger Übersicht die Werte aller verwendeten Variablen anzeigen lassen (siehe hierzu auch Kap. <Debugger Übersicht>). Klicken Sie auf 'Resume'. Der Debugger bleibt an erneut an dieser Stelle stehen und nicht am zweiten Haltepunkt. Dies liegt daran, dass der zweite Haltepunkt erst erreicht wird, wenn die Variable 'x' in unserem Beispielprogramm den Wert 2 hat. Sie können sich den Wert der Variable anzeigen lassen, indem Sie mit dem Mauszeiger über die entsprechende Variable gehen. Die Variable x hat nun den Wert 2. Betätigen Sie erneut 'Resume', um zum Haltepunkt in Zeile 109 zu springen.

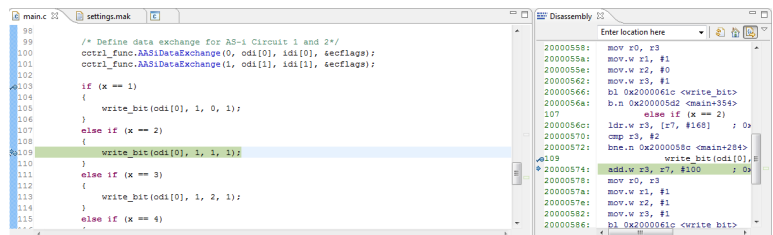


Abb. 3-9. Zweiter Haltepunkt im Debug-mode

Bei dieser Codezeile bietet sich die Möglichkeit zwischen 'Step Over' oder 'Step Into'. Bei 'Step over' springt das Programm in Zeile 120, also überspringt den Funktionsaufruf 'write_bit(...)' und macht mit der nächsten gültigen Zeile im Code weiter. Bei 'Step Into' wird die entsprechende Datei (control_io.c) geöffnet und der 'Debugg-Modus' an der entsprechenden Stelle fortgeführt. Möchten Sie wieder zum nächsten Haltepunkt springen, so wählen Sie erneut den Button 'Resume'. Beenden Sie den Debugger mit einem Klick auf 'Terminate'.

4 Programmieren von Control III

Das Programmieren von **Control III** basiert auf der weitverbreiteten und höchst modularen Programmiersprache 'C'. Um ein Control III Programm zu erstellen, müssen alle Aufgaben, welche das Gateway übernehmen soll, in 'C' geschrieben und in das Gateway geladen werden.

Um mit **Control III** zu programmieren, beinhaltet die Header-Datei 'control.h' eine Reihe von Bibliotheksfunktionen, welche verwendet werden können, um bestimmte Master-Funktionen auszuführen. Im Folgenden werden diese Funktionen aufgelistet und erläutert.

4.1 Übersicht der ASI-3 Befehle

AASiDataExchange	ASi Data Exchange dient zum Austausch von ASi Daten und liest die 'execution control' flags.
AASiReadIDI	Liest die Eingangsdaten von ASi Teilnehmern und die 'execution control' flags.
AASiWriteODI	Schreibt die Ausgangsdaten von ASi Teilnehmern.
AASiReadODI	Liest die Ausgangsdaten aus dem ASi Master.
AASiWritePP	Schreibt die ständigen Parameter eines ASi Teilnehmers im Master.
AASiReadPP	Liest die ständigen Parameter eines ASi Teilnehmers im Master.
AASiSendParameter	Sendet die Parameter zu einem ASi Teilnehmer.
AASiReadPI	Liest die aktuellen Parameter eines ASi Teilnehmers.
AASiStorePI	Speichert die aktuellen ASi Teilnehmer-Parameter als ständige Parameter.
AASiReadDuplicateAddrList	Liest die Liste aller Doppeladressen.
AASiReadFaultDetector	Liest Überspannung, Nois, EFLT und Doppeladressen.
AASiWritePCD	Schreibt die projektierte Konfiguration eines ASi Teilnehmers.
AASiReadPCD	Liest die projektierte Konfiguration eines ASi Teilnehmers.
AASiStoreCDI	Speichert die aktuelle Konfiguration als ständige Konfiguration.
AASiReadCDI	Liest die aktuelle Konfiguration eines ASi Teilnehmers.
AASiWriteExtID1	Schreibt den extended ID-Code 1 von ASi Teilnehmer 0.
AASiWriteLPS	Schreibt die projektierten ASi Teilnehmer.
AASiReadLPS	Liest die projektierten ASi Teilnehmer.
AASiReadLAS	Liest die aktivierten ASi Teilnehmer
AASiReadLDS	Liest die erkannten ASi Teilnehmer.
AASiReadLCS	Liest alle fehlerhaften ASi Teilnehmer.
AASiWriteLOS	Liste der ASi Teilnehmer welche bei einem Konfigurationsfehler offline gehen sollen.
AASiReadLOS	Liest die Liste der Teilnehmer, welche bei einem Konfigurationsfehler offline gehen.
AASiReadLPF	Liest die Liste der Teilnehmer mit Peripheriefehler.
AASiReadEcFlags	Liest die 'execution control' flags.
AASiSetConfigMode	Setzt den ASi Master in den Konfigurationsmodus oder in den geschützten Betriebsmodus.

Tab. 4-1.

AASiWriteHiFlags	Schreibt die Host-Interface-Flags des ASi Masters.
AASiReadHiFlags	Liest die Host-Interface-Flags des ASi Masters.
AASiAddressSlave	Ändert die Adresse eines ASi Teilnehmers.
AASiExecuteCommand	Direkt zu sendendes ASi Kommando.
AASiReadAllConfig	Liest alle Konfigurationsdaten (z.B. LPS, PP[] und PCD []) aller angeschlossenen Teilnehmer.
AASiWriteAllConfig	Schreibt alle Konfigurationsdaten (z.B. LPS, PP[] und PCD []) aller angeschlossenen ASi Teilnehmer.
AASiReadErrorCounters	Liest den Slave-Error-Counter
AASiMailbox	Generic Mailbox-Funktion.
AASiWrite16BitODI	Schreibt vier Kanäle 16bit ODI an einen ASi Teilnehmer mit z.B. Analog-Teilnehmer Profil 7.3 oder 7.4.
AASiRead16BitODI	Liest vier 16bit ODI Kanäle eines ASi Teilnehmer mit z.B. Analog-Teilnehmer Profil 7.3 oder 7.4.
AASiRead16BitIDI	Liest vier 16bit IDI Kanäle eines ASi Teilnehmers mit z.B. Analog-Teilnehmer Profil 7.3 oder 7.4.
AASiReadCtrlAccODI	Liest die Control III Berechtigung, um Ausgangsdaten zu verändern.
AASiWriteCtrlAccODI	Schreibt die Control III Berechtigungen der ASi Teilnehmer, um Ausgangsdaten zu verändern.
AASiReadCtrlAccAODI	Liest die Control III Berechtigung, um analoge Ausgangsdaten zu verändern.
AASiWriteCtrlAccAODI	Schreibt die Control III Berechtigungen der ASi Teilnehmer, um analoge Ausgangsdaten zu verändern.
AASiReadPE	Liest Echo Parameter eines ASi Teilnehmers.
AASiSetCPRL	Schreibt Liste der zyklischen Parameterabfragen.
AASiSendParameterNon-Blocking	Sendet Parameter an einen ASi Teilnehmer ohne auf das ASi Teilnehmer-Echo zu warten.
GetSystime	Gibt die aktuelle System-Zeit zurück

Tab. 4-1.

4.1.1 **ASi Data Exchange**

ASi Data Exchange dient zum Austausch von ASi Daten zwischen dem Master und der Applikation und liest die 'execution control' flags.

```
int (*AASiDataExchange) (unsigned char Circuit, AASiProcessData  
ODI, AASiProcessData IDI, AASiEcFlags *EcFlags);
```

Parameter:

Circuit: ASi Master Kreis
ODI: 32 Byte Ausgangsdaten

Return:

IDI: 32 Byte Eingangsdaten
EcFlags: Execution control flags (2 Bytes) des ASi Master.

ASi Kreis 1 / 2 : Eingangsdatenabbild IDI

Bit	7	6	5	4	3	2	1	0	Bit	7	6	5	4	3	2	1	0
Byte	D3	D2	D1	D0	D3	D2	D1	D0	Byte	D3	D2	D1	D0	D3	D2	D1	D0
0	TN-Nr. 1/1A				TN-Nr. 0/0A				1	TN-Nr. 3/3A				TN-Nr. 2/2A			
2	TN-Nr. 5/5A				TN-Nr. 4/4A				3	TN-Nr. 7/7A				TN-Nr. 6/6A			
4	TN-Nr. 9/9A				TN-Nr. 8/8A				5	TN-Nr. 11/11A				TN-Nr. 10/10A			
6	TN-Nr. 13/13A				TN-Nr. 12/12A				7	TN-Nr. 15/15A				TN-Nr. 14/14A			
8	TN-Nr. 17/17A				TN-Nr. 16/16A				9	TN-Nr. 19/19A				TN-Nr. 18/18A			
10	TN-Nr. 21/21A				TN-Nr. 20/20A				11	TN-Nr. 23/23A				TN-Nr. 22/22A			
12	TN-Nr. 25/25A				TN-Nr. 24/24A				13	TN-Nr. 27/27A				TN-Nr. 26/26A			
14	TN-Nr. 29/29A				TN-Nr. 28/28A				15	TN-Nr. 31/31A				TN-Nr. 30/30A			
16	TN-Nr. 1/1B				TN-Nr. 0/0B				17	TN-Nr. 3/3B				TN-Nr. 2/2B			
18	TN-Nr. 5/5B				TN-Nr. 4/4B				19	TN-Nr. 7/7B				TN-Nr. 6/6B			
20	TN-Nr. 9/9B				TN-Nr. 8/8B				21	TN-Nr. 11/11B				TN-Nr. 10/10B			
22	TN-Nr. 13/13B				TN-Nr. 12/12B				23	TN-Nr. 15/15B				TN-Nr. 14/14B			
24	TN-Nr. 17/17B				TN-Nr. 16/16B				25	TN-Nr. 19/19B				TN-Nr. 18/18B			
26	TN-Nr. 21/21B				TN-Nr. 20/20B				27	TN-Nr. 23/23B				TN-Nr. 22/22B			
28	TN-Nr. 25/25B				TN-Nr. 24/24B				29	TN-Nr. 27/27B				TN-Nr. 26/26B			
30	TN-Nr. 29/29B				TN-Nr. 28/28B				31	TN-Nr. 31/31B				TN-Nr. 30/30B			

Tab. 4-2. Eingangsdatenabbild IDI

ASi Kreis 1 / 2 : Ausgangsdatenabbild ODI

Bit	7	6	5	4	3	2	1	0	Bit	7	6	5	4	3	2	1	0
Byte	D3	D2	D1	D0	D3	D2	D1	D0	Byte	D3	D2	D1	D0	D3	D2	D1	D0
0	TN-Nr. 1/1A				TN-Nr. 0/0A				1	TN-Nr. 3/3A				TN-Nr. 2/2A			
2	TN-Nr. 5/5A				TN-Nr. 4/4A				3	TN-Nr. 7/7A				TN-Nr. 6/6A			
4	TN-Nr. 9/9A				TN-Nr. 8/8A				5	TN-Nr. 11/11A				TN-Nr. 10/10A			
6	TN-Nr. 13/13A				TN-Nr. 12/12A				7	TN-Nr. 15/15A				TN-Nr. 14/14A			
8	TN-Nr. 17/17A				TN-Nr. 16/16A				9	TN-Nr. 19/19A				TN-Nr. 18/18A			
10	TN-Nr. 21/21A				TN-Nr. 20/20A				11	TN-Nr. 23/23A				TN-Nr. 22/22A			
12	TN-Nr. 25/25A				TN-Nr. 24/24A				13	TN-Nr. 27/27A				TN-Nr. 26/26A			
14	TN-Nr. 29/29A				TN-Nr. 28/28A				15	TN-Nr. 31/31A				TN-Nr. 30/30A			
16	TN-Nr. 1/1B				TN-Nr. 0/0B				17	TN-Nr. 3/3B				TN-Nr. 2/2B			
18	TN-Nr. 5/5B				TN-Nr. 4/4B				19	TN-Nr. 7/7B				TN-Nr. 6/6B			
20	TN-Nr. 9/9B				TN-Nr. 8/8B				21	TN-Nr. 11/11B				TN-Nr. 10/10B			
22	TN-Nr. 13/13B				TN-Nr. 12/12B				23	TN-Nr. 15/15B				TN-Nr. 14/14B			
24	TN-Nr. 17/17B				TN-Nr. 16/16B				25	TN-Nr. 19/19B				TN-Nr. 18/18B			
26	TN-Nr. 21/21B				TN-Nr. 20/20B				27	TN-Nr. 23/23B				TN-Nr. 22/22B			
28	TN-Nr. 25/25B				TN-Nr. 24/24B				29	TN-Nr. 27/27B				TN-Nr. 26/26B			
30	TN-Nr. 29/29B				TN-Nr. 28/28B				31	TN-Nr. 31/31B				TN-Nr. 30/30B			

Tab. 4-3. Ausgangsdatenabbild ODI

4.1.2 ASi Read IDI

ASi Read IDI liest die Eingangsdaten von ASi Teilnehmern und die execution control flags.

```
int (*AASiReadIDI) (unsigned char Circuit, AASiProcessData IDI,
AASiSlaveAddr First, unsigned char Amount, AASiEcFlags *EcFlags);
```

Parameter:

Circuit: ASi Master Kreis
 First: Index des ersten ASi Teilnehmers
 Amount: Anzahl der zu lesenden ASi Teilnehmer

Return:

IDI: 32 Byte Eingangsdaten
 Jeder ASi Teilnehmer verwendet 4 Bit (Nibble) eines Bytes. Unbenutzte Bytes werden auf Null gesetzt
 EcFlags: Execution control flags (2 Bytes) des ASi Master

4.1.3 ASi Write ODI

ASi Write ODI schreibt die Ausgangsdaten von ASi Teilnehmer.

```
int (*AASiWriteODI) (unsigned char Circuit, AASiProcessData ODI,
AASiSlaveAddr First, unsigned char Amount);
```

Parameter:

Circuit: ASi Master Kreis
 ODI: 32 Byte Ausgangsdaten
 First: Index des ersten ASi Teilnehmers
 Amount: Anzahl der zu schreibenden ASi Teilnehmer

Jeder ASi Teilnehmer verwendet 4 Bit (Nibble) eines Bytes. Unbenutzte Bytes werden auf Null gesetzt.

Return: —

4.1.4 ASi Read ODI

ASi Read ODI liest die Ausgangsdaten aus dem ASi Master.

```
int (*AASiReadODI) (unsigned char Circuit, AASiProcessData ODI);
```

Parameter:

Circuit: ASi Master Kreis

Return:

ODI: 32 Byte Ausgangsdaten

Jeder ASi Teilnehmer verwendet 4 Bit (Nibble) eines Bytes. Unbenutzte Bytes werden auf Null gesetzt.

4.1.5 ASi Write Permanent Parameter

ASi Write Permanent Parameter schreibt die Dauerparameter eines ASi Teilnehmers im Master.

```
int (*AASiWritePP) (unsigned char Circuit, AASiSlaveAddr Address, AASiSlaveData PP);
```

Parameter:

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmers

PP: permanente Parameter (low Nibble)

Return: —

4.1.6 ASi Read Permanent Parameter

ASi Read Permanent Parameter liest die ständigen Parameter eines ASi Teilnehmers im Master.

```
int (*AASiReadPP) (unsigned char Circuit, AASiSlaveAddr Address, AASiSlaveData *PP);
```

Parameter

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmers

Return:

PP: permanente Parameter (low nibble)

4.1.7 ASi Send Parameter

ASi Send Parameter sendet die Parameter zu einem ASi Teilnehmer.

```
int (*AASiSendParameter) (unsigned char Circuit, AASiSlaveAddr Address, AASiSlaveData PI, AASiSlaveData *Return);
```

Parameter:

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmers

PI: zu sendende Parameter (low Nibble)

Return:

Return: Bei einem Fehler wird PI invertiert zurückgegeben.

4.1.8 ASi Read PI

ASi Read PI liest die aktuellen Parameter eines ASi Teilnehmers.

```
int (*AASiReadPI) (unsigned char Circuit, AASiSlaveAddr Address,
AASiSlaveData *PI);
```

Parameter:

Circuit: ASi Master Kreis
Address: Adresse des ASi Teilnehmers

Return:

PI: zu sendende Parameter (low Nibble)

4.1.9 ASi Store PI

ASi Store PI speichert die aktuelle ASi Teilnehmer-Parameter als ständige Parameter.

```
int (*AASiStorePI) (unsigned char Circuit);
```

Parameter:

Circuit: ASi Master Kreis
Address: Adresse des ASi Teilnehmers

Return: —

4.1.10 ASi Read Duplicate Address List

ASi Read Duplicate Address List liest die Liste aller Doppeladressen.

```
int (*AASiReadDuplicateAdrList) (unsigned char Circuit,
AASiSlaveList, *DpAdrList);
```

Parameter:

Circuit: ASi Master Kreis

Return:

DpAdrList: Liste der doppelten Adressen

4.1.11 ASi Read Fault Detector

ASi Read Fault Detector liest Überspannung, Nois, EFLT und Doppeladressen.

```
int (*AASiReadFaultDetector) (unsigned char Circuit,
unsigned char *pucFaultDetectorActiv,
unsigned char *pucFaultDetectorHistoric);
```

Parameter:

Circuit: ASi Master Kreis

Return:

PucFaultDetectorActiv: Aktiver Fehlererfasser
PucFaultDetectorHistoric: Historischer Fehlererfasser

4.1.12 ASi Write PCD

ASi Write PCD schreibt die projektierte Konfiguration eines ASi Teilnehmers.

```
int (*AASiWritePCD) (unsigned char Circuit, AASiSlaveAddr
Address, AASiConfigData PCD);
```

Parameter:

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmers

Return:

PCD: zu schreibende projektierte Konfiguration des ASi Teilnehmers

Bit	7	6	5	4	3	2	1	0
Byte	D3	D2	D1	D0	D3	D2	D1	D0
0	TN-Nr. ID				I/O Configuration			
1	Extended ID2				Extended ID1			

Tab. 4-4. PCD Konfiguration

4.1.13 ASi Read PCD

ASi Read PCD liest die projektierte Konfiguration eines ASi Teilnehmers.

```
int (*AASiReadPCD) (unsigned char Circuit, AASiSlaveAddr Address,
AASiConfigData *PCD);
```

Parameter:

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmers

Return:

PCD: projektierte Konfiguration des ASi Teilnehmers (siehe Tab. <PCD Konfiguration>).

4.1.14 ASi Store CDI

ASi Store CDI speichert die aktuelle Konfiguration als ständige Konfiguration.

```
int (*AASiStoreCDI) (unsigned char Circuit);
```

Parameter:

Circuit: ASi Master Kreis

4.1.15 ASi Read CDI

ASi Read CDI liest die aktuelle Konfiguration eines ASi Teilnehmers.

```
int (*AASiReadCDI) (unsigned char Circuit, AASiSlaveAddr Address,
AASiConfigData *CDI);
```

Parameter:

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmer

Return:

CDI: Konfiguration des ASi Teilnehmers

Bit	7	6	5	4	3	2	1	0
Byte	D3	D2	D1	D0	D3	D2	D1	D0
0	TN-Nr. ID				I/O Configuration			
1	Extended ID2				Extended ID1			

Tab. 4-5. CDI Konfiguration

4.1.16 ASi Write Extended ID1

ASi Write Extended ID1 schreibt den extended ID-Code 1 von ASi Teilnehmer 0.

```
int (*AASiWriteExtID1) (unsigned char Circuit, AASiSlaveData
ID1);
```

Parameter:

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmers

ID1: Extended ID-Code 1

Return:

Spezielle Fehlercodes, welche den Grund der fehlerhaften Übertragung beschreiben.

4.1.17 ASi Write LPS

ASi Write LPS schreibt die projizierten Salves.

```
int (*ASiWriteLPS) (unsigned char Circuit, AASiSlaveList LPS);
```

Parameter:

Circuit: ASi Master Kreis

LPS: 8 Byte ASi Teilnehmer-Liste

Jedes Bit in der LPS entspricht einem ASi Teilnehmer wie folgt:

Bit	ASi Teilnehmer	Bit	ASi Teilnehmer	Bit	ASi Teilnehmer	Bit	ASi Teilnehmer
0	TN-Nr. 0/0A kann nicht eingestellt werden!	16	TN-Nr. 16/16A	32	TN-Nr. 0/0B kann nicht eingestellt werden!	48	TN-Nr. 16/16B
1	TN-Nr. 1/1A	17	TN-Nr. 17/17A	33	TN-Nr. 1/1B	49	TN-Nr. 17/17B
2	TN-Nr. 2/2A	18	TN-Nr. 18/18A	34	TN-Nr. 2/2B	50	TN-Nr. 18/18B
3	TN-Nr. 3/3A	19	TN-Nr. 19/19A	35	TN-Nr. 3/3B	51	TN-Nr. 19/19B
4	TN-Nr. 4/4A	20	TN-Nr. 20/20A	36	TN-Nr. 4/4B	52	TN-Nr. 20/20B
5	TN-Nr. 5/5A	21	TN-Nr. 21/21A	37	TN-Nr. 5/5B	53	TN-Nr. 21/21B
6	TN-Nr. 6/6A	22	TN-Nr. 22/22A	38	TN-Nr. 6/6B	54	TN-Nr. 22/22B
7	TN-Nr. 7/7A	23	TN-Nr. 23/23A	39	TN-Nr. 7/7B	55	TN-Nr. 23/23B
8	TN-Nr. 8/8A	24	TN-Nr. 24/24A	40	TN-Nr. 8/8B	56	TN-Nr. 24/24B
9	TN-Nr. 9/9A	25	TN-Nr. 25/25A	41	TN-Nr. 9/9B	57	TN-Nr. 25/25B
10	TN-Nr. 10/10A	26	TN-Nr. 26/26A	42	TN-Nr. 10/10B	58	TN-Nr. 26/26B
11	TN-Nr. 11/11A	27	TN-Nr. 27/27A	43	TN-Nr. 11/11B	59	TN-Nr. 27/27B
12	TN-Nr. 12/12A	28	TN-Nr. 28/28A	44	TN-Nr. 12/12B	60	TN-Nr. 28/28B
13	TN-Nr. 13/13A	29	TN-Nr. 29/29A	45	TN-Nr. 13/13B	61	TN-Nr. 29/29B
14	TN-Nr. 14/14A	30	TN-Nr. 30/30A	46	TN-Nr. 14/14B	62	TN-Nr. 30/30B
15	TN-Nr. 15/15A	31	TN-Nr. 31/31A	47	TN-Nr. 15/15B	63	TN-Nr. 31/31B

Tab. 4-6. LPS

Der ASi Teilnehmer ist projiziert, wenn das Bit gesetzt ist.

4.1.18 ASi Read LPS

ASi Read LPS liest die projizierten ASi Teilnehmer.

```
int (*ASiReadLPS) (unsigned char Circuit, AASiSlaveList *LPS);
```

Parameter:

Circuit: ASi Master Kreis

Return:

LPS: 8 Byte ASi Teilnehmer-Liste (siehe Tab. <LPS>)

Der ASi Teilnehmer ist projiziert, wenn das Bit gesetzt ist.

4.1.19 ASI Read LAS

ASI Read LAS liest die aktivierten ASI Teilnehmers.

```
int (*AASiReadLAS) (unsigned char Circuit, AASiSlaveList *LAS);
```

Parameter:

Circuit: ASI Master Kreis

LAS: 8 Byte ASI Teilnehmer-Liste

Jedes Bit in der LAS entspricht einem ASI Teilnehmer.

Bit	ASI Teilnehmer	Bit	ASI Teilnehmer	Bit	ASI Teilnehmer	Bit	ASI Teilnehmer
0	TN-Nr. 0/0A	16	TN-Nr. 16/16A	32	TN-Nr. 0/0B	48	TN-Nr. 16/16B
1	TN-Nr. 1/1A	17	TN-Nr. 17/17A	33	TN-Nr. 1/1B	49	TN-Nr. 17/17B
2	TN-Nr. 2/2A	18	TN-Nr. 18/18A	34	TN-Nr. 2/2B	50	TN-Nr. 18/18B
3	TN-Nr. 3/3A	19	TN-Nr. 19/19A	35	TN-Nr. 3/3B	51	TN-Nr. 19/19B
4	TN-Nr. 4/4A	20	TN-Nr. 20/20A	36	TN-Nr. 4/4B	52	TN-Nr. 20/20B
5	TN-Nr. 5/5A	21	TN-Nr. 21/21A	37	TN-Nr. 5/5B	53	TN-Nr. 21/21B
6	TN-Nr. 6/6A	22	TN-Nr. 22/22A	38	TN-Nr. 6/6B	54	TN-Nr. 22/22B
7	TN-Nr. 7/7A	23	TN-Nr. 23/23A	39	TN-Nr. 7/7B	55	TN-Nr. 23/23B
8	TN-Nr. 8/8A	24	TN-Nr. 24/24A	40	TN-Nr. 8/8B	56	TN-Nr. 24/24B
9	TN-Nr. 9/9A	25	TN-Nr. 25/25A	41	TN-Nr. 9/9B	57	TN-Nr. 25/25B
10	TN-Nr. 10/10A	26	TN-Nr. 26/26A	42	TN-Nr. 10/10B	58	TN-Nr. 26/26B
11	TN-Nr. 11/11A	27	TN-Nr. 27/27A	43	TN-Nr. 11/11B	59	TN-Nr. 27/27B
12	TN-Nr. 12/12A	28	TN-Nr. 28/28A	44	TN-Nr. 12/12B	60	TN-Nr. 28/28B
13	TN-Nr. 13/13A	29	TN-Nr. 29/29A	45	TN-Nr. 13/13B	61	TN-Nr. 29/29B
14	TN-Nr. 14/14A	30	TN-Nr. 30/30A	46	TN-Nr. 14/14B	62	TN-Nr. 30/30B
15	TN-Nr. 15/15A	31	TN-Nr. 31/31A	47	TN-Nr. 15/15B	63	TN-Nr. 31/31B

Tab. 4-7. LAS, LDS, LOS und LPF

Der ASI Teilnehmer ist aktiviert, wenn das Bit gesetzt ist.

4.1.20 ASI Read LDS

ASI Read LDS liest die erkannten ASI Teilnehmer.

```
int (*AASiReadLDS) (unsigned char Circuit, AASiSlaveList *LDS);
```

Parameter:

Circuit: ASI Master Kreis

Return:

LDS: 8 Byte ASI Teilnehmer-Liste

Jedes Bit der LDS entspricht einem ASI Teilnehmer (siehe Tab. <LAS, LDS, LOS und LPF>).

4.1.21 ASi Read LCS

ASi Read LCS liest alle fehlerhaften ASi Teilnehmer.

```
int (*AASiReadLCS) (unsigned char Circuit, AASiSlaveList *LCS);
```

Parameter:

Circuit: ASi Master Kreis

Return:

LCS: 8 Byte ASi Teilnehmer-Liste

Jedes Bit der LCS entspricht einem ASi Teilnehmer (siehe Tab. <LAS, LDS, LOS und LPF>).



Hinweis!

Die Liste wird nach dem Lesen zurückgesetzt!

4.1.22 ASi Write LOS

ASi Write LOS schreibt die Liste der ASi Teilnehmer, welche bei einem Konfigurationsfehler offline gehen sollen.

```
int (*AASiWriteLOS) (unsigned char Circuit, AASiSlaveList LOS);
```

Parameter:

Circuit: ASi Master Kreis

Return:

LOS: 8 Byte ASi Teilnehmer-Adressen-Liste

Jedes Bit der LOS entspricht einem ASi Teilnehmer (siehe Tab. <LAS, LDS, LOS und LPF>).

4.1.23 ASi Read LOS

ASi Read LOS liest die Liste der ASi Teilnehmer, welche bei einem Konfigurationsfehler offline gehen.

```
int (*AASiReadLOS) (unsigned char Circuit, AASiSlaveList *LOS);
```

Parameter:

Circuit: ASi Master Kreis

Return:

LOS: 8 Byte ASi Teilnehmer-Adressen-Liste

Jedes Bit der LOS entspricht einem ASi Teilnehmer (siehe Tab. <LAS, LDS, LOS und LPF>).

4.1.24 ASi Read LPF

ASi Read LPF liest die Liste der ASi Teilnehmer mit Peripheriefehler.

```
int (*AASiReadLPF) (unsigned char Circuit, AASiSlaveList *LPF);
```

Parameter:

Circuit: ASi Master Kreis

Return:

LPF: 8 Byte ASi Teilnehmer-Adressen-Liste

Jedes Bit der LPS entspricht einem ASi Teilnehmer (siehe Tab. <LAS, LDS, LOS und LPF>).

4.1.25 ASi Read Ec Flags

ASi Read Ec Flags liest die execution control flags.

```
int (*AASiReadEcFlags) (unsigned char Circuit, AASiEcFlags
*EcFlags);
```

Parameter:

Circuit: ASi Master Kreis

Return:

EcFlags: Zwei Byte EcFlags.

4.1.26 ASi set Config Mode

ASi set Config Mode setzt den ASi Master in den Konfigurationsmodus oder in den geschützten Betriebsmodus.

```
int (*AASiSetConfigMode) (unsigned char Circuit, unsigned char
Mode);
```

Parameter:

Circuit: ASi Master Kreis

Mode: 1 = Konfigurationsmodus

0 = Geschützter Betriebsmodus

Return: —

4.1.27 ASi Write Hi Flags

ASi Write Hi Flags schreibt Host-Interface-Flags des ASi Masters.

```
int (*AASiWriteHiFlags) (unsigned char Circuit, AASiHiFlags HiF-
lags);
```

Parameter:

Circuit: ASi Master Kreis

HiFlags: Host-Interface-Flags (ein Byte)

Return: —

4.1.28 ASi Read Hi-Flags

ASi Read Hi-Flags liest die Host-Interface-Flags des ASi Masters.

```
int (*AASiReadHiFlags) (unsigned char Circuit, AASiHiFlags *HiF-
lags);
```

Parameter:

Circuit: ASi Master Kreis

Return:

HiFlags: Host-Interface-Flags (ein Byte)

4.1.29 ASi Address Slave

ASi Address Slave ändert die Adresse eines ASi Teilnehmers.

```
int (*AASiAddressSlave) (unsigned char Circuit, AASiSlaveAddr
OldAddress, AASiSlaveAddr NewAddress);
```

Parameter:

Circuit:	ASi Master Kreis
OldAddress:	Alte ASi Teilnehmer-Adresse
NewAddress:	Neue ASi Teilnehmer-Adresse

Return:

Spezielle Fehlercodes, welche den Grund der fehlerhaften Übertragung beschreiben.

4.1.30 ASi Execute Command

ASi Execute Command ist ein direkt zu sendendes ASi Kommando.

```
int (*AASiExecuteCommand) (unsigned char Circuit, AASiSlaveAddr
Address, AASiSlaveData Request, AASiSlaveData *Response);
```

Parameter:

Circuit:	ASi Master Kreis
Address:	Adresse des ASi Teilnehmers
Request:	ASi Request Befehl

Return:

Response: ASi Request

4.1.31 ASi Read All Config

ASi Read All Config liest alle Konfigurationsdaten (z.B. LPS, PP[] und PCD []) aller angeschlossenen ASi Teilnehmers.

```
int (*AASiReadAllConfig) (unsigned char Circuit, AASiConfig *Con-
figurations );
```

Parameter:

Circuit:	ASi Master Kreis
Configurations:	Array zur Aufnahme der Konfigurationsdaten

Return:

Die Konfigurationsdaten in Configurations []

4.1.32 ASi Write All Config

ASi Write All Config schreibt alle Konfigurationsdaten (z.B. LPS, PP[] und PCD []) aller angeschlossenen ASi Teilnehmers.

```
int (*AASiWriteAllConfig) (unsigned char Circuit, AASiConfig Con-
figurations);
```

Parameter:

Circuit:	ASi Master Kreis
Configurations:	Array zur Aufnahme der Konfigurationsdaten

Return: —

4.1.33 ASi read Error Counter

ASi read Error Counter liest den ASi Teilnehmer-Error-Counter.

```
int (*AASiReadErrorCounters) (unsigned char Circuit, AASiError-
Counters Counters);
```

Parameter:

Circuit: ASi Master Kreis

Return:

Counters: 64 Byte (ein Byte pro ASi Teilnehmer)



Hinweis!

Die Liste wird nach dem Lesen zurückgesetzt!

4.1.34 ASi Mail Box

Generic Mailbox-Funktion

```
int (*AASiMailbox) (unsigned char Circuit, AASiMbRequestType
Request, int ExpResLen, AASiMbResponseType *Response);
```

Parameter:

Circuit: ASi Master Kreis

Request: Struktur für Mailbox Anfrage

ExpResLen: Erwartete Länge der Antwort (-1 = unbekannt)

Return:

Response: Struktur für Mailbox Antwort

4.1.35 ASi Write 16Bit ODI

ASi Write 16Bit ODI schreibt vier 16bit ODI Kanäle eines ASi Teilnehmers mit z.B. Analog-Teilnehmer-Profil 7.3 oder 7.4.

```
int (*AASiWrite16BitODI) (unsigned char Circuit, AASiSlaveAddr
Address, AASi16BitData Out);
```

Parameter:

Circuit: ASi Master Kreis

Address: Adresse des ASi Teilnehmers

ExpResLen: Erwartete Länge der Antwort (-1 = unbekannt)

Out: 4 Kanäle mit 16 Bit Werten

Word 0 : Kanal 1

Word 1 : Kanal 2

Word 2 : Kanal 3

Word 3 : Kanal 4

Return:—

4.1.36 ASi Read 16Bit ODI

ASi Read 16Bit ODI liest vier 16bit ODI Kanäle eines ASi Teilnehmers mit z.B. Analog-Teilnehmer-Profil 7.3 oder 7.4.

```
int (*AASiRead16BitODI) (unsigned char Circuit, AASiSlaveAddr
Address, AASi16BitData In);
```

Parameter:

Circuit:	ASi Master Kreis
Address:	Adresse des ASi Teilnehmers
In:	4 Kanäle mit 16 Bit Werten
	Word 0 : Kanal 1
	Word 1 : Kanal 2
	Word 2 : Kanal 3
	Word 3 : Kanal 4

4.1.37 ASi Read 16Bit IDI

ASi Read 16Bit IDI liest vier 16bit IDI Kanäle eines ASi Teilnehmers mit z.B. Analog-Teilnehmer-Profil 7.3 oder 7.4.

```
int (*AASiRead16BitIDI) (unsigned char Circuit, unsigned char
Address, AASi16BitData In);
```

Parameter:

Circuit:	ASi Master Kreis
Address:	Adresse des ASi Teilnehmers
In:	4 Kanäle mit 16 Bit Werten
	Word 0 : Kanal 1
	Word 1 : Kanal 2
	Word 2 : Kanal 3
	Word 3 : Kanal 4

4.1.38 ASi Read Ctrl Acc ODI

ASi Read Ctrl Acc ODI liest die Control III Berechtigungen, um Ausgangsdaten von ASi Teilnehmer zu schreiben.

```
int (*AASiReadCtrlAccODI) ( unsigned char Circuit, AASiCtrlAcc-
cODI ODI );
```

Parameter:

Circuit:	ASi Master Kreis
----------	------------------

Return:

ODI:	32 Byte Ctrl Zugangsdaten
------	---------------------------

4.1.39 ASi Write Ctrl Acc ODI

ASi Write Ctrl Acc ODI schreibt die Control III Berechtigungen der ASi Teilnehmer, um Ausgangsdaten zu verändern.

```
int (*AASiWriteCtrlAccODI) ( unsigned char Circuit, AASiCtrlAcc-
cODI ODI, AASiSlaveAddr First, unsigned char Amount );
```

Parameter:

Circuit:	ASi Master Kreis
ODI:	32 Byte Ctrl Zugangsdaten
First:	Index des ersten ASi Teilnehmers
Amount:	Anzahl der folgenden ASi Teilnehmer nach First

Return:—

4.1.40 ASi Read Ctrl Acc AODI

ASi Read Ctrl Acc AODI liest die Control III Berechtigungen, um analoge Ausgangsdaten von ASi Teilnehmern zu schreiben.

```
int (*AASiReadCtrlAccAODI) ( unsigned char Circuit, AASiCtrlAccAODI AODI );
```

Parameter:

Circuit: ASi Master Kreis

Return:

AODI: 16 Byte Ausgangsdaten Ctrl Zugang

Bit	7	6	5	4	3	2	1	0	Bit	7	6	5	4	3	2	1	0
Kanal	3	2	1	0	3	2	1	0	Kanal	3	2	1	0	3	2	1	0
Byte	D3	D2	D1	D0	D3	D2	D1	D0	Byte	D3	D2	D1	D0	D3	D2	D1	D0
0	TN-Nr. 1/1A				TN-Nr. 0/0A				1	TN-Nr. 3/3A				TN-Nr. 2/2A			
2	TN-Nr. 5/5A				TN-Nr. 4/4A				3	TN-Nr. 7/7A				TN-Nr. 6/6A			
4	TN-Nr. 9/9A				TN-Nr. 8/8A				5	TN-Nr. 11/11A				TN-Nr. 10/10A			
6	TN-Nr. 13/13A				TN-Nr. 12/12A				7	TN-Nr. 15/15A				TN-Nr. 14/14A			
8	TN-Nr. 17/17A				TN-Nr. 16/16A				9	TN-Nr. 19/19A				TN-Nr. 18/18A			
10	TN-Nr. 21/21A				TN-Nr. 20/20A				11	TN-Nr. 23/23A				TN-Nr. 22/22A			
12	TN-Nr. 25/25A				TN-Nr. 24/24A				13	TN-Nr. 27/27A				TN-Nr. 26/26A			
14	TN-Nr. 29/29A				TN-Nr. 28/28A				15	TN-Nr. 31/31A				TN-Nr. 30/30A			
16	TN-Nr. 1/1B				TN-Nr. 0/0B				17	TN-Nr. 3/3B				TN-Nr. 2/2B			
18	TN-Nr. 5/5B				TN-Nr. 4/4B				19	TN-Nr. 7/7B				TN-Nr. 6/6B			
20	TN-Nr. 9/9B				TN-Nr. 8/8B				21	TN-Nr. 11/11B				TN-Nr. 10/10B			
22	TN-Nr. 13/13B				TN-Nr. 12/12B				23	TN-Nr. 15/15B				TN-Nr. 14/14B			
24	TN-Nr. 17/17B				TN-Nr. 16/16B				25	TN-Nr. 19/19B				TN-Nr. 18/18B			
26	TN-Nr. 21/21B				TN-Nr. 20/20B				27	TN-Nr. 23/23B				TN-Nr. 22/22B			
28	TN-Nr. 25/25B				TN-Nr. 24/24B				29	TN-Nr. 27/27B				TN-Nr. 26/26B			
30	TN-Nr. 29/29B				TN-Nr. 28/28B				31	TN-Nr. 31/31B				TN-Nr. 30/30B			

Tab. 4-8. Ausgangsdatenabbild AODI

4.1.41 ASi Write Ctrl Acc AODI

ASi Write Ctrl Acc AODI schreibt die Control III Berechtigungen der ASi Teilnehmer, um analoge Ausgangsdaten zu verändern.

```
int (*AASiWriteCtrlAccAODI) ( unsigned char Circuit, AASiCtrlAccAODI AODI, AASiSlaveAddr First, unsigned char Amount;
```

Parameter:

Circuit: ASi Master Kreis

AODI: 16 Byte Ctrl Zugangsdaten (siehe Tab. <Ausgangsdatenabbild AODI>)

First: Index des ersten ASi Teilnehmers

Amount: Anzahl der folgenden ASi Teilnehmer nach First

Return:—

4.1.42 AASi Read PE

Diese Funktion liest Echo-Parameter eines ASi Teilnehmers.

```
int (*AASiReadPE) (unsigned char Circuit, AASiSlaveAddr Address,
AASiSlaveData *PE);
```

Parameter:

Circuit: ASi Master Kreis

Address: ASi Teilnehmer-Adressen mit den Echo-Parametern
(1...31, 33...63).

Return:

PE: Echo-Parameter des ASi Teilnehmers (low nibble)

4.1.43 AASi Set CPRL

Diese Funktion schreibt Liste der zyklischen Parameterabfragen.

```
int (*AASiSetCPRL) (unsigned char Circuit, AASiSlaveList CPRL);
```

Parameter:

Circuit: ASi Master Kreis

LOS: 8 Bytes Liste der ASi Teilnehmer-Adressen.

Jedes Bit in der SPL entspricht einem ASi Teilnehmer
nach dem folgenden Schema:

Bit 0: TN-Nr. 0/0A

Bit 1: TN-Nr. 1/1A

...

Bit31: TN-Nr. 31/31A

Bit32: TN-Nr. 0B

Bit33: TN-Nr. 1B

...

Bit63: TN-Nr. 31B

Die zyklischen Parameter-Anfragen sind aktiv,
wenn das Bit gesetzt ist

Return: —

4.1.44 AASi Send Parameter Non Blocking

Diese Funktion sendet Parameter an einen ASi Teilnehmer ohne auf das ASi Teilnehmer-Echo zu warten.

```
int (*AASiSendParameterNonBlocking) (unsigned char Circuit,
AASiSlaveAddr Address, AASiSlaveData PI);
```

Parameter:

Circuit: ASi Master Kreis

Address: ASi Teilnehmer-Adressen mit Sende-Parametern
(1...31, 33...63).

PI: zu sendende Parameter (low nibble)

Return:

! 0 Fehler

0 kein Fehler

4.1.45 Get Systeme

Diese Funktion gibt die aktuelle System-Zeit zurück (1ms).

Achtung: Dies ist ein 32-Bit-Wert, der nach ~ 49 Tagen abläuft!

Parameter: —

Return:

systeme

4.2 Übersicht der ASI-5 Befehle

Asi5ReadCtrlAccODI	ASI-5 Ctrl-Ausgangsdatenabbild lesen
Asi5WriteCtrlAccODI	ASI-5 Ctrl-Ausgangsdatenabbild schreiben
Asi5ClearCtrlAccODI	ASI-5 Ctrl-Ausgangsdatenabbild löschen
Asi5ReadEcFlags	ASI-5 Execution Control Flags lesen
Asi5OdcRequest	ASI-5 ODC Anfrage über logische Adresse senden
Asi5OdcRequestByAsiid	ASI-5 ODC Anfrage per asiid senden
Asi5SetOperatingMode	ASI-5 Betriebsmodus einstellen
Asi5GetOperatingMode	ASI-5 Betriebsmodus abfragen
Asi5SetDataExchange-Mode	ASI-5 Datenaustauschmodus einstellen
Asi5GetDataExchange-Mode	ASI-5 Datenaustauschmodus abfragen
Asi5SetOfflineMode	ASI-5 Datenaustauschmodus für zyklischen Datenaustausch im Normalbetrieb einstellen
Asi5GetOfflineMode	Status des ASI-5 Offlinemodus abfragen
Asi5SetAutoAddressEnable	ASI-5 Auto-Adressierungsmodus aktiv oder inaktiv setzen
Asi5GetAutoAddressAssignMode	Status der automatischen Adressierung in ASI-5 abfragen
Asi5SetAsi3Coexisting-Mode	ASI-5 Master in ASI-3 CoExisting Mode versetzen
Asi5GetAsi3Coexisting-Mode	Status des ASI-3 CoExisting Mode des ASI-5 Masters abfragen
Asi5ChangeSlaveAddress	logische Adresse auf neue logische Adresse eines ASI-5 Teilnehmers ändern
Asi5ChangeSlaveAddressByAsiid	logische Adresse auf neue logische Adresse eines anhand seiner ASIID identifizierten ASI-5 Teilnehmers ändern
Asi5InLds	prüfen, ob eine logische ASI-5 Teilnehmeradresse in ASI5LDS vorhanden ist
Asi5InLps	prüfen, ob eine logische ASI-5 Teilnehmeradresse in ASI5LPS vorhanden ist
Asi5InLpf	Statuscode eines ASI-5 Teilnehmers abfragen
Asi5InLms	prüfen, ob eine logische ASI-5 Teilnehmeradresse in ASI5LMS vorhanden ist
Asi5InLas	prüfen, ob eine logische ASI-5 Teilnehmeradresse in ASI5LAS vorhanden ist
Asi5InLis	prüfen, ob eine logische ASI-5 Teilnehmeradresse in ASI5LIS vorhanden ist
Asi5InLda	prüfen, ob eine logische ASI-5 Teilnehmeradresse in ASI5LDA vorhanden ist
Asi5InLdd	prüfen, ob eine logische ASI-5 Teilnehmeradresse in ASI5LDD vorhanden ist
Asi5SetCdi	Konfigurationsdatenabbild für einen ASI-5 Teilnehmer ändern
Asi5GetCdi	Konfigurationsdaten für eine Adresse aus dem ASI-5 Konfigurationsdatenabbild abrufen
Asi5SetPcd	Konfigurationswert für eine logische Adresse in permanente Konfigurationsdaten speichern

Tab. 4-9. Übersicht der ASI-5 Befehle

Ausgabedatum: 16.05.2024

Asi5GetPcd	Konfigurationsdaten für eine logische Adresse aus permanenten ASI-5 Konfigurationsdaten abrufen
Asi5StoreCdi	Werte des ASI-5 Konfigurationsdatenabbildes in die permanenten ASI-5 Konfigurationsdaten für einen ASI-5 Teilnehmer kopieren
Asi5StoreSlaveCdi	Werte des ASI-5 Konfigurationsdatenabbildes in die permanenten ASI-5 Konfigurationsdaten für einen ASI-5 Teilnehmer kopieren
Asi5ResetPcd	permanente ASI-5 Konfigurationsdaten für eine logische Adresse auf Standardwert setzen
Asi5ResetAllPcd	permanente ASI-5 Konfigurationsdaten für eine logische Adresse auf Standardwert setzen
Asi5GetLas	komplette Teilnehmerliste ASI5LAS abrufen
Asi5GetLps	komplette Teilnehmerliste ASI5LPS abrufen
Asi5GetLds	komplette Teilnehmerliste ASI5LDS abrufen
Asi5GetLms	komplette Teilnehmerliste ASI5LMS abrufen
Asi5GetLpf	komplette Teilnehmerliste ASI5LPF abrufen
Asi5GetLis	komplette Teilnehmerliste ASI5LIS abrufen
Asi5GetLda	komplette Teilnehmerliste ASI5LDA abrufen
Asi5GetLdd	komplette Teilnehmerliste ASI5LDD abrufen
Asi5GetSlaveAsiid	Liste von ASI-5 Teilnehmern abrufen, die eine bestimmte logische Adresse nutzen
Asi5GetSlaveInfo	komplette Informationsstruktur der ASI-5 Teilnehmer erhalten
Asi5GetSlaveInfoAsiid	komplette Informationsstruktur der durch die ASIID identifizierten ASI-5 Teilnehmer erhalten
Asi5GetSystemResources	Überblick über die genutzten Ressourcen des ASI-5 Systems erhalten
Asi5IdentifySlaveByLogicalAddress	ASI-5 Teilnehmer zur optischen Identifikation über logische Adresse zwingen
Asi5IdentifySlaveByAsiid	ASI-5 Teilnehmer zur optischen Identifikation über ASIID zwingen
Asi5SetDeviceDesignatorByLogicalAddress	Gerätebezeichner des ASI-5 Teilnehmers über logische Adresse speichern
Asi5SetDeviceDesignatorByAsiid	Gerätebezeichner des ASI-5 Teilnehmers über ASIID speichern
Asi5GetDeviceDesignatorByLogicalAddress	Gerätebezeichner des ASI-5 Teilnehmers über logische Adresse abrufen
Asi5GetDeviceDesignatorByAsiid	Gerätebezeichner des ASI-5 Teilnehmers über ASIID abrufen
Asi5AutoSetLogicalAddresses	logische Adresse aller ASI-5 Teilnehmer automatisch mit logischer Adresse 0 einstellen
Asi5SetEnergySavingState	Energiesparzustand aller ASI-5 Teilnehmer der ausgewählten Energiespargruppe ändern
Asi5GetEnergySavingState	Energiesparstatus der ausgewählten Energiespargruppe abrufen
Asi5WriteAsi5Odi	Ausgangsdatenabbild von ASI-5 Teilnehmern erstellen
Asi5ReadAsi5Odi	Ausgangsdatenabbild von ASI-5 Teilnehmern abrufen
Asi5ReadAsi5Idi	Eingangsdatenabbild von ASI-5 Teilnehmern abrufen
Asi5DiagGetMasterStatus	Diagnosestatus des ASI-5 Masters abrufen

Tab. 4-9. Übersicht der ASI-5 Befehle

Asi5GetParameters	Parameter aus dem Parameterabbild eines ASI-5 Teilnehmers abrufen
Asi5SetParameters	Parameter in das Parameterabbild des ASI-5 Teilnehmers übertragen
Asi5SetPsMode	aktuelle Betriebsart des Parameterservers einstellen
Asi5GetPsMode	aktuelle Betriebsart des Parameterservers zu ermitteln
Asi5GetStoredPi	gespeicherte PI für einen ASI-5 Teilnehmer aus dem ASI-Master auslesen
Asi5SetStoredPi	PI für ASI-5 Teilnehmer in den Parameterabbildsatz der aktiven Konfiguration im ASI Master speichern
Asi5StorePi	neue Version des Parameterabbildes vom ASI-5 Teilnehmer in den ASI-5 Master laden und speichern
Asi5RestorePi	gespeicherte PI in den ASI-5 Teilnehmer laden
Asi5StoreAllSlavePi	aktuell verwendete Parameterabbilder aller ASI-5 Teilnehmer in den ASI-5 Master speichern
Asi5RestoreAllSlavePi	Parameterabbild vom ASI-5 Master in alle ASI-5 Teilnehmer herunterladen
Asi5SlavnumberToLogAddr	Teilnehmernummer in eine logische Adresse konvertieren
Asi5LogAddrToSlavnumber	logische Adresse in eine Teilnehmernummer umwandeln
Asi5DeltaList	Liste der Konfigurationsfehler zurückgeben
Asi5InDeltaList	prüfen, ob logische Adresse in der Deltaliste enthalten ist
Asi5GetIoLength	Länge der E/A-Daten des Teilnehmers zurückgeben
Asi5GetIoLengthByAsiid	Länge der E/A-Daten des Teilnehmers anhand der ASI-ID zurückgeben
Asi5ResetAllSlaves	alle ASI-5 Teilnehmer zurücksetzen

Tab. 4-9. Übersicht der ASI-5 Befehle

4.2.1 Asi5ReadCtrlAccODI

Mit dieser Funktion werden die aktuellen Ctrl-Zugangsdaten für das Asi-5 Ctrl-Ausgangsdatenabbild zurückgelesen.

```
return_t (*Asi5ReadCtrlAccODI) (const unsigned char circuit,
    unsigned char *mask, unsigned short *length, const unsigned short
    offset).
```

Parameter:

circuit:	ASi Master Kreis
mask:	Bit-Maske der Acc
length:	Länge der zu lesenden Maske
offset :	Offset der zu lesenden Maske

Return:

0	wenn OK
!0	wenn nicht OK

4.2.2 Asi5WriteCtrlAccODI

Mit dieser Funktion werden die aktuellen Ctrl-Zugangsdaten für das Asi-5 Ausgangsdatenabbild geschrieben.

```
return_t (*Asi5WriteCtrlAccODI) (const unsigned char circuit,
    const unsigned char *mask, const unsigned short length, const
    unsigned short offset).
```

Parameter:

circuit:	ASi Master Kreis
mask:	Bit-Maske der Acc
length:	Länge der zu schreibenden Maske
offset :	Offset der zu schreibenden Maske

Return:

0	wenn OK
!0	wenn nicht OK

4.2.3 Asi5ClearCtrlAccODI

Mit dieser Funktion werden die aktuellen Ctrl-Zugangsdaten für das Asi-5 Ausgangsdatenabbild gelöscht.

```
return_t (*Asi5ClearCtrlAccODI) (const unsigned char circuit,
    const unsigned short length, const unsigned short
    offset).
```

Parameter:

circuit:	ASi Master Kreis
length:	Länge der zu löschenden Maske
offset:	Offset der zu löschenden Maske

Return:

0	wenn OK
!0	wenn nicht OK

4.2.4 Asi5ReadEcFlags

Mit dieser Funktion werden Asi-5 Execution Control Flags gelesen.

```
return_t (*Asi5ReadEcFlags) (const unsigned char circuit, unsigned short *ecflags).
```

Parameter:

ecflags: die Execution Control Flags

Return:

0 wenn OK
!0 wenn nicht OK

4.2.5 Asi5OdcRequest

Mit dieser Funktion wird die Asi-5 ODC Anfrage an einen Teilnehmer anhand seiner logischen Adresse gesendet.

```
return_t (*Asi5OdcRequest) (const unsigned char circuit, odc_data_t *respData, const unsigned short logAddr, const odc_data_t *reqData).
```

Parameter:

circuit: Asi Master Kreis

respData: Antwort-Daten

logAddr: logische Adresse des Asi-5 Teilnehmers

Return:

0 wenn OK
!0 wenn nicht OK

4.2.6 Asi5OdcRequestByAsiid

Mit dieser Funktion wird die Asi-5 ODC-Anfrage per ASIID gesendet.

```
return_t (*Asi5OdcRequestByAsiid) (const unsigned char circuit, odc_data_t *respData, const asiid_t asiid, const odc_data_t *reqData).
```

Parameter:

circuit: Asi Master Kreis

respData: Antwort-Daten

ASIID: ASIID des Asi-5 Teilnehmers

reqData: Anfrage-Daten

Return:

0 wenn OK
!0 wenn nicht OK

4.2.7 Asi5SetOperatingMode

Mit dieser Funktion wird der Asi-5 Betriebsmodus eingestellt.

```
return_t (*Asi5SetOperatingMode) (const unsigned char circuit, const unsigned char asi_5_cfg_instance, const unsigned char asi_5_opmode).
```

Parameter:

circuit: Asi Master Kreis

asi_5_cfg_instance: enthält die abgerufene Konfigurationsinstanz

asi_5_opmode: abgerufener Asi-5 Betriebsmodus

Return:

0 wenn OK
!0 wenn nicht OK

4.2.8 Asi5GetOperatingMode (circuit, ...)

Mit dieser Funktion wird der ASI-5 Betriebsmodus abgefragt.

```
return_t (*Asi5GetOperatingMode) (const unsigned char circuit,
    unsigned char *asi_5_cfg_instance, unsigned char *asi_5_opmode).
```

Parameter:

circuit:	ASi Master Kreis
asi_5_cfg_instance:	enthält die abgerufene Konfigurationsinstanz
asi_5_opmode:	zurückgemeldeter ASI-5 Betriebsmodus

Return:

0	wenn OK
!0	wenn nicht OK

4.2.9 Asi5SetDataExchangeMode

Mit dieser Funktion wird der ASI-5 Datenaustauschmodus eingestellt.

```
return_t (*Asi5SetDataExchangeMode) (const unsigned char circuit,
    const unsigned char asi5_demode).
```

Parameter:

circuit:	ASi Master Kreis
asi5_demode:	abgerufener ASI-5 Datenaustauschmodus

Return:

0	wenn OK
!0	wenn nicht OK

4.2.10 Asi5GetDataExchangeMode

Mit dieser Funktion wird der ASI-5 Datenaustauschmodus abgefragt.

```
return_t (*Asi5GetDataExchangeMode) (const unsigned char circuit,
    unsigned char *asi5_demode).
```

Parameter:

circuit:	ASi Master Kreis
asi5_demode:	zurückgemeldeter ASI-5 Datenaustauschmodus

Return:

0	wenn OK
!0	wenn nicht OK

4.2.11 Asi5SetOfflineMode

Mit dieser Funktion wird der ASI-5 Datenaustauschmodus eingestellt, welcher den zyklischen Datenaustauschmodus im Normalbetrieb startet bzw. stoppt.

```
return_t (*Asi5SetOfflineMode) (const unsigned char circuit,
    const unsigned char asi5_offmode).
```

Parameter:

circuit:	ASi Master Kreis
asi5_offmode:	abgerufener ASI-5 Offlinemodus

Return:

0	wenn OK
!0	wenn nicht OK

4.2.12 Asi5GetOfflineMode

Mit dieser Funktion wird der Status des ASI-5 Offlinemodus abgefragt.

```
return_t (*Asi5GetOfflineMode) (const unsigned char circuit,
unsigned char *asi5_offmode).
```

Parameter:

circuit:	ASi Master Kreis
asi5_offmode:	zurückgemeldeter ASI-5 Offlinemodus

Return:

0	wenn OK
!0	wenn nicht OK

4.2.13 Asi5SetAutoAddressEnable

Mit dieser Funktion wird der ASI-5 Auto-Adressierungsmodus aktiv oder inaktiv gesetzt. Dieses Bit wird nichtflüchtig gespeichert.

```
return_t (*Asi5SetAutoAddressEnable) (const unsigned char circuit,
const unsigned char asi5_aaamode).
```

Parameter:

circuit:	ASi Master Kreis
asi5_aaamode:	abgerufener ASI-5 Auto-Adressierungsmodus

Return:

0	wenn OK
!0	wenn nicht OK

4.2.14 Asi5GetAutoAddressAssignMode

Mit dieser Funktion wird geprüft, ob der ASI-5 Master sich im Autoadressiermodus befindet.

```
return_t (*Asi5GetAutoAddressAssignMode) (const unsigned char circuit,
unsigned char *asi5_aaamode).
```

Parameter:

circuit:	ASi Master Kreis
asi5_aaamode:	zurückgemeldeter ASI-5 Auto-Adressierungsmodus

Return:

0	wenn OK
!0	wenn nicht OK

4.2.15 Asi5SetASI3CoexistingMode

Mit dieser Funktion wird der ASI-5 Master in ASI-3 CoExisting Mode versetzt.

Dieser Merker muss nichtflüchtig gespeichert werden. Der Modus wird nach der nächsten Offline-Phase geändert.

```
return_t (*Asi5SetASI3CoexistingMode) (const unsigned char circuit,
const unsigned char asi3_coexisting).
```

Parameter:

circuit:	ASi Master Kreis
asi3_coexisting:	abgerufener ASI-3 CoExisting Mode

Return:

0	wenn OK
!0	wenn nicht OK

4.2.16 Asi5GetASi3CoexistingMode

Mit dieser Funktion wird der Status des ASi-3 CoExisting Mode des ASi-5 Masters abgefragt.

Dieser Merker stellt den nichtflüchtig gespeicherten Wert dar. Der tatsächliche Modus kann anders sein.

```
return_t (*Asi5GetASi3CoexistingMode) (const unsigned char circuit, unsigned char *asi3_coexisting).
```

Parameter:

circuit: ASi Master Kreis

asi3_coexisting: zurückgemeldeter ASi-3 CoExisting Mode

Return:

0 wenn OK

!0 wenn nicht OK

4.2.17 Asi5ChangeSlaveAddress

Mit dieser Funktion wird die Ablaufkontrolle aufgefordert, die logische Adresse auf eine neue logische Adresse eines entsprechenden ASi-5 Teilnehmers zu ändern.

```
return_t (*Asi5ChangeSlaveAddress) (const unsigned char circuit, const unsigned short newLogAddr, const unsigned short oldLogAddr).
```

Parameter:

circuit: ASi Master Kreis

newLogAddr: neue logische Adresse

oldLogAddr: alte logische Adresse

Return:

0 wenn OK

!0 wenn nicht OK

4.2.18 Asi5ChangeSlaveAddressByAsiid

Mit dieser Funktion wird die ASi-5 Teilnehmernummer geändert. Das ASi-5 Modul wird anhand seiner ASIID identifiziert.

```
return_t (*Asi5ChangeSlaveAddressByAsiid) (const unsigned char circuit, const unsigned short newLogAddr, const asiid_t ASIID).
```

Parameter:

circuit: ASi Master Kreis

newLogAddr: neue logische Adresse

ASIID: ASIID des Teilnehmers

Return:

0 wenn OK

!0 wenn nicht OK

4.2.19 Asi5InLds

Mit dieser Funktion wird geprüft, ob eine bestimmte logische ASI-5 Teilnehmeradresse im ASI5LDS vorhanden ist.

```
return_t (*Asi5InLds) (const unsigned char circuit, unsigned char
                        *in_lds, const unsigned short LogAddr).
```

Parameter:

circuit:	ASI Master Kreis
in_lds:	0: ASI5Saddr nicht in ASI5LDS enthalten !0: ASI5Saddr in ASI5LDS enthalten
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.20 Asi5InLps

Mit dieser Funktion wird geprüft, ob eine bestimmte logische ASI-5 Teilnehmeradresse in der ASI5LPS vorhanden ist.

```
return_t (*Asi5InLps) (const unsigned char circuit, unsigned char
                        *in_lps, const unsigned short LogAddr).
```

Parameter:

circuit:	ASI Master Kreis
in_lps:	0: ASI5Saddr nicht in ASI5LPS enthalten !0: ASI5Saddr in ASI5LPS enthalten
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.21 Asi5InLpf

Mit dieser Funktion wird der Statuscode eines entsprechenden ASI-5 Teilnehmers abgefragt.

```
return_t (*Asi5InLpf) (const unsigned char circuit, asi_5_lpf_t
                        *in_lpf, const unsigned short LogAddr).
```

Parameter:

circuit:	ASI Master Kreis
in_lpf:	in_lpf Statuscode des ASI-5 Teilnehmers
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.22 Asi5InLms

Mit dieser Funktion wird geprüft, ob eine bestimmte logische ASI-5 Teilnehmeradresse in der ASI5LMS vorhanden ist.

```
return_t (*Asi5InLms) (const unsigned char circuit, unsigned char
                        *in_lms, const unsigned short LogAddr).
```

Parameter:

circuit:	ASi Master Kreis
in_lms:	0: ASI5Saddr nicht in Asi5InLms enthalten !0: ASI5Saddr in Asi5InLms enthalten
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.23 Asi5InLas

Mit dieser Funktion wird geprüft, ob eine bestimmte logische ASI-5 Teilnehmeradresse in der ASI5LAS vorhanden ist.

```
return_t (*Asi5InLas) (const unsigned char circuit, unsigned char
                        *in_las, const unsigned short LogAddr).
```

Parameter:

circuit:	ASi Master Kreis
in_las:	0: ASI5Saddr nicht in ASI5LAS enthalten !0: ASI5Saddr in ASI5LAS enthalten
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.24 Asi5InLis

Mit dieser Funktion wird geprüft, ob eine bestimmte logische ASI-5 Teilnehmeradresse in der ASI5LIS vorhanden ist.

```
return_t (*Asi5InLis) (const unsigned char circuit, unsigned char
                        *in_lis, const unsigned short LogAddr).
```

Parameter:

circuit:	ASi Master Kreis
in_lis:	0: ASI5Saddr nicht in ASI5LIS enthalten !0: ASI5Saddr in ASI5LIS enthalten
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.25 Asi5InLda

Mit dieser Funktion wird geprüft, ob eine bestimmte logische ASI-5 Teilnehmeradresse in der ASI5LDA vorhanden ist.

```
return_t (*Asi5InLda) (const unsigned char circuit, unsigned char
                        *in_lda, const unsigned short LogAddr).
```

Parameter:

circuit:	ASI Master Kreis
in_lda:	0: ASI5Saddr nicht in ASI5LDA enthalten !0: ASI5Saddr in ASI5LDA enthalten
LogAddr:	logische Adresse

Return:

	0 wenn OK !0 wenn nicht OK
--	-------------------------------

4.2.26 Asi5InLdd

Mit dieser Funktion wird geprüft, ob eine bestimmte logische ASI-5 Teilnehmeradresse in der ASI5LDD vorhanden ist.

```
return_t (*Asi5InLdd) (const unsigned char circuit, unsigned char
                        *in_ldd, const unsigned short LogAddr).
```

Parameter:

circuit:	ASI Master Kreis
in_ldd:	0: ASI5Saddr nicht in ASI5LDD enthalten !0: ASI5Saddr in ASI5LDD enthalten
LogAddr:	logische Adresse

Return:

	0 wenn OK !0 wenn nicht OK
--	-------------------------------

4.2.27 Asi5SetCdi

Mit dieser Funktion wird das Konfigurationsdatenabbild für einen entsprechenden ASI-5 Teilnehmer geändert.

```
return_t (*Asi5SetCdi) (const unsigned char circuit, const
                        asi_5_slv_config_t *cdi, const unsigned short LogAddr).
```

Parameter:

circuit:	ASI Master Kreis
cdi:	abgerufenes Konfigurationsdatenabbild
LogAddr:	logische Adresse

Return:

	0 wenn OK !0 wenn nicht OK
--	-------------------------------

4.2.28 Asi5GetCdi

Mit dieser Funktion werden die Konfigurationsdaten für eine bestimmte Adresse aus dem ASI-5 Konfigurationsdatenabbild (ASI5CDI) abgerufen.

```
return_t (*Asi5GetCdi) (const unsigned char circuit, asi_5_slv_
config_t *cdi, const unsigned short LogAddr).
```

Parameter:

circuit:	ASi Master Kreis
cdi:	zurückgemeldetes Konfigurationsdatenabbild
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.29 Asi5SetPcd

Mit dieser Funktion wird ein neuer Konfigurationswert gespeichert.

Diese Funktion darf nur im ASI-5 Konfigurationsmodus ausgeführt werden.

```
return_t (*Asi5SetPcd) (const unsigned char circuit, const
asi_5_slv_config_t *pcd, const unsigned short LogAddr).
```

Parameter:

circuit:	ASi Master Kreis
pcd:	zu speichernde Konfigurationsdaten
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.30 Asi5GetPcd

Die Funktion dient dazu, die Konfigurationsdaten für eine bestimmte logische Adresse aus den permanenten ASI-5 Konfigurationsdaten (ASI5PCD) der aktuellen Konfigurationsinstanz abzurufen.

```
return_t (*Asi5GetPcd) (const unsigned char circuit, asi_5_slv_
config_t *pcd, const unsigned short LogAddr).
```

Parameter:

circuit:	ASi Master Kreis
pcd:	permanente Konfigurationsdaten
LogAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.31 Asi5StoreCdi

Die Funktion dient dazu, die Werte des ASI-5 Konfigurationsdatenabbildes (ASI5CDI) in die permanenten ASI-5 Konfigurationsdaten (ASI5PCD) der aktuellen Konfigurationsinstanz für einen einzelnen ASI-5 Teilnehmer zu kopieren.

Wenn das ASI-5 Modul in der ASI-5 Liste der aktivierten ASI-5 (ASI5LAS) vorhanden ist, wird die logische Adresse in die ASI-5 Liste der projizierten ASI-5 Teilnehmer (ASI5LPS) der aktuellen Konfigurationsinstanz aufgenommen.

Diese Funktion darf nur im ASI-5 Konfigurationsmodus ausgeführt werden!

```
return_t (*Asi5StoreCdi) (const unsigned char circuit).
```

Parameter:

circuit: ASI Master Kreis

Return:

0 wenn OK
!0 wenn nicht OK

4.2.32 Asi5StoreSlaveCdi

Die Funktion dient dazu, die Werte des ASI-5 Konfigurationsdatenabbildes (ASI5CDI) in die permanenten ASI-5 Konfigurationsdaten (ASI5PCD) der aktuellen Konfigurationsinstanz für einen einzelnen ASI-5 Teilnehmer zu kopieren.

Falls das ASI-5 Modul in der ASI-5 Liste der aktivierten ASI-5 Teilnehmer (ASI5LAS) vorhanden ist, wird die logische Adresse in die ASI-3 Liste der projizierten ASI-5 Teilnehmer (ASI5LPS) der aktuellen Konfigurationsinstanz aufgenommen.

Diese Funktion darf nur im ASI-5 Konfigurationsmodus ausgeführt werden!

```
return_t (*Asi5StoreSlaveCdi) (const unsigned char circuit, const unsigned short LogAddr).
```

Parameter:

circuit: ASI Master Kreis

Return:

0 wenn OK
!0 wenn nicht OK

4.2.33 Asi5ResetPcd

Die Funktion dient dazu, die permanenten Konfigurationsdaten für eine bestimmte logische Adresse in die permanenten ASI-5 Konfigurationsdaten (ASI5PCD) der aktuellen Konfigurationsinstanz nichtflüchtig auf den Standardwert zu setzen.

Diese Funktion darf nur im ASI-5 Konfigurationsmodus ausgeführt werden!

```
return_t (*Asi5ResetPcd) (const unsigned char circuit, const unsigned short LogAddr).
```

Parameter:

circuit: ASI Master Kreis

Return:

0 wenn OK
!0 wenn nicht OK

4.2.34 Asi5ResetAllPcd

Die Funktion dient dazu, die permanenten Konfigurationsdaten für eine bestimmte logische Adresse in die permanenten ASI-5 Konfigurationsdaten (ASI5PCD) der aktuellen Konfigurationsinstanz nichtflüchtig auf den Standardwert zu setzen.

Diese Funktion darf nur im ASI-5 Konfigurationsmodus ausgeführt werden!

```
return_t (*Asi5ResetAllPcd) (const unsigned char circuit).
```

Parameter:

circuit: ASi Master Kreis

Return:

0 wenn OK
!0 wenn nicht OK

4.2.35 Asi5GetLas

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LAS abzurufen.

```
return_t (*Asi5GetLas) (const unsigned char circuit, unsigned char *length, unsigned short *las).
```

Parameter:

circuit: ASi Master Kreis

length: Länge des las Puffers

las: Liste der aktivierten Teilnehmer

Return:

0 wenn OK
!0 wenn nicht OK

4.2.36 Asi5GetLps

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LPS abzurufen.

```
return_t (*Asi5GetLps) (const unsigned char circuit, unsigned char *length, unsigned short *lps).
```

Parameter:

circuit: ASi Master Kreis

length: Länge des lps Puffers

lps: Liste der geschützten Teilnehmer

Return:

0 wenn OK
!0 wenn nicht OK

4.2.37 Asi5GetLds

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LDS abzurufen.

```
return_t (*Asi5GetLds) (const unsigned char circuit, unsigned char *length, unsigned short *lds).
```

Parameter:

circuit: ASi Master Kreis

length: Länge des lds Puffers

lds: Liste der erkannten Teilnehmer

Return:

0 wenn OK
!0 wenn nicht OK

4.2.38 Asi5GetLms

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LMS abzurufen.

```
return t (*Asi5GetLms) (const unsigned char circuit, unsigned
char *length, unsigned short *lms).
```

Parameter:

circuit: ASI Master Kreis

length: Länge des LMS Puffers

lms: Liste der Teilnehmer mit überlappendem Prozessdatenab-
bild

Return:

0 wenn OK
!0 wenn nicht OK

4.2.39 Asi5GetLpf

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LPF abzurufen.

```
return t (*Asi5GetLpf) (const unsigned char circuit, unsigned
char *length, asi_5_lpf_list_t *lpf).
```

Parameter:

circuit: ASI Master Kreis

length: Länge des Lpf Puffers

lpf: Liste der Peripheriefehler

Return:

0 wenn OK
!0 wenn nicht OK

4.2.40 Asi5GetLis

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LIS abzurufen.

```
return t (*Asi5GetLis) (const unsigned char circuit, unsigned
char *length, unsigned short *lis).
```

Parameter:

circuit: ASI Master Kreis

length: Länge des lis Puffers

lis: Liste der inkonsistenten ASI-5 Teilnehmer

Return:

0 wenn OK
!0 wenn nicht OK

4.2.41 Asi5GetLda

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LDA abzurufen.

```
return t (*Asi5GetLda) (const unsigned char circuit, unsigned
char *length, unsigned short *lda).
```

Parameter:

circuit: ASI Master Kreis

length: Länge des lda Puffers

lda: Liste der doppelten logischen Adressen

Return:

0 wenn OK
!0 wenn nicht OK

4.2.42 Asi5GetLdd

Die Funktion dient dazu, die komplette Teilnehmerliste ASI5LDD abzurufen.

```
return_t (*Asi5GetLdd) (const unsigned char circuit, unsigned
char *length, unsigned short *ldd).
```

Parameter:

circuit: ASi Master Kreis

length: Länge des Ldd Puffers

ldd: Liste der erkannten Doppeladressen

Return:

0 wenn OK

!0 wenn nicht OK

4.2.43 Asi5GetSlaveAsiid

Die Funktion dient dazu, eine Liste ASIID von ASi-5 Teilnehmern abzurufen, die eine bestimmte logische Adresse nutzen.

```
return_t (*Asi5GetSlaveAsiid) (const unsigned char circuit, unsi-
gned char *elements, asiid_t *ASIIDList, const unsigned short
logAddr).
```

Parameter:

circuit: ASi Master Kreis

elements: Länge der ASIID Liste

ASIIDList: ASIID Liste der ASi-5 Teilnehmer, welche die logische Adresse ASI5Saddr nutzen

logAddr: logische Adresse

Return:

0 wenn OK

!0 wenn nicht OK

4.2.44 Asi5GetSlaveInfo

Die Funktion wird von der Host-Schnittstelle verwendet, um die komplette Informationsstruktur der ASi-5 Teilnehmer zu erhalten.

```
return_t (*Asi5GetSlaveInfo) (const unsigned char circuit,
asi_5_slvinfo_t *slv_info, const unsigned short logAddr).
```

Parameter:

circuit: ASi Master Kreis

slv_info: Informationsstruktur des ASi-5 Teilnehmers

logAddr: logische Adresse

Return:

0 wenn OK

!0 wenn nicht OK

4.2.45 Asi5GetSlaveInfoAsiid

Die Funktion dient dazu, die komplette Informationsstruktur der ASI-5 Teilnehmer zu erhalten.

Der spezifische ASI-5 Teilnehmer wird durch seine ASIID identifiziert.

```
return t (*Asi5GetSlaveInfoAsiid) (const unsigned char circuit,
    asi_5_slvinfo_t *slv_info, asiid_t ASIID).
```

Parameter:

circuit: ASI Master Kreis

slv_info: Informationsstruktur des ASI-5 Teilnehmers

ASIID: ASIID des ASI-5 Moduls, von dem die ASI-5 Teilnehmer-
Informationsstruktur abgerufen werden soll

Return:

0 wenn OK
!0 wenn nicht OK

4.2.46 Asi5GetSystemResources

Die Funktion dient dazu, einen Überblick über die genutzten Ressourcen des ASI-5 Systems zu erhalten.

```
return t (*Asi5GetSystemResources) (const unsigned char circuit,
    cm_systemResources_t *systemResources).
```

Parameter:

circuit: ASI Master Kreis

systemResources: Überblick über die genutzten Ressourcen des ASI-5 Sys-
tems

Return:

0 wenn OK
!0 wenn nicht OK

4.2.47 Asi5IdentifySlaveByLogicalAddress

Durch diese Funktion startet der ASI-5 Teilnehmer zur Identifikation eine Blinksequenz an den LEDs.

Wenn die logische ASI-5 Teilnehmer-Adresse Teil der ASI5LDA ist, wird mehr als ein Teilnehmer die optische Identifikation anwenden.

```
return t (*Asi5IdentifySlaveByLogicalAddress) (const unsigned
    char circuit, const unsigned short LogAddr).
```

Parameter:

circuit: ASI Master Kreis

logAddr: logische Adresse

Return:

0 wenn OK
!0 wenn nicht OK

4.2.48 Asi5IdentifySlaveByAsiid

Durch diese Funktion startet der ASI-5 Teilnehmer zur Identifikation eine Blinksequenz an den LEDs. Der jeweilige ASI-5 Teilnehmer wird über seine ASIID identifiziert.

```
return t (*Asi5IdentifySlaveByAsiid) (const unsigned char cir-
    cuit, const asiid_t ASIID).
```

Ausgabedatum: 16.05.2024

Parameter:

circuit: ASi Master Kreis

ASIID: ASIID des ASi-5 Teilnehmers

Return:

0 wenn OK

!0 wenn nicht OK

4.2.49 Asi5SetDeviceDesignatorByLogicalAddress

Diese Funktion speichert den Device Designator des ASi-5 Teilnehmers.

```
return t (*Asi5SetDeviceDesignatorByLogicalAddress) (const unsigned char circuit, const unsigned short LogAddr, const unsigned char *deviceDesignator).
```

Parameter:

circuit: ASi Master Kreis

logAddr: logische Adresse

deviceDesignator: enthält den Device Designator des ASi-5 Moduls

Return:

0 wenn OK

!0 wenn nicht OK

4.2.50 Asi5SetDeviceDesignatorByAsiid

Diese Funktion speichert den Device Designator des ASI-5 Teilnehmers.

```
return t (*Asi5SetDeviceDesignatorByAsiid) (const unsigned char
circuit, const asiid_t ASIID, const unsigned char *deviceDesigna-
tor).
```

Parameter:

circuit: ASI Master Kreis

ASIID: ASIID des ASI-5 Teilnehmers

deviceDesignator: enthält den Device Designator des ASI-5 Moduls

Return:

0 wenn OK
!0 wenn nicht OK

Beispiel:

```
return t ret;
asiid_t asiid;
unsigned char deviceDesignator[32] = "001:DeviceDesignator"; //
[TeilnehmerNummer:DeviceDesignator] max 31 + '0'

asiid.vendorId      = 0x0002; // vendor id B+W
asiid.productId[0] = 0x00;
asiid.productId[1] = 0x00;
asiid.productId[2] = 0x12;
asiid.productId[3] = 0x34;
asiid.productId[4] = 0x56;

ret = ctrl_func.Asi5SetDeviceDesignatorByAsiid(0, asiid, (const
unsigned char*)&deviceDesignator);
```

4.2.51 Asi5GetDeviceDesignatorByLogicalAddress

Diese Funktion liefert den Device Designator des ASI-5 Teilnehmers.

```
return t (*Asi5GetDeviceDesignatorByLogicalAddress) (const unsi-
gned char circuit, const unsigned short LogAddr, unsigned char
*deviceDesignator).
```

Parameter:

circuit: ASI Master Kreis

logAddr: logische Adresse

deviceDesignator: enthält den Device Designator des ASI-5 Moduls

Return:

0 wenn OK
!0 wenn nicht OK

4.2.52 Asi5GetDeviceDesignatorByAsiid

Diese Funktion liefert den Device Designator des Asi-5 Teilnehmers.

```
return_t (*Asi5GetDeviceDesignatorByAsiid) (const unsigned char
circuit, const asiid_t ASIID, unsigned char *deviceDesignator).
```

Parameter:

circuit: Asi Master Kreis

ASIID: ASIID des Asi-5 Teilnehmers

deviceDesignator: enthält den Device Designator des Asi-5 Moduls

Return:

0 wenn OK

!0 wenn nicht OK

4.2.53 Asi5AutoSetLogicalAddresses

Die Funktion wird von der Host-Schnittstelle genutzt, um die logische Adresse aller Asi-5 Teilnehmer automatisch mit der logischen Adresse 0 einzustellen.

Die Teilnehmer sind nach der Gerätekennung geordnet.

```
return_t (*Asi5AutoSetLogicalAddresses) (const unsigned char circuit).
```

Parameter:

circuit: Asi Master Kreis

Return:

0 wenn OK

!0 wenn nicht OK

4.2.54 Asi5SetEnergySavingState

Diese Funktion ändert den Energiesparzustand aller Asi-5 Teilnehmer, die der ausgewählten Energiespargruppe zugeordnet sind.

```
return_t (*Asi5SetEnergySavingState) (const unsigned char circuit,
const unsigned char EnSvGrp, const unsigned char EnSvState).
```

Parameter:

circuit: Asi Master Kreis

EnSvGrp: Asi-5 Energiespargruppe

EnSvState: Energiesparzustand:

0 Energiesparen ausgeschaltet

1 Energiesparen eingeschaltet

Return:

0 wenn OK

!0 wenn nicht OK

4.2.55 Asi5GetEnergySavingState

Diese Funktion liefert den Energiesparstatus der ausgewählten Energiespargruppe.

```
return_t (*Asi5GetEnergySavingState) (const unsigned char circuit, const unsigned char EnSvGrp, unsigned char *EnSvState).
```

Parameter:

circuit: ASi Master Kreis

EnSvGrp: ASi-5 Energiespargruppe

EnSvState: Energiesparzustand:

0 Energiesparen ausgeschaltet

1 Energiesparen eingeschaltet

Return:

0 wenn OK

!0 wenn nicht OK

4.2.56 Asi5WriteASi5Odi

Diese Funktion schreibt das Ausgangsdatenabbild von ASi-5 Teilnehmern.

```
return_t (*Asi5WriteASi5Odi) (const unsigned char circuit, const unsigned char *odi, const unsigned short length, const unsigned short offset).
```

Parameter:

circuit: ASi Master Kreis

odi: Ausgangsdatenabbild

length: Länge des zu schreibenden Abbildes

offset: Offset des zu schreibenden Abbildes

Return:

0 wenn OK

!0 wenn nicht OK

4.2.57 Asi5ReadASi5Odi

Diese Funktion liest das Ausgangsdatenabbild von ASi-5 Teilnehmern.

```
return_t (*Asi5ReadASi5Odi) (const unsigned char circuit, unsigned char *odi, unsigned short *length, const unsigned short offset).
```

Parameter:

circuit: ASi Master Kreis

odi: Ausgangsdatenabbild

length: Länge des zu lesenden Abbildes

offset: Offset des zu lesenden Abbildes

Return:

0 wenn OK

!0 wenn nicht OK

4.2.58 Asi5ReadASi5Idi

Diese Funktion liest das Eingangsdatenabbild von ASI-5 Teilnehmern.

```
return_t (*Asi5ReadASi5Idi) (const unsigned char circuit, unsigned char *idi, unsigned short *length, const unsigned short offset).
```

Parameter:

circuit: ASI Master Kreis

idi: Eingangsdatenabbild

length: Länge des zu lesenden Abbildes

offset: Offset des zu lesenden Abbildes

Return:

0 wenn OK

!0 wenn nicht OK

4.2.59 Asi5DiagGetMasterStatus

Diese Funktion gibt den Diagnosestatus des ASI-5 Masters zurück.

```
return_t (*Asi5DiagGetMasterStatus) (const unsigned char circuit, const unsigned char startIndex, unsigned char reqNumOfEntries, unsigned char *totalNumOfEntries, unsigned char *retNumOfEntries, diag_info_t *entries).
```

Parameter:

circuit: ASI Master Kreis

startIndex: Index des ersten Elements, das aus dem Diagnosestatus gelesen werden soll. Der erste Index ist 0.

reqNumOfEntries: Gewünschte Anzahl von Einträgen von Diagnoseinformationen aus dem Master-Diagnosestatus

totalNumOfEntries: Gesamtanzahl der Diagnoseinformationen im Master-Diagnosestatus

retNumOfEntries: Zurückgegebene Anzahl von Einträgen von Diagnoseinformationen aus dem Master-Diagnosestatus

entries: Einträge der Diagnoseinformationen des Diagnosestatus

Return:

0 wenn OK

!0 wenn nicht OK

4.2.60 Asi5GetParameters

Diese Funktion dient dazu, Parameter aus dem Parameterabbild eines ASI-5 Teilnehmers zu erhalten.

```
return_t (*Asi5GetParameters) (const unsigned char circuit, const unsigned short logAddr, const unsigned char startIndex, const unsigned char count, unsigned char *data).
```

Parameter:

circuit: ASI Master Kreis

logAddr: logische Adresse

startIndex: erster Index, um einen Teil des PI zu erhalten

count: Anzahl der auszulesenden Indizes

data: Daten, die im PI-Abschnitt des StartIndex und den folgenden Indizes abzulegen sind

Return:

0 wenn OK

!0 wenn nicht OK

4.2.61 Asi5SetParameters

Diese Funktion dient dazu, Parameter in das Parameterabbild des ASI-5 Teilnehmers zu übertragen.

```
return_t (*Asi5SetParameters) (const unsigned char circuit, const
unsigned short logAddr, const unsigned char startIndex, const
unsigned char count, const unsigned char *data).
```

Parameter:

circuit: ASI Master Kreis

logAddr: logische Adresse

startIndex: erster Index zur Modifizierung des PI

count: Anzahl der Indizes zur Modifizierung

data: Daten, die im PI auf dem StartIndex und den folgenden Indizes abzulegen sind

Return:

0 wenn OK

!0 wenn nicht OK

4.2.62 Asi5SetPsMode

Diese Funktion dient dazu, die aktuelle Betriebsart (ASI5PSMode) des Parameterservers einzustellen.

```
return_t (*Asi5SetPsMode) (const unsigned char circuit, ps_mode_e
psMode).
```

Parameter:

circuit: ASI Master Kreis

psMode: nächste Betriebsart des Parameterservers

Return:

0 wenn OK

!0 wenn nicht OK

4.2.63 Asi5GetPsMode

Diese Funktion dient dazu, die aktuelle Betriebsart (ASI5PSMode) des Parameterservers zu ermitteln.

```
return_t (*Asi5GetPsMode) (const unsigned char circuit, ps_mode_e
*actualMode, ps_mode_e *storedMode).
```

Parameter:

circuit: ASI Master Kreis

actualMode: Meldet die aktuelle Betriebsart des Parameterservers zurück

storedMode: Meldet die gespeicherte Betriebsart des Parameterservers zurück

Return:

0 wenn OK

!0 wenn nicht OK

4.2.64 Asi5GetStoredPi

Diese Funktion liest die gespeicherte PI für einen ASI-5 Teilnehmer aus dem ASI-Master. Die zurückgegebene PI ist abhängig vom Parameterabbild der aktuell aktiven Konfiguration.

```
return_t (*Asi5GetStoredPi) (const unsigned char circuit, const
unsigned short logAddr, unsigned long *profileID, unsigned short
*vendorID, unsigned char *PI).
```

Parameter:

circuit:	ASI Master Kreis
logAddr:	logische Adresse
profileID:	ProfileID, welche der Teilnehmer zum Zeitpunkt des Parameter-Backups hatte
vendorID:	VendorID, welche der Teilnehmer zum Zeitpunkt des Parameter-Backups hatte
PI:	Zeiger für das Parameterabbild des ASI-5 Teilnehmers

Return:

0	wenn OK
!0	wenn nicht OK

4.2.65 Asi5SetStoredPi

Diese Funktion speichert die PI für einen ASI-5 Teilnehmer in den Parameterabbildsatz der aktuell aktiven Konfiguration im ASI Master.

```
return_t (*Asi5SetStoredPi) (const unsigned char circuit, const
unsigned short logAddr, const unsigned char *PI).
```

Parameter:

circuit:	ASI Master Kreis
logAddr:	logische Adresse
PI:	Zeiger für das zu speichernde Parameterabbild des ASI-5 Teilnehmers

Return:

0	wenn OK
!0	wenn nicht OK

4.2.66 Asi5StorePi

Mit dieser Funktion wird eine neue Version des Parameterabbildes vom ASI-5 Teilnehmer in den ASI-5 Master geladen und gespeichert. Das Parameterabbild wird abhängig von der aktuellen Konfiguration im aktuellen Parameterabbildsatz gespeichert.

```
return_t (*Asi5StorePi) (const unsigned char circuit, const unsigned
short logAddr).
```

Parameter:

circuit:	ASI Master Kreis
logAddr:	logische Adresse

Return:

0	wenn OK
!0	wenn nicht OK

4.2.67 Asi5RestorePi

Diese Funktion lädt die gespeicherte PI in den Asi-5 Teilnehmer. Die PI ist abhängig von der aktuell aktiven Konfiguration.

```
return_t (*Asi5RestorePi) (const unsigned char circuit, const unsigned short logAddr).
```

Parameter:

circuit: Asi Master Kreis

logAddr: logische Adresse

Return:

0 wenn OK
!0 wenn nicht OK

4.2.68 Asi5StoreAllSlavePi

Diese Funktion wird vom Host-Interface genutzt, um die aktuell verwendeten Parameterabbilder aller Asi-5 Teilnehmer in den Asi-5 Master zu speichern. Die Parameterabbilder werden nichtflüchtig gespeichert.

Fällt das Speichern für einen Teilnehmer aus, so löscht der Asi-5 Master den gesamten PI-Satz.

```
return_t (*Asi5StoreAllSlavePi) (const unsigned char circuit).
```

Parameter:

circuit: Asi Master Kreis

Return:

0 wenn OK
!0 wenn nicht OK

4.2.69 Asi5RestoreAllSlavePi

Diese Funktion wird vom Host-Interface genutzt, um das Parameterabbild vom Asi-5 Master in alle Asi-5 Teilnehmer herunterzuladen.

```
return_t (*Asi5RestoreAllSlavePi) (const unsigned char circuit).
```

Parameter:

circuit: Asi Master Kreis

Return:

0 wenn OK
!0 wenn nicht OK

4.2.70 Asi5SlavenumberToLogAddr

Diese Funktion wandelt die Teilnehmernummer in eine logische Adresse um.

```
return_t (*Asi5SlavenumberToLogAddr) (const unsigned char circuit, const unsigned char slavenumber, unsigned short *p_logAddr).
```

Parameter:

circuit: Asi Master Kreis

slavenumber: Teilnehmernummer des Moduls

p_logAddr: logische Adresse des angefragten Teilnehmers

Return:

0 wenn OK
!0 wenn nicht OK

4.2.71 Asi5LogAddrToSlavenumber

Diese Funktion wandelt eine logische Adresse in eine Teilnehmernummer um.

```
return_t (*Asi5LogAddrToSlavenumber) (const unsigned char circuit, const unsigned short logAddr, unsigned char *p_slavenumber).
```

Parameter:

circuit: ASi Master Kreis

logAddr: logische Adresse des Moduls

p_slavenumber: Teilnehmernummer des angefragten Teilnehmers

Return:

0 wenn OK

!0 wenn nicht OK

4.2.72 Asi5DeltaList

Diese Funktion gibt die Liste der Konfigurationsfehler zurück.

```
return_t (*Asi5DeltaList) (const unsigned char circuit, unsigned short *length, unsigned short *deltaList).
```

Parameter:

circuit: ASi Master Kreis

length: Länge des Delta List Puffers

deltaList: Ausgabe der Deltaliste

Return:

0 wenn OK

!0 wenn nicht OK

4.2.73 Asi5InDeltaList

Diese Funktion prüft, ob die logische Adresse in der Liste der Konfigurationsfehler enthalten ist.

```
return_t (*Asi5InDeltaList) (const unsigned char circuit, unsigned char *in_deltaList, const unsigned short logAddr).
```

Parameter:

circuit: ASi Master Kreis

logAddr: logische Adresse

in_deltaList: 0: ASI5Saddr ist nicht in der Deltaliste enthalten
!0: ASI5Saddr ist in der Deltaliste enthalten

Return:

0 wenn OK

!0 wenn nicht OK

4.2.74 Asi5GetIoLength

Diese Funktion gibt die Länge der E/A-Daten des Teilnehmers zurück.

```
return_t (*Asi5GetIoLength) (const unsigned char circuit, const
unsigned short logAddr, unsigned char *odiLength, unsigned char
*idiLength).
```

Parameter:

circuit:	ASi Master Kreis
logAddr:	logische Adresse
odiLength:	Länge der Ausgangsdaten
idiLength:	Länge der Eingangsdaten

Return:

0	wenn OK
!0	wenn nicht OK

4.2.75 Asi5GetIoLengthByAsiid

Diese Funktion gibt die Länge der E/A-Daten des Teilnehmers anhand der ASIID zurück.

```
return_t (*Asi5GetIoLengthByAsiid) (const unsigned char circuit,
const _asiid_t ASIID, unsigned char *odiLength, unsigned char
*idiLength).
```

Parameter:

circuit:	ASi Master Kreis
ASIID:	ASIID
odiLength:	Länge der Ausgangsdaten
idiLength:	Länge der Eingangsdaten

Return:

0	wenn OK
!0	wenn nicht OK

4.2.76 Asi5ResetAllSlaves

Diese Funktion setzt alle ASi-5 Teilnehmer zurück.

```
return_t (*Asi5ResetAllSlaves) (const unsigned char circuit).
```

Parameter:

circuit:	ASi Master Kreis
----------	------------------

Return:

0	wenn OK
!0	wenn nicht OK

4.3 Übersicht der Control Befehle

CCtrlInitTimer	Initialisierung einer Timerfunktion
CCtrlDelay	Verzögerung in ms
CCtrlInitWdg	Initialisierung des Control III Watchdogs
CCtrlTriggerWdg	Trigger Control III Watchdog.
CCtrlEvalCycletime	Bewertung der Control III Zykluszeit
CCtrlReadParameter	Liest Control NV-Parameter.
CCtrlWriteParameter	Schreibt Control NV-Parameter
CCtrlReadFlags	Liest Control-Flags.
CCtrlWriteFlags	Schreibt Control-Flags.
CCtrlReadKey	Liest benutzerdefinierten eindeutigen Control-Schlüssel.
CCtrlPrintf	Printf-Funktion
CCtrlBreakpoint	Initialisierung des Debuggers
CCtrlDisplay	Control III Displayfunktion
CCtrlEnetSetipdata	Setzt IP-Stack Informationen ins richtige Format.
CCtrlEnetSocket	Öffnet den Socket.
CCtrlEnetSetsockopt	Setzt die Optionen für einen Socket.
CCtrlEnetBind	Verknüpft einen Socket mit einem Namen.
CCtrlEnetListen	Hört einen Socket nach Verbindungsanforderungen ab.
CCtrlEnetAccept	Akzeptiert eine Verbindung an einem Socket
CCtrlEnetConnect	Baut eine Verbindung über einen Socket auf.
CCtrlEnetSend	Sendet Daten an einen verbundenen Socket.
CCtrlEnetSendto	Sendet eine Nachricht an einen Socket, egal ob dieser verbunden ist oder nicht.
CCtrlEnetRecv	Empfängt Daten von einem verbundenen Socket.
CCtrlEnetRecvfrom	Empfängt Daten von einem Socket, egal, ob verbindungsorientiert oder nicht.
CCtrlEnetSocketclose	Schließt eine Socket-Verbindung.
CCtrlReadParameterAdditional	Liest 'Control additional NV' Parameter.
CCtrlWriteParameterAdditional	Schreibt 'Control NV' Parameter.
CCtrlFcntl	Schreibt / Liest 'fcntl' Merker

Tab. 4-10.

4.3.1 Ctrl Init Timer

Initialisierung eines Timer Interrupts

```
int (*CCtrlInitTimer) (unsigned long ticks_ms, void (*timer_
func) ( void ) );
```

Parameter:

Circuit:	ASi Master Kreis
ticks_ms:	Interrupt Time in ms
timer_func:	Callback Funktion des Timer-Interrupts

Return:—

4.3.2 Ctrl Delay

Verzögerungsfunktion

```
int (*CCtrlDelay) ( unsigned long ticks_ms );
```

Parameter:

Circuit:	ASi Master Kreis
ticks_ms:	Verzögerung in ms

Return:—

4.3.3 Ctrl Init wgd

Initialisierung eines Watchdogs für Control III

```
int (*CctrlInitWdg) ( unsigned long ticks );
```

Parameter:

Circuit:	ASi Master Kreis
ticks:	Watchdog Zeit in ms

Return:—

4.3.4 Ctrl Trigger wdg

Triggern des Control III Watchdogs

```
int (*CctrlTriggerWdg) ( void );
```

Parameter: —

Return:—

4.3.5 Ctrl Eval Cycle time

Die Funktion ermittelt die Zykluszeit des Control III Programms.

```
int (*CctrlEvalCycletime) ( void );
```

Parameter: —

Return:—

4.3.6 Ctrl Read Parameter

Ctrl Read Parameter liest nichtflüchtige Daten aus dem Flash.

```
int (*CtrlReadParameter) ( unsigned char *buffer, unsigned
short len, unsigned short adr );
```

Parameter:

len: Länge des zu lesenden Speichers

adr: Adresse des ersten Bytes der zu lesenden Daten

Return:

buffer: Lese-Puffer

4.3.7 Ctrl Write Parameter

Ctrl Write Parameter schreibt nichtflüchtige Daten in den Flash.

```
int (*CtrlWriteParameter) ( unsigned char *buffer, unsigned
short len, unsigned short adr );
```

Parameter:

buffer: Schreib-Puffer

len: Länge des Puffers

adr: Adresse des ersten Bytes, an welches die Daten geschrieben werden.

Return:—

4.3.8 Ctrl Read Flags

Ctrl Read Flags liest die ASi Control Flags.

```
int (*CtrlReadFlags) ( unsigned char *flags );
```

Parameter: —

Return:

flags ASi Control Flags

4.3.9 Ctrl Write Flags

Ctrl Write Flags schreibt die ASi Control Flags.

```
int (*CtrlWriteFlags) ( unsigned char flags );
```

Parameter:

flags: ASi Control Flags

Return:—

4.3.10 Ctrl Read Key

Ctrl Read Key liest benutzerdefinierten, eindeutigen Control III Schlüssel.

```
int (*CtrlReadKey) ( unsigned int *key );
```

Parameter: —

Return:

key: benutzerdefinierter Control III Schlüssel

4.3.11 Ctrl printf

Printf-Funktion.

```
int (*CCTRLPrintf) ( const char *format, ... );
```

Parameter: —**Return:—****4.3.12 Ctrl Breakpoint**

Initialisierung des Debuggers; „Erster“ Breakpoint.

```
void (*CCTRLBreakpoint) (void);
```

Parameter: —**Return:—****4.3.13 Ctrl Display**

Selbstdefinierbare Anzeige auf dem Display des Gateway.

```
int (*CCTRLDisplay) ( unsigned char mode, cctrl_disp_t disp_buffer );
```

Parameter:

```
mode CCTRL_DISP_MODE_TRADITIONAL
      CCTRL_DISP_MODE_SPONTANEOUS
```

```
disp_buffer.show: 0 = löschen
                  1 = anzeigen
```

```
disp_buffer.type: CCTRL_DISP_TYP_4LINES
                  = 4 Zeilen Text
                  CCTRL_DISP_TYP_BIGNUM
                  = große Zahl + 1 Zeile Text
```

```
disp_buffer.time: Zeit der Darstellung im Display
                  (min 2 sek.)
```

```
disp_buffer.big:  Puffer für große Zahl
```

```
disp_buffer.lines: Puffer für 4 Zeilen Text
```

Return: 0 = OK

!0 = nicht Ok

4.3.14 CCtrl Enet Set IP data

Diese Funktion setzt IP-Stack Informationen ins richtige Format.

```
int (*CCEnetSetipdata) ( unsigned char address_family, unsigned short port, char *ip, unsigned char *buffer );
```

Parameter:

address_family: -> enet.h (address families)

Internet IPV4: AF_INET

Internet IPV6: AF_INET6

port: Port Nummer

ip: IP Adresse String

buffer: Puffer für folgende Informationen

u_char sa_len; Gesamtlänge

u_int sa_family; Adress-Familie

char sa_data[30]; tatsächlich länger; Adresswert

-> [0] - [1] port

-> [2] - [5] ip address

Return:

0 wenn OK

! wenn Fehler

4.3.15 CCtrl Enet Socket

Diese Funktion öffnet einen Socket (max 3).

```
int (*CCEnetSocket) ( unsigned char address_family, unsigned char types, int protocol );
```

Verbotene Ports ->

port == 80 /HTTP/

port == 44818 /ENIP_C01_PORT offiziell assigned as 'rockwell-encap/

port == 2222 /ENIP_PORT officially assigned as 'unreg-ab2'/

port == 502 /TCP listener port/

port == 546 /DHCPv6 Client/

port == 547 /DHCPv6 Server/

port == 57 /UDP 57 already used/

port == 87 /UDP 87 already used/

port == 67 /Bootstrap Protocol (BOOTP) Client; also used by DHCP/

port == 68 /Bootstrap Protocol (BOOTP) Client; also used by DHCP/

port == 69 /TFTP Port/

Parameter:

address_family -> enet.h (address families)

Internet IPV4: AF_INET

Internet IPV6: AF_INET6

types -> enet.h

TCP: SOCK_STREAM

UDP: SOCK_DGRAM

Raw Protocol at Network Layer: SOCK_RAW

protocol -> enet.h

Internet Protocol (IP): 0 or IPPROTO_IP

ICMP: 1

Return:

=0 = Socket Nummer

-1 = Fehler

4.3.16 CCtrl Enet Set sockopt

Diese Funktion setzt Optionen für einen Socket.

```
int (*CCEnetSetsockopt) ( int sock, int level, int optname,
const void *optval, int len );
```

Parameter:

sock	Ein gültiger Socket-Deskriptor
level	SOL_SOCKET
optval	-> enet.h (Option 'Merker pro Socket' / zusätzliche Optionen, nicht enthalten in 'so_options')
len	sizeof(optval)

Return:

0	wenn OK
-1	fehlgeschlagen
1	kein Control III Socket

4.3.17 CCtrl Enet Bind

Diese Funktion verknüpft einen Socket mit einem Namen.

```
int (*CCEnetBind) ( int sock, unsigned char address_family,
unsigned short port, unsigned int address );
```

Parameter:

sock	Ein gültiger Socket-Deskriptor
address_family	-> enet.h (address families)
port	Port
address	ausgewählte Adresse

Return:

0	wenn OK
1	Port verboten
2	Anbindung fehlgeschlagen
3	no control III port

4.3.18 CCtrl Enet Listen

Diese Funktion hört einen Socket nach Verbindungsanforderungen ab.

```
int (*CCEnetListen) ( int sock, int backlog );
```

Parameter:

sock	Ein gültiger Socket-Deskriptor
backlog	Anzahl der zwischengespeicherten Anfragen
returns:	
0	wenn OK
-1	wenn kein Control III Port oder Hören fehlgeschlagen

4.3.19 CCtrl Enet Accept

Diese Funktion akzeptiert eine Verbindung an einem Socket.

```
int (*CCEnetAccept) ( int sock, const void *addr, const void
*len );
```

Parameter:

sock	Ein gültiger Socket-Deskriptor
addr	points on struct sockaddr
len	sizeof struct sockaddr

Return:

0	wenn OK
-1	wenn Fehler akzeptiert
1	wenn kein Control III Socket

4.3.20 CCtrl Enet Connect

Diese Funktion baut eine Verbindung über einen Socket auf.

```
int (*CCtrlEnetConnect) ( int sock, unsigned char *address );
```

Parameter:

sock Ein gültiger Socket-Deskriptor

address points on struct sockaddr returned from SetIpData

Return:

0 wenn OK

-1 wenn Verbindungsfehler

4.3.21 CCtrl Enet Send

Diese Funktion sendet Daten an einen verbundenen Socket.

```
int (*CCtrlEnetSend) ( int sock, unsigned char *buffer, int len,
int flag );
```

Parameter:

sock Ein gültiger Socket-Deskriptor

buffer Ein Puffer, der die Daten enthält, die an den entfernten Host gesendet werden.

len Die Anzahl der Bytes, die aus dem Puffer an den entfernten Host gesendet werden.

flag Gibt den Typ des Nachrichtenempfangs an.

Der Parameter 'flags' kann beliebige Kombinationen der folgenden Flags enthalten, die mit einem binären OR (|) verknüpft werden.

-> enet.h

Return:

Anzahl der gesendeten

Bytes

-1 wenn kein Control III Socket oder Fehler

4.3.22 CCtrl Enet Send to

Diese Funktion sendet eine Nachricht an einen Socket, egal ob dieser verbunden ist oder nicht.

```
int (*CCtrlEnetSendto) ( int sock, unsigned char *address, unsigned char *buffer, int len );
```

Parameter:

sock Ein gültiger Socket-Deskriptor

buffer: Die Sendedaten werden aus dem Puffer gelesen.

address Stack Informationen aus EnetSetipdata();

len Die Anzahl Bytes, die aus dem Puffer gesendet werden.

Return:

Anzahl der an den ent-

fernten Host gesendeten

Bytes

-1 wenn kein Control III Socket oder Fehler

4.3.23 Ctrl Read Parameter

CCTRL Enet Recv

Diese Funktion empfängt Daten von einem verbundenen Socket.

```
int (*CCTRLEnetRecv) ( int sock, unsigned char *buffer, int len,
int flag );
```

Parameter:

sock	Der socket muss eine socket Ressource sein.
buffer	Empfangspuffer
len	Pufferlänge
flag	Gibt den Typ des Nachrichteneempfangs an. Der Wert von flags kann jede beliebige Kombination der folgenden Flags sein, verknüpft mit dem binären ODER () Operator. -> enet.h

Return:

Anzahl der empfangenen
Bytes

4.3.24 CCTRL Enet Recv from

Diese Funktion empfängt Daten von einem Socket, egal, ob verbindungsorientiert oder nicht.

```
int (*CCTRLEnetRecvfrom) ( int sock, unsigned char *buffer, int
len, unsigned char *address, unsigned int *address_len );
```

Parameter:

sock	Der socket muss eine socket Ressource sein.
buffer	Empfangspuffer
len	Pufferlänge
address	Puffer für Stack Informationen
address_len	Adresslänge

Return:

Anzahl der empfangenen
Bytes

4.3.25 CCTRL Enet Socket close

Diese Funktion schließt eine Socket-Verbindung.

```
void (*CCTRLEnetSocketclose) ( int sock );
```

Parameter:

sock	Ein gültiger Socket-Deskriptor
------	--------------------------------

Return: —

4.3.26 CCTRL Read Parameter Additional

Diese Funktion liest 'Control additional NV' Parameter.

```
int (*CCTRLReadParameterAdditional) ( unsigned char *buffer,
unsigned int len, unsigned int adr );
```

Parameter:

len	Pufferlänge
adr	Adresse des ersten Bytes, der gelesen werden soll

Return:

buffer	Lesepuffer
--------	------------

4.3.27 CCtrl Write Parameter Additional

Diese Funktion schreibt 'Control additional NV' Parameter.

```
int (*CCtrlWriteParameterAdditional) ( unsigned char *buffer,
unsigned int len, unsigned int adr );
```

Achtung: Der Parameter wird nicht auf der Chipkarte gespeichert!

Parameter:

buffer:	Schreibpuffer
len:	Pufferlänge
adr:	Adresse des ersten Bytes, der geschrieben werden soll

Return: —

4.3.28 CCtrlFcntl

Mit dieser Funktion wird der 'fcntl' Merker aufgerufen bzw. gesetzt.

```
error = CCtrlFcntl ( sock, command, argument );
```

Parameter:

sock:	the control III socket
command:	Befehl, siehe <code>enet.h</code>
argument:	die zu schreibenden Merker

Return:

	Merker wenn ok
	!0 wenn nicht ok

4.4 Übersicht der BACnet-Befehle

CCtrlBACnetWriteBinaryOutputPresentValue	Present-Value der BACnet Objektinstanz vom Typ Binary Output schreiben
CCtrlBACnetReadBinaryOutputPresentValue	Present-Value der BACnet Objektinstanz vom Typ Binary Output lesen
CCtrlBACnetClearBinaryOutputPresentValue	NULL in das Priority-Array der BACnet Objektinstanz vom Typ Binary Output schreiben
CCtrlBACnetWriteAnalogOutputPresentValue	Present-Value der BACnet Objektinstanz vom Typ Analog Output schreiben
CCtrlBACnetReadAnalogOutputPresentValue	Present-Value der BACnet Objektinstanz vom Typ Analog Output lesen
CCtrlBACnetClearAnalogOutputPresentValue	NULL in das Priority-Array der BACnet Objektinstanz vom Typ Analog Output schreiben
CCtrlBACnetWriteIntegerValuePresentValue	Present-Value der BACnet Objektinstanz vom Typ Integer Value schreiben
CCtrlBACnetReadIntegerValuePresentValue	Present-Value der BACnet Objektinstanz vom Typ Integer Value lesen
CCtrlBACnetClearIntegerValuePresentValue	NULL in das Priority-Array der BACnet Objektinstanz vom Typ Integer Value schreiben
CCtrlBACnetReadBinaryInputPresentValue	Present-Value der BACnet Objektinstanz vom Typ Binary Input lesen
CCtrlBACnetReadAnalogInputPresentValue	Present-Value der BACnet Objektinstanz vom Typ Analog Input lesen
CCtrlBACnetWriteODI	Ausgangsdaten über den BACnet WriteProperty-Dienst an ASi Teilnehmer schreiben
CCtrlBACnetWrite16BitODI	vier Kanäle 16bit ODI an einen ASi Teilnehmer schreiben
CCtrlBACnetDataExchange	ASi Teilnehmerdaten austauschen und execution control flags auslesen
CCtrlBACnetReadInstance	BACnet Instanz lesen
CCtrlBACnetWriteInstance	BACnet Instanz schreiben

Tab. 4-11. Übersicht der BACnet-Befehle

4.4.1 CCtrlBACnetWriteBinaryOutputPresentValue

Diese Funktion schreibt den Present-Value der angegebenen BACnet Objektinstanz vom Typ Binary Output.

```
int (*CCtrlBACnetWriteBinaryOutputPresentValue) (unsigned int
instance, unsigned char priority, unsigned char value).
```

Parameter:

instance:	Nummer der Objektinstanz
priority:	Priorität für das Schreiben (1-16)
value:	zu schreibender Wert
	0: BACNET_BINARY_INACTIVE
	1: BACNET_BINARY_ACTIVE

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.2 CCtrlBACnetReadBinaryOutputPresentValue

Diese Funktion liest den Present-Value der angegebenen BACnet Objektinstanz vom Typ Binary Output.

```
int (*CCtrlBACnetReadBinaryOutputPresentValue) (unsigned int
instance, unsigned char * value).
```

Parameter:

instance:	Nummer der Objektinstanz
value:	Zeiger auf den Speicher, in dem das Ergebnis gespeichert werden soll
	0: BACNET_BINARY_INACTIVE
	1: BACNET_BINARY_ACTIVE

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.3 CCtrlBACnetClearBinaryOutputPresentValue

Diese Funktion schreibt den Wert NULL in das Priority-Array der angegebenen BACnet Objektinstanz vom Typ Binary Output.

```
int (*CCtrlBACnetClearBinaryOutputPresentValue) (unsigned int
instance, unsigned char priority).
```

Parameter:

instance:	Nummer der Objektinstanz
priority:	Priorität für das Schreiben (1-16)

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.4 CCtrlBACnetWriteAnalogOutputPresentValue

Diese Funktion schreibt den Present-Value der angegebenen BACnet Objektinstanz vom Typ Analog Output.

```
int (*CCtrlBACnetWriteAnalogOutputPresentValue) (unsigned int
instance, unsigned char priority, float value).
```

Parameter:

instance:	Nummer der Objektinstanz
priority:	Priorität für das Schreiben (1-16)
value:	zu schreibender Wert

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.5 CCtrlBACnetReadAnalogOutputPresentValue

Diese Funktion liest den Present-Value der angegebenen BACnet Objektinstanz vom Typ Analog Output.

```
int (*CCtrlBACnetReadAnalogOutputPresentValue) (unsigned int
instance, float * value).
```

Parameter:

instance:	Nummer der Objektinstanz
value:	Zeiger auf den Speicher, in dem das Ergebnis gespeichert werden soll

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.6 CCtrlBACnetClearAnalogOutputPresentValue

Diese Funktion schreibt NULL in das Priority-Array der angegebenen BACnet Objektinstanz vom Typ Analog Output.

```
int (*CCtrlBACnetClearAnalogOutputPresentValue) (unsigned int
instance, unsigned char priority).
```

Parameter:

instance:	Nummer der Objektinstanz
priority:	Priorität für das Schreiben (1-16)

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.7 CCtrlBACnetWriteIntegerValuePresentValue

Diese Funktion schreibt den Present-Value der angegebenen BACnet Objektinstanz vom Typ Integer Value.

```
int (*CCtrlBACnetWriteIntegerValuePresentValue) (unsigned int
instance, unsigned char priority, int value).
```

Parameter:

instance:	Nummer der Objektinstanz
priority:	Priorität für das Schreiben (1-16)
value:	zu schreibender Wert

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.8 CCtrlBACnetReadIntegerValuePresentValue

Diese Funktion liest den Present-Value der angegebenen BACnet Objektinstanz vom Typ Integer Value.

```
int (*CCtrlBACnetReadIntegerValuePresentValue) (unsigned int
instance, int * value).
```

Parameter:

instance:	Nummer der Objektinstanz
value:	Zeiger auf den Speicher, in dem das Ergebnis gespeichert werden soll

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.9 CCtrlBACnetClearIntegerValuePresentValue

Diese Funktion schreibt NULL in das Priority-Array der angegebenen BACnet Objektinstanz vom Typ Integer Value.

```
int (*CCtrlBACnetClearIntegerValuePresentValue) (unsigned int
instance, unsigned char priority).
```

Parameter:

instance:	Nummer der Objektinstanz
priority:	Priorität für das Schreiben (1-16)

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.10 CCtrlBACnetReadBinaryInputPresentValue

Diese Funktion liest den Present-Value der angegebenen BACnet Objektinstanz vom Typ Binary Input.

```
int (*CCtrlBACnetReadBinaryInputPresentValue) (unsigned int
instance, unsigned char * value).
```

Parameter:

instance:	Nummer der Objektinstanz
value:	Zeiger auf den Speicher, in dem das Ergebnis gespeichert werden soll

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.11 CCtrlBACnetReadAnalogInputPresentValue

Diese Funktion liest den Present-Value der angegebenen BACnet Objektinstanz vom Typ Analog Input.

```
int (*CCtrlBACnetReadAnalogInputPresentValue) (unsigned int
instance, float * value).
```

Parameter:

instance:	Nummer der Objektinstanz
value:	Zeiger auf den Speicher, in dem das Ergebnis gespeichert werden soll

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.12 CCtrlBACnetWriteODI

Diese Funktion schreibt Ausgangsdaten an bestimmte ASi Teilnehmer über den BACnet WriteProperty-Dienst.

```
int (*CCtrlBACnetWriteODI) (unsigned char Circuit, AASiProcess-
Data ODI, AASiSlaveAddr First, unsigned char Amount, unsigned
char Priority).
```

Parameter:

circuit:	ASi Master Kreis
ODI:	32 Byte Ausgangsdaten (Ausgangsdatenabbild) Jeder ASi Teilnehmer verwendet ein Nibble eines Bytes. Format: Byte 0, low nibble: 'erste' Teilnehmerdateneinheit Byte 0, high nibble: 'erste+1' Teilnehmerdateneinheit
first:	Index des ersten ASi-Teilnehmers, an den Daten gesendet werden sollen (0-63)
amount:	Anzahl der ASi Teilnehmer, an die Daten gesendet werden sollen (1-64)
priority:	Priorität für das Schreiben (1-16)

Return:

—

4.4.13 CCtrlBACnetWrite16BitODI

Diese Funktion schreibt vier Kanäle 16 Bit ODI an einen ASi Teilnehmer mit z.B. analogem Teilnehmerprofil 7.3 oder 7.4.

```
int (*CCtrlBACnetWrite16BitODI) ( unsigned char Circuit, AASiSlaveAddr Address, AASi16BitData Out, unsigned char Priority).
```

Parameter:

circuit:	ASi Master Kreis
address:	Teilnehmeradresse (1-31)
out:	4 Kanäle mit 16-Bit Werten Wort 0: Kanal 1 ... Wort 3: Kanal 4
priority:	Priorität für das Schreiben (1-16)

Return:

—

4.4.14 CCtrlBACnetDataExchange

Diese Funktion übernimmt den Austausch von ASi Teilnehmer-Daten und liest die EcFlags aus.

```
int (*CCtrlBACnetDataExchange) (unsigned char Circuit, AASiProcessData ODI, AASiProcessData IDI, AASiEcFlags *EcFlags, unsigned char Priority).
```

Parameter:

circuit:	ASi Master Kreis
ODI:	32 Byte Ausgangsdaten an die Teilnehmer (Ausgangsdatenabbild) Jeder ASi Teilnehmer verwendet ein Nibble eines Bytes. Format: Byte 0, low nibble: Daten des Teilnehmers 0/0A Byte 0, high nibble: Daten des Teilnehmers 1/1A ... Byte 15, low nibble: Daten des Teilnehmers 30/30A Byte 15, high nibble: Daten des Teilnehmers 31/31A Byte 16, low nibble: Daten des Teilnehmers 0B Byte 16, high nibble: Daten des Teilnehmers 1B ... Byte 31, low nibble: Daten des Teilnehmers 30B Byte 31, high nibble: Daten des Teilnehmers 31B
priority:	Priorität für das ODI-Schreiben
Return:	
IDI:	32 Byte Eingangsdaten von den Teilnehmern (Eingangsdatenabbild) Format: identisch mit ODI
ECFlags:	Execution control flags (2 Bytes) des ASi Masters

4.4.15 CCtrlBACnetReadInstance

Diese Funktion liest Eigenschaften.

```
int (*CCtrlBACnetReadInstance) (unsigned char objtype, unsigned int instance, int property, unsigned int arrayindex, void * data_buffer, unsigned int data_buffer_size, unsigned int number_elements, unsigned int datatype_tag).
```

Parameter:

objtype:	Kennung des Objekttyps
instance:	Nummer der Objektinstanz
property:	Kennung der Eigenschaft
arrayindex:	Index oder alle Einsen (~0) für Nicht-Array-Typen
data_buffer:	Zeiger auf Puffer geeigneter Größe des gewünschten Datentyps
data_buffer_size:	Größe des Puffers für den gewünschten Datentyp
number_elements:	Anzahl der Elemente im Puffer
datatype_tag:	Tag, der den Pufferinhalt angibt

Return:

0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

4.4.16 CCtrlBACnetWriteInstance

Diese Funktion schreibt Eigenschaften.

```
int (*CCtrlBACnetWriteInstance) (unsigned char objtype, unsigned
int instance, int property, unsigned int arrayindex, unsigned
char priority, void * data_buffer, unsigned int data_buffer_size,
unsigned int number_elements, unsigned int datatype_tag).
```

Parameter:

objtype:	Kennung des Objekttyps
instance:	Nummer der Objektinstanz
property	Kennung der Eigenschaft
arrayindex:	Index oder alle Einsen (~0) für Nicht-Array-Typen
priority:	Priorität für das Schreiben (1-16)
data_buffer:	Zeiger auf Puffer geeigneter Größe des gewünschten Datentyps
data buffer size:	Größe des Puffers für den gewünschten Datentyp
number_elements:	Anzahl der Elemente im Puffer
datatype_tag:	Tag, der den Pufferinhalt angibt

Return:


0:	kein Fehler aufgetreten
!0:	Funktion gibt Fehlercode zurück

5. Technische Daten


Bei der Installation der **Control III**-Funktionalität ist auf die Auswahl der richtigen Version zu achten:

Die folgenden Downloads stehen zur Verfügung:

[1]

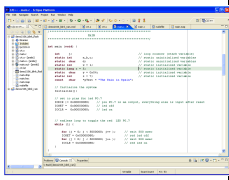
[Download: BW2582 | Control III, Programmierung in C \(Klein-SPS\) 8.0.0.0 \(ASI-5/ASI-3\)](#)
Downloads: 0385.3 MB

[2]

[Download: BW2582 | Control III, Programmierung in C \(Klein-SPS\) 7.0.5.0 \(ASI-3\)](#)
Downloads: 0385.2 MB

- [1] für ASI-3 Master (bis Version 7.0.5.0)
- [2] für ASI-5/ASI-3 Master (ab Version 8.0.0.0).

5.1 Übersicht

Artikel Nr.	BW2582	
	für ASI-3 Master (bis Version 7.0.5.0)	für ASI-5/ASI-3 Master (ab Version 8.0.0.0)
Allgemein		
<div></div> <div>Komplette IDE mit Eclipse und GCC Vollständiges Debugging des Steuerprogramms mit Eclipse und GDB</div>		
Systemvoraussetzungen	32 Bit Version von Java Runtime Environment (JRE) oder Java Development Kit (JDK) zwingend erforderlich, die Eclipse Funktionalität ist nicht geeignet für die Verwendung in einer 64 Bit Java Version	
	Windows Version XP / Vista / Windows 7 / Windows 8 / Win10 (ab Version 7.00)	Windows 7 / Windows 8 / Win10
Klein-SPS Beschreibung		
Arbeitsspeicher (RAM) / Programmspeicher (ROM) dynamisch aufgeteilt	ASI-3 max. 28 KByte	ASI-5 max. 256 KByte
Merkerbereich / Feldbus	256 Byte	
Remanenter Datenspeicher (auch über Chipkarte)	1 KByte	1 KByte ⁽¹⁾
Zykluszeit (1 KBit-/1000 Wortanw.)	1,0 ... 20 ms je nach Gerät und Applikation	typische Zykluszeit 1 ms
Verarbeitung		
API	Angelehnt an ASIDRV	
Timer	Ein konfigurierbarer Timerinterrupt, aus dem beliebig viele Timer und Zähler abgeleitet werden können	
Programmierbare Timerzeiten	1 ms bis 2 ³² ms	
Ein- und Ausgänge	bis zu 496 digitale E/As und 248 Analogwerte von ASI-3 Modulen	bis zu 3 MB E/A-Daten von ASI-5 und ASI-3 Modulen

Tab. 5-12.

Ausgabedatum: 16.05.2024

Artikel Nr.	BW2582	
	für ASi-3 Master (bis Version 7.0.5.0)	für ASi-5/ASi-3 Master (ab Version 8.0.0.0)
Programmierung		
Compiler	GCC ARM C-Compiler	
Debugger	GDB mit Eclipse	
Programmiersprache	C (oder Assembler)	
Programmiergeräte	PC	
Busanschlüsse	PROFIBUS, PROFINET, EtherNet/ IP, Modbus TCP	PROFINET, EtherNet/IP, Modbus TCP, Sercos III, POWERLINK, BACnet, EtherCAT
Extras		
Identifikation	Eindeutige 32 Bit Kennung im Gerät	
Chipkarte	Redundanter Speicher für Steuerprogramm und Parameter	
Ethernet-Schnittstelle	Fernwartung	

Tab. 5-12.

(1) EtherNet/IP-Gateways haben zusätzlich 1 MB Speicher, welcher nicht mit der Chipkarte synchronisiert wird.

5.2 Merker

Der Merkerbereich ist transparent und wird durch den Anwender verwaltet.

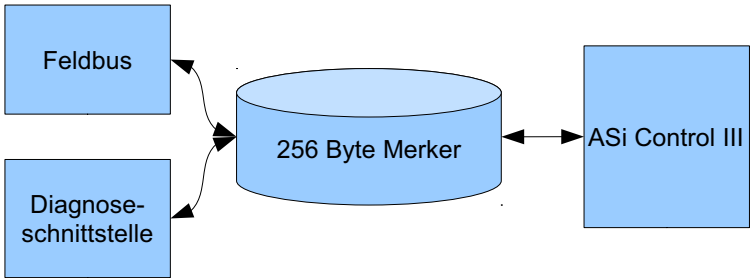


Abb. 5-10. Darstellung Merkerbereich

5.3 Nichtflüchtige Parameter

Die nichtflüchtigen Parameter sind transparent und werden durch den Anwender verwaltet.

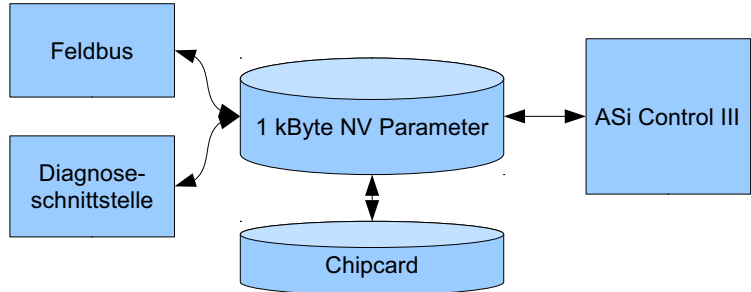


Abb. 5-11. Darstellung nichtflüchtige Parameter

5.4 Zugriffsrechte auf den Ausgangsdatenbereich

Feldbus und Control III Programm können gleichzeitig Ausgänge setzen. Die Zugriffsrechte können bitweise oder kanalweise vergeben werden.

Die dafür relevanten Kommandos werden in Kapitel 4 näher beschrieben:

- <Kap. 4.1.38 "ASi Read Ctrl Acc ODI">
- <Kap. 4.1.39 "ASi Write Ctrl Acc ODI">
- <Kap. 4.1.40 "ASi Read Ctrl Acc AODI">
- <Kap. 4.1.41 "ASi Write Ctrl Acc AODI">
- <Kap. 4.2.1 "Asi5ReadCtrlAccODI">
- <Kap. 4.2.2 "Asi5WriteCtrlAccODI">
- <Kap. 4.2.3 "Asi5ClearCtrlAccODI">

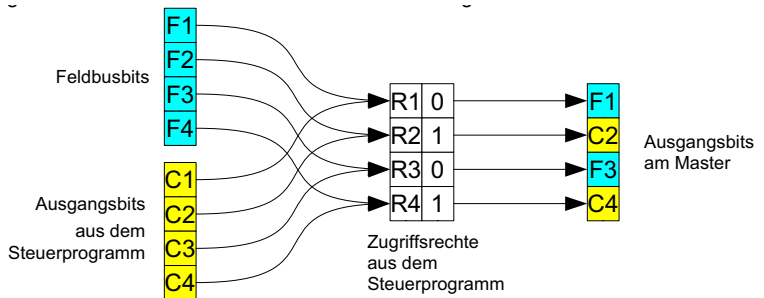


Abb. 5-12. Darstellung Zugriffsrechte Ausgangsdatenbereich

6. Fehlermeldungen

Dieses Kapitel soll Ihnen bei eventuell auftretenden Problemen helfen, die Fehler zu erkennen und zu beheben.

6.1 error: control not activated!

Sollte die Eclipse-Konsole folgende Fehlermeldung anzeigen so muss Control III für Ihr Gateway noch freigeschaltet werden (siehe Kap. <Installation von Eclipse Control III>).

```

-----
+++ CONTROL III +++
-----

communication port set to UDP:192.168.42.149.

error: control not activated!

have a nice day.

```

Abb. 6-13. error: control not activated

6.2 error: wrong control version

Steht in der Eclipse-Konsole die Meldung 'error: wrong control version', so besitzen Sie ein Gateway von Bihl+Wiedemann mit einer anderen Control-Version, welche nicht über die Fähigkeit der C-Programmierung verfügt. Wenden Sie sich in diesem Fall an den Bihl+WiedemannHersteller Support.

```

-----
+++ CONTROL III +++
-----

communication port set to COM4.

error: wrong control version!

have a nice day.

```

Abb. 6-14. error: wrong control version!

6.3 Launching problem

Bei dieser Fehlermeldung fehlt Eclipse die Zuordnung zu Ihrem Projekt. Klicken Sie hierzu einfach in das Editorfenster und führen Sie ihre Eingabe erneut aus.

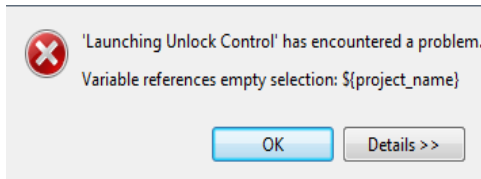


Abb. 6-15. Launching problem

6.4 Es wird keine oder eine falsche Zykluszeit angezeigt.

Sollten Sie die Zykluszeit über Eclipse auslesen und immer den Wert 0 angezeigt bekommen, so fehlt im Programmcode die Zeile:

```
/* check Cycletime */
cctrl_func.CCtrlEvalCycletime();
```

6.5 Das Gateway geht in keinen Haltepunkt.

Sollte das Gateway nicht in einem Haltepunkt stoppen, besteht die Möglichkeit, dass das auto-start-flag gesetzt ist oder es fehlt im Programmcode folgende Zeile zur Initialisierung des Debuggers:

```
//initialization of the Debugger
cctrl_func.CCtrlBreakpoint();
```

6.6 Das Programm geht immer in einen Haltepunkt.

Ist dies der Fall und es ist keine Initialisierung des Debuggers im Code vorhanden, so liegt der Fehler im Programmcode. Die häufigste Ursache hierfür ist ein uninitialisierter Zeiger/Pointer. Bitte prüfen Sie ihren Programmcode. Sollten Sie zudem noch das Autostart-Flag gesetzt haben, so können Sie beim Start des Gateways einen Reset durchführen. Betätigen Sie hierfür die beiden Tasten 'Mode' und 'Set' und schalten das Gateway ein. Der vorhandene Programmcode wird gelöscht und das Gateway startet wieder.

6.7 Es lassen sich keine Ausgänge durch Control III beeinflussen.

Prüfen Sie, ob die Slaves richtig projektiert sind und das Gateway keinen Konfigurationsfehler anzeigt. Führen sie ggf. die Projektierung der Slaves erneut durch. Sollte der Fehler immer noch bestehen, dann fehlen möglicherweise die Zugriffsrechte. Diese werden durch folgende Funktion vergeben:

```
cctrl_func.AASiWriteCtrlAccODI ( ... );
```

Weitere Informationen finden Sie hierzu im Kap. [<ASi Read Ctrl Acc AODI>](#).

7. Anzeigen der Ziffernanzeige

Im Grundzustand des Projektierungsmodus werden im Zweisekundentakt nacheinander die Adressen aller erkannten ASi Teilnehmer angezeigt. Ein leeres Display deutet auf eine leere LDS (Liste der erkannten ASi Teilnehmer) hin, d.h., es wurden keine ASi Teilnehmer erkannt.

Im Grundzustand des geschützten Betriebsmodus ist die Anzeige leer oder zeigt die Adresse einer Fehlbelegung an.

Während einer manuellen Adressenprogrammierung hat die Anzeige einer ASi Adresse natürlich eine andere Bedeutung.

Alle Anzeigen, die größer als 31 sind, also nicht als ASi Adresse interpretiert werden können, sind Status- oder Fehlermeldungen des Gerätes.

Sie haben folgende Bedeutung:

Anzeige	Status- oder Fehlermeldungen des Gerätes
39	Erweiterte ASi Diagnose, Fehler wird nach dem Drücken der <Set>-Taste angezeigt: <ul style="list-style-type: none"> es ist ein kurzzeitiger Spannungszusammenbruch auf ASi aufgetreten.
40	Der ASi Master befindet sich in der Offline-Phase.
41	Der ASi Master befindet sich in der Erkennungsphase.
42	Der ASi Master befindet sich in der Aktivierungsphase.
43	Der ASi Master beginnt den Normalbetrieb.
68	Hardwarefehler: <ul style="list-style-type: none"> gestörte interne Kommunikation.
69	Hardwarefehler: <ul style="list-style-type: none"> gestörte interne Kommunikation.
70	Hardwarefehler: <ul style="list-style-type: none"> Das EEPROM des Masters kann nicht geschrieben werden.
71	Falscher PIC-Typ
72	Hardwarefehler: <ul style="list-style-type: none"> Falscher PIC-Prozessor.
73	Hardwarefehler: <ul style="list-style-type: none"> Falscher PIC-Prozessor.
74	Prüfsummenfehler im EEPROM.
75	Fehler im internen RAM.
76	Fehler im externen RAM.
77	ASi Control-Softwarefehler: <ul style="list-style-type: none"> Stack overflow (ASi Control II).

Anzeige	Status- oder Fehlermeldungen des Gerätes
78	<u>ASi Control-Softwarefehler:</u> <ul style="list-style-type: none"> • Prüfsummenfehler im Steuerprogramm.
	<u>"control checksum":</u> <ul style="list-style-type: none"> • Die Checksumme des Control III C-Programms (bin.File) ist nicht korrekt. Eventuell ist die Datei beschädigt.
	<u>"control exec err":</u> <ul style="list-style-type: none"> • Fehler im Control III C-Programm.
	<u>"control watchdog":</u> <ul style="list-style-type: none"> • Der im Control III C-Programm definierte Watchdog ist abgelaufen.
	<u>"control incomp":</u> <ul style="list-style-type: none"> • Control III C-Programm von einem anderen Gateway Typ geladen (z.B. EtherNet IP in Profibus Gateway).
	<u>"function not supported"</u> <ul style="list-style-type: none"> • Die verwendete Funktion wird nicht unterstützt.
79	Prüfsummenfehler bei den Menüdaten: <u>"breakpoint":</u> <ul style="list-style-type: none"> • Control III C-Programm steht im Breakpoint.
80	Fehler beim Verlassen des Projektierungsmodus: <ul style="list-style-type: none"> • Es existiert ein ASi Teilnehmer mit Adresse Null.
81	Allgemeiner Fehler beim Ändern einer ASi Adresse.
82	Die Tastenbedienung wurde gesperrt. Bis zum nächsten Neustart des Masters sind Zugriffe auf das Gerät nur vom Host aus über die Schnittstelle möglich.
83	Programm-Reset des ASi Control-Programms: <ul style="list-style-type: none"> • Das ASi Kontrollprogramm wird gerade aus dem EEPROM ausgelesen und ins RAM kopiert.
88	Anzeigentest beim Anlaufen des ASi Masters.
90	Fehler beim Ändern einer ASi Adresse im geschützten Betriebsmodus: <ul style="list-style-type: none"> • Es existiert kein ASi Teilnehmer mit der Adresse '0'.
91	Fehler beim Ändern einer ASi Adresse: <ul style="list-style-type: none"> • Die Zieladresse ist bereits belegt.
92	Fehler beim Ändern einer ASi Adresse: <ul style="list-style-type: none"> • Die neue Adresse konnte nicht gesetzt werden.
93	Fehler beim Ändern einer ASi Adresse: <ul style="list-style-type: none"> • Die neue Adresse konnte im ASi Teilnehmer nur flüchtig gespeichert werden.

Anzeige	Status- oder Fehlermeldungen des Gerätes
94	<p>Fehler beim Ändern einer ASi Adresse im geschützten Betriebsmodus:</p> <ul style="list-style-type: none"> Der ASi Teilnehmer hat falsche Konfigurationsdaten.
95	<p>Die "95" wird angezeigt, wenn der Fehler nicht ein fehlender ASi Teilnehmer, sondern ein ASi Teilnehmer zu viel war. Dadurch ist die Zieladresse durch den überzähligen ASi Teilnehmer belegt.</p> <p>Im geschützten Betriebsmodus kann man durch Drücken der <Set>-Taste alle ASi Adressen anzeigen, die für einen Konfigurationsfehler verantwortlich sind. ASi Master ohne grafisches Display unterscheiden nicht zwischen einem fehlenden ASi Teilnehmer, einem falschen ASi Teilnehmer oder einem ASi Teilnehmer zu viel. Alle fehlerhaften Adressen werden angezeigt. Drückt man die <Set>-Taste 5 Sek., fängt die Adresse an, zu blinken. Ein erneuter Druck versucht, den ASi Teilnehmer, der sich auf der Adresse '0' befindet, auf die fehlerhafte Adresse zu programmieren.</p>



**Bihl
+ Wiedemann**