

Dezentrale Automation

LU 183.169

ASi Library

V1.1

Dieter Etz

Matr. Nr. 0327813

Kennzahl 086881

dieter@auto.tuwien.ac.at

Wien, am 18. März 2007

Inhaltsverzeichnis

1	Einleitung	1
2	Tilab-Targetsystem ASi-Komponenten	1
2.1	ASi-Slaves auf dem Tilab-Targetsystem	1
2.1.1	Slave 1 (Digital IO)	1
2.1.2	Slave 2 (2-Kanal analog Output)	1
2.1.3	Slave 3 (2-Kanal analog Input)	1
2.1.4	Slave 4 (Digital IO)	1
2.1.5	Slave 5 (2-Kanal analog Input)	2
2.1.6	Slave 6 (Digital I)	2
2.1.7	Slave 7 (AirBox)	2
2.1.8	Slave 8 (Induktiver Näherungsschalter)	2
2.1.9	Slave 9 (Sonar)	2
3	Verwenden der Library	3
3.1	Einbindung in Step7-Micro/WIN	3
3.2	Memorylayout der Library	3
3.2.1	User-Space	4
3.2.2	Library-Space	5
3.3	Unterprogramme und Flags	7
3.3.1	ASi_init	7
3.3.2	ASi_main	7
3.3.3	getPressure	7
3.3.4	getDistance	7
3.3.5	getJoystick_X	7
3.3.6	getJoystick_Y	7
3.3.7	setMCSpeed	7
3.3.8	getOpt_NS	8
3.3.9	getTaster	8
3.3.10	setLamps	8
3.3.11	Flags	8
3.4	Demoapplikation	9
4	Funktionsweise der Library	10
4.1	Digitale ASi-Slaves	11
4.2	Analoge ASi-Slaves	11
	Literatur	11

History

Lib-Version	Datum	Änderungen
v1.0	05.01.2007	Initial Release
v1.1	18.03.2007	Automatic Bankselect Die Bankeinstellung wird jetzt in asi_main vorgenommen und danach wieder auf die ursprüngliche Einstellung zurückgesetzt, damit werden Kollisionen mit anderen Bankeinstellungen (z.B. Profibus-DP) verhindert.

1 Einleitung

Diese ASi-Library hat den Zweck die Komponenten des Tilab-Targetsystems über ein einfaches API ansprechen zu können. Abhängig von der verwendeten CPU kann man parametrisierte Unterprogrammaufrufe verwenden oder aber, wie bei der CPU214, die ASi-Informationen direkt über einen definierten Variablenspeicherbereich beziehen.

2 Tilab-Targetsystem ASi-Komponenten

2.1 ASi-Slaves auf dem Tilab-Targetsystem

2.1.1 Slave 1 (Digital IO)

Auf diesem ASi-Slave werden folgende Größen ausgegeben/eingelese:

4x dig. In: 4 Taster
4x dig. Out: 4 Lampen

Das Lesen bzw. Schreiben der Ein- bzw. Ausgänge erfolgt direkt über das ASi-Input-Word AEW0 bzw. ASi-Output-Word AAW0.

2.1.2 Slave 2 (2-Kanal analog Output)

Auf diesem ASi-Slave werden folgende Größen ausgegeben:

Kanal 0: Digital-Voltmeter (0 .. 10V)
Kanal 1: Motor-Speed (0 .. 8 mm/s)

Das Schreiben auf die beiden Kanäle bzw. die Implementierung des ASi-Protokolls wird im Unterprogramm "write_DVM_MCSpeed" bewerkstelligt. In diesem Unterprogramm wird abwechselnd die Variable DVM auf Kanal 0 und MCSPEED auf Kanal 1 geschrieben.

2.1.3 Slave 3 (2-Kanal analog Input)

Auf diesem ASi-Slave werden folgende Größen eingelese:

Kanal 0: Joystick left/right (-0.5V .. 0.5V)
Kanal 1: Joystick up/down (-0.5V .. 0.5V)

Das Einlesen der beiden Kanäle bzw. die Implementierung des ASi-Protokolls wird im Unterprogramm "read_joystick" bewerkstelligt. In diesem Unterprogramm wird abwechselnd der Kanal 0 in JOY_X und Kanal 1 in JOY_Y eingelese.

2.1.4 Slave 4 (Digital IO)

Auf diesem ASi-Slave werden folgende Größen ausgegeben/eingelese:

4x dig. In: OptNS, KapNS, EinLS
4x dig. Out: Motorcontroller Go und Dir

Das Lesen bzw. Schreiben der Ein- bzw. Ausgänge erfolgt direkt über das ASi-Input-Word AEW2 bzw. ASi-Output-Word AAW2.

2.1.5 Slave 5 (2-Kanal analog Input)

Auf diesem ASi-Slave werden folgende Grössen eingelesen:

- Kanal 0: Druck (0 .. 10bar)
- Kanal 1: Distanz (4 - 20mA)

Das Einlesen der beiden Kanäle bzw. die Implementierung des ASi-Protokolls wird im Unterprogramm "read_druck_dist" bewerkstelligt. In diesem Unterprogramm wird abwechselnd der Kanal 0 in DRUCK und Kanal 1 in DIST eingelesen.

2.1.6 Slave 6 (Digital I)

Auf diesem ASi-Slave werden folgende Grössen eingelesen:

- 4x dig. In: MagE_D, MagE_U, GlasLS

Das Lesen der Eingänge erfolgt direkt über das ASi-Input-Word AEW2.

2.1.7 Slave 7 (AirBox)

Auf diesem ASi-Slave werden folgende Grössen ausgegeben/eingelesen:

- 2x dig. In: Zylinder Endschalter oben und unten
- 2x dig. Out: Zylinder Up und Down

Das Lesen bzw. Schreiben der Ein- bzw. Ausgänge erfolgt direkt über das ASi-Input-Word AEW2 bzw. ASi-Output-Word AAW2.

2.1.8 Slave 8 (Induktiver Näherungsschalter)

Der Sensor stellt seine Information über 3 Bits bereit:

- D0: Schaltsignal (0 ... Gegenstand weg)
- D1: Schaltsignal valid (0 ... invalid)
- D2: Betriebszustand (0 ... Fehlerfall)

Das Lesen der Bits erfolgt direkt über das ASi-Input-Word AEW4.

2.1.9 Slave 9 (Sonar)

Der Sensor stellt seine Information über 3 Bits bereit:

- D0: Gegenstand 55-125mm vor Stirnfläche
- D1: Gegenstand 125-220mm vor Stirnfläche
- D2: Gegenstand 220-300mm vor Stirnfläche

Das Lesen der Bits erfolgt direkt über das ASi-Input-Word AEW4.

3 Verwenden der Library

Um die Library verwenden zu können sind folgende Schritte notwendig:

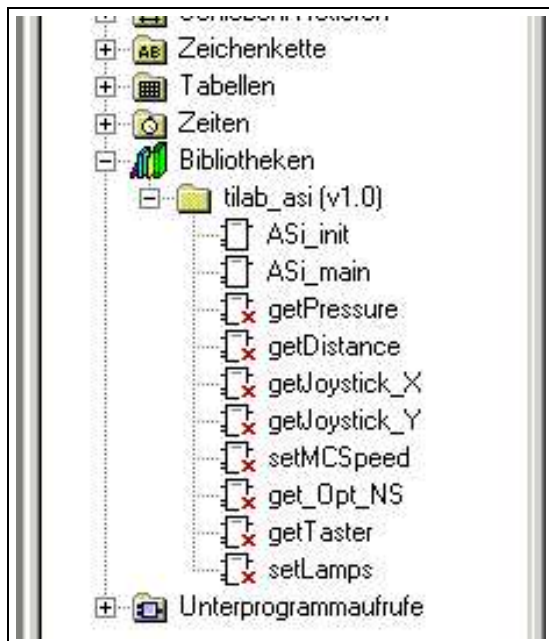
- Einbinden der Bibliothek in Step7-Micro/WIN
- Verwenden der bereitgestellten Unterprogramme bzw. Zugriff auf den Speicherbereich der Bibliothek

3.1 Einbindung in Step7-Micro/WIN

In Step7-Micro/WIN wird die Bibliothek mittels

Datei → Bibliothek ergänzen/entfernen...

hinzugefügt. Danach erscheint im Operationsbaum unter Bibliotheken die tilab_asi Bibliothek mit ihren Unterprogrammen.



Anmerkung: Wie im Bild zu sehen ist, sind einige Unterprogrammaufrufe mit einem roten Kreuz gekennzeichnet, dies bedeutet daß diese Aufrufe auf der verwendeten CPU214 nicht unterstützt werden. Ab einer CPU 221 kann man diese Aufrufe verwenden.

Danach muß der Bibliothek ein Speicherbereich zugewiesen werden, dies wird mittels

Datei → Speicher für Bibliothek...

durchgeführt.

3.2 Memorylayout der Library

Der verwendete Speicher der Library gliedert sich in zwei Bereiche

User-Space dieser Bereich stellt die Informationen für den User bereit

Library-Space dieser Bereich wird ausschliesslich von der Library verwendet

Insgesamt benötigt die Library 48 Bytes Variablenspeicher, davon sind die ersten 20 Bytes als User-Interface und die restlichen 28 Bytes als library-interner Speicher vorgesehen.

3.2.1 User-Space

Die ersten 3 D-Words des Speichers (VD0 .. VD8) bilden alle Eingänge ab, die folgenden 2 D-Words (VD12 .. VD16) bilden alle Ausgänge ab.

VD0											
VW0				VB1				VB2			
VB0				VB1				VB2			
V0.7	V0.6	V0.5	V0.4	V0.3	V0.2	V0.1	V0.0	V1.7	V1.6	V1.5	V1.4
								V1.3	V1.2	V1.1	V1.0
_Opt_NS				_Tast_3				_MagE_U			
_Kap_NS				_Tast_2				_MagE_D			
_Ein_LS				_Tast_1				_Glas_LS			
_DigIn_Reg0_B				_Tast_0				_ZEnd_Up			
								_ZEnd_Do			
				_DigIn_Reg1_B				_IndNS_2			
								_IndNS_1			
								_IndNS_0			
				_DigIn_Reg2_B				_Sonar_2			
								_Sonar_1			
								_Sonar_0			
VD4											
VW4				VB5				VB6			
VB4				VB5				VB6			
V4.7	V4.6	V4.5	V4.4	V4.3	V4.2	V4.1	V4.0	V5.7	V5.6	V5.5	V5.4
								V5.3	V5.2	V5.1	V5.0
_press_sign				_press_bit12				_press_bit8			
				_press_bit11				_press_bit7			
				_press_bit10				_press_bit6			
				_press_bit9				_press_bit5			
								_press_bit4			
								_press_bit3			
								_press_bit2			
								_press_bit1			
				_dist_sign				_dist_bit12			
								_dist_bit11			
								_dist_bit10			
								_dist_bit9			
								_dist_bit8			
								_dist_bit7			
								_dist_bit6			
								_dist_bit5			
								_dist_bit4			
								_dist_bit3			
								_dist_bit2			
								_dist_bit1			
VD8											
VW8				VB9				VB10			
VB8				VB9				VB10			
V8.7	V8.6	V8.5	V8.4	V8.3	V8.2	V8.1	V8.0	V9.7	V9.6	V9.5	V9.4
								V9.3	V9.2	V9.1	V9.0
_joy_x_sign				_joy_x_bit12				_joy_x_bit8			
				_joy_x_bit11				_joy_x_bit7			
				_joy_x_bit10				_joy_x_bit6			
				_joy_x_bit9				_joy_x_bit5			
								_joy_x_bit4			
								_joy_x_bit3			
								_joy_x_bit2			
								_joy_x_bit1			
				_joy_y_sign				_joy_y_bit12			
								_joy_y_bit11			
								_joy_y_bit10			
								_joy_y_bit9			
								_joy_y_bit8			
								_joy_y_bit7			
								_joy_y_bit6			
								_joy_y_bit5			
								_joy_y_bit4			
								_joy_y_bit3			
								_joy_y_bit2			
								_joy_y_bit1			
VD12											
VW12				VB13				VB14			
VB12				VB13				VB14			
V12.7	V12.6	V12.5	V12.4	V12.3	V12.2	V12.1	V12.0	V13.7	V13.6	V13.5	V13.4
								V13.3	V13.2	V13.1	V13.0
_S5_enable											
_S3_enable											
_S2_enable											
VD16											

VD32			VD32		
VW32			VW34		
VB32	V32.7	_S2_wr_temp _S2_wr_actChannel _S2_wr_chReq _S2_wr_bit3Old _S2_wr_init	VB33	V32.6	VB34
	V32.6			V32.5	
	V32.4			V32.3	
	V32.2			V32.1	
VB33	V33.7	_S2_wr_wrBit3 _S2_wr_wrBit2 _S2_wr_wrBit1 _S2_wr_wrBit0	VB34	V33.6	VB35
	V33.6			V33.5	
	V33.4			V33.3	
	V33.2			V33.1	
VB34	V33.0			V33.0	
VB35	V34.7	_S2_wr_data_d12 _S2_wr_data_d11 _S2_wr_data_d10 _S2_wr_data_d9	VB36	V34.6	VB37
	V34.6			V34.5	
	V34.4			V34.3	
	V34.2			V34.1	
VB36	V34.0			V34.0	
VB37	V35.7	_S2_wr_data_d8 _S2_wr_data_d7 _S2_wr_data_d6 _S2_wr_data_d5	VB38	V35.6	VB39
	V35.6			V35.5	
	V35.4			V35.3	
	V35.2			V35.1	
VB38	V35.0			V35.0	

VD36			VD36		
VW36			VW38		
VB36	V36.7	_S3_rd_actChannel _S3_rd_chOK _S3_rd_bit3Old _S3_rd_init	VB37	V36.6	VB38
	V36.6			V36.5	
	V36.4			V36.3	
	V36.3			V36.2	
VB37	V36.1		VB38	V36.0	VB39
	V37.7			V37.6	
	V37.6			V37.5	
	V37.4			V37.3	
VB38	V37.2		VB39	V37.1	VB40
	V37.0			V37.0	
	V38.7			V38.6	
	V38.6			V38.5	
VB39	V38.4		VB40	V38.3	VB41
	V38.3			V38.2	
	V38.2			V38.1	
	V38.0			V38.0	
VB40	V39.7	_S3_rd_signed _S3_rd_overfbw _S3_rd_valid	VB41	V39.6	VB42
	V39.6			V39.5	
	V39.4			V39.3	
	V39.2			V39.1	
VB41	V39.0		VB42	V39.0	

VD40			VD40		
VW40			VW42		
VB40	V40.7	_S3_rd_value_sign _S3_rd_data_d12 _S3_rd_data_d11 _S3_rd_data_d10 _S3_rd_data_d9	VB41	V40.6	VB42
	V40.6			V40.5	
	V40.4			V40.3	
	V40.3			V40.2	
VB41	V40.1		VB42	V40.0	VB43
	V41.7			V41.6	
	V41.6			V41.5	
	V41.4			V41.3	
VB42	V41.2		VB43	V41.1	VB44
	V41.0			V41.0	
	V42.7			V42.6	
	V42.6			V42.5	
VB43	V42.4		VB44	V42.3	VB45
	V42.3			V42.2	
	V42.2			V42.1	
	V42.0			V42.0	
VB44	V43.7	_S5_rd_actChannel _S5_rd_chOK _S5_rd_bit3Old _S5_rd_init	VB45	V43.6	VB46
	V43.6			V43.5	
	V43.4			V43.3	
	V43.2			V43.1	
VB45	V43.0		VB46	V43.0	

VD44			VD44		
VW44			VW46		
VB44	V44.7	_S5_rd_signed _S5_rd_overfbw _S5_rd_valid	VB45	V44.6	VB46
	V44.6			V44.5	
	V44.4			V44.3	
	V44.3			V44.2	
VB45	V44.1		VB46	V44.0	VB47
	V45.7			V45.6	
	V45.6			V45.5	
	V45.4			V45.3	
VB46	V45.2		VB47	V45.1	VB48
	V45.0			V45.0	
	V46.7			V46.6	
	V46.6			V46.5	
VB47	V46.4	_S5_rd_value_sign _S5_rd_data_d12 _S5_rd_data_d11 _S5_rd_data_d10 _S5_rd_data_d9	VB48	V46.3	VB49
	V46.3			V46.2	
	V46.2			V46.1	
	V46.0			V46.0	
VB48	V47.7		VB49	V47.6	VB50
	V47.6			V47.5	
	V47.4			V47.3	
	V47.3			V47.2	
VB49	V47.2		VB50	V47.1	VB51
	V47.1			V47.0	
	V47.0			V47.0	
	V47.0			V47.0	

3.3 Unterprogramme und Flags

3.3.1 ASi_init

Zum Initialisieren der Library muß im ersten Zyklus genau einmal das Unterprogramm ASi_init aufgerufen werden.

Parameter: keine

3.3.2 ASi_main

Das Unterprogramm ASi_main muss in jedem Zyklus einmal aufgerufen werden.

Parameter: keine

3.3.3 getPressure

Liefert den Wert des Drucksensors von Slave 5.

Parameter:

Pressure OUT WORD

3.3.4 getDistance

Liefert den Wert des Distanzsensors von Slave 5.

Parameter:

Distance OUT WORD

3.3.5 getJoystick_X

Liefert die X-Position des Joysticks (-0.5 .. +0.5V).

Parameter:

Joy_X OUT WORD

3.3.6 getJoystick_Y

Liefert die Y-Position des Joysticks (-0.5 .. +0.5V).

Parameter:

Joy_Y OUT WORD

3.3.7 setMCSpeed

Setzt den Wert für die Schlittengeschwindigkeit.

Parameter:

MCSpeed IN WORD

3.3.8 getOpt_NS

Liefert den Zustand des optischen Näherungssensors.

Parameter:

Opt_NS OUT BOOL

3.3.9 getTaster

Liefert den Zustand der 4 Taster von Slave 1.

Parameter:

Tast_0 OUT BOOL
Tast_1 OUT BOOL
Tast_2 OUT BOOL
Tast_3 OUT BOOL

3.3.10 setLamps

Setzt den Zustand der 4 Lampen von Slave 1.

Parameter:

Lampe_0 IN BOOL
Lampe_1 IN BOOL
Lampe_2 IN BOOL
Lampe_3 IN BOOL

3.3.11 Flags

Im Gegensatz zu den digitalen ASi-Slaves, bei denen die Werte direkt eingelesen bzw. gesetzt werden können, muß bei den analogen ASi-Slaves ein Protokoll abgearbeitet werden. Um dieses Protokoll nur bei benötigten Slaves abzuarbeiten werden für diese Slaves Enable-Flags verwendet. Um einen analogen ASi-Slave verwenden zu können muß man deshalb das korrespondierende Flag im Output-Register setzen.

Folgende Flags sind verfügbar:

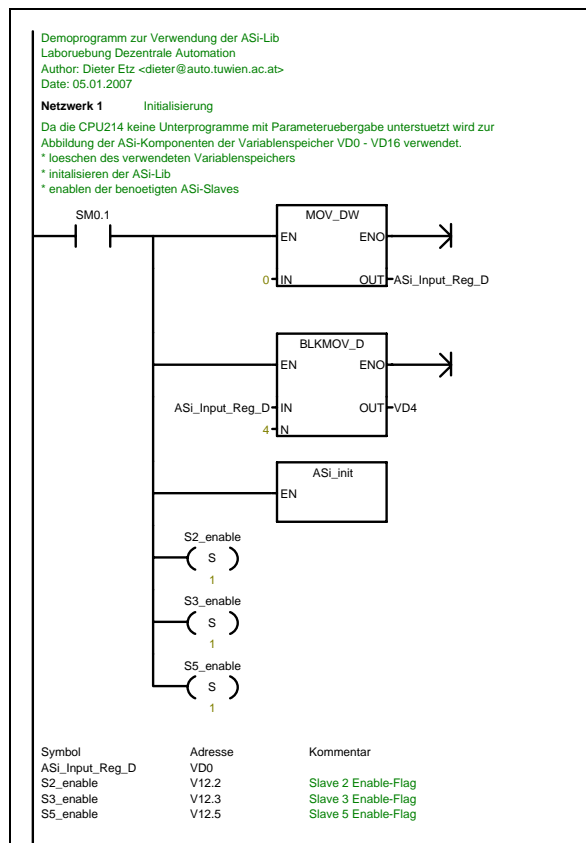
Flag	Adresse
S2_enable	V12.2
S3_enable	V12.3
S5_enable	V12.5

3.4 Demoapplikation

Da die CPU214 keine parametrisierten Unterprogrammaufrufe unterstützt, muß man den Umweg über den Zugriff auf den Library-Speicher nehmen. Die folgende Demo-Applikation zeigt, wie man die Library auf einer CPU214 verwenden kann.

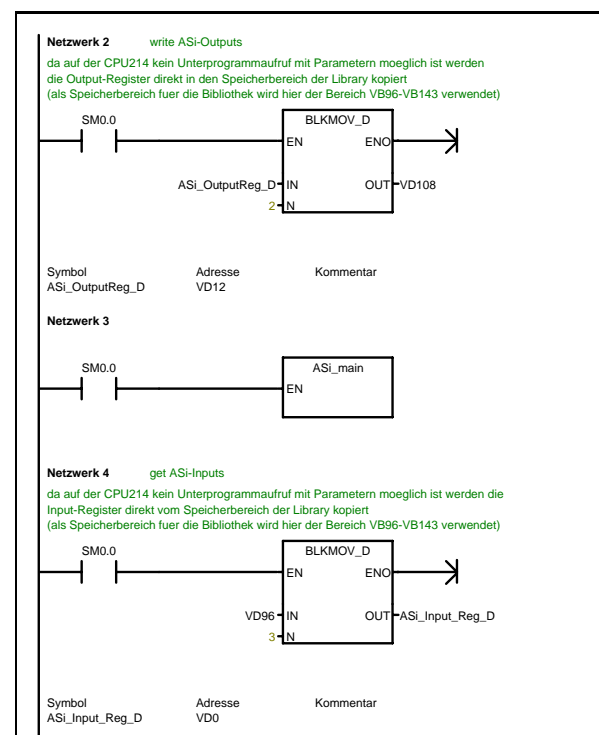
Bei dieser Applikation wurde als Speicher für die Library der Bereich VB96-VB143 festgelegt.

Symbol	Adresse	Kommentar
ASi_Input_Reg_D	VD0	Input Register
Opt_NS	V0.7	optischer Näherungsschalter
Kap_NS	V0.6	kapazitiver Näherungsschalter
Ein_LS	V0.4	Ein Lichtschränke
Tast_3	V0.3	Taster 3
Tast_2	V0.2	Taster 2
Tast_1	V0.1	Taster 1
Tast_0	V0.0	Taster 0
MagE_U	V1.6	magnetischer Endschalter oben
MagE_D	V1.5	magnetischer Endschalter unten
Glas_LS	V1.4	Glas Lichtschränke
ZEnd_Up	V1.1	Endschalter Zylinder oben
ZEnd_Do	V1.0	Endschalter Zylinder unten
IndNS_2	V2.6	induktiver Näherungsschalter Bit2
IndNS_1	V2.5	induktiver Näherungsschalter Bit1
IndNS_0	V2.4	induktiver Näherungsschalter Bit0
Sonar_2	V2.2	Sonar Bit2
Sonar_1	V2.1	Sonar Bit1
Sonar_0	V2.0	Sonar Bit0
Pressure	VW4	Zylinderdruck
Distance	VW6	Distanz
Joy_X	VW8	Joystick X-Achse
Joy_Y	VW10	Joystick Y-Achse
ASi_OutputReg_D	VD12	Output Register
S2_enable	V12.2	Slave 2 Enable-Flag
S3_enable	V12.3	Slave 3 Enable-Flag
S5_enable	V12.5	Slave 5 Enable-Flag
MC_Dir	V15.7	Motorcontroller Direction
MC_Go	V15.6	Motorcontroller Go
Zyl_Up	V15.5	Zylinder Auf
Zyl_Do	V15.4	Zylinder Ab
Lampe_3	V15.3	Lampe 3
Lampe_2	V15.2	Lampe 2
Lampe_1	V15.1	Lampe 1
Lampe_0	V15.0	Lampe 0
DVM	VW16	Digital Voltmeter
MCSpeed	VW18	Motorspeed



Die Symboltabelle verwendet die ersten 20 Bytes des Variablenspeichers für ein lokales Abbild der ASi-Variablen.

In Netzwerk 1 wird der verwendete Variablenspeicher initialisiert und mittels ASi_init die Library initialisiert. Weiters werden die Slave-enable Flags gesetzt damit die Library nur die benötigten Slaves abarbeitet.



In Netzwerk 2 werden die Ausgangsvariablen des lokalen ASi-Registers in die Library-Register (Startadresse: VD96 + 12) kopiert, danach wird ASi_main aufgerufen und zuletzt die Eingangsvariablen von der Library (Startadresse: VD96 + 0) ins lokale Register kopiert. Ein direkter Zugriff auf die Variablen bzw. Symbole der Library ist nicht möglich.

Danach erfolgt die Implementierung des User-Programms welches dann über die eigene Symboltabelle Zugriff auf alle ASi-Komponenten hat.

4 Funktionsweise der Library

Um auf die EA-Daten der ASi-Slaves zugreifen zu können muß einerseits das Bit PLC_RUN im Steuerbyte gesetzt sein und andererseits die richtige Bank ausgewählt sein. Die Bank kann durch die Bits BS0 .. BS3 im Steuerbyte ausgewählt werden. In der Library wird das PLC_RUN-Bit in asi_init gesetzt. Die Bankauswahl ist in asi_main implementiert.

Zu Begin jedes Zyklus wird Bank 0 ausgewählt (die zuvor eingestellte Bank wird gespeichert) und die ASi-Input-Wörter AEW0 .. AEW4 in die library-internen Shadow-Register VW20 .. VW24 eingelesen damit die Library beliebigen Zugriff hat.

Eingangsdaten

AEW0															
MSB												LSB			
reserviert				Slave1 LAMPETAST_0-3				Slave2				Slave3 ASiUIN			
				TAST_3	TAST_2	TAST_1	TAST_0								
D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

AEW2															
MSB												LSB			
Slave4 ASiDIO				Slave5 ASiIIN				Slave6 ASiDIN				Slave7 ASiAIR			
OPTNS	KAPNS		EINLS					MAGE_U	MAGE_D	GLASLS		ZENDUP	ZENDDO		
D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

AEW4															
MSB												LSB			
Slave8 INDNS				Slave9 SONAR				Slave10				Slave11			
				SONAR_2	SONAR_1	SONAR_0									
D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Ausgangsdaten

AAW0															
MSB												LSB			
reserviert				Slave1 LAMPETAST_0-3				Slave2 ASiUOUT				Slave3			
				LAMPE_3	LAMPE_2	LAMPE_1	LAMPE_0								
D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

AAW2															
MSB												LSB			
Slave4 ASiDIO				Slave5				Slave6				Slave7 ASiAIR			
MCDIR	MCGO											ZYLUP	ZYLDO		
D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0	D3	D2	D1	D0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Am Ende jedes Zyklus werden die library-internen Shadow-Register VW28 .. VW30 auf die ASi-Output-Wörter AAW0 .. AAW2 geschrieben. Danach wird die zuvor eingestellte Bankauswahl wieder hergestellt.

Grundsätzlich muß man bei den ASi-Slaves zwei Arten unterscheiden die sich in Bezug auf das Lesen und Schreiben von Sensordaten wesentlich unterscheiden:

- Digitale ASi-Slaves
- Analoge ASi-Slaves

4.1 Digitale ASi-Slaves

Der Zugriff auf Daten der digitalen ASi-Slaves erfolgt durch lesen bzw. schreiben der entsprechenden Bits in den ASi-Input-Wörtern bzw. ASi-Output-Wörtern. Diese Bits werden einfach in den User-Space-Bereich VD0 (Input) und VD12 (Output) gemappt.

4.2 Analoge ASi-Slaves

Der Zugriff auf die Werte der analogen ASi-Slaves benötigt ein Protokoll da die Daten mittels der zur Verfügung stehenden 4 Bits in den ASi-Daten nicht direkt übertragen werden können.

Um einen Analogwert übertragen zu können werden 6 Zyklen benötigt:

Zyklus	Daten
1	Kanalnummer
2	4.Datentripel
3	3.Datentripel
4	2.Datentripel
5	1.Datentripel
6	SOV-Tripel

Da das Protokoll immer vollständig und für beide Känale ausgeführt werden muß, liest die Library immer alle Werte eines Input-Slaves ein bzw. beschreibt immer alle Kanäle eines Output-Slaves.

Damit das Protokoll nur bei benötigten Slaves abgearbeitet wird steht ein Enable-Flag in VB12 für jeden Analog-Slave zur Verfügung. Diese Flags sind initial abgeschaltet, der Benutzer muß diese Flags einschalten um Analogwerte der entsprechenden Slaves lesen bzw. schreiben zu können.

Literatur

- [SYS-S7] Systemhandbuch SIMATIC Automatisierungssystem S7-200, Siemens, 1998
- [SYS-CP242] Systemhandbuch CP242-8, AS-Interface Master / PROFIBUS-DP Slave, Ausgabe 01, Siemens, 1998
- [ASi-VO06] Vorlesungsunterlagen "Einführung in die Automation", Aktuator-Sensor-Interface, Wolfgang Kastner, 2006
- [ProzAut04] Targetbeschreibung, Laborübung Prozeßautomatisierung Targetsysteme, Schmid, Kastner, Kastner-Masilko, 2004