



POWERLINK SET UP GUIDE

B&R AUTOMATION STUDIO

RICCARDO PICCININI – ELKER CAVINA

SOMMARIO

OVERVIEW AND PROCESS DATA.....	2
Overview.....	2
DS301 Communication layer	2
Physical layer	2
SDO protocol	3
TIME protocol.....	3
PDO protocol	3
EMCY protocol.....	3
NMT protocol	4
HEARTBEATING	4
Object dictionary	4
Communication area	4
Manufacturer parameter area	5
Manufacturer specific area	6
Profiled objects	8
APPLICATION EXAMPLE	10
TOOLS	10
OVERVIEW	10
DBS IMPORT	10
PROFILE.....	11
Initial Steps	11
Sample application	12
Code	14
CSP	0
Initial steps	0
Add Axis to the NDC	1
SDC – DS402 Connection.....	4
Code	5
Sample Application.....	7

ETHERNET POWERLINK

OVERVIEW AND PROCESS DATA

OVERVIEW

The present document describes the adherence of the Minimotor stack to the following specification:

CanOPEN

- CiA DS301 v4.02 (CanOPEN DLL)
- CiA DS402 v3.0.1.15 (servodrive profile)

CanOPEN over EtherCAT:

- ETG1000-2 v1.0.1 (physical layer)
- ETG1000-3 v1.0.1 (DLL services)
- ETG1000-4 v1.0.1 (DLL protocols)
- ETG1000-5 v1.0.1 (AL services)
- ETG1000-6 v1.0.1 (AL protocols)
- CiA DS402 v3.0.1.15 (servodrive)
- ETG6010 v1.1.0 (CiA 402 impl. directives)

Ethernet PowerLink (EPL)

- EPSG DS 301 V1.2.0 (Powerlink DLL)

The purpose of the present document is to describe what optional feature are implemented by the servo-motor CanOPEN, CoE or EPL.

***** Applicable DBS55/---/--- software release: ≥v4.036 *****

DS301 COMMUNICATION LAYER

This paragraph describe the details of the DS301 layer of the CanOpen interface.
It applies only to the CanOPEN over can-bus implementation.

PHYSICAL LAYER

Supported baudrate:

- 1Mbps tq=71ns SamplingPoint=11tq

- 500Kbps tq=142ns SamplingPoint=12tq
- 250Kbps tq=284ns SamplingPoint=12tq
- 125Kbps tq=571ns SamplingPoint=12tq
- 50Kbps tq=1.43us SamplingPoint=12tq
- 20Kbps tq=3.57us SamplingPoint=12tq

SDO PROTOCOL

- Maximum supported object size: 32bit
- Eexpedited transfer supported SUPPORTED
- Segmented transfer: NOT SUPPORTED
- Block transfer: NOT SUPPORTED

TIME PROTOCOL

The time protocol is NOT supported.

PDO PROTOCOL

The implementation supports up to 4 Pdo TX and 4 Pdo RX; each object can map up to 8 objects; mapping can be done only using the whole size of the object (i.e. is not possible to map 8 bit of a 32 bit object).

The following transmit type are implemented for PDO TX:

Trasmission type	
0	PDO is emitted when changes after SYNC is received
1..240	PDO emitted after SYNC, depending on SYNC message occurrence (1 every sync, 2 every 2 sync etc.).
241..251	NOT supported
252..253	PDO is emitted after RTR reception: • 252: data sampled on SYNC, emitted after RTR • 253: data sampled and emitted after RTR
254	NOT supported
255	PDO is emitted asynchronously; two feature are supported for emission control: • inhibit time for emission only when PDO content changes • event timer for periodical (asynchronous) emission

Defaults:

- inhibit time: 100ms
- event time: 0 (periodic emission on TT=255 disabled by default)

See application profile for default PDO configuration. (See paragraph **Errore. L'origine riferimento non è stata trovata.**)

EMCY PROTOCOL

Emergency protocol is implemented for reporting alarms from the servodrive; to control the emission of EMCY messages, the inhibit time (object 1015h) can be modified.

See the application profile for detailed list of alarms. (See paragraph **Errore. L'origine riferimento non è stata trovata.**)

NMT PROTOCOL

The NMT implementation includes standard commands for NMT state machine management.

For node alive checking, the NODE GUARDING protocol is implemented; the behaviour of node guarding is set by objects 100Ch (guard time) and 100Dh (life factor).

HEARTBEATING

Heartbeating Protocol is supported. Register 1016h defines the time in multiples of 1ms for the Heartbeat protocol consumer. Register 1017h determines the time in multiples of 1ms for the producer of the protocol. Object 1029h determines the behaviour when the error is triggered.

OBJECT DICTIONARY

Object dictionary is divided in the following sections:

Range oggetti	Funzioni
1000h ... 1FFFh	Communication profile area
2000h ... 21FFh	Manufacturer parameter area
3000h ... 3FFFh	Manufacturer specific object area
6000h ... 67FFh	Profiled objects

Other areas are unused; the object list is common between CanOPEN, CoE and EPL implementation.

COMMUNICATION AREA

The following table shows the list of communication area objects; since some object are relevant only for CanOPEN or CoE implementation, the table shows in which fieldbus the object is relevant.

Index	Description	CanOPEN	CoE	EPL
1000	Device type	X	X	X
1001	Error register	X	X	X
1003	Predefined error	X	X	
1006	Cycle length			X
100C	Guard time	X		
100D	Life factor	X		
1010	Store parameters (*1)	X	X	X
1014	EMCY cob-id	X		
1015	EMCY inhibit time	X	X	
1016	Consumer Heartbeat Time	X		
1017	Producer Heartbeat Time	X		
1018	Identity object	X	X	X
1020	CFM Verify Configuration			X
1029	Error Behaviour	X		
1030	NMT Interface Group 0h			X
1300	SDO Sequ Layer Timeout			X
1400	PDO RX 1 – config (*2)	X	X	X

1401	PDO RX 2 – config	X		
1402	PDO RX 3 – config	X		
1403	PDO RX 4 – config	X		
1600	PDO RX 1 – mapping (*2)	X	X	X
1601	PDO RX 2 – mapping	X		
1602	PDO RX 3 – mapping	X		
1603	PDO RX 4 – mapping	X		
1800	PDO TX 1 – config (*2)	X	X	X
1801	PDO TX 2 – config	X		
1802	PDO TX 3 – config	X		
1803	PDO TX 4 – config	X		
1A00	PDO TX 1 – mapping (*2)	X	X	X
1A01	PDO TX 2 – mapping	X		
1A02	PDO TX 3 – mapping	X		
1A03	PDO TX 4 – mapping	X		
1C00	Sync manager types		X	
1C0A	DLL_CnCollision REC			X
1C0B	DLL_CnLoss REC			X
1C0C	DLL_CnLossSoC_REC			X
1C0D	DLL_CnLossSoA REC			X
1C0E	DLL_CnSpCJitter REC			X
1C0F	DLL_CnCRCErrror REC			X
1C10	Sync manager 0 config		X	
1C11	Sync manager 1 config		X	
1C12	Sync manager 2 config		X	
1C13	Sync manager 3 config (*2)		X	
	DLL_CnSocJitterRange (*2)			X
1C14	DLL_CnLossOfSocTolerance			X
1C32	Output sync parameters		X	
1C33	Input sync parameters		X	
1E40	NWL_IpAddrTable_0h_REC			X
1E4A	NWL_IpGroup_REC			X
1F82	NMT_FeatureFlags_U32			X
1F83	NMT_EPLVersion_U8			X
1F8C	NMT_CurrNMTState_U8			X
1F93	NMT_EPLNodeID_REC			X
1F98	NMT_CycleTiming_REC			X
1F99	NMT_CNBasicEthernetTimeout_U32			X
1F9A	NMT_HostName_VSTR			X
1F9E	NMT_ResetCmd_U8			X

(*1) To save 2000h parameters to non-volatile storage a 32bit write to object 1010h.1 needs to be performed; in particular the value to be written to save parameter is 65766173h (“SAVE”).

(*2) PDO config and mapping have different encoding for CoE/CanOPEN and EPL. Please check DLL manual for details.

MANUFACTURER PARAMETER AREA

The object range 2000h ... 21FFh allows to access servodrive parameters; the following table summarize the details of a parameter object:

Object details

Subidx	0
Object type	16bit, signed integer
Access	Read/Write
Pdo mappable	NO

Object dictionary for manufacturer parameters

Index	
2000	Parameter 0
2001	Parameter 1
...	
21FF	Parameter 511

For details of the parameter meaning and encoding refer to the servodrive manual.

NOTE: write objects in the 2000h ... 21FFh range will alter only RAM image of the parameter; to actually save parameter to non-volatile storage a write to object 1010h.0 needs to be performed (see 1010h).

MANUFACTURER SPECIFIC AREA

This area contains several object that allows to read/write manufacturer specific data.

Index		Access	Data type	Pdo mappable
3000	Ain0 value -32768 ... 32768	RO	INT16	YES
3001	Digital input bit0: digital input0 bit1: digital input1 bit2: digital input2 bit3:	RO	UINT16	YES

	digital input3 bit4: digital input4			
3002	Digital output bit0: digital output0	RW	UINT16	YES
3003	Heatsink temperature as tenth of °C (200 >> 20.0°C)	RO	INT16	YES
3005	Motor temperature as tenth of °C (200 >> 20.0°C)	RO	INT16	YES
3008	Actual alarm: 0: no alarm >0: an alarm is pending NOTE: value read from this object is the alarm code as defined in the DBS manual (e.g. OVERVOLTAGE=3 etc) Value IS NOT the EMCY code as defined in this document.	RO	UINT16	YES
3009	Events 0: 16 1-16: event code (see 6.1)	RO	UINT	NO
3010	Power stage tension as tenth of Volt (240 >> 24.0V)	RO	UINT16	YES
3011	Motor IQ current as milliAmpere (+10 >> +1.0A)	RO	INT16	YES
3012	Motor IQ limit as milliampere (10 >> 1.0A)	RW	UINT16	YES
3014	Output Power (1Watt/lsb)	RO	IN16	YES
3020	Touchprobe Positioning offset	RW	INT32	YES
3021	Touchprobe Positioning Modulo	RW	UINT32	YES
3030	Acceleration rms modulus (milli-G)	RO	UINT16	YES
3031	Acceleration rms x (milli-G)	RO	UINT16	YES
3032	Acceleration rms y (milli-G)	RO	UINT16	YES
3033	Acceleration rms z (milli-G)	RO	UINT16	YES
3034	Acceleration dc x (milli-G)	RO	UINT16	YES
3035	Acceleration dc y (milli-G)	RO	UINT16	YES
3036	Acceleration dc z (milli-G)	RO	UINT16	YES
3037	Istantaneous Acceleration x (milli-G)	RO	INT16	YES

3038	Istantaneous Acceleration y (milli-G)	RO	INT16	YES
3039	Istantaneous Acceleration z (milli-G)	RO	INT16	YES

PROFILED OBJECTS

The range 6000h ... 67FFh of objects are defined as specified by CiA402; refer to the application area for further information.

Index		Access	Type	Pdo mappable
6040	Control word	RW	U16	YES
6041	StatusWord	RO	U16	YES
605A	Quickstop option code	RW	U16	
605B	Shutdown option code	RW	U16	
605C	Disable operation option code	RW	U16	
605D	Halt option code	RW	U16	
605E	Fault reaction option code	RW	U16	
6060	Mode of operation	RW	S8	YES
6061	Mode of operation display	RO	S8	YES
6062	Position demand value	RO	S32	YES
6064	Position actual value	RO	S32	YES
6065	Following error window	RW	U32	
6066	Following error time	RW	U16	
6067	Position window	RW	U32	
6068	Position window time	RW	U16	
606B	Velocity demand	RO	S32	YES
606C	Velocity actual value	RO	S32	YES
606D	Velocity window	RW	U16	
606E	Velocity window time	RW	U16	
606F	Velocity threshold	RW	U16	
6070	Velocity threshold time	RW	U16	
6071	Target torque	RW	S16	YES
6072	Max torque	RW	U16	YES
6073	Max current	RW	U16	YES
6074	Torque demand value	RO	S16	YES
6075	Motor rated current	RO	U32	YES
6076	Motor rated torque	RO	U32	YES
6077	Torque actual value	RO	S16	YES
6078	Current actual value	RO	S16	YES
6079	Dclink voltage	RO	U32	YES
607A	Target position	RW	S32	YES
607B	Position Range Limits			
	607D.0: number of sub-idx (2)	RO	U8	
	607D.1: Negative limit	RW	S32	NO
	607D.2: Positive limit	RW	S32	NO
607C	Home offset	RW	S32	YES
607D	Software position limit			
	607D.0: number of sub-idx (2)	RO	U8	

	607D.1: Negative limit	RW	S32	YES
	607D.2: Positive limit	RW	S32	YES
607E	Polarity	RW	U8	
607F	Max speed	RO	U32	YES
6080	Max speed	RO	U32	YES
6081	Profile velocity	RW	U32	YES
6083	Profile acceleration	RW	U32	YES
6084	Profile deceleration	RW	U32	YES
6085	Quick stop deceleration	RW	U32	YES
6087	Torque slope	RW	U32	YES
6098	Homing method	RW	S8	YES
6099	Homing speed			
	6099.0 – number of subidx (2)	RO	U8	
	6099.1 – Switch speed	RW	U32	YES
	6099.2 – Index speed	RW	U32	YES
609A	Homing acceleration	RW	U32	YES
60B0	Position offset	RW	S32	YES
60B1	Velocity offset	RW	S32	YES
60B2	Torque offset	RW	S16	YES
60B8	Touch-probe function	RW	U16	YES
60B9	Touch-probe status	RW	U16	YES
60BA	Touch-probe positive edge latched position	RW	S32	YES
60BB	Touch-probe negative edge latched position	RW	S32	YES
60C0	IpData submode selection	RO	S16	NO
60C1	IpDataRecord			
	60C1.0 – Number of sub-idx (1)	RO	U8	
	60C1.1 – IpData record	RW	S32	YES
60C2	Interpolation time period			
	60C2.0 – number of sub-idx	RO	U8	
	60C2.1 – Ip time units	RW	S8	
	60C2.2 – Ip time index	RW	S8	
60F2	Position Option Code	RW	S16	NO
60FD	Digital inputs	RO	U32	YES
60FE	Digital outputs			
	60FE.0 – number of sub-idx (2)	RO	U8	
	60FE.1 – physical outputs	RW	U32	YES
	60FE.2 – output mask	RW	U32	YES
60FF	Target velocity	RW	S32	YES
6502	Supported mode of operation	RO	U32	NO

APPLICATION EXAMPLE

TOOLS

>> [Powerlink CSP Example](#) <<

>> [Powerlink Profile Example](#) <<

>> [DBS/FC BSI Interface Software](#) <<

OVERVIEW

This document show the creation of a DBS/FC smart motor application using Powerlink communication interface; a B&R controller is configured and an application is written into the Automation Studio development environment.

- B&R PLC 4PPC50 for Profile and 4PPC70 for CSP example
- DBS/FC Firmware ≥ 4.041
- Automation Studio V 4.9.2.46
- DBS and FC systems have the same electronics and firmware, so they follow the same procedure.

DBS IMPORT



The first time using the DBS product, import the XDD descriptor into Automation Studio:

- Tools >> Import Fieldbus device

- Search for MiniMot_DBS55_EPL.xdd in the DBS software distribution and press OK
- In the catalog, under Powerlink devices, the Minimotor DBS device should now be present

Now the next steps depends on how you want to control the motor. As a simple node using the Canopen DS402 Profile Velocity, Torque, Position and Homig or using it in interpolated mode, using as in the example the motor with the ACP10 Subsystem.

PROFILE

INITIAL STEPS

- Create new Project
- Configure Powerlink Cycle time

Name	Value	Unit
IF1		
Module type	Type 4	
Operating mode	POWERLINK V2	
MTU size	300	
Baud rate	100 MBit half duplex	
POWERLINK parameters		
Activate POWERLINK communication	on	
Device name	<InterfaceAddress>	
Cycle time	4000	µs
Multiplexing prescale	8	
Mode	managing node	

- Sync CPU system timer with Powerlink Time

Name	Value	Unit
System		
Reboot		
Communication		
Timing		
System timer	EPL/X2X Interface	
Interface	4PPC70_0573_20B.IF1	
Cycle time of interface	4000	µs
Multiply cycle time by	1	
System tick	4000	µs
Idle time		
Inter network cross link task		
Resources		
File devices		
Time synchronization		
Internet file system		
Ethernet parameters		
DNS parameters		

- Add the DBS using “Add Hardware Module” and configure the channels.

Channel mappings depends on the specific requirement of the application. For this example, we will map:

Channel Name	Process Variable	Data Type
➤ ModuleOk	::M1_ok	BOOL
➤ ActualAlarm_I3008		UINT
➤ DC_LinkVoltage_0_1V_Isb_I3010	::M1_VDC	UINT
➤ ControlWord_I6040Out	::M1_Cw	UINT
➤ StatusWord_I6041	::M1_SW	UINT
➤ ModesOfOperation_I6060Out	::M1_ModeofOp	SINT
➤ ModesOfOperationDisplay_I6061	::M1_ModeofOpDisplay	SINT
➤ PositionActualValue_I6064	::M1_ActualPos	DINT
➤ VelocityActualValue_I606C	::M1_Actualvel	DINT
➤ TargetTorque_I6071Out	::M1_TargetTorque	INT
➤ MaxCurrent_I6073Out	::M1_TorqueLimit	UINT
➤ TorqueActualValue_I6077	::M1_ActualTorque	INT
➤ DC_linkCircuitVoltage_I6079		UDINT
➤ TargetPosition_I607AOut	::M1_TargetPos	DINT
➤ HomeOffset_I607COut	::M1_HomingOffset	DINT
➤ NegativeLimit_I607D_S01Out	::M1_NegLim	DINT
➤ PositiveLimit_I607D_S02Out	::M1_PosLim	DINT
➤ ProfileVelocity_I6081Out	::M1_ProfileVel	UDINT
➤ ProfileAcceleration_I6083Out	::M1_ProfileAcc	UDINT
➤ ProfileDeceleration_I6084Out	::M1_ProfileDec	UDINT
➤ HomingMethod_I6098Out	::M1_HomMethod	USINT
➤ TargetVelocity_I60FFOut	::M1_TargetVelocity	DINT

SAMPLE APPLICATION

In the sample application, we will create a program that can run the motor in speed mode, in position mode towards a single position and in position mode between two different positions.

To do so, in the Global_typ we will build a Machine_State type.

TYPE

```
Machine_State :
(
  Idle,
  Velocity,
  Position,
  Auto_Enter,
  Auto_A,
  Auto_B,
  Auto_Moving,
  Auto_Wait,
  Home,
  Stop
);
```

END_TYPE

And in the Global variables we will have two sets of variables. The GUI_ will be used in the user interface, meanwhile the M1_ will be linked to the motor channels

VAR

```
Case_OP : Machine_State;
GUI_auto : BOOL;
GUI_Dir : BOOL;
GUI_en_hom : BOOL;
GUI_en_pos : BOOL;
```

```
GUI_en_vel : BOOL;  
GUI_Home : BOOL;  
GUI_hom_done : INT;  
GUI_m1_ok : INT;  
GUI_new_set : BOOL;  
GUI_pos : DINT;  
GUI_pos_reach : INT;  
GUI_Run : BOOL;  
GUI_Stop : BOOL;  
GUI_tpos : DINT;  
GUI_tposA : DINT;  
GUI_tposB : DINT;  
GUI_trq : REAL;  
GUI_tvel : DINT;  
GUI_vdc : REAL;  
GUI_vel : DINT;  
M1_ActualAlarm : UINT;  
M1_ActualPos : DINT;  
M1_ActualTorque : INT;  
M1_Actualvel : DINT;  
M1_Cw : UINT;  
M1_HomingOffset : DINT;  
M1_HomMethod : USINT;  
M1_ModeofOp : SINT;  
M1_ModeofOpDisplay : SINT;  
M1_NegLim : DINT;  
M1_ok : BOOL;  
M1_PosLim : DINT;  
M1_ProfileAcc : UDINT;  
M1_ProfileDec : UDINT;  
M1_ProfileVel : UDINT;  
M1_SW : UINT;  
M1_TargetPos : DINT;  
M1_TargetTorque : INT;  
M1_TargetVelocity : DINT;  
M1_TorqueLimit : UINT;  
M1_VDC : UINT;
```

END_VAR

Link the M1 variables to the Channels as shown in the picture before.

In the program part, under the Variables.Var we will write the following variables:

VAR

```
Counter : USINT;  
Arrived_A : BOOL;  
SetAck : BOOL;  
Arrived_B : BOOL;
```

END_VAR

CODE

INIT

```
PROGRAM _INIT
(* Insert code here *)
Case_OP := Idle;
M1_PosLim:=2147483647;
M1_NegLim:=-2147483648;
M1_TorqueLimit := 1000;
Counter := 0;
```

END_PROGRAM

CYCLYC

```
CASE Case_OP OF
```

```
    Idle:
```

```
        M1_Cw:=16#06;
        GUI_new_set:=FALSE;
        GUI_hom_done:=0;
        Counter :=0;
        Arrived_A :=0;
        Arrived_B :=0;
        SetAck :=0;

        IF (NOT GUI_Run) THEN
            GUI_en_vel := FALSE;
            GUI_en_pos := FALSE;
            GUI_auto := FALSE;
        END_IF;
```

```
        (*Exit*)
        IF (GUI_en_hom) THEN
            Case_OP:=Home;
        END_IF;
        IF (GUI_Run) THEN
            IF (GUI_en_vel) THEN
                Case_OP:=Velocity;
            END_IF;
            IF (GUI_en_pos) THEN
                Case_OP:=Position;
            END_IF;
            IF (GUI_auto) THEN
                Case_OP:=Auto_Enter;
            END_IF;
        END_IF;
```

```
    Velocity:
```

```
        M1_ModeofOp := 3;
        IF (M1_ModeofOpDisplay=3) THEN
            M1_Cw := 16#0F;
        ELSE
            M1_Cw := 16#06;
        END_IF;
        (* setpoint *)
        IF (GUI_Dir) THEN
            M1_TargetVelocity:=-GUI_tvel;
        ELSE
            M1_TargetVelocity:=GUI_tvel;
        END_IF ;
```

```
(*Exit*)
IF (NOT GUI_en_vel) THEN
    Case_OP:=Idle;
END_IF;
```

Position:

```
M1_ModeofOp := 1;
IF (M1_ModeofOpDisplay=1) THEN
    M1_Cw := 16#0F;
    (* new set *)
ELSE
    M1_Cw := 16#06;
END_IF;

(* setpoint *)
IF (GUI_pos < GUI_tpos-2 OR GUI_pos > GUI_tpos+2 ) THEN
    Counter := Counter +1;
    IF (Counter > 3) THEN
        M1_Cw := 16#1F;
    ELSIF (M1_SW AND 16#1000) = 16#1000) THEN
        M1_Cw := 16#0F;
        Counter := 0;
    END_IF;
END_IF;

M1_TargetPos:=GUI_tpos;
M1_ProfileVel:=GUI_tvel;
(*Exit*)
IF (NOT GUI_en_pos) THEN
    Case_OP:=Idle;
END_IF;
```

Auto_Enter:

```
M1_ModeofOp := 1;
IF (M1_ModeofOpDisplay=1) THEN
    M1_Cw := 16#0F;
ELSE
    M1_Cw := 16#06;;
END_IF;

Counter := Counter +1;
(*Exit*)
IF (Counter >3) THEN
    IF (Arrived_A) THEN
        Case_OP:=Auto_B;
        Counter := 0;
    ELSIF (Arrived_B) THEN
        Case_OP:=Auto_A;
        Counter := 0;
    ELSE
        Case_OP:=Auto_A;
        Counter := 0;
    END_IF;
END_IF;

IF (NOT GUI_auto) THEN
    Case_OP:=Idle;
END_IF;
```

Auto_A:

```
M1_TargetPos:=GUI_tposA;
M1_ProfileVel:=GUI_tvel;
M1_Cw := 16#1F;

IF (M1_SW.12 AND M1_Cw = 16#1F) THEN
    SetAck := TRUE;
END_IF;

(*Exit*)
IF (SetAck) THEN
    Case_OP:=Auto_Moving;
    Arrived_B := FALSE;
END_IF;
IF (NOT GUI_auto) THEN
    Case_OP:=Idle;
END_IF;
```

Auto_B:

```
M1_TargetPos:=GUI_tposB;
M1_ProfileVel:=GUI_tvel;
M1_Cw := 16#1F;

IF (M1_SW.12 AND M1_Cw = 16#1F) THEN
    SetAck := TRUE;
END_IF;

(*Exit*)
IF (SetAck) THEN
    Case_OP:=Auto_Moving;
    Arrived_A := FALSE;
END_IF;
IF (NOT GUI_auto) THEN
    Case_OP:=Idle;
END_IF;
```

Auto_Moving:

```
M1_Cw := 16#0F;
Counter := Counter +1;
SetAck := FALSE;

IF (Counter > 2) THEN
    IF (M1_SW.10 AND M1_Actualvel = 0) THEN
        IF (GUI_tposA-2 < M1_ActualPos AND M1_ActualPos < GUI_tposA+2 )THEN
            Arrived_A := TRUE;
        ELSIF(GUI_tposB-2 < M1_ActualPos AND M1_ActualPos < GUI_tposB+2 )THEN
            Arrived_B := TRUE;
        END_IF;
    END_IF;
END_IF;

(*Exit*)
IF (Arrived_A OR Arrived_B) THEN
    Counter := 0;
    Case_OP:=Auto_Wait;
END_IF;
IF (NOT GUI_auto) THEN
    Case_OP:=Idle;
END_IF;
```

```
Auto_Wait:
    Counter := Counter +1;
    (**)
    IF (Counter > 6) THEN
        Case_OP:=Auto_Enter;
    END_IF;
    IF (NOT GUI_auto) THEN
        Case_OP:=Idle;
    END_IF;

Home:
    M1_ModeofOp := 6;
    M1_HomMethod := 35;
    IF (M1_ModeofOpDisplay=6) THEN
        M1_Cw := 16#1F;
    ELSE
        M1_Cw := 16#06;
    END_IF

    (* Status *)
    IF (M1_SW AND 16#400)=16#400) THEN
        ; //HomingStatus:='DONE';
        GUI_hom_done:=10; (* green *)
    ELSIF (M1_SW AND 16#2000)=16#2000) THEN
        ; //HomingStatus:='ERROR';
        GUI_hom_done:=45; (* red *)
    ELSE
        ; //HomingStatus:='IN PROGRESS';
        GUI_hom_done:=41; (* orange *)
    END_IF

    (*Exit*)
    IF (GUI_hom_done=10) THEN
        GUI_en_hom:=FALSE;
        Case_OP:=Idle;
    END_IF

Stop:
    M1_Cw:=16#06;;
    GUI_new_set:=FALSE;
    GUI_hom_done:=0;
    GUI_en_pos := FALSE;
    GUI_en_vel := FALSE;
    GUI_en_hom := FALSE;
    GUI_auto := FALSE;
    GUI_Stop := FALSE;
    Case_OP:=Idle;

END_CASE;

IF (GUI_Stop) THEN
    GUI_Run := FALSE;
    Case_OP := Stop;
END_IF;
```

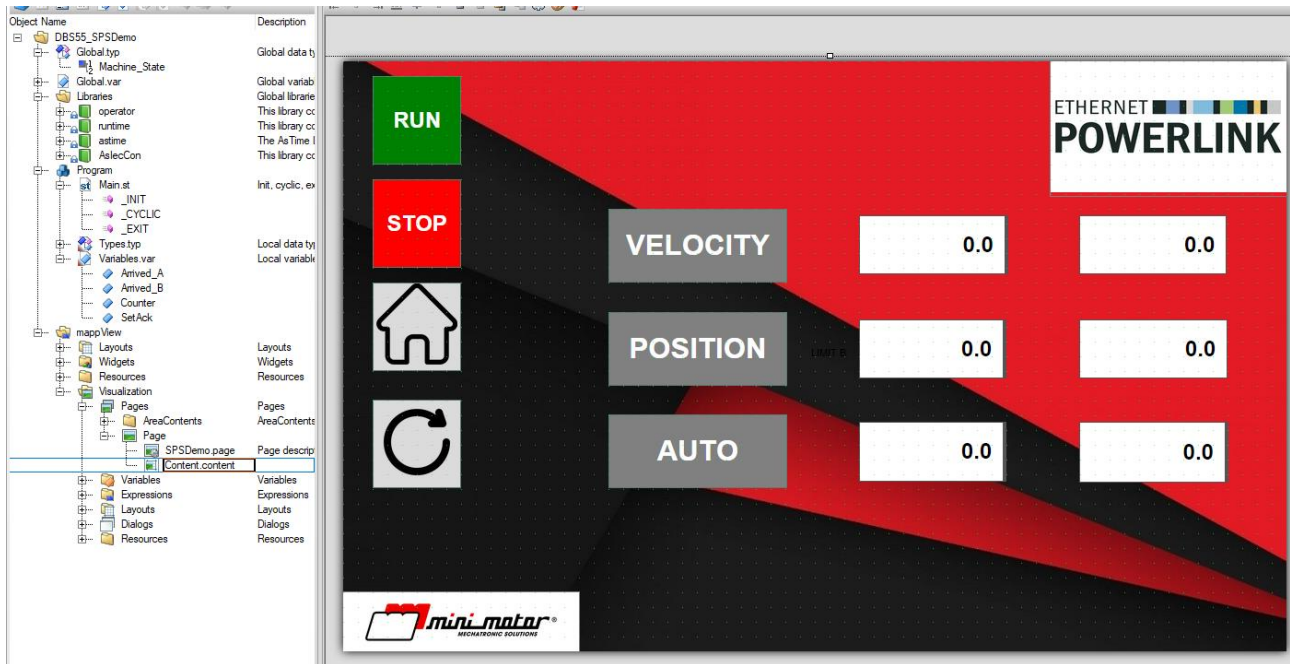
END_PROGRAM

END

```
PROGRAM_EXIT
(* Insert code here *)

END_PROGRAM
```

The program comes with a GUI suitable for a 4PPC50.



CSP

The example show how to interface a DBS integrated motor CSP enabled with the B&R platform using the ACP10 library

INITIAL STEPS

- Create new Project
- Configure Powerlink Cycle time

Name	Value	Unit
IF1		
Module type	Type 4	
Operating mode	POWERLINK V2	
MTU size	300	
Baud rate	100 MBit half duplex	
POWERLINK parameters		
Activate POWERLINK communication	on	
Device name	<InterfaceAddress>	
Cycle time	4000	µs
Multiplexing prescale	8	
Mode	managing node	

- Sync CPU system timer with Powerlink Time

Name	Value	Unit
System		
Reboot		
Communication		
Timing		
System timer	EPL/X2X Interface	
Interface	4PPC70_0573_20B.IF1	
Cycle time of interface	4000	µs
Multiply cycle time by	1	
System tick	4000	µs
Idle time		
Inter network cross link task		
Resources		
File devices		
Time synchronization		
Internet file system		
Ethernet parameters		
DNS parameters		

- Add the DBS using “Add Hardware Module” and configure the channels.
The minimum required objects for the axis function are:
Write: ControlWorld, Mode of Oper, target Position
Read: Status Word, Mode of Oper Display, Position Actual Value
- Setup the interpolation time to match the Powerlink cycle rate.

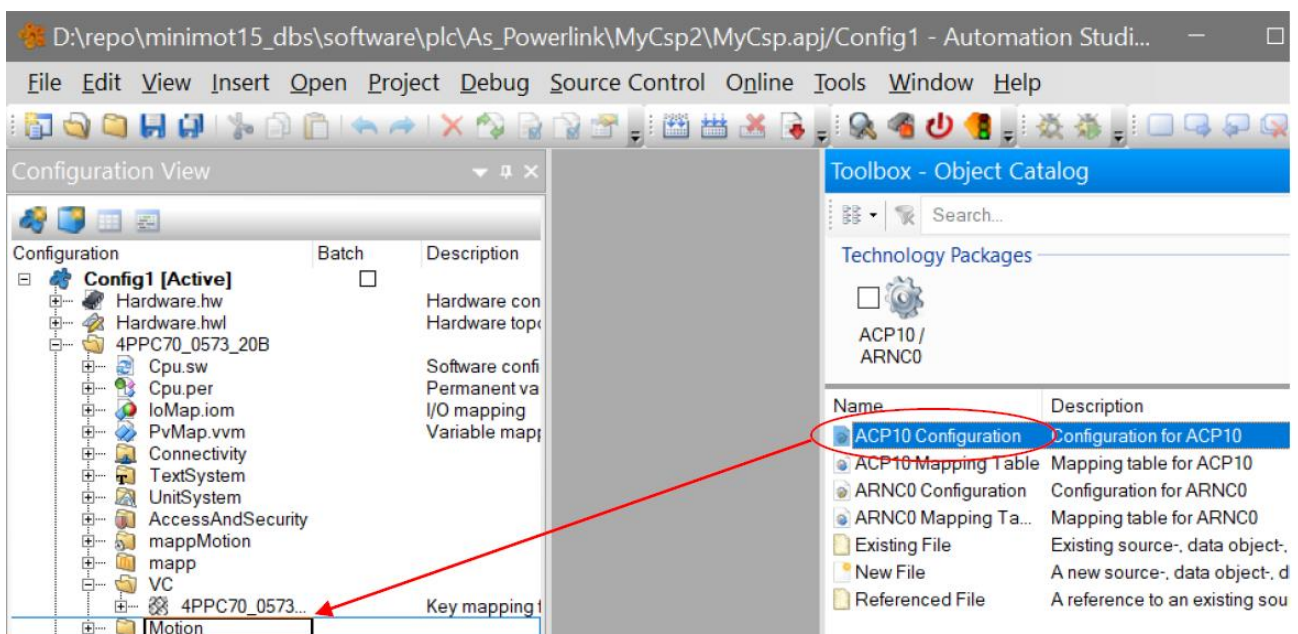
MiniMot_DBS55 [Configuration] ×	
Name	Value
InterpolationSubmodeSelect_I60C0	
Interpolation time period_I60C2 RECORD[2]	
InterpolationTimePeriodValue_I60C2_S01	
Datatype	USINT
Init value	4
InterpolationTimeIndex_I60C2_S02	
Datatype	SINT
Init value	-3
Simulation	
Simulation device	

The setup in the picture is 4ms as in $4^{-3}s$

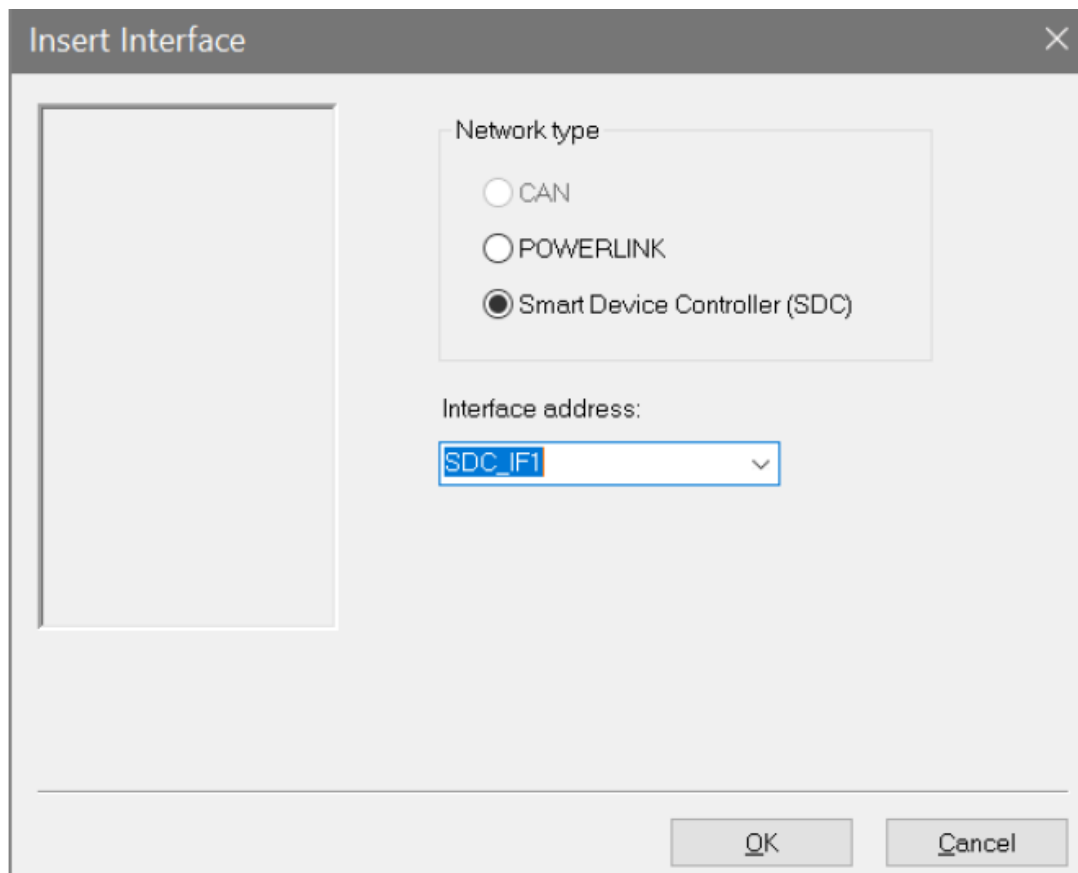
- Add the relevant variables in the Global.Var and link them to the DBS I/O mapping. In the provided example, there are more variables linked, to better monitor the DBS system

ADD AXIS TO THE NDC

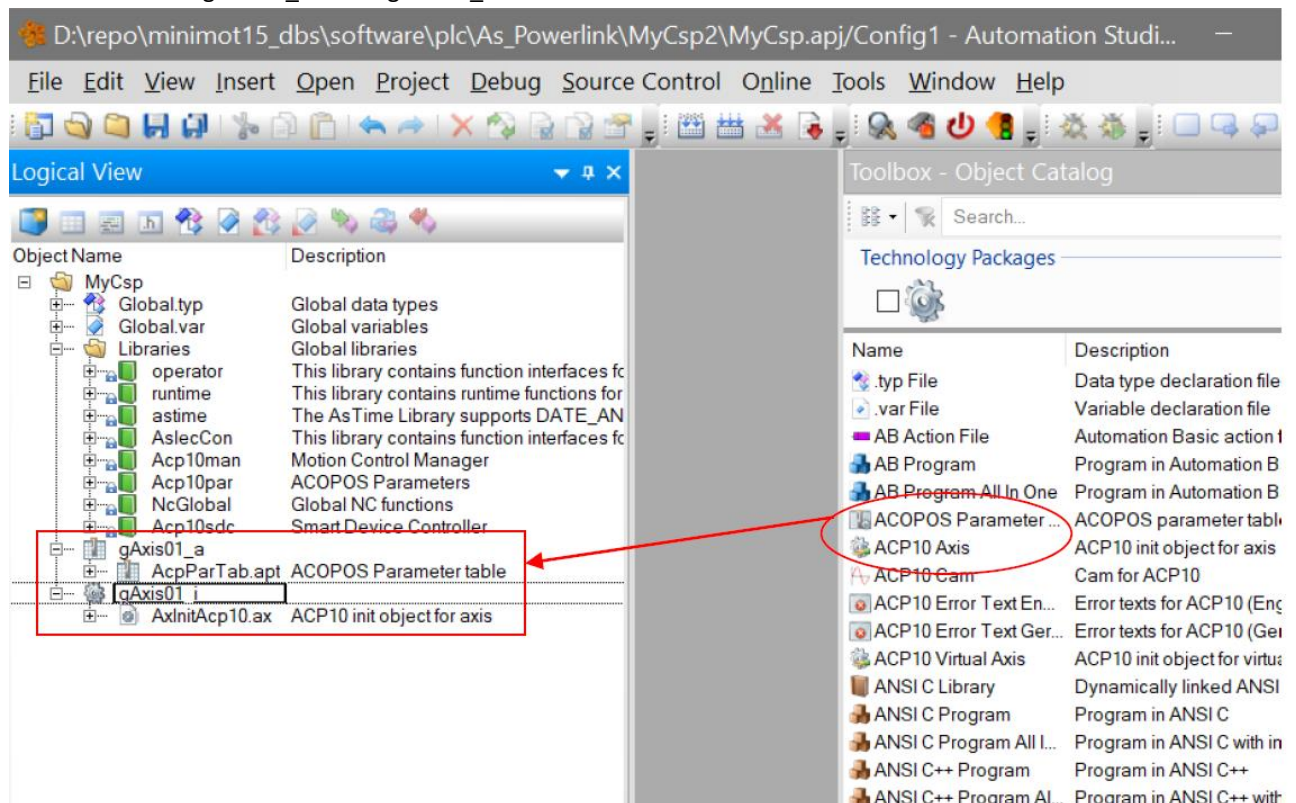
- Add the motion folder in the configuration of the CPU. Adding the ACP10 Configuration



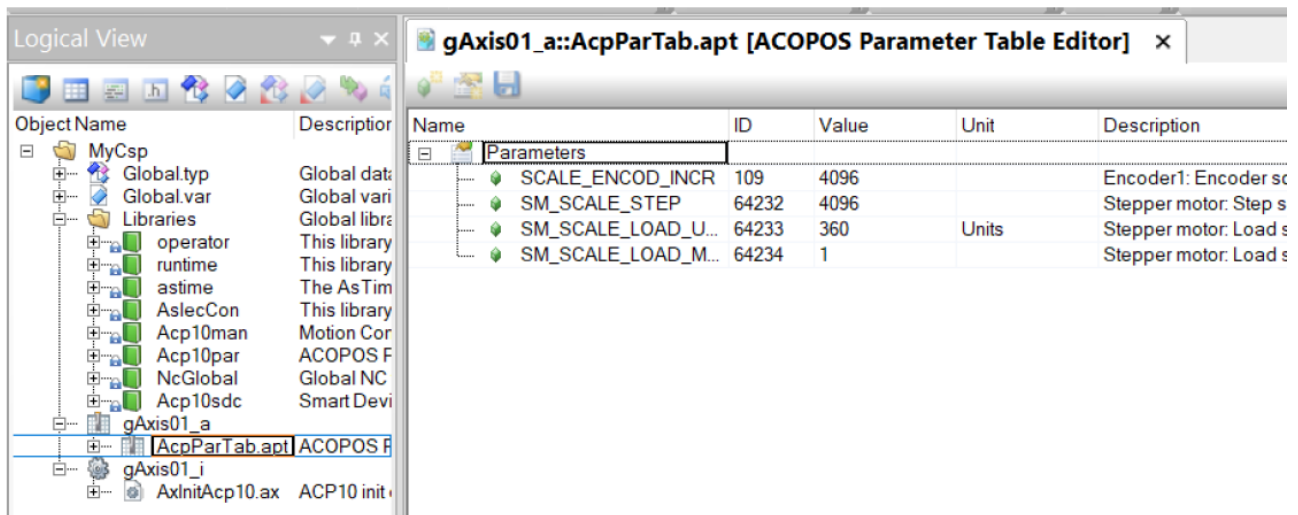
- Open Acp10cfg.ncc, right click in the blank space and select “Insert Interface ...”, then select Smart Device Controller and confirm with OK



- In the logical tab, select the root node and add a "ACOPOS parameter table" and a "ACP10 Axis"; then rename them as "gAxis01_a" and "gAxis01_i"



- In the configuration tab, open “Acp10map1.ncm”, right click and select “Add NC Object” then compile it as:
 - Object name: gAxis01
 - PLC Address: SDC_IF1.ST1
 - Nc object: ncAxis
 - Channel: 1
 - Init parameter gAxis01_i
 - ACOPOS Parameter gAxis01_a
- “gAxis01_a” parameters and setup following parameter:
 - SCALE_ENCOD_INCR (ID=109) 4096 (pulses per revolution of DBS)
 - SM_SCALE_STEP (ID=64232) 4096 (pulses per revolution of DBS)
 - SM_SCALE_LOAD_UNITS (ID=64233) 360 (units per revolution, degrees)
 - SM_SCALE_LOAD_MOTREV (ID=64234) 1 (units referred to one revolution)



The screenshot shows the software interface with two main windows. On the left is the 'Logical View' pane, which displays a tree structure of objects. The 'gAxis01_a' object is selected, and its sub-objects are visible, including 'AcpParTab.apr' (ACOPOS F) and 'gAxis01_i' (ACP10 init). On the right is the 'gAxis01_a::AcpParTab.apr [ACOPOS Parameter Table Editor]' window, which displays a table of parameters.

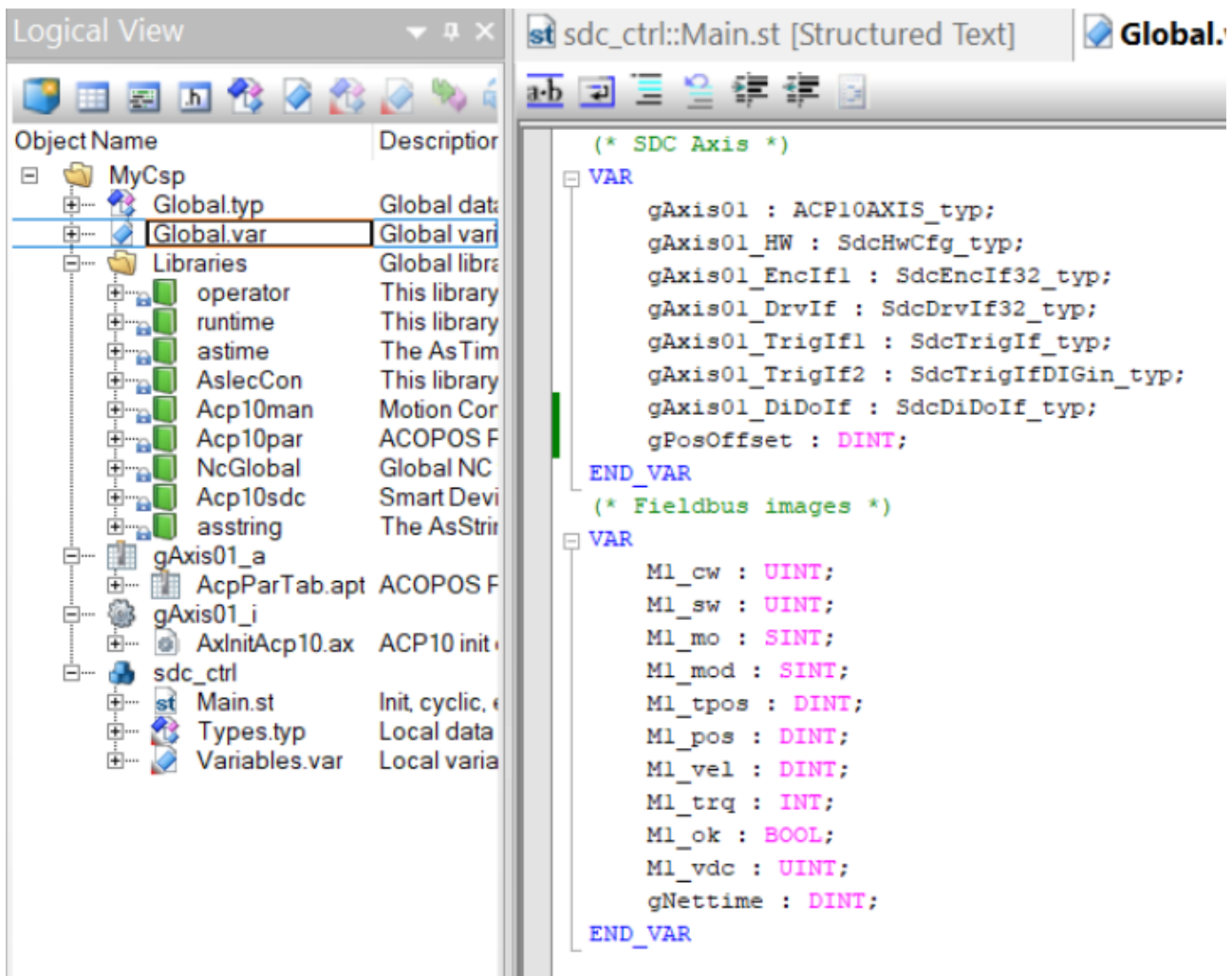
Name	ID	Value	Unit	Description
Parameters				
SCALE_ENCOD_INCR	109	4096		Encoder1: Encoder sc
SM_SCALE_STEP	64232	4096		Stepper motor: Step s
SM_SCALE_LOAD_U...	64233	360	Units	Stepper motor: Load s
SM_SCALE_LOAD_M...	64234	1		Stepper motor: Load s

- Open “gAxis01_i” parameter tree and set the following parameters:
 - dig_in level: ncACTIVE_HI
 - encoder_if\parameter\scaling\load\units 360 (degrees per rev)
 - limit\parameter\v_* 18000 (degrees/sec, 3000rpm)
 - limit\parameter\a* 180000 (degrees/sec^2, 30000rpm/s)
 - move\basis\parameter\v_* 18000 (degrees/sec, 3000rpm)
 - move\basis\parameter\a* 180000 (degrees/sec^2, 30000rpm/s)

Object Name	Description	Name	Value	Unit	Description
MyCsp		ACP10AXIS_type			
Global.typ	Global data	dig_in			Digital Inputs
Global.var	Global vari	level			Active Input Level
Libraries	Global libra	reference	ncACTIV_HI		Reference switch
operator	This library	pos_hw_end	ncACTIV_HI		Positive HW end switch
runtime	This library	neg_hw_end	ncACTIV_HI		Negative HW end switch
astime	The AsTim	trigger1	ncACTIV_HI		Trigger1
AslecCon	This library	trigger2	ncACTIV_HI		Trigger2
Acp10man	Motion Cor	encoder_if			Encoder Interface
Acp10par	ACOPOS F	parameter			Parameters
NcGlobal	Global NC	count_dir	ncSTANDARD		Count direction
Acp10sdc	Smart Devi	scaling			Scaling
gAxis01_a		load			Load
AcpParTab.apr	ACOPOS F	units	360	Units	Units at the load
gAxis01_i		rev_motor	1		Motor revolutions
AxInitAcp10.ax	ACP10 init	limit			Limit value
		parameter			Parameters
		v_pos	18000.0	Units/s	Speed in positive direction
		v_neg	18000.0	Units/s	Speed in positive direction
		a1_pos	180000.0	Units/s ²	Acceleration in positive direction
		a2_pos	180000.0	Units/s ²	Acceleration in positive direction
		a1_neg	180000.0	Units/s ²	Acceleration in positive direction
		a2_neg	180000.0	Units/s ²	Acceleration in positive direction
		t_jolt	0	s	Jolt time
		t_in_pos	0	s	Settling time before message 'In Posit
		pos_sw_end	2000000000	Units	Positive SW end
		neg_sw_end	-2000000000	Units	Negative SW end
		ds_warning	500	Units	Lag error limit for display of a warning
		ds_stop	1000	Units	Lag error limit for stop of a movement
		a_stop	1.0e30	Units/s ²	Acceleration limit for stop of a movem
		dv_stop	0	1/s	Speed error limit for stop of a movem
		dv_stop_mode	ncOFF		Mode for speed error monitoring
		controller			Controller
		move			Movement
		stop			Stop Movement
		homing			Homing procedure
		basis			Basis movements
		parameter			Parameters
		v_pos	18000.0	Units/s	Speed in positive direction
		v_neg	18000.0	Units/s	Speed in positive direction
		a1_pos	180000.0	Units/s ²	Acceleration in positive direction
		a2_pos	180000.0	Units/s ²	Acceleration in positive direction
		a1_neg	180000.0	Units/s ²	Acceleration in positive direction
		a2_neg	180000.0	Units/s ²	Acceleration in positive direction
		message			Messages (errors, warnings)

SDC – DS402 CONNECTION

- In the Global.var add the following SDC Axis variables:



Logical View

Object Name	Description
MyCsp	
Global.typ	Global data
Global.var	Global vari
Libraries	Global libra
operator	This library
runtime	This library
astime	The AsTim
AslecCon	This library
Acp10man	Motion Cor
Acp10par	ACOPOS F
NcGlobal	Global NC
Acp10sdc	Smart Devi
asstring	The AsStri
gAxis01_a	
AcpParTab.apl	ACOPOS F
gAxis01_i	
AxlnitAcp10.ax	ACP10 init
sdc_ctrl	
Main.st	Init, cyclic, e
Types.typ	Local data
Variables.var	Local varia

Structured Text Editor: sdc_ctrl::Main.st

```

(* SDC Axis *)
VAR
    gAxis01 : ACP10AXIS_typ;
    gAxis01_HW : SdcHwCfg_typ;
    gAxis01_EncIf1 : SdcEncIf32_typ;
    gAxis01_DrvIf : SdcDrvIf32_typ;
    gAxis01_TrigIf1 : SdcTrigIf_typ;
    gAxis01_TrigIf2 : SdcTrigIfDIGin_typ;
    gAxis01_DiDoIf : SdcDiDoIf_typ;
    gPosOffset : DINT;
END_VAR

(* Fieldbus images *)
VAR
    M1_cw : UINT;
    M1_sw : UINT;
    M1_mo : SINT;
    M1_mod : SINT;
    M1_tpos : DINT;
    M1_pos : DINT;
    M1_vel : DINT;
    M1_trq : INT;
    M1_ok : BOOL;
    M1_vdc : UINT;
    gNettime : DINT;
END_VAR
  
```

- Add new "ST Program all in one" and rename it "sdc_ctrl"
- Add the B&R library "asstring"
- Open Cpu.sw and ensure that "sdc_ctrl" runs in the "Cyclic 1" at 4ms (it needs to run synchronously to Powelink data exchange)

CODE

INIT

In the _INIT code part, add the configuration of the SDC axis; the code actually links all the sub-structures.

```

(* initialize EncIf1 *)
gAxis01_HW.EncIf1_Typ := ncSDC_ENC32;
strcpy( ADR(gAxis01_HW.EncIf1_Name[0]), ADR('gAxis01_EncIf1') );
(* initialize DrvIf *)
gAxis01_HW.DrvIf_Typ := ncSDC_DRVSM32;
strcpy( ADR(gAxis01_HW.DrvIf_Name[0]), ADR('gAxis01_DrvIf') );
(* initialize TrigIf1 *)
gAxis01_HW.TrigIf1_Typ := ncSDC_TRIG;
strcpy( ADR(gAxis01_HW.TrigIf1_Name[0]), ADR('gAxis01_TrigIf1') );
(* initialize TrigIf2 *)
  
```

```

gAxis01_HW.TrigIf2_Typ := ncSDC_TRIGDIGin;
strcpy( ADR(gAxis01_HW.TrigIf2_Name[0]), ADR('gAxis01_TrigIf2') );
(* initialize DiDolf *)
gAxis01_HW.DiDolf_Typ := ncSDC_DIDO;
strcpy( ADR(gAxis01_HW.DiDolf_Name[0]), ADR('gAxis01_DiDolf') );

```

CYCLYC

The _CYCLIC code part is composed by two blocks; the first simulates the life counters in order to comply to liveness management from the numerical control; the OK flags are linked to the ModuleOK flag coming from powerlink

```

(* life counter simulation *)
gAxis01_DrvIf.iLifeCnt := gAxis01_DrvIf.iLifeCnt + 1;
gAxis01_Enclf1.iLifeCnt := gAxis01_Enclf1.iLifeCnt + 1;
gAxis01_Enclf1.iActTime := DINT_TO_INT(gNettime AND 16#FFFF);
gAxis01_DiDolf.iLifeCntDriveEnable := gAxis01_DiDolf.iLifeCntDriveEnable + 1;
gAxis01_DiDolf.iLifeCntDriveReady := gAxis01_DiDolf.iLifeCntDriveReady + 1;
gAxis01_DiDolf.iLifeCntNegHwEnd := gAxis01_DiDolf.iLifeCntNegHwEnd + 1;
gAxis01_DiDolf.iLifeCntPosHwEnd := gAxis01_DiDolf.iLifeCntPosHwEnd + 1;
gAxis01_DiDolf.iLifeCntReference := gAxis01_DiDolf.iLifeCntReference + 1;
gAxis01_TrigIf1.iLifeCnt := gAxis01_TrigIf1.iLifeCnt + 1;
gAxis01_TrigIf2.iLifeCnt := gAxis01_TrigIf2.iLifeCnt + 1;
(* drive ready emulation *)
gAxis01_Enclf1.iEncOK := M1_ok;
gAxis01_DiDolf.iDriveReady := M1_ok;
gAxis01_DrvIf.iDrvOK := M1_ok;

```

The second part of _CYCLIC code is the actual DS402 glue, where control word is driven from the gAxis enable and the target is linked to gAxis output reference

```

(* ds402 linking *)
gAxis01_Enclf1.iActPos := M1_pos; (* position actual value to axis *)
gAxis01_DrvIf.iStatusEnable := gAxis01_DrvIf.iDrvOK;
M1_mo := 8; (* CSP *)
IF (M1_mod=8) THEN
  IF ((M1_sw AND 16#4F) = 16#08) THEN
    M1_cw := 16#80; (* FAULT ACK *)
  ELSE
    IF gAxis01_DiDolf.oDriveEnable THEN
      M1_cw := 16#0F; (* ENABLE *)
    ELSE
      M1_cw := 16#06; (*DISABLE *)
    gPosOffset := M1_pos - gAxis01_DrvIf.oSetPos;
  END_IF
END_IF
ELSE
  M1_cw:=16#06; (*DISABLE *)
END_IF
(* motor target position *)
M1_tpos := gAxis01_DrvIf.oSetPos + gPosOffset; (* position target to drive *)

```

SAMPLE APPLICATION

```
GUI_csp_acc:=36000.0;
GUI_csp_vel:=3600.0;
GUI_csp_pos1:=0.0;
GUI_csp_pos2:=3600.0;
```

CYCLYC

In the _CYCLIC the main part is the PLCOPEN blocks for driving the motor and the state machine to drive them.

```
gAxis01_Reset(Axis := ADR(gAxis01), Execute:=gAxis01_ResetEx);
gAxis01_Power(Axis := ADR(gAxis01), Enable:=gAxis01_PowerEn);
gAxis01_ReadPos(Axis := ADR(gAxis01), Enable:=TRUE);
gAxis01_Home(Axis := ADR(gAxis01), Execute:=gAxis01_HomeEx);
gAxis01_MoveAbs1(Axis := ADR(gAxis01), Execute:=gAxis01_Move1Ex, Position:=GUI_csp_pos1,
Velocity:=GUI_csp_vel, Acceleration:=GUI_csp_acc, Deceleration:=GUI_csp_acc);
gAxis01_MoveAbs2(Axis := ADR(gAxis01), Execute:=gAxis01_Move2Ex, Position:=GUI_csp_pos2,
Velocity:=GUI_csp_vel, Acceleration:=GUI_csp_acc, Deceleration:=GUI_csp_acc);
(* Axis switch on *)
IF (GUI_en_csp) THEN
CASE (gAxis01_State) OF
1: (* Init reset *)
gAxis01_ResetEx:=FALSE;
gAxis01_PowerEn:=FALSE;
gAxis01_HomeEx:=FALSE;
gAxis01_Move1Ex:=FALSE;
gAxis01_Move2Ex:=FALSE;
gAxis01_State:=2;
2: (* Reset *)
gAxis01_ResetEx:=TRUE;
IF (gAxis01_Reset.Done) THEN
gAxis01_State:=3;
END_IF
3: (* Power *)
gAxis01_ResetEx:=FALSE;
gAxis01_PowerEn:=TRUE;
IF (gAxis01_Power.Status) THEN
gAxis01_State:=11;
END_IF
11: (* Home *)
gAxis01_HomeEx:=TRUE;
IF (gAxis01_Home.Done) THEN
gAxis01_State:=21;
END_IF
21: (* Move POS1 *)
gAxis01_Move1Ex:=TRUE;
gAxis01_Move2Ex:=FALSE;
IF (gAxis01_MoveAbs1.Done) THEN
gAxis01_State:=22;
```

```
END_IF
22: (* Move POS2 *)
gAxis01_Move1Ex:=FALSE;
gAxis01_Move2Ex:=TRUE;
IF (gAxis01_MoveAbs2.Done) THEN
gAxis01_State:=21;
END_IF
END_CASE
ELSE
gAxis01_State:=1;
gAxis01_ResetEx:=FALSE;
gAxis01_PowerEn:=FALSE;
END_IF
```

Finally the provided example adds a GUI suitable for a 4PPC70 that allows switching on and off the drive and setting up the targets and the parameters for the position trajectories.