Application Note
1 SI Transparent CAN Function Blocks

Important User Information

This document is intended to provide a good understanding of the function block functionality offered with the 1 SI CANopen module for ET200S when using the Transparent CAN functionality.

The reader of this document is expected to be familiar with high level software design and communication systems in general.

Liability

Every care has been taken in the preparation of this application note. Please inform HMS Industrial Networks AB of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks AB, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks AB. HMS Industrial Networks AB assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the applications meets all performance and safety requirements including any applicable laws, regulations, codes, and standards

HMS Industrial Networks AB will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features, timing, or functional side effects found outside the documented scope of this product. The effects caused by any direct or indirect use of such aspects of the product are undefined, and may include e.g. compatibility issues and stability issues.

The examples and illustrations in this document are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks AB cannot assume responsibility for actual use based on these examples and illustrations.

Intellectual Property Rights

HMS Industrial Networks AB has intellectual property rights relating to technology embodied in the product described in this document. These intellectual property rights may include patents and pending patent applications in the US and other countries.

Trademark Acknowledgements

All trademarks are the property of their respective holders.

Document History

Revision	Date	Description	Author
0.90	2011-01-24	Created	Lars-Åke Väldesjö
0.91	2011-02-11	Internal update	Lars-Åke Väldesjö
0.92	2011-08-18	Updated with more details	Mikael Mårtensson
0.93	2011-08-26	Updated after review	Mikael Mårtensson
1.00	2011-09-29	First Release	Kaspar Dahlqvist

Referenced Documents

Document name	Author	Document id.	Revision
1 SI CANopen User Manual	HMS	SCM-1200-005	1.30

More Information About the Product

The latest information can be downloaded from www.et200can.com.

Contents

1.Required Equipment and Software

The following items are needed to perform the installation according to the example file:

Description	Name / Type	Version
Hardware		
ET200S system Either IM-type connected to PLC or CPU-type as standalone. See further information in User Manual in appendix “Siemens Interface Modules Compatibility”	An IM151-8 PN/DP CPU rack is used standalone in the example. (6ES7 151-8AB00-0AB0)	
Power module	(6ES7 138-4CA50-0AB0)	
Input module	(6ES7 131-4BF00-0AA0)	
Output module	(6ES7 132-4BF00-0AA0)	
Two 1 SI CANopen Modules configured as slaves	020570-B	2.11 or later
Software		
SIMATIC STEP7 tool	N/A	V5.4 SP 5 or later
USB to CAN adapter (Only to monitor CAN-traffic, not required)	020810-B	
PLC Function blocks for Transparent CAN (2.0A)	From www.et200can.com	1.01 or later
HSP 2066 (STEP7 configuration file) or GSD files for the 1 SI CANopen Module	From www.et200can.com	1.00 or later
Other		
CANopen cables	N/A	N/A
PLC configuration cable	N/A	N/A

2. Application Note Overview

2.1. Purpose

The purpose of this application note is to describe the use of function blocks available for the Transparent CAN functionality of the 1 SI CANopen module. This application note describes how to import, compile and use the Transparent CAN Function Blocks available for download from www.et200can.com. It also refers to an example consisting of a STEP7 project. In the STEP7 example an IM151-8 PN/DP CPU (6ES7 151-8AB00-0AB0) is used with the program running in the internal CPU.

The example program will use 2 1 SI CANopen modules. Initialize one (Node 2) to operate in Transparent CAN mode (see "6 Initialization of Transparent CAN Example" for details), and the second to operate in CANopen mode (Node 1). Set the network to OPERATIONAL by sending the CANopen command "Go to Operational".

The program will then transfer 4 bytes from Node 1 to Node 2 with the value 01020304h, and 4 bytes from Node 2 to Node 1 with the value AABBCCDDh. It will also allow the user to monitor the status of the Transparent CAN module by using the CAN Status function block.

2.2. Prerequirements

Two 1 SI CANopen modules should be mounted in an IM151-8 PN/DP CPU with reference 6ES7 151-8AB00-0AB0 (other ET200 systems can be used, but in that case the ET200S system has to be changed in the example project before download) and a CANopen cable should be connected between the two modules with correct termination (see section "Installation" in the user manual for details on how to connect the two modules). The program downloaded with this application note should be downloaded into the ET200S system.

If the CAN network should be monitored, the CAN analyzer must also be connected to the CAN network.

2.3. Detailed Description

The example program will transfer a CANopen PDO message (CAN frame) from the CANopen module (Node 1), with the contents 01020304h, to the Transparent CAN node (Node 2). This CAN frame will be received in the module (Node 2), setup to operate in Transparent CAN mode, by the CAN Receive function block (see "10 CAN Receive Function Block" for details).

The example program will also send one CAN frame (with the contents of a CANopen SDO Write command) from the Transparent CAN node (Node 2) CAN Send function block (see "9 CAN Send Function Block" for details). This frame will be received as an SDO command in the CANopen module (Node 1), writing data AABBCCDDh to the first 4 bytes of the memory area (the response to the SDO request will also be received by the Transparent CAN receive block).

The example program will also show how to use the CAN Status function block (see "8 CAN Status Function Block" for details) to see the amount of frames that has been received in the Transparent CAN node (Node 2).

The section below describes the example step by step. The document assumes that the reader is familiar with industrial communication as well as CAN and Siemens STEP7.

1. The two modules will start up and send out BOOTUP messages on CAN (See section “12.2 CAN-log” row 1 and 2).

2. The 1 SI CANopen Module (Node 2) will be set in Transparent CAN mode and configured by Network 10 (see “6 Initialization of Transparent CAN Example” for details) and a NMT command “Go to Operational” will be sent (RUN-LED on CANopen slave module (Node 1) will go to stable green (Operational)). See section “12.2 CAN-log” row 3.

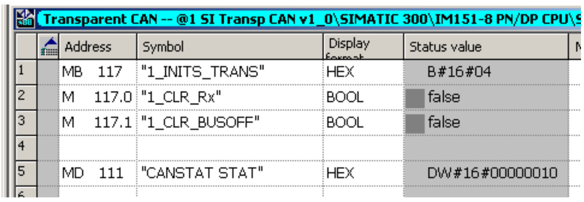
3. The slave node will send out its four default PDOs (they will be received by the Transparent CAN node via function block CAN Receive). See section “12.2 CAN-log” row 4 to 7.

4. Start to monitor variables in STEP7 VAR table “Transparent CAN”.

5. Enter the value 01020304h for QD32 (row 25) in the VAR-table and press F9 to transfer the new value to the module. This makes the slave send out its updated PDO (COBID 181h). See section “12.2 CAN-log” row 8. (Change the data in row 25 to make a new transmission of the PDO (COBID 181h) or change IO in the digital input module). The PDO (COBID 181h) is received by the Transparent CAN node (Node 2) via function block CAN Receive.

6. F9 will also set appropriate data to be sent out via a SDO Write command by setting “CAN Send REQ” to TRUE on row 15. Data AABBCCDDh will be set in the first 4 bytes of the slave. The SDO-response from the slave is received by the Transparent node, function block CAN_RCV. (“CAN Send REQ” is cleared after every transmission). See section “12.2 CAN-log” row 9 for SDO-command and row 10 for response.

7. Continue to set “CAN Send REQ” four more times so that “CANSTAT STAT” on row 5 changes to 10hex, indicating that the Buffer Limit in the Receiver is reached. 10 or more messages are received without being read out from the 1 SI CANopen Module:

   Transparent CAN -- @1 SI Transp CAN v1_0\SIMATIC 300\IM151-8 PN/DP CPU\

	Address	Symbol	Display format	Status value
1	MB 117	"1_INITS_TRANS"	HEX	B#16#04
2	M 117.0	"1_CLR_Rx"	BOOL	false
3	M 117.1	"1_CLR_BUSOFF"	BOOL	false
4				
5	MD 111	"CANSTAT STAT"	HEX	DW#16#00000010

   (When more than 127 messages are in the buffer, “CANSTAT STAT” is changed to 18h, indicating that the buffer is half full. When the buffer is full and the first message is lost, “CANSTAT STAT” is changed to 1Ch, indicating that the buffer is full).

8. Every time “CAN Reci ACK p out” is set, the next received CAN-frame will be shown in VAT lines 9-13 (CAN-frame data). (The data in CANSTAT STAT will change if buffer goes under actuating levels.)

3.The CAN-frame (2.0A)

The standard CAN frame consists of the following bits:

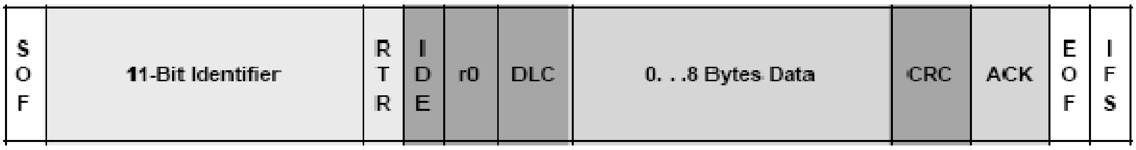


Figure 1 Standard CAN frame

- **SOF, IDE, r0, CRC, ACK, EOF, IFS** - Control bits and CRC used for CAN communication. No change needed. This will be handled automatically by the software.
- **Identifier** - The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority. Also referred to as COBID.
- **RTR** - The single Remote Transmission Request bit is set high when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way all data being used in a system is uniform.
- **DLC** - The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.
- **Data** - Up to 8 bytes of application data may be transmitted in one CAN frame.

NOTE: Extended (29-bit) identifiers (2.0B) are not supported by the 1 SI CANopen module in Transparent CAN mode.

4. How to Import Blocks into the STEP7 Project

The four function blocks handling the Transparent CAN functionality have to be imported and compiled into your STEP7 project. The available function blocks for Transparent CAN are “CAN Receive”, “CAN Send”, “CAN Control” and “CAN Status”. This is done by following the steps below.

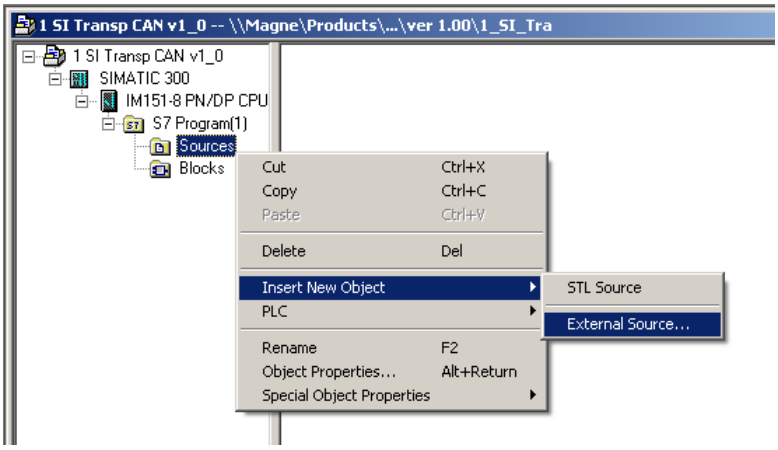


Figure 2 Select external source for blocks.

1. Select “Sources”
2. Right click in white area and select “Insert New Object” from “External Source”

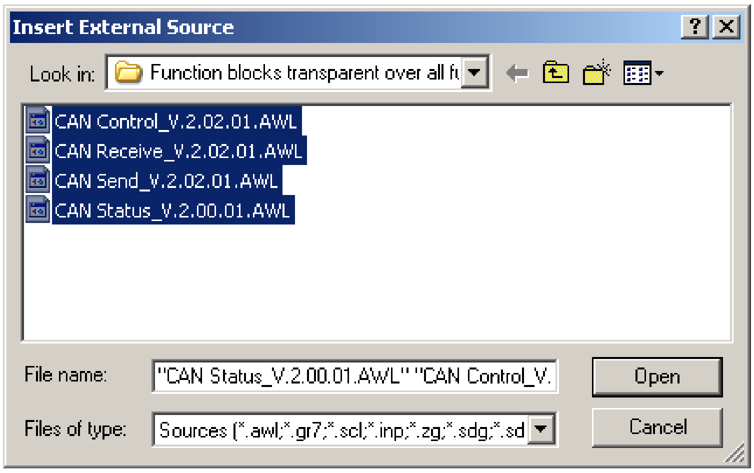


Figure 3 Select files.

3. Select the files needed and press “OPEN”

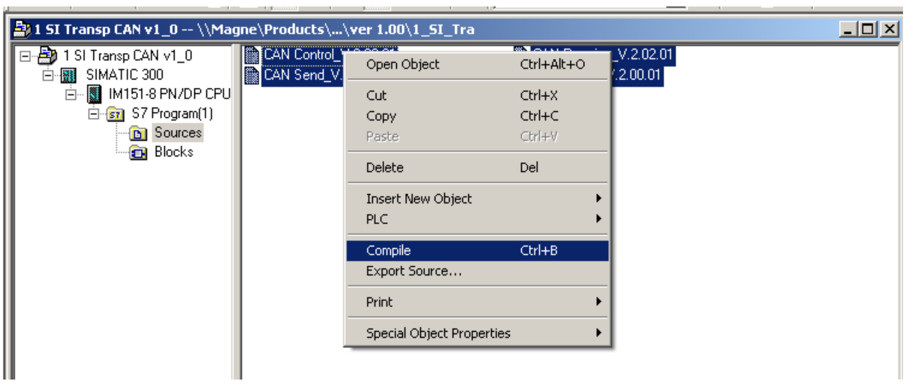


Figure 4 Compile Blocks.

4. To compile the blocks: select the files, right click and choose compile
   Note: *English language is required in STEP7 for the compilation to succeed.*

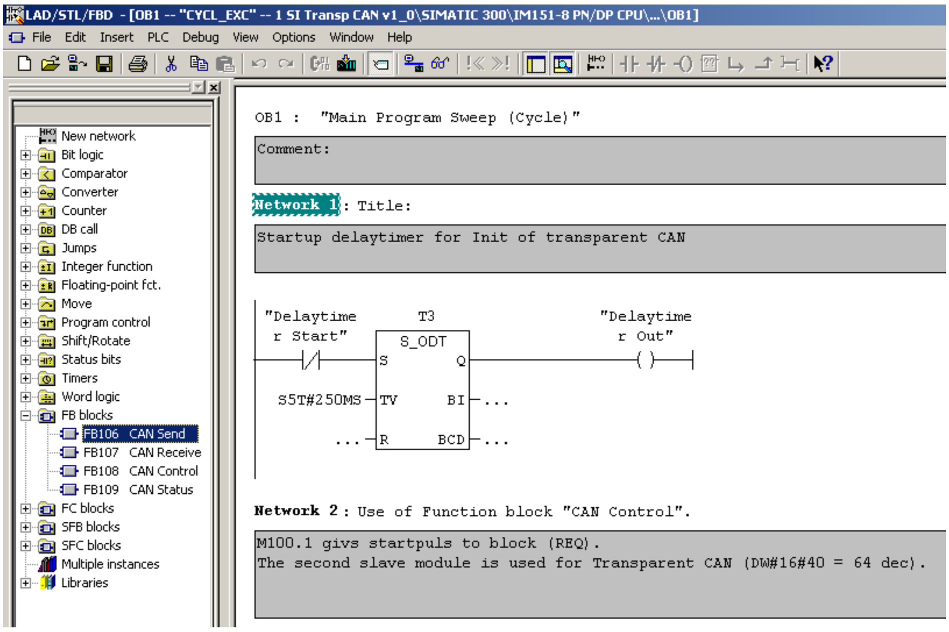


Figure 5 Insert Block from “FB blocks”.

5. Close "compile windows". The blocks can now be selected from **FB blocks**. They have now been given numbers as below:

   FB106: Named “CAN Send” in symbol table in example code.
   FB107: Named “CAN Receive” in symbol table in example code.
   FB108: Named “CAN Control” in symbol table in example code.
   FB109: Named “CAN Status” in symbol table in example code.

5. Hardware Configuration in STEP 7

When the hardware has been setup the STEP7 project has to be configured accordingly. This is done by the following steps.

First set up the backplane size of the module. This is done by selecting the appropriate module from the “hardware list” under “Special modules”. In the example two 16 byte 1 SI CANopen modules are used.

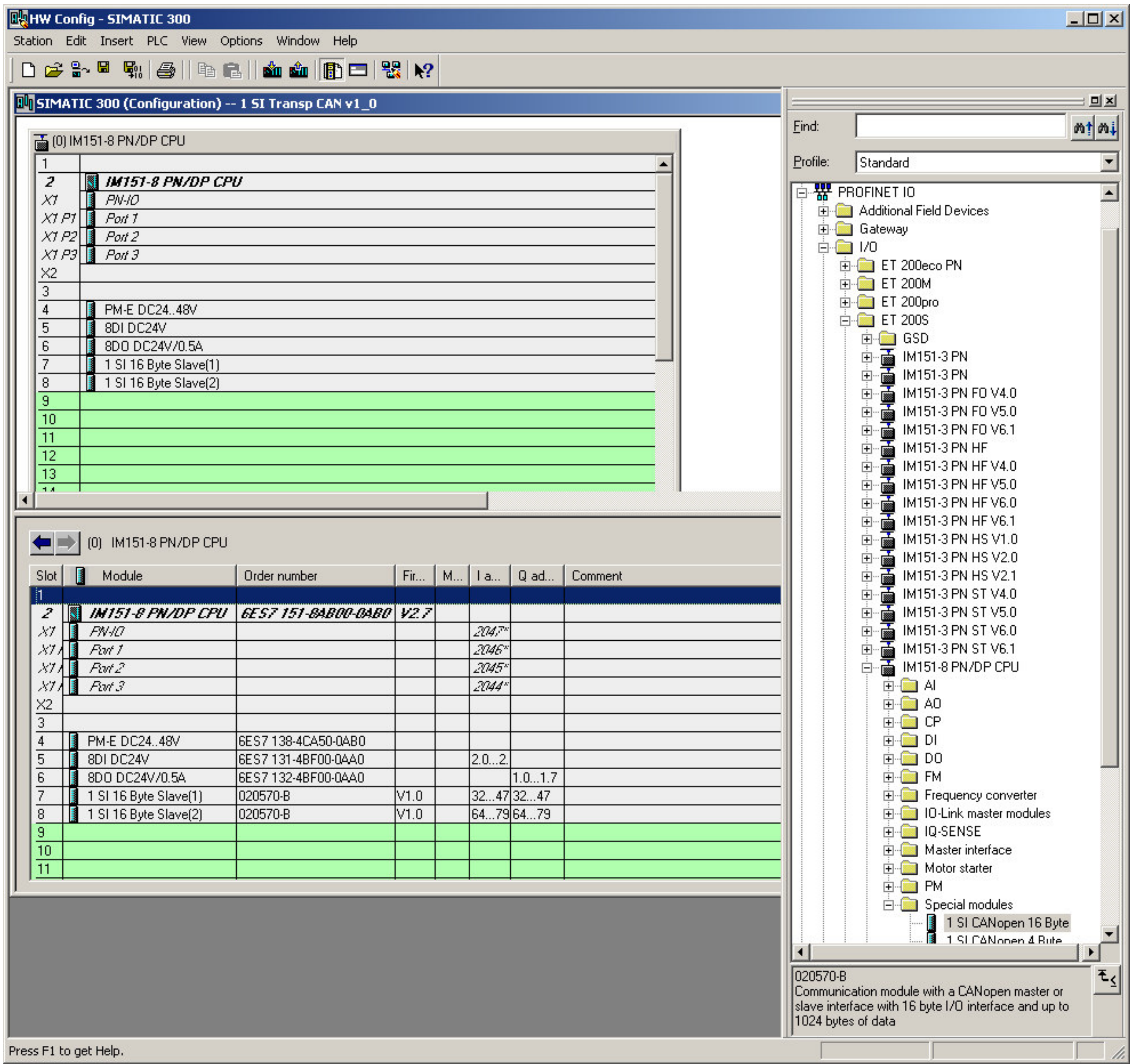


Figure 6 Hardware configuration, hardware list.

NOTE: A 16 or 32 byte module is necessary for Transparent CAN to work

When the two modules have been selected the appropriate offset has to be configured. Double click the appropriate module and select the "Addresses" tab. In this example we use offset 32 for the CANopen node and offset 64 for the Transparent CAN node.

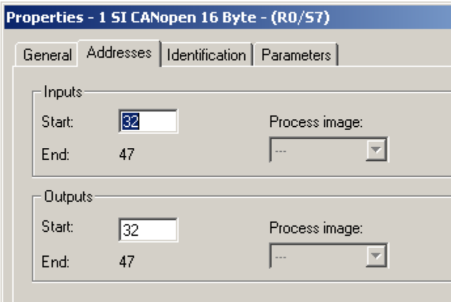


Figure 7 Hardware configuration, hardware list.

When the two offsets have been configured, the settings for each node have to be changed. This is done by selecting the "Parameters" tab in the same window. The same setting can be used for all parameters except "Node number" that have to be set to 1 for the CANopen node and 2 for the Transparent CAN node. See settings below.

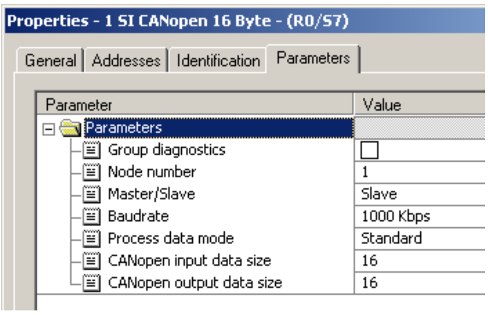


Figure 8 Setting properties for the first 1 SI CANopen Module (Node 1).

Download the hardware configuration to the ET200S system.

NOTE: The example code requires that the I/O of the 1 SI CANopen module is in the process image.

6. Initialization of Transparent CAN Example

The initialization of Transparent CAN in the 1 SI CANopen module is made, in this example, by using Network 10 in section OB1 of the STEP7 example project.

The functionality of the state machine in Network 10 is described in the network as comments, and a short functional description is also made in the "header" of the network as well as in the text below.

The PLC-program makes the module with node ID 2 switch from CANopen mode to Transparent CAN mode by sending a CAN Control message with FCN= 0 (RUN-LED will flash three times to indicate the correct mode).

The PLC-program sets the Transparent CAN node to receive all messages by sending a CAN Control message with FCN= 1, LEN=1 and PARAM=FFFFh (receive all COBIDs).

The next step for the PLC-program is to set Buffer Limit Reached in Transparent node to 0Ah (10 Decimal), by sending a CAN Control message with FCN= 2, LEN=1 and PARAM=000Ah.

The PLC-program makes the Transparent node send the "Go to Operational" command to the other node by sending a CAN Send message with RTR=0, COBID= 0, DATA=01 00 and CANSIZE=02h. This is done to make the CANopen slave node send out its PDOs.

6.1. Important Comments about the Code

- If the CAN Receive Block is not set to the "repower" state when the 1 SI CANopen module is restarted, the Block and the module can become unsynchronized. This is done by resetting bits "fActive", "BUSY" and "NEW" in the Data Base for the Block.

- In state 1 below RET1: The filter setting for the CAN Receive block can be setup to receive only the COBIDs required. In the example all COBIDs are received. It is however recommended to setup the COBIDs actually used, to decrease the load of the PLC and let the module transfer only the required messages to the PLC. (An example on how to enable COBIDs 602 and 181 is entered into the project as a comment, and also change LEN to the number of parameters given).

- In state 2 below RET2: The limit for BUFFER LIMIT REACHED can be changed by modifying the value A0000 (10 messages) to the appropriate value. The first word of the double-word is used as only one parameter is given.

- When the initialization is done the state machine stays in state 4.

    - By setting bit 0 to TRUE in "1_INITS_TRANS", MB117, the state-machine clears the Rx-buffer in CAN Receive.

    - By setting bit 1 to TRUE in "1_INITS_TRANS", MB117, the state-machine clears the BUS OFF-condition.

Below is a screen dump of Network 10:

```
Network 10: Title:
State machine to init Transparent CAN (Value for "1_INITS_TRANS" MB117)
0)   Power up value. Init (FCN=0),
     1) Read filter (FCN=1; FFFF or other),
     2) Limit to 10 (0Ah, FCN=2, PARAM=000Ah)
        and send "goto operational" with CAN SEND
     3) Wait until finnished and no error.
     4) Start receiver and stay in state 4.
1-4) se above.

5)   Reset Rx-buffer (FCN=3). Then go to state 4). Done by setting bit 0 to TRUE.

6)   Reset BUS-off (FCN=5). Then go to state 4).Done by setting bit 1 to TRUE.
```

```
//If 1_INITS_TRANS is = 0 (Power up) Enable transparent CAN.

      L     0
      L     "1_INITS_TRANS"
      <>I
      JCB   ST01                        // If 0 continue below else test next state.

      A     "CANCTRL BUSY"              // Jump out if still BUSY
      JCB   ST00

      A     "Delaytimer Out"            // Jump out if start delay timer is running
      JNB   ST00

// Set CAN Receive to repower state by clearing fActive-bit, BUSY-bit and NEW-bit
// In CAN Receive DB (OBS! Correct DB for CAN Receive has to be called).
      R     "CAN Receive DB135".fActive
      R     "CAN Receive DB135".BUSY
      R     "CAN Receive DB135".NEW

RET0: NOP   0

      L     0                           // Enable transparent CAN
      T     "CANCTRL FCN"
      L     0
      T     "CANCTRL LEN"
      L     0
      T     "CANCTRL PARAM"
      S     "CANCTRL REQ"

      L     1                           // 1 Count up state machine.
      T     "1_INITS_TRANS"

      JU    ST00                        // Jump to THE END

ST01: NOP   0                           // State 1 Enable Receiver

      L     1
      L     "1_INITS_TRANS"
      <>I
      JCB   ST02                        // If 1 continue below else test next state.

      A     "CANCTRL BUSY"              // Jump out if still BUSY
      JCB   ST00

      L     0
      L     "CANCTRL RET"
      <>I
      JCB   RET0                        // If CANCTRL RET says Error try "Enable transparent CAN" again.

RET1: NOP   0

      L     1                           // Enable receiver for all incomming CAN-frames
      T     "CANCTRL FCN"
      L     1                           // (1) Works with  RECEIVE all COB-id:s, data FFFF0000
//      L     2                           // (2) Works with RECEIVE COB-id 602 and 181 below
      T     "CANCTRL LEN"
      L     DW#16#FFFF0000              // (1) Works with 1 as LEN
//      L     DW#16#86028181              // (2) Works with 2 as LEN above (RECEIVE COB-id 602 and 181)
      T     "CANCTRL PARAM"
      S     "CANCTRL REQ"               // Enable CAN CTRL REQ

      L     2                           // 2 Count up state machine.
```

```
      T     "1_INITS_TRANS"

      JU    ST00                        // Jump to THE END

ST02: NOP   0                           // State 2 set buffer limit to 10

      L     2
      L     "1_INITS_TRANS"
      <>I
      JCB   ST03                        // If 2 continue below else test next state.

      A     "CANCTRL BUSY"              // Jump out if still BUSY
      JCB   ST00

      L     0
      L     "CANCTRL RET"
      <>I
      JCB   RET1                        // If CANCTRL RET says Error try "Enable Receiver" again.

RET2: NOP   0

      L     2                           // Set BUFFER LIMIT REACHED
      T     "CANCTRL FCN"
      L     0
      T     "CANCTRL LEN"
      L     DW#16#A0000                 // Set limit to 10 (0x000A 0000)
      T     "CANCTRL PARAM"
      S     "CANCTRL REQ"               // Enable CAN CTRL REQ
      L     3                           // 3 Count up state machine.
      T     "1_INITS_TRANS"
      JU    ST00                        // Jump to THE END

ST03: NOP   0                           //  Send "Goto Operational" with CAN SEND command.

      L     3
      L     "1_INITS_TRANS"
      <>I
      JCB   ST04                        // If 3 continue below else test next state.
      A     "CANCTRL BUSY"              // Jump out if still BUSY
      JCB   ST00
      L     0
      L     "CANCTRL RET"
      <>I
      JCB   RET2                        // If CANCTRL RET says Error try "set buffer limit to 10" again.

      A     "CAN Send BUSY"             // Jump out if still BUSY
      JCB   ST00

      L     0                           // Set COB-ID to 0
      T     "CAN Send COBID"
      L     DW#16#1000000               // Set data to 0100 (0x0100 0000)
      T     "CAN Send DATA1"
      L     DW#16#0                     // Clear rest of data to 0000 (0x0000 0000)
      T     "CAN Send DATA2"
      L     2
      T     "CAN Send SIZE"             // Send first two bytes of data
      S     "CAN Send REQ"              // Enable CAN Send REQ

      L     4                           // 4 Count up state machine.
      T     "1_INITS_TRANS"

      JU    ST00                        // Jump to THE END

ST04: NOP   0                           // 4 Rest state

      L     4
      L     "1_INITS_TRANS"
      <>I
      JCB   ST05                        // If 4 continue below else test next state.
      S     "CAN Reci REQ"              // Enable Receiver block
      S     "CANSTAT REQ"               // Enable Status block

ST05: NOP   0                           // Reset Rx buffer state

      L     5
      L     "1_INITS_TRANS"
      <>I
      JCB   ST06                        // If 5 continue below else test next state.
      A     "CANCTRL BUSY"              // Jump out if still BUSY
      JCB   ST00

      L     3                           // Set Reset Rx buffer
```

```
      T     "CANCTRL FCN"
      L     0
      T     "CANCTRL LEN"
      S     "CANCTRL REQ"            // Enable CAN CTRL REQ
      L     4                        // 4 Go to rest state.
      T     "1_INITS_TRANS"
      JU    ST00                     // Jump to THE END

ST06: NOP   0                        // Reset BUS OFF state

      L     6
      L     "1_INITS_TRANS"
      <>I
      JCB   ST00                     // If 6 continue below else jump to THE END.
      A     "CANCTRL BUSY"           // Jump out if still BUSY
      JCB   ST00
      L     5                        // Reset BUS OFF
      T     "CANCTRL FCN"
      L     0
      T     "CANCTRL LEN"
      S     "CANCTRL REQ"            // Enable CAN CTRL REQ
      L     4                        // 4 Go back to rest state.
      T     "1_INITS_TRANS"
      JU    ST00                     // Jump to THE END

ST00: NOP   0                        // Here is THE END
```

7. CAN Control Function Block

This block is used to control the state of the transparent CAN layer and to set its parameters. It consists of 5 different function calls.

Function (FCN)	Description
0	Enables Transparent CAN mode. All ordinary CANopen functionality is disabled, and the module has to be restarted to re-enable CANopen functionality. The rest of the function codes in CAN Control can only be used if the Transparent CAN mode is enabled.
1	Configures the CAN Receive acceptance filter in the module, i.e. what COBIDs will be accepted when receiving data frames. The module will not listen to the CAN bus, if no COBID in the filter is enabled. The acceptance filter can be changed at any time.
2	Configures the warning limit for the CAN Receive buffer. It gives the opportunity to define the amount of frames that will be stored in the receive buffer, before the BUFFER LIMIT REACHED in the status code is set.
3	Empties the receive buffer. CAN Receive still holds the latest frame that has not been acknowledged (NEW is cleared by holding ACK high for on scan cycle).
5	Clears BUS OFF condition. If a BUS OFF condition is generated, the CAN controller has to be reset, before communication can be resumed.

7.1. How to Configure the CAN Control Function Block

Load the function block "CAN Control" into a network and set input and output data.

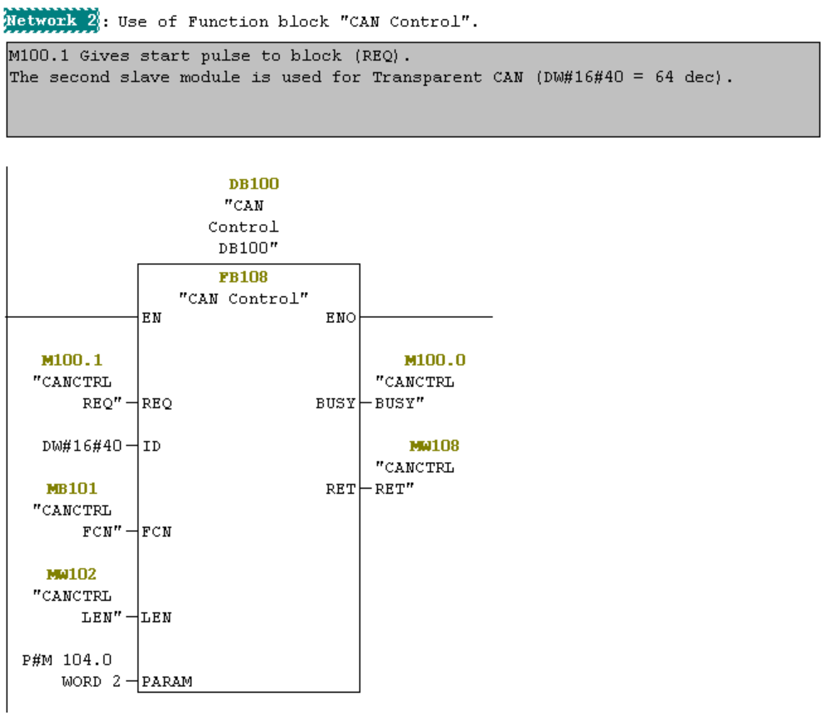


Figure 9 CAN Control block

Inputs:

Name	**Explanation**	**Example info**
DBx	Enter a data block id DBx where x is unique. (Used by the function block for internal data variables and calculations)	*DB100*
REQ	If TRUE for one scan cycle of the PLC-program, a CAN Control request is started (see RET below for handling instructions for REQ)	*"CANCTRL REQ":* M100.1
ID	Logic address for the position where the INPUT data starts for the 1 SI CANopen module in the rack.	• *DW#16#:* Data is entered as doubleword in hex • *40:* The actual first address in hex, I64 (decimal) set in Hardware Configuration of the module
FCN	Select the function that will be sent to the module.	• *"CANCTRL FCN":* MB101
LEN	Only valid when FCN = 1, otherwise ignored. Defines the size of the acceptance filter array that is sent in PARAM.	• *"CANCTRL LEN":* MW102
PARAM	Place and maximum size of the location that holds the Parameters to be sent along with the function. Only valid when FCN = 1 or 2: **FCN** **Contents** 1 Acceptance filter array 2 Buffer limit value	• *P#M:* Pointer to M-area • *104.0:* Start address in M-area • *WORD 2:* Maximum size to use in M-area (up to two Parameters of word size in this example)
EN	TRUE runs the CAN Control block. Always run in example.	

Outputs:

Name	**Explanation**	**Example info**
ENO	Not used in this example.	**-**
BUSY	If the request is not finished within one scan cycle, BUSY turns TRUE and stays TRUE until the request is finished, when it returns to FALSE.	*"CANCTRL BUSY":* M100.0
RET	Holds result from execution of block. Available when BUSY turns FALSE, until REQ is set to TRUE again. For Error codes, see appendix "Error Codes (RET)" in User manual.	*"CANCTRL RET":* MW108

8. CAN Status Function Block

This block is used to request the status of the module. The status value is found in the parameter STAT.

The status code is represented by 32 bits, where each bit represents a status code.

NOTE: If Group Diagnosis is enabled in the module via HW configuration in STEP7, the channel diagnostics interrupt will report "External error" as long as any bit in the Status Code is set.

Status code	Name	Description
0x00000000	NO ERRORS	-
0x00000001	BUS OFF	The CAN error counter have passed BUS OFF level.
0x00000002	ERROR PASSIVE	The CAN error counter has passed the warning level.
0x00000004	BUFFER FULL	255 messages are stored, and the receive buffer is full. Reset by reading out one message from the buffer.
0x00000008	BUFFER HALF FULL	Set when 128 messages are stored in the receive buffer, i.e. the buffer is half full.
0x00000010	BUFFER LIMIT REACHED	Set at 255 initially, can be changed by FCN 2, see "CAN Control"

8.1. How to Configure the CAN Status block

Load the function block "CAN Status" into a network and set input and output data.

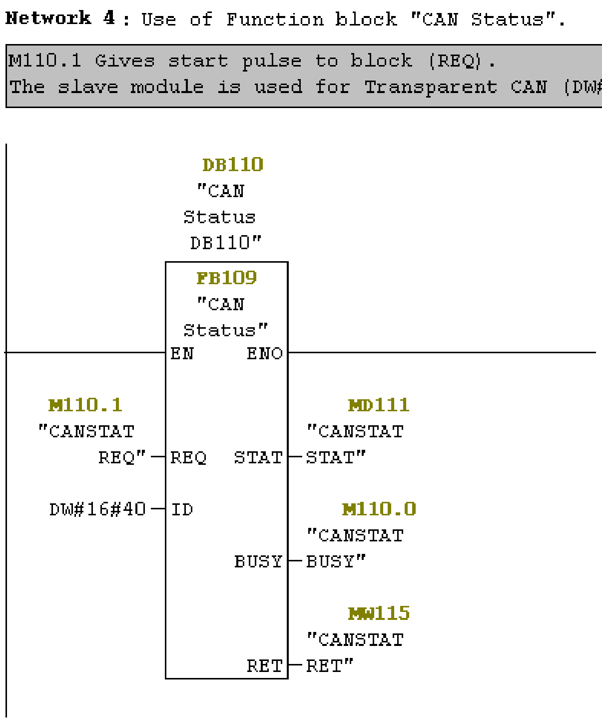


Figure 10 CAN Status block

Inputs:

Name	Explanation	Example info
DBx	Enter a data block id DBx where x is unique. (Used by the function block for internal data variables and calculations)	*DB110*
REQ	TRUE for one scan cycle of the PLC-program starts the CAN Status (see RET below for handling instructions for REQ)	*"CANSTAT REQ":* M110.1
ID	Logic address for the position where the INPUT data starts for the 1 SI CANopen module in the rack.	• *DW#16#:* Data is entered as doubleword in hex • *40:* The actual first address in hex, I64 (decimal) set in Hardware Configuration of the module
EN	TRUE runs the CAN Status block	Always run

Outputs:

Name	Explanation	Example info
ENO	Not used in this example.	-
STAT	The returned status of the module	*"CANSTAT STAT":* MD111
BUSY	If the request is not finished within one scan cycle, BUSY turns TRUE and stays TRUE until the request is finished, when it returns to FALSE.	*"CANSTAT BUSY":* M110.0
RET	Holds result from execution of block. Available when BUSY turns FALSE, until REQ is set to TRUE. For Error codes, see appendix "Error Codes (RET)" in User manual.	*"CANSTAT RET":* MW115

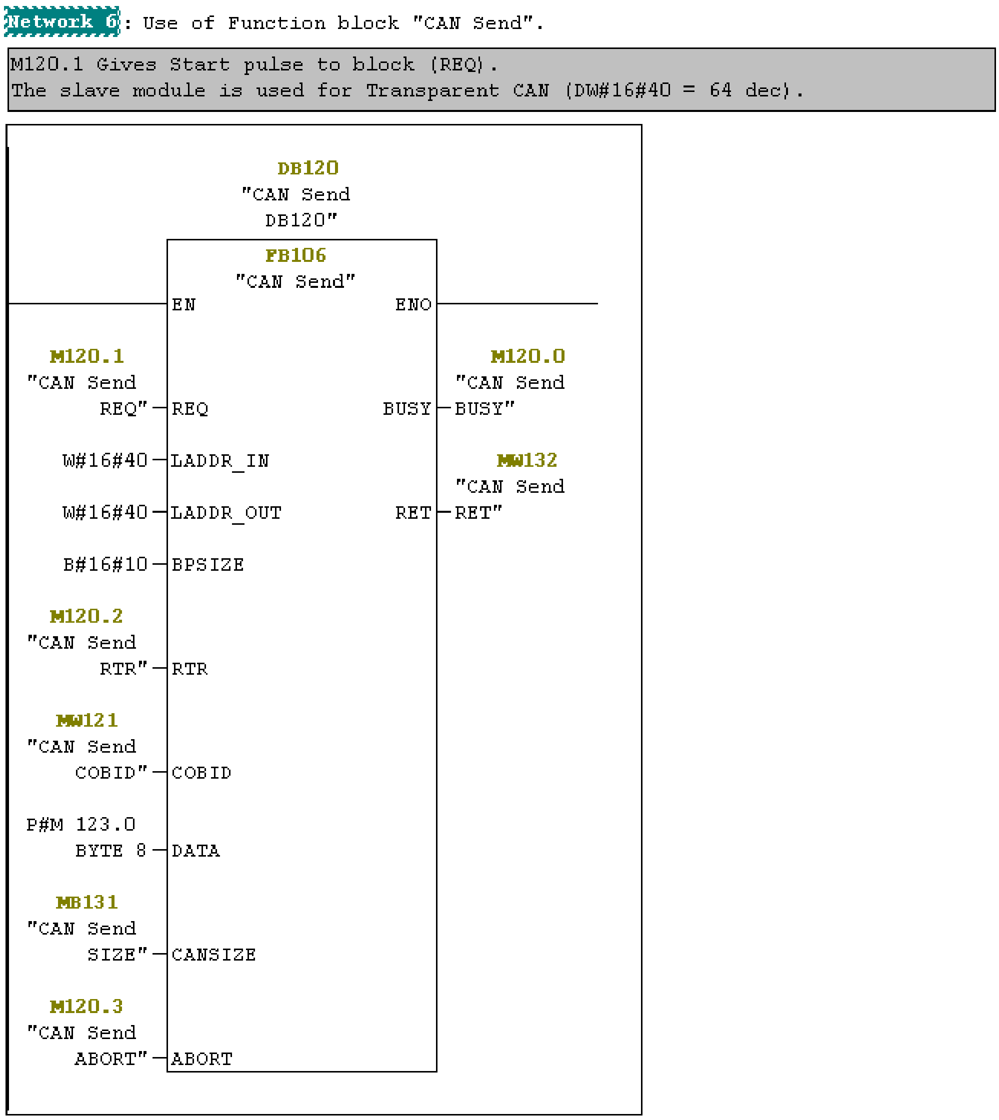
9. CAN Send Function Block

The CAN Send function block handles the frames that are sent from the 1 SI CANopen Module in Transparent CAN mode (Node 2) to the CAN bus.

When REQ turns TRUE, a CAN frame is sent according to the parameters defined by the user. When the frame has been acknowledged by the module, BUSY will go FALSE, and a new frame can be sent. This will continue as long as REQ is set to TRUE.

9.1. How to Configure the CAN Send block

Load the function block “CAN Send” into a network and set input and output data.



CAN Send block

Inputs:

Name	Explanation	Example info
DBx	Enter a data block id DBx where x is unique. (Used by the function block for internal data variables and calculations)	*DB120*
REQ	TRUE for one scan cycle of the PLC-program starts the CAN Send request (see RET below for handling instructions for REQ)	*"CAN Send REQ":* M120.1
LADDR_ IN	Set logic address for the position where the INPUT data starts for the 1 SI CANopen module in the rack.	• *W#16#:* Data put in as word in hex • *40:* The actual first address in hex, I64 (decimal) set in Hardware Configuration of the module
LADDR_ OUT	Set logic address for the position where the OUTPUT data starts for the 1 SI CANopen module in the rack.	• *W#16#:* Data put in as word in hex • *40:* The actual first address in hex, Q64 (decimal) set in Hardware Configuration of module
BPSIZE	Set 1SI CANopen module backplane size. The size of selected module from Hardware Configuration	• *B#16#:* Data put in as bytes in hex • *10:* The actual size in hex. Put in 10h for a 16 byte module.
RTR	Remote Transmission Request bit on the CAN net	*"CAN Send RTR":* M120.2
COBID	The standard CAN 11-bit identifier (COBID) is placed in a Word	*"CAN Send COBID":* MW121
DATA	Place and maximum size of the location to read the data to send (up to eight bytes of data can be sent via CAN)	• *P#M:* Pointer to M-area • *123.0:* Start address in M-area • *BYTE 8:* Data size of the M-area
CANSIZE	Number of bytes to send	*"CAN Send SIZE":* MB131
ABORT	Aborts the current transmission when set to TRUE (if BUS OFF is reached, ABORT transmission and clear BUS OFF condition, CAN Control function 5.)	*"CAN Send ABORT":* M120.3
EN	TRUE runs the CAN Send block	Always run

Outputs:

Name	Explanation	Example info
ENO	Not used in this example.	-
BUSY	If the request is not finished within one scan cycle, BUSY turns TRUE and stays TRUE until the request is finished, when it returns to FALSE.	*"CAN Send BUSY":* M120.0
RET	Holds result from execution of block. Available when BUSY turns FALSE, until REQ is set to TRUE again. For Error codes, see appendix "Error Codes (RET)" in User manual.	*"CAN Send RET":* MW132

10. CAN Receive Function Block

CAN Receive handles the frames that have been received from the 1 SI CANopen Module in Transparent CAN mode.

When REQ is TRUE, the block looks for new frames waiting in the CAN buffer of the module. If there are any, the module will read the oldest frame, set NEW to TRUE, and wait for an ACK. This will be repeated until all frames have been read. When all frames have been read BUSY will go FALSE. The received frames will be buffered until they are read by the PLC. The buffer can hold up to 255 frames.

For polling operation, set REQ to TRUE permanently.

NOTE: ACK must be reset to FALSE after one scan cycle. Otherwise incoming frames will automatically be acknowledged as read, before they have been accepted by the PLC. This may result in data loss, as new frames may overwrite frames that haven't been handled completely.

10.1. How to Configure the CAN Receive block

Load the function block "CAN Receive" into a network and set input and output data.

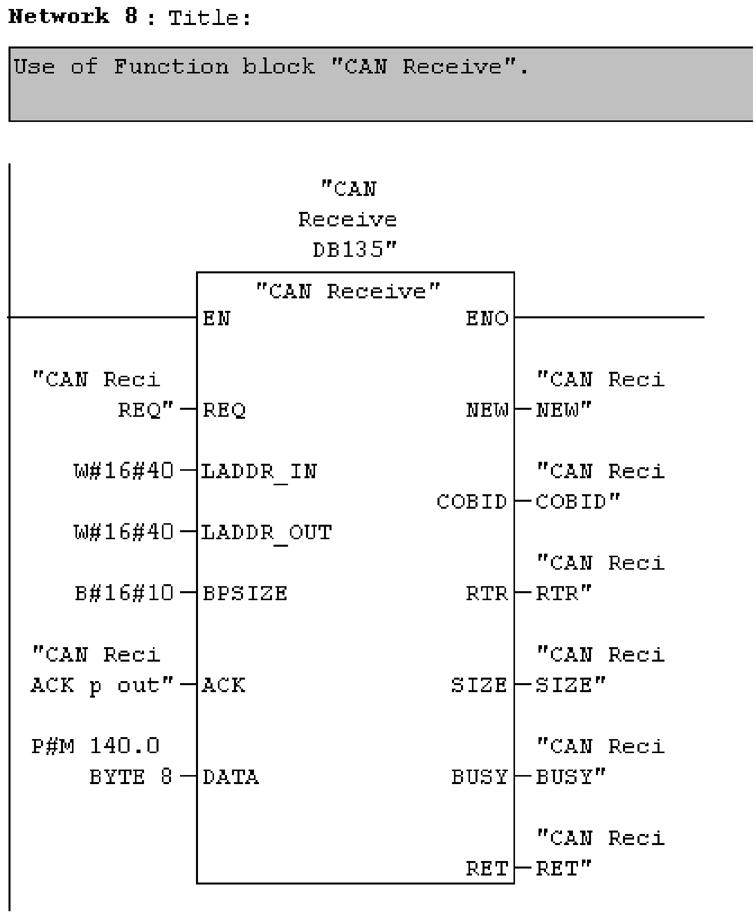


Figure 11 CAN Receive block

Inputs:

Name	**Explanation**	**Example info**
DBx	Enter a data block id DBx where x is unique. (Used by the function block for internal data variables and calculations)	*DB135*
REQ	TRUE for one scan cycle of the PLC-program starts the CAN Receive request (see RET below for handling instructions for REQ). For polling operation, set REQ to TRUE permanently	*"CAN Reci REQ":* M135.1
LADDR_IN	Set logic address for the position where the INPUT data starts for the 1 SI CANopen Module in the rack.	• *W#16#:* Data put in as word in hex • *40:* The actual first address in hex, I64 (decimal) set in Hardware Configuration of the module
LADDR_OUT	Set logic address for the position where the OUTPUT data starts for the 1 SI CANopen Module in the rack.	• *W#16#:* Data put in as word in hex • *40:* The actual first address in hex, Q64 (decimal) set in Hardware Configuration of module
BPSIZE	Set 1SI CANopen module backplane size. The size of selected module from Hardware Configuration	• *B#16#:* Data put in as bytes in hex • *10:* The actual size in hex. Put in 10h for a 16 byte module.
ACK	Set to TRUE for one scan cycle to acknowledge a newly read CAN frame (constant TRUE reads out all messages, one per scan cycle)	*"CAN Reci ACK p out":* M135.6
DATA	Place and maximum size of the location to store the received data (up to eight bytes of data can be received via CAN)	• *P#M:* Pointer to M-area • *140.0:* Start address in M-area • *BYTE 8:* Data size of the M-area
EN	TRUE runs the CAN Receive block	Always run

Outputs:

Name	**Explanation**	**Example info**
ENO	Not used in this example.	-
NEW	Set to TRUE when a new CAN frame has been read from the module.	*"CAN Reci NEW":* M135.4
COBID	The standard CAN 11-bit identifier (COBID) is placed in a Word	*"CAN Reci COBID":* MW138
RTR	Remote Transmission Request bit on the CAN net	*"CAN Reci RTR":* M135.3
SIZE	Number of bytes received	*"CAN Reci SIZE":* MB134
BUSY	Set to TRUE if there is data in the receive buffer when REQ is set. BUSY will be TRUE until buffer is empty.	*"CAN Reci BUSY":* M135.0
RET	Holds result from execution of block. Available when BUSY turns FALSE, until REQ is set to TRUE again. For Error codes, see appendix "Error Codes (RET)" in User manual.	*"CAN Reci RET":* MW136

11. The Use of OB82 to get Group Diagnostics Interrupt

It is possible to get diagnostic information from the 1 SI CANopen Module. This is done by checking the Group diagnostic checkbox in the hardware configuration. The different errors that can be generated can be found in the User manual in section "PROFIBUS/PROFINET Channel Diagnostics".

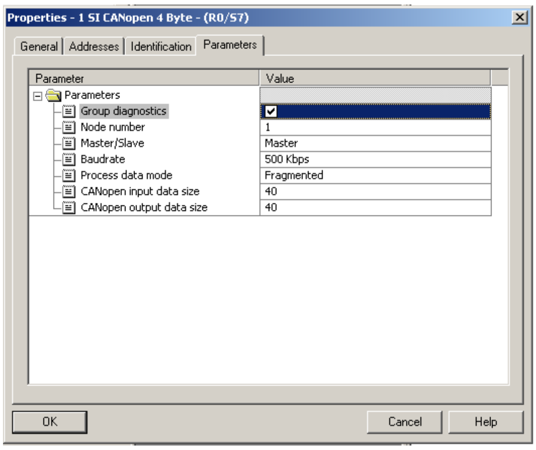


Figure 12 Group diagnostics is enabled

When the 1 SI CANopen Module is in Transparent CAN mode, the OB82 is called with "External error" when an event in the 1SI CANopen module is triggered. The user can then use the CAN Status function block to determine the reason for the event indication.

The different events that can be indicated are listed in the table below. See "8 CAN Status Function Block" for details on the specific error codes that can be triggered.

If OB82 is not included in the STEP7 project, the CPU changes to STOP mode when a diagnostic interrupt is triggered and enabled.

The information in section 0 is copied from the Help-file in STEP7. For further information see the manual and help-file for STEP7.

11.1. Diagnostic Interrupt (OB82)

Description

The operating system of the CPU calls OB82 when a module with diagnostics capability, on which you have enabled the diagnostic interrupt, detects an error and also when the error is eliminated (the OB is called when the event comes and goes).

11.2. Programming OB82

You must create OB82 as an object in your S7 program using STEP7. Write the program to be executed in OB82 in the generated block and download it to the CPU as part of your user program.

You can, for example, use OB82 for the following purposes:

- To evaluate the start information of OB82.
- To obtain exact diagnostic information about the error that has occurred.

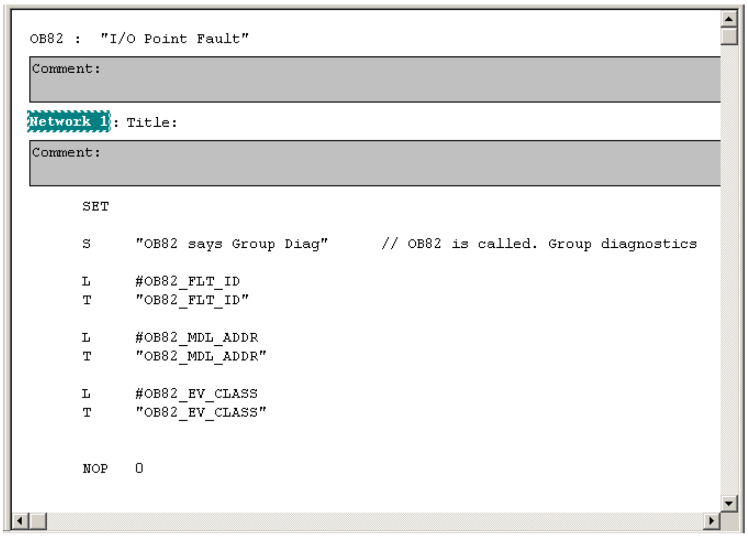
When a diagnostic interrupt is triggered, the module on which the problem has occurred automatically enters 4 bytes of diagnostic data and their start address in the start information of the diagnostic interrupt OB and in the diagnostic buffer. This provides you with information about when an error occurred and on which module.

With a suitable program in OB82, you can evaluate further diagnostic data for the module (for example, which channel the error occurred on or what error occurred). Using SFC51 RDSYSST, you can read out the module diagnostic data and enter this information in the diagnostic buffer with SFC52 WRUSRMSG. You can also send a user defined diagnostic message to a monitoring device.

You can find detailed information on OBs, SFBs, and SFCs in the corresponding Help on Blocks.

11.3. OB82 Sample Code

In the example only Fault ID, Module address and Event Class is read out in OB82 to be controlled in OB1.



```
OB82 :  "I/O Point Fault"
Comment:

Network 1: Title:
Comment:

      SET

      S      "OB82 says Group Diag"      // OB82 is called. Group diagnostics

      L      #OB82_FLT_ID
      T      "OB82_FLT_ID"

      L      #OB82_MDL_ADDR
      T      "OB82_MDL_ADDR"

      L      #OB82_EV_CLASS
      T      "OB82_EV_CLASS"


      NOP    0
```

Figure 13 OB82 code

12. Use of Variables in "Transparent CAN" Example

12.1. Explanation of Variables in VAR Table

In the example a variable table named "Transparent CAN" is saved. A short explanation about what can be seen in it is shown below.

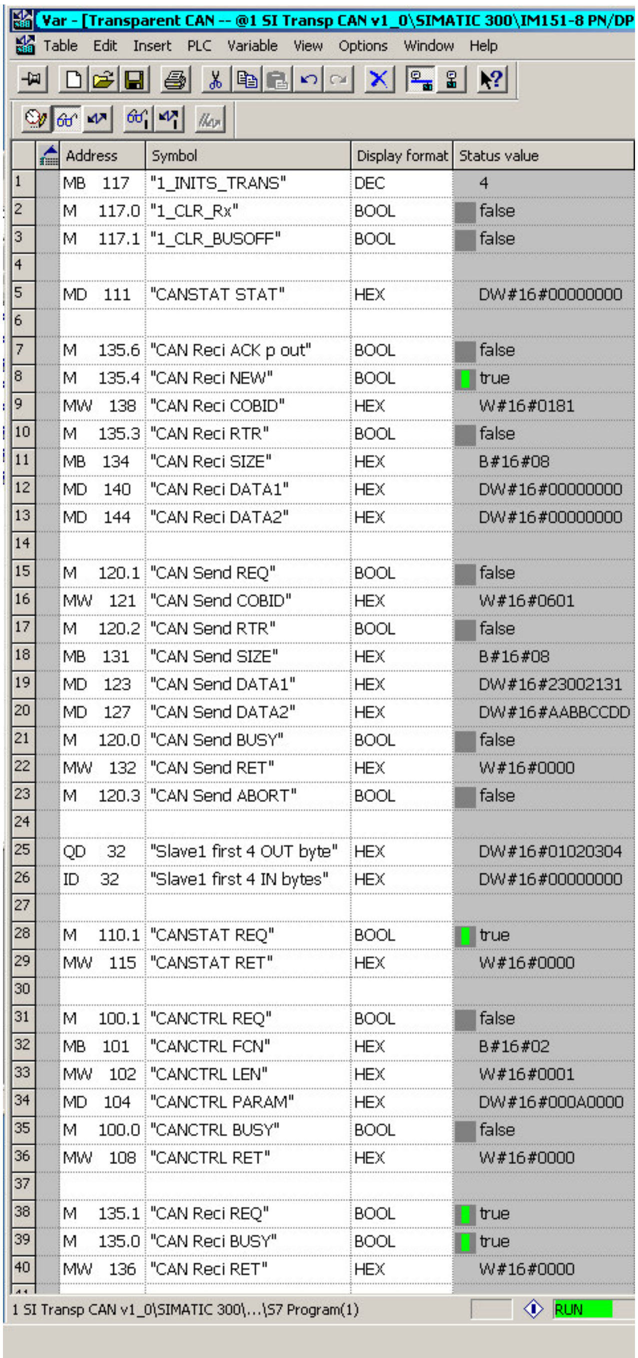

Var - [Transparent CAN -- @1 SI Transp CAN v1_0\SIMATIC 300\IM151-8 PN/DP

Table Edit Insert PLC Variable View Options Window Help

	Address		Symbol	Display format	Status value
1	MB	117	"1_INITS_TRANS"	DEC	4
2	M	117.0	"1_CLR_Rx"	BOOL	false
3	M	117.1	"1_CLR_BUSOFF"	BOOL	false
4					
5	MD	111	"CANSTAT STAT"	HEX	DW#16#00000000
6					
7	M	135.6	"CAN Reci ACK p out"	BOOL	false
8	M	135.4	"CAN Reci NEW"	BOOL	true
9	MW	138	"CAN Reci COBID"	HEX	W#16#0181
10	M	135.3	"CAN Reci RTR"	BOOL	false
11	MB	134	"CAN Reci SIZE"	HEX	B#16#08
12	MD	140	"CAN Reci DATA1"	HEX	DW#16#00000000
13	MD	144	"CAN Reci DATA2"	HEX	DW#16#00000000
14					
15	M	120.1	"CAN Send REQ"	BOOL	false
16	MW	121	"CAN Send COBID"	HEX	W#16#0601
17	M	120.2	"CAN Send RTR"	BOOL	false
18	MB	131	"CAN Send SIZE"	HEX	B#16#08
19	MD	123	"CAN Send DATA1"	HEX	DW#16#23002131
20	MD	127	"CAN Send DATA2"	HEX	DW#16#AABBCCDD
21	M	120.0	"CAN Send BUSY"	BOOL	false
22	MW	132	"CAN Send RET"	HEX	W#16#0000
23	M	120.3	"CAN Send ABORT"	BOOL	false
24					
25	QD	32	"Slave1 first 4 OUT byte"	HEX	DW#16#01020304
26	ID	32	"Slave1 first 4 IN bytes"	HEX	DW#16#00000000
27					
28	M	110.1	"CANSTAT REQ"	BOOL	true
29	MW	115	"CANSTAT RET"	HEX	W#16#0000
30					
31	M	100.1	"CANCTRL REQ"	BOOL	false
32	MB	101	"CANCTRL FCN"	HEX	B#16#02
33	MW	102	"CANCTRL LEN"	HEX	W#16#0001
34	MD	104	"CANCTRL PARAM"	HEX	DW#16#000A0000
35	M	100.0	"CANCTRL BUSY"	BOOL	false
36	MW	108	"CANCTRL RET"	HEX	W#16#0000
37					
38	M	135.1	"CAN Reci REQ"	BOOL	true
39	M	135.0	"CAN Reci BUSY"	BOOL	true
40	MW	136	"CAN Reci RET"	HEX	W#16#0000

1 SI Transp CAN v1_0\SIMATIC 300\...\S7 Program(1) RUN

- MB117 holds the state for Network 10 in the PLC program. Set M117.0 to clear Rx-buffer. Set M117.1 to reset CAN transceiver after BUS OFF failure
- Line 5 holds response from CAN Status Block
- Line 7-13 holds CAN Receive data:
  M135.6: Set bit to acknowledge present data
  M135.4: New, unacknowledged data
  Line 9-13 CAN-frame data
- Line 15-23 holds CAN Send data:
  M120.1: Set to send a CAN-frame
  Line 16-20 CAN-frame data

  M120.0: CAN Send Busy
  MW132: CAN Send RET value
  M120.3: Set to abort CAN Send.
- Line 25 and 26 holds first four OUT- and IN-bytes in CANopen slave (1)
- (Line 28 and 29 holds CAN Status REQ and RET values.)
- (Line 31 to 36 holds CAN Control data)
- (Line 38 to 40 holds CAN Receive data)

12.2. CAN-log

Variables in the CAN Log with a description for each line.

No.	Dir	Id	Rtr	Len	Data	Stored Frame No. In Receive Buffer	Description
1	Rx	702	0	1	00	-	Bootup message from node 2
2	Rx	701	0	1	00	-	Bootup message from node 1
3	Rx	0	0	2	01 00	-	Go to Operational from Transparent CAN node
4	Rx	181	0	8	00 00 00 00 00 00 00 00	1	First of four default TPDO:s in the CANopen slave, node 1
5	Rx	281	0	8	00 00 00 00 00 00 00 00	2	Default TPDO no. 2
6	Rx	381	0	8	00 00 00 00 00 00 00 00	3	Default TPDO no. 3
7	Rx	481	0	8	00 00 00 00 00 00 00 00	4	Default TPDO no. 4
8	Rx	181	0	8	01 02 03 04 00 00 00 00	5	New data in TPDO 181 from node 1
9	Rx	601	0	8	23 00 21 31 AA BB CC DD	-	SDO Request from Transparent CAN node
10	Rx	581	0	8	60 00 21 31 00 00 00 00	6	Response to SDO Request from node 1
11	Rx	601	0	8	23 00 21 31 AA BB CC DD	-	SDO Request from Transparent CAN node
12	Rx	581	0	8	60 00 21 31 00 00 00 00	7	Response to SDO Request from node 1
13	Rx	601	0	8	23 00 21 31 AA BB CC DD	-	SDO Request from Transparent CAN node
14	Rx	581	0	8	60 00 21 31 00 00 00 00	8	Response to SDO Request from node 1
15	Rx	601	0	8	23 00 21 31 AA BB CC DD	-	SDO Request from Transparent CAN node
16	Rx	581	0	8	60 00 21 31 00 00 00 00	9	Response to SDO Request from node 1
17	Rx	601	0	8	23 00 21 31 AA BB CC DD	-	SDO Request from Transparent CAN node
18	Rx	581	0	8	60 00 21 31 00 00 00 00	10	Response to SDO Request from node 1

12.3. Data in Digital Input/Output Blocks

It is possible to generate TPDOs from the CANopen slave node 1 and to see received data in the CANopen slave node 1, via the digital Input/Output modules:

The first byte in the received CANopen area of the slave, IB32, is copied to the digital output module in the PLC-rack, QB1.

The digital input module in the PLC-rack, IB2, is copied to byte number eight in the transmitted CANopen area of the slave, QB39.