SilverLode™ CANopen®
User Manual

Revision 1.5
22 July 2009
For QuickControl Rev 4.6

Table of Contents

Trademarks

® QuickControl® and QCI® are Registered Trademarks of QuickSilver Controls, Inc. SilverLode™, SilverNugget™, SilverDust™, PVIA™, QuickSilver Controls™, and AntiHunt™ are trademarks of QuickSilver Controls, Inc.

CANopen® and CiA® are registered community trade marks of CAN in Automation e.V.

Copyright

The SilverLode servo family's embedded software, electronic circuit board designs, embedded CPLD logic, and this User Manual are Copyright © 1996-2006 by QuickSilver Controls, Inc.

Chapter 1 - Getting Started

This chapter will get take you through the basics of using CAN on the SilverDust controller/driver including register and I/O sharing.

Hardware Setup

There are four physical connections for the CAN bus:

CAN V+ (Power in DC 7V-24V)
CAN V- (Power Ground 0V)
CAN H (CAN High)
CAN L   (CAN Low)
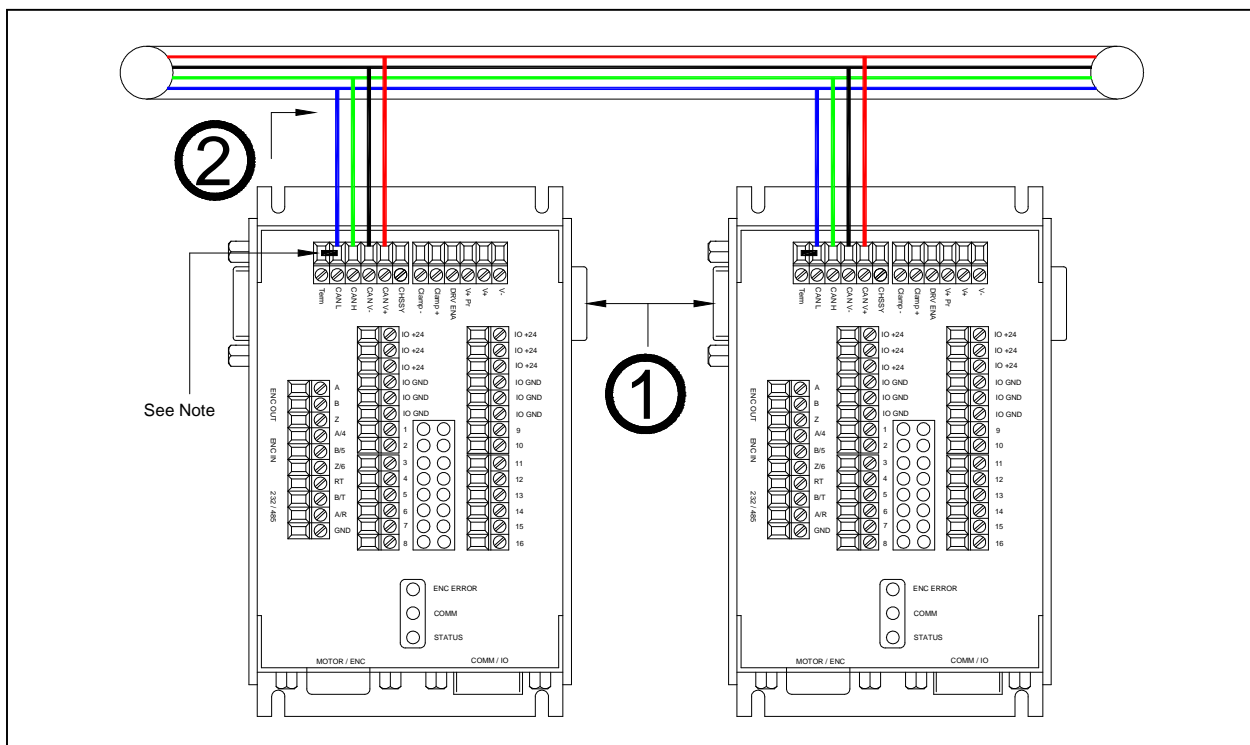
1) If all QCI-D2-IGB units are within a few feet of each other (or on the same DIN rail), connect the two DB-9 together to form the CAN bus. Please note that power (**24V max**) must be provided to at least one unit through the green terminal block. In addition, termination should only apply at the end of the bus. In this case, there are only two units on the bus, so the terminations are on both of the units.

2) The second option is wiring all the connections through the green terminals. There are four physical connections on the QCI-D2-IGB. QuickSilver recommends using CAT 5 twisted pair cables commonly referred to as Ethernet cable. CAT 5 cables are inexpensive, rugged, reliable, and are available in almost every local electronic store. They also are twisted pairs with controlled impedances and relatively low capacitances.

3) A 5th pin is provided to connect a CAN termination. This is jumpered to CAL_L only at the far ends of the run.

4) QCI-D2-IG8 controllers have three physical connections for CAN. Wires may be landed on the 5 pin screw terminal connectors on the front panel.
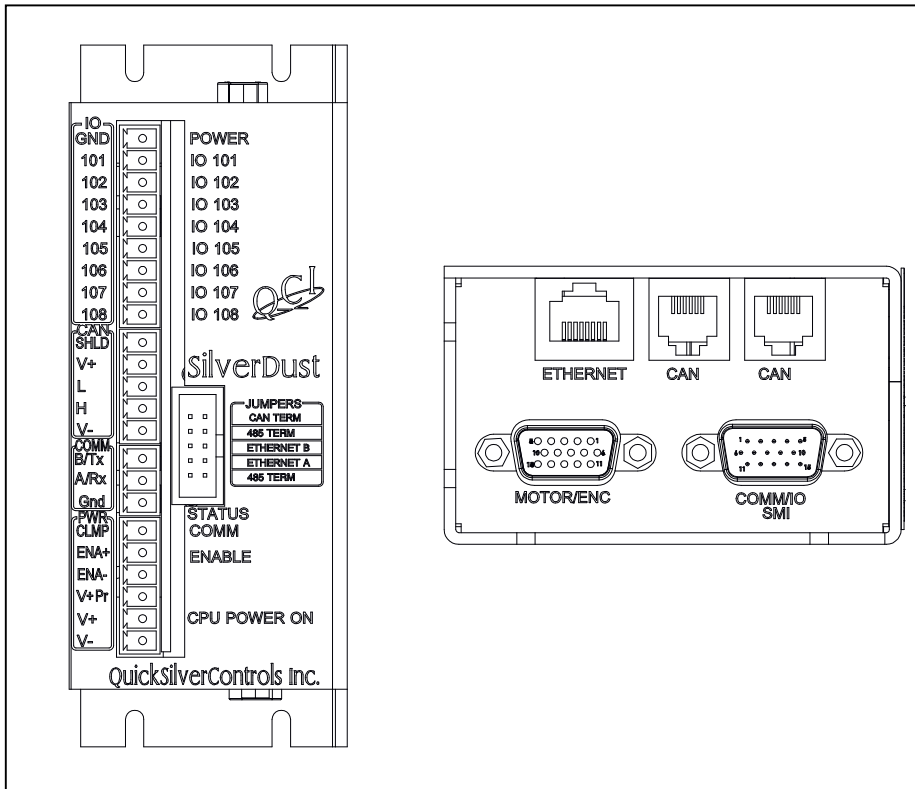
These pins are labeled as:
SHLD (Shield)
V+   (12-24v)
L   (CAN_L)
H   (CAN_H)
V-   (0v)

Only L, H, and V- are needed for the IG8 as an isolated
CAN power is derived from the processor/driver power input. The Shield and V+ signals are provided to power the RJ12 connectors, but are not otherwise used in the IG8.

All 5 signals are also connected to the RJ12 connectors on the bottom of the unit.

These connectors provide for easy daisy-chaining of the QCI-D2-IG8 units via untwisted RJ12 patch cords. These are available from QCI. This unit also has CAN status LEDs and CAN address and Baud Rate switches. See QCI-DS-018 for details.

(BP2) CAN Daisy Chain Interface

Pinout

1 CAN SHIELD
2 CAN V+
3 CAN H
4 CAN L
5 CAN GND
6 CAN SHIELD
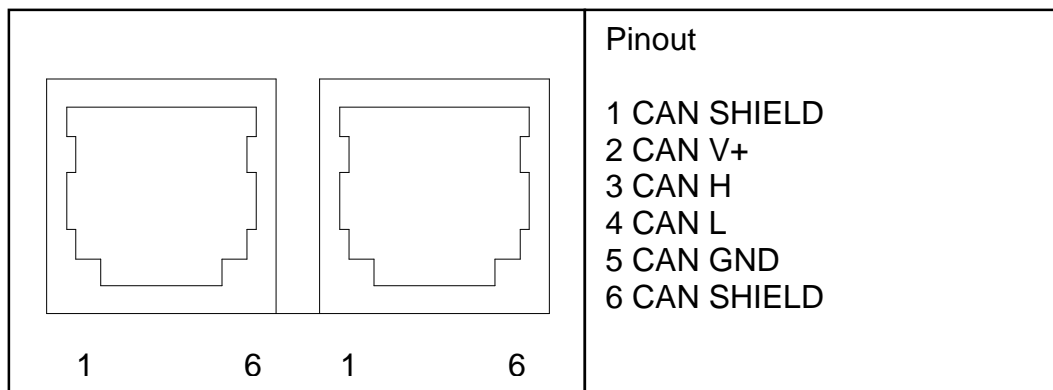
5) The CAN signals for the QCI-D2-IGB and the QCI-D2-IG8 are galvanicly isolated from the other controller signals. The CAN signals for the QCI-D2-MG-C are NOT isolated, rather the CAN transceiver is powered from the local 5V supply. The CAN transceivers used are internally protected to +/- 80v. This configuration allows deployment of CAN within smaller systems at minimal cost.

6) The QCI-D2-MG provides only three, non-isolated, CAN connections: CAN_L, CAN_H, and GND. These signals may be connected to isolated CAN signals, or to other non-isolated CAN signals if the Ground signals are common with in the system.

CAN_L
CAN_H
GND
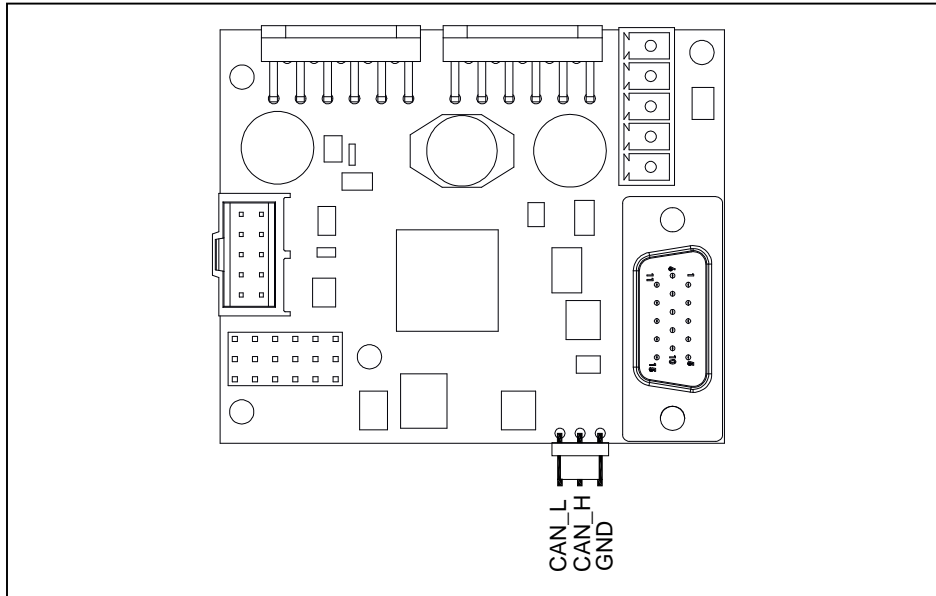
The power for the CAN is derived from the local +5V supply, thus no extra power source for CAN is required.

No provision for onboard CAN termination is provided. The user must provide a 120 ohm, ¼ to ½ watt termination resistor at each end of the CAN run.

QuickControl and CANopen®

QuickControl 4.4 or greater required. QuickControl 4.4 can be found on QuickSilver Controls website, www.QuickSilverControls.com under the Software section.

Combo-Commands

Combo-Commands were introduced in QuickControl Rev 4.4. Combo-Commands provide a macro like program construct in which user selections cause the parameters of multiple native commands to be simultaneously edited. All native commands have three letter acronyms, where as the Combo-Commands have four letter acronyms, to allow for easy recognition. The combo commands may be expanded to see the underlying commands by right clicking on the Combo-Command and selecting Expand from the pop up menu. They may be restored to a single line by the same process. The individual commands are "greyed out" as they may not be edited individually. However, they may be copied and pasted in to a program by selecting only the individual commands (and not the Combo-Command) and performing a copy and then a paste operation. At this point, they are no longer associated with the Combo-Command and may be individually edited.

Slave, Master, Peer (Network Structure)

The QuickSilver CANopen implementation supports both Master-Slave configuration, in which a "Master" device configures the other "Slave" devices via the CAN bus, as well as Peer-To-Peer operation in which each node configures itself. These modes may also be mixed, with some nodes "self configuring" while other nodes are remotely configured, as could be the case with the use of CANopen encoders or I/O blocks.

CAN Initialization

Each unit should be configured using the Initialization Wizard and the "Factory Default Initialization – CAN.qcp" initialization file. This file has two extra commands, CAN Identity (CID) and CAN Baud Rate (CBD). By default, it configures each node to communicate at 1Mbit/sec using the same CAN ID as Unit ID (see below).
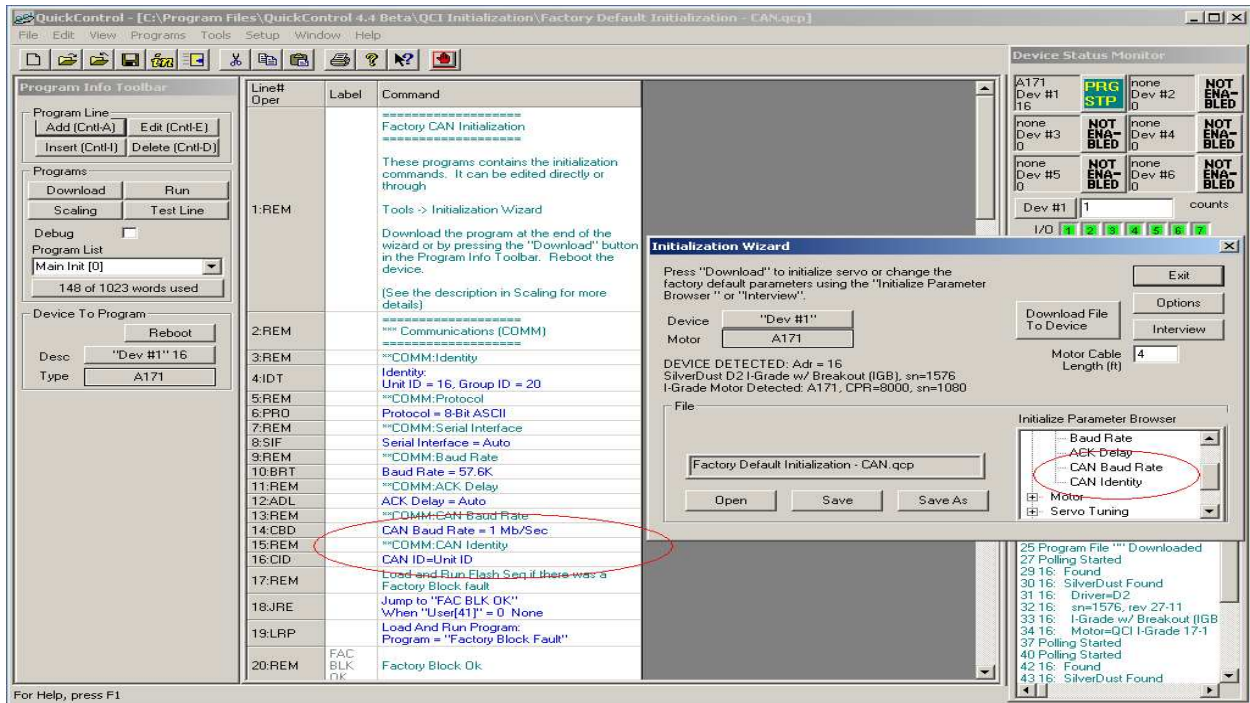
Factory Default Initialization – CAN.qcp

Details: CAN Identity (CID)

Every unit on the network must have a unique CAN ID. A CAN ID of zero forces the CAN ID to be the same as the serial communication Unit ID as set by the Identity (IDT) command. For example, if the IDT command sets Unit ID to 16, setting the CAN ID to 0, will force the CAN ID to match the Unit ID of 16.
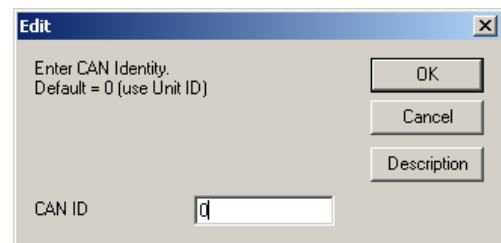
Note: Unit ID in the IDT command is addressable from 1-255. If Unit ID is set to 128+, and CAN ID=0, the CID command will error out.

Details: CAN Baud Rate (CBD)

CAN networks can operate up to 1 megabit per second (1 Mb/sec). The trade off for lower baud rate is bus length. QuickSilver's default baud rate is 1Mb/sec
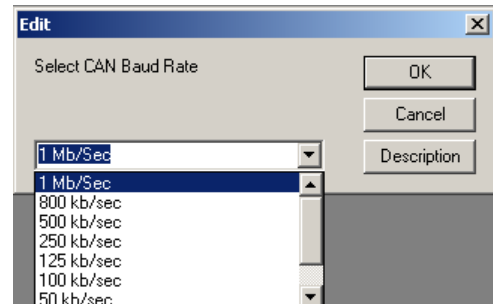
Register Sharing Peer-To-Peer

User registers may be easily shared across the CAN network in a multi-axis application. There are two ways to setup register sharing, one is Peer-To-Peer and the other is Master-Slave (discussed in the next section).

Peer-To-Peer Network Diagram
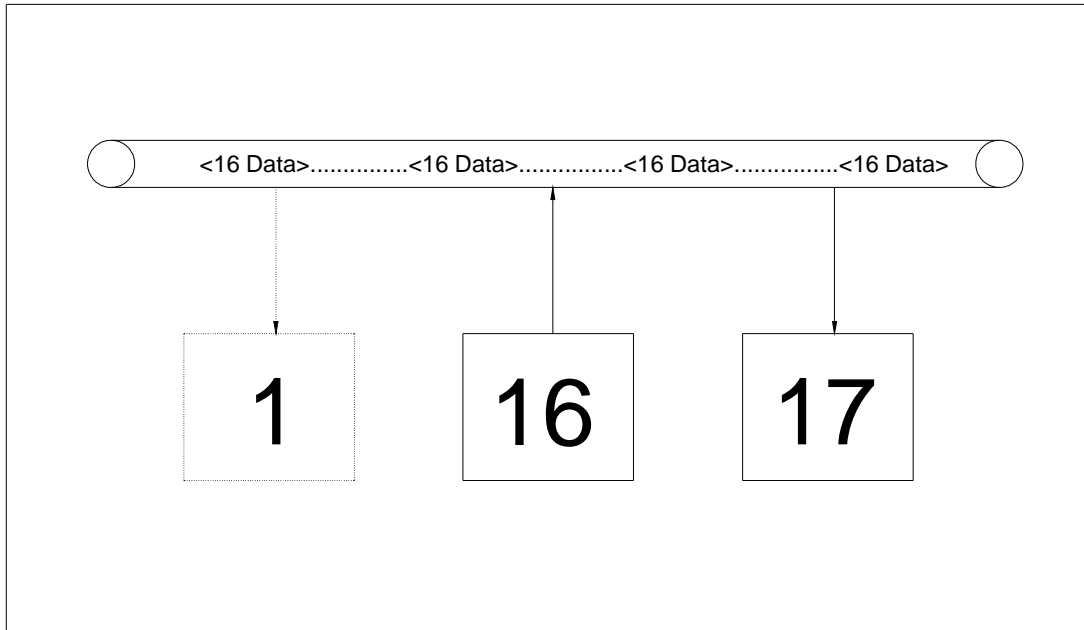
Transmitting

In Peer-To-Peer, each unit will "locally" configure itself to either transmit or receive. In this example, unit 16 is configured to transmit its register onto the bus. Once configured, any unit on the bus may receive the data. This is done using the CAN Transmit Register, Local (CTRL) Combo-Command.

Each device has four independent communication channels to transmit data. Once configured, the data register will transmit data "continuously" onto the bus. In CANopen terminology, this is called a Transmit Process Data Object (TPDO). See TPDO section for more details.
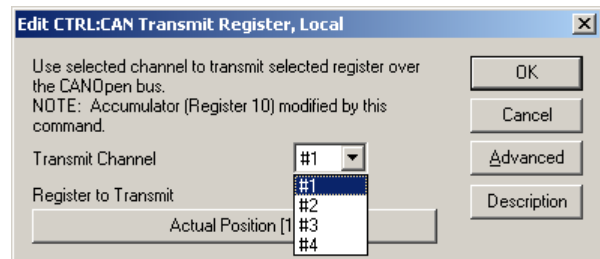
Once data transmission is configured, use the CAN Set NMT State, Local (CNL) command to set the Network Management (NMT) state to Operational. This allows the unit to start transmitting data.
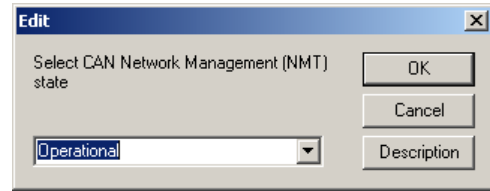
Example Program: Unit 16 transmits its actual position onto the bus. See Diagram above.

Line# Oper	Label	Command
1:REM		Broadcast the Actual Position
2:CTRL		CTRL:CAN Transmit Register, Local Tx Channel: #1 Register: Actual Position[1]
15:CNL		CAN NMT State = Operational

Receiving (Mapping)

One or more units on the CAN bus may receive or map the transmitted register into any local user Register using the CAN Register Map, Local (CRML) Combo-Command.
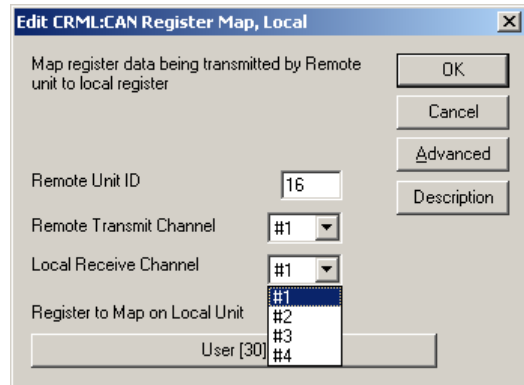
The local receive channels are independent of the local transmitting channels. In this example, unit 17 will continuously receive data from the unit 16. In CANopen terminology, this is called a Receive Process Data Object (RPDO). See RPDO section.

Example Program:

Line# Oper	Label	Command
1:REM		Receive Node 16 channel 1 data into register 30
2:CRML		CRML:CAN Register Map, Local Remote Unit ID: 16 Remote Tx Channel: #1 Local Rx Channel: #1 Local Register: User[30]
10:CNL		CAN NMT State = Operational

Note: If unit ID 1 wants to receive the same data transmitted by unit 16, repeat this process.

At this point, the unit sourcing the data need merely modify its local register to cause the same data to appear in the remote node's mapped register. The register number of the source (producer) is independent from the receiving (consumer) node register. For example, the actual position of the producer node may be broadcast, with the consumer node mapping it to register 30.

Advanced TPDO and RPDO
CAN Transmit Register, Local (Advanced)
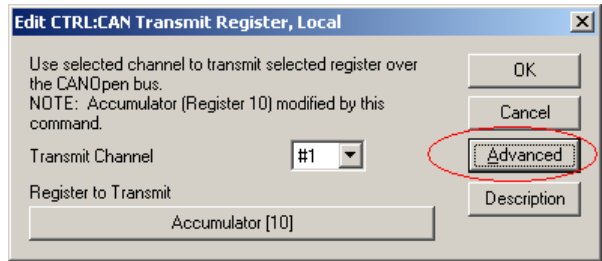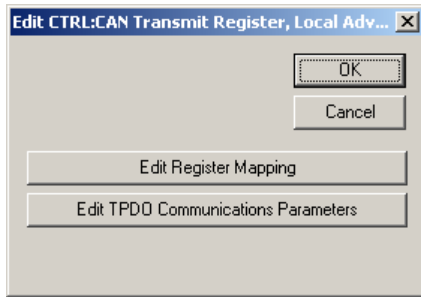
The Advanced option has two parameters:
- Edit Register Mapping
- Edit TPDO communication Parameters.

Edit Register Mapping Option
This advanced function allows user to select a second register for the same communication channel. One TPDO channel can transmit up to two registers at the same time.

By default, the second register transmission is disabled. User must enable the second channel and select the desired register.
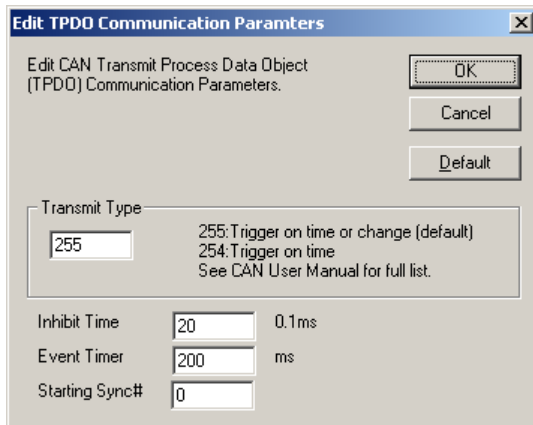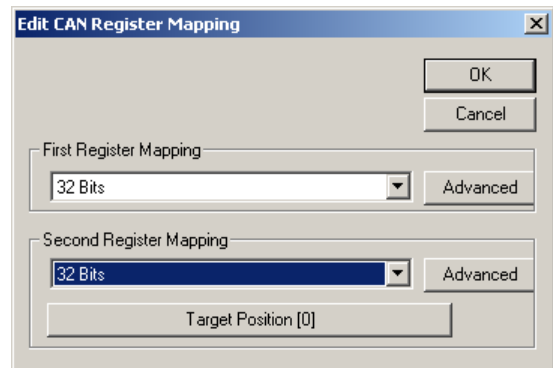
TPDO Communication Parameters
This advanced function allows user to select the type and frequency of transmission. See Process Data Objects in Chapter 3 for details.

Register Sharing Master-Slave

Master-Slave is the second option in setting up the CAN network. The advantage of Master-Slave configuration is centralized control in large networks. Setting up each peer to transmit and receive a PDO locally is not practical in a large network in terms of software management because there are too many programs to keep track and debug. In Master-Slave configuration, the master remotely configures the TPDO and RPDO on the other nodes. There is only one program on the master unit, which makes debugging easier.
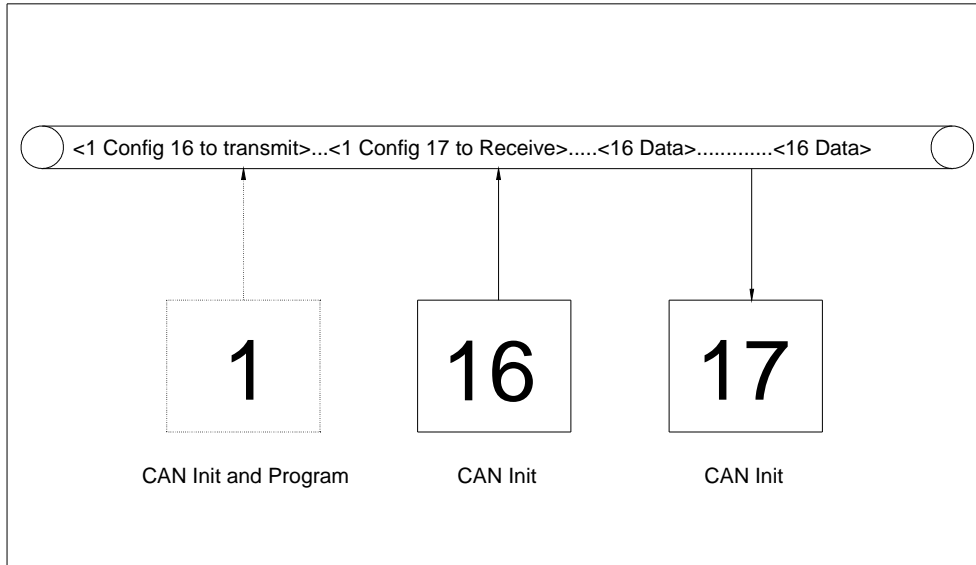
In this example, unit 1 will configure unit 16 to transmit it's Actual position register onto the bus using a TPDO. Then unit 1 will configure unit 17 to map unit 16's Actual Position register into unit 17's local register using an RPDO.

Programming Unit 16 from Unit 1

Before any unit can be configured remotely, a connection must be established. This is done using the CAN Connect to Remote (CCTR) Combo-Command.
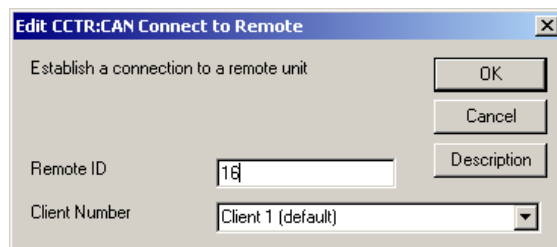
Once the CCTR command is executed, Unit 1 can configure Unit 16 to transmit its register using the CAN Transmit Register, Remote (CTRR) Combo-Command. CTRR is just like the local version, CTRL except it configures the "connected" remote unit to transmit a register.
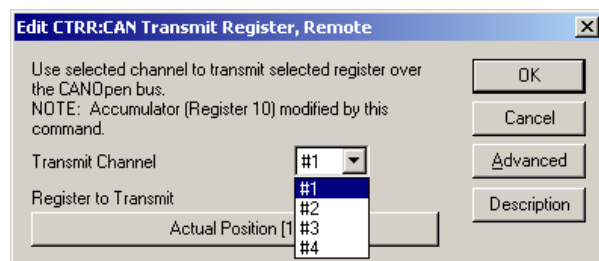
Like the Peer-to-Peer example, the final step is to put the remote unit into NMT Operational state. Master units put slaves into Operational state using the CAN NMT State, Remote (CNR) command.

Configuring any remote unit to transmit a TPDO is a three step process. First, connect to the remote unit. Second, configure the TPDO. Third, put the remote unit into operational mode.

Programming Unit 17 from Unit 1

1) Use CCTR to connect to Unit 17.
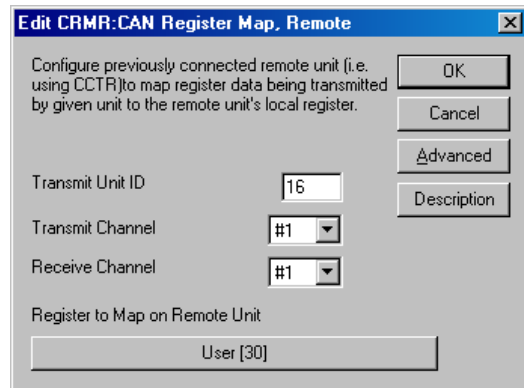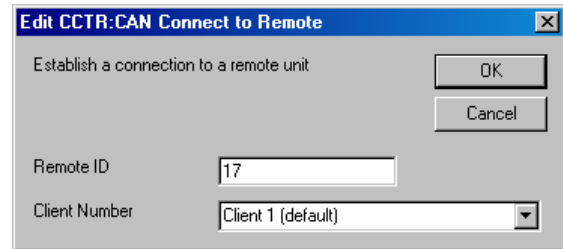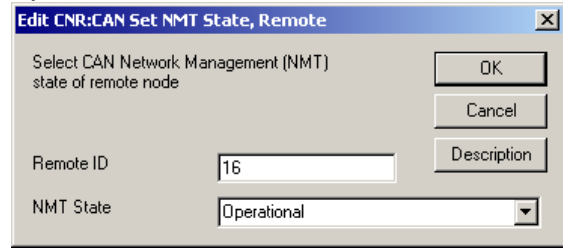
2) Use CRMR to map 16's Transmit Channel #1 data to 17's register 30 through 17's Receive Channel #1.
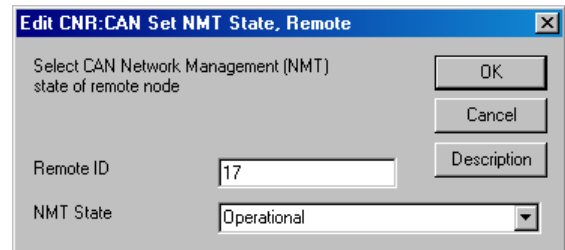
3) Set 17 to Operational using CNR.

Now, unit 16 will transmit its actual position register onto the bus. Unit 17 will receive unit 16 TPDO into register 30 through its RPDO. The entire configuration was done through unit ID 1. See diagram above.

Example program

Line# Oper	Label	Command
1:REM		Connect to unit 16
2:CCTR		CAN Connect to Unit 16
6:REM		Setup 16 to transmit Register 1 onto the bus.
7:CTRR		CTRR:CAN Transmit Register, Remote Tx Channel: #1 Register: Actual Position[1]
21:REM		Put unit 16 into operational mode
22:CNR		ID=16:CAN NMT State = Operational
23:REM		Connect to unit 17
24:CCTR		CAN Connect to Unit 17
28:REM		Setup 17 to receive 16 actual position into 17 local register 30.
29:CRMR		CRMR:CAN Register Map, Remote Tx Unit ID: 16 Tx Channel: #1 Rx Channel: #1 Register: User[30]
38:REM		Put unit 17 into operational mode
39:CNR		ID=17:CAN NMT State = Operational
40:REM		End of program

Output Sharing

The following diagram and procedure shows how to have unit 16 share 17's outputs.

OUTPUT

16   17   Outputs
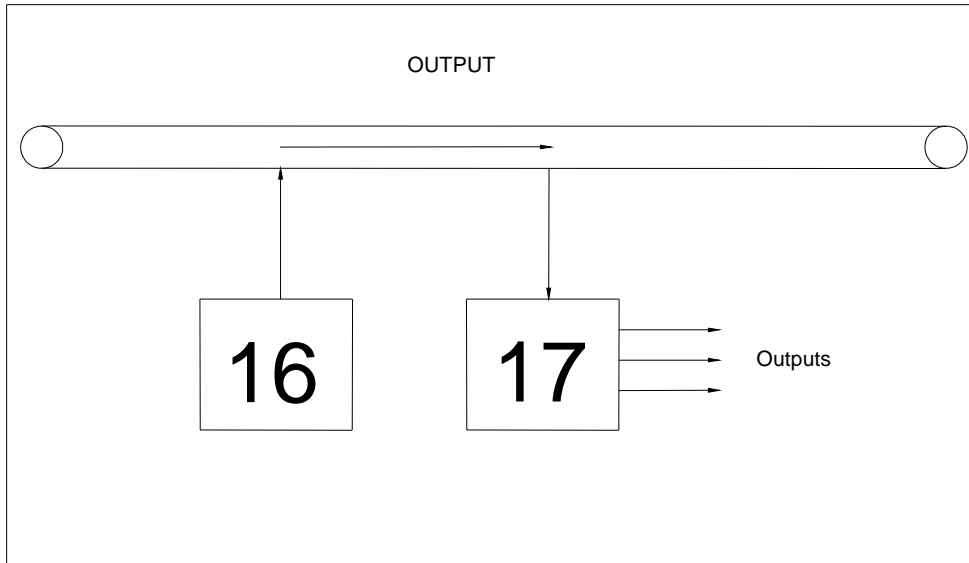
1) From unit 16, use CCTR to connect unit 17.

Edit CCTR:CAN Connect to Remote

Establish a connection to a remote unit   OK   Cancel

Remote ID   17

Client Number   Client 1 (default)

Dialog

Access remote CAN Data Dictionary.
NOTE: Assumes CCTR used to
connect to remote.

Description   OK
Advanced   Cancel

General | Remote Output | Remote Register Access |

Remote Output

Set or Clear remote output #101-116.

Remote Output #110

Mode

○ Output Clear (LOW)
◉ Output Set (HIGH)

2) Use the CAN Dictionary Access, Remote (CDR) command (Remote Output tab) to clear or set a remote unit's output.

For an example program see "QCI Examples\CAN\CDR Remote Output.qcp" in the QuickControl folder.

Input Sharing

Any device may share its extended inputs (i.e. #101-116) with everybody else on the CAN network.   A unit shares a remote unit's inputs by mapping a specific register on the remote unit to a specific local register. Once mapped, the "Remote Inputs" can be used in many commands just like local inputs (see below). For details on the specific registers 238 and 199, see Input Sharing in Chapter 4.

The following diagram and procedure shows unit 16 sharing unit 17's inputs.

INPUTS

<17 TPDO Register 238>....<16 RPDO Register 199>

Map I/O Register 199

Map I/O commands

16

17
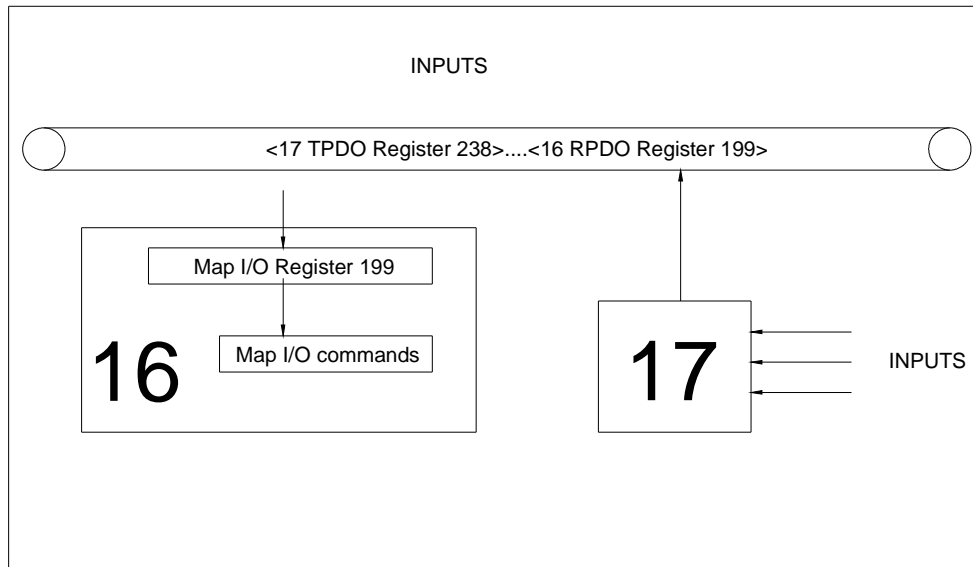
INPUTS

Unit 16 Sharing Unit 17's Inputs

1) Unit 17 program uses CTRL to share the upper word of register 238 (extended I/O input states) with everybody on the CAN bus.

Edit CTRL:CAN Transmit Register, Local A...
- OK
- Cancel
- Edit Register Mapping
- Edit TPDO Communications Parameters

2:CTRL	CTRL:CAN Transmit Register, Local Tx Channel: #1 Upper 16 Bits of Register: XIO |XIO Output[238]
15:CNL	CAN NMT State = Operational

Edit CTRL:CAN Transmit Register, Local

Use selected channel to transmit selected register over the CANOpen bus.
NOTE: Accumulator (Register 10) modified by this command.

- OK
- Cancel
- Advanced
- Description

Transmit Channel   #1

Register to Transmit
XIO |XIO Output [238]

Edit CAN Register Mapping

- OK
- Cancel

First Register Mapping
Upper 16 Bits   Advanced

Second Register Mapping
Disabled   Advanced

Target Position [0]

2) Unit 16 program uses CRML to map unit 17's remote inputs to lower word of register 199.

2:CRML	CRML:CAN Register Map, Local Remote Unit ID: 17 Remote Tx Channel: #1 Local Rx Channel: #1 Lower 16 Bits of Local Register: User[199]
10:CNL	CAN NMT State = Operational
11:WBE	Wait On Bit Edge Until "Remote Input #1" is LOW/FALSE
12:MRT	Move 8000 counts @ ramp time=100 mSec total time=1000 mSec

Edit CRML:CAN Register Map, Local Advan...

- OK
- Cancel
- Edit Register Mapping

Receive Type   254   0=Sync
254=async (default)

Edit CRML:CAN Register Map, Local

Map register data being transmitted by Remote unit to local register

- OK
- Cancel
- Advanced
- Description

Remote Unit ID   17

Remote Transmit Channel   #1

Local Receive Channel   #1

Register to Map on Local Unit
User [199]

Edit CAN Register Mapping

- OK
- Cancel

First Register Mapping
Lower 16 Bits   Advanced

Second Register Mapping
Disabled   Advanced

Accumulator [10]

Using Remote Inputs

In Move Commands

All move commands can stop on remote unit's input. Under the "Advance" option of any move command, the user may select any remote input.
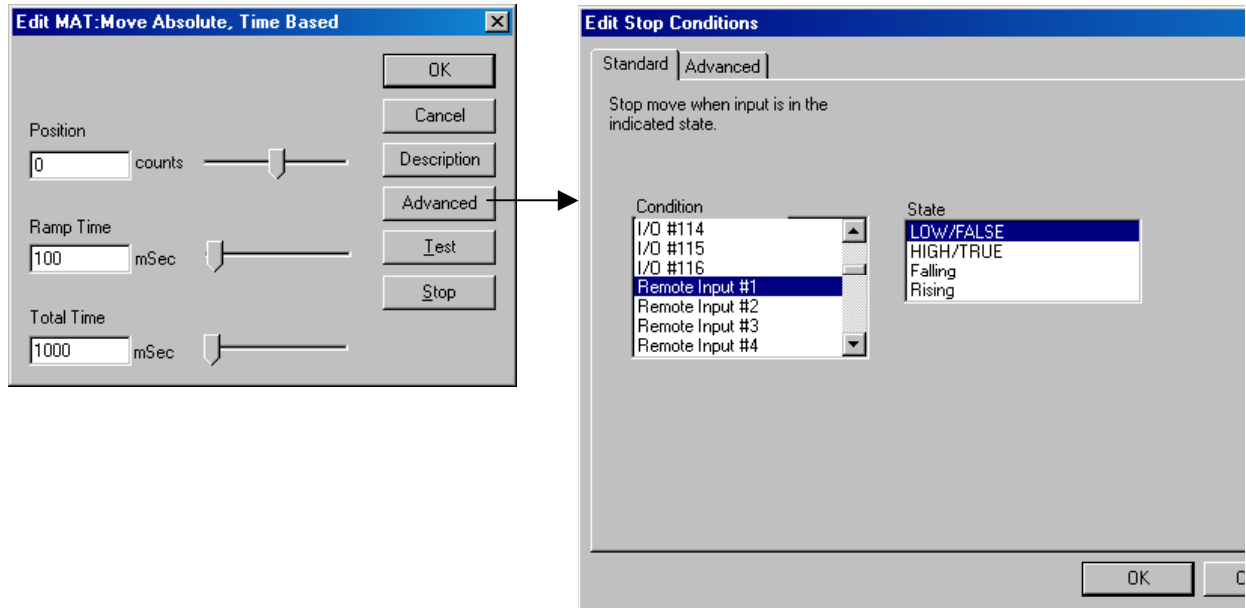
Flow Commands
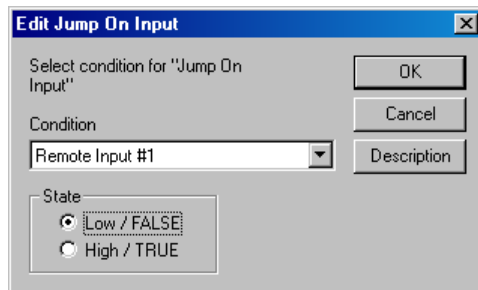
The following Program Flow commands can use remote inputs:

- Jump On Input (JOI)
- Program Call On Input (PCI)
- Program Return On Input (PRI)
- Wait On Bit Edge (WBE)
- Wait On Bit State (WBS)

This is the end of the Getting Started chapter. For users who want to fully understand CAN structures and how it really works "under the hood", please continue….

Chapter 2 – Introduction to CAN

CAN Capabilities

CAN provides a robust networking capability, and has been in use for more than 15 years. Originally designed for the harsh under-hood car environment, it has multiple error detection and correction methods built in to provide predictable, robust, and virtually error-free communications for industrial control. The network allows multi-master, multi-destination communications, with communication speeds up to 1Mbit per second. Each frame of data sent includes a message identifier; this is used by all of the receiving nodes to determine if they are configured to react to the frame. This allows data to be sent from one to many nodes, with all nodes receiving the message at the same point in time.

CAN uses a message arbitration scheme rather than a message collision scheme to decide which node on the network is allowed to transmit its data in a particular time slot. The message assigned the highest priority goes first with no impact on its sending time, even if other nodes are attempting to send lower priority messages. The lower priority messages then follow, highest priority to lowest priority.

Using the CAN framework for communications, nodes may share registers, with any change in the shared register automatically reflected in multiple other nodes. A master node may control other nodes – even resetting them. Error conditions may be conveyed between nodes on occurrence rather then requiring constant polling to determine error conditions. Register read and write operations are allowed between nodes. Data may be exchanged based on update times, synchronous events, upon changes in data, or combinations of these, with provisions for minimum and maximum update rates, all operating modally in the background without user program intervention. This allows cam following operations to use a CANopen encoder, or another node's target or position to control another node, without extra step-and-direction wiring.

It is also easy to implement automatic Heartbeat monitoring, in which each node produces a timed heartbeat signal, and up to eight other nodes are monitored for their presence as well as critical changes in state. A consistent system time may be distributed across multiple nodes, with each other node frequency locking their local time to the designated master node to eliminate the effects of differences in oscillator frequencies.

CAN

The CAN network and its principles of operation were originally defined by Bosch: (www.can.bosch.com), standardized by the International Organization for Standardization as ISO11898

CAN Physical Layer

CANopen defines a Physical layer having at least two nodes connected by a twisted pair data bus, having each end of the data bus terminated with 120 ohms. (See data sheets for connections, power requirements, etc.) The Bus assumes one of two states at any point in time, Passive or Dominant. Passive state exists when no drivers are

active on the bus, causing the differential voltage to be pulled close to zero due to the action of the terminating resistors. The Dominant state exists when one or more of the bus drivers are driving the bus; in the Dominant state the CAN_H line is driven high (approximately 4v typically) while the CAN_L line is driven low (approximately 1v). The state of the bus thus assumes a level which is the logical "OR" (active low logic) of the transmitters on each of the nodes of the bus – that is it is in the Passive state if all of the nodes are transmitting a passive state (driver inactive) and it is in the dominant state if any of the nodes are transmitting a dominant state (driver active). Each node monitors the state of the bus, both when listening and when transmitting. The Dominant State Represents a "0" level, while the Recessive State represents a "1" level.

CAN Bus Termination

120 ohm termination resistors are required at each end of the bus. These terminating resistors are required even for very small networks, as the drivers only drive in the *Dominant* state, while the line terminators return the network to the *Passive* state levels.

 The wiring between nodes should be twisted pair 120 ohm impedance wire, preferably shielded. The two CAN_H and CAN_L should be one pair of wires, while CAN_V+ and CAN_V- should be on a separate set of wire. The CAN enabled SilverLode controllers include a 120 ohm terminating resistor that may be connected by wiring between CAN_L and TERM; only the units at the ends of the bus should be terminated.

On the CAN enabled SilverLode controllers, the CAN bus signals CAN_H and CAN_L should be twisted pair wiring, preferably shielded. The signals may be connected via the topside terminal strips or via the 9-pin D-Sub connectors. It is common practice to feed the CAN power from approximately the center of the network, along with the CAN signals. Use only one connection from the power source to the CAN power bus to prevent ground loops which may degrade signals and increase EMI emissions and susceptibility.

WARNING: The 9 pin CAN/COMM connectors carry both Communications and CAN signals and **do not** follow the standard CANopen signal pin-out convention. See the documentation before connecting anything to these connectors.

CANopen Bus Length versus Baud Rate

Maximum cable length is dependent upon the baud rate and upon the number of nodes and wire gauge (See DR303 V 1.3). CANopen defines the following rates versus bus lengths (we also support 100kbps):

Baud Rate	Max Bus length
1 Mbps	25 m
800 kbps	50 m
500 kbps	100 m
250 kbps	250 m
125 kbps	500 m
50 kbps	1000 m
20 kbps	2500 m
10 kbps	5000 m

The above Maximum bus length should include the length of all stubs on the bus, due to their loading of the bus. These individual stub lengths should be kept to less than 2% of the maximum bus length, with the sum of all of the stubs less than 10% of the maximum Bus length.

Bus Length	Length related resistance	Wire Cross-Section	Wire Gauge (approximate)	Termination resistance
meters	Milliohm/meter	Square mm	AWG	ohms
0 to 40	70	0.25 to 0.34	24 GA	124
40 to 300	<60	0.34 to 0.6	22 GA	150 to 300
300 to 600	<40	0.5 to 0.6	20 GA	150 to 300
600 to 1000	<26	0.75 to 0.8	18 GA	150 to 300

(Recommendations from DR 303-1 V1.3)

The baud rate for the Node is set via the CAN Baud Rate (CBD) command.

CAN Message Frame Structure

CAN and Message Identifiers

Each message sent across the bus is uniquely identified by a pre-assigned Communications Object Identifier or COB-ID. The COB-ID not only designates the type of communications and how its data will be handled, it also specifies the priority of the message, with the lowest numbered COB-ID's receiving the highest priority in transmission.

Each Node must be assigned a unique CAN ID in the range of 1 to 127. The CAN ID is used to build the default COB-ID values used by the frames, so a lower numbered node will be assigned higher priorities by default. The CAN ID is set using the CAN Identity (CID) command; this command assigns the ID, builds the default COB-ID values for the various objects, and then starts up the CAN background processes.

CAN Frame Structure

The CAN Frame includes:

- Start of Frame (synchronizes multiple devices to arbitrate the bus)
- Arbitration Field (contains the Communication Object Identifier or COB-ID of the Frame)
- Control Field (defines frame type and number of data bytes)
- Data Field (0 to 8 bytes of data)
- CRC Field (16 bit CRC to check for errors)
- ACK Field (Response that at least one other CAN device properly decoded the frame)
- End of Frame (Quiet time at end of frame so new Start of Frame may be detected)

A new CAN frame is permitted following a BUS IDLE period. The BUS IDLE period consists of a sufficiently long period of Recessive state to indicate that no active frame is present. Following a valid BUS IDLE, all nodes having messages to transmit assert a Start of Frame. As the bus is wire-OR, all nodes will read the bus as Dominate State. The falling edge of Start of Frame is used to synchronize all of the nodes for the Arbitration Field (COB-ID).

Priority Arbitration

During the Arbitration Field, each node drives the bus the COB-ID of the message it is sending, starting with the Most Significant Bit. Each node monitors the bus to determine the resulting state of the CAN BUS. If the Node sees the same state on the BUS as it was asserting, it is allowed to continue the arbitration the following bit cycle. If the node sees a different state of the bus, it has lost the arbitration and must wait until the next Interframe space to try again. Because the Dominant State represents a "0" level, and because the Dominant State is present on the bus when both Dominant and Recessive states are asserted by different nodes, the Node transmitting the message with the lowest COB-ID (arbitration field) wins the arbitration cycle. This is repeated for all 11 (or 29) bits of the COB-ID. The node that was sending the lowest numbered COB-ID remains active and continues to transmit the balance of the frame except for the ACK bit. The ACK bit must be provided by a different Node to indicate that the frame was

received properly. If any error active Node detects a problem with the frame, it asserts an Error Frame, which causes the sending Node to stop sending and to retry. (There are several error recovery mechanisms built into CAN, not described here to simplify the description.)

To restate, the highest priority messages need to be assigned the lowest COB-ID values, while the lower priority messages are assigned the higher COB-ID values.

Upon receipt of a valid frame, each node (other than the node that originated the Frame) then examines the COB-ID (transmitted during the Arbitration Frame) to see if it needs to act upon the frame or whether it may discard the frame.

Note: It is important that all COB-ID values are unique to prevent more than one node from winning arbitration, only to have mismatching data collide. This will result in resending of data until nodes go offline due to excessive errors.

Non-Destructive Message Arbitration Process

COB-ID/
Arbitration field

Start Bit

1 1 0 1 0 0 0 1 0 1 1
Node 1
COB-ID=68B

1 1 1 X X X X X X X X
Node 2
COB-ID=723
     Node 2 loses arbitration

1 1 0 1 0 0 1 X X X X
Node 3
COB-ID=692
     Node 3 loses arbitration

1 1 0 1 0 0 0 1 0 1 1
Bus
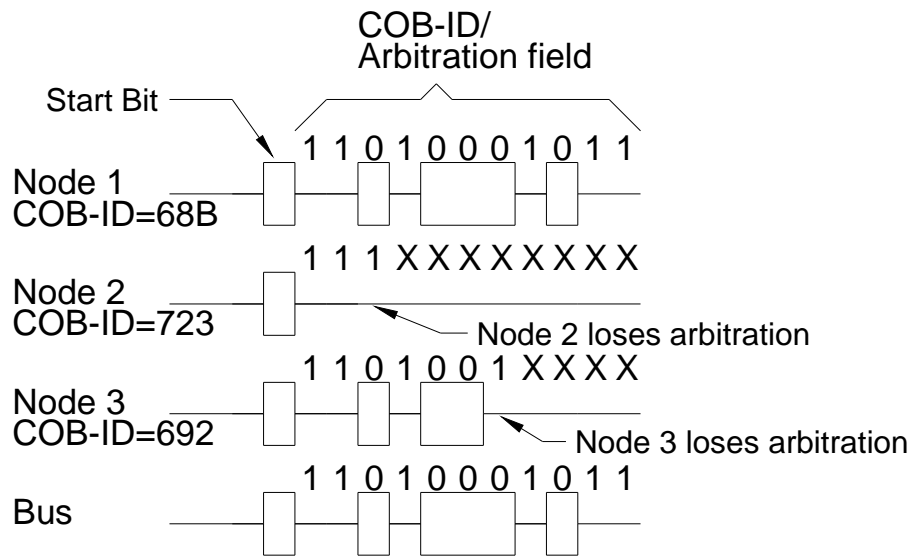
CAN Bus Frame Fields

The basic message frame (including inter-frame intermission) consists of the following bits, excluding data and bit stuffing:

Bits	Purpose
3	Intermission
1	Start of Frame (SOF)
11	Identifiers (COB-ID)
1	Remote Transmit Request
2	R0/R1 (reserved bits)
4	Data Length Count (DLC) = # of bytes
0	Data (0 to 8 bytes)
16	Cyclic Redundancy Code (CRC)
2	Acknowledge slot (ACK)
7	End of Frame (EOF)
47	Total

This represents the minimum frame size with no data payload. Eight bits of data are added for each byte of data payload. However, the CAN specification prevents more than 5 consecutive bits of the same value with in a packet (excluding the EOF and intermission time); in the case of 5 consecutive bits of the same value, a "stuff bit" of the opposite state is automatically inserted at the transmitter and removed by the receiver. The Acknowledge slot must also have a fixed Passive Guard, Active Ack, Passive Guard timing. This leaves 34 of the basic frame bits subject to bit stuffing rules, as well as all of the data bytes. Bit stuffing is data/identifier dependent.

Minimum packet size it therefore $47+8*d$ where d=number of data bytes

Maximum packet size is $47+8*d + \{34+8d-1\}/4$:

Bytes of Data	0	1	2	3	4	5	6	7	8
Minimum Packet - bits	47	55	63	71	79	87	95	103	111
Maximum Packet - bits	55	65	75	85	95	105	115	125	135
Min. Time @ 1Mbps - microseconds	47	55	63	71	79	87	95	103	111
Max. Time @ 1Mb/Sec - microseconds	55	65	75	85	95	105	115	125	135
Min. Time @ 250kbps - microseconds	188	220	252	284	316	348	380	412	444
Max. Time @ 250kbps - microseconds	220	260	300	340	380	420	460	500	540

Thus a frame carrying 0 bytes of data requires between 47us and 55us at 1Mbps baud rate, while it takes 188us to 220us for the same packet if the bus speed is lowered to a 250kbps baud rate. A frame carrying 8 bytes of data takes between 111uS and 135us vs. 444us and 540us for the same 1Mbps and 250 kbps data rates.

Example Data Frame with Two Bytes of Data

COB-ID/
Arbitration field

Control Field

Data Field

16 bit CRC
(example)

ACK
Slot

End of Frame

SOF
(Start of Frame)

1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1

RTR (1=request, 0=message

IDE (0=11 bit, 1=29 bit)

R0 (Reserved Bit)

Stuff bit

0 0   1 0

Data Length
Count (DLC)

Low Byte
29h

High Byte
52h

ACK is supplied by One
or more Other Nodes

ACK Delimiter - Must be Recessive,
or start of Error Frame

Chapter 3 – CANopen Protocol

Introduction to CANopen Communications

CANopen provides communications between sensors, controllers, drives, I/O, and other devices. This communication provides:

- Access to device and communication parameters
- Cyclical and event based process data communications
- Synchronization between devices
- Configuration
- Fault detection

Four Communication Objects (COB), each with its own characteristics are defined:

- Network Management (NMT, Heartbeat)
- Service Data Objects (SDO)
- Process Data Objects (PDO)
- Pre-defined objects (SYNC, EMCY, TIME)

CAN in Automation (CiA) is responsible for the standards and profiles that define the CANopen standard (www.CAN-CiA.org).

All of the Node Data and Configuration available to and through the CAN bus is defined in a data structured called the Data Dictionary. Each entry in the Data Dictionary is called a Data Dictionary Object, and is referenced by an Index. These Objects may be simple, such as bytes, words, long words, strings, etc., or may be complex, such as arrays or data structures. A Sub-Index is used to reference the elements of complex structures. Simple Objects use the appropriate Index with a Sub-Index = 0. Uploading (reading) and Downloading (writing) of these Data Dictionary Objects will be described in the SDO Object section, below. The object data may also be conveyed from it source node to multiple destination nodes using various PDO Objects (see below).

Network Management (NMT) Objects

Each node on the network has a NMT state machine. The NMT state machine indicates the Network state of the node.

- Resetting = Initializing and testing hardware
- Initializing Communications = Hardware OK, initializing CAN and default COB-ID's
- Pre-Operatonal = able to process NMT and SDO objects, but not PDO objects and Predefined Objects. Use this state to configure Nodes before they go active.
- Operational = able to process all object types (although some configuration changes may not be allowed).
- Stopped = an error was detected, and the Node will only respond to NMT frames.

A Node is allowed to either change its NMT state itself (Master or Peer), or to wait until a Master node changes its state.

The Master node may change the state of any other node by use of the Node Control Protocol. The NMT operation may be used to cause the Node to transition to Pre-

Operational, Operational or Stopped state; it may also reboot the node, or to cause the node to reset its communication parameters their initialization state.

The local Node NMT state may be set via the CAN Set NMT State, Local (CNL) command.

If the SilverLode is operating as the master, other nodes NMT state may be set via the CAN Set NMT State, Remote (CNR) command.

See Starting up CAN for details.

Monitoring NMT State Status

Each node, upon completion of Initialization, transmits a Boot-Up Frame, and then transitions to PreOperational state. This Boot-Up frame is of the same COB-ID and form as the HeartBeat frame (described below), having a COB-ID of 1792 (700h) + CAN ID, and having one byte of data to convey the NMT State. The "NMT State" field is zero to indicate boot-up.

Each node may be configured to produce a "Heartbeat" message. The combination of the COB-ID and the data indicates the current state of the given node. The presence of the message being repeated/updated within the expected time frame indicates the node is still alive and well. The NMT states expected in the heartbeat are:

        0 = boot-up
        4 = Stopped
        5 = Operational
        127 = Pre-Operational

The Heartbeat protocol allows each node to produce a "Heartbeat" frame at the selected interval in milliseconds. The Heartbeat Producer time is configured by setting the CAN Dictionary Object 1017:0 = to the desired time in milliseconds. A value of zero disables the heartbeat.

Each node may also be configured to monitor one or more (SilverLode units allow up to 8) heartbeats associated with other nodes, via object 1018h. The user program may be configured to react to the absence of a heartbeat as well as to the change in state of another node to determine its own actions. The detecting nodes software (.qcp) must be programmed to determine whether to shut down and/or disable itself and/or other drives in the system, or to take other corrective or reporting action, if any. The Heartbeat Consumer time should be configured to be somewhat longer than the monitored node's Heartbeat Producer Time to allow for bus loading delaying the heartbeat packet, as it is a low priority message.

Note that the various NMT objects are "Unconfirmed" services, meaning the frame is sent, but no response/confirmation is produced to the frame. Zero or more nodes may be consumers of the given frame.

Service Data Objects (SDO)

Service Data Objects provide communications between nodes to allow the uploading (reading) or downloading (writing) of Data Dictionary Objects of other nodes. These services allow wide access to the various Data Dictionaries, but are slower, typically using lower priority COB-Ids, and requiring a response, as the SDO services are all Confirmed. The SDO services also require a one-to-one mapping from the SDO Client (requestor of the read or write) to the SDO Server (node being read or written). Only one Client should be mapped to each Server at a time, although each Node may support multiple clients and servers. The SilverLode CANopen software provides two clients and two servers per node.

The SDO service may be accessed through the CAN Dictionary Access, Remote (CDR) command. Prior to using the CDR command, the SDO Communications Parameters must be configured. Each Node has one Root SDO server that is configured to default COB-Ids (Server Rx at 1536 (600h) + CAN ID, Tx at 1408 (580h) + CAN ID. The communications parameters for this server are Read Only.

Accessing a remote SDO Server requires configuring the local Data Dictionary Objects for the SDO Client being used (Client 1 or Client2). The Client Tx COB-ID must be set to the wanted Server Rx COB-ID, and the Client Rx COB-ID must be set to the wanted Server Tx COB-ID. Client SDO parameters are accessed via Data Dictionary Object 1280 Sub-Indexes 1 through 3).

The local Data Dictionary Objects may be accessed using the access CAN Dictionary Access, Local (CDL) command. This command allows uploading (reading) from the Data Dictionary object into User Registers, as well as downloading (writing) to the Data Dictionary object from a constant or from User Registers.

Process Data Objects (PDO)

Process data objects provide real time data communications between nodes with minimal program intervention. These objects are unconfirmed, meaning that the data is sent to zero or more consumers, with each object having only one producer. Each PDO object is provided with a unique COB-ID. PDO objects are normally configured to operate in a Modal Fashion:– once they have been configured they continue to operate autonomously in the background until they are again reconfigured.

The producer of a PDO must be configured to produce the PDO (send data). Likewise, every consumer of a PDO must be configured to receive the PDO. Each PDO may only have one producer, but may have zero or more consumers.

The producer configuration includes selecting:
- Which local Data Dictionary Objects to map into the PDO data
  - Order of mapping and size of each object
  - Number of objects mapped (up to 4) - Maximum of 8 bytes of data
- Synchronous or Asynchronous transmission
  - Synchronous simultaneous updates all nodes at SYNC event
  - May be sent every SYNC event, or every X SYNC events.
  - May be sent on particular SYNC events, such as 1,4,7,…
  - May be triggered by time or change, but sent at the next SYNC event.
- Triggering mechanism and/or Time interval
- Inhibit Time to prevent overloading the bus if rapid changes would otherwise cause overly rapid triggering and transmission of data.
- COB-ID of the given PDO.

Each PDO consumer must also be configured, but require fewer parameters:

- Which local Data Dictionary Objects receive the PDO data
  - Order of mapping and size of each object
  - Number of objects mapped - Maximum of 8 bytes of data
  - The data does NOT need to be mapped to the same Data Dictionary Object as was the PDO producer, and in practice is usually not mapped to the same Object. The data sizes of the producer and consumer should be compliant, however.
  - The Consumer does NOT need to map all of the data sent by the producer, but will produce an (optional) error is the more data is mapped than sent by the producer.
- Synchronous or Asynchronous operation - Synchronous updates the local object at the next SYNC event, Asynchronous updates the object immediately.
- COB-ID of the given PDO.

Once these objects have been configured, and the nodes are in the Operational NMT state, the PDO producers automatically produce the PDO data frames, and the PDO consumers automatically consume them. Common uses for PDO objects would be to allow one node to send an operation state to one or more other nodes; to convey I/O status from one node to one or more other nodes; to broadcast the position of a master axis to one or more CAM following axes; or to use a CAN open encoder to provide position feedback information to close a dual loop control operation.

Predefined Objects

CANopen also defines certain other objects to be used by the nodes. These include the SYNC object, the EMCY object, and the TIME object.

SYNC

The SYNC object, with a default COB-ID of 128 (80H) is broadcast by the designated node in the system, and consumed by the other nodes (that have configured to use the same SYNC COB-ID). The SYNC event (completion of transmission of the SYNC object for the SYNC producer, and the reception of the SYNC object for all SYNC producers) causes all synchronous PDO producers to sample their data and begin sending data, as well as all synchronous consumers to update their internal Data Dictionary Objects with any data received since the last SYNC event.

The SYNC producer must be configured to select the SYNC period. It may optionally be configured to send a SYNC cycle counter with the given cycle modulus. The SYNC cycle counter may be used to cause PDO data production on the wanted SYNC cycle from multiple nodes. For example, to prevent bus overloading, three different PDOs could send on different SYNC cycles, with PDO "x" sending on cycles 1, 4, 7; PDO "y" sending on cycles 2, 5, 8; and PDO "z" sending on cycles 3, 6, 9. Other PDOs could produce data every SYNC event, while yet others send only when the Data changes.

EMCY

The EMCY object is used to signal emergency conditions from a Node to other nodes monitoring the given node. These conditions include communications problems, voltage, current and temperature problems, user or runtime code errors, etc.,. The candidates for generating EMCY messages are enabled by setting the appropriate bits via Object 2001H. (By default all are enabled unless Object 2001h is otherwise configured.) Similarly, Object 2000h sets local CAN Error triggers (which cause the CAN Error – Bit 10 in IS2 – to be set, able to trigger a Kill Motor Extended (KMX) condition if KMX has been configured to include a trigger on the CAN Error bit.

The EMCY object includes the ID implicitly via the COB-ID, defaulting to 80h+NodeID, unless modified via Object 1014h. Object 1015h sets the inhibit time for EMCY messages so that they do not overload the communications.

Each time a qualified (see Object 2001h) error occurs, the contents of the Error register data (Object 1001h) as well as a specific error code is transmitted. The error code information is also saved to the Predefined Error Field, Object 1003h, with Sub-Index 0 indicating how many errors (0 to 4) are queued in the FIFO buffer of 1003h, with the most recent error stored at Sub-Index 1, and the oldest at Sub-Index 4. (The FIFO buffer implementation allows up to four errors before discarding the oldest). The error storage may be cleared by writing a zero (0) to Sub-Index 0. When a qualified error that has already been reported has cleared, then an Error Cleared frame is sent, indicating any error conditions (Object 1001h) that may still be pending. Transient errors, such as a short PDO frame will produce an Error frame followed by an Error Cleared frame, as these errors are transient by nature. If the error repeats, it will only be reported after the Error Cleared frame associated with that error has been sent, the multiple detections

being considered as a single error occurrence. Other errors, such as drive disabled or over temperature, will only generate a single error message at the onset of the error, and a single Error Cleared frame when the error has been resolved. Local action for any of these errors may be triggered by configuring object 2000h as well as the appropriate bits (including the CAN Error bit) in the Kill Motor Extended command.

The EMCY frame consists of eight bytes of data. The first two bytes are the EMCY Error Code (EEC - see Object 2002h for a list of codes), sent low byte, high byte. Next byte is a copy of the contents of the Error Register (Object 1001h), followed by an error type byte, bit 0 indicates a hardware error, while bit 1 indicates a communications error. The last four bytes are the current state of the Error Status bits (as of the time of the EMCY frame), the same as the contents of Object 2007h.

TIME

The TIME object is used to broadcast Time of Day to all nodes in the system. The nodes thus all keep time with the Time Master Node. The SilverLode CAN software also provides crystal frequency compensation to allow the node to lock onto the master time to cancel out crystal tolerance and drift between nodes. (The master may have drift with respect to true time, but all nodes on the bus will drift together.) The time object maps Date and milliseconds since midnight. The SilverLode may be configured as a Time Consumer, but lacking time of day / calendar capability, it may not be configured as a Time Producer.

NOTE: The High Speed Time Object may be mapped to a PDO to produce a time basis to allow locking the time/frequency of multiple SilverLode units. The high-speed Time counter a 32-bit microsecond counter, so it repeats approximately every 71.58 minutes. The suggested PDO transmit time 10 to 50 milliseconds. The time base between units should substantially lock within 30 to 60 seconds. Large differences in time will cause a direct setting of local High Speed Time, while smaller changes will adjust the local time base and reset the local Time. Still smaller changes will only adjust the local time base, as random variation in the BUS communications (such as the length of the preceding frame) will be greater than the actual clock drift. This PDO should be mapped to a high priority (low COB-ID) for best accuracy.

Chapter 4 - QuickControl And CANopen

This chapter documents using QuickControl to access some of CANopen's advanced features. Please read Chapter 1 for basic CANopen initialization, register sharing and I/O sharing.

Input Sharing Details

I/O #101 through I/O #116 may be shared by transmitting register 238, which contains the extended input states in the upper word, and the output drive state (1=output transistor turned on, load energized, output low, 0 = output transistor off, load not energized, output passively pulled high). Register 238 may be transmitted using either the CTRL or CTRR commands.

The consumer node need merely map this to register 199 (default register for mapped I/O – may be changed via Object 2008h). Jump, wait, and motion completion may be selected by use of the Remote Input Enable Codes, #1 through #32 corresponding to bits 0 through 31. In the given example, the unit will wait for a 0 to 1 transition on unit 16's IO116.

Too share inputs from two remote devices, map the first to the lower word of register 199 and the second to the upper word. The first remote unit's inputs will be accessible via Enable Codes Remote Input #1-#16 and the second via Remote Input #17-#32.

Remote Output Control

Individual or multiple extended I/O outputs may be set and cleared using an SDO (similar to the Configure I/O (CIO) command). The remote unit must first be connected to using the CAN Connect to Remote (CCTR) command (stays connected to the given client until changed). The outputs may then be set or cleared via the SDO command CAN Dictionary Access, Remote (CDR). Sub-Index 9 is used to set bits in the lower word, while Sub-Index D is used to clear bits in the lower word, Index 21EEh corresponding to register 244, the XIO register.

Note, the CCTR command allows a connection to be established using one of two SDO clients. This means that two remote units can be "connected" at the same time. If your application only needs to access the outputs of one or two remote units, you only need to establish the connection(s) at the beginning of the program. The CDR command defaults to using SDO Client 1. SDO Client 2 can be selected from the CDR Advanced button as follows.
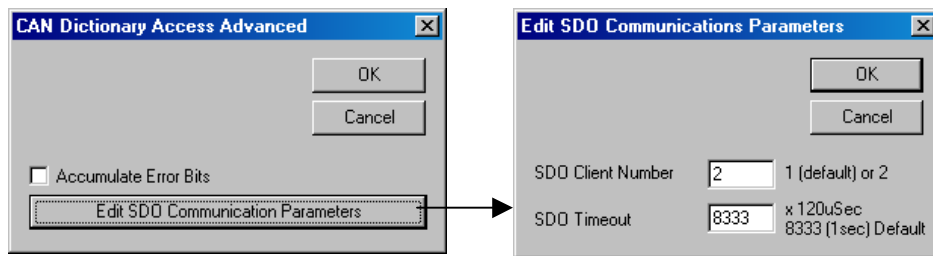
Advanced CANopen Configuration

CANopen provides many advanced capabilities through the configuration of the CAN Data Dictionary. Additional information about the node status is also available via the CAN dictionary. For example, the ISW and IS2 status words of any node are available via Object 1002h of that node. These may be shared via register sharing, using the advanced addressing, or via the CAN Dictionary Access Remote to see a snapshot.

Various error status sources are available via objects 1001h and 1003h, with 1001h holding the present state, and 1003h holding up to 4 previous error states. The particular errors that are serious enough to generate emergency frames (EMCY) and to be logged to 1003 are selected via Object 2001h.

The SYNC communications cycle may be configured via objects 1005h and 1006h. For example to set the local node as the SYNC producer. The CAN Dictionary Access, Local (CDL) command is used to Access the CAN Dictionary. CDL allows data to be read from the object into a register via Mode 0 (the second parameter is the User Register), to be written from a user register via Mode 1 (the second parameter is the register containing the data to be written, or written from a constant immediate value via Mode 2 (the second parameter contains the data to be written). The Index and Sub-Index are the pointers to the object to be read or modified.

In this case, we want to write a 4000 0080h into object 1005.0 (object 1005 Sub-Index 0). We will use Mode 2 to perform this operation. Bit 30 indicates the node is to be a SYNC producer, while the lower 11 bits indicate the default SYNC message ID of 80h.

To set a communications period of 2 milliseconds (2000 microseconds), the time in microseconds needs to be set via object 1006h. Convert 2000 into Hexadecimal (0x07D0).

The resulting program does this configuration.

Line# Oper	Label	Command
1:REM		Set unit as SYNC producer with a Commuications cycle time of 2 milliseconds (2000 microseconds)
2:CDL		CAN Dictionary Access, Local 0x1005.0 = 0x40000080
3:CDL		CAN Dictionary Access, Local 0x1006.0 = 0x000007D0

The time base of the units may be locked to a master unit to avoid the slight drift caused by differences in crystal frequencies. This is done by transmitting object 1013h via a time triggered PDO roughly between 10 and 100 milliseconds (ok to be more frequent if sharing a PDO with other data needing more frequent update rate). The PDO data for the High Resolution Time Stamp Object 1013h from the time master must be mapped to Object 1013 on all the units to be synchronized to the master. The High Resolution

Time Stamp is the free running time in microseconds (locally updated every 40 microseconds). The data is treated in a special manner when updated via a PDO. If the times are significantly differing, the slave unit will merely update its local time data. If the time is fairly close, a the interrupt rate is modified to occasionally add or delete a tick (25nS) to the interrupt rate to frequency lock the local interrupt rate to master unit. If the local time is very to the master time (there is an internal offset added to compensate for sending and processing overhead), then no changes are made, allowing for some random variation in transport times. The free running high speed time should remain locked within a couple of ticks on all units.

Heartbeat

Each unit may be configured to produce a local heartbeat signal. Each unit may also be configured to monitor one to eight other heartbeat sources. The heartbeat is produced every x milliseconds (as configured via object 1017h) and identifies the node producing the heartbeat as well as the NMT state of the producing node. The monitoring node configures, via Object 1016h, the nodes it wishes to monitor, as well as the expected heartbeat time. As the heartbeat message is sent as a very low priority message, the expected heartbeat time should typically be set to some 50% greater than the producer heartbeat to allow for delay caused by higher priority traffic. The monitoring node is notified if the heartbeat stops or is late, as well as if the remote Node has changed its NMT state due to an error condition.

These conditions may be configured to produce errors/EMCY messages, or to trigger a Kill Motor Recovery if the error is considered critical.

Critical error detection is enabled via Object 2000h (the entire mask word may be written to Sub-Index 1, bits may be Set via Sub-Index 2, or Cleared via Sub-Index 3.) A critical error sets bit 10 in the IS2 word. If this bit is enabled in the kill motor recovery extended (KMX) word, a kill motor recovery operation will result. Many different error sources are available. See object 2000h.

Emergency (EMCY) errors generate EMCY messages and log the errors to object 1003h if EMCY is enabled (via object 1014h). The specific error sources that generate EMCY messages are selected via object 2001h.

These conditions may change over the operation of a system. A limit switch may be used in homing. While homing, the tripping of the limit switch should not result in an error. However, following the homing routine, the limit switches may be configured to generate EMCY messages and/or to cause a kill motor recovery if the normal travel should never reach the limit switch.

Limit and Home Switch Mapping

CANopen Profile 402 uses limit switches for homing and other operations. To take advantage of the advanced motion stop conditions, which allow compound stop conditions on motions without having to do multiple motions. These make it easy to implement a homing routine such as move until off of limit switch and stop when first index pulse is found (see advanced stop conditions). These options were previously only able to use IO 1,2, and 3 (plus index) for their operation. These inputs to the advanced stop conditions are now mappable to any IO, including remotely mapped IO

and other status conditions as are available to the Jump command. This mapping is done via object 2004h. Advanced Stop condition "IO1" or Positive limit switch is mapped via 2004.1, "IO2" or Negative limit switch is mapped via 2004.2, and "IO3" or Home switch is mapped via 2004.3. "External Drive Enable", reported via the profile 402 registers, is configured via 2004.4.

Positive limit switch, Negative Limit Switch, and Home switch default to IO1, IO2, and IO3 by default for back compatibility.

Profile 402 Objects

Profile 402 is a CAN open profile for servo drives. It describes a standard set of registers used to control the motions of a remote drive. Registers 100 through 126 are reserved for a special "402" user program if the unit is to be operated as a 402 device these are mapped to various objects in the 6xxxh range. This user program interprets these registers to provide the requested motions and operations. The other 402 objects directly access the related data without intervention of the user program. See the Data Dictionary 402V02 Object Mapping for more information.

CAN STATUS LED and CAN ERR LED

Some units, such as the QCI-D2-IG8, provide two additional LEDs for CAN STATUS and CAN ERROR. The CAN STATUS LED (GREEN) indicates the NMT status of the unit:

CAN STATUS LED (GREEN)	
Off	CAN not initialized
Blinking	Pre-Operational
Single Flash	Stopped
On	Operational

The CAN ERROR LED (RED) indicates the following CAN system errors:

CAN ERROR LED (RED)	
Off	No error
Single Flash	Warning Limit Reached
Double Flash	Heartbeat error
On	Bus off

The highest error is indicated.

Chapter 5 - CANopen Commands

CAN Baud Rate (CBD)

Description
The CAN Baud Rate command sets the CAN baud rate from the standard list of
CANopen Baud Rates (see below). The power-on default CAN baud rate is 1Mb/Sec.
Note that Baud rate 5 is "Reserved" in the CANopen implementation; a 100 kb/sec rate
is included for compatibility with other CAN systems which use that baud rate.

BAUD Rotary Switch
One controllers with a BAUD Rotary Switch, setting CBD to 255 ("Rotary Switch" in
QuickControl) will cause the CAN Baud Rate to be read from the BAUD Rotary Switch.
This allows a user to change the CAN Baud Rate without re-programming. The BAUD
Rotary Switch information is available on bits 4-7 of CAN object 200Ah.

Command Info

Command Name	Command Type/Num	Parameters	Param Type	Parameter Range

CAN Baud Rate (CBD)	Program Class D Code (Hex): 71 (0x47) 2 words	Baud Rate	U16	0 = 1 Mb/Sec
				1 = 800 kb/sec
				2 = 500 kb/sec
				3 = 250 kb/sec
				4 = 125 kb/sec
				5 = 100 kb/sec *Reserved
				6 = 50 kb/sec
				7 = 20 kb/sec
				8 = 10 kb/sec
				255 = Get Baud from
				    BAUD Rotary Switch
				    0-8: Same as above
				    9-F: 1Mb/Sec.

Example
Configure baud rate to 1 Mb/sec.

@16 71 0 (CR)

Response
ACK only

QuickControl Example
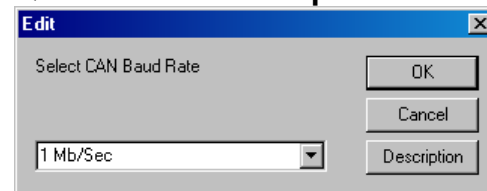
CAN Connect to Remote (CCTR)

Description
This command configures the CAN SDO client 1 or client 2 on the local unit to communicate with the default SDO server on the selected unit. This configuration is required to select the remote node prior to using the **CAN Dictionary Access Remote (CDR)** command.

The CCTR command sets the appropriate parameters in object 1280h (Client 1) or 1281h (Client 2) in the local Data Dictionary. CCTR is a Combo Command, internally consisting of three **CAN Dictionary Access, Local (CDL)** commands.

Command Info

Command Name	Command Type/Num	Parameters	Parameter Range
CAN Connect to Remote (CCTR)	Program Class COMBO D Code	Remote ID	Select the CAN ID with which to establish communications 1 to 127 (1 to 7fh)
	18 words	Client Number	Select which local client to use for communications

QuickControl Example
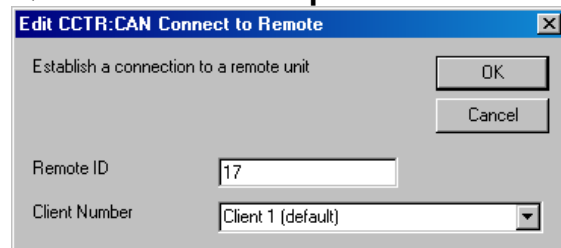
CAN Dictionary Access, Local (CDL)

See Also CAN Dictionary Access, Remote (CDR)

Description

The CAN Dictionary Access Command provides read/write access to the local CAN Data Dictionary Objects. Read access copies the Data Dictionary Object value to a User Register. Write Access copies the value from a Register to the selected Data Directory Object, or alternately, from a Constant to the selected Data Dictionary Object.

The Data Dictionary contains all objects accessible from CAN; some of these must be configured by a controller serving as a Master prior to accessing CAN. The Data Dictionary is accessed via a 16-bit Index and an 8-bit Sub-Index.

See CANopen User manual for a detailed listing and explanation of supported Data Dictionary Objects.

Note: An invalid access will generate a Command Error and halt the program

Note: Time of Day Objects require two Registers.

Command Info

Command Name	Command Type/Num	Parameters	Param Type	Parameter Range
CAN Dictionary Access, Local (CDL)	Program Class D 72 (0x48) 6 words	Mode	U16	0 = Read (Dictionary => Register) 1 = Write (Register => Dictionary) 2 = Write (Constant => Dictionary)
		Data Register or Constant	S32	Holds either Register number or Data: Register Number (Modes 0 or 1) 32-bit Constant (Mode 2)
		Index	U16	0 to FFFFh (0 to 65536)
		Sub-Index	U16	0 to FFh (0 to 255)

Example
Set Communications Cycle Time (Sync period) to 1000 microseconds. Data Dictionary 1006h Sub-Index 00h

@16 72 2 1000 0x1006 0x00 (CR)
or
@16 72 2 1000 4102 0 (CR)

Response
ACK only

Second Example
Read Heartbeat Status of first Heartbeat Consumer (Data Dictionary 2005h Sub-Index 01h) into User Register 30

@16 72 0 30 0x2005 0x1(CR)
or
@16 72 0 30 8197 1(CR)

Response
ACK only

QuickControl Example

Dialog

Access local CAN Data Dictionary   Description   OK   Cancel

Remote Dictionary Access
Index   0x1006
Sub-Index   0x00

Mode
○ Read Dictionary to Register
○ Write Register to Dictionary
● Write Constant to Dictionary

Edit Constant
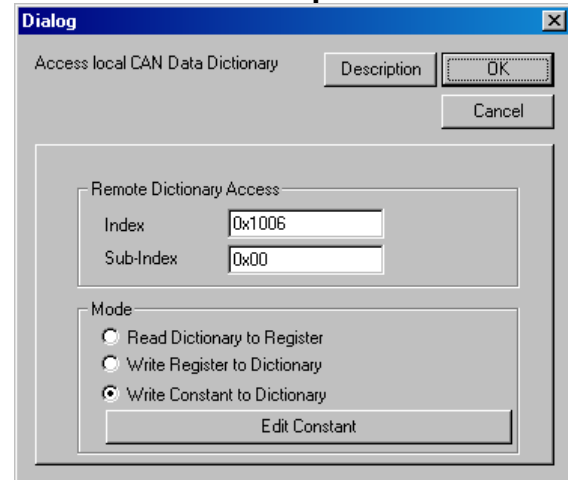
CAN Dictionary Access, Remote (CDR)

See Also CAN Dictionary Access, Local (CDL)

Description

The CDR command provides a means to access another Node's Data Dictionary. This command will not operate properly until the SDO (Service Data Object) Client Communication objects in the local Data Dictionary have been initialized to specify which node is to be contacted (see Combo Cmd CCTR). Unless the node has been previously configured (either locally by its internal user program) or by a Master node, the initial communications must be configured to use the remote Node's default SDO server COB-ID's. These must be configured using the CAN Dictionary Access, Remote (CDR) command to set the local SDO Client Communications via Object 1280h (Client 1) or 1281h (Client2). These do not need to be reconfigured as long as the local Node is communicating with the same remote Node, but do need to be reconfigured to communicate (with the same client) to a different Node. With two SDO clients available, a Master Node may communicate to up to two slave/peer nodes without reconfiguring the SDO client communications each time.

Mode: Specifies the type of action requested for this SDO communication. Mode 0 Performs an upload (Read remote) to Register, Mode 1 performs a download (write remote) from register, Mode 2 performs a download (write remote) from constant. See the Errors section below for more details. QuickControl determines this automatically when using the Remote Output tab.

Data Register or Constant: For Mode types 0 and 1, this parameter specifies the starting register to use, with additional data taken from/delivered to subsequent registers, as needed. For Mode type 2 (Download/write Constant), the second parameter is a 32 bit constant. If the transfer is greater than four bytes, additional registers will be used, in an ascending order. Strings will be transferred to Registers low byte of the lowest numbered Register, up to the high byte, to the next register low byte, and so on. Mode 2, Constant style downloads, are limited to no more than four bytes. QuickControl determines this automatically when using the Remote Output tab.

Index and Sub Index: These parameters specify the entry in the remote Data Dictionary being accessed. Note that these addresses are normally specified in Hexadecimal, and many of the values written are specified in Hexadecimal for consistency with CANopen convention. QuickControl determines this automatically when using the Remote Output and Remote Register Access tabs.

Byte Count: In the case of a download (write) action, this is the number of bytes available for transfer, which may exceed those required by the object accessed in the remote Node. (Data is sent low byte first; an 8-bit transfer from a 32-bit source will only transfer the lowest byte, even if four bytes were specified as being available.) In the case of an upload (read) action, the byte count specifies the maximum number of bytes to transfer from the remote node so as not to exceed the local register space reserved for the transfer. For single register transfers - upload or download - this parameter may be set to 4. When uploading strings, setting bit 15 in addition to the number of bytes will allow up to the number of bytes to be transferred without an error if more byte of data are available (i.e. only read up to first x bytes of the string). QuickControl determines this automatically when using the Remote Output and Remote Register Access tabs.

Timeout: This parameter specifies the number of cycles to wait for the remote node to complete the SDO transfer before the local node times out. This is needed to prevent the user program from hanging on a remote node not present, excessive bus usage, etc. This value is dependent upon the bus loading as well as the baud rate and the size of the transfer. At 1Mb/sec, a value of 40 (4.8 ms) should normally be sufficient if the bus is not overly loaded, and the transfer is up to four bytes. You may need to experiment to determine the setting for your configuration. In QuickControl, this is edited using the Advanced button.

Client: This parameter specifies which local SDO client to use. The use of more than one client allows access to more than one node without reconfiguring the SDO client communication parameters. Each Node has at least one SDO server (the SilverLode CANopen code provides two) to service SDO client requests. Each server/client connection is a one-to-one mapping, that is, each client may only access one server and each server may only service one client. In QuickControl, this is edited using the Advanced button.

Error Bits

A time-out may occur with this command if the remote node does not respond within the specified time period. If a timeout occurs, the command will terminate, but the ISW "Positive" condition bit will be set (testable with the Jump command testing for Positive) while the "Zero" and "Negative" bits will be cleared. This allows the user code to determine that the command timed out.   An error may also occur if attempting to write to a read-only variable, or attempting to access an object that does not exist. If this type of error occurs, the command will terminate, but the ISW "Negative" condition bit will be set, while the "Positive" and "Zero" bits will be cleared.

If the command terminates normally, the ISW "Zero" bit is set, and the "Positive" and "Negative" bits are cleared. A jump on Positive or Negative to an error recovery routine after each CDR command should be used if the data sent or received is critical.

Accumulating Error Bits

Setting bit 2 in the Mode word (i.e. actions 4,5,6) does not alter the action, but allows accumulation of the returned ISW "error" bits to allow a single test at the end of a series of CDR commands (as long as no other register type commands – such as a calculation command – have been executed). To implement this, the first CDR command would not have the accumulate bit (bit2) set in the Mode word, so as to clear out any prior settings of the ISW Zero, Negative, and Positive bits, replacing them with the results of the first CDR command. The rest of the CDR commands would have the accumulate bit set. At the end of a series of CDR commands, a Jump on Negative would detect any disallowed operations (such as attempting to write to a read-only object), and a Jump on Positive would indicate if there were any timeouts (as would be caused by a busy bus, an improperly wired bus, or a remote module not powered up/initialized for CAN). If neither of these bits were set, then all of the series of CDR commands succeeded. In QuickControl, this is edited using the Advanced button.

See Service Data Objects (SDO) section in the CAN Data Dictionary.

Command Info

Command Name	Command Type/Num	Parameters	Param Type	Parameter Range
CAN Dictionary Access, Remote (CDR)	Program Class D 80 (0x50) 9 words	Mode	U16	Dictionary refers to Remote Dictionary 0 = Read (Dictionary => Register) 1 = Write (Register => Dictionary) 2 = Write (Constant => Dictionary) Setting Bit 2 (i.e. 4,5,6) does same function, but accumulates status bits.
		Data Register or Constant	S32	Register (Actions 0 or 1) 32 bit constant (Mode 2)
		Index	U16	Remote Node Index
		Sub-Index	U16	Remote Node Sub-Index
		Byte Count	U16	Number of Bytes to
		Timeout	U16	Number of 120uS ticks before Timeout occurs.
		Client	U16	Local SDO client to use, 1 or 2

Example:
Upload remote unit Actual Position (Register 1) via Data Dictionary Object Index 2101h Sub-Index 00h, 4.8ms timeout using Client 1. Results are written to Register 30.

@16 80 0 30 0x2101 0 4 40 1 (CR)

or
@16 80 0 30 8449 0 4 40 1(CR)

Response
ACK only

QuickControl Example
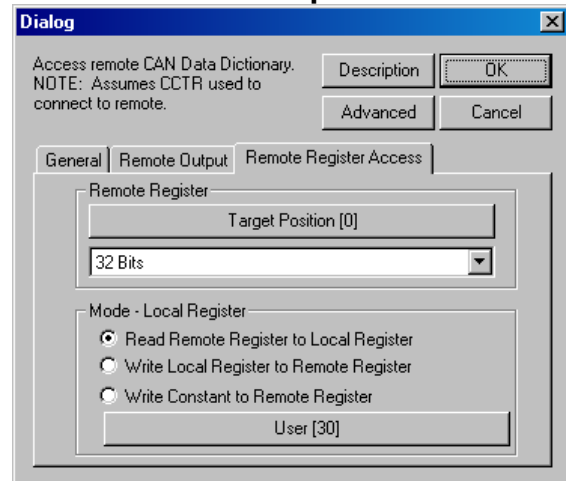
CAN Identity (CID)

Description
CAN Identity for the first time sets the CANopen CAN ID and starts up CAN frame processing; it also results in an initialization of the COB-ID's (including those previously configured). This command cannot be processed (will produce a command error) if the CAN NMT state is "Operational" or "Stopped"; it will only work in "Pre-Operational" or prior to configuring CAN (CAN initialization state).

Setting the CAN ID after it has been previously set only changes the current CAN CAN ID; to force a re-initialization of the COB-ID's, it is necessary to negate the ID value (i.e. –1 to –127).

The CAN CAN ID may be set explicitly (1 to 127) or it may be set to the lower 7 bits Node's Serial ID by setting the ID to zero (0). If using the lower 7 bits of the Serial ID, do not use Serial ID 128 as this would result in an invalid CAN CAN ID of 0, which is reserved for broadcast (will produce a command error). The user must assure the resulting CAN ID values are unique within a system as duplicate CAN ID values will cause communications errors.

Command Info

Command Name	Command Type/Num	Parameters	Param Type	Parameter Range
CAN Identity (CID)	Program Class D 73 (0x49) 2 words	CAN CAN ID	S16	0 (use lower 7 bits of Serial ID) 1-127 (Set and initialize) -1 to –127 (Set)

Example
Set CAN CAN ID to 0

@16 73 0 (CR)

Response
ACK only

QuickControl Example
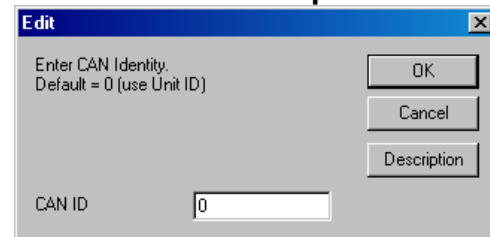
CAN Set NMT State, Local (CNL)

See Also CAN Set NMT State, Remote (CNR)

Description

Transitions the local NMT (Network Management) State. Used by Peer or Master mode Nodes to change between NMT states "Pre-Operational", "Operational", "Stopped", and to Re-initialize Communications parameters.

The NMT State of each node determines what types of CAN communications are allowed to take place. Some Data Dictionary Objects may only be written while in the Pre-Operational State (see CAN Data Dictionary).

See CNR for state definitions.

Note: The Transition Request Value is limited to the four documented values. The NMT State is transitioned to the requested state, but the Transition Value **does not** correspond to the resulting NMT State. See table

See **Network Management (NMT) Objects** in Chapter 3 for more details.

Command Info

Command Name	Command Type/Num	Parameters	Param Type	Parameter Range
CAN Set NMT State, Local (CNL)	Program Class D 74(0x4A) 2 words	NMT State Transition Request	U16	1 = Transition to Operational 2 = Transition to Stopped 128 = Go Pre-Operational 130 = Reset Communications Parameters (when done transition to Pre-Operational) (NMT State 127)

Example

Transition to NMT State "Operational"

@16 74 1 (CR)

Response

ACK only

QuickControl Example
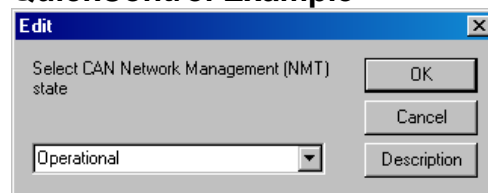
CAN Set NMT State, Remote (CNR)

See Also CAN Set NMT State, Local (CNL)

Description

Transitions the Network Management (NMT) State for one or all other nodes on the CANopen Network. Used by the Master Node to transition other nodes to the desired NMT state "Pre-Operational", "Operational", "Stopped", and to Re-initialize Communications parameters of the target node(s) or to Reset the target node(s).

The NMT State of each node determines what types of CAN communications are allowed take place. Some Data Dictionary Objects may only be written while in the Pre-Operational State (see CAN Data Dictionary).

Pre-Operational (NMT State 127)

Access to NMT State dependent Data Dictionary Objects allowed. NMT communications allowed. PDO communications not allowed. SDO communications allowed.

Operational (NMT State 5)

All communication modes allowed

Stopped (NMT State 4)

Only NMT communications allowed.

Reset Communications

Sets Communications parameters (COB-ID's) back to their default value, and then transitions to "Pre-Operational"

Reset Node

Forces a full hardware reset of the selected node(s), the nodes should return to Pre-Operational when done.

NOTE: Resetting a Node causes the Node to temporarily revert to RS-232 mode, single drop until the initialization has reached a certain point. This may cause the serial communications to temporarily "drop out". This is normal.

The NMT management commands structures in CANopen require that only one Master Node **produce** Commands and zero or more Nodes **consume** them. There is no direct handshake mechanism, however, if the consumer node has its heartbeat configured, then the heartbeat will reflect the new state on its next transmission.

If the transmit buffer is free to transmit (no other pending transmission from a CNL command), this command returns only the Zero flag set. If the prior transmission has not yet successfully completed, the command terminates with only the Negative flag set. The Jump conditional command may be used to retry the command or to enter an error recovery routine.

Note: The Transition Request Value is limited to the five documented values. The NMT State is transitioned to the requested state, but the Transition Value **does not** correspond to the resulting NMT State. See table below.

See **Network Management (NMT) Objects** in the CAN Data Dictionary Document for more details.

Command Info

Command Name	Command Type/Num	Parameters	Param Type	Parameter Range
CAN Set NMT State, Remote (CNR)	Program Class D 81(0x51) 3 words	CAN ID	U16	0 = Broadcast (All Nodes) 1-127 = Nodes 1 to 127
		NMT State Transition Request	U16	1 = transition to Operational (NMT State 5) 2 = transition to Stopped (NMT State 4) 128 = Go Pre-Operational (NMT State 127) 129 = Reset Node (when done transition to Pre-Operational) (NMT State 127) 130 = Reset Communications Parameters (when done transition to Pre-Operational) (NMT State 127)

Example
Transition Node 16 to NMT state "Operational".

@16 81 16 1 (CR)

Response
ACK only

QuickControl Example

CAN Register Map, Local (CRML)

Description
This combo-command configures CAN to receive the selected Producer Data Object (PDO) into a user register (or registers). Multiple nodes may be configured to simultaneously consume the PDO data produced by any remote node.

PDO data streams may be used to dynamically share a register contents from a producer (sending node) to zero or more consumers (receiving nodes).

The PDO identifier is selected by choosing the Node number and transmit channel of the PDO producer. The local receive channel merely selects which local resource is used to receive the data; any receive channel not already in use may be used. The data is deposited into the selected register whenever it is received.

The default configuration maps the receive PDO data onto a single register, configured for Asynchronous (immediate) update.

The Advanced button allows selecting Rx Type (Asynchronous or Synchronous), as well the editing the register mapping. Synchronous Rx Type holds the received data until the next SYNC (Synchronization) frame is sent, allowing all nodes to simultaneously update their data from multiple data sources (as well as sampling the new data to be sent synchronized to the SYNC frame, if the Transmit PDO data set to type synchronous).

The Edit Register Mapping button on allows for finer mapping of the PDO data. The incoming data may be directed to up to two destinations. The destinations may be long words, word, 24 bit data, or 8-bit bytes. The data may be written, used to set bits (OR function) or used to clear bits (AND with NOT of data) in the designated registers.

Additionally, under the advanced button on the Edit Can Register Mapping panel, the data may be also be manually mapped to CAN Directory Objects by specifying the desired index, subindex, and number of bits to be mapped to the selected object (number of bits must correspond to size of object). The Mode pull down box must be set to manual to manually map this data.

The CRMLcommand sets the appropriate parameters in objects 1400h and 1600h (Rx channel 1), 1401h and 1601h (Rx Channel 2), 1402h and 1602h (Rx channel 3), or 1403h and 1603h (Rx Channel 4) in the local Data Dictionary. CRML is a Combo Command, internally consisting of seven **CAN Dictionary Access, Local (CDL)** commands.

Note: The Receive and Transmit PDO objects may have up to 4 objects and up to 64 bits mapped to them if configured manually. The CRML combo command is limited to the more common configurations, allowing up to 2 objects to be mapped. See Data Dictionary for information on manual mapping.

Command Info

Command Name	Command Type/Num	Parameters	Parameter Range
CAN Register Map Local (CRML)	Program Class COMBO D Code 42 words	Remote Unit ID	Select the node ID of the remote unit producing the data. 1 to 127 (1 to 7fh)
		Remote Tx Channel	Select the transmit channel used by the remote unit. 1 to 4
		Local Rx Channel	Choose the desired receive channel (does not need to match Tx channel) 1 to 4
		Register to Map	Local user register to be updated with the received data. Register must be writable.
		Advanced	See above notes

QuickControl Example

CAN Register Map, Remote (CRMR)

Description
This combo-command configures another node via CAN to receive the selected Producer Data Object (PDO) into its user register (or registers). Multiple nodes may be configured to consume the PDO data produced by the producer node.

This combo-command performs a function very similar to the **CAN Register Map Local (CRML)**, except that instead of configuring the local node, a remote node is being configured to receive data. The configuration is done via the CAN bus, using SDO operations.

PDO data streams may be used to dynamically share a register contents from a producer (sending node) to zero or more consumers (receiving nodes).

The unit to be configured must first be selected via the **CAN Connect to Remote (CCTR)** command. (CRMR defaults to Client 1, but may use either client via the advanced options).

The PDO identifier is selected by choosing the Node number and transmit channel of the PDO producer. The receive channel merely selects which local resource of the node being configured is used to receive the data; any receive channel not already in use may be used. The data is deposited into the selected register of the selected node whenever it is received.

The default configuration maps the receive PDO data onto a single register, configured for Asynchronous (immediate) update. CRMR uses Client 1 as its default SDO channel, and a 1 second timeout per operation. A failure of communications will cause this combo-command to repeat until successful.

The Advanced button allows selecting Rx Type (Asynchronous or Synchronous), as well the editing the register mapping. Synchronous Rx Type holds the received data until the next SYNC (Synchronization) frame is sent, allowing all nodes to simultaneously update their data from multiple data sources (as well as sampling the new data to be sent synchronized to the SYNC frame, if the Transmit PDO data set to type synchronous).

The Edit Register Mapping button on allows for finer mapping of the PDO data. The received PDO data may be directed to up to two destinations. The destinations may be long words, word, 24 bit data, or 8-bit bytes. The data may be written, used to set bits (OR function) or used to clear bits (AND with NOT of data) in the designated registers.

The Edit SDO Communications Parameters button under the advanced tab allows selection of either Client 1 or Client 2 operation, as well as setting the SDO timeout.

Additionally, under the advanced button on the Edit Can Register Mapping panel, the data may be also be manually mapped to CAN Directory Objects by specifying the desired index, subindex, and number of bits to be mapped to the selected object

(number of bits must correspond to size of object). The Mode pull down box must be set to manual to manually map this data.

The CRMR command sets the appropriate parameters in objects 1400h and 1600h (Rx channel 1), 1401h and 1601h (Rx Channel 2), 1402h and 1602h (Rx channel 3), or 1403h and 1603h (Rx Channel 4) in the selected node's Data Dictionary. CRMR is a Combo Command, internally consisting of seven **CAN Dictionary Access, Local (CDL)** commands and one **Jump (JMP)** command.

Note: The Receive and Transmit PDO objects may have up to 4 objects and up to 64 bits mapped to them if configured manually. The CRML combo command is limited to the more common configurations, allowing up to 2 objects to be mapped. See Data Dictionary for information on manual mapping.

Command Info

Command Name	Command Type/Num	Parameters	Parameter Range
CAN Register Map Local (CRML)	Program Class COMBO D Code 46 words	Remote Unit ID	Select the node ID of the remote unit producing the data. 1 to 127 (1 to 7fh)
		Remote Tx Channel	Select the transmit channel used by the remote unit. 1 to 4
		Local Rx Channel	Choose the desired receive channel (does not need to match Tx channel) 1 to 4
		Register to Map	Local user register to be updated with the received data. Register must be writable.
		Advanced	See above notes

QuickControl Example

CAN Transmit Register, Local (CTRL)

Description
This command configures the selected local Transmit PDO to broadcast the selected register data.

The register to be transmitted is selected, as well as the PDO transmit channel to be configured.

The default configuration selects 32 bits from the given register, has a Transmission Type of 255: Asynchronous, set to transmit:
1) when the unit first goes into operation state (or the Transmit PDO is configured if dynamically configured (already in operational mode).
2) whenever the data changes
3) At least every 200 milliseconds (so that the state is refreshed)
4) But not more than every 2 milliseconds (so constantly changing data will not overload the bus.

Via the Advanced tab, the transmit type may be selected to be Synchronous (0 through 240 SYNC cycles, with 0 indicating to send synchronously only on change. The Inhibit time determines how fast back to back transmissions may occur. If using synchronous mode, the inhibit time may be set to 0. The event timer determines minimum frequency of transmission. In synchronous mode, the transmission will still be delayed until the next SYNC signal. Normally, the event timer is set to 0 (disabled) in synchronous mode. The starting SYNC # is used to delay the given number of sync cycles before transmitting. This may be further enhanced by configuring Data Dictionary Object 1019h for the least common factor of the various Synchronous transmission times. See Object 1019h and Synchronous Communications sections for more details.

The transmit channel combined with the node number determine the priority of the data frame. The lower 7 bits of the frame address (by default) are the transmitting CAN ID, while the upper 5 bits grow in value as the transmit channel is increased. The frame identifier for Tx channel 1 is 180h + CAN ID, Tx channel 2 is 280h + CAN ID, Tx channel 3 is 380h + CAN ID, Tx channel 4 is 480h + CAN ID. The frame with the lowest identifier has the highest transmission priority over the CAN bus.

Internally, the **CCTR** combo-command consists of 11 **CDL** commands and one **CLD** command which configure data dictionary objects 1800h and 1A00h for Tx channel 1, objects 1801h and 1A01h for Tx channel 2, objects 1802h and 1A02h for Tx channel 3, or objects 1803h and 1A03h for Tx channel 4.

Note: transmissions will not begin until the transmitting unit is in NMT state Operational. Units configured to receive data will not react to PDO data until they are set to NMT-state Operational. If synchronous mode is configured, one of the nodes must be configured to produce a SYNC signal by configuring objects 0x1005 (bit 30 must be set on producer), and 1006h (sync time in microseconds).

Command Info

Command Name	Command Type/Num	Parameters	Parameter Range
CAN Transmit Register Local (CTRL)	Program Class COMBO D Code 72 words	Tx Channel	1 to 4 lowest numbered channel has highest priority for node. For the same channel, lowest numbered node has highest priority.
		Data Register	Selects the Data register to transmit
		Advanced options: 2nd transmit channel, Transmit type, inhibit time, event timer, starting sync.	See description above for details

QuickControl Example
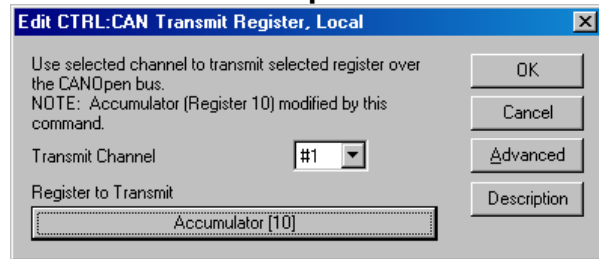
CAN Transmit Register, Remote (CTRR)

Description
The **CAN Transmit Register, Remote (CTRR)** combo–command is used to configure a remote node to transmit data via a PDO object.

This combo-command performs a function very similar to the **CAN Register Map Local (CRML)**, except that instead of configuring the local node, a remote node is being configured to transmit data. The configuration is done via the CAN bus, using SDO operations.

PDO data streams may be used to dynamically share a register contents from a producer (sending node) to zero or more consumers (receiving nodes).

The unit to be configured must first be selected via the **CAN Connect to Remote (CCTR)** command. (CTRR defaults to Client 1, but may use either client via the advanced options).

The register to be transmitted is selected via the pull down menu, as well as the PDO transmit channel to be configured.

The default configuration selects 32 bits from the given register, has a Transmission Type of 255: Asynchronous, set to transmit:
1) when the unit first goes into operation state (or the Transmit PDO is configured if dynamically configured (already in operational mode).
2) whenever the data changes
3) At least every 200 milliseconds (so that the state is refreshed)
4) But not more than every 2 milliseconds (so constantly changing data will not overload the bus.

Via the Advanced tab, the transmit type may be selected to be Synchronous (0 through 240 SYNC cycles, with 0 indicating to send synchronously only on change. The Inhibit time determines how fast back to back transmissions may occur. If using synchronous mode, the inhibit time may be set to 0. The event timer determines minimum frequency of transmission. In synchronous mode, the transmission will still be delayed until the next SYNC signal. Normally, the event timer is set to 0 (disabled) in synchronous mode. The starting SYNC # is used to delay the given number of sync cycles before transmitting. This may be further enhanced by configuring Data Dictionary Object 1019h for the least common factor of the various Synchronous transmission times. See 1019h and Synchronous Communications sections for more details.

Under the SDO Client Parameters tab of the Advanced panel, the SDO Client number may be selected, as well as the SDO timeout period (for each CDR command).

The transmit channel combined with the node number determine the priority of the data frame. The lower 7 bits of the frame address (by default) are the transmitting CAN ID, while the upper 5 bits grow in value as the transmit channel is increased. The frame identifier for Tx channel 1 is 180h + CAN ID, Tx channel 2 is 280h + CAN ID, Tx

channel 3 is 380h + CAN ID, Tx channel 4 is 480h + CAN ID. The frame with the lowest identifier has the highest transmission priority over the CAN bus.

Internally, the **CCTR** combo-command consists of 11 **CDR** commands, one **CLD** command, and one **JMP** command which configure data dictionary objects 1800h and 1A00h for Tx channel 1, objects 1801h and 1A01h for Tx channel 2, objects 1802h and 1A02h for Tx channel 3, or objects 1803h and 1A03h for Tx channel 4. The Jump command repeats the sequence in the case of errors or timeouts. The jump may be manually modified to vector to an error recovery routine.

Note: transmissions will not begin until the transmitting unit is in NMT state Operational. Units configured to receive data will not react to PDO data until they are set to NMT-state Operational. If synchronous mode is configured, one of the nodes must be configured to produce a SYNC signal by configuring objects 1005h (bit 30 must be set on producer), and 1006h (sync time in microseconds).

Command Info

Command Name	Command Type/Num	Parameters	Parameter Range
CAN Transmit Register, Remote (CTRR))	Program Class COMBO D Code 109 words	Tx Channel	1 to 4 lowest numbered channel has highest priority for node. For the same channel, lowest numbered node has highest priority.
		Data Register	Selects the Data register to transmit
		Advanced options: 2nd transmit channel, Transmit type, inhibit time, event timer, starting sync., SDO client, timeout.	See description above for details

QuickControl Example
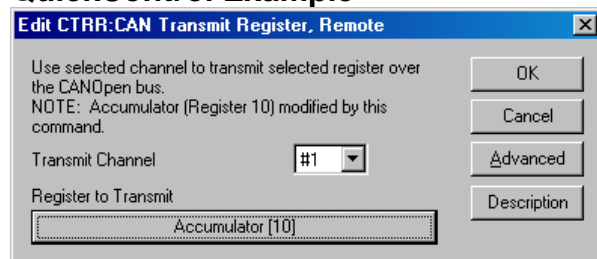
Chapter 6 - CANopen Configuration

Starting Up CAN

Before the CAN network can be accessed, the device must be configured.

First, the CAN Baud Rate must be selected. This is done using the **CAN Baud Rate (CBD)** command. If not set, the default baud rate is 1 Mbps.

The next step is to set a CAN CAN ID using the **CAN Identity (CID)** command. This configures the default addresses or CAN ID. A parameter of 0 will cause the unit to use the lower 7 bits of its Unit ID (serial port) as set by the IDT command as its CAN ID, values of 1 to 127 will cause the CAN ID to be set to 1 through 127 respectively. Note: 0 is reserved as a broadcast address. After the default CAN communication parameters have been initialized, the CAN routines are started: A boot up message is sent, ant then the NMT-State transitions to **Pre-Operational**. In this state, the CAN Dictionary objects may be changed via the **CAN Dictionary Access, Local (CDL)** command, as well as by remote nodes using the Service Data Object (SDO) functions (through the CAN bus) accessed via the **CAN Dictionary Access, Remote (CDR)** command on the remote node.

NOTE: The commands CID and CBD are included in the "Factory Default Initialization - CAN.qcp" file when QuickControl is installed. If the device is initialized with this file (using the Initialization Wizard), CID is set to 0 making the CAN CAN ID the same as the Unit ID (serial) and CBD is set to 1 Mbps.

If the node (SilverLode controller) is operating as a slave to a remote CAN master, then that master should then configure the local Node, and, when ready, switch the NMT-State to Operational.

However, if the node is operating as either a peer or master, then the local program should configure the needed entries in the Data Dictionary by means of the CDL command. The program may set the local node to Operational via the CAN Set NMT State, Local (CNL) command.

The Node responds to all communication types when in Operational state, although some objects become Read Only (RO) in NMT Operational state. SDO and NMT objects (but not PDO objects) are available in the Pre-Operational NMT state. Only NMT objects (heartbeat, NMT state) are available in NMT STOPPED state.

If the COB-ID (Communication Object Identifier) corresponds to any of the Node's Communication objects, and the Node is in a state in which the objects are active, then the Node will act upon the received frame; otherwise, the frames are discarded.

Configuring Process Data Objects (PDO)

Process data objects provide a means of automatically communicating data between a data provider and one or more data consumers. There are four default Receive PDO COB-Ids assigned for each CAN ID, as well as four default Transmit PDO COB-Ids.

A Transmit PDO is configured by Mapping Objects (data to be sent), configuring Communications Parameters (Type, Timing, etc.), and setting a COB-ID for the transmission.

Similarly, a Receive PDO must be configured by Mapping Objects (where the received data is to be stored), configuring Communication Parameters (Type), and setting the COB-ID of the message to be consumed. The Receive COB-ID must match the Transmit COB-ID of the wanted data, or no data will be received by the Receive PDO.

The PDO traffic is only enabled in the NMT state **OPERATE**.

The Data Dictionary Objects used to configure a PDO must be done in a certain sequence to prevent errors. This sequence prevents accidental data transmission with a partially configured or partially altered PDO configuration.

Transmit PDO 1 (TPDO 1)

1800: Comm. Parameters
1800.1: COB-ID
1800.2: Type
1800.3: Inhibit Time
1800.5: Event Timer
1800.6: Starting Sync #

1A00:Object Mapping
1A00.0: # of Objects
1A00.1: First Object, # Bits
1A00.2: 2nd Object, # Bits
1A00.3: 3rd Object, # Bits

Data Frame

Node "A"

Receive PDO 3 (RPDO 3)

1402: Comm. Parameters
1402.1: COB-ID
1402.2: Type

1602:Object Mapping
1602.0: # of Objects
1602.1: First Object, # Bits
1602.2: 2nd Object, # Bits
1602.3: 3rd Object, # Bits

Node "B"

Initial PDO Configuration at Startup

Both Receive (RPDO) and Transmit (TPDO) configuration COB-ID registers default to disabled with their addresses set to their default COB-ID. The Type, Inhibit Time, Event Timer, and Sync Start Value fields all default to zero. If the PDO has previously been mapped, then the order for destructing the PDO is to first disable the PDO by setting bit 31 of the COB-ID high (1) Next, the number of mapped objects must be set to zero (0) if any of the object mapping is being change.

Receive PDO communications for RPDO 1 through RPDO 3 are configured through Objects 1400h through 1403h respectively. Object mapping for RPDO 1 through RPDO3 configured via objects 1600h through 1603h, respectively

Transmit PDO communications for TPDO 1 through TPDO 3 are configured through Objects 1800h through 1803h. Object mapping for TPDO 1 through TPDO 3 are configured via objects 1A00h through 1A03.

Transmit PDO Configuration

Synchronous PDO Operation
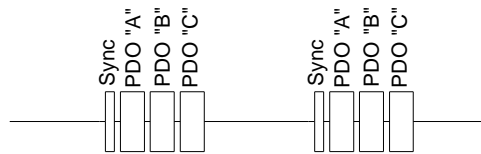
"A" "B" &"C" Type = 1: Send data every SYNC
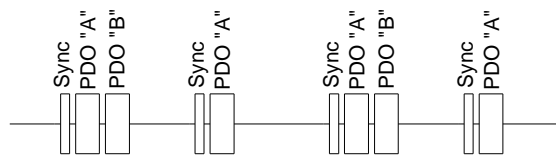
"A" Type = 1: Send data every SYNC
"B" Type = 2: Send data every other Sync

The **Transmission Type** should be determined first. These may be Synchronous or Asynchronous. They may also be triggered, time based, or RTR based.

Synchronous operation requires one of the nodes to be configured as a SYNC producer.
Upon detection of a SYNC signal (or the completion of Transmission of the SYNC signal for the producer), each CAN bus node determines if there are any Synchronous transmissions that need to be produces, and samples the data for those synchronous TPDOs ready to transmit. Each node must also update any pending synchronous RPDO data received since the last SYNC signal. The Synchronous transmissions may be sent every SYNC signal, or may be sent every N sync signals. The user may also

select every N sync cycle starting with sync cycle M, where N and M are between 1 and 240. A synchronous TPDO may also be configured as triggered or Remote Transmit Request (RTR) based. In the case of a synchronous triggered or RTR based TPDO, following their trigger (change in data) or the receipt of an RTR frame, the PDO is marked as transmit pending. Data will be sent following the next SYNC signal.

Asynchronous PDO Operation
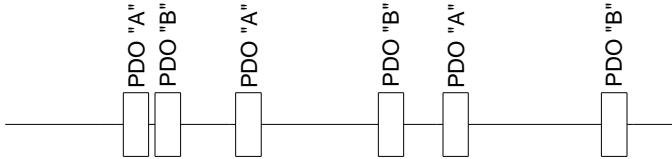
"A" Type = FFh, Inhibit = 13: Transmits every data change
          But not faster than every 1.3 milliseconds
"B" Type = FEh, Event Timer =3: Transmits every 3 milliseconds

Asynchronous TPDOs may be configured as triggered (due to data change), and/or time based. They may also be configured to be RTR triggered and/or time based. This allows a data set to be produced every N milliseconds or when the data changes, whichever happens first. An inhibit time may also be specified which sets the minimum time between transmissions to prevent constantly changing data from overloading the CAN bus.

The Type, Inhibit Time, Event Timer, and Sync Start value must be configured while the COB-Ids associated with the PDO are disabled (Bit 31 high).

Next, the Objects to be mapped onto the selected PDO must be selected and configured.
The **Object Mapping** may not be updated while the **Number of Objects Mapped** is non-zero. The **Number of Objects Mapped** may not be changed while the associated **COB-Id is** enabled.

At reset, the **COB-ID**s are disabled and the **Number of Objects Mapped** are set to zero for all objects, so the next step is to map the Data Dictionary Objects onto the PDO. The PDO is capable of carrying up to eight bytes of information. The purpose of mapping is determining which data (if any) is sent in each of those bytes. The first object mapped starts in Byte 0 and consumes as many bytes as are needed for its data type. The smallest increment of data supported in this implementation is eight bits (byte), so 1,2,3,4 byte (8, 16, 24, and 32 bit) data fields are supported. The Index and Sub-Index of the Data Dictionary Object, as well as the number of bits for that object form the map. The object corresponding to the next set of data is mapped next. Up to four objects may be mapped to each PDO in this implementation. The data type of the object is compared with the number of bits and an error will result if the two are not

consistent. After all objects for a given PDO have been mapped, the **Number of Objects Mapped** parameter may be written. If fewer objects have been mapped than are indicated by **Number of Objects Mapped** parameter, then an error will result.

Finally, the COB-ID may be set and enabled.

Once the COB-ID has been enabled, bits 30:0 may not be altered without first disabling the COB-ID. This then enables changes to the PDO communications parameters, and the **Number of Objects Mapped** parameter. Setting the **Number of Objects Mapped** parameter to zero again allows the PDO mapping parameters to be modified.

There are fewer Communication parameters associated with the RPDOs than the TPDOs because they do not require trigger times nor inhibit times, nor starting SYNC values. One or more TPDOs being used must be configured on each CANopen node supplying process data. A corresponding RPDO must be configured on each CANopen node consuming each process data stream.

If any of the data streams is synchronous, then one of the nodes must be configured to be a SYNC producer (1005h, 1006h and optionally 1007h).

The SYNC counter (1019h) may be configured to include a modulo count as part of the SYNC message. This provides a reliable method of determining SYNC cycle count so that the variously TPDO transmissions will occur on the wanted cycle number. The starting Sync number for the TPDO may be selected via Objects1800h-1803h Sub-Index 6 for the corresponding TPDO 1 through TPDO 3.

Example SYNC Producer Configuration (Only One Node)

1005h (SYNC COB-ID) = 80h (Default COB-ID, pre-configured)
1006h (Comm. SYNC Cycle) = 3000 (BB8h) for 3 millisecond Sync (time in microseconds)
1007h (SYNC Window) = 2000 (7D0h) for 2 millisecond (may only transmit SYNC PDO
                     inside this window to leave time for other communications
1019h (SYNC Counter) = 6 (Sync will carry a payload which counts 1 through 6, 1 through 6
                     this is set to the least common multiple of all Type numbers in
                     the range of 1 through 240 (Synchronous types). This is optional.

CANopen Message Structure: COB-ID Allocation

The CANopen document DS310V4 (communication profile) specifies the various communication objects, Data Directory structure and standard objects, as well as the allocation structure for COB-Ids (Communication Object Identifiers).

The Basic COB-ID structure breaks the 11-bit COB-ID into a 4-bit function code (bits 7:10) and a 7-bit node ID (bits 0:6). Remember, the lower the COB-ID, the higher priority of the message. The COB-ID's are broken into Broadcast Messages (CAN ID bits 0:6 = 0) and Peer-to-Peer messages (CAN ID <> 0).

- Broadcast
    - NMT: Function Code = 0, CAN ID = 0 => COB-ID = 0
    - SYNC: Function Code = 1, CAN ID = 0 => COB-Id = 80h (128)
    - TIME: Function Code = 2, CAN ID = 0 => COB-ID = 100h (256)
- Peer-to-Peer
    - EMCY: Function Code = 1, CAN ID => COB-ID = 81h to FFh (129 to 255)
    - TPDO1: Function Code = 3, CAN ID => COB-ID = 181h to 1FFh (385 to 511)
    - RPDO1: Function Code = 4, CAN ID => COB-ID = 201h to 27fh (513 to 639)
    - TPDO2: Function Code = 5, CAN ID => COB-ID = 281h to 2FFh (641 to 767)
    - RPDO2: Function Code = 6, CAN ID => COB-ID = 301h to 37fh (769 to 895)
    - TPDO3: Function Code = 7, CAN ID => COB-ID = 381h to 3FFh (897 to 1023)
    - RPDO3: Function Code = 8, CAN ID => COB-ID = 401h to 47fh (1025 to 1151)
    - TPDO4: Function Code = 9, CAN ID => COB-ID = 481h to 4FFh (1153 to 1279)
    - RPDO4: Function Code = Ah, CAN ID => COB-ID =501h to 57fh (1281 to 1407)
    - SDO Tx: Function Code = Bh, CAN ID => COB-ID=581h to 5FFh (1409 to 1535)
    - SDO Rx: Function Code = Ch, CAN ID => COB-ID=601h to 67Fh (1537 to 1663)
    - Heartbeat: Function Code = Eh, CAN ID => COB-ID=701h to 77Fh (1793 to 1919)
- Reserved
    - 0h NMT (as described above, not configurable)
    - 1h (1) reserved
    - 71h-7Fh (113 to 127) reserved
    - 101h to 180h (257 to 384) reserved
    - 581h to 5FFh (1409 to 1535) default SDO (Tx) (described above, not configurable)
    - 601h to 67Fh (1537 to 1663) default SDO (Rx) (described above, not configurable)
    - 6E0h to 6FFh (1760 to 1791) reserved
    - 701h to 77fh (1793 to 1919) NMT Error Control / Heartbeat (described above, not configurable)
    - 780h to 7ffh (2020 to 2047) reserved

COB-Ids that are not reserved may be used for SDO server/clients and for PDOs. Note: Some devices only support the default SDO Server and PDO COB-ID enumeration (their COB-Id fields are not configurable); the paired nodes must be mapped to the fixed nodes' enumeration.

EMCY Configuration

1014h (COB-ID EMCY) = 80h + COB-ID (set by default, no need to modify)
1015h (EMCY Inhibit) = 30 (1Eh) (Example value – don't update EMCY messages more
                            Than once every 3 milliseconds (100uS increments)
(Optional)
2001h (EMCY Report) = 11FF FFFFh (report all errors except limit switches via EMCY)

Emergency Communications frames (EMCY) provide means to notify other nodes of error problems. These nodes must support EMCY consumer (not currently implemented). The selected errors also are indicated via objects 1001h, 1003h, and in raw bit form in 2002h and 2007h.

Heartbeat Configuration

Each (SilverLode) node is capable of both producing and consuming (monitoring) Heartbeat messages. Each node to be monitored by any other node (or nodes) must be configured via Object 1017h (Heartbeat Producer Time in milliseconds). To have a node report its continued operation as well as its current NMT state every 10 milliseconds, configure 1017h=10 (0Ah).

To configure a node to monitor other node's Heartbeats, configure Object 1016h (Consumer Heartbeat Time). For example, to monitor Node 17 (11h) and Node 18 (12h), with each producing a heartbeat every 10ms, the heartbeat consumer should be configured for a slightly larger value, such as 15ms (0Fh) to allow for a busy bus transmitting higher priority frames.

1016.1 = 0011 000Fh (Node 17, 15ms)
1016.2 = 0012 000Fh (Node 18, 15ms)

The status of these nodes is indicated in the contents of Object 2005h, 2005.01 for Node 17, and 2005.02 for Node 18.

Immediately following configuring the heartbeat monitoring, the pending bit (Bit 14 in object 2005 with the corresponding Sub-Index) is set and the heartbeat timed out (Bit 15) is cleared. The pending bit stays high until the first heartbeat is detected; until the first heartbeat is detected, the heartbeat will not time-out. The pending bit may be checked to see if the unit being monitored has started up (if it is self configured to produce a heartbeat) prior to initiating other configuration. With each heartbeat, the current NMT state is sent as well as the CAN ID; Bit 13 in the corresponding Sub-Index of Object 2005 is set if the NMT state has changed since configuring object 1016, or since clearing this bit by writing a 1 to it. Bit 12 is set if the node has changed from Operational to either Pre-Operational or Stopped State.

A logical OR of all monitored heartbeat errors is reported in CAN_Errors, reflected in Objects 2002h and 2007h ; CAN_Errors is also used by 2000h and 2001h to generate the CAN Error Status bit (Bit 10 in IS2), as well as to report EMCY frames, respectively. The same is true of Changed to non-operational status via Bit 22 of CAN_Errors. This provides rapid access to these conditions without constantly polling them. In order for a local serious condition to be reported via the Heartbeat, the detecting routine (such as the Kill Motor Recovery routine) should change state to either Pre-Operational or Stopped when the error is detected. This will signal the serious problem to other nodes which may then respond.

Object 2000h may be configured to report any serious error condition to IS2 via the CAN Error bit. The CAN Error bit may be, in turn, monitored via the Kill Motor Extended (KMX) command to cause a Kill Motor Recovery (KMR) for any condition considered serious enough to warrant it. Alternatively, a second thread program may monitor the CAN Error bit to determine other actions. Combinations of these may also be employed.

Note: If Thread 2 is being used to check for serious errors, the Thread 2 Active bit in IS2 word may be monitored via the KMX command to trigger appropriate action if Thread 2 dies (is not operating - Low).

Chapter 7 - CANopen Data Dictionary

Object Dictionary Structure

Index	Description
0000h – 001Fh	Static data types
0020h – 003Fh	Complex data types
0040h – 005Fh	Manufacturer specific data types
0060h – 02FFh	Device profile specific data types
0260h – 0FFFh	Reserved
1000h – 1FFFh	Communications Profile Area
2000h – 5FFFh	Manufacturer Specific Profile Area
6000h – 67FFh	Standardized profile area
6800h – 9FFFh	Profile area for additional logical devices
A000h – AFFFh	Standardized interface profile area
B000h – FFFFh	Reserved

Supported Simple Data Types

Index	Type	Behavior when destination is user register
0002h	I8	Single word written, sign extended to full word
0003h	I16	Sign extends to both words
0004h	I32	Both words written
0005h	U8	Single word written, lower 8 bits contain data
0006h	U16	Single word written
0007h	U32	Both words written
000Ch	Time of Day	2 registers written, day into base register, milliseconds into base+1 register
000Dh	High Speed Time	Single word written
0010h	I24	2 words written, sign extended
0016h	U24	2 words written, no sign extension

Supported Manufacturer Data types:

Index	Type	Behavior when destination is user register
0060h	O32	32-bit OR the bits into the register
0061h	O16	16-bit OR the bits into the register
0062h	O8	8-bit OR the bits into the register
0063h	C32	32-bit Clear the indicated bits
0064h	C16	16 bit Clear the indicated bits
0007h	C8	8-bit Clear the indicated bits

These Data types provide the ability to OR bits into the given register, or to clear the given bits. They are interpreted as unsigned numbers when evaluating. They are Write Only.

Object Dictionary Object Type Codes

Object Code	Object Name	Comment
00h	NULL	A dictionary entry with no data fields
02h	DOMAIN	Large variable block of data, such as program code
05h	DEFTYPE	Denotes a TYPE definition, U16, I24, etc.
06h	DEFSTRUCT	Defines a record type, such as PDO mapping
07h	VAR	Single value, such as U16, I8, Visible string.
08h	ARRAY	Multiple data field object with each field having the same data type (such as U16, etc.). Note: Sub-index 0 is U8 and represents the size of the array. It is not part of the array data.
09h	RECORD	Multiple data field object where data fields are any combination of simple variables. Sub index 0 describes the number of data fields; it is U8 and not a part of the Record data.
	All	Note: Sub-Index 255 for all complex objects is of type U32, and is not part of any data object. It returns the OBJECT CODE of the object in bits 0:7, and the Data type in bits 8:23; bits 24:31 are reserved (00h).

Supported Structures / Complex data types

Index	Sub	Type	Purpose
0020h			PDO Communication Parameter
0020h	00	U8	Highest Sub-Index supported
0020h	01	U32	COB-ID
0020h	02	U8	Transmission Type
0020h	03	U16	Inhibit Time
0020h	04	U8	Reserved
0020h	05	U16	Event Timer
0020h	06	U8	Sync Start Value

Index	Sub	Type	Purpose
0021h			PDO Mapping Parameter Record
0021h	00	U8	Number of Objects Mapped
0021h	01	U32	Object 1
0021h	02	U32	Object 2
0021h	03	U32	Object 3
0021h	04	U32	Object 4

Index	Sub	Type	Purpose
0022h			PDO Mapping Parameter Record
0022h	00	U8	Highest Sub-Index supported
0022h	01	U32	COB-ID Client -> Server
0022h	02	U32	COB-ID Server -> Client
0022h	03	U8	CAN ID of Client

Index	Sub	Type	Purpose
0023h			PDO Mapping Parameter Record
0023h	00	U8	Highest Sub-Index supported
0023h	01	U32	Vendor ID
0023h	02	U32	Product Code
0023h	03	U32	Revision Number
0023h	04	U32	Serial Number

Index	Sub	Type	Purpose
0040h			User Register Access Type Mapping
0040h	0	U8	Number of Access types
0040h	1	U32	32 bit access to User Register
0040h	2	U16	16 bit access to upper User Register
0040h	3	U16	16 bit access to lower User Register
0040h	4	U24	U24 access to User Register (no sign extension)
0040h	5	I24	I24 access to User Register (sign extension to 32 bits)
0040h	6	U8	8 bit access to lowest byte of register
0040h	7	O32	32-bit OR the bits into the register
0040h	8	O16	16-bit OR the bits into upper User Register
0040h	9	O16	16-bit OR the bits into lower User Register
0040h	Ah	O8	8 bit OR the bits into lowest byte of register
0040h	Bh	C32	32-bit Clear the indicated bits
0040h	Ch	C16	16-bit Clear the bits upper User Register
0040h	Dh	C16	16-bit Clear the bits lower User Register
0040h	Eh	C8	8-bit Clear the bits lowest byte of register

Note: Sub-index 7 through E are Write Only operations, and will produce an error if attempting to modify a Read Only (RO) register. These accesses provide the ability to set bits or clear bits in the given register without affecting the other bits. These mode are available while through the Dictionary (Local or Remote), as well as via PDO services.

Supported Objects

1000h Device Type
Describes the type of device and its functionality. Defines a Servo Drive with configurable PDOs

Index	Sub	Type	Access	Purpose	Value	Ref	PDO
1000h	00	U32	RO	Device Type = Servo Drive	0002 0192h	402v02	No

1001h Error Register
Bit field of current errors.

Bit	Meaning
7	Manufacturer specific
6	Reserved = 0
5	device profile specific
4	communications error
3	temperature error
2	voltage error
1	current error
0	generic error - set for ANY error

Note: Bit 0 is set if any of the other bits is set.

See Object 2001h information for specific error sources, gating of these sources as errors, as well as the related error codes. Object 2000h selects which of these conditions generate a CAN Error bit in Internal Status Word 2 (IS2), triggering a Kill Motor Recovery if so enabled.

See 1003 Predefined Error Field for specific sources of these errors.

Index	Sub	Type	Access	Purpose	Ref	PDO
1001h	00	U8	RO	Error Register - Bit Mapped	402v02	Yes

1002h Manufacturer Status Word

32-bit Manufacturer Status word. The upper 16 bits is IS2, the lower 16 bits is ISW

Index	Sub	Type	Access	Purpose	Ref	PDO
1002h	00	U32	RO	Manufacturer Status Word	301v04	Yes

bit 31 IO7 (Dynamic)
bit 30 IO6 (Dynamic)
bit 29 IO5 (Dynamic)
bit 28 IO4 (Dynamic)
bit 27 THREAD2 Running (1=active)
bit 26 CAN Communication Error (1=error) (Dynamic)
bit 25 Extended I/O has isolated power missing (LATCHED)
bit 24 Encoder Analog Signals Out of Spec (LATCHED)
bit 23 Hardware over temp bit set (LATCHED)
bit 22 External Drive Enable Low (LATCHED)
bit 21 High power driver over temp analog sensors (LATCHED)
bit 20 Motor temperature fault (too high) (LATCHED)
bit 19 Motor Driver Disabled by Factory block (Dynamic)
bit 18 Encoder re-phased itself (lost encoder counts)(LATCHED)
bit 17 Velocity limit exercised (LATCHED)
bit 16 Millisecond Timeout counter active (Dynamic)
bit 15 reserved
bit 14 under-voltage
bit 13 over-voltage
bit 12 wait exhausted
bit 11 sensor found on last move
bit 10 halt command
bit 9 position error
bit 8 motion error
bit 7 Driver Enabled (low from multiple causes including over temp)
bit 6 IO3
bit 5 IO2
bit 4 IO1
bit 3 Negative result
bit 2 positive result
bit 1 zero
bit 0 Index found

1003h – Predefined Error Field

This array is a First-In-First-Out buffer (FIFO) holding (up to) the four most recent errors produced (as configured by Object 2001 to produce EMCY messages).

Sub-Index 0 is an error counter, taking values in the range of 0 to 4. This error counter may be cleared by writing a zero (0) to Sub-Index 0. Any other value will produce an error. Clearing the error counter also clears any errors stored at Sub-Indexs 1 through 4.

Sub-Indexes 1 through 4 hold the error information. Bits 31:16 are Manufacturer specific, currently reserved, set to 0, but subject to change. Bits 15:0 are the respective Error Codes.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1003h				Predefined Error Object		301v04	No
1003h	0	U8	RW*	Number of Errors in Array			
1003h	1	U32	RO	Most recent Error			
1003h	2	U32	RO	Next most recent error			
1003h	3	U32	RO	Next to oldest error			
1003h	4	U32	RO	Oldest error			

Error Source	2001h Enable Bit	Error Bits	Error Code	
Driver over temp	0	0, 3	4310h	Driver Over Temp Digital output, HC drive Thermistor
Driver over voltage	1	0, 2	3210h	As set by OVT command
Driver under voltage	2	0, 2	3120h	As set by LVP,LVT
Motor over temp	3	0, 3	4000h	As configured via upper word of Register 241
Encoder Analog Error	4	0, 7	7305h	Encoder voltages indicate open or short
Phasing error	5	0, 7	7122h	Apparently lost encoder counts - Encoder re-phrased via Index
N.V Memory error	6	0, 7	5530h	EEPROM read failure
PDO data out of range	7	0, 4	6320h	PDO data out of range for Object
Position error	8	0, 7	8611h	As configured by the ERL command
Motion error	9	0, 7	8612h	As configured by the ERL command
Command error	10	0, 7	6200h	Command error - command parameters not valid at processing time
Heartbeat Error	11	0, 4	8130h	One of the nodes monitored via Object 1016h timed out
Error passive	12	0, 4	8120h	Excessive Error responses from this node, no longer signaling errors
Protocol Error	13	0, 4	8200h	Not sufficient data for Time of Day message
Extended IO power off	14	0, 2	3000h	Extended I/O not functioning, usually power not applied
Drive disabled	15	0, 7	5440h	Drive Enable signal not present
Velocity limit exercised	16	0, 7	8400h	Closed loop velocity loop limit (VLL) command restricting motion
PDO data too short	17	0, 4	8210h	Received PDO not processed because of insufficient data in frame
PDO data too long	18	0, 4	8211h	Received PDO had excess data, PDO was still processed
Processor over temp	19	0, 3	4110h	As set by the MTT command
Recovered from bus-off	20	0, 4	8140h	CAN experienced a Bus-Off condition
Sync Late	21	0, 4	0F001h	Sync not sent by time of next sync - bus overload
Heartbeat - NMT	22	0, 4	0F002h	Received Heartbeat changed to non-operational
CAN Error Warning	23	0, 4	0F003h	Almost at Error Passive due to error count
Against CW/CCW limits	24	0, 7	0F004h	Either soft limits SSL or hard limits LCW & LCC
Negative Limit Switch	25	0, 7	5441h	Negative Limit Switch Active
Positive Limit Switch	26	0, 7	5442h	Positive Limit Switch Active
Reserved	27	0, 7	5443h	Reserved
Driver Interlock	28	0, 7	5444h	Driver Interlock Inactive (Disabled)
Reserved	29	0, 7	5445h	Reserved
Reserved	30	0, 7	5446h	Reserved
Reserved	31	0, 7	5447h	Reserved

1005h COB-ID SYNC

Sets the consumer or producer COB-ID for the SYNC (Synchronization) signal. The default value is 80H or 128 in reception mode. Bits 29:0 are used to map the COB-ID, as used for other COB-Ids in this manual. Setting bit 29 indicates a 29 bit extended ID, with bits 0:28 forming the ID. Clearing bit 29 indicates standard 11-bit ID in bits 0:10. Setting bit 30 causes the device to produce the SYNC signal rather than to monitor it. The period between SYNC signals when defined as a producer is determined by the value in Communication Cycle Period (1006h)

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1005h	00	U32	RW*	COB-ID for SYNC	0080h	301v04	No

*RW only when in PreOperational NMT state. RO in all other NMT states.

1006h Communication Cycle (SYNC) Period

Sets the time in microseconds between SYNC periods. A value of 0 disables the SYNC production.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1006h	00	U32	RW*	COMM Cycle Period	0000h	301v04	No

*RW only when in PreOperational NMT state. RO in all other NMT states.

Note: time will be rounded up to next highest 120uS period. Maximum setting is 7,864,199uS.

1007 h Synchronous Window Length

Defines the time window for Synchronous communications following a SYNC signal, time in microseconds. The maximum value is 7,864,199 (7.86 seconds) in SilverLode implementation.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1007h	00	U32	RW*	Sync. Window Length	0000h	301v04	No

*RW only when in PreOperational NMT state. RO in all other NMT states.

Note: time will be rounded up to next highest 120uS period. Maximum setting is 7,864,199uS.

100Ch Guard Time

CANopen provides two different guarding methods, Guarding and HeartBeat. The preferred method is HeartBeat as it eliminates the extra CAN frames associated with the Guard method sending RTR frames to poll the other devices. HeartBeat protocol has been implemented; Guarding protocol is not permitted.

Index	Sub	Type	Access	Purpose	Value	Ref	PDO
100Ch	00	U16	RO	Guard Time	0000h	301v04	No

100Dh Life Guarding

Used with Lifetime for the life guarding protocol. Life Guarding not implemented. See notes above on 100Ch.

Index	Sub	Type	Access	Purpose	Value	Ref	PDO
100Dh	00	U8	RO	Guard Time	0000h	301v04	No

1012h TIME STAMP COB-ID

Defines the COB-ID of the time stamp object and whether it is a consumer or generator. Setting bit 31 enables the TIME consumer. Bit 30 is used to enable the TIME producer; TIME producer is not implemented, attempting to set Bit 30 will produce an error. Bits 29:0 define the COB-ID

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1012h	00	U32	RW*	COB-ID for TIME	0100h	301v04	No

*RW only when in PreOperational NMT state. RO in all other NMT states.

1013h High Resolution Time Stamp

The High Resolution Time Stamp is a 32-bit free running microsecond counter. It starts at zero when reset. The High Resolution Time Stamp may be included in a TPDO to accurately determine the time of a transaction, or it may be mapped to a RPDO to cause multiple units to keep their local time synchronized. A write operation to 1013h (by PDO or SDO or local operation) is handled in one of two ways. If the difference between the value being written and the current value is small, the write is blocked, but the difference between the two time values is used to adjust the ISR cycle so as to lock the local clock onto the transmitted clock. If the time difference is too great, the time is merely updated with no modification to the ISR cycle time, as this may be an initial setting, or one of the units may have restarted.

It is recommended that one unit is configured (via a TPDO) to transmit the High Resolution Time Stamp approximately every 10ms to 100ms, and the other units be configured to consume the Time Stamp (Via an RPDO).

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1013h	00	U32	RW	High Resolution Time Stamp	None	301v04	Yes

1014h COB-ID EMCY

This object sets the COB-ID used for the Emergency (EMCY) frames. The sources for the EMCY frames are configured via Object 2001h. The default COB-ID is 128+ CAN ID (80h+NodeID), with EMCY enabled. See Object 1003h for a description of Error Codes and their causes.

Bit 31 is set low to enable the EMCY producer. Bit 30 is reserved and must be set to 0. Bits 29:0 are the standard COB-ID format: Bit 29 set (1) indicates an extended frame 29 bit ID with bits 28:0 holding the extended frame COB-ID; Bit 29 cleared (0) indicates a standard frame with bits 10:0 holding the standard frame COB-ID.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1014h	00	U32	RW	EMCY COB-ID	80h+NodeID	301v04	No

1015h EMCY Inhibit Time

EMCY Inhibit time sets the minimum time in 100uS increments between successive EMCY messages to prevent overloading the bus due to repeated intermittent errors. A value of 0 disables the inhibit time.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1015h	00	U32	RW	EMCY Inhibit Time (100uS)	0	301v04	No

1016h Consumer Heartbeat Time

This array may be used to denote the CAN ID and the respective HeartBeat timeout time (in 1millisecond increments) for up to 8 CANopen devices.

Bits 31:24 are reserved (Set to 0)
Bits 23:16 contain the CAN ID to monitor
Bits 15:0 contain the related Heartbeat time in 1mS increments (0 = disabled)

Attempting to configure the same CAN ID at different Sub-Indexes will produce an error. Reconfiguring the same CAN ID at the same Sub-Index will reset the heartbeat consumer state back to pending (that is no timeout countdown until the first heartbeat is detected). Setting the CAN ID to a valid node, but setting the heartbeat time to 0 will allow detection of state and state changes, as well as proper operation of the pending detection, but will disable the timeout function.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1016h				Heartbeat Consumer Array		301v04	No
1016h	00	U8	RO	Highest Sub-Index supported	8		
1016h	01	U32	RW*	CAN ID and Heartbeat Time	0		
1016h	02	U32	RW*	CAN ID and Heartbeat Time	0		
1016h	03	U32	RW*	CAN ID and Heartbeat Time	0		
1016h	04	U32	RW*	CAN ID and Heartbeat Time	0		

1016h	05	U32	RW*	CAN ID and Heartbeat Time		0	
1016h	06	U32	RW*	CAN ID and Heartbeat Time		0	
1016h	07	U32	RW*	CAN ID and Heartbeat Time		0	
1016h	08	U32	RW*	CAN ID and Heartbeat Time		0	

*RW only when in PreOperational NMT state. RO in all other NMT states.

1017h Heartbeat Producer Time

Heartbeat Producer Time sets the time in milliseconds between generated Heartbeat frames. A time of 0 disables the function. A non-zero value immediately starts transmitting a heartbeat.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1017h	00	U16	RW*	Heartbeat Producer Time	0	301v04	No

*RW only when in PreOperational NMT state. RO in all other NMT states.

1018h Identity Object

The identity object array contains four 32-bit data entries.

Sub-Index 01 contains the unique vendor code assigned to each vendor.

Sub-Index 02 contains a vendor unique product code.

Sub-Index 03 contains the software revision number information: The high word contains the major revision number, which changes if CANopen functionality has changed; the low word contains minor revision number, which is used to track minor code revisions.

Sub-Index 04 contains the product serial number. These must be unique for all CANopen devices.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1018h				PDO Mapping Parameter Record		301v04	No
1018h	00	U8	RO	Highest Sub-Index supported	4		
1018h	01	U32	RO	Vendor ID	0000 01CCh		
1018h	02	U32	RO	Product Code	Varies		
1018h	03	U32	RO	Revision Number	Varies		
1018h	04	U32	RO	Serial Number	Varies		

1019h Synchronous Counter

The SYNC message may carry no data, or may carry an 8-bit synchronous cycle counter value. The cycle counter allows easy triggering of different CANopen nodes to synchronize up to specific cycles or groups of cycles for their various TPDO operations. An example would be to have one node send its data on odd cycles and another on even cycles to balance the communications load.

Setting the Synchronous Counter to 0 (default) transmits the Sync message with no data. The setting of 1 is reserved, as is 241 through 255. Setting the Sync message to 2 through 240 will cause a count to be included in the Sync message, ranging from 1 through the selected count, and then back to 1 in a cyclic fashion. The value should be selected to be the smallest number (Least common denominator) into which all of the cyclic counts used within the CAN group can be evenly divided.

1019h	Sub	Type	Access	Purpose	Default	Ref	PDO
1019h	00	U8	RW	Synchronous Counter	0	301v04	No

1029h Error Behavior Object

This object allows the selection of the response to the detection of a serious error when the NMT state is **Operational**. By default this value is 0, causing the devious to autonomously change to NMT state **Pre-Operational**. A value of 1 causes no change of NMT state. A value of 02 causes a change in the NMT state to **Stopped**. A serious communication error includes a Bus-off condition, and a Heartbeat event with state "occurred".

Severe internal errors may also trigger changes in NMT state. The same state transitions are selected by values of 00, 01 or 02.

The value written to Sub-Index 01 selects the reaction to severe communications problems. The value written to Sub-Index 02 selects the reaction to severe internal problems.

NOTE: These responses must be implemented in the user code Kill Motor Recovery (KMR) routine, using Kill Motor Extended (KMX) to trip on CAN ERROR as one of the sources. The Kill Motor Recovery routine must examine Registers 122 (Response to COMM ERROR) and Register 123 (Response to Internal Error) to determine what it must do.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1029h				Error Behavior Object		301v04	No
1029h	00	U8	RO	Highest Sub-Index supported	2		
1029h	01	U8	RW	Response to Communication Error	0		
1029h	02	U8	RW	Response to Internal Error	0		

1200h SDO Server 1 Parameters

The first SDO server is not able to be modified. It provides an SDO server accessible by an SDO client on another node. The SDO server provides access to the local nodes Data Dictionary. The Rx and Tx COB-ID values update when the CANopen CAN ID is changed. "ID" is the CAN ID (1 to 127). Rx is the COB-ID associated with an SDO request; the response is transmitted to Tx COB-ID.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1200h				SDO Server 1 Parameter		301v04	No
1200h	00	U8	RO	Highest Sub-Index supported	2		
1200h	01	U32	RO	COB-ID Rx	0000 0600h+ID		
1200h	02	U32	RO	COB-ID Tx	0000 0580h+ID		

1201h SDO SERVER 2 Parameters

The second SDO server also provides access to the local Data Dictionary. It must be configured prior to use. Bit 31 in both the Rx and Rx COB-ID fields must be set low to enable the server. Bit 30 in each field designates the COB-ID as static (0) or dynamic (1). Dynamic SDO mapping is not currently supported; attempting to set Bit 30 = 1 will result in an error. Bits 29:0 are the standard COB-ID format, described above. The third entry is the node-id of the SDO client. Configuring the third entry is optional.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1201h				SDO Server 2 Parameter		301v04	No
1201h	00	U8	RO	Highest Sub-Index supported	3		
1201h	01	U32	RW	COB-ID Rx Client->Server	8000 0000h		
1201h	02	U32	RW	COB-ID Tx Server->Client	8000 0000h		
1201h	03	U8	RW	SDO client ID	0		

1280h SDO CLIENT 1 Parameters

The SDO Client Parameters configure the communications information to allow the local CANopen node to access the SDO servers on other nodes so as to access (read and write) information through their data dictionaries. SDO communications may be initiated via local program control: With the CAN communications enabled (see **Starting Up CAN**), first configure the Client Parameters via the CAN Dictionary Access, Local (CDL). Next read (upload) and write (download) to the remove node via the CAN Dictionary Access, Remote (CDR) command.

The client parameters must be configured prior to use. Bit 31 in both the Rx and Rx COB-ID fields must be set low to enable the server. Bit 30 in each field designates the COB-ID as static (0) or dynamic (1). Dynamic SDO mapping is not currently supported; attempting to set Bit 30 = 1 will result in an error. Bits 29:0 are the standard COB-ID format, described above. The third entry is the node-id of the SDO server. Configuring the third entry is optional.

Note: The default SDO Server 1 addresses for each node are 600h + NodeID for Tx and 580h + NodeID for Rx. Tx and Rx are with respect to the **Client** when setting up both the Client and the Server communications configurations.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1280h				SDO Client 1 Parameter		301v04	No
1280h	00	U8	RO	Highest Sub-Index supported	3		
1280h	01	U32	RW	COB-ID Tx Client->Server	8000 0000h		
1280h	02	U32	RW	COB-ID Rx Server->Client	8000 0000h		
1280h	03	U8	RW	SDO Server ID	0		

1281h SDO Client 2 Parameters

Provides the same configuration information for Client 2 as 1280h does for Client 1. Provides a second SDO communications channel.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1281h				SDO Server Parameter		301v04	No
1281h	00	U8	RO	Highest Sub-Index supported	3		
1281h	01	U32	RW	COB-ID Tx Client->Server	8000 0000h		
1281h	02	U32	RW	COB-ID Rx Server->Client	8000 0000h		
1281h	03	U8	RW	SDO Server ID	0		

1400h 1st Receive PDO Communications Record

Provides COB-ID for the first Receive PDO (RPDO1). Bit 31 low enables the PDO as valid. Bit 30 high enables RTR (Remote Transmit Request frame) data request compatibility. The Node is compatible with RTR requests from the TPDO, but will not generate RTR requests; in this respect, this bit is ignored. Bits 29:0 form a standard COB-ID, as described above.

The Rx Type parameter selects whether incoming data updates the Data Dictionary object immediately upon receipt, or whether it waits until the next SYNC event. Values 0 to F0h as well as FCh indicate synchronous operation. FDh-FFh indicate event driven (immediate update). FCh, FDh also indicate Remote Transmit Request (RTR) operation, which this unit may receive. The node responds to RTR messages but does not support the generation of RTR messages.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1400h				Receive PDO Communication		301v04	No
1400h	00	U8	RO	Highest Sub-Index supported	2		
1400h	01	U32	RW	COB-ID for RPDO 1	8000 0200h+ID		
1400h	02	U8	RW	Rx Type	0		

1401h 2nd Receive PDO Communications Record

Provides COB-ID for the second Receive PDO (RPDO2). Bit 31 low enables the PDO as valid. Bit 30 high enables RTR (Remote Transmit Request frame) data request compatibility. The Node is compatible with RTR requests from the Transmit PDO (TPDO), but will not generate RTR requests; in this respect, this bit is ignored. Bits 29:0 form a standard COB-ID, as described above.

The Rx Type parameter selects whether incoming data updates the Data Dictionary object immediately upon receipt, or whether it waits until the next SYNC event. Values 0 to F0h as well as FCh indicate synchronous operation. FDh-FFh indicate event driven (immediate update). FCh, FDh also indicate Remote Transmit Request (RTR) operation, which this unit may receive. The node responds to RTR messages but does not support the generation of RTR messages.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1401h				Receive PDO Communication		301v04	No
1401h	00	U8	RO	Highest Sub-Index supported	2		
1401h	01	U32	RW	COB-ID for RPDO 2	8000 0300h+ID		
1401h	02	U8	RW	Rx Type	0		

1402h 3rd Receive PDO Communications Record

Provides COB-ID for the third Receive PDO (RPDO3). Bit 31 low enables the PDO as valid. Bit 30 high enables RTR (Remote Transmit Request frame) data request compatibility. The Node is compatible with RTR requests from the Transmit PDO (TPDO), but will not generate RTR requests; in this respect, this bit is ignored. Bits 29:0 form a standard COB-ID, as described above.

The Rx Type parameter selects whether incoming data updates the Data Dictionary object immediately upon receipt, or whether it waits until the next SYNC event. Values 0 to F0h as well as FCh indicate synchronous operation. FDh-FFh indicate event driven (immediate update). FCh, FDh also indicate Remote Transmit Request (RTR) operation, which this unit may receive, but the node does not generate RTR messages.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1402h				Receive PDO Communication		301v04	No
1402h	00	U8	RO	Highest Sub-Index supported	2		
1402h	01	U32	RW	COB-ID for RPDO 3	8000 0400h+ID		
1402h	02	U8	RW	Rx Type	0		

1403h 4th Receive PDO Communications Record

Provides COB-ID for the fourth Receive PDO (RPDO4). Bit 31 low enables the PDO as valid. Bit 30 high enables RTR (Remote Transmit Request frame) data request compatibility. The Node is compatible with RTR requests from the Transmit PDO (TPDO), but will not generate RTR requests; in this respect, this bit is ignored. Bits 29:0 form a standard COB-ID, as described above.

The Rx Type parameter selects whether incoming data updates the Data Dictionary object immediately upon receipt, or whether it waits until the next SYNC event. Values 0 to F0h as well as FCh indicate synchronous operation. FDh-FFh indicate event driven (immediate update). FCh, FDh also indicate Remote Transmit Request (RTR) operation, which this unit may receive, but the node does not generate RTR messages.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1403h				Receive PDO Communication		301v04	No
1403h	00	U8	RO	Highest Sub-Index supported	2		
1403h	01	U32	RW	COB-ID for RPDO 4	8000 0500h+ID		
1403h	02	U8	RW	Rx Type	0		

1600h First Receive PDO Mapping

Each Receive PDO (RPDO) may carry one to four data elements. These data elements must be mapped to their destination Data Dictionary. This is the function of the RPDO mapping. Each element map entry contains the Index and Sub-Index of its destination Data Dictionary object, as well as the number of bits corresponding to that object.

Object Mapping Bits 31:16 contain the Index. Bits 15:8 contain the Sub-Index. Bits 7:0 contain the number of bits. A consistency check between the number of bits and the data type of the mapped object is done as each object mapping parameter is written; an inconsistency results in an error being generated.

The PDO being mapped must be inactive before any changes may be made to the mapping objects; The PDO is inactive if Bit31 of the COB-ID is set high. Any write operations to the mapping object will generate an error if the associated COB-ID is active.

All objects must be mapped prior to configuring **Number of Objects Mapped** . The first N objects must be mapped if **Number of Objects Mapped** is written to N, or an error will result. Mapping objects with Number of Objects Mapped not equal to zero will also result in an error.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1600h				Receive PDO Mapping		301v04	No
1600h	00	U8	RW	Number of Objects Mapped.	0		
1600h	01	U32	RW	1st Object Mapping	0		
1600h	02	U32	RW	2nd Object Mapping	0		
1600h	03	U32	RW	3rd Object Mapping	0		
1600h	04	U32	RW	4th Object Mapping	0		

1601h Second Receive PDO Mapping

All four of the Receive PDO mapping parameter objects work in the same fashion.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1601h				Receive PDO Mapping		301v04	No
1601h	00	U8	RW	Highest Sub-Index supported	0		
1601h	01	U32	RW	1st Object Mapping	0		
1601h	02	U32	RW	2nd Object Mapping	0		
1601h	03	U32	RW	3rd Object Mapping	0		
1601h	04	U32	RW	4th Object Mapping	0		

1602h Third Receive PDO Mapping

All four of the Receive PDO mapping parameter objects work in the same fashion.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1602h				Receive PDO Mapping		301v04	No
1602h	00	U8	RW	Highest Sub-Index supported	0		
1602h	01	U32	RW	1st Object Mapping	0		
1602h	02	U32	RW	2nd Object Mapping	0		
1602h	03	U32	RW	3rd Object Mapping	0		
1602h	04	U32	RW	4th Object Mapping	0		

1603h Fourth Receive PDO Mapping

All four of the Receive PDO mapping parameter objects work in the same fashion.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1603h				Receive PDO Mapping		301v04	No
1603h	00	U8	RW	Highest Sub-Index supported	0		
1603h	01	U32	RW	1st Object Mapping	0		
1603h	02	U32	RW	2nd Object Mapping	0		
1603h	03	U32	RW	3rd Object Mapping	0		
1603h	04	U32	RW	4th Object Mapping	0		

1800h – 1803h Transmit PDO Communications Parameters

The communications parameters for the Transmit PDOs are configured in the same manner as for the Receive PDO channels (See **140xh**), except that Inhibit Time, Event Timer, and Starting Sync may also be configured for each TPDO.

Sub-Index 1 holds the COB-ID for the first Receive PDO (RPDO1). Bit 31 low enables the PDO as valid. Bit 30 high enables RTR (Remote Transmit Request frame) data request compatibility. The Node is compatible with RTR requests from the TPDO, but will not generate RTR requests; Bits 29:0 form a standard COB-ID, as described above.

The Tx Type parameter selects when and how Transmit PDO data is sent. A value of 0 is a Synchronous Triggered event. This means the event must be triggered by a time elapse or by a manual trigger via object **2003h**, but that the data won't be sent until the next SYNC event. Values 1 through F0h cause the data to be sent every 1 through 240 Sync events, respectively. A value of FCh indicates a synchronous RTR transmit, meaning a Remote Transmit Request is required to trigger the transmission, which will be delayed until the next SYNC event.
A value of FDh indicates an asynchronous RTR triggered transmission, meaning the frame is sent as soon as the RTR is received.

FEh, and FFh are asynchronous triggered. They may be triggered by the elapse of the time counter (if not zero), or manually triggered by writing to Object **2003H** in the CAN Dictionary. Type FFh may also be triggered by a change in data value for any of the mapped values since the last transmission.

The Inhibit time is the minimum time, in increments of 100uS, required between successive transmissions of the TPDO. This prevents a change driven TPDO from consuming excessive bus bandwidth. A value of 0 disables the inhibit time function.

The Event Timer is the time in milliseconds between triggering of the TPDO transmission. A value of 0 disables the time trigger function.

The starting Sync Number may be used to delay the given number of sync cycles before transmitting for synchronous TPDOs. If the Synchronous Counter Parameter (1017h) has been configured on the SYNC master, the event will be triggered when (SYNC Cycle – Starting Sync Number)/Type has a remainder of 0 (for Types 1 through 240). In this mode, Type should be greater than or equal to the Starting Sync Number.

For example, to have two TPDOs transmit on alternate cycles, the Synchronous Counter Parameter (**1017h**) on the SYNC producer node should be set to 2; The Event Timer on both TPDOs should be configured for 2, and the Starting Sync Number of the first TPDO should be 1, and the Starting Sync Number of the second TPDO should be 2. The Event Timer should be set to 0 (disabled).

1800h First Transmit PDO Communications Parameters

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1800h				Transmit PDO Communication		301v04	No
1800h	00	U8	RO	Highest Sub-Index supported	3		
1800h	01	U32	RW	COB-ID for TPDO 1	C000 0180h+ID		
1800h	02	U8	RW	Tx Type	0		
1800h	03	U16	RW	Inhibit Time	0		
1800h	04	U8	RO	Reserved	0		
1800h	05	U16	RW	Event Timer	0		
1800h	06	U8	RW	Starting Sync Number	0		

1801h Second Transmit PDO Communications Parameters

The Transmit PDO Communications Parameters of all four TPDOs are configured similarly.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1801h				Transmit PDO Communication		301v04	No
1801h	00	U8	RO	Highest Sub-Index supported	3		
1801h	01	U32	RW	COB-ID for TPDO 2	C000 0280h+ID		
1801h	02	U8	RW	Tx Type	0		
1801h	03	U16	RW	Inhibit Time	0		
1801h	04	U8	RO	Reserved	0		
1801h	05	U16	RW	Event Timer	0		
1801h	06	U8	RW	Starting Sync Number	0		

1802h Third Transmit PDO Communications Parameters

The Transmit PDO Communications Parameters of all four TPDOs are configured similarly.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1802h				Transmit PDO Communication		301v04	No
1802h	00	U8	RO	Highest Sub-Index supported	3		
1802h	01	U32	RW	COB-ID for TPDO 3	C000 0380h+ID		
1802h	02	U8	RW	Tx Type	0		
1802h	03	U16	RW	Inhibit Time	0		
1802h	04	U8	RO	Reserved	0		
1802h	05	U16	RW	Event Timer	0		
1802h	06	U8	RW	Starting Sync Number	0		

1803h Fourth Transmit PDO Communications Parameters

The Transmit PDO Communications Parameters of all four TPDOs are configured similarly.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1803h				Transmit PDO Communication		301v04	No
1803h	00	U8	RO	Highest Sub-Index supported	3		
1803h	01	U32	RW	COB-ID for TPDO 4	C000 0480h+ID		
1803h	02	U8	RW	Tx Type	0		
1803h	03	U16	RW	Inhibit Time	0		
1803h	04	U8	RO	Reserved	0		
1803h	05	U16	RW	Event Timer	0		
1803h	06	U8	RW	Starting Sync Number	0		

1A00h First Transmit PDO Mapping

The Transmit PDO mapping is identical to the Receive PDO mapping (1600h) except the transmit PDO mapping is defining a data producer, and the Receive PDO mapping is defining a data consumer. The object mapping may only be changed when the Highest Object Mapped (Sub-Index 00) = 0. The Highest Object Mapped may only be set to 1 through 4 if at least that number of objects have been mapped (starting at 1 though number indicated). See notes in *1800h First Transmit PDO Communications Parameters.*

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1A00h				Transmit PDO1 Mapping		301v04	No
1A00h	00	U8	RW	Highest Object Mapped	0		
1A00h	01	U32	RW	1st Object Mapping	0		
1A00h	02	U32	RW	2nd Object Mapping	0		
1A00h	03	U32	RW	3rd Object Mapping	0		
1A00h	04	U32	RW	4th Object Mapping	0		

1A01h Second Transmit PDO Mapping

The Transmit PDO mapping is identical to the Receive PDO mapping (1600h) except the transmit PDO mapping is defining a data producer, and the Receive PDO mapping is defining a data consumer.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1A01h				Transmit PDO2 Mapping		301v04	No
1A01h	00	U8	RW	Highest Object Mapped	0		
1A01h	01	U32	RW	1st Object Mapping	0		
1A01h	02	U32	RW	2nd Object Mapping	0		
1A01h	03	U32	RW	3rd Object Mapping	0		
1A01h	04	U32	RW	4th Object Mapping	0		

1A02h Third Transmit PDO Mapping

The Transmit PDO mapping is identical to the Receive PDO mapping (1600h) except the transmit PDO mapping is defining a data producer, and the Receive PDO mapping is defining a data consumer.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1A02h				Transmit PDO3 Mapping		301v04	No
1A02h	00	U8	RW	Highest Object Mapped	0		
1A02h	01	U32	RW	1st Object Mapping	0		
1A02h	02	U32	RW	2nd Object Mapping	0		
1A02h	03	U32	RW	3rd Object Mapping	0		
1A02h	04	U32	RW	4th Object Mapping	0		

1A03h Fourth Transmit PDO Mapping

The Transmit PDO mapping is identical to the Receive PDO mapping (1600h) except the transmit PDO mapping is defining a data producer, and the Receive PDO mapping is defining a data consumer.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
1A03h				Transmit PDO4 Mapping		301v04	No
1A03h	00	U8	RW	Highest Object Mapped	0		
1A03h	01	U32	RW	1st Object Mapping	0		
1A03h	02	U32	RW	2nd Object Mapping	0		
1A03h	03	U32	RW	3rd Object Mapping	0		
1A03h	04	U32	RW	4th Object Mapping	0		

Manufacturer Specific Data Dictionary Objects 2000H – 2FFFh

The objects mapped between 2000h and 2FFFh are Manufacturer Specific – that is, they vary from vendor to vendor.

The Silver Lode CANopen software divides these up as:

2000h – 2080h = special purpose CAN registers
2100h – 21ffh = User registers

The User Registers correspond to the local program accessible Registers, as are defined in the user manual. (As not all 255 registers are currently defined, those not defined will not exist in the Data Dictionary).

2000h Critical Error Mask

The Critical Error Mask determines which errors are of Critical status, causing a CAN Error (Bit 10) to be set in Internal Status Word 2 (IS2). Sub-Index 1 provides a RW operation, while Sub-Index 2 provides a WO set bit operation, and Sub-Index 3 provides a WO clear bit operation. Enabling the CAN Error bit in the **Kill Motor Extended (KMX)** command will cause a critical error to force a Kill Motor Recovery.

The mask must be initialized before any error sources will be reported via the CAN STATUS bit.

Bit #	Error Cause
0	Driver Over Temperature
1	Driver Over Voltage
2	Driver Under Voltage
3	Motor Over Temp
4	Encoder Analog Error
5	Motor Commutation Realign
6	Non-Volatile Memory Error
7	PDO data out of range
8	Position Error
9	Motion Error
10	Command error
11	Heartbeat Error
12	Error Passive
13	Protocol Error
14	Extended IO power missing
15	Drive Disabled
16	Velocity Limit Exercised
17	PDO Data Too Short
18	PDO Data Too Long
19	Processor Over Temp
20	Recovered from Bus Off
21	Bus Overload – new data can not be sent
22	HeartBeat changed to non-operational
23	CAN Error Warning
24	Against CW/CCW limits
25	Negative Limit Switch Active
26	Positive Limit Switch Active
27	Reserved
28	Driver Interlock Inactive
29	Reserved
30	Reserved
31	Reserved

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2000h	0	U8	RO	Max Array Size	3		No
2000h	1	U32	RW	R/W Critical Error Mask	0		No
2000h	2	U32	WO	Set bits Critical Error Mask			No
2000h	3	U32	WO	Clear Bits Critical Error Mask			No

2001 EMCY Report Mask

The EMCY Report Mask determines which errors will be reported via EMCY, and through 1001h Errors, 1003h Predefined Errors, as well as 603fh most recent error code. These errors will only be reported via EMCY and register 1003h if EMCY is enabled via Object 1014h.

The mask must be initialized before any error sources will be reported via EMCY.

Bit #	Error Code	Error Cause
0	4310h	Driver Over Temperature
1	3210h	Driver Over Voltage
2	3120h	Driver Under Voltage
3	4000h	Motor Over Temp
4	7305h	Encoder Analog Error
5	7122h	Motor Commutation Realign
6	5530h	Non-Volatile Memory Error
7	6320h	PDO data out of range
8	8611h	Position Error
9	8612h	Motion Error
10	6200h	Command error
11	8130h	Heartbeat Error
12	8120h	Error Passive
13	8200h	Protocol Error
14	3000h	Extended IO power missing
15	5440h	Drive Disabled
16	8400h	Velocity Limit Exercised
17	8210h	PDO Data Too Short
18	8211h	PDO Data Too Long
19	4110h	Processor Over Temp
20	8140h	Recovered from Bus Off
21	F001h	Bus Overload – new data can not be sent
22	F002h	HeartBeat changed to non-operational or timed out
23	F003h	CAN Error Warning
24	F004h	Against CW/CCW limits
25	5441h	Negative Limit Switch Active
26	5442h	Positive Limit Switch Active
27	5443h	Reserved
28	5444h	Driver Interlock Inactive (Disabled)
29	5445h	Reserved
30	5446h	Reserved
31	5447h	Reserved

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2001h	0	U8	RO	Max Array Size	3		No
2001h	1	U32	RW	R/W EMCY Report Mask	0		No
2001h	2	U32	WO	Set bits EMCY Report Mask			No
2001h	3	U32	WO	Clear Bits EMCY Report Mask			No

2002 CAN Errors Reported Register

This register contains all Errors that have been reported, whether or not they have been enabled to produce a CAN_ERROR or whether or not they have been enabled to be reported via EMCY frames. These bits may be set high for test purposes, and may be cleared to remove backed up error reporting while unit was in a NMT "STOPPED" state, or EMCY disabled state. Bits so cleared may not be reported. The sending of the EMCY message with a cleared error for the related error bit also clears the bit in this register.

Note: bits 27, 29, and 30 are reserved. Bit 31 is User, meaning the user may trigger an error that is uniquely reported by setting via Sub-Index 2 bit 31. The resulting actions are defined via the bit configurations of objects 2000h and 2001h.

Bit #	Error Cause
0	Driver Over Temperature
1	Driver Over Voltage
2	Driver Under Voltage
3	Motor Over Temp
4	Encoder Analog Error
5	Motor Commutation Realign
6	Non-Volatile Memory Error
7	PDO data out of range
8	Position Error
9	Motion Error
10	Command error
11	Heartbeat Error
12	Error Passive
13	Protocol Error
14	Extended IO power missing
15	Drive Disabled
16	Velocity Limit Exercised
17	PDO Data Too Short
18	PDO Data Too Long
19	Processor Over Temp
20	Recovered from Bus Off
21	Bus Overload – new data can not be sent
22	HeartBeat changed to non-operational
23	CAN Error Warning
24	Against CW/CCW limits
25	Negative Limit Switch Active
26	Positive Limit Switch Active
27	Reserved
28	Driver Interlock Inactive (Disabled)
29	Reserved
30	Reserved
31	Reserved

Index	Sub	Types	Access	Purpose	Default	Ref	PDO
2002h	0	U8	RO	Max Array Size	3		No
2002h	1	U32	RW	R/W Reported Errors			No
2002h	2	U32	WO	Set bits Reported Errors (Simulate or errors)			No
2002h	3	U32	WO	Clear Bits Reported Errors			No

2003h Trigger Event Driven PDO

2003h Sub Index 1 to 4 corresponds to Transmit PDO1 through Transmit PDO4. Downloading (writing) any value (value ignored) to the object will trigger the corresponding Transmit PDO. This corresponds to triggered PDO types 0, Fen, Fifth. 0, Fen may be triggered by a time elapse or this trigger event mechanism; type Fifth is also triggered by a change in value of any of the mapped objects. This auxiliary trigger may originate within the node, or may be sent via CAN. These 4 objects are write only (WO).

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2003h				Trigger TPDO			No
2003h	00	U8	RO	Highest Sub-Index supported	4		
2003h	01	U8	WO	Trigger TPDO 1			
2003h	02	U8	WO	Trigger TPDO 2			
2003h	03	U8	WO	Trigger TPDO 3			
2003h	04	U8	WO	Trigger TPDO 4			

2004h Limit Switch and Home Switch Mapping

The Negative Limit Switch is configured via Sub-Index 1. The Positive Limit Switch is configured via Sub-Index 2, and the Home Switch is configured via Sub-Index 3. These three bits are mapped to bits inputs #1, #2, and #3 of the Advanced Stop Conditions, with a default mapping to IO1, IO2, and IO3, respectively.

The Interlock input is configured via Sub-Index 4, and defaults to Test 36, (FLGINP bit 7), which is the combination of External Driver Enable, Factory Enable, No Driver Over Temperature, No Over Voltage.

Mapping to IO is done by selecting the appropriate IO number. For example, to map the Home Switch to IO115, configure Index 2004, Sub-Index 3 to 115 (0073h). To configure the Home switch to IO115, but with an active low state, configure Index 2004, Sub-Index 3 to –115 (FF8Dh). Mapping the IO to a value of 0 disables (always returns the selected bit as 0 - inactive.

NOTE: These mappings may NOT be changed in NMT state "Operational"
NOTE: "Mapped Register" is a user register selected via Object 2008h.
In addition to mapping I/O bits, other status bits may be mapped. See the table below. Again, negating the value inverts sense of the input data.

Bit Test	Test	Word	Bit
0	No Test	N/A	N/A
1	IO1	FLGINP	Bit 4
2	IO2	FLGINP	Bit 5
3	IO3	FLGINP	Bit 6
4	IO4	FLGINP	Bit 12
5	IO5	FLGINP	Bit 13
6	IO6	FLGINP	Bit 14
7	IO7	FLGINP	Bit 15
8	Current Index found	FLGINP	Bit 0
9	Internal Index found	FLGINP	Bit 1
10	External Index found	FLGINP	Bit 2
11	Position Error	FLGINP	Bit 8
12	Motion Error	FLGINP	Bit 9
13	Trajectory Active	FLGINP	Bit 3
14	Delay Counter Active	FLGINP	Bit 10
15	Millisecond Delay Active	IS2	Bit 0
16	Encoder Re-phased	IS2	Bit 2
17	Driver Disable Factory	IS2	Bit 3
18	Motor Over Temp	IS2	Bit 4
19	Driver Analog Over Temp	IS2	Bit 5
20	Driver Not Enabled	IS2	Bit 6
21	Driver Digital Over Temp	IS2	Bit 7
22	Encoder Analog Error	IS2	Bit 8
23	External IO Power off	IS2	Bit 9
24	Velocity Limit Exercised	IS2	Bit 1
25	CAN Error	IS2	Bit 10
26	Thread 2 Active	IS2	Bit 11
27	CAN NMT is "Operational"	CAN_STATE	Bit 15
28	CAN initialized	CAN_STATE	Bit 14
29	CAN able to receive frames	CAN_STATE	Bit 13
30	Can able to process PDO	CAN_STATE	Bit 12
31	CAN NMT is "Stopped"	CAN_STATE	Bit 11
36	ALL Driver Enable OK	FLGINP	Bit 7
40	Mapped Register		Bit 0
41	Mapped Register		Bit 1
42	Mapped Register		Bit 2
43	Mapped Register		Bit 3
44	Mapped Register		Bit 4
45	Mapped Register		Bit 5
46	Mapped Register		Bit 6
47	Mapped Register		Bit 7
48	Mapped Register		Bit 8
49	Mapped Register		Bit 9
50	Mapped Register		Bit 10
51	Mapped Register		Bit 11
52	Mapped Register		Bit 12
53	Mapped Register		Bit 13
54	Mapped Register		Bit 14
55	Mapped Register		Bit 15
56	Mapped Register		Bit 16

57	Mapped Register		Bit 17
58	Mapped Register		Bit 18
59	Mapped Register		Bit 19
60	Mapped Register		Bit 20
61	Mapped Register		Bit 21
62	Mapped Register		Bit 22
63	Mapped Register		Bit 23
64	Mapped Register		Bit 24
65	Mapped Register		Bit 25
66	Mapped Register		Bit 26
67	Mapped Register		Bit 27
68	Mapped Register		Bit 28
69	Mapped Register		Bit 29
70	Mapped Register		Bit 30
71	Mapped Register		Bit 31
101 #	IO101	G_16_BITS	Bit 0
102 #	IO102	G_16_BITS	Bit 1
103 #	IO103	G_16_BITS	Bit 2
104 #	IO104	G_16_BITS	Bit 3
105 #	IO105	G_16_BITS	Bit 4
106 #	IO106	G_16_BITS	Bit 5
107 #	IO107	G_16_BITS	Bit 6
108 #	IO108	G_16_BITS	Bit 7
109 #	IO109	G_16_BITS	Bit 8
110 #	IO110	G_16_BITS	Bit 9
111 #	IO111	G_16_BITS	Bit 10
112 #	IO112	G_16_BITS	Bit 11
113 #	IO113	G_16_BITS	Bit 12
114 #	IO114	G_16_BITS	Bit 13
115 #	IO115	G_16_BITS	Bit 14
116 #	IO116	G_16_BITS	Bit 15

* 32 through 35 = Not available for mapping CAN switches, available for jumps
    and motion end conditions. They correspond to CAN_IO bits 0 to 3
= only if extended IO is present, Set to 1 if
    External IO power is not present.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2004h				Switch Mapping			No
2004h	00	U8	RO	Highest Sub-Index supported	4		
2004h	01	U16	RW	IO select for Negative Limit Sw.	1		
2004h	02	U16	RW	IO select for Positive Limit Sw.	2		
2004h	03	U16	RW	IO select for Home Sw.	3		
2004h	04	U16	RW	IO select for Interlock	36 (24h)		

2005h Heartbeat Monitoring Status/State

2005h Sub-Index 1 through 8 correspond to the nodes configured in 1016h Sub-Index 1 to 8, respectively. The upper 4 bits indicate the status of the heartbeat timer, while the lower 7 bits correspond to the most recently received Node NMT State.

Bit 15 is set to 1 if the Heartbeat is currently timed out.
Bit 14 is set to 1 if the first Heartbeat is still pending (no heartbeats received since configuration of Node via 1016.
Bit 13 is set to 1 if NMT states have changed
Bit 12 is set to 1 if NMT states changed to Pre-Operational or Stopped

Note: These objects are read/clear. The write operation may be used to clear the selected bits by writing a 1 to that value. Typically, only bits 12, 13, and/or 14 should be cleared. Bit 15 will immediately (within 1 ms) retrigger if no heartbeat has been detected. The Heartbeat Consumer may also be reset (back to pending bit set) by writing (downloading) to Object 1016 with the Sub-Index selecting the specific consumer; the same data should be configured.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2005h				Trigger TPDO			No
2005h	00	U8	RO	Highest Sub-Index supported	8		
2005h	01	U16	RW*	Status|State 1st HB Consumer			
2005h	02	U16	RW*	Status|State 2nd HB Consumer			
2005h	03	U16	RW*	Status|State 3rd HB Consumer			
2005h	04	U16	RW*	Status|State 4th HB Consumer			
2005h	05	U16	RW*	Status|State 5th HB Consumer			
2005h	06	U16	RW*	Status|State 6th HB Consumer			
2005h	07	U16	RW*	Status|State 7th HB Consumer			
2005h	08	U16	RW*	Status|State 8th HB Consumer			

*Actually a Read/Clear register. Bits written will be cleared from the object.

2006h Read/Clear CAN Hardware Error Status Bits

This object allows access to the hardware register that accumulates the various CAN error bits. This may be used to monitor the accumulation of various error bits as well as to clear out the non-critical bits. The self clearing bits may only be cleared by allowing the normal operation of the hardware CAN error recovery protocols; the other bits are latched and may be cleared by writing a "1" to them.

Bit0 = (self clearing) Error Warning (at least 1 error counter reached 96)
Bit1 = (self clearing) Error Passive Mode
Bit2 = (Self clearing) Bus-Off State (TEC reached 256 = no CAN Rx or Tx allowed
   until it recovers)
Bit3 = ACK error - we did not receive an acknowledge
Bit4 = Stuff bit error rule violated
Bit5 = CRC Error detected
Bit6 = Stuck-At-Dominant error - seen after Bus-Off recovery
Bit7 = Bit error flag - Rx bit did not match Tx bit outside arbitration field - or
   inside arbitration field, a dominant bit set, and a passive detected
Bit8 = Form Error Flag - fixed form field bit had wrong level

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2006h	00	U16	RW*	Read/Clear CAN Error Status Bits			No

 * Read/Clear, only bits 3:8 can be cleared by writing a 1 to the respective bits.

2007h Current CAN ERRORS Register

Provides the current state of the CAN ERRORS showed latched in Object 2002h. These bits are dynamically updated every 480uS. See Object 2002h for details on bit configuration.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2007h	00	U16	RO	Read CAN Error Register			No

2008h Remote Input Register Map

Selects which register Enable Codes 40 through 71 (Remote Input #1-32) uses. Select a USER REGISTER 0 through 199.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2008h	00	U16	RW	Register Mapping for Bit Test	199		No

2009h SSI Data Port

The SilverDust QCI-D2-IG8 provides an SSI port for synchronous data exchange. The data received from the SSI port is stored to Object 2009 Subindex 1, while data from Object 2009 Subindex 2 is transmitted to the SSI port. Both the reception and transmission are dependent upon the SSI port being configured for operation.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
2009h				Trigger TPDO			
2009h	0	U8	RO	Highest Sub-Index supported	2		
2009h	1	U32	RO	SSI Received Data			Yes
2009h	2	U32	RW	SSI Transmit Data			Yes

200Ah CAN Switch Data

Read back register for CAN switch Data (lower 8 bits) as well as related information. Only valid for units having CAN Switches (currently SilverDust IG8).

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
200Ah	00	U16	RO	CAN Switch Data			Yes

Bit 15	Input from IO3 of Ethernet Interface
Bit 14	Input from IO2 of Ethernet Interface
Bit 13	Reserved
Bit 12	Reserved
Bit 11	SSI MISO Input
Bit 10	SSI CS/Z Input
Bit 9	SSI CLK/B Input
Bit 8	SSI MOSI/A Input
Bit 7	High switch 8
Bit 6	High switch 4
Bit 5	High switch 2
Bit 4	High switch 1
Bit 3	Low switch 8
Bit 2	Low switch 4
Bit 1	Low switch 2
Bit 0	Low switch 1

Default IG8 configuration uses these switches to set unit ID (1..16 for positions 1..15,0), and CAN baud rate (Valid CAN baud rates; 1Mb/sec for invalid selections).

User Register Mapping to CAN Data Dictionary

The SilverLode Registers are mapped in multiple fashions to the Data Dictionary to allow them to be accessed as 8, 16, 24, or 32 bit numbers, either signed or unsigned. The User Register space is mapped to values 0 to 255 (see User Manual, not all 255 are implemented) are mapped as objects 2100h through 21FFh. The Access (RO, RW, etc.) for each register is as it is defined for the corresponding register. Attempted access to non-existent registers will produce the corresponding errors. The register may be accesses as a 32-bit number by referencing Sub-Index 1 of the User Register Object. The upper word of the register may be accessed by referencing Sub-Index 2, etc., See Table:

Index	Sub	Type	Purpose	Access	PDO
21xxh			User Register Access Type Mapping		
21xxh	0	U8	Number of Access types	RO	Yes
21xxh	1	U32	32 bit access to User Register	See User Manual	Yes
21xxh	2	U16	16 bit access to upper User Register	See User Manual	Yes
21xxh	3	U16	16 bit access to lower User Register	See User Manual	Yes
21xxh	4	U24	U24 access to User Register (no sign extension)	See User Manual	Yes
21xxh	5	I24	I24 access to User Register (sign extension to 32 bits)	See User Manual	Yes
21xxh	6	U8	8 bit access to lowest byte of register	See User Manual	Yes
21xxh	7	O32	32 bit OR to User Register	WO**	Yes
21xxh	8	O16	16 bit OR to upper User Register	WO**	Yes
21xxh	9	O16	16 bit OR to lower User Register	WO**	Yes
21xxh	A	O8	8 bit OR to lowest byte of register	WO**	Yes
21xxh	B	C32	32 bit Clear bits of User Register	WO**	Yes
21xxh	C	C16	16 bit Clear bits of upper User Register	WO**	Yes
21xxh	D	C16	16 bit Clear bits of lower User Register	WO**	Yes
21xxh	E	C8	8 bit Clear bits of lowest byte of register	WO**	Yes

**   Only avaliable if base register is RW.
O32, O16, O8, C32, C16, and C8 are special manufacturer data types which perform the set bits (OR) and clear bits functions. They are Write Only.

The **OR** functions are used to set one or more bits in the destination register without modifying unselected bits. The **Clear** function is used to clear one or more bits in the destination register without modifying unselected bits.

Objects 2100h to 21FCh

Index	Reg #	Access	Purpose High Word	Purpose Low Word	Notes
2100h	0	RW	Target Position		Make only slight/gradual adjustments to prevent rapid motions
2101h	1	RW	Actual Position		
2102h	2	RW	Last Index Position		
2103h	3	RO	Internal Status Word	Reserved	
2104h	4	RW	Last Trig Position		
2105h	5	RW	Delay Counter		
2106h	6	RW	Max Position Error	Current Position Error	
2107h	7	RO	Velocity 1	Velocity 2	
2108h	8	RO	Integrator Value		
2109h	9	RO	Reserved	Torque	
210Ah	10	RW	User Register 10		Thread 1 Register 10
210Bh	11	RW	User Register 11		
210Ch	12	RW	User Register 12		
210Dh	13	RW	User Register 13		
210Eh	14	RW	User Register 14		
210Fh	15	RW	User Register 15		
2110h	16	RW	User Register 16		
2111h	17	RW	User Register 17		
2112h	18	RW	User Register 18		
2113h	19	RW	User Register 19		
2114h	20	RW	User Register 20		
2115h	21	RW	User Register 21		
2116h	22	RW	User Register 22		
2117h	23	RW	User Register 23		
2118h	24	RW	User Register 24		
2119h	25	RW	User Register 25		
211Ah	26	RW	User Register 26		
211Bh	27	RW	User Register 27		
211Ch	28	RW	User Register 28		
211Dh	29	RW	User Register 29		
211Eh	30	RW	User Register 30		
211Fh	31	RW	User Register 31		
2120h	32	RW	User Register 32		
2121h	33	RW	User Register 33		
2122h	34	RW	User Register 34		
2123h	35	RW	User Register 35		
2124h	36	RW	User Register 36		
2125h	37	RW	User Register 37		
2126h	38	RW	User Register 38		
2127h	39	RW	User Register 39		
2128h	40	RW	User Register 40		
2129h	41	RW	User Register 41		
212Ah	42	RW	User Register 42		

Index	Reg #	Access	Purpose High Word	Purpose Low Word	Notes
212Bh	43	RW	User Register 43		
212Ch	44	RW	User Register 44		
212Dh	45	RW	User Register 45		
212Eh	46	RW	User Register 46		
212Fh	47	RW	User Register 47		
2130h	48	RW	User Register 48		
2131h	49	RW	User Register 49		
2132h	50	RW	User Register 50		
2133h	51	RW	User Register 51		
2134h	52	RW	User Register 52		
2135h	53	RW	User Register 53		
2136h	54	RW	User Register 54		
2137h	55	RW	User Register 55		
2138h	56	RW	User Register 56		
2139h	57	RW	User Register 57		
213Ah	58	RW	User Register 58		
213Bh	59	RW	User Register 59		
213Ch	60	RW	User Register 60		
213Dh	61	RW	User Register 61		
213Eh	62	RW	User Register 62		
213Fh	63	RW	User Register 63		
2140h	64	RW	User Register 64		
2141h	65	RW	User Register 65		
2142h	66	RW	User Register 66		
2143h	67	RW	User Register 67		
2144h	68	RW	User Register 68		
2145h	69	RW	User Register 69		
2146h	70	RW	User Register 70		
2147h	71	RW	User Register 71		
2148h	72	RW	User Register 72		
2149h	73	RW	User Register 73		
214Ah	74	RW	User Register 74		
214Bh	75	RW	User Register 75		
214Ch	76	RW	User Register 76		
214Dh	77	RW	User Register 77		
214Eh	78	RW	User Register 78		
214Fh	79	RW	User Register 79		
2150h	80	RW	User Register 80		
2151h	81	RW	User Register 81		
2152h	82	RW	User Register 82		
2153h	83	RW	User Register 83		
2154h	84	RW	User Register 84		
2155h	85	RW	User Register 85		
2156h	86	RW	User Register 86		
2157h	87	RW	User Register 87		
2158h	88	RW	User Register 88		

Index	Reg #	Access	Purpose High Word	Purpose Low Word	Notes
2159h	89	RW	User Register 89		
215Ah	90	RW	User Register 90		
215Bh	91	RW	User Register 91		
215Ch	92	RW	User Register 92		
215Dh	93	RW	User Register 93		
215Eh	94	RW	User Register 94		
215Fh	95	RW	User Register 95		
2160h	96	RW	User Register 96		
2161h	97	RW	User Register 97		
2162h	98	RW	User Register 98		
2163h	99	RW	User Register 99		
2164h	100	RW	User Register 100 Reserved for 402 object 607Ch		See 402V02 Object Mapping
2165h	101	RW	User Register 101 Reserved for 402 object 6098h		See 402V02 Object Mapping
2166h	102	RW	User Register 102 Reserved for 402 object 6099h		See 402V02 Object Mapping
2167h	103	RW	User Register 103 Reserved for 402 object 6099h		See 402V02 Object Mapping
2168h	104	RW	User Register 104 Reserved for 402 object 609Ah		See 402V02 Object Mapping
2169h	105	RW	User Register 105 Reserved for 402 object 60C5h		See 402V02 Object Mapping
216Ah	106	RW	User Register 106 Reserved for 402 object 60C6h		See 402V02 Object Mapping
216Bh	107	RW	User Register 107 Reserved for 402 object 6060h		See 402V02 Object Mapping
216Ch	108	RW	User Register 108 Reserved for 402 object 6061h		See 402V02 Object Mapping
216Dh	109	RW	User Register 109 Reserved for 402 object 605Ah		See 402V02 Object Mapping
216Eh	110	RW	User Register 110 Reserved for 402 object 605Bh		See 402V02 Object Mapping
216Fh	111	RW	User Register 111 Reserved for 402 object 605Ch		See 402V02 Object Mapping
2170h	112	RW	User Register 112 Reserved for 402 object 605Dh		See 402V02 Object Mapping
2171h	113	RW	User Register 113 Reserved for 402 object 605Eh		See 402V02 Object Mapping
2172h	114	RW	User Register 114 Reserved for 402 object 6081h		See 402V02 Object Mapping
2173h	115	RW	User Register 115 Reserved for 402 object		See 402V02 Object Mapping
2174h	116	RW	User Register 116 Reserved for 402 object 6083h		See 402V02 Object Mapping
2175h	117	RW	User Register 117 Reserved for 402 object 6084h		See 402V02 Object Mapping
2176h	118	RW	User Register 118 Reserved for 402 object 6085h		See 402V02 Object Mapping
2177h	119	RW	User Register 119 Reserved for 402 object		See 402V02 Object Mapping
2178h	120	RW	User Register 120 Reserved for 402 object 6040h		See 402V02 Object Mapping
2179h	121	RW	User Register 121 Reserved for 402 object 6041h		See 402V02 Object Mapping
217Ah	122	RW	User Register 122 Reserved for 402 object 1029h		See 402V02 Object Mapping
217Bh	123	RW	User Register 123 Reserved for 402 object 1029h		See 402V02 Object Mapping
217Ch	124	RW	User Register 124 Reserved for 402 object 607Ah		See 402V02 Object Mapping
217Dh	125	RW	User Register 125 Reserved for 402 object 607Fh		See 402V02 Object Mapping
217Eh	126	RW	User Register 126 Reserved for 402 object 6007h		See 402V02 Object Mapping
217Fh	127	RW	User Register 127		
2180h	128	RW	User Register 128		
2181h	129	RW	User Register 129		
2182h	130	RW	User Register 130		
2183h	131	RW	User Register 131		
2184h	132	RW	User Register 132		
2185h	133	RW	User Register 133		
2186h	134	RW	User Register 134		

Index	Reg #	Access	Purpose		Notes
			High Word	**Low Word**	
2187h	135	RW	User Register 135		
2188h	136	RW	User Register 136		
2189h	137	RW	User Register 137		
218Ah	138	RW	User Register 138		
218Bh	139	RW	User Register 139		
218Ch	140	RW	User Register 140		
218Dh	141	RW	User Register 141		
218Eh	142	RW	User Register 142		
218Fh	143	RW	User Register 143		
2190h	144	RW	User Register 144		
2191h	145	RW	User Register 145		
2192h	146	RW	User Register 146		
2193h	147	RW	User Register 147		
2194h	148	RW	User Register 148		
2195h	149	RW	User Register 149		
2196h	150	RW	User Register 150		
2197h	151	RW	User Register 151		
2198h	152	RW	User Register 152		
2199h	153	RW	User Register 153		
219Ah	154	RW	User Register 154		
219Bh	155	RW	User Register 155		
219Ch	156	RW	User Register 156		
219Dh	157	RW	User Register 157		
219Eh	158	RW	User Register 158		
219Fh	159	RW	User Register 159		
21A0h	160	RW	User Register 160		
21A1h	161	RW	User Register 161		
21A2h	162	RW	User Register 162		
21A3h	163	RW	User Register 163		
21A4h	164	RW	User Register 164		
21A5h	165	RW	User Register 165		
21A6h	166	RW	User Register 166		
21A7h	167	RW	User Register 167		
21A8h	168	RW	User Register 168		
21A9h	169	RW	User Register 169		
21AAh	170	RW	User Register 170		
21ABh	171	RW	User Register 171		
21ACh	172	RW	User Register 172		
21ADh	173	RW	User Register 173		
21AEh	174	RW	User Register 174		
21AFh	175	RW	User Register 175		
21B0h	176	RW	User Register 176		
21B1h	177	RW	User Register 177		
21B2h	178	RW	User Register 178		
21B3h	179	RW	User Register 179		
21B4h	180	RW	User Register 180		

Index	Reg #	Access	High Word	Low Word	Notes
			Purpose		
21B5h	181	RW	User Register 181		
21B6h	182	RW	User Register 182		
21B7h	183	RW	User Register 183		
21B8h	184	RW	User Register 184		
21B9h	185	RW	User Register 185		
21BAh	186	RW	User Register 186		
21BBh	187	RW	User Register 187		
21BCh	188	RW	User Register 188		
21BDh	189	RW	User Register 189		
21BEh	190	RW	User Register 190		
21BFh	191	RW	User Register 191		
21C0h	192	RW	User Register 192		
21C1h	193	RW	User Register 193		
21C2h	194	RW	User Register 194		
21C3h	195	RW	User Register 195		
21C4h	196	RW	User Register 196		
21C5h	197	RW	User Register 197		
21C6h	198	RW	User Register 198		
21C7h	199	RW	User Register 199		
21C8h	200	RW	External Encoder Position		
21C9h	201	RW	External Index Position		
21CAh	202	RO	Reserved		
21CBh	203	RO	Reserved		
21CCh	204	RO	Target Acceleration		
21CDh	205	RO	Target Velocity		
21CEh	206	RW	Closed Loop Torque Hold	Closed Loop Torque Move	
21CFh	207	RW	Open Loop Torque Hold	Open Loop Torque Move	
21D0h	208	RW	Error Limit Moving	Error Limit Holding	
21D1h	209	RO	Sense Mask	IO Status Word	
21D2h	210	RO	Program Buffer Size	Program Buffer Start	
21D3h	211	RW	Kill Motor Conditions ISW	Kill Motor States ISW	Cause of KMR
21D4h	212	RO	Analog Input 1	Analog Input 2	A/D from IO4 and IO5, respectively
21D5h	213	RO	Analog Input 3	Analog Input 4	A/D from IO6 and IO7, respectively
21D6h	214	RO	Driver Volt	Processor Temp	
21D7h	215	RO	N2/N3 Process Volt	N2/N3 Analog Driver Temp	
21D8h	216	RO	Max Driver Volt	Driver Cal	
21D9h	217	RO	Max HC Driver Temp	HC Processor Volt Cal	
21DAh	218	RW	Reserved		
21DBh	219	RW	Group ID	Unit ID	
21DCh	220	RW	DIF IO Line 1 Filter Constant	DIF IO Line 1 Filter Count	
21DDh	221	RW	DIF IO Line 2 Filter Constant	DIF IO Line 2 Filter Count	
21DEh	222	RW	DIF IO Line 3 Filter Constant	DIF IO Line 3 Filter Count	
21DFh	223	RW	DIF IO Line 4 Filter Constant	DIF IO Line 4 Filter Count	
21E0h	224	RW	DIF IO Line 5 Filter Constant	DIF IO Line 5 Filter Count	
21E1h	225	RW	DIF IO Line 6 Filter Constant	DIF IO Line 6 Filter Count	
21E2h	226	RW	DIF IO Line 7 Filter Constant	DIF IO Line 7 Filter Count	
21E3h	227	RO	Reserved		

Index	Reg #	Access	Purpose High Word	Low Word	Notes
21E4h	228	RW	Reserved		
21E5h	229	RO	Reserved		
21E6h	230	RO	Reserved		
21E7h	231	RO	Reserved		
21E8h	232	RO	Reserved		
21E9h	233	RO	Reserved		
21EAh	234	RO	Encoder CPR	Encoder Modulo Position	Locked in once index has been found
21EBh	235	RO	Reserved		
21ECh	236	RO	IS2	Reserved	
21EDh	237	RW	Reserved		
21EEh	238	RW	XIO In	XIO Output	Only write to lower word
21EFh	239	RW	Reserved		
21F0h	240	RW	Reserved		
21F1h	241	RW	Motor Max Temp	Motor Temp	Available on IP65 motors
21F2h	242	RO	Reserved		
21F3h	243	RW	Reserved		
21F4h	244	RW	Millisecond Free running Count Up Timer		
21F5h	245	RW	Millisecond Count Down timer		
21F6h	246	RO	CAN Error Register	CAN State	
21F7h	247	RC	CANESR	CANGSR	Read/Clear (not all bits are clearable)
21F8h	248	RW	Thread 2 local copy of Register 10		
21F9h	249	RW	Reserved		
21FAh	250	RW	Reserved		
21FBh	251	RO	Reserved		
21FCh	252	RO	Reserved		

402V02 Object Mapping

Device Profiles are used to establish common object usage for common functions. The 402 profile defines such common object usage for Servo and Stepper Drives.

Some of these objects are mapped to user registers in the SilverLode memory map. These are intended to provide an interface to the User Program running on the SilverLode controller that then implements the requested function via program control. The balance of the objects are internal data used by the CANopen processing routines to configure operation.]

For Further information on CiA 402 implementation see AN060 CiA 402 Implementation

Register	Description	Type	Mapping	Object	Sub
100	Home Offset	I32	Long Word	607Ch	00
101	Homing Method	I8	Lowest Byte	6098h	00
102	Homing Speed Switch	U32	Long Word	6099h	01
103	Homing Speed Zero	U32	Long Word	6099h	02
104	Homing Acceleration	U32	Long Word	609Ah	00
105	Max Acceleration	U32	Long Word	60C5h	00
106	Max Deceleration	U32	Long Word	60C6h	00
107	Mode of Operation: Command	I8	Lowest Byte	6060h	00
108	Mode of Operation: Display	I8	Lowest Byte	6061h	00
109	Quick Stop Option	I16	Low Word	605Ah	00
110	Shutdown Option	I16	Low Word	605Bh	00
111	Disable Operation Option Code	I16	Low Word	605Ch	00
112	Halt Option Code	I16	Low Word	605Dh	00
113	Fault Reaction Operation Code	I16	Low Word	605Eh	00
114	Profile Velocity	U32	Long Word	6081h	00
115	Reserved - not used				
116	Profile acceleration	U32	Long Word	6083h	00
117	Profile deceleration	U32	Long Word	6084h	00
118	Quick Stop Deceleration	U32	Long Word	6085h	00
119	Reserved – not used				
120	Control Word	U16	Low Word	6040h	00
121	Status Word	U16	Low Word	6041h	00
122	Error Behavior - Communications	U8	Lowest Byte	1029h	01
123	Error Behavior - Internal	U8	Lowest Byte	1029h	02
124	New Target Position	I32	Long Word	607Ah	00
125	Max Profile Velocity	U32	Long Word	607Fh	00
126	Abort Connection Option	I16	Lower Word	6007h	00
127	Supported Drive Modes	U32	Long Word	6502h	00
128	Lower Position Limit	I32	Long Word	607D	01
129	Upper Position Limit	I32	Long Word	607D	02

6007h Abort Connection Option Code

This object selects the drive reaction to loss of network connection:

        0 = no action
        1 = malfunction
        2 = Device Control command "Disable Voltage"
        3 = Device Command "Quick Stop"

The user program code interprets this code in the case of loss of network connection. This object is mapped to the lower word of Register 126.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6007h	00	I16	RW	Abort Connection Option	0	402v02	No

603Fh – Most Recent Error Code

Most Recent Error Code reflects the lower 16 bits of Object 1003h Sub-Index 1. See Object 1001h for a listing of error codes. This error code is Not cleared when the error clears, but is cleared by writing a 0 to Object 1003h Sub-Index 0

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
603Fh	00	U16	RO	Most Recent Error		402v02	Yes

6040h Control Word

This object is used to request the new **Operation Mode** and **State Machine** State. The **State Machine** is used to determine and control the readiness of the machine to accept power and to enable the drive. It also handles fault shutdown and recovery.

Object 6040h is used to request a wanted mode or state, which, when accepted, is reflected in the value of object 6041h – Status Word. These to words are used to handshake between the drive the master unit. The several bits in both the Control Word and the Status Word change their use according to the mode selected.

See **Control Word and Status Word – System State Machine** chapter.

The Control Word is mapped to the low word of Register 120

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6040h	00	U16	RW	Control Word		402v02	Yes

6041h Status Word

This object is used to indicate the present **Operating Mode** and
State Machine status. It is used with Object 6040h to provide a feedback response to
the requested State and Mode.

See **Control Word and Status Word – System State Machine** chapter.

The Status Word is mapped to the low word of Register 121

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6041h	00	U16	RO	Status Word		402v02	Yes

605Ah Quick Stop Options

This object is used to determine the reaction to a **Quick Stop** condition triggered by to
change in Machine State to Quick Stop. Various options are available:

1=slow stop ramp -> switch on disabled
2 = quick stop ramp -> switch on disabled
3 = stop abruptly -> switch on disabled
4 ** = slow down on voltage limit -> switch on disabled
5 = slow stop ramp -> quick stop
6 = quick stop ramp -> quick stop
7 = stop abruptly -> quick stop
8 ** = slow down on voltage limit -> quick stop

Mode 3 and 4 produce the same motion with the onboard clamp present. Mode 7 and 8
produce the same motion with the onboard clamp present.

The Quick Stop Options object is mapped to the lower word of Register 109. The User
Program implements its functionality, and must also set the default state prior to CAN
startup.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
605Ah	00	I16	RW	Quick Stop Options	2	402v02	No

605Bh Shutdown Option

The Shutdown option determines what action to take if there is Machine State transition
OPERATION ENABLE => READY TO SWITCH ON

0 = Disable drive function
1 = Slow down with slow down ramp, then disable the drive function

The Shutdown Option is mapped to the low word of Register 110. Functionality is implemented by User Program.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
605Bh	00	I16	RW	Shutdown Option		1 402v02	No

605Ch Disable Option

The Disable option determines what action to take if there is Machine State transition
OPERATION ENABLE => SWITCHED ON

0 = Disable drive function
1 = Slow down with slow down ramp, then disable the drive function

The Disable Option is mapped to the low word of Register 111. Functionality and initialization is implemented by User Program.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
605Ch	00	I16	RW	Disable Option		1 402v02	No

605Dh Halt Option

The Halt Option determines the action to be taken if Bit8 (halt) of 6040h (Control Word) is set active.

  0 = Reserved
  1 = slow down on slow down ramp
  2 = slow down on quick stop ramp
  3 = slow down on current limit
  4 = slow down on voltage limit

The Halt Option is mapped to the low word of Register 112. Functionality is implemented by User Program.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
605Dh	00	I16	RW	Halt Option	1	402v02	No

605Eh Fault Reaction Option

The Fault Reaction Option determines what action should be taken of a fault occurs in the drive.

  0 = disable drive, motor is free to rotate
  1 = slow down on slow down ramp
  2 = slow down on quick stop ramp
  3 = slow down on current limit
  4 = slow down on voltage limit

The Fault Option is mapped to the low word of Register 113. Functionality is implemented by User Program.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
605Eh	00	I16	RW	Fault Reaction Operation	2	402v02	No

6060h Modes of Operation

Modes of Operation Register is used to select the wanted mode of operation:

-1 to -128 = manufacturer specific (user defined)
0 = reserved
1 = Profile Position Mode (pp)
2 = Not supported (Velocity Mode)
3 = Profile Velocity Mode (pv)
4 = Torque Profile mode (tq)
5 = reserved
6 = Homing Mode
7 = Not yet supported (Interpolated Position Mode)
8 to 127 = reserved

Mode of Operation is mapped to Register 107, lowest byte. Both Default value and the interpretation of the mode are implemented in the user program.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6060	00	I8	RW	Operation Mode	1	402v02	Yes

6061h Modes of Operation Display

Displays the current Mode of Operation. Data values correspond to 6060h Modes of Operation. The new mode is not accepted until the previous mode has completed.

Mode of Operation Display is mapped to Register 108, lowest byte. Update of this values representing the current mode is implemented in the user program.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6061h	00	I8	RO	Operation Mode Display	1	402v02	Yes

6062h Position Demand Value

Displays the current Position Demand Value (Target Position). This is the same data as User Register 0.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6062h	00	I32	RO	Position Demand Value		402v02	Yes

6063h Position Actual Value

Displays the current Actual Position Value (Actual Position). This is the same data as User Register 1.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6063h	00	I32	RO	Position Actual Value		402v02	Yes

6064h Position Actual Value

This Register displays the same information as 6063h. Displays the current Actual Position Value (Actual Position). This is the same data as Register 1.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6064h	00	I32	RO	Position Actual Value		402v02	Yes

607Ah New Target Position

Defines the new Target Position. It may be either absolute or relative, according to the state of the ABS/REL bit in the Control Word.

New Target Position is mapped to Register 124. The user program is responsible for updating the trajectory generator from this value.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
607Ah	00	I32	RW	New Target Position		402v02	Yes

607Ch Home Offset

Home Offset is the difference between the zero position for the application and the machine home position found during homing, in encoder counts. During the homing motion, the machine home position is found and once the homing is completed, the zero position is offset from the home position by adding the home offset to the home position.

User program is responsible to implement this function. Home offset is mapped to register 100.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
607Ch	00	I32	RW	Home Offset	0	402v02	Yes

607Dh Position Limits Array

The Position Limits Array defines the Lower and Upper software limits. Distance is in encoder counts relative to the home zero point. If both values are equal or the upper is less than the lower limit, then the limits are not engaged.

The limits must be configured via user code.

Lower Limit is mapped to Register 128.
Upper Limit is mapped to Register 129.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
607Dh				Homing Speeds Array		402v03	
607Dh	00	U8	RO	Highest Sub-Index supported	2		No
607Dh	01	I32	RW	Lower Software Position Limit	0		Yes
607Dh	02	I32	RW	Upper Software Position Limit	0		Yes

607Fh Maximum Profile Velocity

Maximum Profile Velocity is the maximum slowed speed (magnitude) during a profiled move. It has the same units as Profile Velocity.

The user code is responsible for implementing this limiting function. Maximum Profile Velocity is mapped to Register 125.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
607Fh	00	U32	RW	Maximum Profile Velocity		402v02	Yes

6081h Profile Velocity

Profile Velocity is the velocity normally attained at the end of the acceleration ramp during a profiled move. The units are in SilverLode units (unless converted by user program).

The user code is responsible for implementing limits and setting up the profiled move operation. Profile Velocity is mapped to Register 114

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6081h	00	U32	RW	Profile Velocity		402v02	Yes

6083h Profile Acceleration

Profile Acceleration in SilverLode units (unless converted by user program).

The user code is responsible for setting up the profiled move operation. Profile Acceleration is mapped to Register 116.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6083h	00	U32	RW	Profile Acceleration		402v02	Yes

6084h Profile Deceleration

Profile Deceleration in SilverLode units (unless converted by user program).

The user code is responsible for setting up the profiled move operation. Profile Deceleration is mapped to Register 117.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6084h	00	U32	RW	Profile Deceleration		402v02	Yes

6085h Quick Stop Deceleration

Determines the deceleration used if the 'Quick Stop' command is given and the Quick Stop Option Code (605Ah) is set = 2. The units are the same as for Profile Acceleration.

User Code is responsible for implementation. Quick Stop Deceleration is mapped to Register 118.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6085h	00	U32	RW	Quick Stop Deceleration		402v02	Yes

6098h Homing Method

Homing Method determines the method that will be used during homing. Methods 1..35 are defined in 402v02 section 13.4.1.1 (See Homing Methods Section). Methods –1 to -128 are available for custom methods, to be implemented in user code.

The homing procedure, including decoding the method, is implemented in user code. Homing Method is mapped to Register 101.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6098h	00	I8	RW	Homing Method		402v02	Yes

6099h Homing Speeds Array

The homing Speeds Array defines the speeds used during homing. Speed is given in SilverLode units, unless converted by user program. The user program is responsible for implementing the homing routines, including configuring velocity.

Speed during search for Switch is mapped to Register 102.
Speed during search for Zero is mapped to Register 103.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6099h				Homing Speeds Array		402v02	
6099h	00	U8	RO	Highest Sub-Index supported	2		No
6099h	01	U32	RW	Speed during search for Switch	0		Yes
6099h	02	U32	RW	Speed during search for Zero	0		Yes

609Ah Homing Acceleration

Homing Acceleration determines the acceleration used during the Homing Operation. It is given in SilverLode Acceleration units unless converted by the user program. The user program is responsible for implementing the homing procedures.

Homing Acceleration is mapped to Register 104.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
609Ah	00	U32	RW	Homing Acceleration		402v02	Yes

60C5h Maximum Acceleration

Maximum Acceleration defines the maximum acceleration for all operations. It is in SilverLode Acceleration units unless converted by user program. The user program is responsible for implementing the limiting function.

Maximum Acceleration is mapped to Register 105.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
60C5h	00	U32	RW	Maximum Acceleration		402v02	Yes

60C6h Maximum Deceleration

Maximum Deceleration defines the maximum deceleration for all operations. It is in SilverLode Acceleration units unless converted by user program. The user program is responsible for implementing the limiting function.

Maximum Deceleration is mapped to Register 106.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
60C6h	00	U32	RW	Maximum Deceleration		402v02	Yes

60F2h Position Demand Value

Position Demand Value is the output from the Trajectory Generator defining what position is currently being provided as the desired position for the position control loop. It is in Encoder Counts.

Position Demand Value is an alternate mapping of Register 0.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
60F2h	00	I32	RO	Position Demand Value		402v02	Yes

60F4h Following Error Actual Value

Position Error is the difference in encoder counts between Position Demand Value and Actual Position. It is in Encoder Counts.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
60F4h	00	I32	RO	Following Error Actual Value		402v02	Yes

60FCh Position Demand Value

This is a duplicate of 60F2h.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
60FCh	00	I32	RO	Position Demand Value		402v02	Yes

60FDh Digital Inputs

Provides links to the various IO. The Negative, Positive, and Home Switches are user definable via 2004H, defaulting to IO 1, 2, and 3, respectively. The interlock is the Driver Enable input, forming both a hardware and software interlock.

Bit 0 = Negative Limit Switch
Bit 1 = Positive Limit Switch
Bit 2 = Home Switch
Bit 3 = Interlock
Bits 16:31 = Extended IO bits 101 through 116 (copy from G_16_BITS each cycle)

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
60FDh	00	U32	RO	Digital Input Word		402v02	Yes

60FEh Digital Outputs

Provides link to Digital Outputs 101 through 116.

Sub-Index 01 allows reading and writing the IO bits (once Sub-Index 02 and 03 have been configured). IO101 through IO116 are mapped to bits 16 through 31, respectively. Bits 0 through 15 are reserved. They are not acted upon.

Sub-Index 02 is a gating mask to enable output bits to be altered via the Digital Output command, preventing non-gated bits from being altered. This allows a division between IO which is locally controlled and that which may be controlled through the CAN bus. Setting a 1 in the mask allows the corresponding IO to be updated when Sub-Index 01 is written.

Sub-Index 03 selects the physical output state corresponding to the logical output state. A "0" in the corresponding bit will cause the output to be non-inverting, while a "1" causes the output to be inverted. Inverting the output is useful to allow VIO+ referenced devices, such as solenoids, to be energized when a "1" is output to the digital output word. Non-Inverted outputs allow connection to other logic inputs with a high output for a "1" input. Again bits 16 through 31 correspond to IO101 through IO116.

Sub-Index 02 and 03 must be configured before writing to Sub-Index 01, or the action will be ignored. The IO is not affected by writing to Sub-Index 02 or 03 until Sub-Index 01 is written. Note that reading back the IO, the same inversion from Sub-Index 03 is applied so that the written data should produce the same read results for those bits which are enabled in Sub-Index 02.

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
60FEh				Homing Speeds Array		402v02	
60FEh	0	U8	RO	Highest Sub-Index supported	2		No
60FEh	1	U32	RW	IO Word	0		Yes
60FEh	2	U32	RW	IO Mask	0		Yes
60FEh	3	U32	RW	IO Sense	0		No

6502h Supported Drive Modes

Mapped to User Register 127. Indicates those modes supported by the drive. These are implemented in user code, so this register must be initialized to indicate the modes that have been implemented.

Bit 0 = pp   Position Profile
Bit 1 = vi   Velocity
Bit 2 = pv   Profile velocity
Bit 3 = tq   Torque mode
Bit 4 = r   Reserved
Bit 5 = hm   Homing
Bit 6 = ip   Interpolated Profile
Bit 7..15   Additional reserved modes
Bit 16..31   Manufacturer Specific Modes

Index	Sub	Type	Access	Purpose	Default	Ref	PDO
6502h	00	U32	RO	Digital Input Word	No	402v03	Yes

67FFh Single Device Type

This is a duplicate of Device Type (1000h)

Describes the type of device and its functionality. Defines a Servo Drive with configurable PDOs

Index	Sub	Type	Access	Purpose	Value	Ref	PDO
67FFh	00	U32	RO	Device Type = Servo Drive	0002 0192h	402v02	No