ASI ACCELERATED SYSTEMS INC.

ASI CANopen Communication Protocol & Implementation

Created by: Evin Ballantyne & Terry Stone

ASI ACCELERATED SYSTEMS INC.

Contents

ASI ACCELERATED SYSTEMS INC.

Document Revision

Version	Date	Author	Change Description
1.0.0	08/25/2017	Evin Ballantyne & Terry Stone	Document Created
1.1.0	01/23/2018	Evin Ballantyne	Fixed SDO Write Response

ASI ACCELERATED SYSTEMS INC.

Over View

This document contains ASI's implementation of the CANopen communication protocol on CAN-enabled ASI Controllers. Not all functions within CANopen are implemented. This document is intended for customers trying to communicate with the controller, or are building peripherals that communicate with the controller. An Object Dictionary should be supplied with this document and used as a reference.

Definitions

The following definitions are used throughout the document.

Term	Description
CAN	Controller Area Network
Bus	The transmission line carrying the signal (bits) between all connected devices.
CANopen	Protocol used to communicate with the CAN Device
Bit	0 or 1
Byte	8 Sequential bits
Word	2 Sequential Bytes
Message	A series of bytes with an identifying header
Header	Metadata identifying a message
Object Dictionary	Large list of registers you can access in the CAN Device
Object Dictionary Index	An address for a block of objects in the Object Dictionary
Object Dictionary Sub Index	An address within a block of objects in the Object Dictionary
Dictionary Object	A single value in the dictionary.
Node	A device on the CAN network
Node Id	A unique id given to the node to identify itself on the network.
COB-ID	Can Object Identifier used in the header to uniquely identify the type of message. Often used in conjunction with the Node Id.
Client	The node making the request in a transmit/receive protocol
Server	The node making the response in a transmit/receive protocol
Little Endian	Byte sequence is interpreted from Least Significant Byte to Most Significant Byte

CANopen Settings

The following functions are standard setting required for CANopen. Most of ASI's settings are configurable on a per-device basis.

Function	Description
Baud rate	1000Kb/s, 800Kb/s, 500Kb/s, 250Kb/s, 125Kb/s, 50Kb/s, 20Kb/s, or 10Kb/s,
Node ID	Ranging from 0x01 to 0x7F
CAN Heartbeat Period (ms)	Ranging from 0x0010 to 0x7FFF (or 0x0 to disable)
CAN Sync Loss Timeout (ms)	Ranging from 0x0001 to 0x7FFF (or 0x0 to disable)

CANopen Supported Communication Calls

The following are messaging protocols that ASI CAN-enabled devices support. More detail on these protocols and their implementations are given in later sections.

Function	Description
NMT	Network Management Protocols
SDO	Service Data Objects
PDO	Process Data Objects
SYNC	Synchronization Object
EMCY	Emergency Objects

CANopen Message Format

The following is the typical representation for all CANopen messages, transmitting or receiving.

Bits 0 to 10	Bit 11	Bits 12 to 15	Variable Length (0 to 8 Bytes)
COB-ID	Rtr bit (not used)	Data Length	Data

Message Prioritization

If multiple messages are sent to the bus at the same time, the message with the lowest COB-ID takes priority.

Endianness

CANopen stores and transfers its data bytes in little endian order. When reading this document, any variable that requires more than one byte should be interpreted as little endian.

For example: Consider the following table.

Byte …	Byte 1	Byte 2	Byte …
…	Index		…

If we have an Index of 0x1122, because it is stored as little endian, **byte 1** would be 0x22, and **byte 2** would be 0x11.

Service Data Objects (SDO)

This is a basic transmit & response protocol. Used for one-time read & writes to Data Objects in the Object Dictionary. All Dictionary objects are four bytes or less in length (typically two bytes), so only SDO-Expedited Messages are necessary. SDO objects can be transmitted while the node is in a pre-operational state, or an operational state (see NMT commands).

Item	Description
CAN Header	Contains Can Network Instruction & Node ID
Rtr	Remote Transmit bit
Length	Data Length
Command	Server Command
Index	Dictionary Object Block location
Sub Index	Dictionary Object Sub-Block Location
Data	Transmitted Data

SDO-Expedited Read

The following is the message format to read from the CAN Object Dictionary.

Client Transmit Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600 + Node Id	0	8	Command	Index		Sub Index	0			

Valid Commands:

Command Code	Meaning
0x40	Read Dictionary Object.

Server Response Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580 + Node Id	0	8	Command	Index		Sub Index	Data			

Valid Commands:

Command Code	Meaning
0x43	Read Dictionary Expedited Response. 4 bytes returned.
0x47	Read Dictionary Expedited Response. 3 bytes returned.
0x4B	Read Dictionary Expedited Response. 2 bytes returned.
0x4F	Read Dictionary Expedited Response. 1 byte returned.

SDO-Expedited Write

The following is the message format to write to the CAN Object Dictionary.

Client Transmit Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600 + Node Id	0	8	Command	Index		Sub Index	Data			

Valid Commands:

Command Code	Meaning
0x23	Write Dictionary Expedited Request. 4 bytes to write.
0x27	Write Dictionary Expedited Request. 3 bytes to write.
0x2B	Write Dictionary Expedited Request. 2 bytes to write.
0x2F	Write Dictionary Expedited Request. 1 byte to write.

Server Response Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580 + Node Id	0	8	Command	Index		Sub Index	Data			

Valid Commands:

Command Code	Meaning
0x60	Write Complete.

SDO-Expedited Error Responses

If something should go wrong during an SDO Read or write, the server side will respond with the following message.

Server Response Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580 + Node Id	0	8	0x80	Index		Sub Index	Error Code			

Valid Error Codes:

Error Code	Meaning
0x05040001	Client/Server command specifier not valid or unknown.
0x06010000	Unsupported access to an object.
0x06020000	Object does not exist in object dictionary.
0x06070010	Data type does not match length. Length of Service Parameter does not match.
0x06090011	Sub Index does not exist.

Non-Data Communication Protocols

The following protocols and message formats are supported.

Network Management Protocol (NMT)

The following message will set the operational state of the node. The node's operational state can be viewed in the heartbeat message.

Client Transmit Message:

CAN Header	Rtr	Length	Byte 0	Byte 1
0x000	0	2	Command	Node Id

Valid Commands:

Command Code	Meaning
0x01	Enter Operational State
0x02	Enter Stopped State
0x80	Enter Pre-Operational State
0x81	Reset Node
0x82	Reset Communication

SYNC Message

The Sync message is used to keep nodes in an operational state and tell nodes to process or transmit their PDOs. If the SYNC message is not broadcast before a Node's SYNC Loss timeout setting, the node will go from an operational state to a Pre-operational state.

Client Transmit Message:

CAN Header	Rtr	Length	Bytes
0x80	0	0	*(None)*

Heartbeat Protocol

Usually, a node will periodically send out a message while operational to signify it still exists on the network. It is not required that a node does this.

Server Transmit Message:

CAN Header	Rtr	Length	Byte 0
0x700 + Node Id	0	1	State

Valid States:

State Code	Meaning
0x00	Boot up / Initializing
0x04	Stopped
0x05	Operational
0x7f	Pre-Operational

Emergency Protocol (EMCY)

Whenever there is a change in the faults, faults2, or warnings, the node will broadcast the following message. The controller will only broadcast this message once per change.

Server Transmit Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x080 + Node Id	0	8	Emergency Code		Error Register	Faults 2	Faults 1		Warnings	

Valid Emergency Codes:

Emergency Codes	Meaning
0x1000	Manufacturer Defined Error Codes

Error Register Bit Structure:

Error Register Bits	Meaning
0	Generic Error
1	Current

2	Voltage
3	Temperature
4	Communication Error
5	Device Profile Specific
6	Reserved (always 0)
7	Manufacturer Specific

Faults 2 Bit Structure:

Faults 2 Bits	Meaning
0	Parameter CRC
1	Current Scaling
2	Voltage Scaling
3	Headlight Under Voltage
4	Torque Sensor
5	CAN Bus
6	Hall Stall
7	Bootloader

Faults 1 Bit Structure:

Faults 1 Bits	Meaning
0	Controller Over Voltage
1	Phase Over Current
2	Current Sensor Calibration
3	Current Sensor Over Current
4	Controller Over Temperature
5	Motor Hall Sensor Fault
6	Controller Under Voltage
7	POST Static Gating Test
8	Network Communication Timeout
9	Instantaneous Phase Over Current
10	Motor Over Temperature
11	Throttle Voltage Outside Range

12	Instantaneous Controller Over Voltage
13	Internal Error
14	POST Dynamic Gating Test
15	Instantaneous Under Voltage

Warnings Bit Structure:

Warnings Bit	**Meaning**
0	Communication Timeout
1	Hall Sensor
2	Hall Stall
3	Wheel Speed Sensor
4	CAN Bus
5	Hall Illegal Sector
6	Hall Illegal Transition
7	Vdc Low Foldback
8	Vdc High Foldback
9	Motor Temperature Foldback
10	Control Temperature Foldback
11	Low SOC Foldback
12	Hi SOC Foldback
13	I2t Foldback
14	*Reserved (not used)*
15	BMS timeout

Process Data Objects (PDO)

Process Data objects are an efficient means of transferring dictionary objects between different nodes. PDOs can only be transmitted and received while the node is in operational mode (see NMT commands).

Transmit PDO (TPDO)

A node will transmit a TPDO depending on its settings. Message structure is as follows:

Node Transmit Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
TPDO Id	0	TPDO Length	TPDO Data 1 *(If set)*		TPDO Data 2 *(If set)*		TPDO Data 3 *(If set)*		TPDO Data 4 *(If set)*	

TPDO Id: TPDO Id is fixed, but typically will be 0x**[X]**80 + Node id, where **[X]** is 1 to Max number of TPDO's assigned by ASI. For example: if the Node ID of the controller is 0x2a, and it's broadcasting its second TPDO, the TPDO Id will be 0x2AA.

TPDO Length: Any length between 2 to 8 bytes.

TPDO Data: The Data can represent anywhere between one to four dictionary objects, depending on how the TPDO has been mapped.

Receive PDO (RPDO)

Anytime a node receives a message that matches it's assigned RPDO, it will write those data values to the dictionary. The message structure the node will receive is as follows:

Node Receive Message:

CAN Header	Rtr	Length	Byte 0	Byte 1	Byte2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
RPDO Id	0	RPDO Length	RPDO Data 1 *(If set)*		RPDO Data 2 *(If set)*		RPDO Data 3 *(If set)*		RPDO Data 4 *(If set)*	

RPDO Id: RPDO Id is fixed, but typically will be 0x**[X+1]**00 + Node id, where **[X]** is 1 to Max number of RPDO's assigned by ASI. For example: if the Node ID of the controller is 0x2a, and it's receiving its first RPDO, the RPDO Id will be 0x22a.

RPDO Length: Any length between 2 to 8 bytes.

RPDO Data: The Data can be written to anywhere between one to four dictionary objects, depending on how the RPDO has been mapped.

Adjusting PDO

ASI's implementation of adjusting the PDOs is done indirectly through the object dictionary. While this is not entirely in line with CANopen, it means that the PDOs can be adjusted in BACDoor via TTL, RS485, or BluetoothLE. The following sections list the name of the dictionary, as well as the expected entry in the object dictionary. You can find these in PC BACDoor under the "Communications" tab.

Adjusting Transmit PDO (TPDO)

The following are the Dictionary objects used to adjust a TPDO. All objects are two bytes in size.

TPDO Size: Number of objects being broadcast

TPDO Transmission Type:

Value	Meaning
0	Acyclic Synchronous: TPDO will transmit on a SYNC Message provided an event time has previously occurred.
1…240	Cyclic Synchronous: TPDO will transmit on a SYNC Message. If value is set higher than one, TPDO will wait that many SYNC messages before transmitting. Example: if the value is set for 4, the TPDO will transmit every fourth SYNC message.
241…244	*(Reserved: Not Used)*
255 or 256	Asynchronous: The TPDO will transmit based on an event time. **Warning:** if the event time is set to 0, the value will revert to 240. Set the event time before setting the value to asynchronous.

TPDO Event Time: The time between an asynchronous TPDO transmission in milliseconds. Minimum time is 10ms.

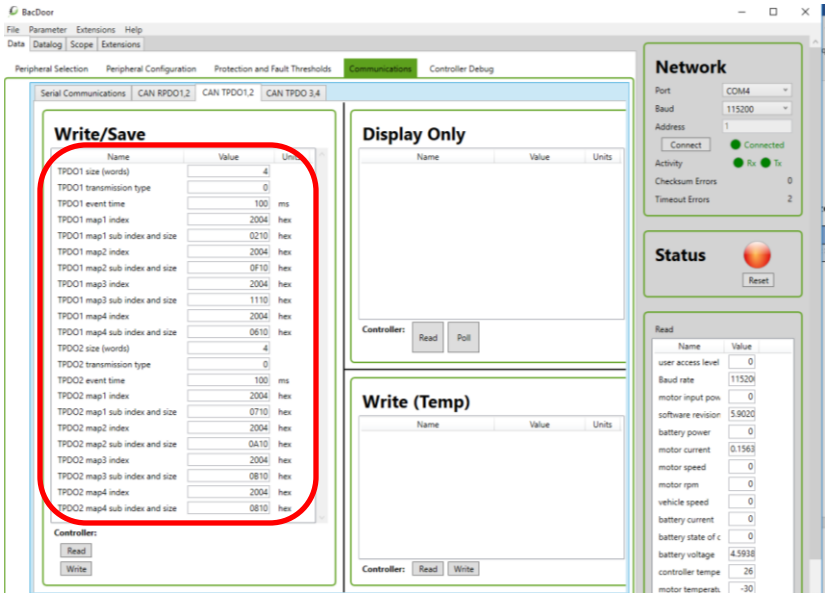
Each TPDO can broadcast up to 4 Dictionary objects.

TPDO Map Index:

Byte 0	Byte 1
Dictionary Index	

TPDO Map Sub Index & Size:

Byte 0	Byte 1
Dictionary Sub Index	Object Size (in bits)



Adjusting Receive PDO (RPDO)

The following are the Dictionary objects used to adjust an RPDO. All objects are two bytes in size.

RPDO Size: Number of objects being received.

RPDO Transmission Type:

Value	Meaning
0…240	Cyclic Synchronous: RPDO is processed on the next SYNC message. Unlike TPDO, the value is irrelevant.
241…254	*(Reserved: not used)*
255 or 256	Asynchronous: The RPDO is processed as soon as the message is received.

RPDO Timeout: The amount of time required before it will receive a PDO again.

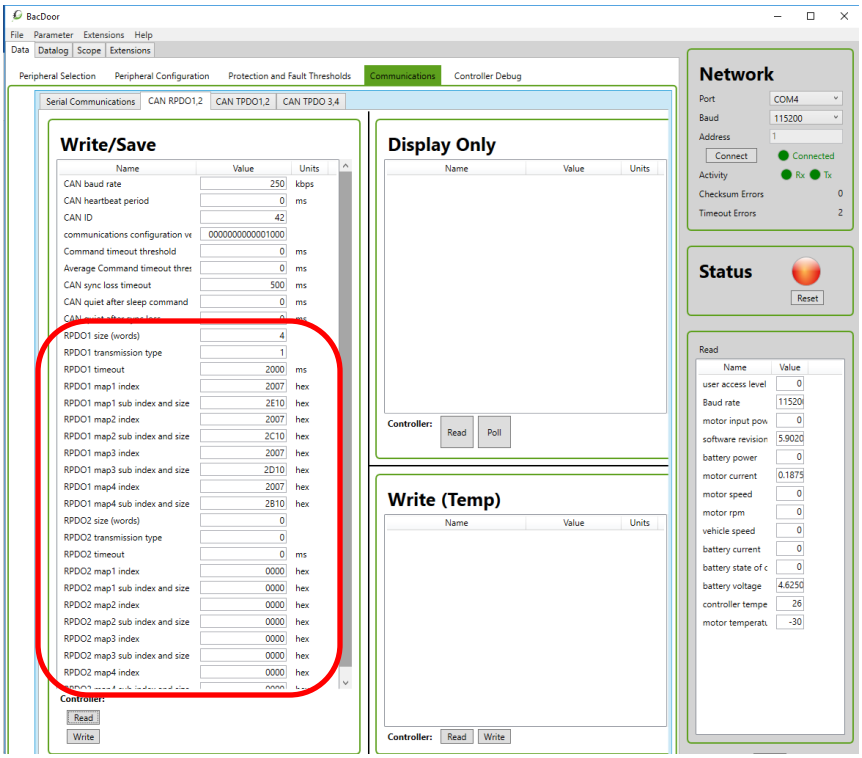
Each RPDO can receive up to 4 Dictionary objects.

RPDO Map Index:

Byte 0	Byte 1
Dictionary Index	

RPDO Map Sub Index & Size:

Byte 0	Byte 1
Dictionary Sub Index	Object Size (in bits)



ASI CAN Object Dictionary

This section has a very brief overview of the object dictionary used for ASI controllers. More detail about the dictionary can be found in the dictionary itself.

CAN Object Dictionary Blocks

The following is a quick reference of the Object Dictionary Blocks used in the controller. For a full description of each block and its sub-indexes, please refer to the ASI Object Dictionary.

Index	Entry
0x1000	Device Type
0x1001	Error Register
0x1005	COB-ID SYNC
0x1007	Synchronous Window Length
0x1009	Manufacturer Hardware Version
0x100A	Manufacturer Software Version
0x1011	Restore Default Parameters
0x1017	Producer Heartbeat Time
0x1018	Identity Object
0x1029	Error behavior
0x1200	Server SDO Parameter 1
0x1201	SDO server parameter 1
0x14XX	RPDO communication parameter
0x16XX	RPDO mapping parameter
0x18XX	TPDO communication parameter
0x1AXX	TPDO mapping parameter
0x1F51	Program Control
0x1F57	Flash status identification
0x2000 – 0X2007	ASI Specific Object Dictionary Entries
0x2009	Commands Control
0x200A	Sleeptimes
0x200B	Custom objects1
0x6000	Supported virtual devices
0x6001	Device control
0x6002	Device status
0x6003	Device capability

0x6006	Device electronic production date
0x6008	Device rated voltage
0x6009	Device alarm status
0x600A	Device alarm capability
0x600B	Device warning status
0x600C	Device warning capability
0x6024	Device maximum continuous input current
0x6025	Device maximum continuous output current
0x6026	Device maximum voltage
0x6027	Device minimum voltage
0x603A	Device peak input current
0x603B	Device peak input current time
0x603C	Device peak output current
0x603D	Device peak output current time
0x603E	Device actual current
0x6040	Device actual voltage
0x6042	Device electronic temperature
0x605A	Device password
0x6200	MCU – Manufacturer Name
0x62D0	MCU Motor controller capability
0x62D8	MCU – Motor limitation capability
0x62E8	MCU – Actual motor values
0x6306	MCU – Set motor assistance level
0x6308	MCU – Set maximum vehicle Speed

Reading ASI Modbus Object Dictionary through CANopen

Many customers are familiar with the ASI Object Dictionary based on the Modbus Protocol. This dictionary was added to the CAN Object Dictionary as well. Use the following formula to calculate the index and the sub-index for the address you want to read/write to. All of these objects are two bytes in length.

Index	**Sub-Index**
FLOOR (address / 64) + 8192	REMAINDER(address / 64) + 1