CANopen protocol

A Controller Area Network (CAN bus) is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based protocol, designed originally for multiplex electrical wiring within automobiles, but is also used in many other contexts.

CANopen is a communication protocol and device profile specification for embedded systems used in automation. In terms of the OSI model, CANopen implements the layers above and including the network layer. The CANopen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation/desegmentation. The lower level protocol implementing the data link and physical layers is usually Controller Area Network (CAN), although devices using some other means of communication (such as Ethernet Powerlink, EtherCAT) can also implement the CANopen device profile.

Use and benefits of CANopen

CANopen protocol allows multiple controllers to be connected into an extensible unified

network. Its flexible configuration capabilities offer easy access to exposed device parameters and real-time automatic (cyclic or event-driven) data transfer.

The benefits of CANopen include:

- Standardized in EN50325-4
- Widely supported and vendor independent
- Highly extensible
- Offers flexible structure (can be used in a wide variety of application areas)
- Suitable for decentralized architectures
- Wide support of CANopen monitoring tools and solutions

Common Configurations

CAN Mode: Used to select one of the 3 operating modes. Off disables all CAN receive and transmit capabilities.

Node ID: CAN Node ID used for transmission from the controller. Value may be between 1 and 126 included.

Bit Rate: Selectable bit rate. Available speeds are 1000, 800, 500, 250, 125, 50, 25, 10 kbit/s. Default is 125kbit and is the recommended speed for RawCAN and MiniCAN modes.

Heartbeat: Period at which a Heartbeat frame is sent by the controller. The frame is CANopen compatible 0x700 + NodeID, with one data byte of value 0x05 (Status: Operational). The Heartbeat is sent in any of the selected modes. It can be disabled by entering a value of 0.

Supported CAN Modes

Three CAN operating modes are available on the CAN-enabled controllers:

1 - RawCAN

2 - MiniCAN

3 - CANopen

RawCAN is a low-level operating mode giving read and write access to CAN frames. It is

recommended for use in low data rate systems that do not obey to any specific standard.

CAN frames are typically built and decoded using the MicroBasic scripting language.

MiniCAN is greatly simplified subset of CANopen, allowing, within limits, the integration of

the controller into an existing CANopen network. This mode requires MicroBasic scripting

to prepare and use the CAN data.

CANopen is the full Standard from CAN in Automation (CIA), based on the DS302 specifcation. It is the mode to use if full compliance with the CANopen standard is a primary requisite.

This section describes the RawCAN and MiniCAN modes, refer to section "CANopen

Interface" for a description of the CANopen mode.

CAN Mode: Used to select one of the 3 operating modes. Off disables all CAN

receive and transmit capabilities.

Node ID: CAN Node ID used for transmission from the controller. Value may be between 1

and 126 included.

Bit Rate: Selectable bit rate. Available speeds are 1000, 800, 500, 250, 125, 50,25, 10 kbit/s. Default is 125kbit and is the recommended speed for RawCAN and MiniCAN modes.

Heartbeat: Period at which a Heartbeat frame is sent by the controller. The frame is CANopen compatible 0x700 + NodeID, with one data byte of value 0x05 (Status: Operational). The Heartbeat is sent in any of the selected modes. It can be disabled by entering a value of 0.

CAN Connection

Connection to a CAN bus is as simple as shown on the diagram above. 120 Ohm Termination Resistors must be inserted at both ends of the bus cable. CAN network can be up to

1000m long. See CAN specifications for maximum length at the various bit rates

CAN Connection

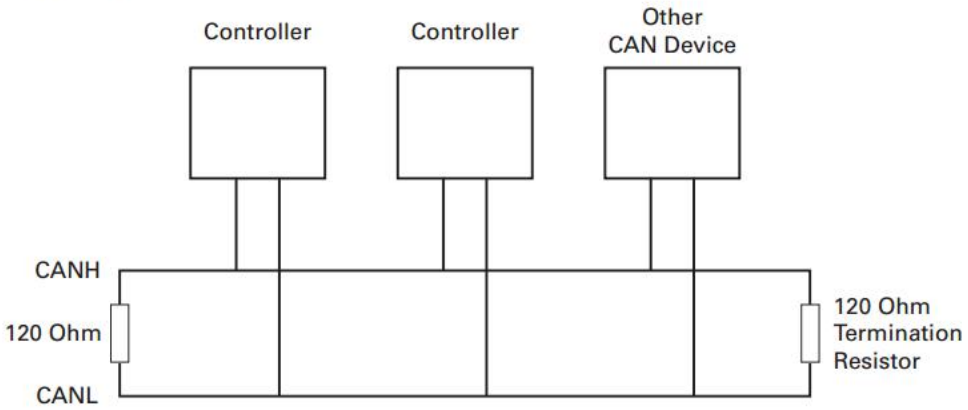

FIGURE 60. CAN connection

CAN Bus Configuration

Use the CAN menu in the Configuration tab in order to enable the CANopen mode. Additionally, the utility can be used to configure the following parameters:

Default configurationL:

- Standard frame
- CAN type: CANopen
- Node ID: 1
- Bit rate:250 kbps
- Heartbeat (ms):1000ms
- Autostart:Enable
- TPDO Enable and Send rate:1500ms

CAN Bus Pinout

Depending on the controller model, the CAN signals are located on the 15-pin female connector or 9-pin male connector. Refer to datasheet for details.

CAN Signals on DB15 connector

Pin Number	Signal	Description
6	CAN_L	CAN bus low
7	CAN_H	CAN bue high

CAN Signals on DB9 connector

Pin Number	Signal	Description
2	CAN_L	CAN bus low
7	CAN_H	CAN bue high

CAN and USB Limitations

On most controller models CAN and USB cannot operate at the same time. On controllers

equipped with a USB connector, if simultaneous connection is not allowed, the controller

will enable CAN if USB is not connected.

The controller will automatically enable USB and disable CAN as soon as the USB is connected to the PC. The CAN connection will then remain disabled until the controller is

restarted with the USB unplugged.

See the controller model datasheet to verify whether simultaneous CAN and USB is supported

The controller operating in the CANopen mode can accept the following types of messages:

- Service Data Objects, or SDO messages to read/write parameter values
- Process Data Objects, or PDO mapped messages to automatically transmit parameters and/or accept commands at runtime
- Network Management, or NMT as defined in the CANopen specification

Service Data Object (SDO) Read/Write Messages

Runtime queries and runtime commands can be sent to the controller in real-time using

the expedited SDO messages.

SDO messages provide generic access to Object Dictionary and can be used for obtaining

parameter values on an irregular basis due to the excessive network traffic that is generated with each SDO request and response message.

Transmit Process Data Object (TPDO) Messages

Transmit PDO (TPDO) messages are one of the two types of PDO messages that are used

during operation.

TPDOs are runtime operating parameters that are sent automatically on a periodic basis

from the controller to one or multiple nodes. TPDOs do not alter object data; they only read

internal controller values and transmit them to the CAN bus.

Receive Process Data Object (RPDO) Messages

RPDOs are configured to capture runtime data destined to the controller

CANopen implementation supports RPDOs. Data received using RPDOs are

stored in 8 user variables from where they can be processed using MicroBasic scripting.

CANopen data transmitting and receiving

Use CANopen to send and receive data , is not a simple specified ID to send commands and parameters, need to meet certain load specification.

Read node Date

The frame

CANopen compatible 0x700 + NodeID,(0x01-0x7e) Range :0x601-0x67E

Data_0	Data_1、2	Data_3	Data_	Data_5	Data_6	Data_7

			4			
0x40	index	sub-index	00	00	00	00

The parameters of the target node returns according to the frame format

Data_0	Data_1、2	Data_3	Data_4	Data_5	Data_6	Data_7
0x4F	index	sub-index	XX	00	00	00
0x4B	index	sub-index	XX	XX	00	00
0x47	index	sub-index	XX	XX	XX	00
0x43	index	sub-index	XX	XX	XX	XX

Read Node parameter

Loading ways of the ID: 0 x600 + target address ID (0 x01-0 x7e), range of 0x601-0x67E

Loading ways of the data frame is as follows:

Data_0	Data_1、2	Data_3	Data_4	Data_5	Data_6	Data_7
0x2F	index	sub-index	XX	00	00	00
0x2B	index	sub-index	XX	XX	00	00
0x27	index	sub-index	XX	XX	XX	00
0x23	index	sub-index	XX	XX	XX	XX

Note: 0 x2F means message sending a byte of data effectively, and so on, 0 x23 means four bytes of data packets sent.

The target node returned by the CAN data frame format

Data_0	Data_1、2	Data_3	Data_4	Data_5	Data_6	Data_7
0x60	index	sub-index	00	00	00	00

Note: This script does not check for loss of communication on the CAN bus. It is provided

for information only

Note:

1. The "XX" said one byte of data effectively."00" said invalid bytes, the data is not standard.

2. The CANopen sending and receiving data fields are eight bytes, extra byte is invalid.

3. Multi-byte index, data are low byte in the front.

4. Send invalid command will return an error frame, knowledge is not in the scope of this agreement is introduced.

(3) command example

1. Speed open control

Explain: Instruction index for0x2000，Speed value range(-1000---1000), to 1000 is forward max speed， to 0 is output is 0， to -1000 is reverse max speed。

If you want to control with ID 1 controller, the speed of the open-loop mode, need to send the message as below

ID :0x601

Data：

Speed value of 500（0x01F4）

0x23	0x00	0x20	0x01	0xf4	0x01	0x00	0x00

Speed value of 1000（0x03E8）

0x23	0x00	0x20	0x01	0Xe8	0x03	0x00	0x00

Speed value of-500（0x0FE0C，即0x01F4的补码+1）

0x23	0x00	0x20	0x01	0x0C	0xFE	0x00	0x00

Speed value of -1000（0x0FC18，即0x03E8的补码+1）

0x23	0x00	0x20	0x01	0x18	0xFC	0x00	0x00

Reply message as below

ID : 0x581

Data ：

0x60	0x00	0x20	0x01	0x00	0x00	0x00	0x00

2、Set position mode target value 4000000（0x3D0900）

0x23	0x01	0x20	0x01	0x00	0x09	0x3D	0x00

3、Set position mode speed 1000（0x03E8）

0x23	0x02	0x20	0x01	0xE8	0x03	0x00	0x00

4、Read position counter

0x40	0x04	0x21	0x01	0x00	0x00	0x00	0x00

5、Read run speed

0x40	0x03	0x21	0x01	0x00	0x00	0x00	0x00

6,stop commend

0x2C	0x0E	0x20	0x01	0x01	0x00	0x00	0x00

When Test, it is recommended to use the CAN card to send packets and monitor the bus status

Object Dictionary

index	sub	Entry Name	Data Type & Access	Command Reference
0x2000	01	Set motor commend,ch 1	S16 WO	“G”
	02	Set motor commend,ch 2		
0x200C	00	Emergency Shutdown	U8 WO	“EX”
0x200D	00	Release Shutdown	U8 WO	“MG”
0x2C0E	01	Set motor commend,ch 1	U8 WO	“MS”
	02	Set motor commend,ch 2		
0x2017	00	Save Config to Flash	U8 WO	“EES”

Instruction index form

index	sub	Entry Name	Data Type & Access	Command Reference
0x2001	01	Set Position commend, ch.1	S32 WO	“P”
	02	Set Position commend, ch.2		
0x2002	01	Set Position speed, ch.1	S16 WO	“S”
	02	Set Position speed, ch.2		
0x2003	01	Set Encoder Counter, ch.1	S32 WO	“C”
	02	Set Encoder Counter, ch.2		
0x2006	01	Set Acceleration , ch.1	S32 WO	“MAC”
	02	Set Acceleration , ch.2		
0x2007	01	Set Deceleration , ch.1	S32 WO	“MDEC”

	02	Set Deceleration , ch.2		

Instruction index form

index	sub	Entry Name	Data Type & Access	Command Reference
0x2100	01	Read Motor Amps, ch.1	S16 RO	“A”
	02	Read Motor Amps, ch.2	S16 RO	
0x2101	01	Read Actual Motor Command, ch.1	S16 RO	“M”
	02	Read Actual Motor Command, ch.2		
0x2103	01	Read Encoder Motor Speed, ch.1	S16 RO	“S”
	02	Read Encoder Motor Speed, ch.2	S16 RO	
0x2107	01	Read Encoder Motor Speed as 1/1000 of Max, ch.1	S16 RO	“SR”
	02	Read Encoder Motor Speed as 1/1000 of Max, ch.2		
0x2108	01	Read Encoder Count Relative, ch.1	S32 RO	“CR”
	02	Read Encoder Count Relative, ch.2		
0x2109	01	Read Brushless Count Relative, ch.1	S32 RO	“CBR”
	02	Read Brushless Count Relative, ch.2		
0x210A	01	Read BL Motor Speed in RPM, ch.1	S16 RO	“BS”
	02	Read BL Motor Speed in RPM, ch.2		
0x210C	01	Read Battery Amps, ch.1	S16 RO	“BA”
	02	Read Battery Amps, ch.2		
0x210D	01	Read Internal Voltages (V Int)	U16 RO	“V”
	02	Read battery Voltages （v）	U16 RO	
	03	Read Internal Voltages (mV)	U16 RO	
0x210E	00	Read All Digital Inputs	U32 RO	“D”
0x210F	01	Read Case & Internal Temperatures (MCU Temperature)	S8 RO	“T”
	02	Read Case & Internal Temperatures (ch.1)		
	03	Read Case & Internal Temperatures (ch.2)		
0x2111	00	Read Status Flags	U8 RO	“FS”
0x2112	00	Read Fault Flags	U8 RO	“FF”
0x2113	00	Read Current Digital Outputs	U8 RO	“DO”
0x2119	00	Read time	U32 RO	“TM”
0x2121	00	Read motor Status	U8 RO	“FM”