AN INTRODUCTION TO CANOPEN AND CANOPEN COMMUNICATION ISSUES

Mohammad Farsi and Karl Ratcliff

Abstract

The Controller Area Network or CAN provides the user with many powerful features including multi-master functionality and the ability to broadcast/multicast telegrams. The low cost of CAN components, high data reliability and short reaction times together with its huge user base will ensure future of CAN is safe for an extremely long time to come. However, to ensure inter-operability between CAN components several higher layer protocols have been developed to allow devices to communicate in a standardised manner. CANopen is one of these protocols and has already seen wide adoption within industry and other sectors such as building automation and public transport. This paper discusses various aspects of CANopen communication and gives a general overview of some of its powerful features.

1. Introduction

The Controller Area Network or CAN has long been used in the automotive industry, textile machinery and has found its way as a fieldbus system within the industrial control sector. However, it has long been a market leader in the industrial fieldbus arena and by the end of 1995 more than 6 million nodes have been installed.

Bus Chips	Cumulative Quantity Sold
P-Net	35,000
FIP	65,000
ASI	80,000
Profibus FMS and DP	500,000
Interbus S	1,000,000
LON	1,500,000
BITBUS	2,500,000
CAN	6,000,000

Figure 1 - Cumulative Bus Nodes Sold Up Until End of 1995. NOTE the figure for CAN does not include automotive applications (figures courtesy of the CiA)

CAN is used in most applications as an internal bus system in embedded networking applications and therefore invisible to the user. Most common uses for CAN are for extension modules in PLCs, drive interfaces and robot controls, internal networks in machines such as those used in food packaging and textiles and medical applications.

As CAN is often used in place of UARTs and the offers the user affordable technology it has acquired significant importance as an open field bus system. CANopen is an open higher layer protocol designed to exploit the full potential of CAN whilst remaining fully open to both integrators and device developers of CANopen products.

Karl Ratcliff and Mohammad Farsi both from the Department of Electrical and Electronic Engineering, Merz Court, The University of Newcastle Upon Tyne.

2. About CANopen and CANopen Profiling

CANopen is a higher layer protocol based on the CAN serial bus system and the application layer CAL (CAN Application Layer).

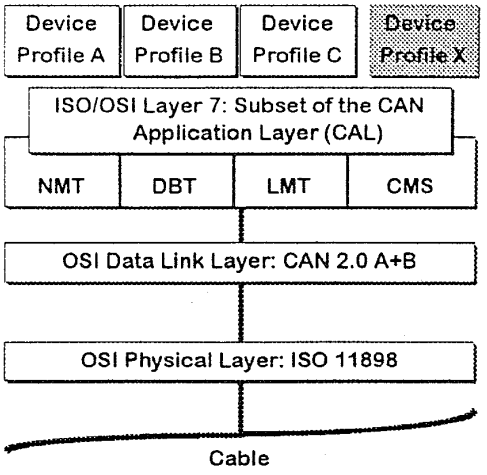


Figure 2 - CANopen ISO/OSI Layer Reference Model

CANopen provides a mechanism whereby devices of different types and makes can be integrated together and communicate in a standardised fashion therefore making different CAN devices interoperable with one another.

At the heart of a CANopen device is the device profile. Using the profiling concept, similar CANopen devices will have guaranteed common operating functions. For example, two digital I/O modules from two different vendors will have common functionality such as setting the outputs and reading the inputs. The profile specifies the functionality which must be common for devices to be interoperable. However, the profiling concept does not constrain manufacturers of devices and allows these manufacturers to implement optional functionality specified within the profile and also implement manufacturer specific operating functions as well.

The fundamental part of the device profile is the object dictionary. This consists of a mixture of data objects, communication objects and commands or actions. CANopen services give the user full access to the object dictionary for a device allowing reading and writing of data and commands. Data and commands are implemented using a 16-bit index addressing mechanism together with 8-bit sub-index giving an address range from 0000H to FFFFH. Parts of the dictionary are divided into different areas depending on the functionality that the parameters in that part of the object dictionary perform. For example, index 6000H is reserved for read a group of 8 inputs on a digital I/O device. Other parameters allow the user to change the communication identifiers and change various device operating parameters.

A typical device object dictionary is shown in Figure 3.

Index	Sub-Index	Type	Name	Data Type	Attribute
1000	00	VAR	device type	Unsigned32	const
1001	00	VAR	error register	Unsigned8	ro
1002	00	VAR	manufacturer status register	Unsigned32	ro
1003		ARRAY	**error register**		
	00	VAR	number of entries	Unsigned8	rw
	01	VAR	error value	Unsigned32	ro
1008	00	VAR	manufacturer device name	String	ro
1009	00	VAR	manufacturer hardware version	String	ro

100A	00	VAR	manufacturer software version	String	ro
1800		**PDOCommPar**	**1st transmit PDO parameter**		
	00	VAR	number of entries	Unsigned8	ro
	01	VAR	receive PDO COB-ID	Unsigned32	rw
	02	VAR	PDO type	Unsigned8	rw
1801		**PDOCommPar**	**2nd transmit PDO parameter**		
	00	VAR	number of entries	Unsigned8	ro
	01	VAR	receive PDO COB-ID	Unsigned32	rw
	02	VAR	PDO type	Unsigned8	rw
1A00		**PDOMapping**	**1st transmit PDO mapping**		
	00	VAR	number of entries	Unsigned8	ro
	01	VAR	1st mapped object	Unsigned32	rw
	02	VAR	2nd mapped object	Unsigned32	rw
1A01		**PDOMapping**	**2nd transmit PDO mapping**		
	00	VAR	number of entries	Unsigned8	ro
	01	VAR	1st mapped object	Unsigned32	rw
	02	VAR	2nd mapped object	Unsigned32	rw
6401		**ARRAY**	**Read_Analogue_Inputs**		
	00	VAR	number of entries	Unsigned16	ro
	01	VAR	analogue_input_1	Unsigned16	ro
	02	VAR	analogue_input_2	Unsigned16	ro
6411		**ARRAY**	**Write_Analogue_Outputs**		
	00	VAR	number of entries	Unsigned16	ro
	01	VAR	analogue_output_1	Unsigned16	rw
	02	VAR	analogue_output_2	Unsigned16	rw

Figure 3 - A Typical Analogue I/O Device Dictionary

Indices between 1000H and 1FFFH are reserved for communication specific parameters such as the identifiers for communication and the type of communication for particular objects or information stored within the dictionary. Indices between 6000H and 9FFFH are reserved for data relating to device specific functionality. In this example index 6401H contains the digital value respresenting the analogue voltage level on one of the devices input lines. Indices at specific locations are reserved for specific functionality described in the device profile. This guarantees that devices with similar characteristics and functionality can be accessed in a common way for example any device implementing digital inputs will always have index 6000H which reads the state of the input lines.

3. Types of Communication In CANopen

CANopen communication can be sub-divided into 4 classes:

1. Service Data Object (SDO) communication. Based on existing CAL CMS domain protocol services mainly used for device parameterisation and configuration.
2. Process Data Object (PDO) communication. Used during normal network operation to transfer data in real-time with little or no protocol processing overhead.
3. Network management functions for coordinating device operation in a controlled manner
4. Predefined format messages such as timing and synchronisation telegrams.

The first three topics are discussed here.

3.1 Service Data Object Communication

This method of data transfer uses the CAL CMS multiplexed domain data transfer mechanisms to access the device object dictionary. Data greater than 8 bytes in length can be transferred using this mechanism by transmission and reception of multiple telegrams.

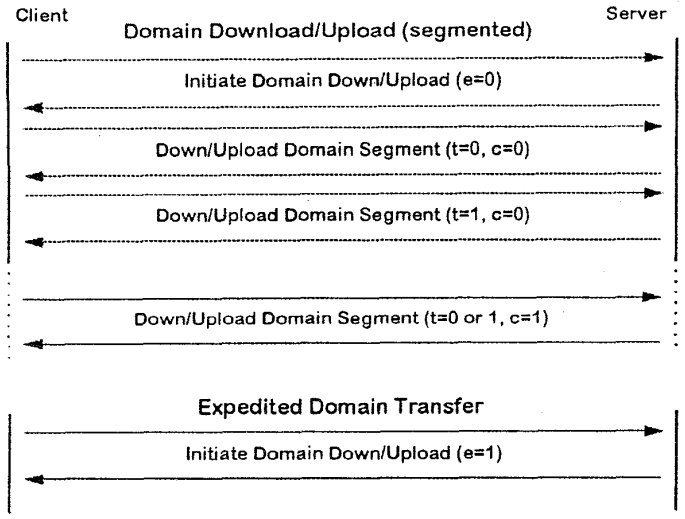


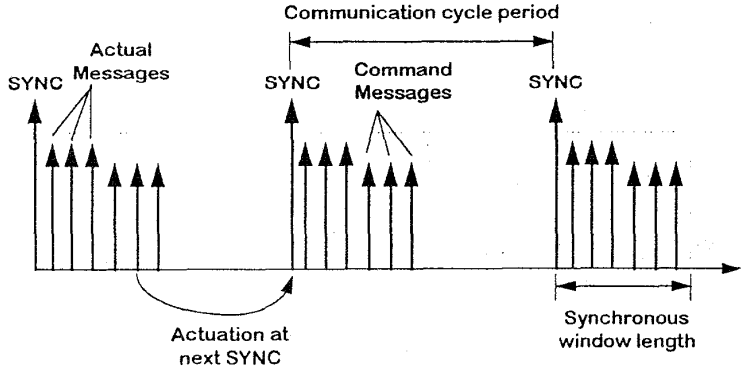
Figure 4 - General Format of SDO Transfers

Using this model, the client is the device reading (domain upload) or writing (domain download) information to and from the server. In the "Initiate Domain Download/Upload" command data is addressed by inclusion of the 16 bit index and 8 bit sub-index in the command. If the data is less than or equal to 4 bytes in length then the SDO transfer requires two telegrams only. An "initiate" command from the client and an initiate response from the server. If the data is greater than four bytes, several telegrams are required to complete the data transfer with the client requesting to read or write each segment of data.

3.2 Process Data Object (PDO) Transfers

CANopen supports a wide variety of PDO communication modes. PDO data is transferred using a single CAN telegram where variables from the device dictionary are mapped into individual bits and bytes of the PDO telegram. This mapping can be predefined or fixed by the device or may be configured dynamically using SDO communication. PDO communication occurs when the network has been made fully operational i.e. after all devices have been powered up and configured (if required).

A powerful feature of PDO communication is the variety of modes which can be used to send PDO information. The first of these is synchronous communication shown in Figure 5.



In this mode a predefined format CANopen telegram known as the SYNC telegram is broadcast to all nodes that require it. On reception of the SYNC nodes transmit PDO data (actual data) on the bus and also receive PDO data (command data). Therefore, for a drive unit the actual messages may represent the current drive position and the command messages represent the new position for the drive to attain. Transmission of the SYNC occurs at preset time intervals and it possible to configure a device to send PDO information on reception of multiple SYNCs e.g. every five cycles.

Event PDO communication is used to transmit operational data on an event basis. For example, Figure 6 shows an analogue input device with a proximity sensor attached. The device is configured in its object dictionary to send an event PDO reflecting the value of the analogue input everytime the the analogue voltage differs by a preset threshold value.

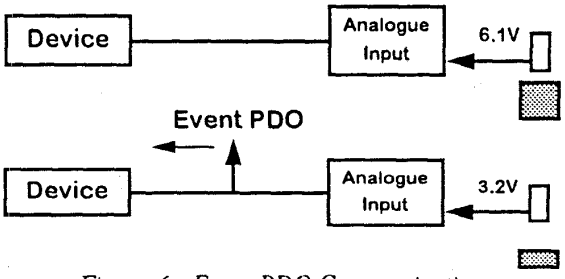


Figure 6 - Event PDO Communication

Finally, it is also possible to request PDO information from a device by means of polling the device. This is acheived using the CAN remote frame mechanism whereby on reception of a remote frame from another device a device transmits a PDO onto the bus.

PDO communication can be configured such that PDOs may be broadcast to several other nodes (consumers of the PDO) and require no intervention of a master node as in some bus systems.

3.3 Network Management

Network management in a CANopen network is based on the NMT and DBT services defined in the CAN Application Layer (CAL). Network management is used to control devices on the network in a coordinated and well defined manner.

When a device powers up on the network and initialises itself it enters an operating state known as PREOPERATIONAL. In this state it is possible to parameterise and configure the device using SDO communication. This includes (if necessary) configuration of PDOs and the identifiers required for PDO communication to take place. At this stage no PDO communication is allowed.

By broadcast of a single 2 byte NMT telegram (start_remote_node) from the network management master all nodes are then made fully OPERATIONAL. In this mode both SDO and PDO communication is allowed. One or all nodes can then be made PREOPERATIONAL again by broadcast of another NMT telegram (enter_preoperational_state), in which case all PDO communication is once again prevented and devices stopped.

A more complex procedure uses the CAL boot up sequence whereby a one to one dialog is established between the network management master and the node. In this mode the node requests identifiers for communication from the distributor (DBT - normally part of the network manager). This includes identifiers for SDO and PDO telegrams and additional CAN identifiers for special messages such as the SYNC. Once again, an NMT start_remote_node can be used to make the node fully OPERATIONAL and the NMT enter_preoperational_state telegram can be used to make the node PREOPERATIONAL.

This mode of operation allows CANopen nodes to be used in existing CAL networks and also gives plug and play capabilities. However, for most networks this mode of operation is not required and simple nodes may not implement this mode as it requires additional memory and code space.

Another feature of CANopen networks is lifeguarding. Lifeguarding enables the network manager to determine whether a node is still alive on the network and also to determine the nodes operating state.

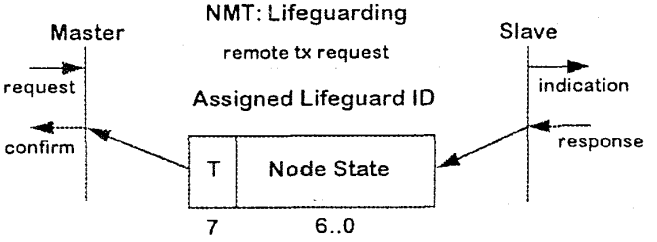


Figure 7 - NMT Lifeguarding

Lifeguarding is periodically performed by the network manager which transmits a CAN remote frame to a device. The device then responds by sending its operating state in a single byte back to the network manager. An error may occur if after a certain period of time the node does not respond with this information or its operating state is incorrect. Optionally, a node may also guard the network manager by timing the interval between successive lifeguard remote frames. If the interval is too great or a lifeguard remote frame is not received within a certain preset time frame the node knows that the network manager is dead.

Other NMT service allow complete node resets (reset_node service) and resetting of the communication parameters (reset_communication service). On power on, nodes are addressed using a node ID which is preset either using DIP switches or stored in non-volatile memory on the device. When the device powers up it enters its PREOPERATIONAL state automatically and uses its predefined connection set in order to perform SDO communication. The reset_communication service is used to restore all the device communication parameters to their default power-on values.

4. Emergency Telegrams

Emergency telegrams may be thought of as very high priority PDO telegrams that occur on an event basis when an error occurs on the device. The structure of this telegram and emergency error codes are defined within the CANopen communication profile and also individual device profiles but also allow for manufacturer specific information to be incorporated.

5. Concluding Remarks

CANopen provides an exciting open protocol which exploits all the potential of the Controller Area Network (CAN). CANopen allows direct data exchange between nodes on the network without participation of a bus master unit. It allows full broadcast/multicast features and a variety of communication modes designed to help keep bus loading minimal and predictable. Therefore, CANopen is well suited to the concept of remote intelligence and is ideal for distributed control solutions.

6. References

[1] CiA DS-301 - CANopen communication profile obtainable from the CAN in Automation (CiA) Group
[2] CAL Communication Standards - CiA DS-201 to DS 207 obtainable from the CiA (see appendix)
[3] A low cost configuration tool for CANopen networks - K Ratcliff and M Farsi, CAN Newsletter March 1997.