System Integration Based on CANopen Protocol for Distributed Input/output Automation Systems

[1] MR. R. M. METRA, [2] PROF. M. N. KAKATKAR , [3] MS. D. R. HERBHA, [4]MR. B. M. DAXINI

[1]M.E. [Embedded & VLSI] Student, Department of Electronics & Communication Engineering, Sinhgad College of Engineering Vadgaon (BK), PUNE, Maharashtra
[2] Professor Department of Electronics & Communication Engineering, Sinhgad College of Engineering, Vadgaon (BK), PUNE, Maharashtra
[3] Asst. Professor Department of Electronics & Communication Engineering, G.K.Bharad Institute of Engineering, Tramba, Rajkot, Gujarat
[4]M.TECH [Control Systems] Student, Department Instrumentation & Control Engineering, Manipal Institute of Technology, Manipal, Karnataka

metra.rincash@gmail.com, dharmi.herbha@gmail.com, bhautik.daxini@gmail.com

***ABSTRACT*:** CANopen is a truly open protocol that has not been developed by one company alone, now under the supervision of the CAN in Automation organization. This paper gives an overview of some of the fundamental concepts of CANopen communication and of CANopen implementation. It allows the use of any CANopen compliant object dictionaries to communicate with the device. Any object can be read or written, for both service data and process data communications .Synchronization messages can be generated automatically, with real time display and control of synchronous data. Full CANopen network management services and life guarding are implemented and there is an object dictionary editing facility. Canopen protocol which is recently new protocol adopted by CiA-CAN in Automation, For the creation of inexpensive de centralized Automation systems, distributed input/output systems and networked sensor/actuator systems. By this we get configuration flexibility, resource sharing like sensors, reduce cabling cost and failure. Reduce system down time.

Keywords— CAN, OD, PDO, SDO, CiA, NMT, DBT

I: INTRODUCTION

CAN is a serial bus system especially suited for networking "intelligent" devices as well as sensors and actuators within a system or sub-system. CAN_H & CAN_L are the two cables form basis for communication. Basically in a CAN network each station is assigned a unique Identifier and each node had right to broadcast a message. Depending upon the priority fixed by Identifier each node is allocated the bus time. When data is transmitted by CAN no stations are addressed. The identifier defines not only the content but also the priority of the message. This is important for bus allocation when several stations are competing for bus access.

II: CANOPEN COMMUNICATION MODEL

CANopen is a networking system based on the CAN serial bus. However, a high level protocol is necessary in order to define how the CAN message frame's 11-bit identifier and 8 data bytes are used. Building CAN based industrial automation systems guaranteeing interoperability and interchange ability of devices of different manufacturers requires a standardized application layer and 'profiles', standardizing the communication system, the device functionality and the system administration[1]

A. Application Layer & Communication Profile

The Application Layer comprise a concept to configure and communicate real time data. E.g. a network configuration on a PC configures the network and maps inputs and outputs as desired.[3][4]. The application layer provides a set of services and protocols useful to every device on the network imaginable. The communication profile provides the means to configure devices and communication data and defines how data is communicated between devices. Device profiles add device-specific behaviour for devices (e.g. digital I/O, analog I/O, motion controllers, encoders, etc.)[3].

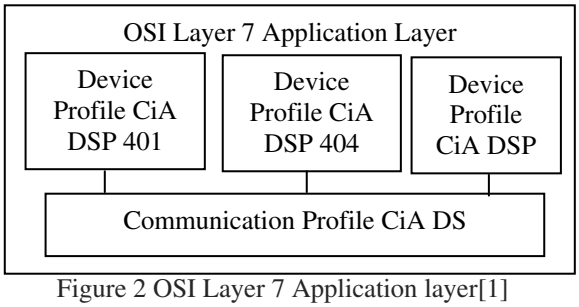

Figure 2 OSI Layer 7 Application layer[1]

B. Data Link Layer

DATA Link Layer of CANopen Protocol is based on the CAN data link layer and the high-speed CAN transceiver. The scope of this layer mainly is the transfer protocol, i.e. controlling the framing, performing arbitration, error checking, error signalling and fault confinement. Within this layer it is decided whether the bus is free for starting a new transmission or whether a reception is just starting[3][4].

C. Physical Layer

The scope of the physical layer is the actual transfer of the bits between the different nodes with respect to all electrical properties. The physical medium is a two-wire bus line being terminated at both ends by resistors[2]. Figure 1 shows CANopen specifies bit timing and recommends pin-assignments for some connectors. The maximum length of a basic CANopen network can go up to 1 km @ baud rate of 10 Kbits/s[5].

II: *DATA-FLOW MODEL*

Since CAN is a broadcast system, a transmitting node places data on the network for all nodes to access. only those nodes requiring updated data allow the message to pass through a filter that is set by the network designer i.e., messages from certain nodes can pass, and all others are ignored. If this filter is not used by a system designer, much of a node's mC processing time is spent sorting through messages that are not needed.

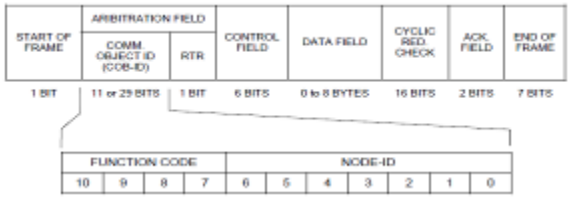

Figure 3 CAN Message (DATA) Frame Structure [2]

In a CAN network the lower the value of the COB-ID, the higher the priority of the corresponding message on the network is. Note that this standard assumes CAN2.0A CAN Standard Message Frame having an Figure 3 shows 11-bit identifier, allowing a range of [0, 2047], but which for historical reasons- is limited to [0, 2031]. However using CAN2.0B CAN Extended Message Frame with 29-bit identifier doesn't change the picture [1][3][4].

III: *CANOPEN OBJECT DICTIONARY*

The CANopen Object Dictionary (OD) is an ordered grouping of objects each object is addressed using a 16-bit index. To allow individual elements of structures of data to be accessed an 8-bit sub index has been defined as well. The general layout of the CANopen OD lies between 1000 and 9FFF.For every node in the network there exists an OD[1][3]. The OD contains all parameters describing the device and its network behaviour

There are the various device profiles defining for a particular type of devices the OD objects there are now about different device profiles and several more are under development [1]. A profile describes for each OD object its function, its name, its index and sub-index, its data type, whether the object is mandatory or optional, whether the object is 'read only' or 'write only' or 'read/write', etc.[1][3]

Table 1 CANopen Object Dictionary [1][3][4]

CANopen Object Dictionary	
Index	**Object**
0000	not used
0001 - 001F	Static Data Types (standard data types, e.g. Boolean, Integer16)
0020 - 003F	Complex Data Types (predefined structures composed of standard data types,e.g.PDOCommPar,SDOParameter)
0040 - 005F	Manufacturer Specific Complex Data Types
0060 - 007F	Device Profile Specific Static Data Types
0080 - 009F	Device Profile Specific Complex Data Types
00A0 -0FFF	reserved
1000 - 1FFF	Communication Profile Area(e.g. Device Type, Error Register, Number of PDOs supported)
2000 - 5FFF	Manufacturer Specific Profile Area
6000 - 9FFF	Standardised Device Profile Area (e.g. "DSP-401 Device Profile for I/0 Modules" [3]: Read State 8 Input Lines, etc.)
A000 - FFFF	reserved

If more features are required than are present in the profile, there is plenty of space in the profile available for the addition of manufacturer specific functionality. So the part of the OD describing the communication parameters is the same for all CANopen devices i.e. object placing in the OD is the same, not necessarily the value of the object, the device specific part of the OD is different for different devices[2]

IV: CANOPEN COMMUNICATION

Now that we have presented the concept of the Object Dictionary we now will look at the messages that are communicated in CANopen networks, their content and their function, in other words: the CANopen communication model. The CANopen communication model defines four types of messages.

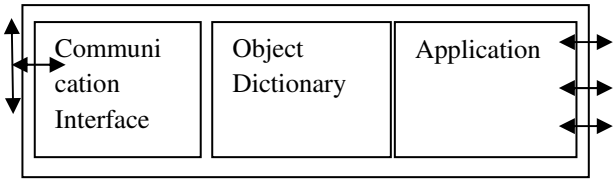

Figure 4 CANopen Device [4]

A. Administrative message

Layer management network management and identifier distribution services i.e. initialisation, configuration and supervision of the network. Services and protocols are according to the LMT, NMT and DBT service elements of CAL[1][3]. These services are all based on a Master-Slave concept in a CAN-network there is only one LMT, NMT or DBT master and one or more slaves.[4].

B. *CMS (CAN-based Message Specification)*

It offers objects of type Variable, Event and Domain to design and specify how the functionality of a device (a node) can be accessed through its CAN interface (e.g. how to up- or download a set of data ('domain') exceeding the 8 bytes maximum data content of a CAN-message, including an 'abort transfer' feature). CMS derives from MMS (Manufacturing Message Specification), which is an OSI application layer protocol designed for the remote control and monitoring of industrial devices. CMS defines 8 priority levels in its messages, each having 220 COB-IDs, occupying COB-IDs 1 to 1760; the remaining identifiers (0, 1761-2031) are reserved for NMT, DBT and LMT. In a CAN-network the lower the value of the COB-ID, the higher the priority of the corresponding message on the network is[1].

C. Service Data Object (SDO)

An SDO provides a client access to entries of a device OD the device is the server using the object's OD index and sub-index, contained in the first few bytes of the CAN message.

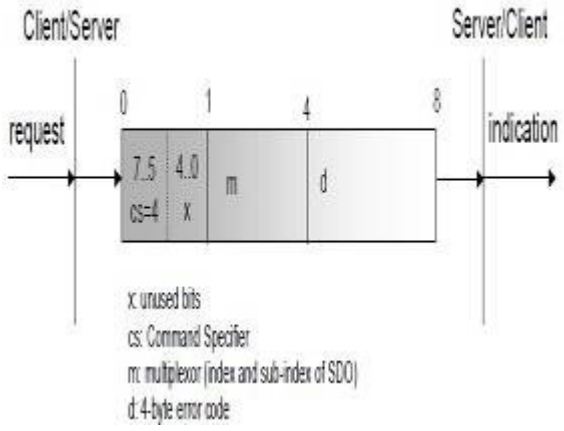

Figure 4 SDO Transfer [4]

An SDO is implemented as a CMS object of type Multiplexed Domain according to CAL, thus permitting transfer of data of any length as data is split up over several CAN messages if necessary, which is when data occupies more than 4 bytes. The protocol is of the confirmed service type a reply is generated for every CAN message one SDO requires 2 CAN identifiers. The SDO request and reply message always contain 8 bytes the number of non-significant bytes is shown as part of the first byte, which carries protocol information, thus communication via an SDO has a considerable overhead.

D. Process Data Object (PDO)

PDO is used to transfer real-time data is transferred from one and only one producer to one or more consumers. Data transfer is limited to 1 to 8 bytes for example one PDO can transfer at maximum 64 digital I/O values, or 416 bit analog inputs. It has no protocol overhead. The data content of a PDO is defined through its CAN identifier only and this content is assumed known to sender as well as receiver(s) of the PDO. Each PDO is described by 2 objects in the Object Dictionary [5].Contents of the PDO message is predefined (or is configured at network start-up

V: *PREDEFINED MESSAGES OR SPECIAL FUNCTION OBJECTS*

A. SYNCHRONIZATION (SYNC)

Used to synchronize tasks network-wide particularly relevant for drive applications actual values of inputs are stored quasi-simultaneously network-wide and subsequently transmitted if required, output values are updated according to messages received after the previous SYNC. Implemented as a CMS object of type Basic Variable CANopen suggests a COB-ID in the highest priority group to ensure a regular synchronization signal to keep the message as short as possible no data bytes are transferred.

A. Time Stamp

Time Stamp provides application devices a common time frame reference. Implemented as a CMS object of type Stored Event.[1][3][4]

B. Emergency

Emergency is triggered by the occurrence of a device internal error. Implemented as a CMS object of type Stored Event.[5]

C. Node/Life Guarding

The NMT master monitors the state of the nodes these are called node guarding. Nodes optionally monitor the state of the NMT master this is called life guarding, it starts on the NMT slave after it has received the first Node Guard message from the NMT master. Implemented according to the NMT node guarding protocol a Remote Transmission Request from the NMT master to a particular node triggers a reply containing the node's state[5].

D. Boot-up

Boot up is Master-slave concept. sending this message the NMT slave indicates to the NMT master that is has transitioned from state initialising to state Preoperational or other.

VI: TRANSMISSION MODES

A. ***Acyclic - synchronous***: the PDO will be transmitted synchronously but not periodically.

B. ***Cyclic – synchronous***: the PDO will be transmitted synchronously, whereby the Number of Sync's gives the number of synchronization messages, which lie between two transmissions of this PDO.

C. ***Synchronous – RTR only***: the PDO will be updated after each synchronization message but not sent. It is only sent when there is an explicit request to do so (Remote Transmission Request)

D. ***Asynchronous – RTR only***: the PDO will only be updated and transmitted when there is an explicit request to do so (Remote Transmission Request)

E. ***Asynchronous*** – device profile specific and asynchronous - manufacturer specific: the PDO will only be transmitted when specific events occur.

VII: *EXPERIMENTAL STUDY OF IMPLEMENTING EXAMPLE*

System integration based on CANopen, Indican 5231Master CPU CAN base CPU, Messung I/O Module c6417 and c6418 as a CAN base node, As input Push button, sensors are Reed switch, Proximity, Optical sensor, As output solenoid coil, or actuator, indication lamp etc.. can use. When all the input and output are deside then by decided I/O list we can design the circuit by using circuit design software like ePLAN and Other. Here system integration based on CANopen for distributed Input/output automation Master CPU and Nodes are shown below figure

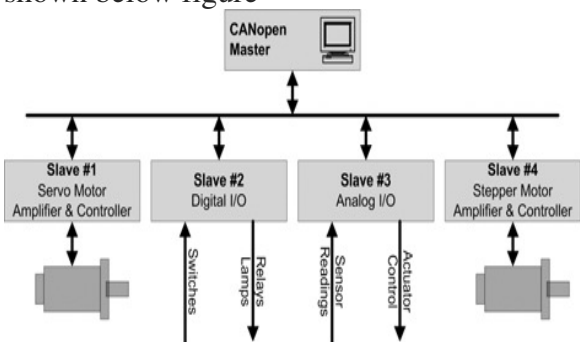

Figure 5 Master Slave consept[6]

In this example we use digital i/o.indiCAN master and nodes C6417 and C6418 Nodes configuration and programming Codesys IDE tool (software) require.

A. Addressing I/O

we give address to all inputs and outputs. For Push button(PB) input we use %IX0.0 to %IX0.1 for one input module. For the output address we use lamp indication %QX0.0 to %QX0.1. and variable declaration.

```
VAR_INPUT
PB1 AT %IX0.0 : BOOL(*CYCLE START PB*)
PB1 AT %IX0.1 : BOOL(*CYCLE STOP PB*)
PR1 AT %IX0.2 : BOOL(*SENSOR 1*)
RS1 AT %IX0.3 : BOOL(*SENSOR 2*)
END_VAR
VAR_OUTPUT
CSIL AT %QX0.0 : BOOL(*CYCLE START IL*)
CSTPIL AT %QX0.1 : BOOL(*CYCLE STOP IL*)
SOL1 AT %QX0.2 : BOOL(*SOLENOID 1*)
SOL2 AT %QX0.3 : BOOL(*SOLENOID 2*)
END_VAR
```

B. Node Id Number

CANopen is transfer data in Master – Slave concept so always Node Id: 0 for Master CPU and respective Node Id: 1 and Node Id : 2 for slaves.

C. Base Parameters

Node:1,is connected to Master it transfer the data between Node: 0 and Node:1 each Node given a unique module.

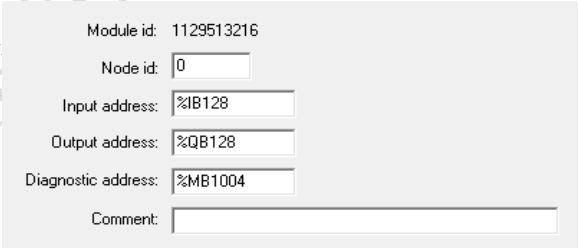

Figure 6 Base Parameter [7]

Module id is a unique identifier of the module with in the entire configuration,. Node number : 0 for CAN Master, Input address %IB128, Output address %QB128, Diagnostic address %MB128 are addresses for Input and Output respectively for the storage of diagnosis (error) data.

D. CAN Parameter

Master is connected to Node Id :1,so all parameter are related to transferring the data between node and master CANopen have baud rate of ~120 feet at a 1 MBit/sec baud rate to 5000M at 10Kbit/sec. As above discuss above COB-ID define the priority level 0-NMT start/stop services and 1-220 CMS priority level 0 [1].

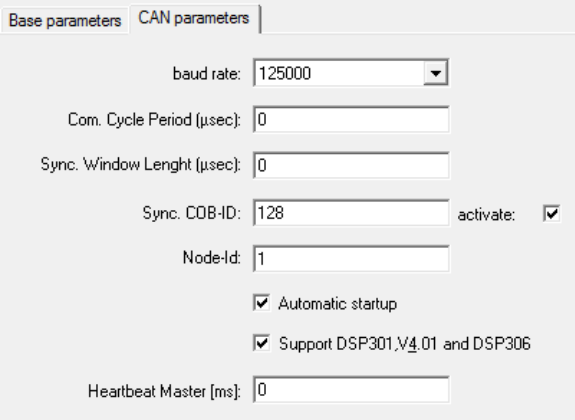

Figure 6 Master CAN Parameter [7]

The synchronization message is sent with a unique number Sync. COB-Id in the interval in microseconds which is given by the Communication Cycle Period. The synchronous PDO's are transmitted directly after the synchronization message in the time window called Sync. Window Length in microseconds. No synchronization message will be sent if the fields Comm. Cycle Period and Sync. Window Length contains 0. activate means synchronization messages will be transmitted between master and slaves. If the option Support DSP301,V3.01 and DSP306 is activated, then modular CAN Slaves as well as some additional extensions concerning the standards DSP301 V3.01 and DSP306 will be supported. In this case e.g. the stroke of the Heartbeat will be adjustable (Heartbeat Master).Heartbeats is an alternative guarding mechanism, In contrast to the Node guarding functionality it can be executed by Master and Slave Modules. Usually the master will be configured to send heartbeats to the slaves.

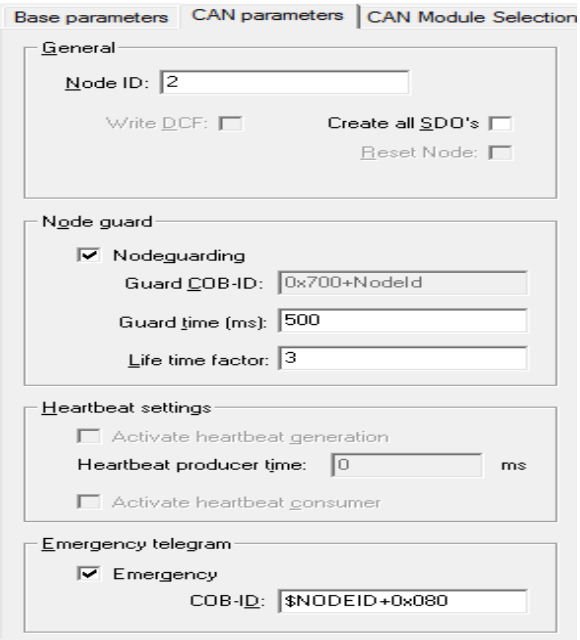

Figure 6 Node CAN Parameter [7]

If the option Node guarding is activated, a message will be sent to the module according to the interval set by Guard Time in milliseconds. If the module does not then send a message with the given Guard COB-ID , it will receive the status "timeout". As soon as the number of attempts has been reached, the module will receive the status "not OK". In are case Master monitor the Node at every 500ms to check the devise and sensors are ok or not we take the Life Time Factor 3 it means it is 3 time of Guard Time. The status of the module will be stored at the diagnosis address. No monitoring of the module will occur if the variables Guard Time and Life Time Factor are not defined (0). Section Emergency Telegram a module sends an emergency message, with a unique COB-Id., when there is an internal error. These messages, which vary from module to module, are stored in the diagnosis address

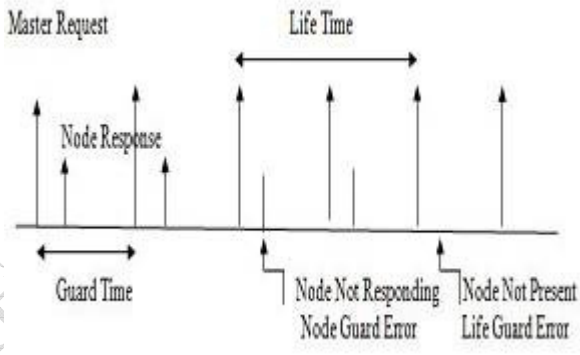

Figure 6 Node Guarding [1]

E. PDO Parameters

Each PDO message requires a unique COB-Id (Communication Object Identifier). The Inhibit Time is the minimum time between two messages from this PDO. In this we use 100µs. This is to prevent PDOs which are sent when the value is changed from being sent too often. Transmission Type offers you a selection of possible transmission modes, Here we use asynchronous–device profile specific transmission mode because PDO only transfer when the Push Button is push .we have also define the event time. It is a time between two transfer. In are case we define 100ms.

F. SDO Transfer

SDO transfer the 8byte frame to access the object dictionary,0-7 byte SDO, Byte 1-3 Multiplexor having 16 bits index and 8 bits sub index, Byte 4-7 contain Data.

G. Programming

Different programming language Ladder Diagram LD, State Transition Diagram STD, Sequential Function Chart SFC, Structured Text ST, Function Block Diagram FBD, Instruction List IL are use for Distribution Input/output automation systems.

VIII: *CAN APPLICATIONS*

CAN networks can be used as an embedded communication system for micro controllers as well as an open communication system for intelligent devices. The major advantages are Low cost, the ability to function in a difficult electrical environment, a high degree of real-time capability and ease of use.
Main application areas are:
1. Automobiles
2. Textile machinery industry
3. Packaging machinery
4. Machinery for paper manufacture and processing.
5. Machine tools for networking sensors and actuators within the line or machine.
6. Medical engineering sector, decided in favours of CAN because they had particularly Stringent safety requirements.
7. Robotic Automation Systems.

IX: ***CONCLUSION***

For the creation of inexpensive de-centralized control systems, distributed input/output systems and networked sensor/actuator systems. It allows the use of any CANopen compliant object dictionaries to communicate with the device. Any object can be read or written, for both service data and process data communications. Synchronization messages can be generated automatically, with real time display and control of synchronous data. Full CANopen network management services and life guarding are implemented and there is an object dictionary editing facilityCANopen have configuration flexibility, We can configure different types of I/O with CANopen. It has resource sharing like we can shares input of the different sensors, Due to sharing of resources, we can reduce cabling cost and failure. We get easy trouble shooting as well as easy to maintain with respect to ordinary PLC. Easy to configure and modify, cost effective to design and implement, it has easy transfer of data from sensors to controller. By using the CANopen it can reduce system down time.

REFERENCES

[1] CANopen high-level protocol for CAN-bus *H. Boterenbrood NIKHEF*, Amsterdam March 20, 2000
[2] Toward a safe design of CANopen distributed instruments, June 2006 Author(s): Benoit,E. Univ. de Savoie, Annecy
[3] Nexgen CPU indiCAN Master Document No.:ED-2002-071 Version: 1.0 Published December 2003
[4] CiA:CAN in Automation organization CAN-CiA.org
[5] Design of fieldbus communication protocol control gateway May 2012
Author(s): Wenqi Zeng Dept. of Electron. Inf., Teachers' Coll. of Beijing Union Univ., Beijing, China
[6] CANopen-Higher layer Protocol based Controller Area Network (CAN) support device profile for I/O Modules, Motion Control, Wilfred Voss esd electronics.inc Hatfield,
[7] IDE Tool for implement CoDeSys V2.3 Software