CANopen
Higher Layer Protocol based on Controller Area Network (CAN)

Presented by Wilfred Voss
esd electronics, Inc.
525 Bernardston Road
Greenfield, MA 01301

http://www.esd-electronics.us
What is CANopen?

- Higher Layer Protocol based on Controller Area Network (CAN)
- All CAN features are available: Simplicity, high reliability, extremely short reaction/error recovery times
- Master/Slave configuration, Multi-Master & direct communication between Slaves supported
- Was originally designed for Motion Control
- “Open” system, non-proprietary
- Supports device profiles for Digital I/O, Analog I/O, Motion Controllers, Sensors, Actuator, etc.
- All CANopen devices “speak the same language”
- Ultimate goal: Manufacturer-independence, devices are interchangeable
Higher Layer Protocols

Why Higher Layer Protocols...

- Data Transport of more than 8 bytes
- Embedded Systems require appropriate communication model based on Master/Slave configuration
- Network Management (Network Start-Up, Node Monitoring, Node Synchronization, etc.)
Higher Layer Protocols

**CANopen**
- Suited for embedded applications
- Was originally designed for motion control
- Developed/Maintained by CAN-in-Automation User Group
- Manufacturer-Independent Protocol
  
http://www.can-cia.org

**DeviceNet**
- Suited for industrial applications (floor automation)
- Developed by Allen Bradley/Rockwell
- Maintained by Open DeviceNet Association (ODVA)
- Standard “controlled” by Allen Bradley/Rockwell

http://www.odva.org

**SAE J 1939**
- Communication for vehicle networks (trucks, buses, etc.)
- Standard developed by Society of Automotive Engineers (SAE)

http://www.sae.org
<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>Start of Bosch internal project to develop in-vehicle network</td>
</tr>
<tr>
<td>1986</td>
<td>Official introduction of the CAN protocol</td>
</tr>
<tr>
<td>1987</td>
<td>First CAN controller chips by Intel &amp; Philips</td>
</tr>
<tr>
<td>1991</td>
<td>Bosch publishes CAN specification 2.0</td>
</tr>
<tr>
<td>1992</td>
<td>CAN in Automation (CiA) established</td>
</tr>
<tr>
<td>1992</td>
<td>CAN Application Layer (CAL) protocol by CiA</td>
</tr>
<tr>
<td>1992</td>
<td>First automobiles equipped with CAN (Mercedes Benz)</td>
</tr>
<tr>
<td>1993</td>
<td>ISO 11898 standard published</td>
</tr>
<tr>
<td>1994</td>
<td>First International CAN Conference (iCC)</td>
</tr>
<tr>
<td>1994</td>
<td>Allen Bradley introduces DeviceNet</td>
</tr>
<tr>
<td>1995</td>
<td>ISO 11898 amendment (extended frame format)</td>
</tr>
<tr>
<td>1995</td>
<td>CANopen protocol introduced</td>
</tr>
</tbody>
</table>
CANopen Applications

- Semiconductor Industry (Wafer Handlers, etc.)
- Robotics, Motion Control Applications
- Passenger/Cargo Trains (Brake Control, Wagon Communication)
- Aircrafts (AC, Seat Adjustment)
- Elevators (e.g. Otis)
- Building Technologies (Light & Door Control Systems, Sensors, etc.)
- Medical Equipment (X-Ray, CAT scanners, etc.)
- Household Utilities (Coffee Machine, Washer, etc.)
- Aerospace (Satellites)
CiA – CAN in Automation

- International Users and Manufacturers Organization
- Develops, supports CAN Standards and CAN based higher layer protocols
- All activities are based on CiA members’ interest
- North American office in Novi, MI

http://www.can-cia.org

CAN Newsletter
To subscribe log on to:
http://www.can-cia.org/newsletter/
Benefits of Using CANopen

Main Benefit:
- Physical and Data Link Layer implemented in Silicon
- SW Development Engineer is not involved with writing protocol features

- Low Cost Implementation
- Very Reliable, Error-Resistant
- Worldwide Acceptance
- Last, but not least...CANopen Saves You Money!

Be aware! Whenever you attempt to add software functions between the CAN Data Link Layer and the Application Layer, you will be adding functionalities that are already covered by off-the-shelf available higher layer protocols such as CANopen and DeviceNet.
Further CANopen Characteristics

- Standardized protocol services provided, available as source code off-the-shelf

- Protocol supports node IDs in addition to CAN message IDs

- Supports up to 127 nodes in a network, each node requires a unique ID

- Very low resources/memory print
  CANopen Slave ~10...20k ROM, <1k RAM
  CANopen Master ~20...30k ROM, ~1k RAM
CANopen Reference Model

Application Layer

Device Profiles
- I/O Modules - DSP 401
- Drives & Motion Control - DSP 402
- IEC 1131 Interface - DSP 405
- and more...

CANopen Application Layer and Communication Profile - DS301
Framework for Programmable CANopen Devices - DSP-302

Data Link Layer

Physical Layer
# Device Profiles

<table>
<thead>
<tr>
<th>Profile</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS-401</td>
<td>Generic I/O modules</td>
</tr>
<tr>
<td>DS-402</td>
<td>Drives and motion control</td>
</tr>
<tr>
<td>DS-403</td>
<td>Not allocated</td>
</tr>
<tr>
<td>DS-404</td>
<td>Measuring devices and closed loop controllers</td>
</tr>
<tr>
<td>DS-405</td>
<td>IEC 61131-3 programmable devices</td>
</tr>
<tr>
<td>DS-406</td>
<td>Encoders</td>
</tr>
<tr>
<td>DS-407</td>
<td>Public transportation - Passenger Information Systems</td>
</tr>
<tr>
<td>DS-408</td>
<td>Fluid Power Technology - Hydraulic drives and proportional valves</td>
</tr>
<tr>
<td>DS-409</td>
<td>Vehicle door control</td>
</tr>
<tr>
<td>DS-410</td>
<td>Declinometers</td>
</tr>
<tr>
<td>DS-412</td>
<td>Medical Devices</td>
</tr>
<tr>
<td>DS-413</td>
<td>Truck Gateways</td>
</tr>
<tr>
<td>DS-414</td>
<td>Weaving Machines</td>
</tr>
<tr>
<td>DS-415</td>
<td>Road Construction Machinery</td>
</tr>
<tr>
<td>DS-416</td>
<td>Building Door Control</td>
</tr>
<tr>
<td>DS-417</td>
<td>Lift Control Systems</td>
</tr>
<tr>
<td>DS-418</td>
<td>Battery Modules</td>
</tr>
<tr>
<td>DS-419</td>
<td>Battery Chargers</td>
</tr>
<tr>
<td>DS-420</td>
<td>Extruder Downstream Devices</td>
</tr>
</tbody>
</table>

**More info:**
[http://www.can-cia.de/services/cialiterature/]
Device Model

Application Layer
Device Functionality

Object Dictionary
Data Types
Communication Objects
Application Objects
Proprietary Objects

Communication Interface
Server/Client SDOs
Rx/Tx PDOs

Network Management
NMT, SYNC, Emergency, Time Stamp

Data Link Layer

Physical Layer
Object Dictionary

Application Layer

Device Functionality

Object Dictionary

Data Types
Communication Objects
Application Objects
Proprietary Objects

Communication Interface

Server/Client SDOs
Rx/Tx PDOs

Network Management

NMT, SYNC, Emergency, Time Stamp

Data Link Layer

Physical Layer
<table>
<thead>
<tr>
<th>Index</th>
<th>Object</th>
<th>Common to any Device</th>
<th>Device Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000h</td>
<td>Not Used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001h – 025Fh</td>
<td>Data Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0260h – 0FFFh</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000h – 1FFFh</td>
<td>Communication Profile Area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000h – 5FFFh</td>
<td>Manufacturer Specific Profile Area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6000h – 9FFFh</td>
<td>Standardized Device Profile Area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A000h – AFFFh</td>
<td>Network Variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B000h – FFFFh</td>
<td>Reserved for Future Use</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Object Dictionary Addressing

0000h  16 Bit index

2100h

8 Bit sub-index

0  Number of Entries
1  1st Object Entry
2  2nd Object Entry

254  254th Object Entry
255  Reference to Data Structure

FFFH
### Object Dictionary Sample Entry

<table>
<thead>
<tr>
<th>Index</th>
<th>Sub-Index</th>
<th>Variable</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2092h</td>
<td>0h</td>
<td>Number of Entries</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>2092h</td>
<td>1h</td>
<td>Baud Rate</td>
<td>Unsigned16</td>
</tr>
<tr>
<td>2092h</td>
<td>2h</td>
<td>Number of Data Bits</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>2092h</td>
<td>3h</td>
<td>Number of Stop Bits</td>
<td>Unsigned8</td>
</tr>
<tr>
<td>2092h</td>
<td>4h</td>
<td>Parity</td>
<td>Unsigned8</td>
</tr>
</tbody>
</table>

#### C Structure Equivalent

```c
typedef struct {
    UNSIGNED8   NumberOfEntries;
    UNSIGNED16  BaudRate;
    UNSIGNED8   NumberOfDataBits;
    UNSIGNED8   NumberOfStopBits;
    UNSIGNED8   Parity;
} RS232;
```
Communication Interface

SDO – Service Data Object
- Supports transfer of data of any length (Configuration data, program download, etc.)
- Confirmed communication, each request results in a response

PDO – Process Data Object
- Provides “backward compatibility” to CAN
- Transfer of max. 8 data bytes without protocol overhead
- Used for real-time transmission of process data
**SDO – Service Data Object**

- Used for point-to-point communication between two nodes acting as SDO Client and SDO Server
- Provides Read/Write access to all entries in Object Dictionary of a node
- Read/Write Data is identified by Index & Sub-Index
- Supports transfer of any length of data
- Confirmed communication, each request results in a response
- 2 CAN identifiers per SDO
- Mainly used for device configuration and download/upload of large data blocks
- Each CANopen device must support at least the Default-Server-SDO which always provides basic access to the device
**SDO – Transfer Modes**

- **Expedited Transfer**
  - Fast SDO transfer mode for data up to 4 bytes

- **Non-Expedited Transfer (Segmented Transfer)**
  - SDO transfer mode for any length of data
  - Flow control after 7 transmitted bytes

- **Blocktransfer**
  - SDO transfer mode for any length of data
  - Fast transfer method with flow control after block transfer of 1...127 bytes
PDO – Process Data Object

- Used for real-time transmission of process data
- Provides very efficient transmission of data according to Consumer-Producer model
- Transfer of max. 8 bytes without protocol overhead
- Definition of transmitted data described per “PDO Mapping”
- Unconfirmed transfer (correct reception of data is handled by CAN protocol)
- 1 CAN identifier per PDO
- Number of Transmit/Receive PDOs of a device according to application needs
PDO – Transmission Modes

- **Event Driven**
  - PDO transmitted upon occurrence of an event, e.g. change of input
  - An optional event timeout can be defined to trigger the transmission after a given time elapsed without any event

- **Polling (Remote Request)**
  - PDO is transmitted only upon request from a remote device (per remote frame)

- **Synchronized**
  - PDO is only transmitted upon reception of a SYNC message
PDO Mapping

• Description of process data to be transmitted/received within a PDO is provided by “PDO Mapping”

• Specifies how the data is “mapped” in a message

• Process data to be transmitted/received within a PDO is specified in a “Mapping Parameter List” in form of a reference (Index, Sub-Index) in the Object Dictionary

• Each PDO has its own “Mapping Parameter List” which can also be accessed through the Object Dictionary
PDO Mapping - Example

Object Dictionary

<table>
<thead>
<tr>
<th>Index</th>
<th>Sub-Index</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000h</td>
<td>00h</td>
<td>N/A</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

PDO Mapping Parameter List

<table>
<thead>
<tr>
<th>Index</th>
<th>Sub-Index</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1B00h</td>
<td>00h</td>
<td>04h</td>
<td>Number of Entries</td>
</tr>
<tr>
<td>01h</td>
<td>6000 01 08h</td>
<td>Idx 6000 - Sidx 1 - 8 Bit</td>
<td></td>
</tr>
<tr>
<td>02h</td>
<td>6000 02 08h</td>
<td>Idx 6000 - Sidx 2 - 8 Bit</td>
<td></td>
</tr>
<tr>
<td>03h</td>
<td>6A00 01 10h</td>
<td>Idx 6A00 - Sidx 1 - 16 Bit</td>
<td></td>
</tr>
<tr>
<td>04h</td>
<td>6A00 02 10h</td>
<td>Idx 6A00 - Sidx 2 - 16 Bit</td>
<td></td>
</tr>
</tbody>
</table>

Process Data

<table>
<thead>
<tr>
<th>Index</th>
<th>Sub-Index</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000h</td>
<td>01h</td>
<td>xx</td>
<td>8 Bit Digital Input</td>
</tr>
<tr>
<td>02h</td>
<td>xx</td>
<td></td>
<td>8 Bit Digital Input</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Sub-Index</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6A00h</td>
<td>01h</td>
<td>xxxx</td>
<td>16 Bit Analog Input</td>
</tr>
<tr>
<td>02h</td>
<td>xxxx</td>
<td></td>
<td>16 Bit Analog Input</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TxPDO n</th>
<th>D IN 1</th>
<th>D IN 2</th>
<th>A IN 1</th>
<th>A IN 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 4</td>
<td>Byte 5</td>
</tr>
</tbody>
</table>

© esd electronics, Inc. • 525 Bernardston Road • Greenfield, MA 01301
System Services

- **Notification of Device Errors**
  - Emergency functionality to signal failures of application or communication

- **System-wide Synchronization of Processes**
  - Simultaneous execution of processes

- **System-wide time reference**
  - Common time base throughout the network
Network Management

Application Layer
Device Functionality

Object Dictionary
Data Types
Communication Objects
Application Objects
Proprietary Objects

Communication Interface
Server/Client SDOs
Rx/Tx PDOs

Network Management
NMT, SYNC, Emergency, Time Stamp

Data Link Layer

Physical Layer
Network Management

• Network Management is based on Master/Slave relationship

• Tasks of a CANopen Master/Manager
  • Controlling the network boot-up process
  • Verification and supervision of system consistency
  • Download of configuration data to new devices
  • Controlling the communication status of a device

• Device Monitoring
  • Node-Guarding (Master/Slave Monitoring)
  • Heartbeat (Device status is transmitted as broadcast information, each device can monitor other devices)
Device Communication State

Power On

Initialization

Pre-Operational Status (Configuration Status)

Operational Status

Stopped
Further CANopen Network Services

• **Configuration Manager**

  Enables automatic download of configuration data to new and unconfigured CANopen devices, e.g. in case of exchange of faulty device. Allows Plug&Play.

• **SDO Manager**

  Manages dynamic establishment of SDO connections between devices during run-time.

• **Layer Setting Services**

  Additional services for configuration of node ID and baud rate via CAN, no DIP Switches required.
Further CANopen Network Services

- **Flying Manager**
  Allows more than one CANopen Master/Manager, e.g. for backup-purposes. Negotiation of active Master performed automatically.

- **Redundancy (Fault Tolerant CANopen)**
  Uses two independent CAN lines for communication.
Implementation of CANopen

• CANopen protocol stack available off-the-shelf, e.g. Phytec
  http://www.phytec.com

• Free CANopen Source Code

http://www.microcanopen.com
CANopen Development Tools

What you need is...

• The “regular” tools such as cross-compiler, emulator, etc.
• CANopen Hardware Interface (Starter Kit)

• CANopen API for your own programming
• CANopen “Analyzer” Software
• CANopen EDS Editor
• CANopen Configuration Tool
What you need to consider...

- Make sure the CANopen API is multi-tasking/multi-threading capable
- Make sure that CANopen API is available for your RTOS

- Don’t pay more than $1,000 for combination of PC CAN Interface Board and PC Analyzer Software.
A Comprehensible Guide To Controller Area Network

By Wilfred Voss

ISBN 0-9765116-0-6

Includes:

• A Brief History of CAN
• Main Characteristics
• Message Frame Architecture
• Message Broadcasting
• Bus Arbitration
• Error Detection and Fault Confinement
• CAN Physical Layer

For more information and for ordering this book log on to: