CAN Primer

Updated Summer 2010    Robert Boys  bob.boys@arm.com

ARM

KEIL
An ARM® Company

Cortex
Intelligent Processors by ARM®

Version 2.0
Agenda:

- What exactly is CAN?
- Identifiers – what they are.
- What does the data look like?
- What the CAN wires look like.
- Example CAN circuits.
- Arbitration – who gets priority.
- Errors and how to detect them!
- Some Tips and Hints....
Myths about CAN:

• Is not only for automotive applications!
• Can be found everywhere.
• Automation is one (CANopen, DeviceNet)
• Is easy to implement...no mystery anymore
• ...unless you use some tricky protocols.
• You can “roll your own” for simplicity.
• Can gateway to USB, Ethernet, RS232 and other networks to form larger systems.
Why Use CAN?

- Electrically robust with built-in error and arbitration features. These are automatic!
- Differential pair reduces EMI in/out.
- Many controllers and parts available.
- Plenty of software and hardware tools.
- Really, really easy to add another node.
- Hard work is done by the CAN controller.
- Many chips have implemented CAN.
- Largest CAN network in world is believed to be the Vancouver Olympic sign.
What exactly is CAN?

- CAN – Controller Area Network

*The Bosch CAN spec includes:*
- Some of the physical layer (wires).
- Message frame description.
- Attributes – Peer-to-Peer network.
- Arbitration scheme - multiple messages.
- Some error detection and handling.
The CAN spec does **not** include:

- Any high level addressing modes.
- Any message descriptions or groupings.
- Diagnostics or messages streams.
- Any acknowledgement a message reached its intended target.
- Bus Speed: see ISO 11898-2, J2284.
- The physical link – you can use anything.
- People often include these items. Shouldn’t!
CAN Speeds

- Is specified from ~10 Kbps to 1Mbps.
- Can’t change speed dynamically…ever!
- 125, 250 & 500 Kbps common.
- Longer cable runs means slower frequency.
- 1 Mbps a bit hard to manage in a real system.
High Points of the CAN Bus

- Uses a Differential twisted pair of wires.
- The highest priority message gets thru.
- Uses non-destructive arbitration.
- The priority of message is its identifier.
- 0 has the highest priority. Always.
- No Master or Slave – Peer to Peer.
- All nodes see all messages on network.
- …except their own…..
Construction of a CAN Network

The main CAN backbone and drops are comprised of a twisted pair of wires.

a,b = 120 Ω ½ watt termination resistors.

Drops use the same twisted pair of wires as the backbone.
Example CAN Node

Typical microcontroller

- FLASH Memory
- Microcontroller
- CAN Controller Chip
- Registers
- Buffers
- CAN Transceiver Chip
- Twisted Pair
- CAN_Hi
- CAN_Lo
A real CAN circuit:

- R12, R13 & C45 create “Split Termination” for extra noise immunity instead of one 120 ohm.
- Note: A Common Mode Choke can be used.
- TIP: Use USB cable to power Keil boards.
The difference between these three is the CAN transceiver and normally the speed.
Differential Twisted Pair

- CAN transceiver converts single-ended Tx to a differential pair output.
- ALSO – takes differential pair and converts back to Rx.
- Node sees in and out at same time.
- Interference coming in & out cancels out.
- Twisted pair and receiver ccts does this.
Recessive and Dominant Bits

- Recessive is 2.5 volts CAN Hi and Lo. Difference is $2.5 - 2.5 = 0$ volts. Call this a “1”
- Dominant is CAN_Hi 3.5v CAN_Lo 1.5v. Difference is $3.5 - 1.5 = 2.0$ volts. Call this a “0”

The important voltage is the difference between CAN Hi and Lo and not to ground.
Recessive and Dominant Bits

- Any node can pull Hi and Lo apart on the bus.
- No node can force Recessive.
- This means an idle bus has zero volts Hi & Lo and about 2.5 volts to ground.
- This is how arbitration works.
- A bus can sit idle for a hundred years and every node will see the first message sent.
- The clocks in CAN need to be good ones.
The Differential Signals

Bottom signals are algebraically subtracted to result in the top signal:
A problem!!
Older system works OK!
Bit Stuffing

- Nodes need an active clock to stay in sync.
- Must come from the bus transitions.
- If no change for 5 bit times – will add a bit.
- CAN frame is actually lengthened by this.
- CAN controllers add & remove stuffed bits.
- Invisible to the programmer.
- Only an oscilloscope will show this...and here is where a problem can surface!
Bit stuffing makes the frame look like it has jitter!

It does not….but it can be confusing.
A CAN Frame Decoded:

- recessive = 1
- dominant = 0

SOF, RTR, IDE, RO

ID = 000 00010000 = $10
0 00
DLC = 1

DATA = 01010101
= $ 55
CRC = 1010100011111111
= $ 547f

= stuffed bit

DEL = 1
EOF = 7
IFS = 3
Data Link: the frame.

- **CAN** is a serial bus. 1 or 2 wires. (SW-CAN)
- **CAN 2.0 A** Standard
  - 11 bit identifiers – 2,048 ID’s. 0 – 7FF
- **CAN 2.0 B** Extended
  - 29 bit identifiers – 536 million ID’s.
    - 0 – 1FFF FFFF
- Both 11 and 29 bit can be used on the bus.
- CAN controllers can easily sort them out.
A little more…

- 0 to 8 bytes of data per CAN frame.
- This can be changed dynamically.
- Bus length to 40 meters @ 1 Mbps, 1 meter drops. Slower = longer.
- Two 120 termination resistors needed at each end of the bus. (measure 60 ohms)
### Standard and Extended CAN Frames

<table>
<thead>
<tr>
<th>Arbitration Field</th>
<th>Control Field</th>
<th>Data Field</th>
<th>CRC Field</th>
<th>ACK Field</th>
<th>End of Frame</th>
<th>IFS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SOF</strong> 11-bit Identifier</td>
<td>RTR</td>
<td>IDE</td>
<td>DLC (4)</td>
<td>0-8 Bytes</td>
<td>5-bit CRC</td>
<td>DEL</td>
</tr>
<tr>
<td>11-bit Identifier</td>
<td>SRR</td>
<td>IDE</td>
<td>18-bit Identifier</td>
<td>RTR</td>
<td>R1</td>
<td>R0</td>
</tr>
</tbody>
</table>

**Designers need only fill in the RED lines!**

**Extended Frame:** SRR=1, IDE=1

- **SOF** - Start of Frame
- **SRR** - Substitute Remote Request
- **IDE** - Identifier Extension
- **RTR** - Remote Transmission Request
- **R0** - Reserved bit
- **R1** - Reserved bit
- **DLC** - Data Length Code
- **CRC** - Cyclic Redundancy Check
- **DEL** - Delimiter
- **ACK** - Acknowledgement bit
CAN Programming Model

- **IDE 1 bit**
  - IDE = 0: 11 bit ID
  - IDE = 1: 29 bit ID

- **DLC 4 bit**
  - How many data bytes are there?
  - DLC = 0 thru 8
  - Data bytes: 0, 1, 2, 3, 4, 5, 6, 7 or 8 bytes (0 to 64 bits)
Acknowledge: Newbie mistake # 1.

- The ACK bit.
- Sender sets this Recessive.
- Someone has to assert this to dominant.
- Else: sender re-transmits again forever.
- So – need at least one other CAN node.
- Is merely 2 µsec wide @ 500 Kbps.
A bug turned into a feature…..

- If bus gets trashed after sender finishes…
- But before others think is over…
- Bus fault occurs and message retransmitted.
- **SO:** don’t increment or toggle a value !
- If you want a variable to be 64 – say it in full.
Priority Levels

- Message with highest priority gets thru.
- Lowest Identifier has priority.
- 000 beats 001, 123 beats 256. Always.
- Arbitration evaluated in real-time.
- Uses Recessive and Dominant bits.
- A node can (and must!) see itself and others on the bus in real time.
- Note: a node can’t see its own ID and data.
Arbitration Notes:

- Illegal to have same identifier at same time.
- Possible a node will never get priority.
- CAN is not deterministic. TTCAN is though.
- Note: if 11 and 29 bit same identifier at the same time – 11 bit wins arbitration.
- i.e. 11010101010 & (wins because IDE = 0) 11010101010101100101001101000
What can you put into the ID?

- Anything at all! CAN does not specify.
- IDs will get on bus according to priority.
- Your System Designer will not agree!
- **Normally**: addresses of modules or devices.
- Or tasks….Request or Response
- Acknowledge (don’t confuse with ACK)
- ID values are carefully selected for filtering.
What can put into Data?

- Anything at all! CAN does not specify.
- Will not be prioritized by CAN controller.
- Your System Designer will not agree!
- Normally: Services, Modes, data (signals).
- Data transfers etc.
- Multi-byte data control bytes. i.e. # bytes sent.
- Number of data bytes can be changed.
Diagnostics and Standard Traffic

- **Standard Mode:** ordinary data from ordinary vehicle operation.
  - Nodes are operating normally.

- **Diagnostic Mode:** Nodes are put into a special mode for queries by a scantool.
  - Is problem or inquiry related.

- Diagnostics handle “Limp Home Mode”
- Not CAN spec – higher layers
Polled and Periodic Messages

- **Periodic Messages**: messages broadcast continually on the bus at certain rate.
- **Polled Messages**: messages provided due to a query by a node or diagnostic tool.
- Are created by high level – not CAN.
- Such as J1939 and proprietary systems.
Real CAN traffic on a 500 Kbps car.

<table>
<thead>
<tr>
<th>Time</th>
<th>ID</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>000:000</td>
<td>OB2</td>
<td>00 48 00 48 00 00</td>
</tr>
<tr>
<td>004:510</td>
<td>2D2</td>
<td>00</td>
</tr>
<tr>
<td>008:240</td>
<td>025</td>
<td>00 A2 00 00 00 00 00 CF</td>
</tr>
<tr>
<td>010:240</td>
<td>OB0</td>
<td>00 48 00 48 00 00</td>
</tr>
<tr>
<td>011:810</td>
<td>2C4</td>
<td>00 00 00 20 00 80 21 8F</td>
</tr>
<tr>
<td>012:260</td>
<td>OB2</td>
<td>00 48 00 48 00 00</td>
</tr>
<tr>
<td>014:360</td>
<td>223</td>
<td>00 00 00 00 00 00 00 2D</td>
</tr>
<tr>
<td>016:420</td>
<td>224</td>
<td>00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>020:510</td>
<td>025</td>
<td>00 A2 00 00 00 00 00 CF</td>
</tr>
<tr>
<td>021:030</td>
<td>2C1</td>
<td>08 05 83 28 06 EC 00 75</td>
</tr>
<tr>
<td>022:500</td>
<td>OB0</td>
<td>00 48 00 48 00 00</td>
</tr>
<tr>
<td>024:540</td>
<td>OB2</td>
<td>00 48 00 48 00 00</td>
</tr>
<tr>
<td>029:220</td>
<td>2D0</td>
<td>00 00 08 00 10 00 00 F2</td>
</tr>
<tr>
<td>032:770</td>
<td>025</td>
<td>00 A2 00 00 00 00 00 CF</td>
</tr>
<tr>
<td>034:770</td>
<td>OB0</td>
<td>00 48 00 48 00 00</td>
</tr>
<tr>
<td>035:370</td>
<td>2C4</td>
<td>00 00 00 20 00 80 21 8F</td>
</tr>
<tr>
<td>036:260</td>
<td>2D2</td>
<td>00</td>
</tr>
<tr>
<td>036:840</td>
<td>OB2</td>
<td>00 48 00 48 00 00</td>
</tr>
</tbody>
</table>
**Real J1939 Traffic**

<table>
<thead>
<tr>
<th>Line No</th>
<th>TimeStamp...</th>
<th>Channel</th>
<th>Frame Id</th>
<th>Header Details</th>
<th>Frame Acronym</th>
<th>Protocol</th>
<th>Data</th>
<th>Rx/Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>3171</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3172</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3173</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3174</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00400</td>
<td>P/T = 3 : PGN = 00F004 : Src Addr = 00</td>
<td>EEC1_00F004</td>
<td>J1939</td>
<td>FE 7D 7D 00 00 00 FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3175</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00400</td>
<td>P/T = 3 : PGN = 00F004 : Src Addr = 00</td>
<td>EEC1_00F004</td>
<td>J1939</td>
<td>FE 7D 7D 00 00 00 FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3176</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3177</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3178</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>18FEF000</td>
<td>P/T = 6 : PGN = 00FEF0 : Src Addr = 00</td>
<td>PTO_00FEF0</td>
<td>J1939</td>
<td>FF FF FF 00 00 00 CC FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3179</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>18FEF000</td>
<td>P/T = 6 : PGN = 00FEF0 : Src Addr = 00</td>
<td>PTO_00FEF0</td>
<td>J1939</td>
<td>FF FF FF 00 00 00 CC FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3180</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>18F0000F</td>
<td>P/T = 6 : PGN = 00F000 : Src Addr = 0F</td>
<td>18F0000F</td>
<td>J1939</td>
<td>C0 7D FF 00 00 FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3181</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>18F0000F</td>
<td>P/T = 6 : PGN = 00F000 : Src Addr = 0F</td>
<td>18F0000F</td>
<td>J1939</td>
<td>C0 7D FF 00 00 FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3182</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3183</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3184</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00400</td>
<td>P/T = 3 : PGN = 00F004 : Src Addr = 00</td>
<td>EEC1_00F004</td>
<td>J1939</td>
<td>FE 7D 7D 00 00 00 FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3185</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00400</td>
<td>P/T = 3 : PGN = 00F004 : Src Addr = 00</td>
<td>EEC1_00F004</td>
<td>J1939</td>
<td>FE 7D 7D 00 00 00 FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3186</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00300</td>
<td>P/T = 3 : PGN = 00F003 : Src Addr = 00</td>
<td>EEC2_00F003</td>
<td>J1939</td>
<td>F1 00 00 FF FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3187</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00300</td>
<td>P/T = 3 : PGN = 00F003 : Src Addr = 00</td>
<td>EEC2_00F003</td>
<td>J1939</td>
<td>F1 00 00 FF FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3188</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3189</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3190</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>18FEF100</td>
<td>P/T = 6 : PGN = 00FEF1 : Src Addr = 00</td>
<td>CCVS_00FEF1</td>
<td>J1939</td>
<td>FF 00 00 00 60 00 00 C0</td>
<td>Rx</td>
</tr>
<tr>
<td>3191</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>18FEF100</td>
<td>P/T = 6 : PGN = 00FEF1 : Src Addr = 00</td>
<td>CCVS_00FEF1</td>
<td>J1939</td>
<td>FF 00 00 00 60 00 00 C0</td>
<td>Rx</td>
</tr>
<tr>
<td>3192</td>
<td>00:07:28...</td>
<td>CH#A</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
<tr>
<td>3193</td>
<td>00:07:28...</td>
<td>CH#B</td>
<td>0CF00203</td>
<td>P/T = 3 : PGN = 00F002 : Src Addr = 03</td>
<td>OCF00203</td>
<td>J1939</td>
<td>CC 00 00 FF 00 FF FF FF FF</td>
<td>Rx</td>
</tr>
</tbody>
</table>
CAN Summary

- This gives a basic understanding of CAN.
- From here you can go to:
  High level protocols connect to CAN
  J1939, CANopen, MilCAN, J2284, …

Now: a bit about tools….and then errors and some handy hints…
Tools

- Design, development and maintenance.
- Hardware and software.
- Cheap, average and high cost.
- Vastly different capabilities.
- Designed to save you time and money.
- Low cost tools are alluring but....
Cheap Tools

- Costs you time and money and worse….
- Adverse to critical Time-To-Market.
- Awkward, hard to use, hard to setup.
- Limited info provided – no trigger or filters.
- Some interesting tests might not happen.
- Time is better to use for testing product.
Network Tools –
Bus Analyzers and CAN Oscilloscopes

- Development of network and diagnostics
- Sits on a bus and monitor.
- Can send and receive messages on bus.
- Save messages, trigger, filter, respond.
CAN Faults

- Regular dual-wire CAN redundancy:
  - One CAN open or shorted to ground.
    - Note: ground must be connected for open.
  - But not to each other or both to ground.
  - Ground can be open.
  - or a ground loop can exist.
Error Frames

- Any node see something wrong on the bus.
- Makes bus dominant.
- All nodes know this is an error frame.
- Sender stops transmitting.
- Increments its Transmit Error Counter by 8.
- If TEC < FF, resends message else busoff.
- Others increment their Receive Error Counters.
- Note: only a node can boot itself off the bus.
Error Counters

- Two 8 bit counters in CAN controller.
- Counts the errors.
- Inc. & decrement.
- Up by 8, down by 1
- Can read as an indication of bus health.
- Busoff return: RESET or 128 good messages: its what you say.

Note: 96 is considered a significant error level.
CAN Bit Timing

- Each bit consists of a # of time quantum (tq).
- TQs added and subtracted as clock syncs.
- Tqs are set by designer in CAN controller chip.
- See your datasheet for help.
CAN Filters

- Allow only selected messages through!
- On ID and sometimes 1\textsuperscript{st} byte of data.
- This is why IDs are carefully chosen.

Typical microcontroller

![Diagram of microcontroller components]
CAN Statistics

- Periodic Auto bus rates about 6 – 7 %
- GM periodic about 15 % @ 500 Kbps.
- Getting higher and higher.
- Design in some slack for expansion.
Tips & Hints

- 2 nodes: never send the same message.
  - Get a bus error or both think successful.
- Use same # data bytes all the time.
  - Software is much simpler to debug and maintain.
- Never change the bus speed!
- Beware of short bursts of high bus traffic.
- Read TEC & REC to view state of your bus
- If use a protocol – implement all of it!
- Through data on the floor but take care of it.
more Tips & Hints

- Don’t mix one protocol with another
  - J1939 and CANopen will crash…maybe…
- Use defaults – don’t do anything “elegant”
  - You don’t want to be the first to find that bug!
- Use timeouts – don’t hang until RESET.
- Don’t make timeouts too tight – sloppy is better.
- Select IDs with care for easy filtering.
- Don’t fudge the bus. Fix it.
And a few more....

- Design in some “elbow room”.
- Reserve some IDs for expansion/bug fixes.
- Take care of “Reserved Bits”.
- Remember protocols are designed by committees. Investigate odd things.
- Watch out for the simple things.....
The End…for now…

- See [www.keil.com/can](http://www.keil.com/can)
  …and look for my CAN Primer.
- Versions for Luminary, NXP and ST.
- ISO 11898-1,-2-3-4 defines CAN further.
- Testing CAN Physical layers:
  [www.dgtech.com/pdfs/techpapers/CIA_article.pdf](http://www.dgtech.com/pdfs/techpapers/CIA_article.pdf)