Controller Area Network

Marco Di Natale
Scuola Superiore S. Anna- Pisa, Italy

Adapted for EECS 124 by Sanjit A. Seshia, UC Berkeley

CAN bus

CAN = Controller Area Network
- Publicly available communications standard [1]
  http://www.semiconductors.bosch.de/pdf/can2spec.pdf

Serial data bus developed by Bosch in the 80s
- Support for broadcast and multicast comm
- Low cost
- Deterministic resolution of the contention
- Priority-based arbitration
- Automotive standard but used also in automation, factory control, avionics and medical equipment
- Simple, 2 differential (copper) wire connection
- Speed of up to 1Mb/s
- Error detection and signalling
Purpose of this Lesson

- Introduction to a widely-used communication protocol standard in the automotive industry
- Develop time analysis for real-time messages
- Understand how firmware can affect the time determinism and spoil the priority assignment

A CAN-based system

- TX buffers (TXobjects) typically 1 to 32
- RX buffers (RXobjects) typically 1 to 32
- Firmware (MAC layer implementation)
CAN bus

CAN standard (MAC protocol)
- Fixed format messages with limited size
- CAN communication does not require node (or system) addresses (configuration information)
  - Flexibility – a node can be added at any time
  - Message delivery and routing – the content is identified by an IDENTIFIER field defining the message content
  - Multicast – all messages are received by all nodes that can filter messages based on their IDs
  - Data Consistency – A message is accepted by all nodes or by no node

CAN bus

Frame types
DATA FRAME
- Carries regular data
REMOTE FRAME
- Used to request the transmission of a DATA FRAME with the same ID
ERROR FRAME
- Transmitted by any unit detecting a bus error
OVERLOAD FRAME
- Used to force a time interval in between frame transmissions
CAN bus

DATA FRAME

Start of frame – 1 dominant bit. A frame can only start when the bus is IDLE. All stations synchronize to the leading edge of the SOF bit.

Identifier – 11 (or 29 in version 2.0) bits. In order from most significant to least significant. The 7 most significant bits cannot be all recessive (all 1s).

RTR – remote transmission request, dominant for REQUEST frames, recessive for DATA frames.

CONTROL – (see figure) maximum data length is 8 (bytes) other values are not used.
CAN bus

DATA FRAME (continued)

*Data* – 0 to 8 bytes of data
*CRC* – 15 CRC bits plus one CRC delimiter bit (recessive)
*ACK* – two bits (SLOT + DELIMITER) all stations receiving the message correctly (CRC check) set the SLOT to dominant (the transmitter transmits a recessive). The DELIMITER is recessive

*END OF FRAME* – seven recessive bits

Bit stuffing

any sequence of 5 bits of the same type requires the addition of an opposite type bit by the TRANSMITTER (and removal from the receiver)

CAN bus

Arbitration

All nodes are synchronized on the SOF bit

The bus behaves as a wired-AND

An example …

\[
\begin{align*}
\text{Id} = 0x15a &:& 00101011010 & 00101011010 \\
\text{Id} = 0x3d2 &:& 01111010010 & 01111010010 \\
\text{Id} = 0x1f6 &:& 00111110110 & 00111110110
\end{align*}
\]

<table>
<thead>
<tr>
<th>sof</th>
<th>0 0 0 1 0 1 0</th>
<th>0 0 0 1 0 1 0</th>
<th>0 0 0 1 0 1 0</th>
<th>0 0 0 1 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
CAN bus

A sender must wait longer than that maximum propagation latency before sending the next bit.

Why?

The type of arbitration implies that the bit time is at least twice the propagation latency on the bus. This defines a relation between the maximum bus length and the transmission speed. The available values are:

<table>
<thead>
<tr>
<th>Bit rate</th>
<th>Bus length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mbit/s</td>
<td>25 m</td>
</tr>
<tr>
<td>800 kbit/s</td>
<td>50 m</td>
</tr>
<tr>
<td>500 kbit/s</td>
<td>100 m</td>
</tr>
<tr>
<td>250 kbit/s</td>
<td>250 m</td>
</tr>
<tr>
<td>125 kbit/s</td>
<td>500 m</td>
</tr>
<tr>
<td>50 kbit/s</td>
<td>1000 m</td>
</tr>
<tr>
<td>20 kbit/s</td>
<td>2500 m</td>
</tr>
<tr>
<td>10 kbit/s</td>
<td>5000 m</td>
</tr>
</tbody>
</table>
**CAN bus**

Error and fault containment

There are 5 types of error

**BIT ERROR**

The sender monitors the bus. If the value found on the bus is different from the one that is sent, then a BIT ERROR is detected.

**STUFF ERROR**

Detected if 6 consecutive bits of the same type are found.

**CRC ERROR**

Detected by the receiver if the received CRC field does not match the computed value.

**FORM ERROR**

Detected when a fixed format field contains unexpected values.

**ACKNOWLEDGEMENT ERROR**

Detected by the transmitter if a dominant value is not found in the ack slot.

A station detecting an error transmits an ERROR FLAG.

For BIT, STUFF, FORM, ACKNOWLEDGEMENT errors, it is sent in the immediately following bit.

For CRC it is sent after the ACK DELIMITER.
CAN bus

Fault containment
Each node can be in 3 states:
   Error active
   Error passive: limited error signalling and transmission features
   Bus off: cannot influence the bus

Each node has two counters:
   TRANSMIT ERROR COUNT:
      increased – (list) by 8 when the transmitter detects an error …
      decreased – by 1 after the successful transmission of a message
         (unless it is 0)
   RECEIVE ERROR COUNT:
      increased – (list) by 1 when the node detects an error, by 8 if it
detects a dominant bit as the first bit after sending an error flag …
decreased – (if between 1 and 127 by 1, if >127 set back to a value
between 119 and 127) after successful reception of a message
Assumption 1: nodes are not synchronized, nor any assumption on local clocks is used by the MW and driver levels.

Assumption 2: messages are always transmitted by nodes based on their priority (ID) – ideal priority queue of messages.

Assumption 3: periodic messages, but no assumption on the message phases.
Critical instant theorem: for a preemptive priority based scheduled resource, the worst case response time of an object occurs when it is released together with all other higher priority objects and they are released with their highest rate.
CAN bus

Timing Analysis – worst case latency – Ideal behavior [2]

The transmission of a message cannot be preempted

\[
q_i = B_i + I_i
\]

\[
w_i = q_i + C_i
\]

\[
I_{i,j} = \sum_{j \in \text{hp}(i)} I_{i,j}
\]

\[
q_i = B_i + \sum_{j \in \text{hp}(i)} \left\lfloor \frac{q_i}{T_j} \right\rfloor C_j
\]

Fixed point formula: solved iteratively by setting \(q(0)=0\) until the minimum solution is found

An example (\(C_i\) computed for maximum size, bus speed 500 kb/s)
CAN bus

In reality, this analysis can give optimistic results!
A number of issues need to be considered …
  – Priority enqueuing in the sw layers
  – Availability of TxObjects at the adapter
  – Possibility of preemption (aborting) a transmission attempt
  – Finite copy time between the queue and the TxObjects
  – The adapter may not transmit messages in the TxObjects by priority

Adapters typically only have a limited number of TXObjects or RxObjects available
CAN bus

A number of issues need to be considered …

– …

– Availability of TxObjects at the adapter

• Let’s check the controller specifications!

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th>Buffer Type</th>
<th>Priority and Abort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microchip</td>
<td>Standalone controller</td>
<td>2 RX + 3 TX</td>
<td>lowest message ID, abort signal</td>
</tr>
<tr>
<td>MCP2515</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATMEGA3285</td>
<td>8 bit MCU</td>
<td>15 TX/RX</td>
<td>lowest message ID, abort signal</td>
</tr>
<tr>
<td>AT90CAN32/64</td>
<td>w. CAN controller</td>
<td>mag. objects</td>
<td></td>
</tr>
<tr>
<td>FUJITSU MR90385/90387</td>
<td>16 bit MCU</td>
<td>8 TX/RX</td>
<td>lowest buffer num. abort signal</td>
</tr>
<tr>
<td>90V495</td>
<td>w. CAN controller</td>
<td>mag. objects</td>
<td></td>
</tr>
<tr>
<td>FUJITSU 903890</td>
<td>16 bit micro</td>
<td>16 TX/RX</td>
<td>lowest buffer num. abort signal</td>
</tr>
<tr>
<td>Intel</td>
<td>16 bit MCU</td>
<td>14 TX/RX</td>
<td>lowest buffer num. abort possible [?]</td>
</tr>
<tr>
<td>87C106 (82527)</td>
<td>w. CAN controller</td>
<td>mag. objects</td>
<td></td>
</tr>
<tr>
<td>INFINITRON</td>
<td>16 bit MCU</td>
<td>32 TX/RX</td>
<td>lowest buffer num., abort possible [?]</td>
</tr>
<tr>
<td>XC9518F/167</td>
<td>w. CAN controller</td>
<td>mag. objects (2 bases)</td>
<td></td>
</tr>
<tr>
<td>PHILIPS 8xc565 (SJA1000)</td>
<td>8 bit MCU</td>
<td>one TX buffer</td>
<td>abort signal</td>
</tr>
<tr>
<td></td>
<td>w. CAN controller</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What happens if only one TxObject is available?

– Assuming preemption of TxObject

| id = 0x304 |
| id = 0x261 |
| id = 0x103 |
| id = 0x341 |
| id = 0x2d2 |
| id = 0x2a1 |

Preemption

id = 0x103

Priority inversion for =0x261

AFTER its queuing time

id = 0x122

id = 0x122
FlexRay
- Successor to CAN, higher bit rate and more ECUs
- Periodic transmission of messages, with period broken into a “static” segment and a “dynamic” segment
- Static segment has slots assigned to ECUs in a fixed way – ensures guaranteed slot
- Dynamic segment provides “extra slots” for soft real-time tasks

![Diagram of FlexRay communication cycle]

CAN bus

Bibliography