|  |
| --- |
| University of Leicester: Msc Reliable Embedded Systems |
| Project Proposal |
| Using remotely-configurable acceptance filtering in a time-triggered task schedule to log data over a Controller Area Network. |
|  |
| Chris Barlow |
| 5/30/2012 |

|  |
| --- |
|  |

# Background

## Electric Commercial Vehicle Telemetry System

An electric commercial vehicle company uses a telemetry device to log data from numerous CAN buses on their vehicles and several dedicated sensors. This data is transmitted over GPRS to an AMQP message queue, where a dedicated server performs the necessary post-processing to store the information in a database.

**Vehicle Comms buses**

**Mass storage (microSD card)**

**GSM Module**

**Remote AMQP message queue**

**Logging**

**Transmission**

Interrupt

Interrupt

**Internal hardware timer**

FIGURE 1: EXISTING REMOTE DEVICE SOFTWARE OVERVIEW

* **Data logging functions** - performed using timer-driven interrupts.
  + CAN and sensor data are collected from the vehicle and buffered internally.
  + Buffered data is compressed and stored on a micro SD card in data ‘blocks’ of around   
    4 – 10 kB.
* **Data transmission functions** – performed using an interrupt on the GSM module, indicating that the module is connected to the network.
  + A connection is made to a remote AMQP message queue.
  + Data is read from the SD card. The large data blocks are split into several 1kB ‘chunks’ and sent to the message queue.
  + If the device loses network signal, the software sits in one of several ‘while’ loops until a GSM interrupt event occurs. During this time, the data storing functions are still operating on timer interrupts, allowing the device to still collect data during periods of low or no GPRS signal.

The compression logic used requires each CAN message, identified by a unique ID, to be stored in the same location every time so that comparisons can be made between old and new data. For this reason the CAN data is filtered and stored by the software as follows:

New message arrives on CAN bus

ID of message on transceiver is read and compared to internal list of log-able IDs

If logging of the message is required, a pointer is retrieved   
to give access to the buffer location for that specific message.

Data is stored in specified location in CAN buffer

FIGURE 2: SOFTWARE FILTERING IN CAN DATA LOGGING PROCESS

New hardware and software approaches are now being explored for an updated device, including the use of a full Time-Triggered Cooperative (TTC) scheduler and the ability to remotely modify which CAN messages are logged by the device. It is therefore advantageous to investigate software logic that complements the inherently predictable nature of the TTC scheduler, while still being compatible with the compression and transmission protocols that are currently in use by the server.

## CAN Acceptance Filtering

At present the devices use several Microchip MCP2515 CAN transceivers to connect to the vehicle’s CAN bus. These transceivers are by the microprocessor using a Serial Peripheral Interface (SPI). This allows control over many features of the transceiver, which include:

* Implements CAN V2.0B at 1 Mb/s:
  + 0 – 8 byte length in the data field
  + Standard and extended data and remote frames
* Receive buffers, masks and filters:
  + Two receive buffers with prioritized message storage
  + Six 29-bit filters
  + Two 29-bit masks
  + Data byte filtering on the first two data bytes (applies to standard data frames)

(Microchip Technology Inc., 2007)

As seen in 1.2 above, message filtering is carried out in the software after each message has been stored in the MCP2515 buffers. The transceivers do, however, possess the ability to filter incoming messages at hardware level. The two acceptance masks (RXM0 & RXM1) and six acceptance filters (RXF0 – RXF5) are applied independently to the two receive buffers (RXB0 & RXB1), allowing different messages to be collected in each buffer. As a CAN message arrives at the transceiver, it is initially stored in a single Message Acceptance Buffer (MAB). The contents of the MAB is then compared to the mask and filters of the two buffers and stored in the appropriate buffer if it passes the acceptance criteria.

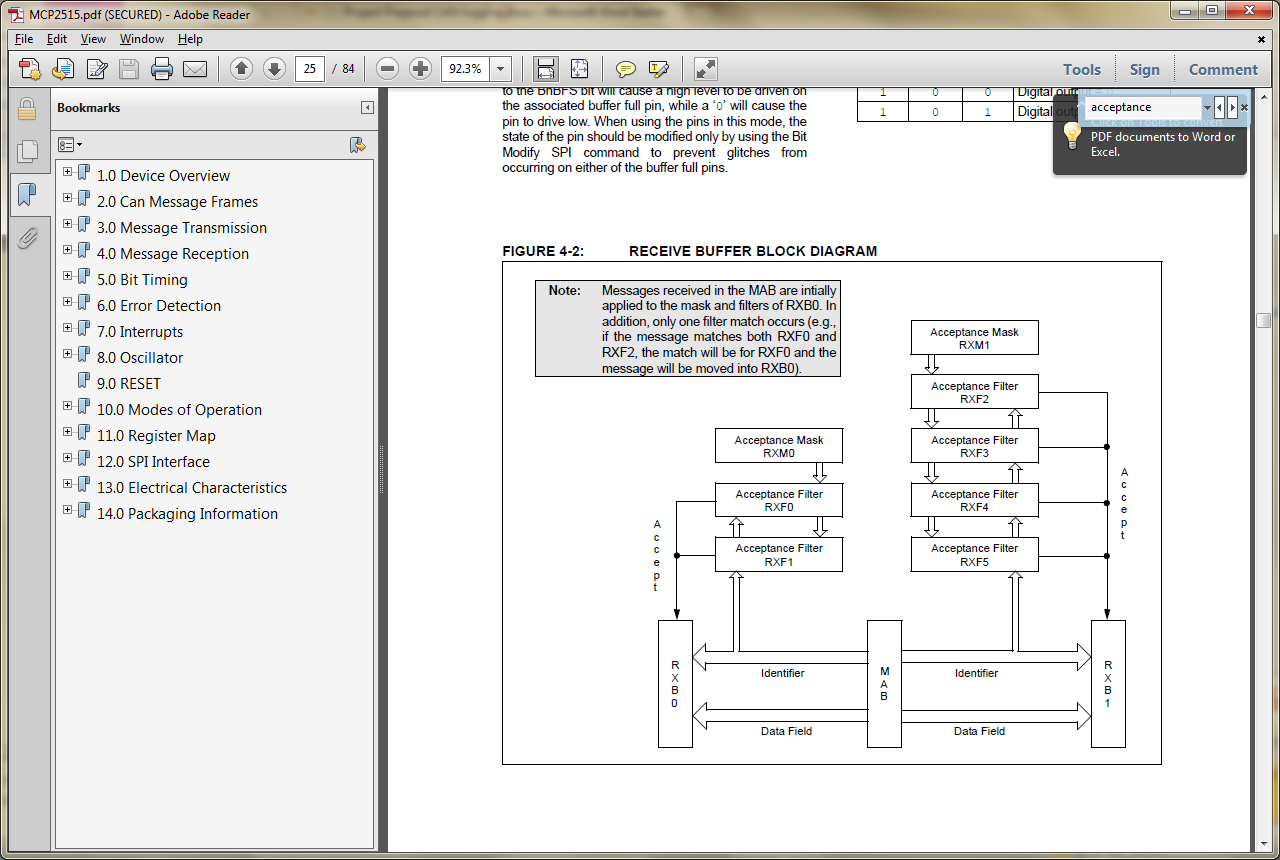


FIGURE 3: RECEIVE BUFFER BLOCK DIAGRAM (Microchip Technology Inc., 2007)

The use of this acceptance filtering feature at hardware level could potentially free up processing time in the device software as there is no need to perform the comparison to check that the message is needed.

# Project Aims

This project will focus on the development of a new method of obtaining CAN data specified in a ‘logging list’ (a configurable list of CAN IDs that need to be logged) and storing this data in a precise location for retrieval by other software functions. This method will exploit the fact that the order and timing in which individual messages are published over the CAN bus is relatively predictable. The logging list will be collected from the remote server and used along with the order / timing of CAN bus messages (‘CAN sequence’) to produce a timed ‘logging sequence’.

In a time-triggered environment, we can use this logging sequence to predict the ID of the next message on the CAN bus at any given ‘tick’. A periodic task can then modify the CAN transceiver’s acceptance filter(s) to only accept the ID of the expected message on the bus for any given time (see Figure 4 below).

When the next CAN message arrives in the receive buffer of the transceiver, the software will then know two points for certain:

* The message is on the logging list.

**TTC Scheduler**

Periodic filtering task

Periodic logging task

Timer

**CAN Bus**

**Logging Sequence**

CAN Sequence\*

Logging List

**Data log**

Correct location for received data

**CAN Transceiver**

RX Buffer

RX Mask / Filter

Unfiltered data in known sequence

Next required CAN message

Filter set to next CAN ID in logging sequence

Time since last logged message

Data storage

Data retrieval

CAN ID and time of next message in sequence

CAN IDs to log

**Remote Server**

* The CAN ID of the message and therefore the location in which it needs to be stored.

FIGURE 4: PROPOSED DYNAMIC ACCEPTANCE FILTERING ARCHITECTURE

For the purposes of this project, the CAN sequence will be initially produced manually from a logged CAN Sequence, however future development could include the ability to create the CAN sequence automatically in software from the data present on the bus when the device first starts up.

This method has several benefits over the existing software filtering:

* The method complements the nature of the TTC scheduler – timing of the data logging task(s) becomes consistent and predictable.
* Only CAN messages that need to be logged are read by the device in software.
* The logging list can be easily modified at the remote server to change the data that the device logs.
* Potential for the CAN transceiver to be put into ‘sleep’ mode between acceptable messages, saving electrical power during this time.

The project will test the hypothesis:

* There will be a saving in processing time compared to the existing software filtering method due to the fact that filter modification only needs to be carried out once between logged messages, compared to interrogating several unwanted CAN messages.

The questions that this research will attempt to answer are:

* How can the system cope with the order of messages changing, or new devices coming online?
* How do we determine the ‘major cycle’ of the logging sequence?
* What is the highest allowable the bus loading before messages are missed?
* Is it more economical to perform software filtering for lower bus loading?
  + If so, what is the lowest level that the proposed method is still beneficial?
* Are there alternatives to using an off-the-shelf CAN transceiver that would improve the efficiency of the architecture?

# Relevance to the Course: *MSc Reliable Embedded Systems*

This use of a TTC scheduler in this method will make this project directly relevant to the MSc course, encompassing knowledge gained (at time of writing) from the following modules:

* ***Module A1a: Programming techniques for reliable embedded systems (“Embedded C”)***
* ***Module A2a: Design and verification of high-integrity embedded systems***
* ***Module B3a: Multi-processor and multi-core designs for reliable embedded systems***

Further potential in the project to move away from off-the-shelf CAN transceivers may additionally be relevant to:

* ***Module B1a: Using FPGAs in reliable embedded systems***

# Progress to date

* An informal but in-depth code examination has been carried out on the current telemetry device software. The knowledge gained from this will allow for an in-depth review of the software as it stands, giving a benchmark that can be used to measure the performance of the proposed system (exact performance metrics are yet to be established).
* A review of the current hardware and software functionality has been carried out including cost analysis, vehicle system integration and future-casting.
* Begun to develop a tool / script to analyse CAN data logs from the vehicles. This will give a better insight into the behaviour of the CAN bus on live vehicles.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Nov 2012 | Dec 2012 | Jan 2013 | Feb 2013 | Mar 2013 | April 2013 | May 2013 | June 2013 | July 2013 | Aug 2013 | Sept 2013 | Oct 2013 | Nov 2013 | Dec 2013 |
| Module B1a |  |  |  |  |  | Course: 15th-19th |  | Exam: 10th |  |  |  |  |  |  |
| Module B2a |  |  |  |  |  |  |  | Course: 10th – 14th |  |  | Exam: 16th |  |  |  |
| Module B3a |  | Exam (early option): 3rd (TBD) |  |  |  | Exam: 15th (TBD) |  |  |  |  |  |  |  |  |
| Proposal update | 9th |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Literature review / background research | 23rd |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Desktop feasibility Simulation & write-up |  | 21st |  |  |  |  |  |  |  |  |  |  |  |  |
| TT implementation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Thesis draft |  |  |  |  |  |  |  |  |  |  |  | 31st |  |  |
| Thesis Submission |  |  |  |  |  |  |  |  |  |  |  |  |  | 31st |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |