|  |
| --- |
| 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 |
| 14/11/2012 |

|  |
| --- |
|  |

# 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 1.1: 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.

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.

## Message Storage in the Existing Device

The compression logic used by the telemetry system 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:



Figure 1.2: SOFTWARE FILTERING IN THE CAN DATA LOGGING PROCESS – EXISTING REMOTE DEVICE

This means that whenever a CAN message is seen on the CAN bus, the device has to perform some processing on it, whether the device is interested in the message or not.

## CAN Acceptance Filtering

Plans are to build the new Remote Device around an ARM Cortex M4 processor such as the STM32F407ZGT6. This processor provides 28 filter banks, each capable of holding four 16-bit Identifiers [1]. As a message arrives on the CAN bus, the processor transparently compares the identifier with those in the filter lists, and stores it in a specific memory location. If the message doesn’t match any of the filters, it is discarded.

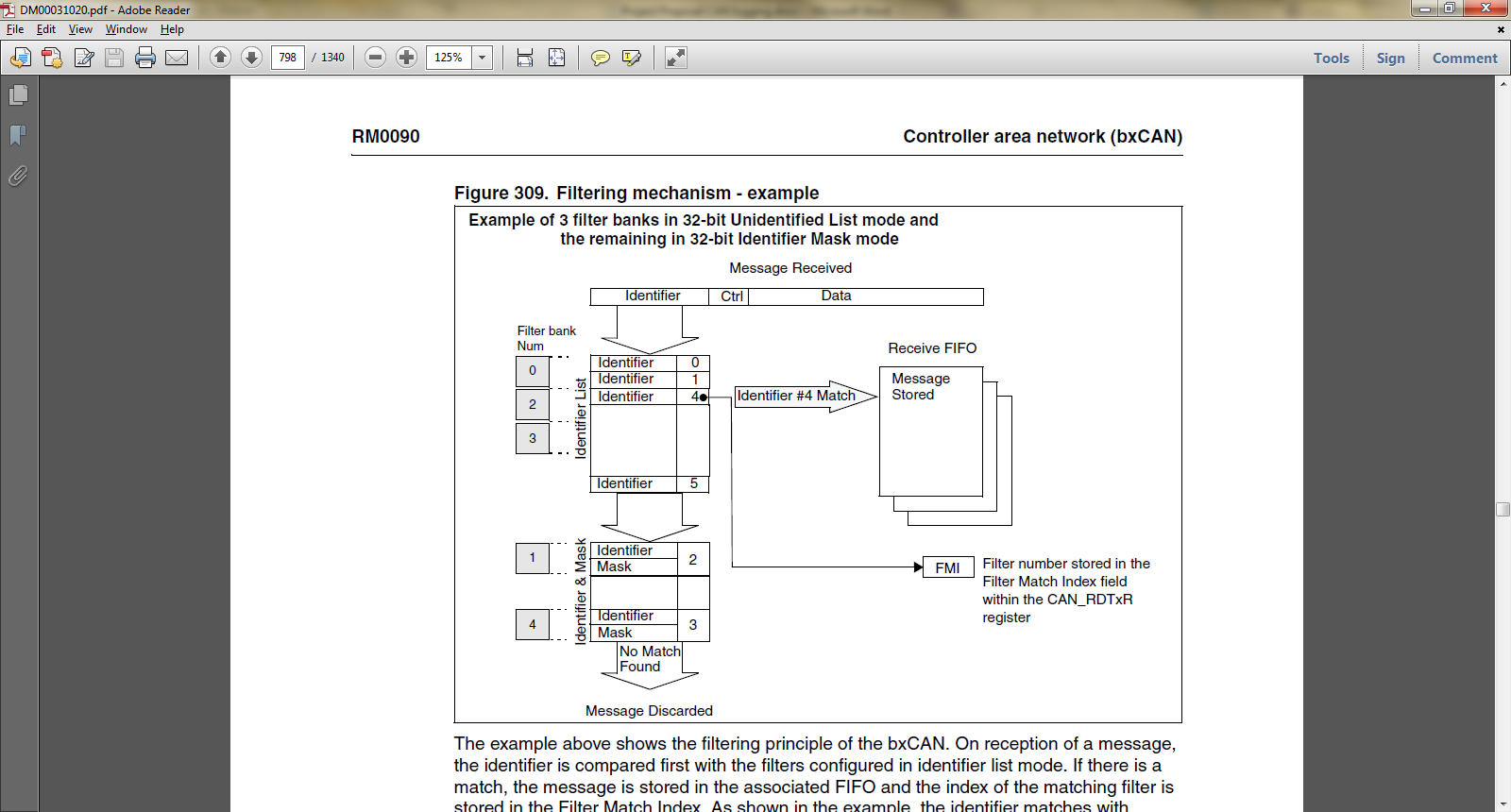


Figure 1.3: CAN Acceptance Filtering on an STM STM32F407ZGT6 Processor

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 known 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 IDs of the next messages on the CAN bus at any given ‘tick’. A periodic task can then modify the bxCAN acceptance filters to only accept the IDs of the expected messages on the bus for any given time.

As the sequence of CAN messages won’t be exactly uniform, we can specify multiple filters in the bxCAN’s identifier list to accept several IDs at the same time. This can then be modified dynamically as messages arrive.

## Proposed Process

The process would be as follows (see diagrams in Appendix A):

* The device will store, in memory:
  + A logging sequence of *x* CAN IDs, identified by a Message ID (array position);
  + A filter reference table which associates the Message IDs currently being filtered to the Filter Index of the bxCAN acceptance filter;
  + The latest Filter Match Index (FMI) given by the receive FIFO.
* The processor’s bxCAN acceptance filter will be configured to accept *y* messages at a time;
* During device initialisation, the bxCAN acceptance filter is loaded with the first *y* CAN IDs from the logging sequence, and the filter reference table is given Message IDs 0 to *y*;
* As a message arrives on the CAN bus, the processor transparently compares the ID with those in the filter list. If it does not match any of the IDs in the list, it is discarded. If it matches one, it is stored in the receive FIFO alongside the Filter Match Index (FMI), which identifies which filter the message matched;
* A periodic logging task, executed from the TTC scheduler, uses this FMI and the Filter Reference Table to determine the location in which to store the message data;
* A second periodic task, the Filter Update task, replaces the last matched bxCAN filter with the next ID in the sequence, and updates the Filter Reference Table accordingly.

This process ensures that, when a 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, and therefore should be logged;
* The CAN ID of the message and therefore the location in which it needs to be stored.

## Predicted Benefits

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;
* Most of the filtering logic is carried out by the hardware, freeing up processing time for the scheduled tasks;
* The logging list can be easily modified at the remote server to change the data that the device logs.

## Uncertainties

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?
* Increasing the number of IDs (*y*) for the filter theoretically increases our allowable error margin for the CAN sequence. Are there disadvantages in using a larger filter list?
* As *y* varies, how does this affect the importance of the order of messages in the logging sequence?
  + Is it necessary to adjust this logging sequence dynamically?
* How do we determine the ‘major cycle’ of the logging sequence?
* Will messages be missed as the bus loading increases, and at what level will this become a problem?
* 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?

# 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 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.
* Developed a tool / script in C to analyse CAN data logs from the vehicles. This has given an insight into the behaviour of the CAN bus on live vehicles, but has been configured around a single-message acceptance filter. Further work is required to adapt this simulation to include the multi-filter scheme proposed in this document.

# Provisional Thesis Headings

The headings below are proposed to be used as an outline plan for the project thesis:

## Introduction

### Review of Existing Software and Technology

### Reasons for Development

### Background Research

### Controller Area Network Overview

### CAN Hardware

### Acceptance Filtering

### Dynamic Filtering

### Related Work / Literature Review

### Feasibility Simulation

### Theory

### Metrics

### Method

### Results

### Hardware Implementation incl. Remote Configuration

### Testing

### Metrics

### Method

### Results

### Discussion

### Simulation and Hardware Implementation Comparison

### Performance Comparison with Existing System

### Future Development

### Conclusions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 to 19th |  | Exam: 10th |  |  |  |  |  |  |
| Module B2a |  |  |  |  |  |  |  | Course: 10th to 14th |  |  | Exam: 16th |  |  |  |
| Module B3a |  | Exam: 3rd |  |  |  |  |  |  |  |  |  |  |  |  |
| Proposal update (This document) | 14th |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Literature review / background research |  |  | 31st |  |  |  |  |  |  |  |  |  |  |  |
| Desktop feasibility Simulation & formal write-up |  |  | 31st |  |  |  |  |  |  |  |  |  |  |  |
| Hardware Implementation |  |  |  | 31st |  |  |  |  |  |  |  |  |  |  |
| Testing and validation |  |  |  |  | 31st |  |  |  |  |  |  |  |  |  |
| Alterations / Modifications reflective of Module B1a |  |  |  |  |  | 30th |  |  |  |  |  |  |  |  |
| Final testing |  |  |  |  |  |  | 10th |  |  |  |  |  |  |  |
| Thesis draft (deadline) |  |  |  |  |  |  |  |  |  |  |  | 31st |  |  |
| Thesis Submission (deadline) |  |  |  |  |  |  |  |  |  |  |  |  |  | 31st |

# Provisional Plan

# References

|  |  |
| --- | --- |
| [1] | STMicroElectronics, “Controller area network (bxCAN),” in *RM0090 Reference Manual: STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx advanced ARM-based 32-bit MCUs*, 2012, pp. 784 - 825. |

# Appendix A: Proposed CAN Filtering Process





