SLIDE 1

# Why MMC?

There’s been a long standing desire to have a cross-platform ‘version’ of FCU

Several projects have been started with this in mind, MMC just being one example

This was originally started by Nigel Phillips, who has kindly allowed me to develop it further

# It’s not a replacement for FCU

My initial aim was to trial some different techniques to build a management tool, and so I make no apologies for MMC working differently to FCU in some areas – that’s by design

The focus has been on getting these areas to work, and many features that the FCU provides are not yet available

Please accept that it’s still a bit ‘rough around the edges’

# So, what’s so different?

Next slide…

Slide 2

# Cross-platform

This has been achieved mainly by the use of a platform called Node-JS, which is supported on all main platforms, such as Windows, Linux & Apple

# Module Descriptor Files

Coding specific ‘pages’ for each new (or modified) module type obviously adds a significant time delay to supporting any module, not to mention the workload on the authors

The notion of creating an external file that could describe any module, such that existing code, without modification, would then know how to present the variables was discussed

This would allow anyone to create a descriptor file for their own design of modules, without waiting for the authors to add it to the tool

I’ve managed to implement a format to do just that – and MMC uses this to present the configuration pages for all modules – no exceptions

The intention is for this format to be available as a ‘standard’ for any tool to use – I hope that FCU might also consider using it

There is a separate project that defines this format and has working examples of most of the existing modules – so it’s not tied to the MMC project

# Refreshing data from nodes

Due to the dynamic nature of the variables on some modules (like CANMIO-Universal), the MMC uses ‘live’ data by reading the node where possible, e.g. refreshing the variables from a node when changes are made. This is different behaviour to FCU, which aims to keep a local ‘copy’ of the data, which, whilst less bus traffic, unfortunately can get out of sync

# Event ‘teaching’

MMC works differently to FCU as it gives direct access to add and teach events, without needing a ‘software node’. This more closely reflects how the actual CBUS commands work

# Automatic CAN adapter detection

Coding has been added to automatically check for CANUSB & CANUSB4 adapters, and use the first one it finds, and if no adapter is found, it will default to a network connection

# VLCB support

This project was designed with CBUS in mind, and predates the VLCB project, so there’s no inherent tie-in between MMC and VLCB. However, most of the extensions that VLCB adds to CBUS have now been added to MMC

# Unit testing

One of the things I learnt in my latter years of professional embedded software development was the value of unit testing, and I know others such as Sven shares this view, But it’s a whole topic in it’s own right, so maybe something for another day

Now the demo…..

Ask for questions at the end

# Start MMC with ‘real’ modules

# Start MMC

1. Explain how MMC doesn’t currently use a stored list of nodes, but finds all nodes on startup

# Describe node list

1. Use CANACC5
2. Show parameter dialog
3. Show node variable dialog – show MDF
4. Show MDF that describes the variables
5. Show naming of nodes – mention names are persisted against ‘layout’
6. Invoke events – show there are no events

# Show name event

1. Use CANMIO
2. Explain we’ll go into detail on CANMIO later
3. Show name event – change event 1 to ‘CANMIO 1’
4. Mention names are persisted against ‘layout’

# Show teach event #1

1. Use CANMIO
2. Show teach event – teach “CANMIO 1” to CANACC5
3. Event variable dialog will be displayed – assign to output 1
4. Show CANACC5 now in list of nodes for this event
5. Demonstrate led controlled by switch (camera)
6. Show bus events
7. Explain ‘bus events’ are just the events that have been seen (transmitted on the actual CAN bus)

# Show event in CANACC5

1. Open events for CANACC5
2. Show variables dialog
3. Show teach for this event lists both modules for event

# Show teach event #2

1. Use CANACE8C
2. Explain it is a producer with default events that aren’t listed as stored events
3. Show switch 1 appearing in bus events & event list
4. Swap to CANACC5 – add event for switch 2 for LED 2
5. Operate switch 2 – see led 2 operate (camera)
6. Show it appear on bus events

# Show test event

1. Use CANACE8C
2. Show test event on event 2 – operates LED 2 (camera)
3. Show test event from ‘bus events’

# Show Universal module

1. Show how changing channel type changes other variable options
2. DON’T show this MDF – it’ll scare people
3. Show testing servo limits

# Show update MDF dialog

1. but not necessary to actually use it

# Show VLCB dialog

1. May need to use simulated module

Other things to consider

1. Start simulator to show a large number of modules
2. Explain multiple client can be run, from other networked machines
3. MMC starts a network port, that FCU can also connect to
4. Data that isn’t stored on modules (like node & event names) are stored on the computer against the layout name
5. Show architecture diagram if asked (below)

Architecture

To facilitate the cross platform capability, The user interface is browser based

When the application is started, this will start the default web browser to provide the user interface

MMC-SERVER

MMC-CLIENT

Web Server

NodeJS application

Web Browser