PowerPC Development Toolchain

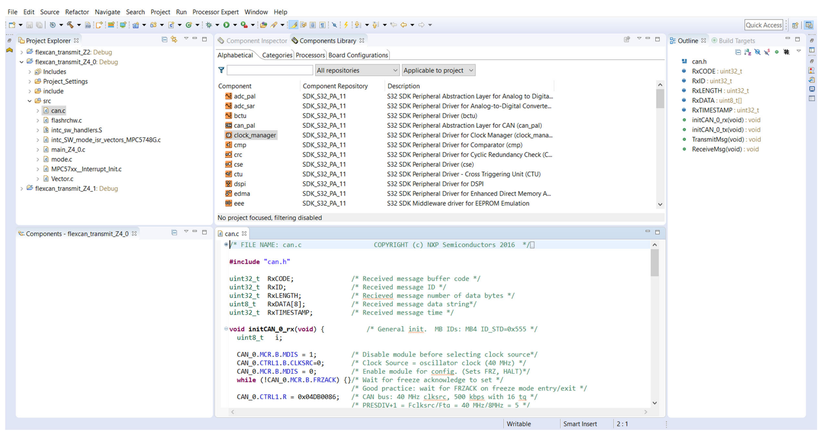
Students: BEN ETTLINGER & SUBHOJEET MUKHERJEE
Advisor: DR. JEREMY DAILY

Processor Architectures

Processor architectures are the methods used by processors to execute assembly/machine code. Common architectures include x86, found in most desktop and laptop processors and ARM, mostly found in mobile devices.

PowerPC

PowerPC is an implementation of POWER Architecture for microprocessors originally introduced in 1992. PowerPC-based microprocessor are commonly found in both legacy and new vehicle electronic control units (ECUs).

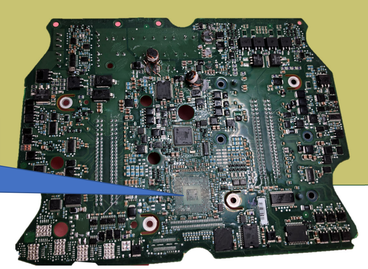


Programming for PowerPC w/ S32 Design Studio

Programming either of the target development boards requires the use of the S32 Design Studio by NXP. High-level programming is done in regular C or C++ languages; porting this code is as simple as refactoring the C code of the project into the S32 Design Studio IDE.

Libraries and Peripherals

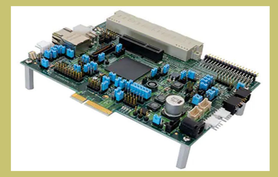
The process becomes more involved when implementing libraries and using peripherals. The toolchain provides granular control over the processor functions. Processor clocks, timers, and subsystems must be individually enabled and configured. Each processor configuration has individually generated libraries which are referenced in the main code.



Conclusion:
Programming for the PowerPC Architecture requires detailed configurations for all the peripherals and Controller Area Network hardware.

Stated Goal

Port a project written in C to the device with PowerPC Architecture, specifically the MPC 5777C-Dev B and the MPC 5748G Development boards pictured below:



MPC 5777C-Dev B

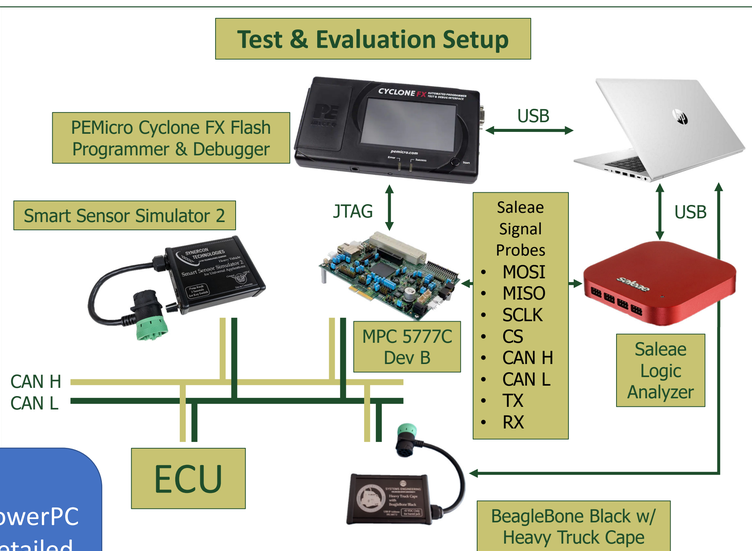
DEVKIT MPC 5748G

While both development boards utilize POWER Architecture, the MPC 5777C more closely resembles the processors found in modern ECUs and was the primary target for development.

Target Project

The target project is written in the C programming language and receives data over CAN, processes the information, and then sends a response over CAN. Successful implementations had already been written for the ARM architecture on the BeagleBone Black microcomputer.

Test & Evaluation Setup



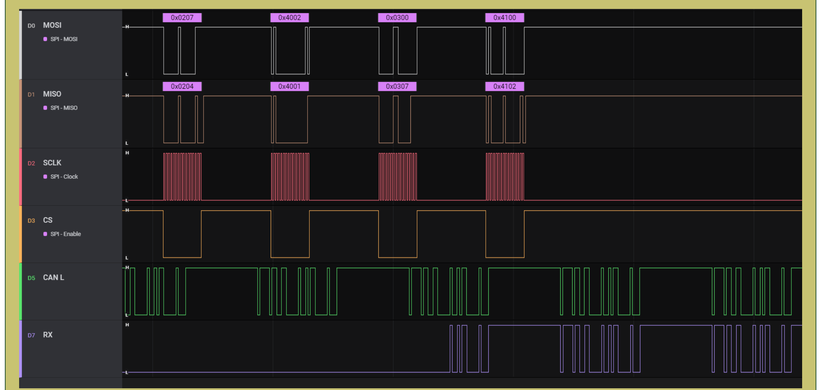
Exploration into CAN on MPC 5777C Dev B

While implementation of CAN interfaces on the 5748G Devkit readily worked from included examples, implementation on to 5777C Dev board required more configuration and troubleshooting.

- Transceiver Troubleshooting w/ SPI

While the 5748G Devkit utilized a transceiver which was hardware-enabled, the 5777C Dev B used a different transceiver which required configuration over SPI (Serial Peripheral Interface), a protocol used to communicate between microprocessors and microchips. Facilitating this exchange required the enabling and configuration of a new peripheral interface on the development board, resulting in the simultaneous troubleshooting of hardware configuration, CAN peripheral configuration, and SPI peripheral configuration. This was achieved through the debugging interface and signal analysis using a Saleae logic analyzer, the output of which is seen below.

Saleae Signal Output



In the above screen capture we see:

- MOSI (microprocessor output)
- MISO (microchip response)
- SCLK (timing clock from microprocessor)
- CS (Chip Select which enables/disables communication with the chosen chip(s))

The SPI messages are shown in hexadecimal format in the purple boxes above the MISO and MOSI signals and were written based on information found in the transceiver datasheets.

CAN Messages can be seen on the CAN L signal, with the RX signal going live after the SPI configuration messages.