E491 - Project Description

Fully Programmable Engine Control Unit

David Tolsma

December 12, 2019

Contents

Abstract

The ECU is an integral part of a modern engine system. It is responsible for taking in sensor data from many places in the vehicle including air temperature, engine temperature, manifold pressure, oxygen sensor data, crank angle and various other inputs, processing the data, then determining the correct amount of fuel to inject, when to trigger a spark, as well as controlling other critical engine components.

This project aims to create a fully programmable engine controller for four-cycle automotive and small engine applications.

1 System Design Formulation and Specifications

1.1 Hardware System-Level Description

1.1.1 Micro-controller Choice

This engine controller has many inputs and outputs and therefore needs a wide range of analogue and digital inputs and outputs. A high level of integration for analogue to digital converters, digital to analogue converters, digital IO pins, chip to chip communications, external communications such as a CAN peripheral, and other integrated features would help to lower overall part counts, board complexity, and cost.

Finding a processor that is rich in peripherals is not enough however as we need a processor that is fast enough to respond in a short amount of time in order to meet the high precision timing requirements that we need. Due to the large number of mathematics computations needed, having support for a floating point unit (FPU) would be a desirable feature in order to reduce overall processing load as well as making software development easier.

As the software component of this project is quite involved, ease of software development is a serious concern, so picking a processor that is established in the industry, has a robust set of programming and debugging tools available, and has a large set of example code is another serious consideration in the micro-controller choice.

When looking at the performance needs, we are pointed toward an ARM Cortex-M4 based micro-controller. The chips based on this core tend to have a good balance between cost and performance for an application such as this.

Looking at the micro-controller choices of other projects we can see that these are common choices for similar projects:

- rusEFI - Various boards all using chips from the STM32F4 family, Cortex-M4F, 160-180 Mhz, FPU, 128k-2Mb flash, 128k-1M RAM, 64-128 pins
- Motorsports Electronics ME221 - Kinetis K66F family, Cortex-M4F 180 Mhz, FPU, 2Mb flash, 256 kB RAM, 144 pins
- Megasquirt 3 Pro - Freescale MC9S12XE, 16 bit, 50 Mhz, 1 Mb flash, 64 kB RAM, 144 pins
- Speeduino - ATmega2560, 16 bit, 16 Mhz, 256 kB flash, 8 kB RAM, 100 pins

We can see with the choices that other projects have made, an ECU can be created on quite varied hardware in terms of performance, with the ATmega2560 and the Freescale MC9S12XE being by modern standards not particularly capable processors. We also see that the flash and RAM requirements can be very low with the Speeduino having only 8kB of RAM. While these projects were able to implement fully featured engine controllers with limited hardware, the more powerful processors of the other two projects allow for more flexibly in program design as optimization for code efficiency is not as important. Running a real time kernel will aid development, but use program memory, RAM, and processing cycles that might not be able to be spared with some of the lower powered processors. Taking lead from the more powerful processors that have been chosen leads me to choosing a similar Cortex-M4F based micro-controller.

The STM32F4 family seems particularly suited to a project such as this as it has a well developed software environment, has low cost development hardware, great community support, low cost, and has some very handy peripherals. While these processors can be a good choice, a relatively new line of processors was released by STMicroelectonics, the STM32G4 series. This is basically an STM32F4 with added features meant to aid in motor control. It has the same floating point and math accelerators as the F4 series, the same communications peripherals, and same ARM core. It however does have some new functionality that could help with a project such as this as this family has an abundance of ADCs, DACs, on-board OP-AMPS, hardware filtering, and better timer options. Various chips exist in this family with varying levels of hardware peripherals, memory, and RAM. The highest model in the lineup is the STM32G474. This processor is a 32 bit, Cortex-M4F core, runs at 170 Mhz, 512 Kb of flash, 128 Kb of RAM, 5 ADCs, 7 DACs, 7 comparators, 6 OPAMPS, 3 CAN-FD, hardware digital filtering, as well as a wide smattering of timers, DMAs, and other communications protocols, and comes in a 100 and 128 pin quad flat pack with pin pitch of 0.4 mm. This high level of analogue integration will help to reduce on board components, the powerful core and math peripherals will allow the project to process a lot of data very quickly (easing software development), has enough IO to communicate with all the necessary sensors and actuators, and has a robust software ecosystem, development boards, and examples.

While not all the peripherals will be used, having these peripherals will allow for future expansion. I also plan to use a large pin count version of the STM32G474 such as the 128 pin version the STM32G474QET, which comes in a 128 pin LQFP. While we need a large number of I/O for the project, a basic implementation will not utilize all 128 pins. The extra pins allow for better routing and board flexibility as well as easing future expansions.

1.1.2 Micro-Controller Resource Map

In 1.1.2 we can see a proposed block diagram for the overall system architecture. Some of the hardware blocks shown such as the CAN inputs and outputs and the USB device will be supported in hardware, but may not be implemented with software in the final project, leaving room for future software expansion. The area with the multiplexer shows the signals coming in from the variable reluctance sensors or the hall effect/optical sensors, only one of these are going to be used in a given engine configuration, so the inputs will be chosen with a hardware jumper or by using different off board connectors. The signal then either goes through a VR conditioner chip and from there goes to the MCU for further processing. The VR conditioner takes the raw VR signals and triggers an output pulse when the signal crosses zero. It is possible for the signal to bypass this conditioner and be fed into the op-amp inputs on the MCU where a combination of hardware triggering and hysteresis along with software filtering could replace the functionality of the VR conditioner chip, lowering cost and reducing board complexity. This would be a future software development requiring more software functionality to be written which is not in the current development plan. The hardware to bypass the conditioner chip will be implemented in hardware, but the device will initially use the signal conditioner until future development is done.

The hardware block diagram has square rectangles to represent on board hardware and rounded rectangles to represent off board signals or devices.

1.1.3 Power Supply Specifications

Most of the high-power peripherals such as injectors, spark plugs, fuel pumps, and fans will be controlled by the micro-controller by switching a large transistor on the 12V automotive rail. This means most of the high current devices will not be drawing on the 3.3 volt rail that the micro-controller is powered by. This makes our power supply demands quite low as only the logic level outputs and the micro-controller will be powered by the 3.3V rail with the rest of the power being supplied by the automotive 12V rail. This means we only need a few hundred milliamps at 3.3V. We can see in Apendix A that the total expected power consumption on the 3.3V rail is approximately 367 milliamps. Any buck converter should be able to supply at least 1A of total current.

Since the input voltage to the device will be 12V or more (possibly a sustained voltage of up to 16V) the voltage step down for the micro-controller will be up to 12V or more. While our current needs are quite small, even a few tens of milliamps can cause a significant amount of power to be dissipated in a linear regulator. This points to us using some kind of buck converter for the 3.3V power rails. This increases the power supply complexity a little bit, but allows us to have a much more efficient power conversion which will limit the heat being produced in our enclosure.

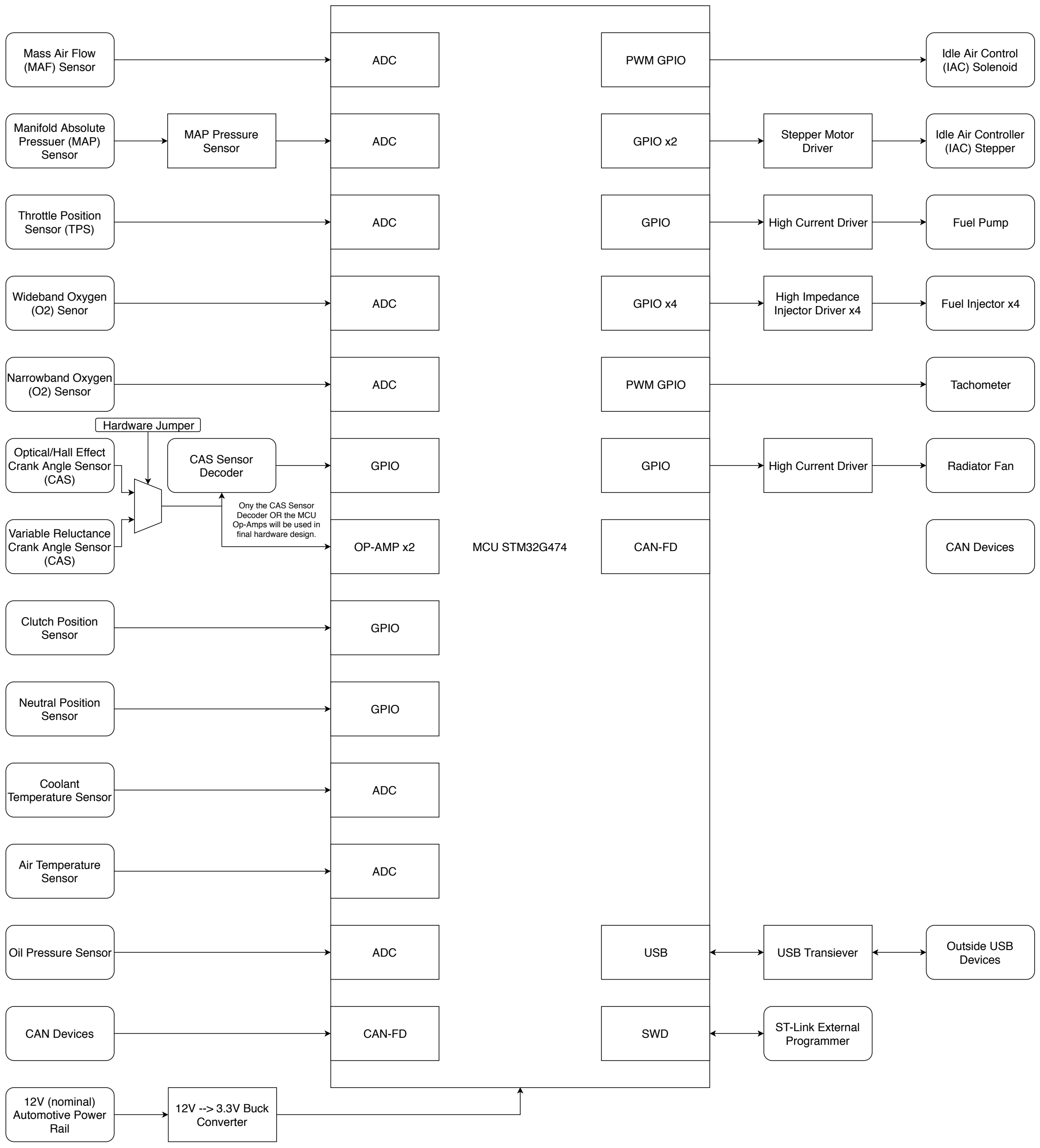
Mass Air Flow (MAF) Sensor
Manifold Absolute Pressuer (MAP) Sensor
MAP Pressure Sensor
Throttle Position Sensor (TPS)
Wideband Oxygen (O2) Senor
Narrowband Oxygen (O2) Sensor
Hardware Jumper
Optical/Hall Effect Crank Angle Sensor (CAS)
CAS Sensor Decoder
Variable Reluctance Crank Angle Sensor (CAS)
Ony the CAS Sensor Decoder OR the MCU Op-Amps will be used in final hardware design.
Clutch Position Sensor
Neutral Position Sensor
Coolant Temperature Sensor
Air Temperature Sensor
Oil Pressure Sensor
CAN Devices
12V (nominal) Automotive Power Rail
12V --> 3.3V Buck Converter
ADC
ADC
ADC
ADC
ADC
GPIO
OP-AMP x2
GPIO
GPIO
ADC
ADC
ADC
CAN-FD
MCU STM32G474
PWM GPIO
GPIO x2
GPIO
GPIO x4
PWM GPIO
GPIO
CAN-FD
USB
SWD
Stepper Motor Driver
High Current Driver
High Impedance Injector Driver x4
High Current Driver
USB Transiever
ST-Link External Programmer
Idle Air Control (IAC) Solenoid
Idle Air Controller (IAC) Stepper
Fuel Pump
Fuel Injector x4
Tachometer
Radiator Fan
CAN Devices
Outside USB Devices

Figure 1: System Block Diagram

1.1.4 Major Peripheral Specifications

For a project like this, we are going to need many external peripherals. Going roughly down the list on the system block diagram, figure 1.1.2:

1. Op-Amp - Since many of the sensor inputs are going to be 12V signals, we will use an op-amp to lower the signal to a 3.3V range. The NCV20074DR2G is a good choice as it is AEC-Q100 qualified, has an input voltage up to 36V and comes in a 4 op-amp package.

2. MAP Pressure Sensor - This is a sensor used to measure the absolute manifold pressure (MAP), an example would be the MPX4250 which is a 2.5 BAR sensor

3. VR Signal Conditioner - This is a device that interprets the variable reluctance (VR) signals coming from the crank angle sensor and outputs a digital square wave that represents the VR signal. This will most likely be done with the MAX9926 which is a dual adaptive VR decoder chip

4. Buck Converter - This is a chip that allows voltage conversion from one voltage to another. A possible good choice for this component is the A5975D, a chip made by ST that is AEC-Q100 qualified (meaning it can survive the large voltage spikes and noise in an automotive system), can operate down to 4V, and provide a current output of up to 3A.

5. Stepper Motor Driver - This is a chip that is able to drive the stepper motor used in some idle air control systems. The L6219R made by ST is a diver that can be powered by the 12V system rail but also be controlled with the 3.3V control signals coming from the micro-controller.

6. Injector Drivers - These are a MOSFET or equivalent that are capable of driving high current to fuel injectors to cause them to flow. Injectors are likely to draw up to 2 amps. Any high speed power MOSFET should do, however a device such as L9339 by ST is suited rather well as it can switch at the high frequencies we need and has input protection that will allow it to stay safe in the noisy automotive environment. STP62NS04Z, or VNS7NV04P-E could be good choices as well.

7. Ignition Drivers - These have similar needs as the Injector drivers, so similar parts could be used.

8. CAN Transceiver - This is a chip that enables the micro-controller to communicate with an automotive CAN bus. There are a lot of 3.3V compatible transceivers, with MAX3051ESA being a good and low cost choice.

9. USB Isolation - This is a device to provide isolation between the ECU and the USB bus to protect external USB devices. ISO7221C would be a good choice for this.

1.2 Software System-Level Description

1.2.1 Programming Languages

The main programming language that will be used to code this project will be C. C is the standard choice for embedded programming and this project is no exception.

1.2.2 Real Time Kernel

A real time kernel is a key part of this project. Much of this project relies on very precise timing for injection and spark firing. This means we need solid and deterministic timing as if the spark is a bit too late, then that could cause serious engine performance and reliability issues. This necessitates a real time preemptive kernel such as FreeRTOS, MicroC/OS, ChibiOS, or others. As an open source project whatever kernel I use needs to have a free and open licence in order to allow open development and open distribution. This is also important to lower the overall cost of the project.

FreeRTOS is under active and heavy development, has a rich feature set, has been time tested and proven, and supports the chosen micro-controller architecture. These factors make FreeRTOS a good choice for this project.

1.2.3 Major Algorithms

There are going to be many software algorithms in order to implement this project. They have not yet been finalized.

1.2.4 OEM Software and Licensing Descriptions

FreeRTOS is a preemptive real time kernel that is distributed under the MIT licence which allows free distribution, modification, and commercial monetization. This very open licence removes any restrictions on using and distributing the software.

The tool-chains provided by STMicroelectonics include STM32MXCube, an free to use IDE, set of example projects and code, Hardware Abstraction Layer (HAL) drivers and low level hardware drivers, and a configuration tool to determine pin-outs, create startup code, and to create peripheral configuration code.

2 Development Plan

2.1 Project Development Tasks and Timeline

Since the software part of this project is going to be so involved, it will be important to get some of the software written before the beginning of Spring quarter. Due to the board being rather complex, it also should be planned to have time for at least two if not ideally three board revisions. This necessitates much of the preliminary part choices and rudimentary schematics to be ready by the start of Winter quarter. Below is a proposed schedule for the project:

Week	Date	Task
1	Jan - 6	Create schematics for initial prototypes
2	Jan - 13	Build and test prototypes, begin assembing full schematic
3	Jan - 20	Board Design and Prototyping
4	Jan - 27	Board Design
5	Feb - 3	Order board rev. 1, Write firmware
6	Feb - 10	Write firmware
7	Feb - 17	Assemble and test board rev. 1
8	Feb - 24	Do board redesign
9	Mar - 2	Order board rev. 2, Write firmware
10	Mar - 9	Write firmware
11	Mar - 16	Assemble and test board rev. 2
12	Mar - 23	Write firmware, create and order board rev. 3 if necessary
13	Mar - 30	Week of Spring Break
14	April - 6	Finalize crank angle sensing software
15	April - 13	Create software to read TPS, Air Flow, and O2 Sensors
16	April - 20	Create software for controlling fuel
17	April - 27	Create software for controlling ignition
18	May - 4	Implement startup, idle control, and general I/O
19	May - 11	Integrate PC tuning software and logging
20	May - 18	Integrate software systems
21	May - 25	Test project and do general debugging
22	June - 1	Create demonstration hardware and create project demonstration documents
23	June - 8	Demonstrate Project

Table 1: Preliminary Schedule

2.2 Required Development Hardware and Software

A development board for the chosen micro-controller family is available in an STM Nucleo-64 board. This board has a lower pin-count, namely the 64 pin, version of the chosen processor, but will be useful for doing proof of concept prototyping. The STM32MXCube ecosystem will be used to do all software development. The device will be programmed and debugged with a ST-Link.

Common lab tools will be used such as oscilloscopes, function generators, power supplies, multi-meters and soldering equipment. The software and computer equipment will be self-supplied.

In order to do bench testing of the device, an engine simulator or a ECU stimulator will need to be obtained or created. There are several commercially available products such as JimStim, which is an ECU stimulator designed for the Mega-squirt line of engine controllers. This is a rather expensive board, but the cost may be worth it to lower development time.

2.3 Safety Plan

There are no major safety risks with the development of this project. Use with a real engine or vehicle does cause safety concerns and it is noted that all hardware and software is developed for off-road use.

2.4 Demonstration Plan

As this project will not be able to be demonstrated with an engine on demonstration day, a test jig will be created to demonstrate the working principle of the project. Sensors will be attached to the project as possible with some sensor data such as oxygen sensor data being artificially inputted into the device. A simulation can be done with artificial data for the cam angle sensing, oxygen sensor data, and airflow being fed into the device. The device can then drive the necessary outputs such as spark plugs and fuel injectors by flashing something such as LED's or other actuators to demonstrate the working principle.

3 Appendix A - Preliminary Parts List

We see a preliminary list of parts below in table 2. The total cost of major parts is $52.32. A PCB is estimated to cost $10, and other required components such as passives and diodes and connectors another $20. This brings the total component cost to approximately $82. With a MSRP = 3x the component cost, we would expect the MSRP to be at least $240.

The total amount of current consumed on the 3.3V rail is 367mA. This means that the 3A buck converter is more than adequate to drive the controller.

Quantity	Part	Part Number	Unit Price	Extended Price	Current Consumption
1	Micro-controller	STM32G474QET	$5.93	$5.93	32mA
2	Op-Amp	NCV20074DR2G	$1.19	$2.38	100mA
1	MAP Sensor	MPX4250	$12.77	$12.77	10mA
1	VR Signal Conditioner	MAX9926	$6.53	$6.53	40mA
1	Buck Converter	A5975D	$4.40	$4.40	Max 86% efficiency, power consumption is 14% of total system current consumption
1	Stepper Motor Driver	L6219R	$2.68	$2.68	N\A (Power driven by automotive power rail)
4	Injector Drivers	L9339	$1.21	$4.84	N\A (Power driven by automotive power rail)
4	Ignition Drivers	VNS7NV04P-E	$0.75	$3.00	N\A (Power driven by automotive power rail)
1	CAN Transiever	MAX3051ESA	$2.16	$2.16	70mA
1	USB Isolator	ISO7221C	$2.81	$2.81	15mA
4	General Purpose I/O	L9339	$1.21	$4.84	N\A (Power driven by automotive power rail)

Table 2: Preliminary Part List

4 Bibliography

[1] Speeduino.com, 2019. [Online].
Available: https://speeduino.com/wiki/index.php/Speeduino. [Accessed: 25- Oct- 2019]

[2] Rusefi.com, 2019. [Online].
Available: https://rusefi.com/. [Accessed: 25- Oct- 2019]