Project 1

ECEN 5803

Mastering Embedded Systems Architecture

Vortex Flowmeter

Embedded Systems Proof of Concept

Author:
David Pasley

Author:
Ismail Yesildirek

October 8, 2018

University of Colorado
Boulder

Contents

Executive Summary

This feasibility study was carried out in response to Sierra Instrumentation's *Request for Services (RFS)* [1], in order to evaluate the Kinetis KL25Z microprocessor and to determine if it is a viable selection for the embedded system design for the proposed Sierra 240 Vortex Flowmeter. The NXP/Freescale FRDM-KL25Z evaluation board was used, and software was written in C and ARM assembly using Mbed SDK and the Keil uVision MDK. The testing comprised of four modules:

- Module 1: A program was written to find the speed at which the KL25Z can approximate a square root integer using the bisection method; results showed that the calculation takes less than 5 µs.
- Module 2: A program was written to read 3-axis accelerometer data over I^2C, analog data from a capacitive touch sensor, and also to drive three PWM outputs. The CPU load for this program was determined to be less than 2%.
- Module 3: A serial user interface was implemented. The CPU load was estimated to be ≈21%. A Dhrystone v2.1 Benchmark test was carried out, and the system was clocked at ≈46.9 DMIPS, or ≈0.978 DMIPS/MHz.
- Module 4: A frequency detection algorithm was modeled in Simulink, and tested with positive results. The algorithm was used to calculate the frequency of provided test data, and ultimately to carry out volumetric flow calculations on the KL25Z. A top level hardware configuration was designed to support the required functionality of the flowmeter. At max load, the power draw of this design barely reaches beyond the maximum *average* power draw listed in the requirements section of [1], and thus the average power of the system easily sits within an acceptable range.

Based on the requirements of the system, coupled with the capabilities of the Kinetis KL25Z microcontroller, it has been decided that the design is feasible for the Vortex Flowmeter, and thus the recommendation is a **GO**.

1 Problem Statement and Objectives

1.1 Problem Statement

Sierra Instrumentation requested a proof of concept design for the proposed Sierra 240 Vortex Flowmeter [1]. The work required includes evaluation and testing of the NXP/Freescale FRDM-KL25Z evaluation board. The Kinetisis KL25Z will be evaluated based on requirements given in Section 3 of [1] and a GO/NoGO recommendation will be made.

1.2 Objectives

- Evaluate the Kinetis KL25Z based on the requirements given in [1], using the FRDM-KL25V evaluation board and the information and guidance given in [2].
- Design a Simulink diagram and C code which supports the functionality of the Vortex Flowmeter.
- Design a top-level hardware implementation of the flowmeter using the KL25Z.
- Provide a GO/NoGO recommendation based on test results.

2 Approach and Methodology of Evaluation

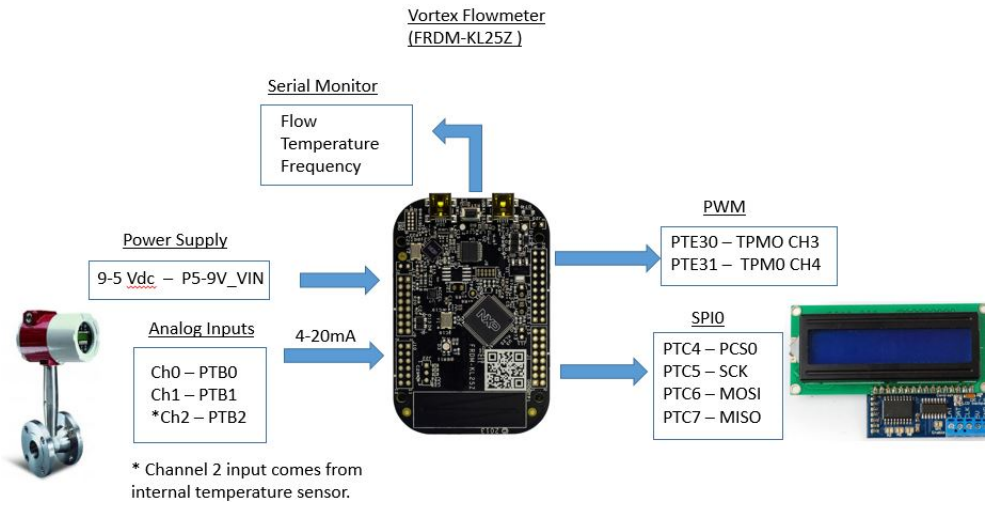


Figure 1: System Block Diagram.

2.1 Block Diagram

The system block diagram in Figure 1 is composed of the following main components: FRDM-KL25Z, Sierra 240 Vortex Flowmeter, power supply, and an LCD display. The diagram below depicts the flow of signal and pinout.

2.2 Hardware Evaluation

The FRDM-KL25Z has an operating range between 1.7V to 3.6V and multiple analog and PWM pins [3]. The input power is provided by an onboard voltage regulator [3], which has a max output rating of 3.3Vdc @ 1Amp (3.3 Watts). However, due to the ability of the FRDM-KL25Z board to operate in low power mode between when readings and calculations are performed, the power consumption will be much lower on average. The table in Figure 2 represents the major components being powered by the FRDM-KL25Z internal voltage regulator.

Figure 2: FRDM-KL25Z Power Consumption

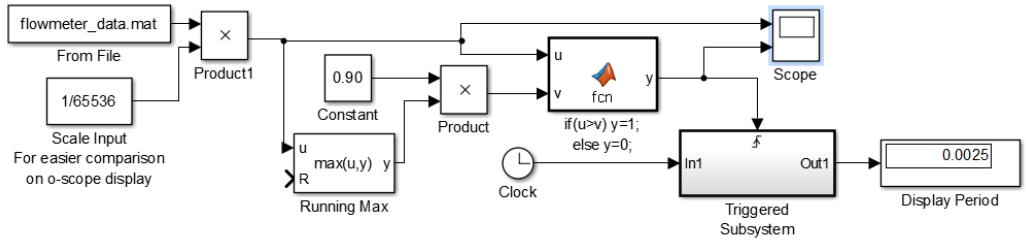
MainComponents	*MaxOutput*	*InputVoltage*	*Power*
Operating Behavior	7.1mA	3.3Vdc	23.43mW
Total all pins [1]	100mA	3.3Vdc	330mW
Oscillator	4mA	3.3Vdc	13.2mW
ADC	1.7mA	3.3Vdc	5.61mW
12bit DAC	0.9mA	3.3Vdc	2.97mW
USB [2]	120mA	3.3Vdc	396mW
TSI Sensor	0.1mA	3.3Vdc	0.33mW
LCD Display	1mA	3.3Vdc	3.3mW

1 Not all pins are active and therefore this value is very conservative.
2 USB UART communication will be available but not used during normal operation, therefore we can ignore this load.

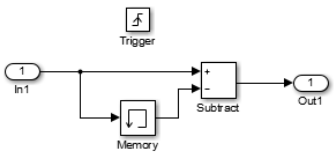
The total load for the KL25Z without considering the 'all pins high' state is 48.84mW. Since we know that not all the pins are going to be high, we could assume 20% of the 'all pins high' load will be active (330*0.2) for a total of 114.84mW of total power consumption. For conservatism, low power mode load consumption for this board was not used in Figure 2 which will take the power consumption below the requested 100mW average limit.

2.3 Algorithm Evaluation

A Simulink model was created to model the algorithm used to detect the period of a discretized waveform (Figure 3). The algorithm was tested on a sine wave generator with added white noise, then simulated ADC data was provided by the contacts listed in section 1.2 of [4]. The period was calculated to be 2.5 ms, corresponding to a frequency of 400Hz, which agrees with direct inspection of the data.



(a) Main Simulink Diagram



(b) Triggered Subsystem

Figure 3: Simulink model used to simulate the frequency detection algorithm.

3 Module Test Results

3.1 Module 1: *Assembly Arithmetic*

3.1.1 Description

An ARM assembly function was written to approximate the square root of an integer argument. The function was based off of pseudocode provided in [5], and code provided by the contacts listed in section 1.2 of [4]. The code was written and tested in Keil uVision MDK v5, and thus Module 1 also entailed the setup of Keil uVision MDK for use with the FRDM-KL25Z. The number of CPU cycles was tracked using Keil's built in hardware simulator; total cycles are shown in the internal registers as 'States', and the total execution time as 'Sec' (see Figure 4). The total number of cycles completed before each execution of the function was subtracted from the number of cycles completed when the function returns.

3.1.2 Results

The function was tested with the following values:

2. function returns 1, takes approximately 219 cycles to complete (≈ 4.57 µs @ 48MHz).
4. function returns 2, takes approximately 186 cycles to complete (≈ 3.87 µs @ 48MHz).
22. function returns 4, takes approximately 234 cycles to complete (≈ 4.88 µs @ 48MHz).
121. function returns 11, takes approximately 204 cycles to complete (≈ 4.25 µs @ 48MHz).
The entire program takes 1207 cycles to complete (≈ 25.15 µs @ 48MHz) (see Figure 4g).

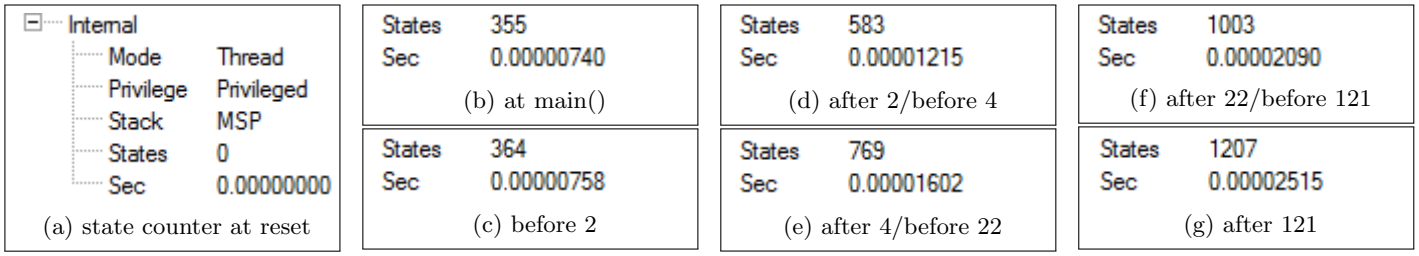


Figure 4: State Counter and Elapsed Time at various points in the program

3.2 Module 2: *Feel the Vibrations*

3.2.1 Description

Using example code provided by Mbed [6], a program was written to read the X, Y, and Z data FRDM-KL25Z's onboard accelerometer and use the values to change the color of the onboard RGB LED. Also, the onboard capacitive touch slider was used as an input to dim the LED. The processor load in % of CPU cycles was estimated by adding a timer to the program; The timer starts before entering the loop, then stops after a number of iterations have been completed. The 100ms delay in each iteration of the loop is assumed to be the idle time; any execution time beyond 1 second per 10 iterations can be used as a representation of CPU load:

$$CPU\% = \frac{t_{total} - t_{expected}}{t_{total}} * 100 \qquad with \qquad t_{expected} = \frac{iterations}{10} seconds$$

3.2.2 Results

The CPU load for this program is approximately 1.677% as shown in Figure 5.

$iterations$	$t_{expected}$	t_{total}	$CPU\%$
100	10 s	10.170614 s	1.67752%
1000	100 s	101.705933 s	1.67732%
10000	1000 s	1017.05957 s	1.67734%

Figure 5: CPU load approximations

3.3 Module 3: *Serial Port Debug Monitor*

3.3.1 Description

Base code given in [4] was modified in order to create a program which provides sensor and debug information to the FRDM-KL25Z's serial port. The following additions were made to the code:

1. Improved user interface by allowing the user to enter lower case letters.
2. Adding ability to toggle the onboard green LED through the serial interface.
3. Reporting the contents of registers r0-r15 and printing the contents of a section of memory to the debug port. 16 memory addresses are shown each iteration, and the addresses cycle from 0x0 to 0x6000.
4. The timer0 interrupt was modified in order to blink the onboard red LED at 1Hz, utilizing a counter (flag) in group IX. This group is called every 6.4ms and the counter was set to increment from 0 to 155 to create the required 1Hz (156*6.4ms = 0.9984s) blink.

The CPU load of the main loop was determined by adding two timers to the program: One which starts and stops at the beginning and end of each iteration of the main loop, and one which tracks the total run time of the program. The execution time divided by the total time is the estimated CPU load. Also, the Dhrystone v2.1 Benchmark program was run on the target processor per [7], using code modified from [8].

3.3.2 Results

Here we answer the questions in the Project Guide [2].

1. timer0 counter is initialized at 0. Group X is active every 51.2ms and this counter is incremented. If the program is run for 30s, the counter would increment to 585 (30/0.0512).
2. The purpose of the Interrupt Service Routine (ISR) is to run a small/fast block of code, then return to the main loop. There are several different groups being called by the ISR at different times but their execution time is minimal compared to the main loop code.
3. All items in the debug window behave as expected. **NORMAL** mode displays flow information, **DEBUG** mode displays the flow information plus the contents of r0-r15 and a section of memory, pressing **L/l** toggles the green LED, pressing **V/v** displays the code version, and **QUIET** mode silences all output. All of the registers are displayed in debug mode, but r13-r15 always show the same values, since it is called from the same function adn executing the same line of code every time those values are read.
4. The new command added to the debug menu is "Hit L - Toggle Green LED". The only function of this command is to change the state of the onboard green LED and communicate back by UART if the green LED is on or off.
5. Having a GPIO turn high at the beginning of the ISR and low before leaving the ISR allows the user to know when the system is in the ISR for debugging purposes.
6. The CPU% of the main cycle was measured to be ≈21%.
7. Dhrystone 2.1 returned results shown in Figure 6, which agree closely with the specs listed at [9]:

```
Microseconds for one run through Dhrystone:   12.1
Dhrystones per Second:                      82470.5
VAX DMIPS:                                    46.9
DMIPS/MHz:                                 0.977880
```

Figure 6: Dhrystone 2.1 Serial Output

3.4 Module 4: Bare Metal Flowmeter Simulation

3.4.1 Description

An algorithm was designed to process data provided by the vortex flowmeter and calculate the fluid flow. Parameters and equations provided in [2] were used, along with simulated ADC data (provided as a separate file) to perform the necessary calculations. Three analog pins were enabled, calibrated, and configured per [2]. The FRDM-KL25Z board communication through SPI with an LCD display was enabled and configured. PWM pin PTE31 generates a pulse based on timer TPM0 channel 4 at the rate of the vortex frequency. Serial communication (UART) is also available with the same information that is provided to the LCD display: flow, temperature, and frequency data.

3.4.2 Results

Here we answer the questions in the Project Guide [2].

1. The frequency estimated is: 400 Hz.
2. The calculated flow is ≈1283 GPM.
3. The temperature starts at 23° C and then random perturbations of ±0.01° C are added to the signal. The signal typically stays in the 20's, but it has the potential to deviate greatly over time.
4. The program takes up over 99% of the CPU cycles. These results were checked using various run times and timer placements, and the results all concur. It is safe to say that the processor is fully utilized for this program.
5. As noted in Section 2.2, our max load is 114.84mW. Under normal conditions, the average power consumption should easily satisfy the 100mW maximum average power requirement.

4 Deliverables

- Code, binaries, zipped mbed/Keil uVision v5 projects, screenshots, and supporting documentation generated by Doxygen for Modules 1-4
- Embedded system design for the Vortex Flowmeter, including Bill of Materials and all other supporting documentation.
- Project Report (this document) formatted as required by [4].

5 Recommendations

The results from Module 1 indicate that a square root approximation takes less than 5 µs to complete; in other words, more than 200,000 of these operations can be carried out per second. As an initial test of the system's capabilities, these results were promising. Also, once up and running, the Keil MDK appears to have several useful built in functions, the debug window possibly being the most valuable. The debug window allows the user to watch higher level variables and objects, as well as hardware registers such as the program counter. The debug window also allows users to set program breakpoints and step through the code line by line.

The program from Module 2 reads 3-axis accelerometer data, reads and computes an analog input, and sets 3 PWM outputs based on floating point computations, hinting that an implementation of the KL25Z in the flowmeter would be feasible, as the input/output requirements have been met. The program was able to update the LED output at a rate of 10 Hz with less than a 2% load on the CPU. This hints that the Kinetis KL25Z should be able to handle the 100ms update rate required in [1] Section 3.3.

Module 3 showed that the serial interface portion of the program would take up approximately 21% of the CPU cycles, which leaves about 79% as overhead to complete the other tasks. Also, the requirements in [1] include an estimated minimum of 40 DMIPS required for processing; Dhrystone testing of the KL25Z showed that the processor does in fact meet this requirement.

Though program memory limitations were encountered while coding Module 4, those limitations were imposed by the linker in the Keil uVision MDK, and thus with a full version of the software that memory constraint would be lifted. However, any additional functionality will require careful optimization since the processor has very little idle time to spare for additional tasks.

Overall, the system meets all of the requirements for the Vortex Flowmeter, and thus it is a feasible selection for the flowmeter's embedded system design. Therefore, the recommendation for the Kinetis KL25Z is a **GO**.

Appendices

A References

[1] Sierra Instrumentaion. *Request for Services*. Version 1.2. 9/18/2017.

[2] Prof. Timothy Scherr. *Project #1 Guide*. ECEN 5308. University of Colorado Boulder. 2018.

[3] Freescale. *FRDM-KL25Z User's Manual*. Freescale Semiconductor, Inc. 2012.

[4] Prof. Timothy Scherr. *Submission Guidelines*. ECEN 5308. University of Colorado Boulder. 2018.

[5] ARM University Program. *Lab Exercise: Square Root Approximation*. ARM Ltd. 2014.

[6] ARM Limited (or its affiliates). *Mbed FRDM KL25Z Examples*. accessed Sept. 2018. URL: `https://os.mbed.com/handbook/mbed-FRDM-KL25Z-Examples`.

[7] ARM Ltd. *Dhrystone Benchmarking for ARM Cortex Processors*. ARM Limited. 2011.

[8] Reinhold P. Weicker. *Dhrystone v2.1 Source Code*. accessed Oct. 2018. URL: `https://github.com/Keith-S-Thompson/dhrystone/tree/master/v2.1`.

[9] ARM Ltd. *Cortex M0+*. accessed Oct. 2018. URL: `https://developer.arm.com/products/processors/cortex-m/cortex-m0-plus`.

B Project Staffing

David Pasley is a PhD student in the Controls, Dynamics and Robotics research group under Dr. Lucy Pao in the Electrical, Computer and Energy Engineering department at the University of Colorado, Boulder. He is also an electronics engineer in the Unmanned Systems branch of NAVSEA, a DoD research organization. His interests are Embedded Systems and Autonomous Robotics.

Ismail Yesildirek is a graduate student in the Electrical, Computer, and Energy (ESE) Engineering department at the University of Colorado, Boulder. He is also an Electrical/Instrumentation and Controls engineer in the nuclear industry. His interests are in Automation and IoT.