Data Recorder for Fan Blade Sensors

Final Project Report
December 9, 1999
E157

Adam Thurston and Justin Pfeiffer

Abstract:

The GM clinic team is purchasing three sensors that will be mounted to fan blades on the engine of a GM Silverado truck. The fan rotates at 1000-5000 rpm, and the clinic team needs some way of recording data taken from these sensors. The data needs to be recorded at 1 KHz, with 1000 samples per sensor, and 12 bits of accuracy. This project involved the development of code for a specialized 68HC11 board to take and store data samples. This project also implements simple external triggering and amplification hardware. The code supports user commands for data retrieval, which can be accomplished by connecting the recorder to a PC via a serial cable.

Introduction

The data recorder consists of four main components:

- The 68HC11 QED Developer's Board
- The light-sensitive trigger
- The amplification circuitry
- The battery power supply

A block diagram of the data recorder is depicted in Figure 1.

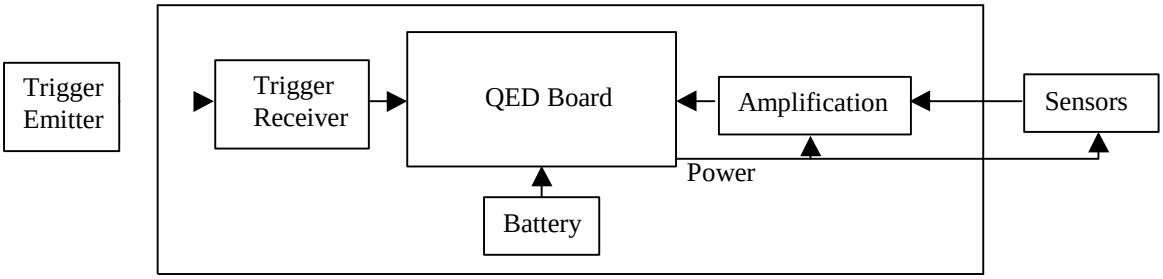

Figure 1. Block Diagram of Data Recorder Components

The heart of the recorder is the QED board, a product of Mosaic Industries, Inc. The QED board's relevant features include a 12-bit A/D converter, 128K user Flash RAM, 128K battery-backed RAM, and support for serial connection. The QED also features a 128K Kernel that includes a debugger, interactive compiler, assembler, and pre-coded device drivers. Programs are written as text files (.txt) in an interpretive language called QED-Forth, that can be sent directly to the board via a serial cable. Extensive hardware and software manuals are available for the board, and are being given to the GM Clinic team. A datasheet for the QED board is included in Appendix B.

The light-sensitive trigger consists of two primary components: an externally operated LED attached to the stationary engine block, and a photodiode attached to the fan blade and connected to the external interrupt (IRQ) pin of the QED board and ground. When the externally operated LED is activated, the photosensitive diode pulls the IRQ pin low, instructing the software to begin taking samples. Because the photosensitive diode will be attached to a blade of the fan, this system allows the GM Clinic team to also control the position that the fan blade is in when the recorder begins taking samples.

The amplification circuitry is necessary to convert to 0-200mV output from the sensors to the 0-5V range required by the A/D converter. It consists of a simple series Op Amps and resistors.

Because the QED board will be attached to the fan, it is necessary that the board be battery powered. The QED board features a DC power regulator that can be used to supply power to the board as well as the sensors and amplification circuitry. The power regulator can accept voltages ranging from 5-15V. We suggest that the power regulator be driven by a package four AA batteries whose nominal voltage is approximately 6V.

Operation

All components of the data recorder, with the exception of the photdiode trigger, will eventually be consolidated into a solid package capable of withstanding the harsh test environment of the GM Silverado engine compartment. The recorder will be attached to the center disc of the fan, and the sensors and photodiode trigger will be attached to positions on the fan blade. The fan will then be rotated at a speed of 1000-5000 rpm.

The GM Clinic team will trigger the sampling of the sensors by activating an external LED mounted to a stationary position in the engine block. The light emitted from the LED will cause the photosensitive diode mounted to the fan blade to conduct, which will pull the IRQ pin on the QED board low. The IRQ pin will cause the QED board to execute an interrupt service routine that will sample the sensors and record the data.

Once data has been acquired, the recorder can be removed from the fan and connected to a PC via a serial cable. The GM Clinic team can run the QED Terminal program, and send the board a command that retrieves the data from a specific sensor. The retrieval can be saved to a text file (.txt) for importation into a program such as Microsoft Excel for processing.

The battery backed RAM will preserve the data if power from the batteries is lost. However, we suggest that power be maintained during the period from when the data was acquired to when it is transferred to the PC. The battery should also be disconnect from the recorder when it is not in use.

Schematics

This is a schematic diagram depicting the components of the data recorder and the sensors.

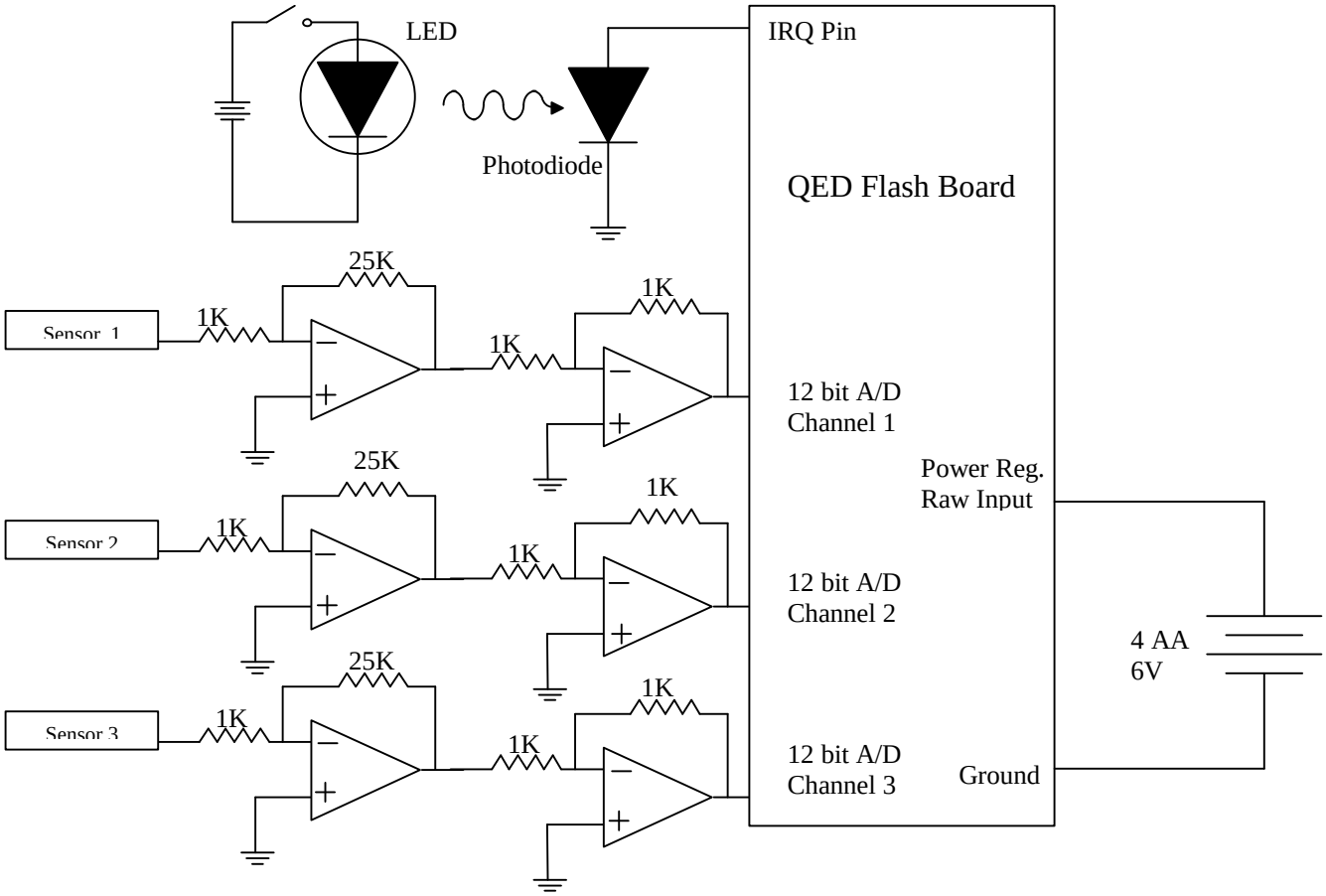

Figure 2. Data Recorder and Sensor Schematic

Microcontroller Design

One of the key advantages of the QED board is its Flash RAM, which preserves the code without the need for power or battery-backing. The process through which is done is described in the actual code, entitled GM_SAMPL.4TH located in Appendix A. Please refer to this code, which also contains detailed comments, for a deeper understanding of its operation. The following is an overview of how this code operates.

The Startup Routine

The startup routine (START.UP, Appendix A: Section 6) is automatically executed on powerup. It's primary functions are to initialize the arrays to be used for data storage, to initialize the A/D converter, and to initialize the location of the heap, where the data arrays are stored. It also attaches an output compare interrupt to the sampling algorithm and attaches the external interrupt request pin to the sampling trigger code.

The Sampling Trigger Code

The sampling trigger code (TRIGGER.SAMPLES, Appendix A: Section 4) is triggered by the external interrupt request (IRQ) pin. It initializes the number of samples that have been taken to zero (actually -1), and enables the output compare interrupt.

The Sampling Routine

The sampling routine (TAKE.SAMPLES, Appendix A: Section 3), after the output compare interrupt has been enabled by the sampling trigger code, is executed every 1ms by adding a constant value to the output compare register. The routine increments the number of samples taken, takes a sample from each of two channels, and then stores these values into separate arrays, using the number of samples taken as the storage index. Data is stored as an integer from 0-4095 corresponding to an input value of 0-5V. Finally, the code tests to see if the number of samples taken equals the total number desired. If so, the output compare interrupt is disabled, so that the recorder stops taking data.

User Interface Commands

The QED board has been programmed to support certain commands for user interface (Appendix A: Section 5). Interface is accomplished by connecting the QED board to a PC via a serial cable and running the QED Terminal program. Currently the QED board supports two commands: PULL.T1 and PULL.T2. These commands translate to "Pull data recorded from Transducer 1" and "Pull data recorded from Transducer 2", respectively. These routines execute a loop that sequentially fetch each value of their respective arrays, perform a floating point conversion to its equivalent voltage, and output the result to the terminal program. Future interface commands may support the retrieval of data taken by additional sensors, and variable settings for sampling frequency and total number of samples taken.

Results

We were successful in meeting all specifications for this project is except for one. The board can be externally triggered, takes and stores 1000 samples, at 1 KHz, with 12-bits of accuracy, and the data can be easily transmitted to a PC for processing. We were unable to meet the 1KHz sampling rate when sampling all three sensors. Currently the design only functions at this speed for two sensors, and it is possible that the GM Clinic team will want to have up to four sensors connected to the data recorder.

The problem essentially lies in the time required to execute the interrupt service routine that samples each device. For 1KHz operation, the device requires that the time for execution be just less the 500 cycles of the free-running counter (1ms). For two devices, we can execute the routine in approximately 440 cycles. For three devices, the routine takes approximately 600 cycles. The limiting instruction appears to be QED Forth's inefficient algorithm for storing the data to an array. We have two proposed solutions to this problem.

The first solution would be to alter the routine that stores the data to an array, so that it employs assembly language and not the QED-Forth interpretive language. This would assuredly make this portion of the code run faster. Whether or not it would save enough time to allow for the sampling of a fourth sensor is questionable.

The second solution is simpler, and involves replacing the QED board's 8 MHz clock with a 16 MHz clock. This would almost certainly allow for the addition of a fourth device with little change to the current code. The ease with which this could be done requires some investigation.

Overall, the GM Clinic team is pleased with our solution. Continued work needs to be done in consolidating all components of the recorder into a package that will withstand the harsh test environment in the engine compartment of the Silverado. We expect to be assisting the GM Clinic team in accomplishing this goal.

References

[1] Getting Started with the QED Board Using Forth Programming Language
[2] QED Software Manual
[3] QED Hardware Manual
[4] The QED-Forth Glossary
[5] The QED-Flash Board Hardware and Software Update Documentation
[6] MC68HC11F1 Technical Data book
[7] Mosaic Industries, `http://www.mosaic-industries.com`

Parts List

Part	**Source**	**Vendor Part #**	**Price**
QED Developer Package	Mosaic Industries, Inc. 510-790-1255	Not Available.	$545.00
4 Batteries	Radio Shack		$5.00
Op Amps	Stockroom		
Photodiode	Stockroom		

```
\ ------------------------------------------------------------------------------
\ APPENDIX A: QED-FORTH CODE FOR DATA RECORDER
\ ------------------------------------------------------------------------------

\ GM_SAMPLE.4TH
\ Created by Justin Pfeiffer and Adam  Thurtson
\ December 7, 1999
\ Questions? Please contact: pfeifdog@hotmail.com or athursto@hmc.edu


\ 1. DESCRIPTION OF CODE ----------------------------------------------------------

\ This program is written in QED-Forth, and can be loaded onto a QED Developer's Board
from Mosaic Industries. It instructs the QED board to sample 2 channels, at 1 KHz, for
1000 samples, upon being triggered by an external device.  The code has been divided into
numbered sections, such as this one, and each one begins with a description of what the
code in that section does.  Each line of Forth is also individually commented.

\ The sections are listed below in the order that they appear:

\ 1. DESCRIPTION OF CODE
\ 2. DECLARATIONS
\ 3. SUBROUTINES
\ 4. TOP LEVEL SAMPLING PROGRAM
\ 5. USER INTERFACE COMMANDS
\ 6. START UP ROUTINE
\ 7. MAKE AUTOSTART AND SAVE TO FLASH

\ NOTE: You will notice that some commands have been commented out.  These commands are
intended either for debugging purposes, or for a future version of this code that will
allow the QED board to take data from a third device.


\ 2. DECLARATIONS -----------------------------------------------------------------

\ These commands are executed upon program loading.  They prepare the QED board for data
transfer, and then initialize all registers, constants, and variables used in the
program.

\ DEBUG ON                                  \ THESE COMMANDS FACILITATE DEBUGGING
\ TRACE ON                                  \ THEY ARE CURRENTLY DISABLED

DOWNLOAD.MAP                                \ CHANGE MEMORY MAP FOR FLASH STORAGE
4 USE.PAGE                                  \ ...AND DEFINES THE MEMORY PAGE TO USE
ANEW SAMPLING.PROGRAM                       \ CREATES A TAG USED WHEN RELOADING PROGRAM

10 WIDTH !                                  \ # OF CHARACTERS OF VARIABLE NAME STORED

HEX                                         \ USE BASE 16

800E    REGISTER:      TCNT                 \ THE 16-BIT FREE RUNNING COUNTER
8016    REGISTER:      TOC1                 \ OUTPUT COMPARE REGISTER #1
8022    REGISTER:      TMSK1                \ TIMER INTERRUPT MASK REGISTER #1
8023    REGISTER:      TFLG1                \ TIMER INTERRUPT FLAG REGISTER #1
803C    REGISTER:      HIPRIO               \ DETERMINES HIGHEST PRIORITY INTERRUPT

80      CONSTANT       OC1.MASK             \ ISOLATES OC1 INTERRUPT FLAG AND MASK BITS
0000 08 XCONSTANT      BOTTOM.OF.HEAP       \ DEFINE MEMORY LOCATION OF BOTTOM OF HEAP
7FFF 08 XCONSTANT      TOP.OF.HEAP          \ DEFINE MEMORY LOCATION OF TOP OF HEAP

DECIMAL                                     \ USE BASE 10

ARRAY: T1.ARRAY                             \ DEFINE DATA ARRAYS
ARRAY: T2.ARRAY
\ ARRAY: S1.ARRAY

VARIABLE SAMPLES                            \ DEFINE VARIABLE USED TO COUNT SAMPLES

\ XVARIABLE START.TIME                      \ THESE COMMANDS ARE USED IN DEBUGGING
\ XVARIABLE STOP.TIME                       \ THEY ARE CURRENTLY DISABLED
\ XVARIABLE FLAG.1
```

```
\ XVARIABLE FLAG.2
\ XVARIABLE FLAG.3

500     CONSTANT       1MS                    \ 500 COUNTS OF 2us COUNTER = 1MS
1000    CONSTANT       NSAMPLES               \ NUMBER OF SAMPLES TO BE TAKEN PER SENSOR
5.0 4096.0 F/ FCONSTANT SAMPLE>VOLTS.FACTOR   \ CREATES A FLOATING POINT CONVERSION FACTOR


\ 3. SUBROUTINES ----------------------------------------------------------------

\ These are subroutines called by either the startup routine (START.UP) or the top level
sampling program (TRIGGER.SAMPLES).

: INIT.DATA.MATRICES ( -- )
\ This subroutine dimensions the arrays.  They contain NSAMPLES rows and 1 column, each
holding 2 bytes of data.

       NSAMPLES 1 2 ' T1.ARRAY DIMENSIONED
       NSAMPLES 1 2 ' T2.ARRAY DIMENSIONED
       \ NSAMPLES 1 2 ' S1.ARRAY DIMENSIONED
;


: SAMPLE>VOLTS ( A/D.SAMPLE--VOLTS )
\ This subroutine accepts a given value and produces its  floating point equivalent.

       FLOT                                   \ CONVERT VALUE TO FLOATING POINT
       SAMPLE>VOLTS.FACTOR F*                 \ MULTIPLY BY CONVERSION FACTOR
;


: TAKE.SAMPLES ( -- )
\ This is the interrupt service routine triggered by output comparator one (TOC1), and is
enabled by the top level sampling program (TRIGGER.SAMPLES).

       \ TCNT X@ START.TIME X!                \ THESE TYPE OF COMMENTS ARE FOR DEBUGGING

       1 SAMPLES +!                           \ INCREMENT SAMPLES
       OC1.MASK TFLG1 C!                      \ CLEAR OC1 INTERRUPT FLAG
       1MS TOC1 +!                            \ UPDATE TOC1 FOR NEXT INTERRUPT

       \ TCNT X@ FLAG.1 X!

       -1 0 (A/D12.SAMPLE)                    \ PUT CHANNEL 0 VALUE ON STACK
       -1 1 (A/D12.SAMPLE)                    \ ...AND CHANNEL 1
       \ -1 2 (A/D12.SAMPLE)                  \ ...AND CHANNEL 2

       \ TCNT X@ FLAG.2 X!

       \ SAMPLES @ S1.ARRAY !
       SAMPLES @ T2.ARRAY !
       SAMPLES @ T1.ARRAY !

       \ TCNT X@ FLAG.3 X!

       SAMPLES @ NSAMPLES -1 + =              \ IF SAMPLES = TOTAL SAMPLES

       \ TCNT X@ STOP.TIME X!

       IF
              ." SAMPLING DONE! " CR
              OC1.MASK TMSK1 CLEAR.BITS       \ DISABLE TAKE.SAMPLES OC1 ISR
              \ START.TIME X@ .
              \ FLAG.1 X@ .
              \ FLAG.2 X@ .
              \ FLAG.3 X@ .
              \ STOP.TIME X@ .
       ENDIF
;
```

```
\ 4. TOP LEVEL SAMPLING PROGRAM -------------------------------------------------------

: TRIGGER.SAMPLES ( -- )
\ This is the top level sampling program, which is triggered by the external interrupt
request pin (IRQ).  It enables the interrupt service routine that takes data
(TAKE.SAMPLES).
        ." SAMPLING TRIGGERED... "
        -1 SAMPLES !                        \ INITIALIZE SAMPLES TAKEN
        TCNT X@ TOC1 !                      \ INITIALIZE OUTPUT COMPARATOR
        OC1.MASK TMSK1 SET.BITS             \ ENABLE OC1 INTERRUPT
;


\ 5. USER INTERFACE COMMANDS ----------------------------------------------------------

\ These are user commands for retrieving data and modifying parameters when the QED board
attached to a PC running QED Terminal.

: PULL.T1 ( -- )
\ This command sequentially converts and outputs each entry of the T1.ARRAY data array;

        NSAMPLES 0 DO
                I T1.ARRAY @ SAMPLE>VOLTS . CR \ CONVERTS DATA AND PRINTS TO TERMINAL
        LOOP
;


: PULL.T2 ( -- )
\ This command sequentially converts and outputs each entry of the T2.ARRAY data array;
        NSAMPLES 0 DO
                I T2.ARRAY @ SAMPLE>VOLTS . CR \ CONVERTS DATA AND PRINTS TO TERMINAL
        LOOP
;


\ : PULL.S1 ( -- )
\ This command sequentially converts and outputs each entry of the s1 .ARRAY data array;
\       NSAMPLES 0 DO
\               I S1.ARRAY @ SAMPLE>VOLTS . CR \ CONVERTS DATA AND PRINTS TO TERMINAL
\       LOOP
\ ;


\ 6. START UP ROUTINE -----------------------------------------------------------------

: START.UP
\ This routine is exectued upon power up, or upon typing "COLD" (for  Coldstart) when the
QED board is connected to a PC running QED Terminal.  It initializes memory for data
storage, data structures, the 12-bit A/D converter, and interrupt parameters.

        CR CR ." Welcome GM Clinic Team " CR  \ PRINT STARTUP MESSAGE
        CR
        ." User Commands: " CR
        ." PULL.T1" CR
        ." PULL.T2" CR
        \ ." PULL.S1" CR

        BOTTOM.OF.HEAP TOP.OF.HEAP IS.HEAP    \ INITIALIZE HEAP
        INIT.DATA.MATRICES                    \ INITIALIZE MATRICES
        INIT.A/D12&DAC                        \ INITIALIZE 12-BIT A/D CONVERTER

        [ LATEST ] 2LITERAL VFORTH X!         \ PRESERVE ACCESS TO NAME LIST
        15 HIPRIO CLEAR.BITS                  \ CLEAR INTERRUPT PRIORITY BITS
        11 HIPRIO SET.BITS                    \ RAISE OC1 TO HIGHEST PRIORITY INTERRUPT
        CFA.FOR TRIGGER.SAMPLES IRQ.ID ATTACH \ ATTACH TRIGGER.SAMPLES TO THE IRQ PIN
        CFA.FOR TAKE.SAMPLES OC1.ID ATTACH    \ ATTACH TAKE.SAMPLES TO OUTPUT COMPARE 1
        ENABLE.INTERRUPTS                     \ ENABLES INTERRUPTS
;


\ 7. MAKE AUTOSTART AND SAVE PROGRAM TO FLASH -----------------------------------------
```

```
\ These commands are executed upon program loading.  They make START.UP the  autostart
routine and save the program to FLASH RAM.

CFA.FOR START.UP PRIORITY.AUTOSTART          \ MAKE START.UP THE AUTOSTART ROUTINE
4 PAGE.TO.FLASH                              \ SAVE PROGRAM TO FLASH RAM
STANDARD.MAP                                 \ RETURN TO NORMAL MEMORY MAP
SAVE                                         \ PRESERVE NAME LIST IN CASE OF CRASH
```

Appendix B: Please refer to http://www.mosaic-industries.com/qed.html