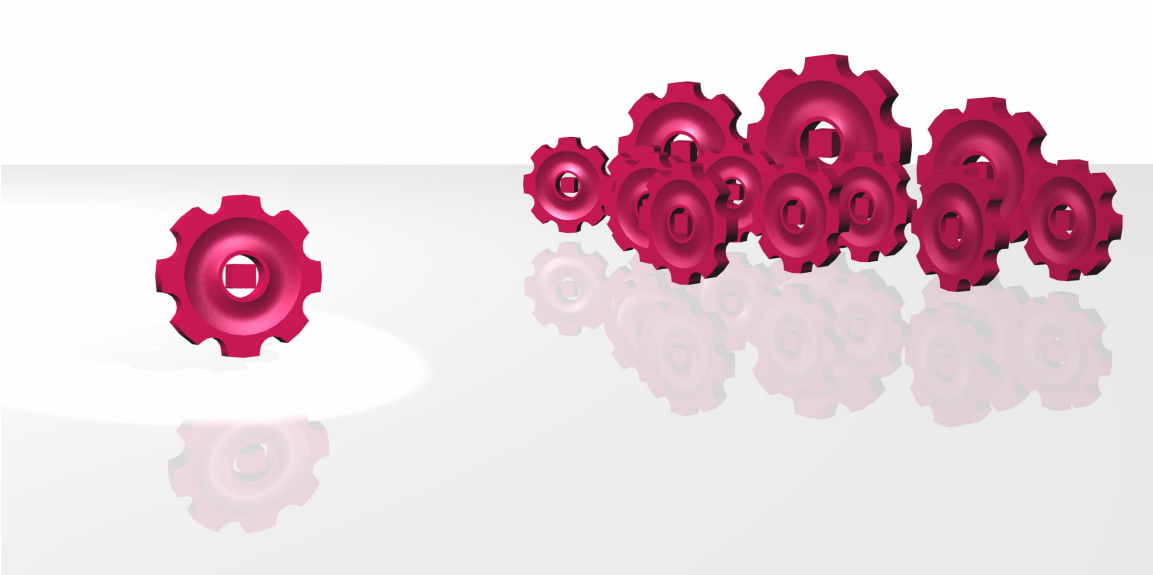
Welcome to the family...



Advance Technical Introduction

C6808 Code Development System for Freescale™ RS08

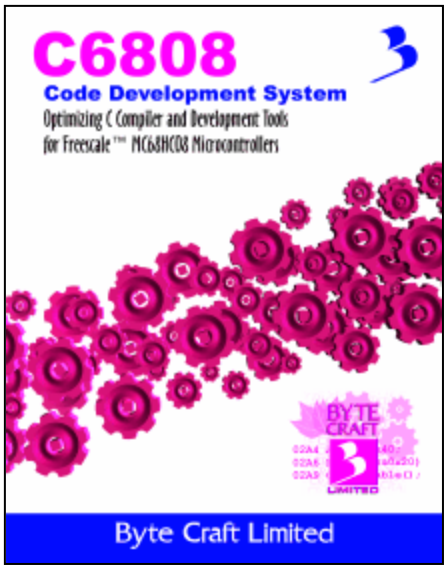
Byte Craft Limited is pleased to announce the only C compiler for Freescale's RS08 architecture. With C6808, you can program RS08 completely in C. You can also migrate existing HC08 programs and other C software to RS08 with ease.

C6808 is shipping now; RS08 support will be released at the same time as the first RS08 parts become available.

Read on for details about RS08 support, reference designs, and more information about Byte Craft Limited...

C6808

Byte Craft Limited's C6808 Code Development System includes an **optimizing C compiler**, optimizing linker, BCLIDE integrated development environment, full documentation, and one year's full technical support.



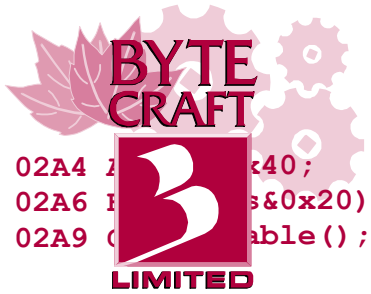
C6808 draws on Byte Craft Limited's experience and expertise in a wide variety of embedded architectures. Translating your programs to different instruction sets is our main business, and we have in-depth understanding of the strengths of modern processors.

Supports all HC08, HCS08, and RS08 devices

Select or change the target device by including a simple header file; **code generation changes automatically**. All library code is shipped with C sources.

Memory Management

C6808 includes enhanced memory management such as **named memory directives**. Local address space directives allows the user to maximize the use of RAM, direct the placement of local variables, re-use RAM locations and pass multiple arguments to functions. Other named memory directives add support for variables, allocated externally or internally, that are managed through software drivers.



Byte Craft Limited
A2-490 Dutton Drive
Waterloo, Ontario
Canada – N2L 6H7

+1 519.888.6911
sales@bytecraft.com

Features in Detail

- **Highly-optimized generated code**. Full versions generate ROMable code; demonstration versions generate listing files with assembly.
- Ports and other resources are declared and protected using `#pragma` directives.
- **Named address** spaces support the grouping of variables at specific memory locations.
- C data types include register-oriented types for direct access to processor registers when necessary.
- Data structures of **arbitrary complexity**, including packed bit fields in structures.

Problem Space, Solution Space

Programming in C is working in the solution space, not the problem space. You can implement an algorithm in C and pay attention to the correctness of your calculations, free from preoccupations about registers, pages, and the subtle differences between addressing modes. The only limit on complexity is the available resources of the processor.

The greater breadth of expression available in C permits you to design programs that are meaningful to you, but still compile to valid, optimized executables.

Data Types

Our research has told us that choosing the right data types for the application is key to a successful embedded project. C offers numerous specialized data types, and the coding environment in which to use them. The only limit is the hardware resources of your chosen part.

Type Sizes: C6808 offers signed and unsigned integer types in 8, 16, 24, and 32 bits. Choose data types to suit the solution of your problem, and let the compiler worry about the carry bit.

Fixed point: C6808 supports TR 18037 fixed point and accumulator types. _Fract types represent fractional values between 0 and 1, and _Accum types have small integer components.

Type	Bits
`unsigned short _Fract`	8
`unsigned _Fract`	16
`unsigned long _Fract`	24
`signed short _Fract`	s.7
`signed _Fract`	s.15
`signed long _Fract`	s.23
`unsigned short _Accum`	8.8
`unsigned _Accum`	8.16
`unsigned long _Accum`	8.24
`signed short _Accum`	s8.7
`signed _Accum`	s8.15
`signed long _Accum`	s8.23

Lucky Numbers

How will your math calculations change due to the instruction set differences in RS08?

C6808 uses optimization strategies like shifted number systems to implement various data types and operations in RS08.

We worry about math problems so you can focus on your solutions.

Fixed point math is implemented by an external library, with accompanying source code.

Unambiguous sizes: C6808 offers ISO types that make it clear what values can be expected.

`uint8_t`	`uint16_t`	`uint24_t`	`uint32_t`
`int8_t`	`int16_t`	`int24_t`	`int32_t`

_Bool: C6808 supports `_Bool`, using bit test and set instructions wherever possible. The deprecated Byte Craft Limited `bit` type is still supported.

bits: Code Development Systems have always offered the `bits` type, a structure of eight bits that are individually addressable (with constant members).

Derived Types and Complexity

C6808 supports single-dimension **arrays, structures, unions**, and any combination of them. This permits data to reflect the solution, not the hardware. The only limit placed on this complexity is the available resources on the target part.

C6808 also supports **bit fields within structures**. This particularly complex code generation and optimization problem can help conserve resources in certain circumstances.

Named Memory

Where the problem space forces you to dwell on addressing modes, the solution space offers **named memory spaces** to organize your variable allocations. This separates low-level details about variable access from the high-level design of the program.

Header file declarations, both standard and user-supplied, organize memory into named sections. They can represent scratchpad RAM versus persistent data, or read-only resources separate from the program and its constant objects. They can even be used to issue warnings when one class of objects exceeds memory budgets, though others do not.

Named memory can be more than just storage. C6808 offers user-implemented memory declarations: accesses to variables declared in these areas are implemented using device driver routines. In the past, this type of memory has been used for serial bus communications, Flash programming, and other driver-oriented applications.

Here's an example of accessing variables declared on an external SPI EEPROM device:

```
1000 0100                          #pragma memory RAM _Access bySPI [0x100] @ 0x1000,
ReadSPI, WriteSPI;

                                   void SELECT_SPI_EEPROM(void);
                                   void DESELECT_SPI_EEPROM(void);
                                   void WRITE_SPI_ADDR(unsigned int addr);
                                   void WRITE_SPI_DATA(char in);
                                   char READ_SPI_DATA(void);



1000                               bySPI char persist_count;
0020                               char count;

                                   char ReadSPI(unsigned int Addr)
0013                               {
3800 EA         STA   $0A
0009                                char temp;

3801 AD 24      BSR   $3827         SELECT_SPI_EEPROM();
3803 CA         LDA   $0A           WRITE_SPI_ADDR(Addr);
3804 AD 31      BSR   $3837
3806 AD 45      BSR   $384D         temp = READ_SPI_DATA();
3808 E9         STA   $09
3809 AD 26      BSR   $3831         DESELECT_SPI_EEPROM();
```

```
380B C9         LDA   $09             return temp;
380C BE         RTS
                                    }

                                    void WriteSPI(unsigned char Data, unsigned int Addr)
0013 0012 0011                      {
380D 4E 12 0F   LDX   $12
3810 F1         STA   $11
3811 AD 14      BSR   $3827           SELECT_SPI_EEPROM();
3813 D1         LDA   $11             WRITE_SPI_ADDR(Addr);
3814 AD 21      BSR   $3837
3816 D2         LDA   $12             WRITE_SPI_DATA(Data);
3817 AD 29      BSR   $3842
3819 30 16      BRA   $3831           DESELECT_SPI_EEPROM();
                                    }

                                    void main(void)
                                    {
381B B6 00      LDA   $00               count = persist_count;
381D AD E1      BSR   $3800
381F B7 20      STA   $20

                                        /* process */

3821 B6 00      LDX   $00               persist_count = count;
3823 B6 20      LDA   $20
3825 30 E6      BRA   $380D
                                    }
```

Combining the user's logical divisions of memory with the different types of physical memory is exactly the type of work most successfully accomplished by an optimizing compiler.

Hardware Access

For those times when hardware access is unavoidable, C6808 brings these details into the C language and into the solution space. *There is nothing you can do in assembly that can't be done in C.*

Register access: Following on from named memory, C6808 offers direct access to processor registers for specialized purposes. Reading and writing variables of (storage class) type `register _A` and `register _CCR` generates stores and loads as appropriate. This applies even to registers that can't be directly accessed through the instruction set, another benefit of C.

Device-specific instructions: These are brought out as C intrinsics:

`WAIT();`	`NOP();`	`SEC();`	`BKGD();`	`ROLA();`
`STOP();`		`CLC();`		

Reset code: The state of the processor is generated automatically. Variables are initialized, and a user-defined `__STARTUP()` routine is executed if defined. `__STARTUP()` can serve as a debugging-specific shim, compiled in for testing and then simply left out for production.

• • •

Why C?

Programming a desktop computer and programming a miniscule embedded processor have a lot in common. Though the resources differ, the underlying model of computing is the same. It makes perfect sense to develop for two computers in the same language, even if the applications are completely different:

- Computers are very good at accounting. A compiler can track actual demand on resources far more closely than a programmer doing so manually. The accounting is redone each time a program is compiled, and all code creation decisions, based on the metrics obtained, are revisited at each compile.
- Compilers use the entire instruction set. The complete list of capabilities and constraints of the instruction set and architecture are embedded in the compiler. As mentioned before, developers are left free to think in the solution space.
- "Programmers' tricks", once added to the knowledge base in the compiler, remain available for new applications forever. Furthermore, the compiler is capable of recombining these techniques into expressions not previously anticipated.

  Optimization strategies that involve data combinations, instruction sequences, or placement-specific code, are extremely hard for assembly language programmers to implement. These optimizations are easy for compilers to identify and integrate into the program.

Quick: what's one thing that RS08 does, that HC08 and HCS08 don't do?

Processor architecture is our specialty. We ask questions like this to investigate the best ways of optimizing code.

The answer?

RS08 can perform single-bit set and clear anywhere in memory—by virtue of its paging mechanism. This feature can make it easier to create and manage Boolean values and structure bit fields, especially when combined with page selection optimization...

It's not just memory use that's at stake.

Optimization reduces one or more of: ROM usage, RAM usage, or application execution time.

- Compression trades off increases and reductions in ROM, RAM and execution time.
- Optimization allows more application code to be run in the same-sized target processor. Optimization reduces execution time for an application, or allows more code to be executed in the same time. Finally, optimization allows clock speeds to be reduced, in turn reducing EMI and power consumption.

• • •

Why C for RS08?

Here are two examples of the effectiveness of C in the RS08:

- Multiple memory spaces. Moving a variable to another data space changes the type of access and instructions available to reference the data. An optimizing compiler can manage the consequences throughout the program of moving a variable.
- RS08 literal data access, using physical address space, requires 16 to 23 bytes to access a random memory location, owing to the use of the PAGE, X and D[X] registers. High-level language tools can create and maintain a virtual address space on the RS08 that will reduce the access to either 7 or 10 bytes. This reduction is transparent to the programmer.

Compiler technology is especially effective for processors with instruction sets optimized for generated code and processors with pared-down resources. The RS08 core employs both these strategies.

Management of rare resources on the RS08 includes:

- Managing multiple address spaces and address specific instructions.
- Reusing RAM as much as possible.
- Eliminating unnecessary PAGE, X, and AC operations.
- Compiling or eliminating program stack accesses. There are numerous optimizations that involve the Shadow Program Counter in flow control, not all of which are obvious or mutually-compatible.
- Tracking condition code register bits. Testing and setting CC bits requires indirect operations; if the compiler can eliminate this at compile time, so much the better.
- Tracking and overlaying TINY and SHORT address space contents, and reallocating variables between the two spaces depending on operations performed on them.

Insider Knowledge

Our experience tells us that hand-written assembly can reach a 16-to-1 ratio of program code to RAM. For every new RAM variable, typically 16 bytes of program store are needed. Compilers' superior memory accounting can generally equal or beat the best handwritten assembly ROM to RAM ratios.

The MC9RS08KA2 has a ROM to RAM ratio of 32. With 32 bytes ROM for every byte of RAM, applications on this processor will benefit from every optimization possible to fit their data into the RAM available.

From Interrupts to Threads

RS08 has the potential to change interrupt programming significantly. Its architecture encourages **event-driven programming** and **run-to-completion routines**.

Even though there are no traditional dispatched interrupts, we've found that RS08's architecture saves on program complexity. The time and memory saved through eliminating context switching more than compensates for the occasional delay in servicing an interrupt.

We've designed two different ways of responding to asynchronous events in C in RS08. Consider them both for your project: the first is simple and traditional, but the second is far more powerful.

Event-driven Programming

Traditional interrupts divert the processor's attention to an asynchronous task, and restore it after the interrupt service is complete. HC08 and HCS08 followed on in the HC05 practice of pushing the entire processor status on the stack during interrupt servicing. The biggest cost of this is the amount of time and resources needed to support this operation. But is it really necessary?

We've dealt with a variety of architectures, some of which are run-to-completion—no asynchronous events disturb the processor between logical tasks. Experience has shown us that run-to-completion, while risking delays in response, helps eliminate:

- Expensive preservation of processor state during context switches.
- Complexity in managing local variable values.
- Complexity in interrupting and restarting main line software.

Event-driven programming, where execution takes place only in response to external stimuli and is usually run-to-completion, is a viable alternative.

RS08-specific Event-driven Programming

This method is RS08-specific. It is also the simplest, but not necessarily the most elegant. It simply performs a `WAIT()`, and then tests possible sources of interrupt. Interrupt priority is determined by control flow.

Within the main control loop, software performs a `WAIT()` or `STOP()` instruction. When the processor wakes due to a hardware interrupt, code must test all relevant interrupt flags to determine what caused the interrupt, take appropriate action, and loop around to the initial `WAIT()`.

```
void main(void)
{
  while(1)
  {
      WAIT();
      if (KBISC.KBF)
      {
```

```
        // service peripheral's request
        // and acknowledge
        KBISC.KBACK = 1;
      }
      else
      if (MTIMSC.TOF)
      {
       //…and so on.
        MTIMSC.TRST = 1;
      }
  }
}
```

Existing code would require significant rewriting to make use of this method. This is suggested for new, RS08-specific applications only.

Threads

Interrupts on RS08 do not actually interrupt the flow of execution. They may wake the processor from a `WAIT()` or `STOP()` mode, but they can only be handled when software invokes a specific thread of execution.

Byte Craft Limited has designed a threaded programming model that makes event-driven programming in C easy and intuitive. We use the compiler to organize sets of event and state tests in one coherent architecture.

To get the definitions right: a **thread** is a C function with the added semantics of centralized conditional dispatch. The elements of C6808 thread programming are as follows:

- Thread declarations appear as `#pragma` directives. Each thread has its own gating expression that evaluates to a Boolean value. The expression can test interrupt source flags, global variables, or I/O ports.

  `#pragma thread` *`identifier keyword`* `(`*`expression`*`);`

  The keyword is one of `SOFTWARE`, `INTERRUPT`, or `RESET`, indicating whether the dispatch should take place at will, after a `WAIT()`, or during reset.

- C6808 generates a function `__DISPATCH()` that implements the thread dispatch code; the developer calls it at an appropriate point in their kernel.

  In `__DISPATCH()`, when the thread conditions are satisfied, control passes to the matching function. If no threads match, control falls through to the main line code.

Threads grow naturally from the practice of declaring interrupt service routines using `#pragma vector` declarations. Interrupt vectors are a facility of the underlying hardware, where threads are selected by compiler-supplied program code. Ultimately, in RS08 all "asynchronous" code must still be dispatched by program flow. Threads can give the appearance of being selected by hardware, but can in fact be selected by dispatch code that references both hardware and software.

A collection of threads can implement a complex peripheral function in an intuitive and straightforward fashion. See the quadrature decoder described later.

Threads and Reset

We've extended this model from event-driven main line code to make it available for reset code as well.

It occurred to us that, in fact, there is one dispatched, high-priority interrupt in RS08: the RESET vector. The number of sources of reset, and the fact that most of them do not alter the internal state of the processor, will encourage designers to take advantage of them for more than just starting the chip. We've added thread support to the RESET code generated by C6808.

By default, C6808 generates code to handle reset as follows:

- `__STARTUP()`, a user-supplied configuration routine, is run. This function can be conditionally compiled in to the program, allowing for debugging or other optional configuration code to be run.
- Reset threads are tested and dispatched. C6808 can dispatch threads based on the flags in the SRS register, combined with any other condition (perhaps established by `__STARTUP()`).
- Normal variable initialization. If any variables are initialized, C6808 zeros out RAM and sets initial values.
- Finally, `main()` is run.

Evaluation

All thread expressions are evaluated within `__DISPATCH()` in declaration order (or during the equivalent phase of RESET). When any expression is completely evaluated, and evaluates to `true`, that thread is run immediately. If no expressions are true, execution falls through to the next main line program instruction after the call to `__DISPATCH()`.

Examples

Real Time Interrupt

Implementing a real time interrupt for a real-time kernel is a matter of choosing the right expressions for `#pragma thread` directives:

```
unsigned int count = 0;

#define SW_PRESCALE a_value

#pragma thread __TIMRESET INTERRUPT
       (SRTISC.RTIF == 1 && count == SW_PRESCALE);

#pragma thread __TIM INTERRUPT
       (SRTISC.RTIF == 1);

void __TIM(void)
{
  count++;
}
```

```
void __TIMRESET(void)
{
  count = 0;
  SRTISC.RTIACK = 1;
}

void main(void)
{
  while(1)
    __DISPATCH();
}
```

Two sources

This example is comparable to the RS08-specific design described at the beginning of the article.

```
#pragma thread KBI wait (KBISC.KBF == 1);

#pragma thread MTIM wait (MTIMSC.TOF == 1);

/* interrupt functions */

void KBI(void)
{
  // actions
  KBISC.KBACK = 1;
}

void MTIM(void)
{
  // actions
  MTIMSC.TRST = 1;
}

void main(void)
{
  while(1) __DISPATCH();
}
```

In this example, only the KBI and MTIM routines will be run, and only after a `WAIT()`.

Future Directions

Threads and Libraries

One novel use of threads is to distribute interrupt service across several libraries. Each library that requires event servicing can declare its own `#pragma` thread condition, complete with macros as expression terms. The compiler will integrate the expressions and thread calls during final compilation.

Threads and Vectored Interrupts

Thread-based design can be integrated with vectored interrupts: a vectored interrupt can perform base-level tasks such as setting one or more of the flags or variables to be tested during thread dispatch.

The benefit: interrupt servicing on different platforms takes a varying amount of time to preserve and restore main-line code state. Using the simplest possible vectored interrupt to set the environment for threads can help minimize the overhead required.

Quadrature Decoder

This example demonstrates a quadrature decoder.

The RS08's keyboard interrupt has options to raise an interrupt on low-to-high or high-to-low transitions. Therefore, we can use the RS08 `WAIT()` to suspend the processor until a transition appears on one of the input sensors. This design is ideal because it integrates with other interrupt tasks, and can ultimately lower the power consumption of the processor during decoding.

The quadrature decoder routine checks the state of the input pins, and inverts them to determine the next edges to look for. It implements a state table to decode whether the input quadrature signal is moving forward or backward.

```
                                    void QDecode (void)
                                      {
004F 004E                               char save_port,state;
3800 4E 10 4F   MOV   $10,$4F           save_port = PORTA;
3803 DE         LDA   $1E               state = ((KBIES << 2) & 0x0c);
3804 48         LSLA
3805 48         LSLA
3806 A4 0C      AND   #$0C
3808 B7 4E      STA   $4E
380A B6 4F      LDA   $4F               state = state + (save_port & 0x03) ;  // state
Old + new value complemented
380C A4 03      AND   #$03
380E BB 4E      ADD   $4E
3810 B7 4E      STA   $4E

                                        if      ((state == 0b1101) ||
                                                 (state == 0b1011) ||
3812 A1 0D      CMP   #$0D                       (state == 0b0010) ||
3814 37 0C      BEQ   $3822
3816 A1 0B      CMP   #$0B                       (state == 0b0100))     tally++;
3818 37 08      BEQ   $3822
381A A1 02      CMP   #$02
381C 37 04      BEQ   $3822
381E A1 04      CMP   #$04
3820 36 08      BNE   $382A
3822 3C 21      INC   $21
3824 36 02      BNE   $3828
3826 3C 20      INC   $20
3828 30 19      BRA   $3843             else if ((state == 0b1110) ||
                                                 (state == 0b1000) ||
382A A1 0E      CMP   #$0E                       (state == 0b0001) ||
382C 37 0C      BEQ   $383A
382E A1 08      CMP   #$08                       (state == 0b0111))     tally--;
3830 37 08      BEQ   $383A
3832 A1 01      CMP   #$01
3834 37 04      BEQ   $383A
3836 A1 07      CMP   #$07
3838 36 09      BNE   $3843
383A 4E 21 21   TST   $21
383D 36 02      BNE   $3841
```

```
383F 3A 20      DEC   $20
3841 3A 21      DEC   $21

3843 B6 4F      LDA   $4F               KBIES = ~ save_port;
3845 43         COMA
3846 FE         STA   $1E
3847 14 1C      BSET  2,$1C             KBISC.KBACK = 1;
3849 BE         RTS                   }
```

This example also demonstrates a threaded software architecture.

Look at the complete source and generated listing files below:

Quadrature Decoder C Source

Quadrature Decoder Listing File

Intelligent A/D with RS08 and C6808

Introduction

This application note describes an interesting implementation of A/D conversion using the RS08 microcontroller from Freescale and the C6808 Code Development System from Byte Craft Limited.

The RS08 part does not include an A/D converter, but with a little bit of external hardware and programming, an RS08 can perform A/D conversion in engineering units. Using the RS08's comparator and modulo timer, the A/D conversion can be performed as an interrupt task, allowing the part to perform other software tasks at the same time.

Theory

This method of A/D conversion is ratiometric. Software keeps two counts: the number of positive comparator readings and the total number of readings taken. The ratio between the two indicates the input voltage. The faster the A/D task can take place, the more accurate the readings.

The circuit strives to keep the input voltage close to the switching voltage of the comparator. The comparator is a flexible device that can operate independently of the RS08. Though it can interrupt the RS08 on a change in its output, this design simply polls it. It functions as an op-amp that the processor can watch.

The comparator's positive input is set to the bandgap reference voltage source (1.218V). Its negative input is tied to the input voltage through a current source resistor. A small capacitor integrates the error current.

The digital output of the comparator is fed back into the input through another current source. Because the voltage input is fed in through the negating input, the output of the comparator is opposite to the difference between the bandgap input and the voltage under test. This will cause the comparator output to oscillate. In a software-only implementation, the same effect is created by constantly inverting an output pin with respect to the input.

The modulo timer task can take place along with other frequent modulo timer tasks. So long as the comparator is sampled at regular intervals, this conversion is not time-constrained.

Design

This is a schematic for the design using the 6-pin RS08KA2 part:

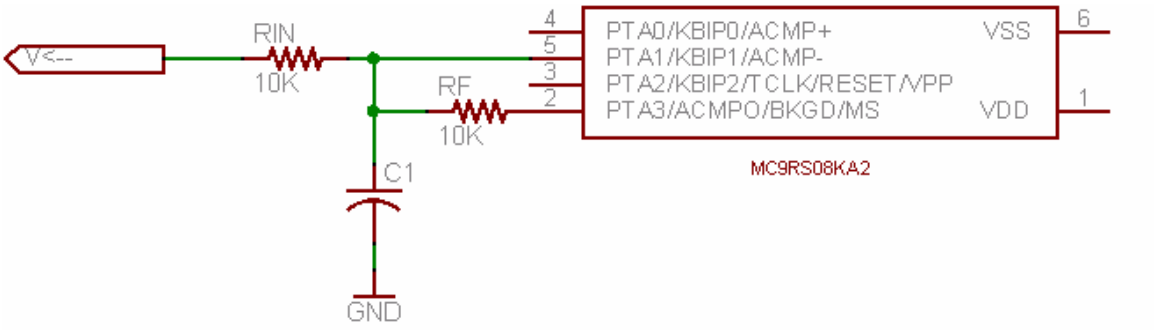


Figure 1: A/D circuit

The values of the resistors are selected to determine the range of input values to be measured. The capacitor is non-critical; use a 0.01 to 0.1 µF capacitor.

With a sense voltage on input of 1.218 volts, Vh of 3.0V and Vl of 0V, the operating range is determined by these equations:

```
Vmin = Vs - ((Vh - Vs) * (Ri / Rf))
Vmax = Vs - ((Vl - Vs) * (Ri / Rf))
```

In software, the comparator is set to run freely (no interrupts on output changes), and an interrupt loop counts the number of iterations and number of positive readings on the comparator. The measured voltage is calculated as follows:

```
Vin = (onecount / totalcount) * (Vmax – Vmin)
```

but calibrating the device allows you to avoid the division.

Software

The required software appears at the end of the document. Some excerpts follow.

Using the RS08's modulo timer, the conversion can take place as an interrupt task, allowing the processor to perform other (interrupt-driven) tasks at the same time.

Configuration

The comparator is configured to run freely, and the modulo timer is set to generate an interrupt.

```
                                    /* configure the comparator */
                                    ACMPSC = 1<<ACME       //enable
                                             |1<<ACBGS     //bandgap select
                                             |1<<ACF       //clear flag
                                             |0<<ACIE      //do not enable interrupt
                                             |1<<ACOPE     //pass output to pin
380C 3E C7 13   MOV   #$C7,$13               |0b11<<ACMOD  //rising or falling edge.
380F                                ;
000C                               #define comparator_hi() (ACMPSC.ACF)

                                    /* configure the modulo timer */
                                    MTIMSC = 1<<TOIE       //modulo timer interrupt
enable
                                             |1<<TRST      //reset the timer
380F 3E 60 18   MOV   #$60,$18               |0<<TSTP      //allow the timer to run
3812                                ;
000D                               #define timer_overflowed() (MTIMSC.TOF)

                                    MTIMCLK = 0b00<<CLKSRC //BUSCLK source
```

```
3812 3F 19      CLR   $19                     |0b0000<<PS  //1:1 prescale
3814                                ;

3814 3E AA 1B   MOV   #$AA,$1B       MTIMMOD = 0xAA;        //modulo value.
```

Event loop

This excerpt shows the while(1) statement that bounds the operating loop and the `WAIT()` that heads it. We used `WAIT()` to allow the modulo timer to continue running. Whenever the modulo clock rolls over, the program increments the total and (if needed) positive counts.

```
while(1)
{

WAIT(); //wait mode allows modulo clock to continue.

/* awaken from interrupt and check sources */
if(timer_overflowed())
 {
 totalcount++;

  if(comparator_hi())
  {
  positivecount++;
```

At the end of the timer interrupt, the interrupt flag is reset and the loop is continued.

Calibration

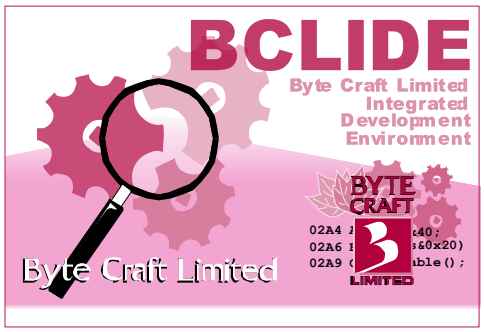
The circuit can give an output in engineering units once calibrated. The sample software incorporates calibration, but here are the steps:

1. Apply the maximum known reference voltage to the input.
2. Signal the software to count samples until the count of positive samples is an engineering unit (i.e., 100, 1000, etc.)
3. Software records the count of total samples needed to reach the proper maximum positive count, and returns to normal operation.
4. Apply a voltage to measure to the input.
5. Software performs a count of total samples matching the recorded value.
6. Use the resulting count of positive samples as an engineering value between zero and the calibration maximum count of positive samples.

If you're tempted to use a low-cost RS08 part in your design, but need full A/D conversion, designing in a software-assisted A/D converter may serve your purposes.

Look at the complete source and generated listing files below:
A/D Converter Source Code
A/D Converter Listing File

BCLIDE



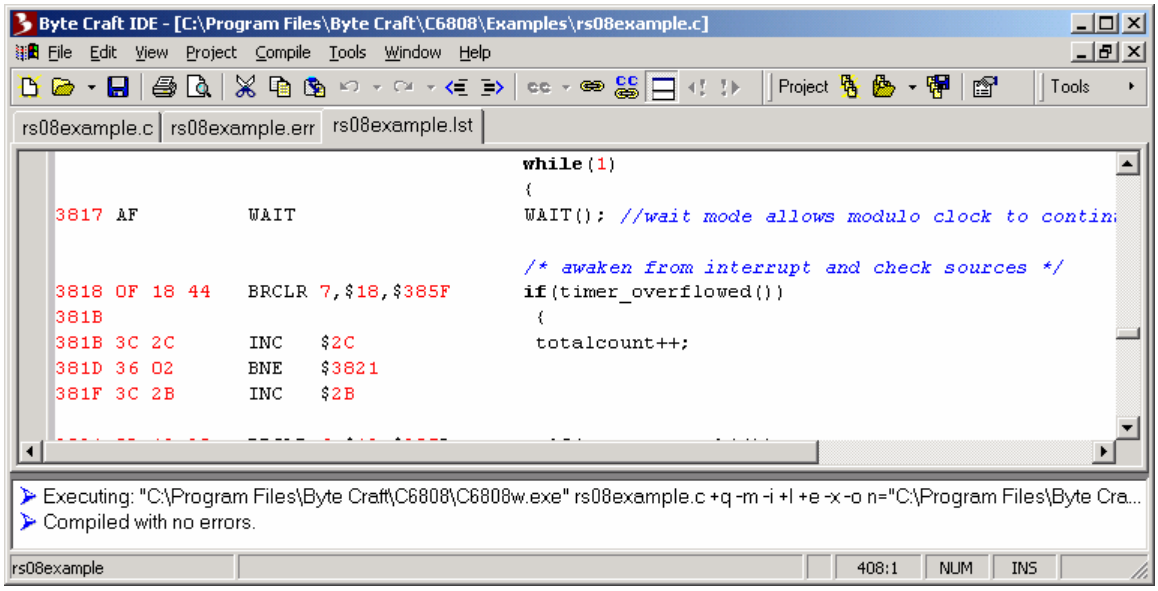
Every Code Development System ships with BCLIDE, a Windows development environment that helps you manage your development projects by keeping all information related to each project together in one place. You can edit, compile and link individual files or the entire project with a few keystrokes. BCLIDE knows the Byte Craft Limited Code Development System; it can generate files for the compiler's use and associate output files generated by the compiler.

You can call any Byte Craft Limited C compiler, the Byte Craft BClink linker, and any other software tools with a single keypress, menu, or toolbar control.

On-line help is available for BCLIDE and the compiler, and you can easily add frequently-used help files to the help menu.

BCLIDE's editor is particularly flexible. You can create code templates for frequently-used structures and bind them to hotkeys. You can describe block boundary character sequences for quick navigation through your code. And you can set and navigate to bookmarks. All these settings are stored with your project for quick recall.

Best of all, BCLIDE is lightweight: startup times are under 3 seconds. Compare this with some other IDEs on the market…



Fuzz-C™

Curious about "that other product" that Byte Craft Limited offers? Have trouble describing it to clients?

Fuzz-C™ is a stand-alone preprocessor that seamlessly integrates **fuzzy logic** into the C language. Now your clients can add fuzzy logic to their applications without expensive, specialized hardware or software. The preprocessor generates C code that is both compact and significantly faster than most current commercial fuzzy logic implementations. Fuzz-C provides a practical, unified solution for applications that can benefit from fuzzy logic control systems. Use existing C libraries for program management, keyboard handlers and display functions without change, and implement system control functions using fuzzy rules.

Fuzz-C™ and Fuzzy Logic

Crisp numbers (the kind we're all familiar with) simply indicate the magnitude of a measurement. Fuzzy values describe the 'truth' or 'falsehood' (or something in-between) of *one or more qualities* relating to a crisp value. These membership functions are collected in a **linguistic variable declaration** complete with **membership functions**.

```
/* degrees Celsius */
LINGUISTIC room TYPE int MIN 0 MAX 50
{
  MEMBER cold   { 0, 0, 15, 20 }
  MEMBER normal { 20, 23, 25 }
  MEMBER hot    { 25, 30, 50, 50 }
}
```

Membership functions encode the “coldness” and “hotness” of different crisp temperatures. The reverse is true as well: the **consequence variables** (which provide us with an ultimate crisp result) relate fuzzy sets to crisp values.

```
/* A.C. on or off */
CONSEQUENCE ac TYPE int DEFUZZ CG
{
  MEMBER ON  { 1 }
  MEMBER OFF { 0 }
}
```

These declarations give us the makings of **fuzzy logic rules**. These rules test the 'truth' or 'falsehood' of the linguistic variable, and specify consequences accordingly.

```
/* Rules to follow */
FUZZY climateControl
{
  IF room IS cold THEN ac IS OFF
  IF room IS normal THEN ac IS OFF
  IF room IS hot THEN ac IS ON
}
```

Fuzzy logic starts with crisp values, analyzes them according to membership functions, and then uses consequence functions to arrive at a crisp result. There's nothing indeterminate about fuzzy logic: given the same input, you get the same output every time. *The difference is in the ease of coding the algorithm.*

```
int main(void)
{
  while(1)
    {
      /* find the temperature */
      room = thermostat;
      /* apply the rules */
      climateControl();
      /* switch the A.C. */
      airCon = ac;
      wait(10);
    }
}
```

Using Fuzz-C fuzzy logic in a program is easy. The LINGUISTIC variables (and CONSEQUENCE blocks alike) become variables in your program. Fuzzy logic rules become a function that you call when needed to perform the fuzzy calculation.

Fuzz-C is a flexible system that allows all (scalar) data types supported by your C compiler. It includes standard defuzzification methods, and accepts user-defined ones.

Fuzzy Logic and C in RS08

Even in the smallest applications, fuzzy logic can assist in implementation. Here are some suggestions on implementing fuzzy logic on RS08:

- Choose appropriate data types. Fuzzy logic passes its information with **Degree of Membership** values. The underlying type needs to have sufficient dynamic range for the problem at hand, but this isn't exclusively decided by the input or output types. An unsigned char is a perfectly acceptable degree of membership type.
- Use a fuzzy function as a soft thread. Cause the fuzzy function to run whenever the crisp input variables have a new value. Fuzz-C's architecture performs only as many calculations as necessary to arrive at a fuzzy result. For example

```
#pragma thread checkcount WAIT (SIP1.RTI == 1);

LINGUISTIC MTIMCNT TYPE unsigned char
{
  MEMBER low { 0 , modulo/4, modulo/2 }
  MEMBER mid { modulo/4, modulo/2, 3*modulo/4 }
  MEMBER hi  { modulo/2, 3*modulo/4, modulo }
}
FUZZY checkcount
{
  IF MTIMCNT IS low THEN adjust IS up;
  IF MTIMCNT IS high THEN adjust IS down;
}
```

Fuzzy PID controller

This is an excerpt of a fuzzy logic PID controller implemented using Fuzz-C. You can see by the fuzzy rules that the fuzzy kernel is processing the proportional, integral and derivative of the input to drive the output.

```
/* Fuzz-C Fuzzy Logic Preprocessor
   Fuzzy PID Controller
   Copyright 2005, 2006 Byte Craft Limited
*/
/* for calculating derivative error */
int OldError;

/* external set point */
int Setpoint;

/* the process that reads the ManVar and updates Error */
int Process(void);

/* proportional error */
LINGUISTIC Error TYPE int MIN -90 MAX 90
{
  MEMBER LNegative { -90, -90, -20, 0 }
  MEMBER normal    { -20, 0, 20 }
  MEMBER close     { -3, 0, 3 }
  MEMBER LPositive { 0, 20, 90, 90 }
}

LINGUISTIC DeltaError TYPE int MIN -90 MAX 90
{
  MEMBER Negative { -90, -90, -10, 0 }
  MEMBER Positive { 0, 10, 90, 90 }
}

LINGUISTIC SumError TYPE int MIN -90 MAX 90
{
  MEMBER LNeg { -90, -90, -5, 0 }
  MEMBER LPos { 0, 5, 90, 90 }
}

CONSEQUENCE ManVar TYPE int MIN -20 MAX 20 DEFUZZ cg
{
  MEMBER LNegative { -18 }
  MEMBER SNegative { -6 }
  MEMBER SPositive { 6 }
  MEMBER LPositive { 18 }
}

FUZZY pid
{
  /* large moves for large proportional errors */
  IF Error IS LNegative THEN ManVar IS LPositive
  IF Error IS LPositive THEN ManVar IS LNegative

  /* small moves for changes in error */
  IF Error IS normal AND DeltaError IS Positive
    THEN ManVar IS SNegative
  IF Error IS normal AND DeltaError IS Negative
    THEN ManVar IS SPositive

  /* small moves for large sums of accumulated error */
  IF Error IS close AND SumError IS LPos
    THEN ManVar IS SNegative
```

```
  IF Error IS close AND SumError IS LNeg
    THEN ManVar IS SPositive
}

void main (void)
{
  while(1)
    {
      OldError = Error;
      Error = Setpoint - Process();
      DeltaError = Error - OldError;
      SumError := SumError + Error;
      pid();
    }
}
```

Look at the complete source and generated listing files below:

Fuzzy PID Fuzz-C Source File

Fuzzy PID C Source

Fuzzy PID Listing File

The Business of Compilers

Our compiler technology is the product of **twenty years' experience** developing code generation strategies for different embedded platforms. We are experts in dealing with processor architectures designed for computer-generated code. Byte Craft compiler code generation technology produces tight, fast code. Byte Craft's tools equal or improve on the code density and execution speed of well-written hand-coded assembler.

Byte Craft Limited works with our silicon partners to develop benchmarks, leading to design wins and silicon sales. Many sales initiatives start with benchmark tests. Get Byte Craft Limited involved in this process; we consistently win benchmarks, and we can demonstrate through this process that we work with our silicon and hardware partners in the best interest of our common customers.

Byte Craft Limited can provide marketing materials, demos, and product information. Our demonstration products generate listing files, with which users can see the code we generate. We can provide machine-readable copies of all of our materials, in a wide variety of electronic formats.

Byte Craft Limited can contribute to joint presentations at customer sites. Byte Craft Limited staff travel around the world for client meetings, sales presentations, and standards body participation.

Byte Craft Limited is located in Waterloo, Ontario, Canada. We are in the Eastern Standard Time zone (EST/EDT, UTC-5).

Industry Presence

Byte Craft Limited sends representatives to selected embedded systems conferences. We decide to attend trade shows on a case-by-case basis, and frequently present in conference sessions. Contact us for more information.

Download demonstration versions of our products at:

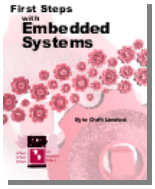
http://www.bytecraft.com/depot/depot.php

Demonstration versions do not create executables, but do create listing files to allow **inspection of generated code**. They come with full documentation and tutorials.

Walter Banks participates in meetings of ISO WG-14, the working group responsible for TR 18037, *Extensions for the Programming Language C to support Embedded Processors*. Walter's leadership has helped advance embedded systems' support in the C language standard.

Publications

We have significant experience in publishing. Our website features two publications free for download:



First Steps in Embedded Systems is a comprehensive general text introducing embedded systems. *Fuzzy Logic in Embedded Microcomputers and Control Systems* provides a tutorial on fuzzy logic as it pertains to control systems. Both publications can be downloaded from our website in PDF format.

To download: http://www.bytecraft.com/publishing.html

Product Development

Feedback and Improvements

Product development and technical support are intimately linked. The staff members responsible for developing our Code Development Systems also provide front-line technical support. Support calls or correspondence that reveals issues with the compiler result in improvements to all of our products.

Toolchain Innovation

We work actively with simulator and emulator vendors to complete the developers' toolchain. The Byte Craft Limited `.COD` file format and BCDIRECT embedded connectivity links allow the compiler to communicate all necessary information to downstream tools. We share standards documents on these technologies freely with other vendors where mutual advantage exists.

Standards

Byte Craft Limited Code Development Systems conform to ISO standards for C as much as possible for the target hardware platform. In many cases, a full implementation of standard C is not possible.

A C standard for embedded systems (ISO Technical Report 18037) now formally makes all of the processor registers, address spaces and internal flags available to the C programmer. This makes it possible to match any assembly language statement in C. By writing code in C and not assembly, every line of a program can benefit from the compiler's optimization and data-flow analysis. This is a big advantage to developers.

Byte Craft's president Walter Banks represented Canada at ISO and was a major player in the development of these standards.

About Byte Craft Limited

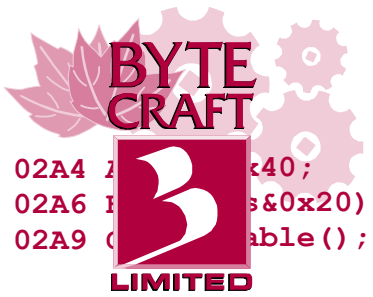
Byte Craft Limited is a Canadian company founded by Walter Banks, Jay Majithia, and Surin Kalra in July 1976. Walter Banks became the sole shareholder of Byte Craft Limited in May 1979. Walter Banks remains President of the company to this day.

In the early 1980's, Byte Craft Limited provided consulting development services for embedded systems in industrial, commercial and consumer products. Some of our achievements include the first mouse for the IBM-PC personal computer, environmental data loggers, and early personal data assistants.

Byte Craft Limited developed software tools to support our product development. The software tools we developed grew into a more important part of our business than original product development.

Byte Craft wrote over 90 assemblers in the 1980's, and shipped our first commercial compiler in 1984. We shipped our first C compiler in 1987.

Today, Byte Craft Limited is a software company specializing in development tools for embedded systems processors with unusual architectures or limited resources. Our main products are C cross-compilers targeted to a variety of microcontroller families. We provide innovative solutions for developers, consultants and manufacturers around the world.



Byte Craft Limited
A2-490 Dutton Drive
Waterloo, Ontario
Canada – N2L 6H7

+1 519.888.6911
sales@bytecraft.com
http://www.bytecraft.com

Byte Craft Limited leads the industry in embedded software engineering solutions; we produced the first C compilers for new platforms released by many of the leading semiconductor manufacturers worldwide. Byte Craft Limited tools are used by developers in automotive, telecommunications, consumer-product, industrial, and aerospace industries.

We have more than twenty-five years' experience in Embedded Systems, and significant experience in publishing and document management. We are devoting our efforts to putting excellent tools and knowledge into developers' hands.

Version	Date	Author	Notes
1	Apr. 2006	KZ	Initial version

Attachments

MC9RS08KA2 Device Header File

```
#ifndef __C6808DEF_H
#define __C6808DEF_H

/*****************************************************************************
 *                                                                           *
 * Byte Craft Limited C6808 Code Development System                          *
 *                                                                           *
 *****************************************************************************
 *                                                                           *
 * Header file information:                                                  *
 *                                                                           *
 * $ DeviceName:       <ALL> $                                               *
 * $ Manufacturer:     FREESCALE $                                           *
 * $ Filename:         MC9RS08KA2.H $                                        *
 * $ HeaderVer:        0.1  $                                                *
 * $ Copyright:        2006 $                                                *
 * $ Compiler:         C6808 $                                               *
 * $ CompilerVer:      2.0 $                                                 *
 *                                                                           *
 *****************************************************************************/

//#pragma option -l;
/*****************************************************************************
 *                                                                           *
 * This code may be adapted for any purpose when                             *
 * used with the C6808 Code Development System.                              *
 * No warranty is implied or given as to their                               *
 * usability for any purpose.                                                *
 *                                                                           *
 * (c) Copyright 2006 Byte Craft Limited                                     *
 * A2-490 Dutton Drive, Waterloo, ON, Canada, N2L 6H7                        *
 * VOICE: 1 (519) 888 6911                                                   *
 * FAX  : 1 (519) 746 6751                                                   *
 * email: support@bytecraft.com                                              *
 *                                                                           *
 *****************************************************************************
 *                                                                           *
 * Revision History:                                                         *
 * ~~~~~~~~~~~~~~~~~                                                         *
 * $ V:1.00  KZ  23/02/06 Initial version for RS08                        $ *
 * $ V:1.01  WB  18/03/06 Initial MC9RS08KA2                              $ *
 *                                                                           *
 *****************************************************************************
 *                                                                           *
 * Notes:                                                                    *
 * ~~~~~~                                                                    *
 * This is a generic header file that will work on many/most RS08 devices.   *
 * For more specific definitions, use the proper header file for the device. *
 *                                                                           *
 * Deviations from datasheet:                                                *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~                                                *
 * MTIMCLK.CLKS -> MTIMCLK.CLKSRC due to conflict with ICSC1 CLKS         *
 * SIP1.LVD -> SIP1.LVDI due to conflict with SRS.LVD                        *
 *                                                                           *
 * Aliases:                                                                  *
 * ~~~~~~~~                                                                  *
 * PTAD -> PORTA                                                             *
 *                                                                           *
 *****************************************************************************/

#pragma has RS08;
#pragma vector __RESET @ 0x3ffd; // reset Vector



/* Non volatile options */

#ifdef RS08_SECURE
#pragma fill @ 0x3FFC = 0x00;
#else
#pragma fill @ 0x3FFC = 0x01;
```

```
#endif /* RS_SECURE */


/* Registers */
#pragma regac AC;    /* Accumulator A */
#pragma regix X;     /* 0x000F        */

/* Memory */
/*
       ||<--TINY-->|<------RAMPROG------>|
       |0x0    0x0D|0x20             0x4F|
       |-----------------------------------|
        <---------RAM------------------->
*/

/* tiny/small memory */
#pragma memory RAM tiny [0x0D] @ 0x00;

#define RAMSIZE 0x2F
#define RAMSTART 0x0020

#pragma memory RAM [0x4F - RAMSTART] @ RAMSTART;
#pragma memory LOCAL[0] @ 0x4F;

#define ROMSIZE 2048-4
#define ROMSTART 0x3800
#pragma memory ROM [ROMSIZE] @ ROMSTART;


/* Interrupt vectors */

#pragma vector __RESET @ 0x3FFD;


/* PORTA data bits */

#pragma portrw PTAD @ 0x0010;
#pragma portrw PORTA @ 0x0010;
#define PTAD5    5
#define PTAD4    4
#define PTAD3    3
//PTAD bit 2 read only
#define PTAD2    2
#define PTAD1    1
#define PTAD0    0

/* PORTA data direction */

#pragma portrw PTADD @ 0x0011;
#define PTADD5   5
#define PTADD4   4
#define PTADD1   1
#define PTADD0   0

/* Analog comparator status and control */

#pragma portrw ACMPSC @ 0x0013;
#define ACME     7
#define ACBGS    6
#define ACF      6
#define ACIE     4
//ACO bit 3 read only
#define ACO      3
#define ACOPE    2
//ACMOD bits 0-1
#define ACMOD    0

/* Internal clock source control register 1 */

#pragma portrw ICSC1 @ 0x0014;
#define CLKS     6
#define IREFSTEN 0

/* Internal clock source control register 2 */

#pragma portrw ICSC2 @ 0x0015;
//BDIV bits 6-7
#define BDIV     6
#define LP       3

/* Internal clock source trim register */
```

```
#pragma portrw ICSTRM @ 0x0016;

/* Internal clock source status and control */

#pragma portrw ICSSC @ 0x0017;
#define CLKST    2
#define FTRIM    0

/* Modulo timer status and control */

#pragma portrw MTIMSC @ 0x0018;
//TOF bit 7 read only
#define TOF      7
#define TOIE     6
//TRST bit 5 write only
#define TRST     5
#define TSTP     4

/* Modulo timer clock configuration */

#pragma portrw MTIMCLK @ 0x0019;
//CLKSRS bits 4-5
//changed from datasheet due to conflict
#define CLKSRC   4
//PS bits 0-3
#define PS       0

/* Modulo timer clock counter */

#pragma portr MTIMCNT @ 0x001A;

/* Modulo timer modulo register */

#pragma portrw MTIMMOD @ 0x001B;

/* Keyboard interrupt status and control */

#pragma portrw KBISC @ 0x001C;
//KBF bit 3 read only
#define KBF      3
//KBACK bit 2 write only
#define KBACK    2
#define KBIE     1
#define KBIMOD   0

/* Keyboard interrupt pin enable */

#pragma portrw KBIPE @ 0x001D;
#define KBIPE5   5
#define KBIPE4   4
#define KBIPE2   2
#define KBIPE1   1
#define KBIPE0   0

/* Keyboard interrupt edge select */

#pragma portrw KBIES @ 0x001E;
#define KBEDG5   5
#define KBEDG4   4
#define KBEDG2   2
#define KBEDG1   1
#define KBEDG0   0

/* Page register */

#pragma portrw PAGE @ 0x001F;

/* System reset status */

#pragma portrw SRS @ 0x0200;
#define POR      7
#define PIN      6
#define COP      5
#define ILOP     4
#define ILAD     3
#define LVD      1
#define CLEAR_COP_TIMER() (SRS=0xFF)

/* System options */
```

```
//Note: write-once register
#pragma portrw SOPT @ 0x0201;
#define COPE     7
#define COPT     6
#define STOPE    5
#define BKGDPE   1
#define RSTPE    0

/* System interrupt pending */

#pragma portr SIP1 @ 0x0202;
#define KBI      4
#define ACMP     3
#define MTIM     2
#define RTI      1
//changed from LVD due to conflict
#define LVDI     0
/* For development
#pragma thread __KBI()  INTERRUPT (SIP1.KBI );
#pragma thread __ACMP() INTERRUPT (SIP1.ACMP);
#pragma thread __MTIM() INTERRUPT (SIP1.MTIM);
#pragma thread __RTI()  INTERRUPT (SIP1.RTI );
#pragma thread __LVDI() INTERRUPT (SIP1.LVDI);
*/

/* System identification (high) */

#pragma portr SDIDH @ 0x0206;
#define SDREV() ((SDIDH>>4)&0x0F)
//ID bits 0-3

/* System identification (low) */

#pragma portr SDIDL @ 0x0207;
#define SDID() ( ( ((unsigned int16)(SDIDH)) << 8 ) | (unsigned int16)(SDIDL) )

/* System real-time interrupt status and control */

#pragma portrw SRTISC @ 0x0208;
//RTIF bit 7 read only
#define RTIF     7
//RTACK bit 6 write only
#define RTIACK   6
#define RTICLKS  5
#define RTIE     4
//RTIS bits 0-2
#define RTIS     0

/* System power management status and control 1 */

#pragma portrw SPMSC1 @ 0x0209;
//LVDF bit 7 read only
#define LVDF     7
//LVDACK bit 6 write only
#define LVDACK   6
#define LVDIE    5
//LVDRE bit 4 write once
#define LVDRE    4
#define LVDSE    3
//LVDE bit 2 write once
#define LVDE     2
#define BGBE     0

/* Flash options */

#pragma portr FOPT @ 0x0210;
#define SECD     0

/* Flash control register */

#pragma portrw FLCR @ 0x0211;
#define HVEN     3
#define MASS     2
#define PGM      0

/* Pullup/pulldown enable */

#pragma portrw PTAPE @ 0x0220;
#define PTAPE5   5
#define PTAPE4   4
#define PTAPE2   2
```

```
#define PTAPE1   1
#define PTAPE0   0

/* Pullup/pulldown control */

#pragma portrw PTAPUD @ 0x0221;
#define PTAPUD5  5
#define PTAPUD4  4
#define PTAPUD2  2
#define PTAPUD1  1
#define PTAPUD0  0

/* Slew rate enable */

#pragma portrw PTASE @ 0x0222;
#define PTASE5   5
#define PTASE4   4
#define PTASE3   3
#define PTASE1   1
#define PTASE0   0

#endif /* __C6808DEF_H */
```

Quadrature Decoder C Source

```
/*****************************************************************************
 *                                                                           *
 * $  File name    :  quadenc.c                                           $ *
 * $  Description  :  Quadrature decoder.                                 $ *
 *****************************************************************************
 *                                                                           *
 *                                                                           *
 *  This routines may be adapted for any purpose when                        *
 *  used with the Byte Craft Limited Code Development                        *
 *  Systems.  No warranty is implied or given as to                          *
 *  their usability for any purpose.                                         *
 *                                                                           *
 *       (c) Copyright 2006                                                  *
 *       Byte Craft Limited                                                  *
 *       A2-490 Dutton Drive                                                 *
 *       Waterloo, Ontario                                                   *
 *       Canada N2L 6H7                                                      *
 *       (519) 888-6911                                                      *
 *                                                                           *
 *       Kirk Zurell                                                         *
 *                                                                           *
 *****************************************************************************
 *                                                                           *
 *   Revision history                                                        *
 * $    V:1.0 KZ Mar 2006 Initial release                                 $ *
 * $    V:1.1 WB APR 2006 Combined the state diagram into one function    $ *
 *                                                                           *
 *****************************************************************************/


#include <MC9RS08KA2.h>



//quadrature encoder

#pragma thread QDecode()   INTERRUPT (SIP1.KBI == 1);

int16 tally;  // Accummulation of quadrature counts


/*

    The Quad decoder is a FSM that cats on the following patterns
     readings ( All these cases will increment the quadrature tally)
     old new   swapped old new
     00  01             11  01
     01  11             10  11
     11  10             00  10
     10  00             01  00

     readings ( All these cases will decrement the quadrature tally)
     old new   swapped old new
     00  10             11  10
     01  00             10  00
```

```
     11  01             00  01
     10  11             01  11



    KBIES contains the complemented value of the last reading.
    PORTA contains the current quadrature reading.
 */

void QDecode (void)
  {
    char save_port,state;
    save_port = PORTA;
    state = ((KBIES << 2) & 0x0c);
    state = state + (save_port & 0x03) ;  // state Old + new value complemeted

    if      ((state == 0b1101) ||
             (state == 0b1011) ||
             (state == 0b0010) ||
             (state == 0b0100))     tally++;
    else if ((state == 0b1110) ||
             (state == 0b1000) ||
             (state == 0b0001) ||
             (state == 0b0111))     tally--;

    KBIES = ~ save_port;
    KBISC.KBACK = 1;
 }



void QuadInit (void)
  {
   KBISC.KBIE = 0;
 //pullup/down
 //port pins as inputs
   PTAD.PTADD0 = 0; PTAD.PTADD1 = 0;

 //rising edge for both as default.
   KBIES |= 0x11;

 //detect edges
   KBISC.KBIMOD = 0;

 //enable pins
   KBIPE |= 0x11;

 //clear false interrupts
   KBISC.KBACK = 1;

 //initialize transitions
   KBIES = ~ PORTA;

 //enable
   KBISC.KBIE = 1;

 // Initialize the Quadrature accummulation
   tally = 0;

  }


void main(void)
{
 QuadInit();
 while(1)
   {
    __DISPATCH();
   }
}
```

Quadrature Decoder Listing File

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE  1


/*****************************************************************************
```

```
                                    *
*
                                    * $  File name    :  quadenc.c
$ *
                                    * $  Description  :  Quadrature decoder.
$ *
*****************************************************************************
                                    *
*
                                    *
*
                                    *  This routines may be adapted for any purpose when
*
                                    *  used with the Byte Craft Limited Code Development
*
                                    *  Systems.  No warranty is implied or given as to
*
                                    *  their usability for any purpose.
*
                                    *
*
                                    *       (c) Copyright 2006
*
                                    *       Byte Craft Limited
*
                                    *       A2-490 Dutton Drive
*
                                    *       Waterloo, Ontario
*
                                    *       Canada N2L 6H7
*
                                    *       (519) 888-6911
*
                                    *
*
                                    *       Kirk Zurell
*
                                    *
*
*****************************************************************************
                                    *
*
                                    *   Revision history
*
                                    * $    V:1.0 KZ Mar 2006 Initial release
$ *
                                    * $    V:1.1 WB APR 2006 Combined the state diagram into one
function      $ *
                                    *
*
*****************************************************************************/


                                   #include <MC9RS08KA2.h>
                                   #ifndef __C6808DEF_H
0006                               #define __C6808DEF_H


/*****************************************************************************
                                    *
*
                                    * Byte Craft Limited C6808 Code Development System
*
                                    *
*
*****************************************************************************
                                    *
*
                                    * Header file information:
*
                                    *
*
                                    * $ DeviceName:       <ALL>
$                                            *
                                    * $ Manufacturer:     FREESCALE
$                                          *
```

```
                                    * $ Filename:          MC9RS08KA2.H
$                                     *
                                    * $ HeaderVer:         0.1
$                                       *
                                    * $ Copyright:         2006
$                                       *
                                    * $ Compiler:          C6808
$                                       *
                                    * $ CompilerVer:       2.0
$                                        *
                                    *
*

*****************************************************************************/

                                   //#pragma option -l;

/*****************************************************************************
                                    *
*
```

```
C6808 "C" COMPILER 0.0.4.0                                  PAGE  2

                                     * This code may be adapted for any purpose when
*
                                     * used with the C6808 Code Development System.
*
                                     * No warranty is implied or given as to their
*
                                     * usability for any purpose.
*
                                     *
*
                                     * (c) Copyright 2006 Byte Craft Limited
*
                                     * A2-490 Dutton Drive, Waterloo, ON, Canada, N2L 6H7
*
                                     * VOICE: 1 (519) 888 6911
*
                                     * FAX  : 1 (519) 746 6751
*
                                     * email: support@bytecraft.com
*
                                     *
*
*****************************************************************************
                                     *
*
                                     * Revision History:
*
                                     * ~~~~~~~~~~~~~~~~~
*
                                     * $ V:1.00  KZ  23/02/06 Initial version for RS08
$ *
                                     * $ V:1.01  WB  18/03/06 Initial MC9RS08KA2
$ *
                                     *
*
*****************************************************************************
                                     *
*
                                     * Notes:
*
                                     * ~~~~~~
*
                                     * This is a generic header file that will work on many/most RS08
devices.    *
                                     * For more specific definitions, use the proper header file for
the device.  *
                                     *
*
                                     * Deviations from datasheet:
*
                                     * ~~~~~~~~~~~~~~~~~~~~~~~~~~
*
                                     * MTIMCLK.CLKS -> MTIMCLK.CLKSRC due to conflict with ICSC1 CLKS
*
                                     * SIP1.LVD -> SIP1.LVDI due to conflict with SRS.LVD
*
                                     *
*
                                     * Aliases:
*
                                     * ~~~~~~~~
*
                                     * PTAD -> PORTA
*
                                     *
*
*****************************************************************************/

                                    #pragma has RS08;
3FFD                                #pragma vector __RESET @ 0x3ffd; // reset Vector
```

```
                                /* Non volatile options */

                                #ifdef RS08_SECURE
                                #pragma fill @ 0x3FFC = 0x00;
                                #else
3FFC 01                         #pragma fill @ 0x3FFC = 0x01;
                                #endif /* RS_SECURE */


                                /* Registers */
0000                            #pragma regac AC;    /* Accumulator A */
0000                            #pragma regix X;     /* 0x000F        */

                                /* Memory */
```

```
C6808 "C" COMPILER 0.0.4.0                                        PAGE  3

                                /*
                                       ||<--TINY-->|<------RAMPROG------>|
                                       |0x0    0x0D|0x20             0x4F|
                                       |-----------------------------------|
                                        <---------RAM------------------->
                                */

                                /* tiny/small memory */
0000 000D                       #pragma memory RAM tiny [0x0D] @ 0x00;

0000002F                        #define RAMSIZE 0x2F
00000020                        #define RAMSTART 0x0020

0020 002F                       #pragma memory RAM [0x4F - RAMSTART] @ RAMSTART;
004F 0000                       #pragma memory LOCAL[0] @ 0x4F;

000007FC                        #define ROMSIZE 2048-4
00003800                        #define ROMSTART 0x3800
3800 07FC                       #pragma memory ROM [ROMSIZE] @ ROMSTART;


                                /* Interrupt vectors */

3FFD                            #pragma vector __RESET @ 0x3FFD;


                                /* PORTA data bits */

0010                            #pragma portrw PTAD @ 0x0010;
0010                            #pragma portrw PORTA @ 0x0010;
00000005                        #define PTAD5   5
00000004                        #define PTAD4   4
00000003                        #define PTAD3   3
                                //PTAD bit 2 read only
00000002                        #define PTAD2   2
00000001                        #define PTAD1   1
00000000                        #define PTAD0   0

                                /* PORTA data direction */

0011                            #pragma portrw PTADD @ 0x0011;
00000005                        #define PTADD5  5
00000004                        #define PTADD4  4
00000001                        #define PTADD1  1
00000000                        #define PTADD0  0

                                /* Analog comparator status and control */

0013                            #pragma portrw ACMPSC @ 0x0013;
00000007                        #define ACME    7
00000006                        #define ACBGS   6
00000006                        #define ACF 6
00000004                        #define ACIE    4
                                //ACO bit 3 read only
00000003                        #define ACO 3
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  4


00000002                        #define ACOPE   2
                                //ACMOD bits 0-1
00000000                        #define ACMOD   0

                                /* Internal clock source control register 1 */

0014                            #pragma portrw ICSC1 @ 0x0014;
00000006                        #define CLKS    6
00000000                        #define IREFSTEN    0

                                /* Internal clock source control register 2 */

0015                            #pragma portrw ICSC2 @ 0x0015;
                                //BDIV bits 6-7
00000006                        #define BDIV    6
00000003                        #define LP  3

                                /* Internal clock source trim register */

0016                            #pragma portrw ICSTRM @ 0x0016;

                                /* Internal clock source status and control */

0017                            #pragma portrw ICSSC @ 0x0017;
00000002                        #define CLKST   2
00000000                        #define FTRIM   0

                                /* Modulo timer status and control */

0018                            #pragma portrw MTIMSC @ 0x0018;
                                //TOF bit 7 read only
00000007                        #define TOF 7
00000006                        #define TOIE    6
                                //TRST bit 5 write only
00000005                        #define TRST    5
00000004                        #define TSTP    4

                                /* Modulo timer clock configuration */

0019                            #pragma portrw MTIMCLK @ 0x0019;
                                //CLKSRS bits 4-5
                                //changed from datasheet due to conflict
00000004                        #define CLKSRC  4
                                //PS bits 0-3
00000000                        #define PS  0

                                /* Modulo timer clock counter */

001A                            #pragma portr MTIMCNT @ 0x001A;

                                /* Modulo timer modulo register */

001B                            #pragma portrw MTIMMOD @ 0x001B;

                                /* Keyboard interrupt status and control */
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  5


001C                            #pragma portrw KBISC @ 0x001C;
                                //KBF bit 3 read only
00000003                        #define KBF 3
                                //KBACK bit 2 write only
00000002                        #define KBACK   2
00000001                        #define KBIE    1
00000000                        #define KBIMOD  0

                                /* Keyboard interrupt pin enable */

001D                            #pragma portrw KBIPE @ 0x001D;
00000005                        #define KBIPE5  5
00000004                        #define KBIPE4  4
00000002                        #define KBIPE2  2
00000001                        #define KBIPE1  1
00000000                        #define KBIPE0  0

                                /* Keyboard interrupt edge select */

001E                            #pragma portrw KBIES @ 0x001E;
00000005                        #define KBEDG5  5
00000004                        #define KBEDG4  4
00000002                        #define KBEDG2  2
00000001                        #define KBEDG1  1
00000000                        #define KBEDG0  0

                                /* Page register */

001F                            #pragma portrw PAGE @ 0x001F;

                                /* System reset status */

0200                            #pragma portrw SRS @ 0x0200;
00000007                        #define POR 7
00000006                        #define PIN 6
00000005                        #define COP 5
00000004                        #define ILOP    4
00000003                        #define ILAD    3
00000001                        #define LVD 1
0007                            #define CLEAR_COP_TIMER() (SRS=0xFF)

                                /* System options */

                                //Note: write-once register
0201                            #pragma portrw SOPT @ 0x0201;
00000007                        #define COPE    7
00000006                        #define COPT    6
00000005                        #define STOPE   5
00000001                        #define BKGDPE  1
00000000                        #define RSTPE   0

                                /* System interrupt pending */

0202                            #pragma portr SIP1 @ 0x0202;
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  6


00000004                            #define KBI 4
00000003                            #define ACMP    3
00000002                            #define MTIM    2
00000001                            #define RTI 1
                                    //changed from LVD due to conflict
00000000                            #define LVDI    0
                                    /* For development
                                    #pragma thread __KBI()  INTERRUPT (SIP1.KBI );
                                    #pragma thread __ACMP() INTERRUPT (SIP1.ACMP);
                                    #pragma thread __MTIM() INTERRUPT (SIP1.MTIM);
                                    #pragma thread __RTI()  INTERRUPT (SIP1.RTI );
                                    #pragma thread __LVDI() INTERRUPT (SIP1.LVDI);
                                    */

                                    /* System identification (high) */

0206                                #pragma portr SDIDH @ 0x0206;
0008                                #define SDREV() ((SDIDH>>4)&0x0F)
                                    //ID bits 0-3

                                    /* System identification (low) */

0207                                #pragma portr SDIDL @ 0x0207;
0009                                #define SDID() ( ( ((unsigned int16)(SDIDH)) << 8 ) | (unsigned
int16)(SDIDL) )

                                    /* System real-time interrupt status and control */

0208                                #pragma portrw SRTISC @ 0x0208;
                                    //RTIF bit 7 read only
00000007                            #define RTIF    7
                                    //RTACK bit 6 write only
00000006                            #define RTIACK  6
00000005                            #define RTICLKS 5
00000004                            #define RTIE    4
                                    //RTIS bits 0-2
00000000                            #define RTIS    0

                                    /* System power management status and control 1 */

0209                                #pragma portrw SPMSC1 @ 0x0209;
                                    //LVDF bit 7 read only
00000007                            #define LVDF    7
                                    //LVDACK bit 6 write only
00000006                            #define LVDACK  6
00000005                            #define LVDIE   5
                                    //LVDRE bit 4 write once
00000004                            #define LVDRE   4
00000003                            #define LVDSE   3
                                    //LVDE bit 2 write once
00000002                            #define LVDE    2
00000000                            #define BGBE    0

                                    /* Flash options */

0210                                #pragma portr FOPT @ 0x0210;
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  7


00000000                        #define SECD    0

                                /* Flash control register */

0211                            #pragma portrw FLCR @ 0x0211;
00000003                        #define HVEN    3
00000002                        #define MASS    2
00000000                        #define PGM 0

                                /* Pullup/pulldown enable */

0220                            #pragma portrw PTAPE @ 0x0220;
00000005                        #define PTAPE5  5
00000004                        #define PTAPE4  4
00000002                        #define PTAPE2  2
00000001                        #define PTAPE1  1
00000000                        #define PTAPE0  0

                                /* Pullup/pulldown control */

0221                            #pragma portrw PTAPUD @ 0x0221;
00000005                        #define PTAPUD5 5
00000004                        #define PTAPUD4 4
00000002                        #define PTAPUD2 2
00000001                        #define PTAPUD1 1
00000000                        #define PTAPUD0 0

                                /* Slew rate enable */

0222                            #pragma portrw PTASE @ 0x0222;
00000005                        #define PTASE5  5
00000004                        #define PTASE4  4
00000003                        #define PTASE3  3
00000001                        #define PTASE1  1
00000000                        #define PTASE0  0

                                #endif /* __C6808DEF_H */



                                //quadrature encoder

                                #pragma thread QDecode()   INTERRUPT (SIP1.KBI == 1);

0020                            int16 tally;  // Accummulation of quadrature counts


                                /*

                                    The Quad decoder is a FSM that cats on the following patterns
                                     readings ( All these cases will increment the quadrature
tally)
                                     old new   swapped old new
                                     00  01             11  01
                                     01  11             10  11
```

```
C6808 "C" COMPILER 0.0.4.0                                        PAGE  8

                                     11  10             00  10
                                     10  00             01  00

                                     readings ( All these cases will decrement the quadrature
tally)
                                     old new   swapped old new
                                     00  10             11  10
                                     01  00             10  00
                                     11  01             00  01
                                     10  11             01  11


                                     KBIES contains the complemented value of the last reading.
                                     PORTA contains the current quadrature reading.
                                  */

                                  void QDecode (void)
                                    {
004F 004E                             char save_port,state;
3800 4E 10 4F   MOV   $10,$4F         save_port = PORTA;
3803 DE         LDA   $1E             state = ((KBIES << 2) & 0x0c);
3804 48         LSLA
3805 48         LSLA
3806 A4 0C      AND   #$0C
3808 B7 4E      STA   $4E
380A B6 4F      LDA   $4F             state = state + (save_port & 0x03) ;  // state Old + new
value complemeted
380C A4 03      AND   #$03
380E BB 4E      ADD   $4E
3810 B7 4E      STA   $4E

                                      if      ((state == 0b1101) ||
                                               (state == 0b1011) ||
3812 A1 0D      CMP   #$0D                       (state == 0b0010) ||
3814 37 0C      BEQ   $3822
3816 A1 0B      CMP   #$0B                       (state == 0b0100))     tally++;
3818 37 08      BEQ   $3822
381A A1 02      CMP   #$02
381C 37 04      BEQ   $3822
381E A1 04      CMP   #$04
3820 36 08      BNE   $382A
3822 3C 21      INC   $21
3824 36 02      BNE   $3828
3826 3C 20      INC   $20
3828 30 19      BRA   $3843           else if ((state == 0b1110) ||
                                               (state == 0b1000) ||
382A A1 0E      CMP   #$0E                       (state == 0b0001) ||
382C 37 0C      BEQ   $383A
382E A1 08      CMP   #$08                       (state == 0b0111))     tally--;
3830 37 08      BEQ   $383A
3832 A1 01      CMP   #$01
3834 37 04      BEQ   $383A
3836 A1 07      CMP   #$07
3838 36 09      BNE   $3843
383A 4E 21 21   TST   $21
383D 36 02      BNE   $3841
```

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE  9


383F 3A 20      DEC   $20
3841 3A 21      DEC   $21

3843 B6 4F      LDA   $4F               KBIES = ~ save_port;
3845 43         COMA
3846 FE         STA   $1E
3847 14 1C      BSET  2,$1C             KBISC.KBACK = 1;
3849 BE         RTS                   }



                                      void QuadInit (void)
                                        {
384A 13 1C      BCLR  1,$1C             KBISC.KBIE = 0;
                                       //pullup/down
                                       //port pins as inputs
384C 11 10      BCLR  0,$10             PTAD.PTADD0 = 0; PTAD.PTADD1 = 0;
384E 13 10      BCLR  1,$10

                                       //rising edge for both as default.
3850 DE         LDA   $1E               KBIES |= 0x11;
3851 AA 11      ORA   #$11
3853 FE         STA   $1E

                                       //detect edges
3854 11 1C      BCLR  0,$1C             KBISC.KBIMOD = 0;

                                       //enable pins
3856 DD         LDA   $1D               KBIPE |= 0x11;
3857 AA 11      ORA   #$11
3859 FD         STA   $1D

                                       //clear false interrupts
385A 14 1C      BSET  2,$1C             KBISC.KBACK = 1;

                                       //initialize transitions
385C D0         LDA   $10               KBIES = ~ PORTA;
385D 43         COMA
385E FE         STA   $1E

                                       //enable
385F 12 1C      BSET  1,$1C             KBISC.KBIE = 1;

                                       // Initialize the Quadrature accummulation
3861 3F 21      CLR   $21               tally = 0;
3863 3F 20      CLR   $20

3865 BE         RTS                    }


                                      void main(void)
                                      {
3866 AD E2      BSR   $384A            QuadInit();
                                       while(1)
3868 AD 02      BSR   $386C              {
```

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE 10


386A                                        __DISPATCH();
386A 30 FC      BRA   $3868                }
                                          }



                  __DISPATCH:
386C AF         WAIT
386D 09 C2 02   BRCLR 4,$0202,$3872
3870 30 8E      BRA   $3800
3872 BE         RTS
                  __MAIN:
3FFD BC 38 66
```

C6808 "C" COMPILER 0.0.4.0 PAGE 11

SYMBOL TABLE

LABEL	VALUE	LABEL	VALUE
ACBGS	0006	ACF	0006
ACIE	0004	ACME	0007
ACMOD	0000	ACMP	0003
ACMPSC	0013	ACO	0003
ACOPE	0002	BDIV	0006
BGBE	0000	BKGDPE	0001
CLKS	0006	CLKSRC	0004
CLKST	0002	COP	0005
COPE	0007	COPT	0006
FLCR	0211	FOPT	0210
FTRIM	0000	HVEN	0003
ICSC1	0014	ICSC2	0015
ICSSC	0017	ICSTRM	0016
ILAD	0003	ILOP	0004
IREFSTEN	0000	KBACK	0002
KBEDG0	0000	KBEDG1	0001
KBEDG2	0002	KBEDG4	0004
KBEDG5	0005	KBF	0003
KBI	0004	KBIE	0001
KBIES	001E	KBIMOD	0000
KBIPE	001D	KBIPE0	0000
KBIPE1	0001	KBIPE2	0002
KBIPE4	0004	KBIPE5	0005
KBISC	001C	LOCAL	0005
LP	0003	LVD	0001
LVDACK	0006	LVDE	0002
LVDF	0007	LVDI	0000
LVDIE	0005	LVDRE	0004
LVDSE	0003	MASS	0002
MTIM	0002	MTIMCLK	0019
MTIMCNT	001A	MTIMMOD	001B
MTIMSC	0018	PAGE	001F
PGM	0000	PIN	0006
POR	0007	PORTA	0010
PS	0000	PTAD	0010
PTAD0	0000	PTAD1	0001
PTAD2	0002	PTAD3	0003
PTAD4	0004	PTAD5	0005
PTADD	0011	PTADD0	0000
PTADD1	0001	PTADD4	0004
PTADD5	0005	PTAPE	0220
PTAPE0	0000	PTAPE1	0001
PTAPE2	0002	PTAPE4	0004
PTAPE5	0005	PTAPUD	0221
PTAPUD0	0000	PTAPUD1	0001
PTAPUD2	0002	PTAPUD4	0004
PTAPUD5	0005	PTASE	0222
PTASE0	0000	PTASE1	0001
PTASE3	0003	PTASE4	0004

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 12


SYMBOL TABLE - Continued

LABEL                           VALUE  LABEL                    VALUE


PTASE5                          0005 | QDecode                 3800
QuadInit                        384A | RAMSIZE                 002F
RAMSTART                        0020 | ROMSIZE                 07FC
ROMSTART                        3800 | RSTPE                   0000
RTI                             0001 | RTIACK                  0006
RTICLKS                         0005 | RTIE                    0004
RTIF                            0007 | RTIS                    0000
SDIDH                           0206 | SDIDL                   0207
SECD                            0000 | SIP1                    0202
SOPT                            0201 | SPMSC1                  0209
SRS                             0200 | SRTISC                  0208
STOPE                           0005 | TOF                     0007
TOIE                            0006 | TRST                    0005
TSTP                            0004 | _SP                     0000
__DISPATCH                      386C | __MAIN                  3866
__RESET                         3FFD | __RESET                 3FFD
__SPAD                          0026 | __longAC                0022
__longIX                        0024 | main                    3866
tally                           0020 | tiny                    0003


RAM USAGE MAP

0010 PTAD                           portrw
0010 PORTA                          portrw
0011 PTADD                          portrw
0013 ACMPSC                         portrw
0014 ICSC1                          portrw
0015 ICSC2                          portrw
0016 ICSTRM                         portrw
0017 ICSSC                          portrw
0018 MTIMSC                         portrw
0019 MTIMCLK                        portrw
001A MTIMCNT                        portr
001B MTIMMOD                        portrw
001C KBISC                          portrw
001D KBIPE                          portrw
001E KBIES                          portrw
001F PAGE                           portrw
0200 SRS                            portrw
0201 SOPT                           portrw
0202 SIP1                           portr
0206 SDIDH                          portr
0207 SDIDL                          portr
0208 SRTISC                         portrw
0209 SPMSC1                         portrw
0210 FOPT                           portr
```

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE 13



RAM USAGE MAP - Continued

0211 FLCR                           portrw
0220 PTAPE                          portrw
0221 PTAPUD                         portrw
0222 PTASE                          portrw
0020 tally                          signed int16
0022 __longAC                       signed int16
0024 __longIX                       signed int16
0026 __SPAD                         unsigned int32
004F save_port                      unsigned char       3800  3848
004E state                          unsigned char       3800  3848


LOCAL RAM USAGE FROM 004E TO 004F



ROM USAGE MAP

    3800 to 3872    3FFC to 3FFF
    Total ROM used 0078 (120)


Errors             :    0
Warnings           :    0
```

A/D Converter Source Code

```
#pragma option f 0;

/* A/D conversion for RS08
 * (C) 2006 Byte Craft Limited. All rights reserved
 * This software is licensed for use with the C6808
 * Code Development System.

 */


#include <rs08def.h>

//to delete:
#define _Bool bit

//to change: RAM -> SPECIAL
#pragma memory RAM Flash [2] @ 0x0000;

//helpful definitions
#define true 1
#define false 0

//for calibration mode
_Bool calibrating;

//for calibrated total count (non-volatile storage)
unsigned int16 totalstore;

//for calibrated total and engineering unit target
unsigned int16 totalcount;
unsigned int16 targetcount;
#define calibtarget (1000)

void main(void)
{
 unsigned int16 positivecount;
 unsigned int16 totaltarget;

 positivecount = 0;

 calibrating = false;

 //load any pre-existing calibrated target
 //won't be of much use until calibrated.
```

```
 totaltarget = totalstore;


 /* configure the comparator */
 ACMPSC = 1<<ACME       //enable
          |1<<ACBGS     //bandgap select
          |1<<ACF       //clear flag
          |0<<ACIE      //do not enable interrupt
          |1<<ACOPE     //pass output to pin
          |0b11<<ACMOD  //rising or falling edge.
 ;
#define comparator_hi() (ACMPSC.ACF)

 /* configure the modulo timer */
 MTIMSC = 1<<TOIE       //modulo timer interrupt enable
          |1<<TRST      //reset the timer
          |0<<TSTP      //allow the timer to run
 ;
#define timer_overflowed() (MTIMSC.TOF)

 MTIMCLK = 0b00<<CLKSRC //BUSCLK source
           |0b0000<<PS  //1:1 prescale
 ;

 MTIMMOD = 0xAA;        //modulo value.


  //determine whether to calibrate or operate
  if(false)
  {
    calibrating = true;
  }

  //main loop
  while(1)
  {

  WAIT(); //wait mode allows modulo clock to continue.

  /* awaken from interrupt and check sources */
  if(timer_overflowed())
   {
   totalcount++;

    if(comparator_hi())
    {
    positivecount++;

     if(calibrating)
     {

       //record total sample count for this maximum positive count
       if(positivecount = calibtarget)
       {
        //store total count (may take a while)
        totalstore = totalcount;
        //stop calibrating
        calibrating = false;
       }

     }
     else
     //not calibrating
     {

      //update display();

      if(totalcount == totaltarget)
      {
       //positivecount contains voltage reading.

       //test positivecount and
       //update_display();
       //turn_on_light();
       //sound_horn();


       //reset for next sample
       totalcount = 0;
       positivecount = 0;
```

```
      }

     }

    }

    MTIMSC.TRST = 1; //reset modulo timer
   }


  }

}
```

A/D Converter Listing File

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE  1

                                    #pragma option f 0;

                                    /* A/D conversion for RS08
                                     * (C) 2006 Byte Craft Limited. All rights reserved
                                     * This software is licensed for use with the C6808
                                     * Code Development System.

                                     */


                                    #include <rs08def.h>
                                    #ifndef __C6808DEF_H
0006                                #define __C6808DEF_H


/*****************************************************************************
                                     *
*
                                     * Byte Craft Limited C6808 Code Development System
*
                                     *
*
*****************************************************************************
                                     *
*
                                     * Header file information:
*
                                     *
*
                                     * $ DeviceName:       <ALL>
$                                              *
                                     * $ Manufacturer:     FREESCALE
$                                           *
                                     * $ Filename:         RS08DEF.H
$                                           *
                                     * $ HeaderVer:        0.1
$                                              *
                                     * $ Copyright:        2006
$                                              *
                                     * $ Compiler:         C6808
$                                              *
                                     * $ CompilerVer:      2.0
$                                                *
                                     *
*
*****************************************************************************/

                                    //#pragma option -l;

/*****************************************************************************
                                     *
*
                                     * This code may be adapted for any purpose when
*
                                     * used with the C6808 Code Development System.
*
```

```
                                    * No warranty is implied or given as to their
*
                                    * usability for any purpose.
*
                                    *
*
                                    * (c) Copyright 2006 Byte Craft Limited
*
                                    * A2-490 Dutton Drive, Waterloo, ON, Canada, N2L 6H7
*
                                    * VOICE: 1 (519) 888 6911
*
                                    * FAX  : 1 (519) 746 6751
*
                                    * email: support@bytecraft.com
*
                                    *
*
*******************************************************************************
                                    *
*
                                    * Revision History:
*
                                    * ~~~~~~~~~~~~~~~~~
*
                                    * $ V:1.00  KZ  23/02/06 Initial version for RS08
$ *
                                    *
*
*******************************************************************************
                                    *
*
                                    * Notes:
*
                                    * ~~~~~~
*
                                    * This is a generic header file that will work on many/most RS08
devices.    *
                                    * For more specific definitions, use the proper header file for
the device.  *
                                    *
*
                                    * Deviations from datasheet:
*
                                    * ~~~~~~~~~~~~~~~~~~~~~~~~~~
*
                                    * MTIMCLK.CLKS -> MTIMCLK.CLKSRC due to conflict with ICSC1 CLKS
*
                                    * SIP1.LVD -> SIP1.LVDI due to conflict with SRS.LVD
*
                                    *
*
                                    * Aliases:
*
                                    * ~~~~~~~~
*
                                    * PTAD -> PORTA
*
                                    *
*
*******************************************************************************/

                                   #pragma has RS08;


                                   /* Non volatile options */

                                   #ifdef RS08_SECURE
                                   #pragma fill @ 0x3FFC = 0x00;
                                   #else
3FFC 01                            #pragma fill @ 0x3FFC = 0x01;
                                   #endif /* RS_SECURE */


                                   /* Registers */
0000                               #pragma regac AC;    /* Accumulator A */
0000                               #pragma regix X;     /* 0x000F        */
```

```
                                /* Memory */
                                /*
                                       ||<--TINY-->|<------RAMPROG------>|
                                       |0x0    0x0D|0x20             0x4F|
                                       |-----------------------------------|
                                        <---------RAM------------------->
                                */

                                /* tiny/small memory */
0000 000D                       #pragma memory RAM tiny [0x0D] @ 0x00;

0000002F                        #define RAMSIZE 0x2F
00000020                        #define RAMSTART 0x0020

0020 002F                       #pragma memory RAM [0x4F - RAMSTART] @ RAMSTART;
0020 0000                       #pragma memory LOCAL[0] @ 0x20;

000007FC                        #define ROMSIZE 2048-4
00003800                        #define ROMSTART 0x3800
3800 07FC                       #pragma memory ROM [ROMSIZE] @ ROMSTART;


                                /* Interrupt vectors */

3FFD                            #pragma vector __RESET @ 0x3FFD;


                                /* PORTA data bits */

0010                            #pragma portrw PTAD @ 0x0010;
0010                            #pragma portrw PORTA @ 0x0010;
00000005                        #define PTAD5   5
00000004                        #define PTAD4   4
00000003                        #define PTAD3   3
                                //PTAD bit 2 read only
00000002                        #define PTAD2   2
00000001                        #define PTAD1   1
00000000                        #define PTAD0   0

                                /* PORTA data direction */

0011                            #pragma portrw PTADD @ 0x0011;
00000005                        #define PTADD5  5
00000004                        #define PTADD4  4
00000001                        #define PTADD1  1
00000000                        #define PTADD0  0

                                /* Analog comparator status and control */

0013                            #pragma portrw ACMPSC @ 0x0013;
00000007                        #define ACME    7
00000006                        #define ACBGS   6
00000006                        #define ACF 6
00000004                        #define ACIE    4
                                //ACO bit 3 read only
00000003                        #define ACO 3
00000002                        #define ACOPE   2
                                //ACMOD bits 0-1
00000000                        #define ACMOD   0

                                /* Internal clock source control register 1 */

0014                            #pragma portrw ICSC1 @ 0x0014;
00000006                        #define CLKS    6
00000000                        #define IREFSTEN    0

                                /* Internal clock source control register 2 */

0015                            #pragma portrw ICSC2 @ 0x0015;
                                //BDIV bits 6-7
00000006                        #define BDIV    6
00000003                        #define LP  3

                                /* Internal clock source trim register */

0016                            #pragma portrw ICSTRM @ 0x0016;

                                /* Internal clock source status and control */

0017                            #pragma portrw ICSSC @ 0x0017;
00000002                        #define CLKST   2
```

```
00000000                        #define FTRIM   0

                                /* Modulo timer status and control */

0018                            #pragma portrw MTIMSC @ 0x0018;
                                //TOF bit 7 read only
00000007                        #define TOF 7
00000006                        #define TOIE    6
                                //TRST bit 5 write only
00000005                        #define TRST    5
00000004                        #define TSTP    4

                                /* Modulo timer clock configuration */

0019                            #pragma portrw MTIMCLK @ 0x0019;
                                //CLKSRS bits 4-5
                                //changed from datasheet due to conflict
00000004                        #define CLKSRC  4
                                //PS bits 0-3
00000000                        #define PS  0

                                /* Modulo timer clock counter */

001A                            #pragma portr MTIMCNT @ 0x001A;

                                /* Modulo timer modulo register */

001B                            #pragma portrw MTIMMOD @ 0x001B;

                                /* Keyboard interrupt status and control */

001C                            #pragma portrw KBISC @ 0x001C;
                                //KBF bit 3 read only
00000003                        #define KBF 3
                                //KBACK bit 2 write only
00000002                        #define KBACK   2
00000001                        #define KBIE    1
00000000                        #define KBIMOD  0

                                /* Keyboard interrupt pin enable */

001D                            #pragma portrw KBIPE @ 0x001D;
00000005                        #define KBIPE5  5
00000004                        #define KBIPE4  4
00000002                        #define KBIPE2  2
00000001                        #define KBIPE1  1
00000000                        #define KBIPE0  0

                                /* Keyboard interrupt edge select */

001E                            #pragma portrw KBIES @ 0x001E;
00000005                        #define KBEDG5  5
00000004                        #define KBEDG4  4
00000002                        #define KBEDG2  2
00000001                        #define KBEDG1  1
00000000                        #define KBEDG0  0

                                /* Page register */

001F                            #pragma portrw PAGE @ 0x001F;

                                /* System reset status */

0200                            #pragma portrw SRS @ 0x0200;
00000007                        #define POR 7
00000006                        #define PIN 6
00000005                        #define COP 5
00000004                        #define ILOP    4
00000003                        #define ILAD    3
00000001                        #define LVD 1
0007                            #define CLEAR_COP_TIMER() (SRS=0xFF)

                                /* System options */

                                //Note: write-once register
0201                            #pragma portrw SOPT @ 0x0201;
00000007                        #define COPE    7
00000006                        #define COPT    6
00000005                        #define STOPE   5
00000001                        #define BKGDPE  1
00000000                        #define RSTPE   0
```

```
                                /* System interrupt pending */

0202                            #pragma portr SIP1 @ 0x0202;
00000004                        #define KBI 4
00000003                        #define ACMP    3
00000002                        #define MTIM    2
00000001                        #define RTI 1
                                //changed from LVD due to conflict
00000000                        #define LVDI    0

                                /* System identification (high) */

0206                            #pragma portr SDIDH @ 0x0206;
0008                            #define SDREV() ((SDIDH>>4)&0x0F)
                                //ID bits 0-3

                                /* System identification (low) */

0207                            #pragma portr SDIDL @ 0x0207;
0009                            #define SDID() ( ( ((unsigned int16)(SDIDH)) << 8 ) | (unsigned
int16)(SDIDL) )

                                /* System real-time interrupt status and control */

0208                            #pragma portrw SRTISC @ 0x0208;
                                //RTIF bit 7 read only
00000007                        #define RTIF    7
                                //RTACK bit 6 write only
00000006                        #define RTIACK  6
00000005                        #define RTICLKS 5
00000004                        #define RTIE    4
                                //RTIS bits 0-2
00000000                        #define RTIS    0

                                /* System power management status and control 1 */

0209                            #pragma portrw SPMSC1 @ 0x0209;
                                //LVDF bit 7 read only
00000007                        #define LVDF    7
                                //LVDACK bit 6 write only
00000006                        #define LVDACK  6
00000005                        #define LVDIE   5
                                //LVDRE bit 4 write once
00000004                        #define LVDRE   4
00000003                        #define LVDSE   3
                                //LVDE bit 2 write once
00000002                        #define LVDE    2
00000000                        #define BGBE    0

                                /* Flash options */

0210                            #pragma portr FOPT @ 0x0210;
00000000                        #define SECD    0

                                /* Flash control register */

0211                            #pragma portrw FLCR @ 0x0211;
00000003                        #define HVEN    3
00000002                        #define MASS    2
00000000                        #define PGM 0

                                /* Pullup/pulldown enable */

0220                            #pragma portrw PTAPE @ 0x0220;
00000005                        #define PTAPE5  5
00000004                        #define PTAPE4  4
00000002                        #define PTAPE2  2
00000001                        #define PTAPE1  1
00000000                        #define PTAPE0  0

                                /* Pullup/pulldown control */

0221                            #pragma portrw PTAPUD @ 0x0221;
00000005                        #define PTAPUD5 5
00000004                        #define PTAPUD4 4
00000002                        #define PTAPUD2 2
00000001                        #define PTAPUD1 1
00000000                        #define PTAPUD0 0

                                /* Slew rate enable */
```

```
0222                                #pragma portrw PTASE @ 0x0222;
00000005                            #define PTASE5  5
00000004                            #define PTASE4  4
00000003                            #define PTASE3  3
00000001                            #define PTASE1  1
00000000                            #define PTASE0  0

                                    #endif /* __C6808DEF_H */

                                    //to delete:
000A                                #define _Bool bit

                                    //to change: RAM -> SPECIAL
0000 0002                           #pragma memory RAM Flash [2] @ 0x0000;

                                    //helpful definitions
00000001                            #define true 1
00000000                            #define false 0

                                    //for calibration mode
0020 0000                           _Bool calibrating;

                                    //for calibrated total count (non-volatile storage)
0021                                unsigned int16 totalstore;

                                    //for calibrated total and engineering unit target
002B                                unsigned int16 totalcount;
002D                                unsigned int16 targetcount;
000B                                #define calibtarget (1000)

                                    void main(void)
                                    {
001F                                 unsigned int16 positivecount;
001D                                 unsigned int16 totaltarget;

3800 3F 20      CLR   $20            positivecount = 0;
3802 3F 1F      CLR   PAGE

3804 11 20      BCLR  0,$20          calibrating = false;

                                     //load any pre-existing calibrated target
                                     //won't be of much use until calibrated.
3806 4E 21 1D   MOV   $21,$1D        totaltarget = totalstore;
3809 4E 22 1E   MOV   $22,$1E


                                     /* configure the comparator */
                                     ACMPSC = 1<<ACME       //enable
                                              |1<<ACBGS     //bandgap select
                                              |1<<ACF       //clear flag
                                              |0<<ACIE      //do not enable interrupt
                                              |1<<ACOPE     //pass output to pin
380C 3E C7 13   MOV   #$C7,$13               |0b11<<ACMOD  //rising or falling edge.
380F                                 ;
000C                                #define comparator_hi() (ACMPSC.ACF)

                                     /* configure the modulo timer */
                                     MTIMSC = 1<<TOIE       //modulo timer interrupt enable
                                              |1<<TRST      //reset the timer
380F 3E 60 18   MOV   #$60,$18               |0<<TSTP      //allow the timer to run
3812                                 ;
000D                                #define timer_overflowed() (MTIMSC.TOF)

                                     MTIMCLK = 0b00<<CLKSRC //BUSCLK source
3812 3F 19      CLR   $19                       |0b0000<<PS  //1:1 prescale
3814                                 ;

3814 3E AA 1B   MOV   #$AA,$1B       MTIMMOD = 0xAA;        //modulo value.


                                      //determine whether to calibrate or operate
                                      if(false)
                                      {
                                        calibrating = true;
                                      }
                                      //main loop
                                      while(1)
                                      {
3817 AF         WAIT                  WAIT(); //wait mode allows modulo clock to continue.
```

```
                                    /* awaken from interrupt and check sources */
3818 0F 18 44   BRCLR 7,$18,$385F   if(timer_overflowed())
381B                                 {
381B 3C 2C      INC   $2C            totalcount++;
381D 36 02      BNE   $3821
381F 3C 2B      INC   $2B

3821 0D 13 39   BRCLR 6,$13,$385D     if(comparator_hi())
3824                                  {
3824 3C 20      INC   $20             positivecount++;
3826 36 02      BNE   $382A
3828 3C 1F      INC   PAGE

382A 01 20 1C   BRCLR 0,$20,$3849     if(calibrating)
382D                                  {

                                        //record total sample count for this maximum positive
count
382D 3E 03 23   MOV   #$03,$23          if(positivecount = calibtarget)
3830 3E E8 24   MOV   #$E8,$24
3833 4E 23 1F   MOV   $23,PAGE          {
3836 4E 24 20   MOV   $24,$20
3839 B6 20      LDA   $20
383B BA 1F      ORA   PAGE
383D 37 08      BEQ   $3847
                                         //store total count (may take a while)
383F 4E 2B 21   MOV   $2B,$21            totalstore = totalcount;
3842 4E 2C 22   MOV   $2C,$22
                                         //stop calibrating
3845 11 20      BCLR  0,$20              calibrating = false;
                                        }

                                       }
3847 30 14      BRA   $385D             else
                                        //not calibrating
                                        {

                                         //update display();

3849 B6 2C      LDA   $2C               if(totalcount == totaltarget)
384B B1 1E      CMP   $1E               {
384D 36 0E      BNE   $385D
384F B6 2B      LDA   $2B
3851 B1 1D      CMP   $1D
3853 36 08      BNE   $385D
                                          //positivecount contains voltage reading.

                                          //test positivecount and
                                          //update_display();
                                          //turn_on_light();
                                          //sound_horn();


                                          //reset for next sample
3855 3F 2C      CLR   $2C                 totalcount = 0;
3857 3F 2B      CLR   $2B
3859 3F 20      CLR   $20                 positivecount = 0;
385B 3F 1F      CLR   PAGE

                                         }

                                        }

                                       }

385D 1A 18      BSET  5,$18             MTIMSC.TRST = 1; //reset modulo timer
                                       }



385F 30 B6      BRA   $3817           }

                                    }

                __MAIN:
3FFD BC 38 00


SYMBOL TABLE

LABEL                           VALUE LABEL                           VALUE
```

```
ACBGS                          0006 | ACF                            0006
ACIE                           0004 | ACME                           0007
ACMOD                          0000 | ACMP                           0003
ACMPSC                         0013 | ACO                            0003
ACOPE                          0002 | BDIV                           0006
BGBE                           0000 | BKGDPE                         0001
CLKS                           0006 | CLKSRC                        0004
CLKST                          0002 | COP                            0005
COPE                           0007 | COPT                           0006
FLCR                           0211 | FOPT                           0210
FTRIM                          0000 | Flash                          0007
HVEN                           0003 | ICSC1                          0014
ICSC2                          0015 | ICSSC                          0017
ICSTRM                         0016 | ILAD                           0003
ILOP                           0004 | IREFSTEN                       0000
KBACK                          0002 | KBEDG0                         0000
KBEDG1                         0001 | KBEDG2                         0002
KBEDG4                         0004 | KBEDG5                         0005
KBF                            0003 | KBI                            0004
KBIE                           0001 | KBIES                          001E
KBIMOD                         0000 | KBIPE                          001D
KBIPE0                         0000 | KBIPE1                         0001
KBIPE2                         0002 | KBIPE4                         0004
KBIPE5                         0005 | KBISC                          001C
LOCAL                          0005 | LP                             0003
LVD                            0001 | LVDACK                         0006
LVDE                           0002 | LVDF                           0007
LVDI                           0000 | LVDIE                          0005
LVDRE                          0004 | LVDSE                          0003
MASS                           0002 | MTIM                           0002
MTIMCLK                        0019 | MTIMCNT                        001A
MTIMMOD                        001B | MTIMSC                         0018
PAGE                           001F | PGM                            0000
PIN                            0006 | POR                            0007
PORTA                          0010 | PS                             0000
PTAD                           0010 | PTAD0                          0000
PTAD1                          0001 | PTAD2                          0002
PTAD3                          0003 | PTAD4                          0004
PTAD5                          0005 | PTADD                          0011
PTADD0                         0000 | PTADD1                         0001
PTADD4                         0004 | PTADD5                         0005
PTAPE                          0220 | PTAPE0                         0000
PTAPE1                         0001 | PTAPE2                         0002
PTAPE4                         0004 | PTAPE5                         0005
PTAPUD                         0221 | PTAPUD0                        0000
PTAPUD1                        0001 | PTAPUD2                        0002
PTAPUD4                        0004 | PTAPUD5                        0005
PTASE                          0222 | PTASE0                         0000
PTASE1                         0001 | PTASE3                         0003
PTASE4                         0004 | PTASE5                         0005
RAMSIZE                        002F | RAMSTART                       0020
ROMSIZE                        07FC | ROMSTART                       3800
RSTPE                          0000 | RTI                            0001
RTIACK                         0006 | RTICLKS                        0005
RTIE                           0004 | RTIF                           0007
RTIS                           0000 | SDIDH                          0206
SDIDL                          0207 | SECD                           0000
SIP1                           0202 | SOPT                           0201
SPMSC1                         0209 | SRS                            0200
SRTISC                         0208 | STOPE                          0005
TOF                            0007 | TOIE                           0006
TRST                           0005 | TSTP                           0004
_SP                            0000 | __MAIN                         3800
__RESET                        3FFD | __SPAD                         0027
___0020                        0020 | __longAC                       0023
__longIX                       0025 | calibrating                    0020
false                          0000 | main                           3800
targetcount                    002D | tiny                           0003
totalcount                     002B | totalstore                     0021
true                           0001 |


RAM USAGE MAP

0010 PTAD                           portrw
0010 PORTA                          portrw
0011 PTADD                          portrw
0013 ACMPSC                         portrw
0014 ICSC1                          portrw
0015 ICSC2                          portrw
```

```
0016 ICSTRM                          portrw
0017 ICSSC                           portrw
0018 MTIMSC                          portrw
0019 MTIMCLK                         portrw
001A MTIMCNT                         portr
001B MTIMMOD                         portrw
001C KBISC                           portrw
001D KBIPE                           portrw
001E KBIES                           portrw
001F PAGE                            portrw
0200 SRS                             portrw
0201 SOPT                            portrw
0202 SIP1                            portr
0206 SDIDH                           portr
0207 SDIDL                           portr
0208 SRTISC                          portrw
0209 SPMSC1                          portrw
0210 FOPT                            portr
0211 FLCR                            portrw
0220 PTAPE                           portrw
0221 PTAPUD                          portrw
0222 PTASE                           portrw
0020 ___0020                         portrw
0021 totalstore                      unsigned int16
0023 __longAC                        signed int16
0025 __longIX                        signed int16
0027 __SPAD                          unsigned int32
002B totalcount                      unsigned int16
002D targetcount                     unsigned int16
001F positivecount                   unsigned int16      3800  3860
001D totaltarget                     unsigned int16      3800  3860
000C comparator_hi                   **                  3800  3860
000D timer_overflowed                **                  3800  3860


LOCAL RAM USAGE FROM 001D TO 0020



ROM USAGE MAP

    3800 to 3860    3FFC to 3FFF
    Total ROM used 0066 (102)


Errors             :    0
Warnings           :    0
```

Fuzz-C™ Header File

```
/* (C) 1993, 2006 Byte Craft Limited */
#define F_ONE      0xff
#define F_ZERO     0x00
#define F_OR(a,b)  ((a) > (b) ? (a) : (b))
#define F_AND(a,b) ((a) < (b) ? (a) : (b))
#define F_NOT(a)   (F_ONE+F_ZERO-a)
```

Fuzzy PID Fuzz-C Source File

```
#include "MC9RS08KA2.h"
/* Fuzz-C Fuzzy Logic Preprocessor

   Fuzzy PID controller

   Copyright 2005 Byte Craft Limited. All rights reserved

   This code may be adapted for any purpose when used with the Fuzz-C
   fuzzy logic preprocessor. No warranty is implied or given as to
   its usability for any purpose.

   Byte Craft Limited
   A2-490 Dutton Drive, Waterloo, Ontario, Canada, N2L 6H7
   VOICE: 1 (519) 888 6911
   FAX  : 1 (519) 746 6751
   email: support@bytecraft.com

   Author: Walter Banks

   Revision History
```

```
   dd mmm yyyy  ii  t
          1993  WB  First version
       Oct 2005  KZ  Comments

   This program implements a fuzzy PID
   (Proportional-Integral-Derivative) controller.

*/


/* for calculating derivative error */
int OldError;

/* external set point */
int Setpoint;

/* the process that reads the ManVar and updates Error */
int Process(void);

/* proportional error */
LINGUISTIC Error TYPE int MIN -90 MAX 90
{
  MEMBER LNegative { -90, -90, -20, 0 }
  MEMBER normal    { -20, 0, 20 }
  MEMBER close     { -3, 0, 3 }
  MEMBER LPositive { 0, 20, 90, 90 }
}

LINGUISTIC DeltaError TYPE int MIN -90 MAX 90
{
  MEMBER Negative { -90, -90, -10, 0 }
  MEMBER Positive { 0, 10, 90, 90 }
}

LINGUISTIC SumError TYPE int MIN -90 MAX 90
{
  MEMBER LNeg { -90, -90, -5, 0 }
  MEMBER LPos { 0, 5, 90, 90 }
}

CONSEQUENCE ManVar TYPE int MIN -20 MAX 20 DEFUZZ cg
{
  MEMBER LNegative { -18 }
  MEMBER SNegative { -6 }
  MEMBER SPositive { 6 }
  MEMBER LPositive { 18 }
}

FUZZY pid
{
  /* large moves for large proportional errors */
  IF Error IS LNegative THEN ManVar IS LPositive
  IF Error IS LPositive THEN ManVar IS LNegative

  /* small moves for changes in error */
  IF Error IS normal AND DeltaError IS Positive
    THEN ManVar IS SNegative
  IF Error IS normal AND DeltaError IS Negative
    THEN ManVar IS SPositive

  /* small moves for large sums of accumulated error */
  IF Error IS close AND SumError IS LPos
    THEN ManVar IS SNegative
  IF Error IS close AND SumError IS LNeg
    THEN ManVar IS SPositive
}

void main (void)
{
  while(1)
    {
      OldError = Error;
      Error = Setpoint - Process();
      DeltaError = Error - OldError;
      SumError := SumError + Error;
      pid();
    }
}
```

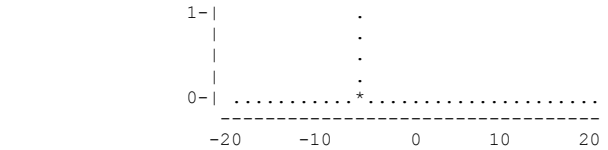
Fuzzy PID C Source

```
#include "MC9RS08KA2.h"
#include "fuzzc.h"
char __IDOM[2];
#pragma has RS08;
/* Fuzz-C Fuzzy Logic Preprocessor

   Fuzzy PID controller
   Copyright 2005 Byte Craft Limited. All rights reserved
   This code may be adapted for any purpose when used with the Fuzz-C
   fuzzy logic preprocessor. No warranty is implied or given as to
   its usability for any purpose.
   Byte Craft Limited
   A2-490 Dutton Drive, Waterloo, ON, Canada, N2L 6H7
   VOICE: 1 (519) 888 6911
   FAX  : 1 (519) 746 6751
   email: support@bytecraft.com
   Author: Walter Banks
   Revision History
   dd mmm yyyy  ii  t
          1993  WB  First version
      Oct 2005  KZ  Comments

   This program implements a fuzzy PID
   (Proportional-Integral-Derivative) controller.
Overlapping linguistic variables are fine
and assure smooth transitions between dominating FUZZY rules.
For example in the following fuzzy PID
controller we conditioned other rules by characterizing
where we were overall with 4 linguistic variables LNegative
(large negative) normal, close, LPositive .  It is interesting
to note that close (to the setpoint) is entirely contained
in normal. By partitioning the problem this way we
reduced the rule set to a half dozen rules to implement
non linear fuzzy based  PID controller.
The plots and PID reference source code follows
*/

 int OldError,SumError,Setpoint;
 int Process(void) { return 1; } // Dummy Process
/*  LINGUISTIC Error  TYPE int  MIN -90  MAX 90 */
/*   { */
int  Error ;
/*   MEMBER LNegative   { -90, -90, -20,  0  } */
/*

  1-| ............
    | .           .
    | .            .
    | .             .
  0-| .              ................
    ----------------------------------
    -90     -45       0      45      90

*/
char Error_LNegative (int __CRISP)
  {
    {
    if (__CRISP <= -20) return(255);
    else
      {
      if (__CRISP <= 0)
        return(( - __CRISP * 12) + 7);
      else
        return(0);
      }
    }
  }
/*   MEMBER normal      { -20,   0,  20      } */
/*

  1-|                .
    |               . .
    |              .   .
    |             .     .
  0-| ............       ............
    ----------------------------------
    -90     -45       0      45      90
```

```
*/
char Error_normal (int __CRISP)
  {
  if (__CRISP < -20) return(0);
  else
    {
    if (__CRISP <= 0) return(((__CRISP + 20) * 12) + 7);
    else
      {
        {
        if (__CRISP <= 20)
          return((( + 20 - __CRISP) * 12) + 7);
        else
          return(0);
        }
      }
    }
  }
/*   MEMBER close       {  -3,   0,   3      } */
/*

  1-|                 .
    |                 .
    |                 ..
    |                . .
  0-| ................ .................
     ----------------------------------
    -90     -45       0      45      90

*/
char Error_close (int __CRISP)
  {
  if (__CRISP < -3) return(0);
  else
    {
    if (__CRISP <= 0) return((__CRISP + 3) * 85);
    else
      {
        {
        if (__CRISP <= 3)
          return(( + 3 - __CRISP) * 85);
        else
          return(0);
        }
      }
    }
  }
/*   MEMBER LPositive   {   0,  20,  90,  90 } */
/*

  1-|                     .............
    |                    .            .
    |                   .             .
    |                  .              .
  0-| .................              .
     ----------------------------------
    -90     -45       0      45      90

*/
char Error_LPositive (int __CRISP)
  {
  if (__CRISP < 0) return(0);
  else
    {
    if (__CRISP <= 20) return((__CRISP * 12) + 7);
    else
      {
        return(255);
      }
    }
  }
/*   } */
/*

     Fuzzy Sets for Error

  1-| ..............   .   ..............
    | .            . ... .            .
    | .             . ...             .
    | .            . . . .            .
  0-| ..............  ...  ..............
```

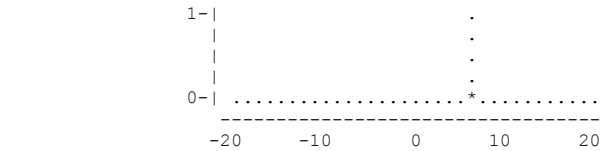
```
     ---------------------------------
    -90     -45       0      45      90


*/
/*  LINGUISTIC DeltaError  TYPE int  MIN -90  MAX 90 */
/*   { */
int  DeltaError ;
/*   MEMBER Negative    { -90, -90, -10,  0  } */
/*

  1-| ..............
    | .              .
    | .              .
    | .               .
  0-| .               ................
    ---------------------------------
    -90     -45       0      45      90

*/
char DeltaError_Negative (int __CRISP)
  {
    {
    if (__CRISP <= -10) return(255);
    else
      {
      if (__CRISP <= 0)
        return(( - __CRISP * 25) + 2);
      else
        return(0);
      }
    }
  }
/*   MEMBER Positive    {   0,  10,  90,  90 } */
/*

  1-|                   ..............
    |                   .             .
    |                  .              .
    |                  .              .
  0-| .................              .
    ---------------------------------
    -90     -45       0      45      90

*/
char DeltaError_Positive (int __CRISP)
  {
  if (__CRISP < 0) return(0);
  else
    {
    if (__CRISP <= 10) return((__CRISP * 25) + 2);
    else
      {
        return(255);
      }
    }
  }
/*   } */
/*

     Fuzzy Sets for DeltaError

  1-| ..............   ..............
    | .              .  .             .
    | .              . .              .
    | .               ..              .
  0-| ................................
    ---------------------------------
    -90     -45       0      45      90


*/
/*  LINGUISTIC SumError  TYPE int  MIN -90  MAX 90 */
/*   { */
/*   MEMBER LNeg    { -90, -90,  -5,  0  } */
/*

  1-| ..............
    | .              .
    | .               .
    | .               .
```

```
  0-| .               .................
    ----------------------------------
    -90     -45       0      45      90

*/
char SumError_LNeg (int __CRISP)
  {
    {
    if (__CRISP <= -5) return(255);
    else
      {
      if (__CRISP <= 0)
        return( - __CRISP * 51);
      else
        return(0);
      }
    }
  }
/*   MEMBER LPos    {   0,   5,  90,  90 } */
/*

  1-|                  .................
    |                  .               .
    |                  .               .
    |                 .                .
  0-| .................                .
    ----------------------------------
    -90     -45       0      45      90

*/
char SumError_LPos (int __CRISP)
  {
  if (__CRISP < 0) return(0);
  else
    {
    if (__CRISP <= 5) return(__CRISP * 51);
    else
      {
        return(255);
      }
    }
  }
/*   } */
/*

     Fuzzy Sets for SumError

  1-| ................. .................
    | .              . .              .
    | .               ..              .
    | .               .               .
  0-| ................................
    ----------------------------------
    -90     -45       0      45      90


*/
/*  CONSEQUENCE  ManVar TYPE int   MIN -20  MAX 20 DEFUZZ cg */
/*  { */
int  ManVar ;
  int fa_ManVar, fc_ManVar;
/*      MEMBER  LNegative  {  -18 } */
/*

  1-|   .
    |   .
    |   .
    |   .
  0-| ..*.............................
    ----------------------------------
    -20     -10       0      10      20


*/
void ManVar_LNegative (int __DOM)
  {
     fc_ManVar += __DOM;
     fa_ManVar += (__DOM * (-18));
  }
/*      MEMBER  SNegative  {   -6 } */
/*
```



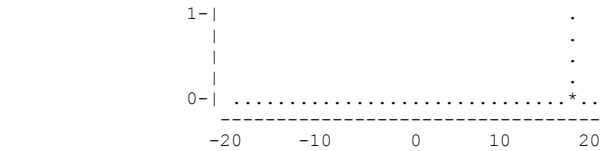
```
  1-|            .
    |            .
    |            .
    |            .
  0-| ...........*.....................
    ----------------------------------
    -20     -10       0      10      20
```

```
*/
void ManVar_SNegative (int __DOM)
  {
     fc_ManVar += __DOM;
     fa_ManVar += (__DOM * (-6));
  }
/*      MEMBER  SPositive  {    6 } */
/*
```



```
  1-|                      .
    |                      .
    |                      .
    |                      .
  0-| ......................*...........
    ----------------------------------
    -20     -10       0      10      20
```

```
*/
void ManVar_SPositive (int __DOM)
  {
     fc_ManVar += __DOM;
     fa_ManVar += (__DOM * (6));
  }
/*      MEMBER  LPositive  {   18 } */
/*
```



```
  1-|                               .
    |                               .
    |                               .
    |                               .
  0-| ..............................*..
    ----------------------------------
    -20     -10       0      10      20
```

```
*/
void ManVar_LPositive (int __DOM)
  {
     fc_ManVar += __DOM;
     fa_ManVar += (__DOM * (18));
  }
/*  } */
/*  FUZZY pid */
void pid (void)
  {
     fa_ManVar = 0;
     fc_ManVar = 0;
/*    { */
/*     IF Error IS LNegative THEN ManVar IS LPositive */
  ManVar_LPositive( Error_LNegative(Error)  );
/*     IF Error IS LPositive THEN ManVar IS LNegative */
  ManVar_LNegative( Error_LPositive(Error)  );
/*     IF Error IS normal AND DeltaError IS Positive */
/*       THEN ManVar IS SNegative */
  __IDOM[1] = Error_normal(Error) ;
  __IDOM[0] = DeltaError_Positive(DeltaError) ;
  __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
  ManVar_SNegative( __IDOM[0] );
/*     IF Error IS normal AND DeltaError IS Negative */
/*       THEN ManVar IS SPositive */
  __IDOM[1] = Error_normal(Error) ;
  __IDOM[0] = DeltaError_Negative(DeltaError) ;
  __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
  ManVar_SPositive( __IDOM[0] );
/*     IF Error IS close AND SumError IS LPos */
/*       THEN ManVar IS SNegative */
  __IDOM[1] = Error_close(Error) ;
  __IDOM[0] = SumError_LPos(SumError) ;
  __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
  ManVar_SNegative( __IDOM[0] );
```

```
/*     IF Error IS close AND SumError IS LNeg */
/*       THEN ManVar IS SPositive */
  __IDOM[1] = Error_close(Error) ;
  __IDOM[0] = SumError_LNeg(SumError) ;
  __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
  ManVar_SPositive( __IDOM[0] );
/*    } */
    if (fc_ManVar == 0)
     ManVar = 0;
    else
     ManVar = fa_ManVar / fc_ManVar;
   }
 void main (void)
   {
    while(1)
      {
        OldError = Error;
        Error = Setpoint - Process();
        DeltaError = Error - OldError;
        SumError = SumError + Error;
        pid();
       }
    }
```

Fuzzy PID Listing File

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE  1

                                    #include "MC9RS08KA2.h"
                                    #ifndef __C6808DEF_H
0006                                #define __C6808DEF_H


/*****************************************************************************
                                     *
*
                                     * Byte Craft Limited C6808 Code Development System
*
                                     *
*
******************************************************************************
                                     *
*
                                     * Header file information:
*
                                     *
*
                                     * $ DeviceName:       <ALL>
$                                                  *
                                     * $ Manufacturer:     FREESCALE
$                                          *
                                     * $ Filename:         MC9RS08KA2.H
$                                        *
                                     * $ HeaderVer:        0.1
$                                                *
                                     * $ Copyright:        2006
$                                                *
                                     * $ Compiler:         C6808
$                                                *
                                     * $ CompilerVer:      2.0
$                                                 *
                                     *
*
*****************************************************************************/

                                    //#pragma option -l;

/*****************************************************************************
                                     *
*
                                     * This code may be adapted for any purpose when
*
                                     * used with the C6808 Code Development System.
*
                                     * No warranty is implied or given as to their
*
```

```
                                * usability for any purpose.
*                               *
*                               *
*                               * (c) Copyright 2006 Byte Craft Limited
*                               * A2-490 Dutton Drive, Waterloo, ON, Canada, N2L 6H7
*                               * VOICE: 1 (519) 888 6911
*                               * FAX  : 1 (519) 746 6751
*                               * email: support@bytecraft.com
*                               *
*
*******************************************************************************
                                *
*                               * Revision History:
*                               * ~~~~~~~~~~~~~~~~~
*                               * $ V:1.00  KZ  23/02/06 Initial version for RS08
$ *                             * $ V:1.01  WB  18/03/06 Initial MC9RS08KA2
$ *                             *
*
*******************************************************************************
                                *
*                               * Notes:
*                               * ~~~~~~
*                               * This is a generic header file that will work on many/most RS08
devices.    *
                                * For more specific definitions, use the proper header file for
the device.  *
                                *
*                               * Deviations from datasheet:
*                               * ~~~~~~~~~~~~~~~~~~~~~~~~~~
*                               * MTIMCLK.CLKS -> MTIMCLK.CLKSRC due to conflict with ICSC1 CLKS
*                               * SIP1.LVD -> SIP1.LVDI due to conflict with SRS.LVD
*                               *
*
```

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE  2

                                    * Aliases:
*
                                    * ~~~~~~~~
*
                                    * PTAD -> PORTA
*
                                    *
*
*******************************************************************************/

                                   #pragma has RS08;
3FFD                               #pragma vector __RESET @ 0x3ffd; // reset Vector



                                   /* Non volatile options */

                                   #ifdef RS08_SECURE
                                   #pragma fill @ 0x3FFC = 0x00;
                                   #else
3FFC 01                            #pragma fill @ 0x3FFC = 0x01;
                                   #endif /* RS_SECURE */


                                   /* Registers */
0000                               #pragma regac AC;    /* Accumulator A */
0000                               #pragma regix X;     /* 0x000F        */

                                   /* Memory */
                                   /*
                                          ||<--TINY-->|<------RAMPROG------>|
                                          |0x0    0x0D|0x20             0x4F|
                                          |-----------------------------------|
                                           <---------RAM------------------->
                                   */

                                   /* tiny/small memory */
0000 000D                          #pragma memory RAM tiny [0x0D] @ 0x00;

0000002F                           #define RAMSIZE 0x2F
00000020                           #define RAMSTART 0x0020

0020 002F                          #pragma memory RAM [0x4F - RAMSTART] @ RAMSTART;
004F 0000                          #pragma memory LOCAL[0] @ 0x4F;

000007FC                           #define ROMSIZE 2048-4
00003800                           #define ROMSTART 0x3800
3800 07FC                          #pragma memory ROM [ROMSIZE] @ ROMSTART;


                                   /* Interrupt vectors */

3FFD                               #pragma vector __RESET @ 0x3FFD;


                                   /* PORTA data bits */

0010                               #pragma portrw PTAD @ 0x0010;
0010                               #pragma portrw PORTA @ 0x0010;
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  3


00000005                        #define PTAD5   5
00000004                        #define PTAD4   4
00000003                        #define PTAD3   3
                                //PTAD bit 2 read only
00000002                        #define PTAD2   2
00000001                        #define PTAD1   1
00000000                        #define PTAD0   0

                                /* PORTA data direction */

0011                            #pragma portrw PTADD @ 0x0011;
00000005                        #define PTADD5  5
00000004                        #define PTADD4  4
00000001                        #define PTADD1  1
00000000                        #define PTADD0  0

                                /* Analog comparator status and control */

0013                            #pragma portrw ACMPSC @ 0x0013;
00000007                        #define ACME    7
00000006                        #define ACBGS   6
00000006                        #define ACF 6
00000004                        #define ACIE    4
                                //ACO bit 3 read only
00000003                        #define ACO 3
00000002                        #define ACOPE   2
                                //ACMOD bits 0-1
00000000                        #define ACMOD   0

                                /* Internal clock source control register 1 */

0014                            #pragma portrw ICSC1 @ 0x0014;
00000006                        #define CLKS    6
00000000                        #define IREFSTEN    0

                                /* Internal clock source control register 2 */

0015                            #pragma portrw ICSC2 @ 0x0015;
                                //BDIV bits 6-7
00000006                        #define BDIV    6
00000003                        #define LP  3

                                /* Internal clock source trim register */

0016                            #pragma portrw ICSTRM @ 0x0016;

                                /* Internal clock source status and control */

0017                            #pragma portrw ICSSC @ 0x0017;
00000002                        #define CLKST   2
00000000                        #define FTRIM   0

                                /* Modulo timer status and control */

0018                            #pragma portrw MTIMSC @ 0x0018;
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  4


                                //TOF bit 7 read only
00000007                        #define TOF 7
00000006                        #define TOIE    6
                                //TRST bit 5 write only
00000005                        #define TRST    5
00000004                        #define TSTP    4

                                /* Modulo timer clock configuration */

0019                            #pragma portrw MTIMCLK @ 0x0019;
                                //CLKSRS bits 4-5
                                //changed from datasheet due to conflict
00000004                        #define CLKSRC  4
                                //PS bits 0-3
00000000                        #define PS  0

                                /* Modulo timer clock counter */

001A                            #pragma portr MTIMCNT @ 0x001A;

                                /* Modulo timer modulo register */

001B                            #pragma portrw MTIMMOD @ 0x001B;

                                /* Keyboard interrupt status and control */

001C                            #pragma portrw KBISC @ 0x001C;
                                //KBF bit 3 read only
00000003                        #define KBF 3
                                //KBACK bit 2 write only
00000002                        #define KBACK   2
00000001                        #define KBIE    1
00000000                        #define KBIMOD  0

                                /* Keyboard interrupt pin enable */

001D                            #pragma portrw KBIPE @ 0x001D;
00000005                        #define KBIPE5  5
00000004                        #define KBIPE4  4
00000002                        #define KBIPE2  2
00000001                        #define KBIPE1  1
00000000                        #define KBIPE0  0

                                /* Keyboard interrupt edge select */

001E                            #pragma portrw KBIES @ 0x001E;
00000005                        #define KBEDG5  5
00000004                        #define KBEDG4  4
00000002                        #define KBEDG2  2
00000001                        #define KBEDG1  1
00000000                        #define KBEDG0  0

                                /* Page register */

001F                            #pragma portrw PAGE @ 0x001F;
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  5


                                /* System reset status */

0200                            #pragma portrw SRS @ 0x0200;
00000007                        #define POR 7
00000006                        #define PIN 6
00000005                        #define COP 5
00000004                        #define ILOP    4
00000003                        #define ILAD    3
00000001                        #define LVD 1
0007                            #define CLEAR_COP_TIMER() (SRS=0xFF)

                                /* System options */

                                //Note: write-once register
0201                            #pragma portrw SOPT @ 0x0201;
00000007                        #define COPE    7
00000006                        #define COPT    6
00000005                        #define STOPE   5
00000001                        #define BKGDPE  1
00000000                        #define RSTPE   0

                                /* System interrupt pending */

0202                            #pragma portr SIP1 @ 0x0202;
00000004                        #define KBI 4
00000003                        #define ACMP    3
00000002                        #define MTIM    2
00000001                        #define RTI 1
                                //changed from LVD due to conflict
00000000                        #define LVDI    0
                                /* For development
                                #pragma thread __KBI()  INTERRUPT (SIP1.KBI );
                                #pragma thread __ACMP() INTERRUPT (SIP1.ACMP);
                                #pragma thread __MTIM() INTERRUPT (SIP1.MTIM);
                                #pragma thread __RTI()  INTERRUPT (SIP1.RTI );
                                #pragma thread __LVDI() INTERRUPT (SIP1.LVDI);
                                */

                                /* System identification (high) */

0206                            #pragma portr SDIDH @ 0x0206;
0008                            #define SDREV() ((SDIDH>>4)&0x0F)
                                //ID bits 0-3

                                /* System identification (low) */

0207                            #pragma portr SDIDL @ 0x0207;
0009                            #define SDID() ( ( ((unsigned int16)(SDIDH)) << 8 ) | (unsigned
int16)(SDIDL) )

                                /* System real-time interrupt status and control */

0208                            #pragma portrw SRTISC @ 0x0208;
                                //RTIF bit 7 read only
00000007                        #define RTIF    7
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  6


                                //RTACK bit 6 write only
00000006                        #define RTIACK  6
00000005                        #define RTICLKS 5
00000004                        #define RTIE    4
                                //RTIS bits 0-2
00000000                        #define RTIS    0

                                /* System power management status and control 1 */

0209                            #pragma portrw SPMSC1 @ 0x0209;
                                //LVDF bit 7 read only
00000007                        #define LVDF    7
                                //LVDACK bit 6 write only
00000006                        #define LVDACK  6
00000005                        #define LVDIE   5
                                //LVDRE bit 4 write once
00000004                        #define LVDRE   4
00000003                        #define LVDSE   3
                                //LVDE bit 2 write once
00000002                        #define LVDE    2
00000000                        #define BGBE    0

                                /* Flash options */

0210                            #pragma portr FOPT @ 0x0210;
00000000                        #define SECD    0

                                /* Flash control register */

0211                            #pragma portrw FLCR @ 0x0211;
00000003                        #define HVEN    3
00000002                        #define MASS    2
00000000                        #define PGM 0

                                /* Pullup/pulldown enable */

0220                            #pragma portrw PTAPE @ 0x0220;
00000005                        #define PTAPE5  5
00000004                        #define PTAPE4  4
00000002                        #define PTAPE2  2
00000001                        #define PTAPE1  1
00000000                        #define PTAPE0  0

                                /* Pullup/pulldown control */

0221                            #pragma portrw PTAPUD @ 0x0221;
00000005                        #define PTAPUD5 5
00000004                        #define PTAPUD4 4
00000002                        #define PTAPUD2 2
00000001                        #define PTAPUD1 1
00000000                        #define PTAPUD0 0

                                /* Slew rate enable */

0222                            #pragma portrw PTASE @ 0x0222;
```

```
C6808 "C" COMPILER 0.0.4.0                                  PAGE  7


00000005                            #define PTASE5  5
00000004                            #define PTASE4  4
00000003                            #define PTASE3  3
00000001                            #define PTASE1  1
00000000                            #define PTASE0  0

                                    #endif /* __C6808DEF_H */
                                    #include "fuzzc.h"
000000FF                            #define F_ONE      0xff
00000000                            #define F_ZERO     0x00
000A                                #define F_OR(a,b)  ((a) > (b) ? (a) : (b))
000B                                #define F_AND(a,b) ((a) < (b) ? (a) : (b))
000C                                #define F_NOT(a)   (F_ONE+F_ZERO-a)
0020 0002                           char __IDOM[2];
                                    #pragma has RS08;
                                    /* Fuzz-C Fuzzy Logic Preprocessor

                                       Fuzzy PID controller
                                       Copyright 2005 Byte Craft Limited. All rights reserved
                                       This code may be adapted for any purpose when used with the
Fuzz-C
                                       fuzzy logic preprocessor. No warranty is implied or given as
to
                                       its usability for any purpose.
                                       Byte Craft Limited
                                       A2-490 Dutton Drive, Waterloo, ON, Canada, N2L 6H7
                                       VOICE: 1 (519) 888 6911
                                       FAX  : 1 (519) 746 6751
                                       email: support@bytecraft.com
                                       Author: Walter Banks
                                       Revision History
                                       dd mmm yyyy  ii  t
                                              1993  WB  First version
                                          Oct 2005  KZ  Comments

                                       This program implements a fuzzy PID
                                       (Proportional-Integral-Derivative) controller.
                                    Overlapping linguistic variables are fine
                                    and assure smooth transitions between dominating FUZZY rules.
                                    For example in the following fuzzy PID
                                    controller we conditioned other rules by characterizing
                                    where we were overall with 4 linguistic variables LNegative
                                    (large negative) normal, close, LPositive .  It is interesting
                                    to note that close (to the setpoint) is entirely contained
                                    in normal. By partitioning the problem this way we
                                    reduced the rule set to a half dozen rules to implement
                                    non linear fuzzy based  PID controller.
                                    The plots and PID reference source code follows
                                    */

0022 0023 0024                       int OldError,SumError,Setpoint;
3800 B6 01      LDA   $01            int Process(void) { return 1; } // Dummy Process
3802 BE         RTS
                                    /*  LINGUISTIC Error  TYPE int  MIN -90  MAX 90 */
                                    /*   { */
0025                                int  Error ;
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  8

                                    /*   MEMBER LNegative   { -90, -90, -20,  0  } */
                                    /*
                                      1-| .............
                                        | .            .
                                        | .             .
                                        | .              .
                                      0-| .               ..................
                                         ------------------------------------
                                        -90     -45       0      45      90

                                    */
                                    char Error_LNegative (int __CRISP)
004E                                  {
3803 B7 45      STA   $45
                                        {
3805 A1 EC      CMP   #$EC              if (__CRISP <= -20) return(255);
3807 B6 FF      LDA   $023F
3809 BE         RTS

                                        else
                                          {
380A B6 45      LDA   $45               if (__CRISP <= 0)
380C A1 00      CMP   #$00
380E 43         COMA                      return(( - __CRISP * 12) + 7);
380F 4C         INCA
3810 3E 0C 0F   LDX   #$0C
3813 BD 39 DF   JSR   $39DF
3816 AB 07      ADD   #$07
3818 BE         RTS

                                          else
3819 4F         CLRA                      return(0);
381A BE         RTS

                                          }
                                        }
                                      }
                                    /*   MEMBER normal      { -20,   0,  20      } */
                                    /*

                                      1-|                  .
                                        |                 . .
                                        |                .   .
                                        |               .     .
                                      0-| ..............       ..............
                                         ------------------------------------
                                        -90     -45       0      45      90

                                    */
                                    char Error_normal (int __CRISP)
004E 004D                             {
381B B7 4D      STA   $4D
381D A1 EC      CMP   #$EC            if (__CRISP < -20) return(0);
381F 4F         CLRA
3820 BE         RTS

                                      else
                                        {
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE  9

3821 B6 4D      LDA   $4D              if (__CRISP <= 0) return(((__CRISP + 20) * 12) + 7);
3823 A1 00      CMP   #$00
3825 B6 14      LDA   $14
3827 BB 4D      ADD   $4D
3829 3E 0C 0F   LDX   #$0C
382C BD 39 DF   JSR   $39DF
382F AB 07      ADD   #$07
3831 BE         RTS
                                      else
                                        {
                                          {
3832 B6 4D      LDA   $4D               if (__CRISP <= 20)
3834 A1 14      CMP   #$14
3836 B6 14      LDA   $14                 return((( + 20 - __CRISP) * 12) + 7);
3838 B0 4D      SUB   $4D
383A 3E 0C 0F   LDX   #$0C
383D BD 39 DF   JSR   $39DF
3840 AB 07      ADD   #$07
3842 BE         RTS
                                          else
3843 4F         CLRA                      return(0);
3844 BE         RTS
                                          }
                                        }
                                      }
                                    }
                                    /*   MEMBER close       {  -3,   0,   3      } */
                                    /*

                                      1-|                 .
                                        |                 .
                                        |                 ..
                                        |                . .
                                      0-| ................. .................
                                         -----------------------------------
                                        -90     -45       0      45      90

                                    */
                                    char Error_close (int __CRISP)
                                      {
004E 004D
3845 B7 4D      STA   $4D
3847 A1 FD      CMP   #$FD            if (__CRISP < -3) return(0);
3849 4F         CLRA
384A BE         RTS
                                      else
                                        {
384B B6 4D      LDA   $4D               if (__CRISP <= 0) return((__CRISP + 3) * 85);
384D A1 00      CMP   #$00
384F B6 03      LDA   $03
3851 BB 4D      ADD   $4D
3853 3E 55 0F   LDX   #$55
3856 BD 39 DF   JSR   $39DF
3859 BE         RTS
                                        else
                                          {
```

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE 10


                                          {
385A B6 4D      LDA   $4D                  if (__CRISP <= 3)
385C A1 03      CMP   #$03
385E B6 03      LDA   $03                    return(( + 3 - __CRISP) * 85);
3860 B0 4D      SUB   $4D
3862 3E 55 0F   LDX   #$55
3865 BD 39 DF   JSR   $39DF
3868 BE         RTS

                                          else
3869 4F         CLRA                       return(0);
386A BE         RTS

                                          }
                                        }
                                      }
                                    }
                                    /*   MEMBER LPositive   {   0,  20,  90,  90 } */
                                    /*

                                      1-|                      .............
                                        |                    .             .
                                        |                   .              .
                                        |                  .               .
                                      0-| .................                .
                                         ----------------------------------
                                        -90     -45       0      45      90

                                    */
                                    char Error_LPositive (int __CRISP)
004E 004D                             {
386B B7 4D      STA   $4D
386D 0F 4D 02   BRCLR 7,$4D,$3872     if (__CRISP < 0) return(0);
3870 4F         CLRA
3871 BE         RTS

                                      else
                                        {
3872 B6 4D      LDA   $4D               if (__CRISP <= 20) return((__CRISP * 12) + 7);
3874 A1 14      CMP   #$14
3876 3E 0C 0F   LDX   #$0C
3879 BD 39 DF   JSR   $39DF
387C AB 07      ADD   #$07
387E BE         RTS

                                        else
                                          {
387F B6 FF      LDA   $023F               return(255);
3881 BE         RTS

                                          }
                                        }
                                      }
                                    /*   } */
                                    /*

                                         Fuzzy Sets for Error

                                      1-| .............   .   .............
                                        | .            . ... .            .
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 11

                                     |  .             . ...             .
                                     |  .            . . . .            .
                                   0-| .............  ...  .............
                                      -----------------------------------
                                     -90     -45       0      45      90


                                   */
                                   /*  LINGUISTIC DeltaError  TYPE int  MIN -90  MAX 90 */
                                   /*   { */
0026                               int  DeltaError ;
                                   /*   MEMBER Negative    { -90, -90, -10,  0  } */
                                   /*

                                     1-| ...............
                                     |  .              .
                                     |  .              .
                                     |  .               .
                                   0-| .                ..................
                                      -----------------------------------
                                     -90     -45       0      45      90

                                   */
                                   char DeltaError_Negative (int __CRISP)
                                     {
004E 004D
3882 B7 4D      STA   $4D
                                       {
3884 A1 F6      CMP   #$F6             if (__CRISP <= -10) return(255);
3886 B6 FF      LDA   $023F
3888 BE         RTS

                                       else
                                         {
3889 B6 4D      LDA   $4D              if (__CRISP <= 0)
388B A1 00      CMP   #$00
388D 43         COMA                     return(( - __CRISP * 25) + 2);
388E 4C         INCA
388F 3E 19 0F   LDX   #$19
3892 BD 39 DF   JSR   $39DF
3895 AB 02      ADD   #$02
3897 BE         RTS

                                         else
3898 4F         CLRA                     return(0);
3899 BE         RTS
                                         }
                                       }
                                     }
                                   /*   MEMBER Positive    {   0,  10,  90,  90 } */
                                   /*

                                     1-|                   ...............
                                     |                     .             .
                                     |                    .              .
                                     |                    .              .
                                   0-| .................                .
                                      -----------------------------------
```

```
C6808 "C" COMPILER 0.0.4.0                                          PAGE 12

                                       -90     -45       0      45      90

                                    */
                                    char DeltaError_Positive (int __CRISP)
004E 004D                             {
389A B7 4D      STA   $4D
389C 0F 4D 02   BRCLR 7,$4D,$38A1     if (__CRISP < 0) return(0);
389F 4F         CLRA
38A0 BE         RTS
                                      else
                                        {
38A1 B6 4D      LDA   $4D               if (__CRISP <= 10) return((__CRISP * 25) + 2);
38A3 A1 0A      CMP   #$0A
38A5 3E 19 0F   LDX   #$19
38A8 BD 39 DF   JSR   $39DF
38AB AB 02      ADD   #$02
38AD BE         RTS
                                        else
                                          {
38AE B6 FF      LDA   $023F                 return(255);
38B0 BE         RTS
                                          }
                                        }
                                      }
                                    /*   } */
                                    /*

                                         Fuzzy Sets for DeltaError

                                      1-| ...............   ................
                                        | .              .  .             .
                                        | .              . .              .
                                        | .               ..              .
                                      0-| ................................
                                        ----------------------------------
                                        -90     -45       0      45      90


                                    */
                                    /*  LINGUISTIC SumError  TYPE int  MIN -90  MAX 90 */
                                    /*   { */
                                    /*   MEMBER LNeg    { -90, -90,  -5,  0  } */
                                    /*

                                      1-| ................
                                        | .              .
                                        | .               .
                                        | .               .
                                      0-| .               ..................
                                        ----------------------------------
                                        -90     -45       0      45      90

                                    */
                                    char SumError_LNeg (int __CRISP)
004E 004D                             {
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 13


38B1 B7 4D      STA   $4D
                                        {
38B3 A1 FB      CMP   #$FB              if (__CRISP <= -5) return(255);
38B5 B6 FF      LDA   $023F
38B7 BE         RTS
                                        else
                                          {
38B8 B6 4D      LDA   $4D               if (__CRISP <= 0)
38BA A1 00      CMP   #$00
38BC 43         COMA                      return( - __CRISP * 51);
38BD 4C         INCA
38BE 3E 33 0F   LDX   #$33
38C1 BD 39 DF   JSR   $39DF
38C4 BE         RTS
                                          else
38C5 4F         CLRA                      return(0);
38C6 BE         RTS
                                          }
                                        }
                                      }
                                    /*   MEMBER LPos    {   0,   5,  90,  90 } */
                                    /*

                                      1-|                  .................
                                        |                  .               .
                                        |                  .               .
                                        |                 .                .
                                      0-| .................                .
                                         ----------------------------------
                                        -90     -45       0      45      90

                                    */
                                    char SumError_LPos (int __CRISP)
004E 004D                             {
38C7 B7 4D      STA   $4D
38C9 0F 4D 02   BRCLR 7,$4D,$38CE     if (__CRISP < 0) return(0);
38CC 4F         CLRA
38CD BE         RTS
                                      else
                                        {
38CE B6 4D      LDA   $4D               if (__CRISP <= 5) return(__CRISP * 51);
38D0 A1 05      CMP   #$05
38D2 3E 33 0F   LDX   #$33
38D5 BD 39 DF   JSR   $39DF
38D8 BE         RTS
                                        else
                                          {
38D9 B6 FF      LDA   $023F               return(255);
38DB BE         RTS
                                          }
                                        }
                                      }
                                    /*   } */
                                    /*
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 14

                                     Fuzzy Sets for SumError

                                   1-| ................. .................
                                     | .               . .               .
                                     | .                ..               .
                                     | .                .                .
                                   0-| ..................................
                                      -----------------------------------
                                     -90     -45       0      45      90


                                   */
                                   /*  CONSEQUENCE  ManVar TYPE int   MIN -20  MAX 20 DEFUZZ cg */
                                   /*  { */
0027                               int  ManVar ;
0028 0029                            int fa_ManVar, fc_ManVar;
                                   /*      MEMBER  LNegative  {  -18 } */
                                   /*

                                     1-|   .
                                       |   .
                                       |   .
                                       |   .
                                     0-| ..*...............................
                                       -----------------------------------
                                      -20     -10       0      10      20


                                   */
                                   void ManVar_LNegative (int __DOM)
004E 004D                            {
38DC B7 4D      STA   $4D
38DE B6 29      LDA   $29                fc_ManVar += __DOM;
38E0 BB 4D      ADD   $4D
38E2 B7 29      STA   $29
38E4 B6 4D      LDA   $4D                fa_ManVar += (__DOM * (-18));
38E6 3E EE 0F   LDX   #$EE
38E9 BD 39 DF   JSR   $39DF
38EC BB 28      ADD   $28
38EE B7 28      STA   $28
38F0 BE         RTS                    }
                                   /*      MEMBER  SNegative  {   -6 } */
                                   /*

                                     1-|            .
                                       |            .
                                       |            .
                                       |            .
                                     0-| ...........*......................
                                       -----------------------------------
                                      -20     -10       0      10      20


                                   */
                                   void ManVar_SNegative (int __DOM)
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 15


004E 004D                               {
38F1 B7 4D      STA   $4D
38F3 B6 29      LDA   $29                   fc_ManVar += __DOM;
38F5 BB 4D      ADD   $4D
38F7 B7 29      STA   $29
38F9 B6 4D      LDA   $4D                   fa_ManVar += (__DOM * (-6));
38FB 3E FA 0F   LDX   #$FA
38FE BD 39 DF   JSR   $39DF
3901 BB 28      ADD   $28
3903 B7 28      STA   $28
3905 BE         RTS                     }
                                      /*      MEMBER  SPositive  {    6 } */
                                      /*

                                        1-|                     .
                                          |                     .
                                          |                     .
                                          |                     .
                                        0-| ......................*...........
                                          ------------------------------------
                                           -20     -10       0      10      20


                                      */
                                      void ManVar_SPositive (int __DOM)
004E 004D                               {
3906 B7 4D      STA   $4D
3908 B6 29      LDA   $29                   fc_ManVar += __DOM;
390A BB 4D      ADD   $4D
390C B7 29      STA   $29
390E B6 4D      LDA   $4D                   fa_ManVar += (__DOM * (6));
3910 3E 06 0F   LDX   #$06
3913 BD 39 DF   JSR   $39DF
3916 BB 28      ADD   $28
3918 B7 28      STA   $28
391A BE         RTS                     }
                                      /*      MEMBER  LPositive  {   18 } */
                                      /*

                                        1-|                               .
                                          |                               .
                                          |                               .
                                          |                               .
                                        0-| ..............................*..
                                          ------------------------------------
                                           -20     -10       0      10      20


                                      */
                                      void ManVar_LPositive (int __DOM)
004E 004D                               {
391B B7 4D      STA   $4D
391D B6 29      LDA   $29                   fc_ManVar += __DOM;
391F BB 4D      ADD   $4D
3921 B7 29      STA   $29
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 16

3923 B6 4D      LDA   $4D                fa_ManVar += (__DOM * (18));
3925 3E 12 0F   LDX   #$12
3928 BD 39 DF   JSR   $39DF
392B BB 28      ADD   $28
392D B7 28      STA   $28
392F BE         RTS                    }
                                      /*  } */
                                      /*  FUZZY pid */
                                      void pid (void)
                                        {
3930 3F 28      CLR   $28                 fa_ManVar = 0;
3932 3F 29      CLR   $29                 fc_ManVar = 0;
                                      /*    { */
                                      /*     IF Error IS LNegative THEN ManVar IS LPositive */
3934 B6 25      LDA   $25               ManVar_LPositive( Error_LNegative(Error)  );
3936 BD 38 03   JSR   $3803
3939 AD E0      BSR   $391B
                                      /*     IF Error IS LPositive THEN ManVar IS LNegative */
393B B6 25      LDA   $25               ManVar_LNegative( Error_LPositive(Error)  );
393D BD 38 6B   JSR   $386B
3940 AD 9A      BSR   $38DC
                                      /*     IF Error IS normal AND DeltaError IS Positive */
                                      /*       THEN ManVar IS SNegative */
3942 B6 25      LDA   $25               __IDOM[1] = Error_normal(Error) ;
3944 BD 38 1B   JSR   $381B
3947 B7 21      STA   $21
3949 B6 26      LDA   $26               __IDOM[0] = DeltaError_Positive(DeltaError) ;
394B BD 38 9A   JSR   $389A
394E B7 20      STA   $20
3950 B6 21      LDA   $21               __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
3952 B0 20      SUB   $20
3954 34 04      BCC   $395A
3956 B6 21      LDA   $21
3958 B7 20      STA   $20
395A B6 20      LDA   $20               ManVar_SNegative( __IDOM[0] );
395C AD 93      BSR   $38F1
                                      /*     IF Error IS normal AND DeltaError IS Negative */
                                      /*       THEN ManVar IS SPositive */
395E B6 25      LDA   $25               __IDOM[1] = Error_normal(Error) ;
3960 BD 38 1B   JSR   $381B
3963 B7 21      STA   $21
3965 B6 26      LDA   $26               __IDOM[0] = DeltaError_Negative(DeltaError) ;
3967 BD 38 82   JSR   $3882
396A B7 20      STA   $20
396C B6 21      LDA   $21               __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
396E B0 20      SUB   $20
3970 34 04      BCC   $3976
3972 B6 21      LDA   $21
3974 B7 20      STA   $20
3976 B6 20      LDA   $20               ManVar_SPositive( __IDOM[0] );
3978 AD 8C      BSR   $3906
                                      /*     IF Error IS close AND SumError IS LPos */
                                      /*       THEN ManVar IS SNegative */
397A B6 25      LDA   $25               __IDOM[1] = Error_close(Error) ;
397C BD 38 45   JSR   $3845
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 17

397F B7 21      STA   $21
3981 B6 23      LDA   $23             __IDOM[0] = SumError_LPos(SumError) ;
3983 BD 38 C7   JSR   $38C7
3986 B7 20      STA   $20
3988 B6 21      LDA   $21             __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
398A B0 20      SUB   $20
398C 34 04      BCC   $3992
398E B6 21      LDA   $21
3990 B7 20      STA   $20
3992 B6 20      LDA   $20             ManVar_SNegative( __IDOM[0] );
3994 BD 38 F1   JSR   $38F1
                                     /*     IF Error IS close AND SumError IS LNeg */
                                     /*       THEN ManVar IS SPositive */
3997 B6 25      LDA   $25             __IDOM[1] = Error_close(Error) ;
3999 BD 38 45   JSR   $3845
399C B7 21      STA   $21
399E B6 23      LDA   $23             __IDOM[0] = SumError_LNeg(SumError) ;
39A0 BD 38 B1   JSR   $38B1
39A3 B7 20      STA   $20
39A5 B6 21      LDA   $21             __IDOM[0] = F_AND(__IDOM[1],__IDOM[0]);
39A7 B0 20      SUB   $20
39A9 34 04      BCC   $39AF
39AB B6 21      LDA   $21
39AD B7 20      STA   $20
39AF B6 20      LDA   $20             ManVar_SPositive( __IDOM[0] );
39B1 BD 39 06   JSR   $3906
                                     /*    } */
39B4 36 03      BNE   $39B9              if (fc_ManVar == 0)
39B6 3F 27      CLR   $27                 ManVar = 0;
39B8 BE         RTS                      else
39B9 B6 28      LDA   $28                 ManVar = fa_ManVar / fc_ManVar;
39BB 4E 29 0F   LDX   $29
39BE 8C         CLR   $0C
39BF 52         DEC   $02
39C0 B7 27      STA   $27
39C2 BE         RTS                    }
                                      void main (void)
                                        {
                                         while(1)
39C3 B6 25      LDA   $25                  {
39C5 B7 22      STA   $22                   OldError = Error;
39C7 BD 38 00   JSR   $3800                 Error = Setpoint - Process();
39CA 43         COMA
39CB 4C         INCA
39CC BB 24      ADD   $24
39CE B7 25      STA   $25
39D0 B0 22      SUB   $22                   DeltaError = Error - OldError;
39D2 B7 26      STA   $26
39D4 B6 23      LDA   $23                   SumError = SumError + Error;
39D6 BB 25      ADD   $25
39D8 B7 23      STA   $23
39DA BD 39 30   JSR   $3930                 pid();
39DD 30 E4      BRA   $39C3                }
                                         }
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 18


39DF B7 47      STA   $47
39E1 A6 08      LDA   #$08
39E3 B7 46      STA   $46
39E5 4F         CLRA
39E6 48         LSLA
39E7 59         DEC   $09
39E8 24         INC   $04
39E9 05 BB 47   BRCLR 2,$BB,$3A33
39EC 24         INC   $04
39ED 01 5C 3A   BRCLR 0,$5C,$3A2A
39F0 46         RORA
39F1 26         INC   $06
39F2 F3         STA   $13
39F3 81         CLR   $01
                  __MAIN:
3FFD BC 39 C3
```

C6808 "C" COMPILER 0.0.4.0 PAGE 19

SYMBOL TABLE

LABEL	VALUE	LABEL	VALUE
ACBGS	0006	ACF	0006
ACIE	0004	ACME	0007
ACMOD	0000	ACMP	0003
ACMPSC	0013	ACO	0003
ACOPE	0002	BDIV	0006
BGBE	0000	BKGDPE	0001
CLKS	0006	CLKSRC	0004
CLKST	0002	COP	0005
COPE	0007	COPT	0006
DeltaError	0026	DeltaError_Negative	3882
DeltaError_Positive	389A	Error	0025
Error_LNegative	3803	Error_LPositive	386B
Error_close	3845	Error_normal	381B
FLCR	0211	FOPT	0210
FTRIM	0000	F_ONE	00FF
F_ZERO	0000	HVEN	0003
ICSC1	0014	ICSC2	0015
ICSSC	0017	ICSTRM	0016
ILAD	0003	ILOP	0004
IREFSTEN	0000	KBACK	0002
KBEDG0	0000	KBEDG1	0001
KBEDG2	0002	KBEDG4	0004
KBEDG5	0005	KBF	0003
KBI	0004	KBIE	0001
KBIES	001E	KBIMOD	0000
KBIPE	001D	KBIPE0	0000
KBIPE1	0001	KBIPE2	0002
KBIPE4	0004	KBIPE5	0005
KBISC	001C	LOCAL	0005
LP	0003	LVD	0001
LVDACK	0006	LVDE	0002
LVDF	0007	LVDI	0000
LVDIE	0005	LVDRE	0004
LVDSE	0003	MASS	0002
MTIM	0002	MTIMCLK	0019
MTIMCNT	001A	MTIMMOD	001B
MTIMSC	0018	ManVar	0027
ManVar_LNegative	38DC	ManVar_LPositive	391B
ManVar_SNegative	38F1	ManVar_SPositive	3906
OldError	0022	PAGE	001F
PGM	0000	PIN	0006
POR	0007	PORTA	0010
PS	0000	PTAD	0010
PTAD0	0000	PTAD1	0001
PTAD2	0002	PTAD3	0003
PTAD4	0004	PTAD5	0005
PTADD	0011	PTADD0	0000
PTADD1	0001	PTADD4	0004
PTADD5	0005	PTAPE	0220

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 20


SYMBOL TABLE - Continued

LABEL                          VALUE  LABEL                          VALUE


PTAPE0                         0000 | PTAPE1                         0001
PTAPE2                         0002 | PTAPE4                         0004
PTAPE5                         0005 | PTAPUD                         0221
PTAPUD0                        0000 | PTAPUD1                        0001
PTAPUD2                        0002 | PTAPUD4                        0004
PTAPUD5                        0005 | PTASE                          0222
PTASE0                         0000 | PTASE1                         0001
PTASE3                         0003 | PTASE4                         0004
PTASE5                         0005 | Process                        3800
RAMSIZE                        002F | RAMSTART                       0020
ROMSIZE                        07FC | ROMSTART                       3800
RSTPE                          0000 | RTI                            0001
RTIACK                         0006 | RTICLKS                        0005
RTIE                           0004 | RTIF                           0007
RTIS                           0000 | SDIDH                          0206
SDIDL                          0207 | SECD                           0000
SIP1                           0202 | SOPT                           0201
SPMSC1                         0209 | SRS                            0200
SRTISC                         0208 | STOPE                          0005
Setpoint                       0024 | SumError                       0023
SumError_LNeg                  38B1 | SumError_LPos                  38C7
TOF                            0007 | TOIE                           0006
TRST                           0005 | TSTP                           0004
_SP                            0000 | __CRISP                        0045
__DOM                          004D | __IDOM                         0020
__MAIN                         39C3 | __MUL8x8                       39DF
__RESET                        3FFD | __RESET                        3FFD
__SPAD                         0046 | __longAC                       004C
__longIX                       004A | fa_ManVar                      0028
fc_ManVar                      0029 | main                           39C3
pid                            3930 | tiny                           0003


RAM USAGE MAP

0010 PTAD                          portrw
0010 PORTA                         portrw
0011 PTADD                         portrw
0013 ACMPSC                        portrw
0014 ICSC1                         portrw
0015 ICSC2                         portrw
0016 ICSTRM                        portrw
0017 ICSSC                         portrw
0018 MTIMSC                        portrw
0019 MTIMCLK                       portrw
001A MTIMCNT                       portr
001B MTIMMOD                       portrw
```

```
C6808 "C" COMPILER 0.0.4.0                                      PAGE 21


RAM USAGE MAP - Continued

001C KBISC                          portrw
001D KBIPE                          portrw
001E KBIES                          portrw
001F PAGE                           portrw
0200 SRS                            portrw
0201 SOPT                           portrw
0202 SIP1                           portr
0206 SDIDH                          portr
0207 SDIDL                          portr
0208 SRTISC                         portrw
0209 SPMSC1                         portrw
0210 FOPT                           portr
0211 FLCR                           portrw
0220 PTAPE                          portrw
0221 PTAPUD                         portrw
0222 PTASE                          portrw
0020 __IDOM                         unsigned char[2]
0022 OldError                       signed char
0023 SumError                       signed char
0024 Setpoint                       signed char
0025 Error                          signed char
0026 DeltaError                     signed char
0027 ManVar                         signed char
0028 fa_ManVar                      signed char
0029 fc_ManVar                      signed char


LOCAL RAM USAGE FROM 004F TO 004F


ROM USAGE MAP

    3800 to 39F3    3FFC to 3FFF
    Total ROM used 01F9 (505)


Errors            :    0
Warnings          :    0
```