Application Note

AN2312/D
7/2002

MC68HC908QY4 Internal Oscillator Usage Notes

By Scott Pape
Systems Engineer
8/16 Bit Products Division

Introduction

This document addresses the internal oscillator module for the M68HC908Qx Family of microcontrollers. The internal oscillator is a low cost alternative to crystal and RC oscillators since it requires no external components.

This document will address three main sub-topics of the internal oscillator:

- **Operational Characteristics** which describes the operational characteristics of the oscillator
- **Trimming the Internal Oscillator** which describes a reliable method for trimming the oscillator
- **Low Power Modes** which describes the operation of the oscillator in low power modes

Features of Internal Oscillator

The internal oscillator is simple to operate. It is the default clock source for the microcontroller unit (MCU) after any reset. The frequency is fixed at a base of 12.8 MHz, but can vary from part-to-part by ±25% due to variations in the manufacturing process. It can be trimmed by ±25% to get the frequency back to the 12.8 MHz base frequency. Once trimmed, the oscillator will remain within 5% of the trimmed frequency across the operating temperature and voltage range.

This base frequency of 12.8 MHz is divided by four to create the bus clock which will run at 3.2 MHz when the internal oscillator is trimmed and used as the clock source. This frequency was chosen so that a part running at +25% frequency (i.e., 4.0 MHz), would not run faster than the maximum specified bus frequency at the lowest specified operating voltage, in this case, 4.0 MHz at 2.7 V.

Advantages of the Internal Oscillator

The main advantage of this clock source is that it requires no external components, thereby freeing the pins normally associated with oscillators for other chip functions such as general purpose input/output or analog-to-digital input channels. This also lowers the cost of the application, as no external oscillator component is required. Another advantage of this oscillator is that it starts up very quickly after being powered down. Therefore, this clock option can be used with the short stop recovery option (SSREC) in chip configuration register 1 (CONFIG1) reducing stop recovery times from 4096 oscillator cycles to 32 oscillator cycles. See **Stop Mode** for more information on the use of stop mode with the internal oscillator.

Internal Oscillator versus Internal Clock Generator (ICG)

The internal oscillator differs from the internal clock generator (ICG) found on several other HC08 devices in that there is no phase-locked-loop (PLL) used to generate multiple frequencies. The internal oscillator is fixed to provide a bus clock of 3.2 MHz whereas an ICG can be programmed to run at multiples of the base frequency. In exchange for programmable frequencies, the internal oscillator gains a size advantage. The reduced size makes it an attractive clock source for lower cost MCU's. The lack of programmability also makes the internal oscillator much easier to use, since the only step required is to set the trim register to the correct value and the part is ready to use.

Another advantage is that the simpler design is more power efficient. The internal oscillator has only one major block, the oscillator itself. An ICG is comprised of several blocks, including an internal reference generator, a digitally controlled oscillator, a frequency comparator, and a frequency divider. Together, these blocks use 50% to 100% more current than the internal oscillator.

Operational Characteristics

The operational characteristics include the typical characteristics of the internal oscillator, the effect of operating voltage and temperature on these characteristics and how the trim value relates to the frequency.

Typical Characteristics

The typical characteristics reflect the basic operation of the oscillator at room temperature and 5 volts V_{DD}.

Duty Cycle

The internal oscillator produces a 50% duty cycle and deviates from this by no more than ±2%. In **Figure 1**, the oscillator output has been routed to the PTA4 pin by setting the OSC2EN bit in the Port A Pullup Enable register (PTAPUE). This picture has captured the oscillator output for several seconds so it is composed of well over 20 million samples. Looking at the variation within blue vertical lines, we can see that the falling edge varies by about 3 ns out of a period of 76.8 ns for a total deviation of less than 4%, or ±2% from an average of 50%.

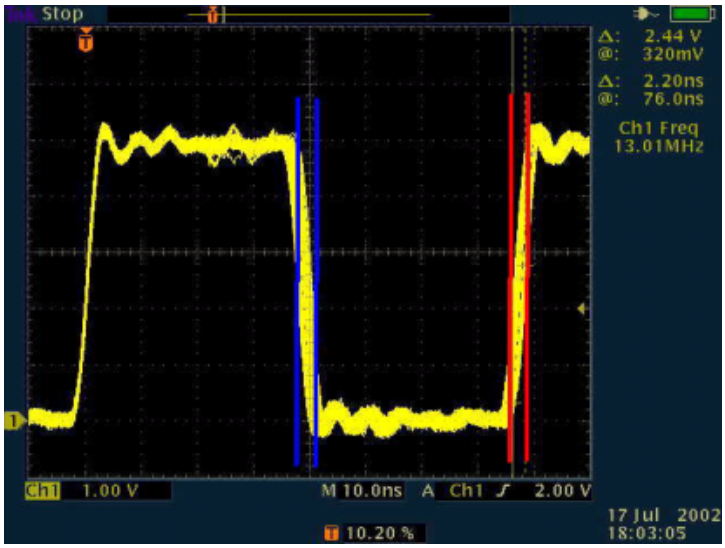


Figure 1. Typical Internal Oscillator Output

Jitter/Stability

Figure 1 shows us the jitter of the internal oscillator. For the purpose of this document, jitter is defined as the percentage variation from the base frequency over an extended period of time. From **Figure 1**, we see that the second rising edge, between the red vertical lines, varies by 2.2 ns. The average frequency is 13.01 MHz for a period of 76.8 ns. This results in a jitter calculation of less than ±2%.

Figure 2 shows the rising edge of a port toggle after 14 bus cycles. Again, millions of samples have been captured over several seconds. We see that this edge, 14 bus cycles after the scope trigger, varies by 3.6 ns, as highlighted, at a period of 4.3 µs. This translates into a jitter of less than 0.1%.

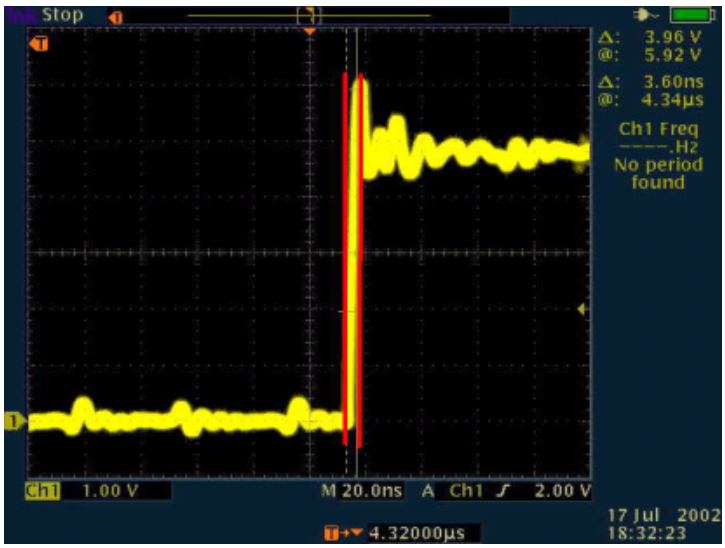


Figure 2. Rising Edge of Port Toggle After 14 Bus Cycles

Figure 3 shows the rising edge of a port toggle after 100 bus cycles. Again, millions of samples have been captured over several seconds. This time the edge varies by 30 ns, as highlighted, at a period of 31 µs which again translates into a jitter of less than 0.1%.

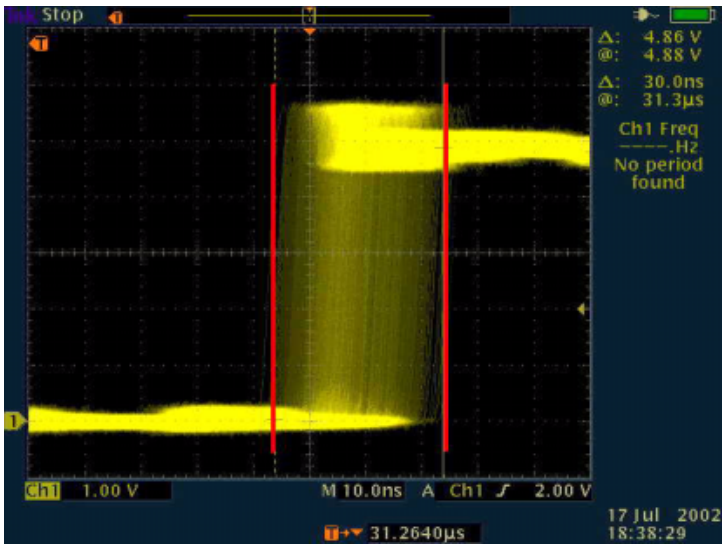


Figure 3. Rising Edge of Port Toggle After 100 Bus Cycles

Startup Characteristics

The internal oscillator has the advantage of a quick startup from a powered off condition. **Figure 4** shows a typical startup sequence of the oscillator from stop mode at 5-V V_{DD}. We can see that the first pulse is shorter than the subsequent pulses. Also, we can see that the oscillator is running slower than the trimmed frequency of 12.8 MHz, starting at roughly 7 MHz. This frequency increases gradually until the trimmed frequency is reached after approximately 6 µs.

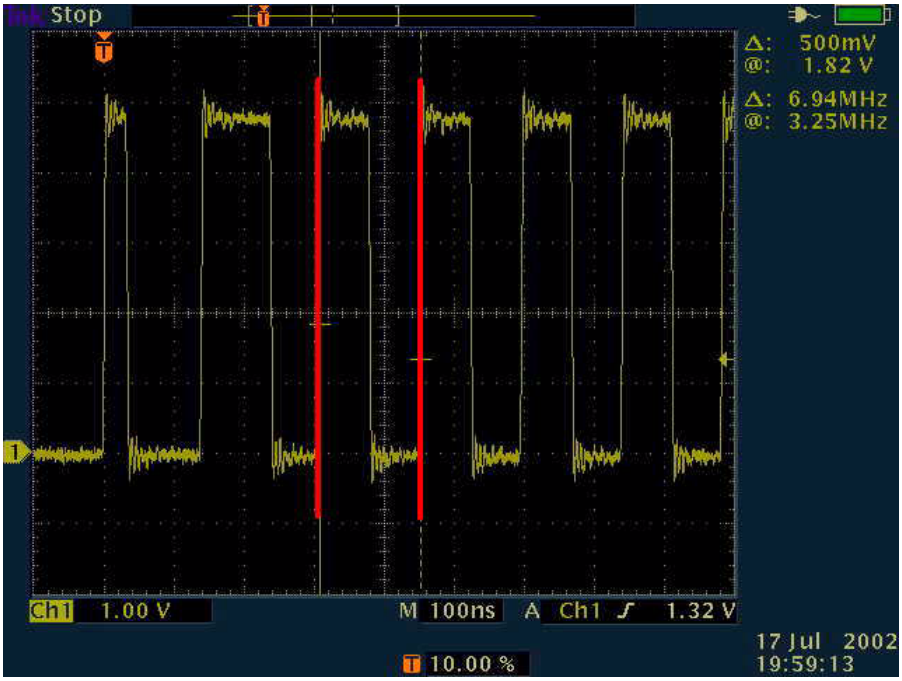


Figure 4. Typical Oscillator Startup Sequence at V_{DD} = 5 V

Figure 5 shows a typical startup sequence of the oscillator from stop mode at 3-V V_{DD}. The waveform is the same shape as at 5 V; however, the startup frequency is about half of what it was at 5 V, 3.7 MHz instead of 7 MHz. This results in the oscillator taking longer to reach the trimmed frequency of 12.8 MHz, approximately 11 µs instead of 6 µs.

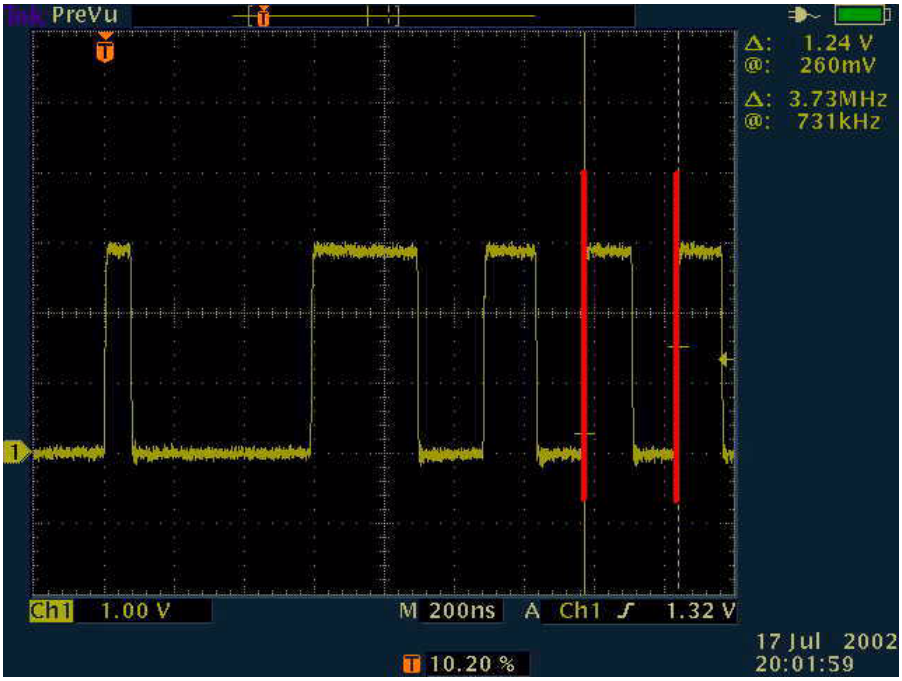


Figure 5. Typical Oscillator Startup Sequence at V_{DD} = 3 V

Frequency versus V_{DD}

Figure 6 shows a graph of bus frequency versus V_{DD} for three typical devices under test (DUTs). We see that the V_{DD} does not have a perfectly linear relation to frequency. This is due to V_{DD} compensation circuits which keep frequency from varying too far from the trimmed value. All three DUTs were trimmed at 5 V to a 3.18 MHz bus frequency. The deviation peaks at 3.5 V before heading back towards the trimmed value. At this peak, the deviation is only 2% from the trimmed value.

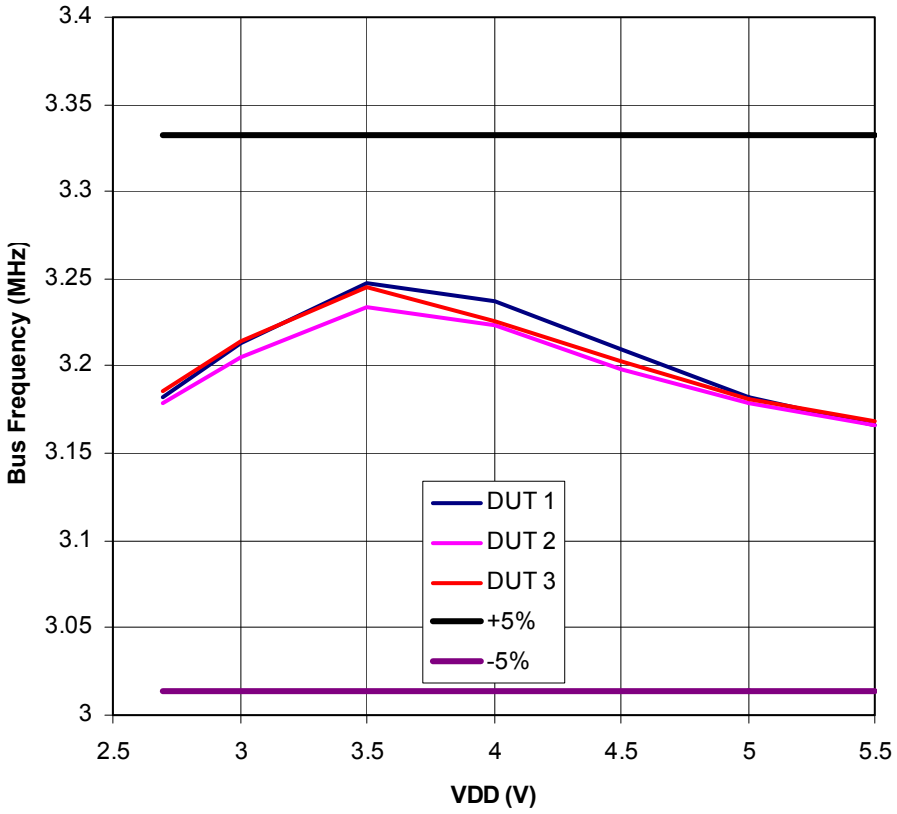


Figure 6. Bus Frequency versus V_{DD}

Frequency versus Temperature

Figure 7 shows a graph of bus frequency versus temperature for three typical DUTs at two different V_{DD}s. These DUTs were trimmed at 5 V and 25°C. We can see that the higher the temperature, the lower the measured frequency. However, all frequency measurements remain within 2% of the trimmed value.

We also see that the frequency tends to be more heavily influenced by V_{DD} at the lower temperatures, with the frequencies clearly grouping into a 3-V group with frequencies around 3.23 MHz and a 5-V group with frequencies around 3.19 MHz. However, as temperatures approach 70°C, we see that for an individual DUT, the difference between the 3-V frequency and the 5-V frequency becomes less and the difference between each DUT becomes more pronounced.

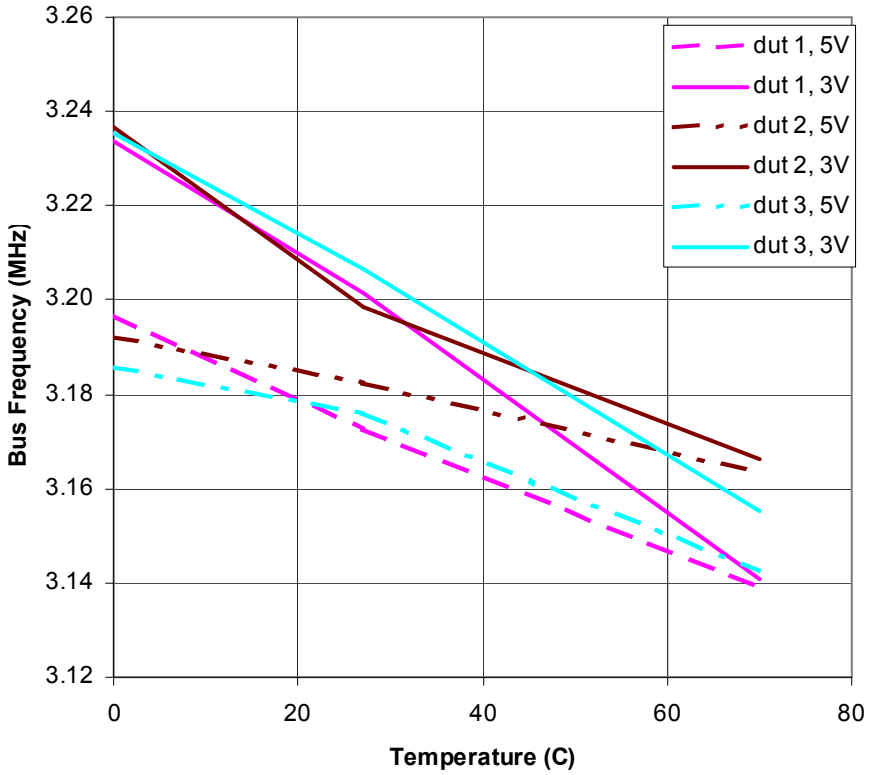


Figure 7. Bus Frequency versus Temperature

Frequency/Period versus Trim Value

The internal oscillator is trimmable by ±25% by writing an 8-bit value to the trim register. This trim value is set to $80 after any reset. Increasing this value by $01, increases the period of the oscillator by approximately 0.2%.

In an ideal circuit, the trim value would be perfectly linear in relation to the clock period. However, due to parasitic effects and wafer fabrication variations, the relationship is not perfectly linear. Also, the circuit board layout and, especially, the placement of the V_{DD} bypass capacitors in relation to the V_{DD}/V_{SS} pins can greatly influence the resulting period.

Figure 8 shows the bus period versus trim value of a typical part with the bypass capacitor placed directly across the V_{DD} and V_{SS} pins and one with no bypass capacitor. At the highest speeds, towards trim value = $00, we see the greatest variation due to the capacitor placement, more than 10% difference between having a capacitor and not having one. Another plot of this chart shows the resulting bus period when the capacitor is moved about 1 inch from the V_{DD}/V_{SS} pins. The shape of the curve changes slightly compared to having the capacitor very close to the pins. This is due to introducing an inductive loop into the power lines to the MCU. Freesca;e recommends placing the bypass capacitor as close to the V_{DD}/V_{SS} pins as possible.

Figure 8 also shows an ideal period plot for the curve with the bypass capacitor placed close. The ideal plot is calculated starting at a trim value of $80. At $80, we take the default bus period of our DUT, in this case, 315 ns. In an ideal device, the period would increase 0.2% for each increment of $01 in the OSCTRIM register. At the worst case, which is the longest periods, the actual period deviates about 4% from the ideal.

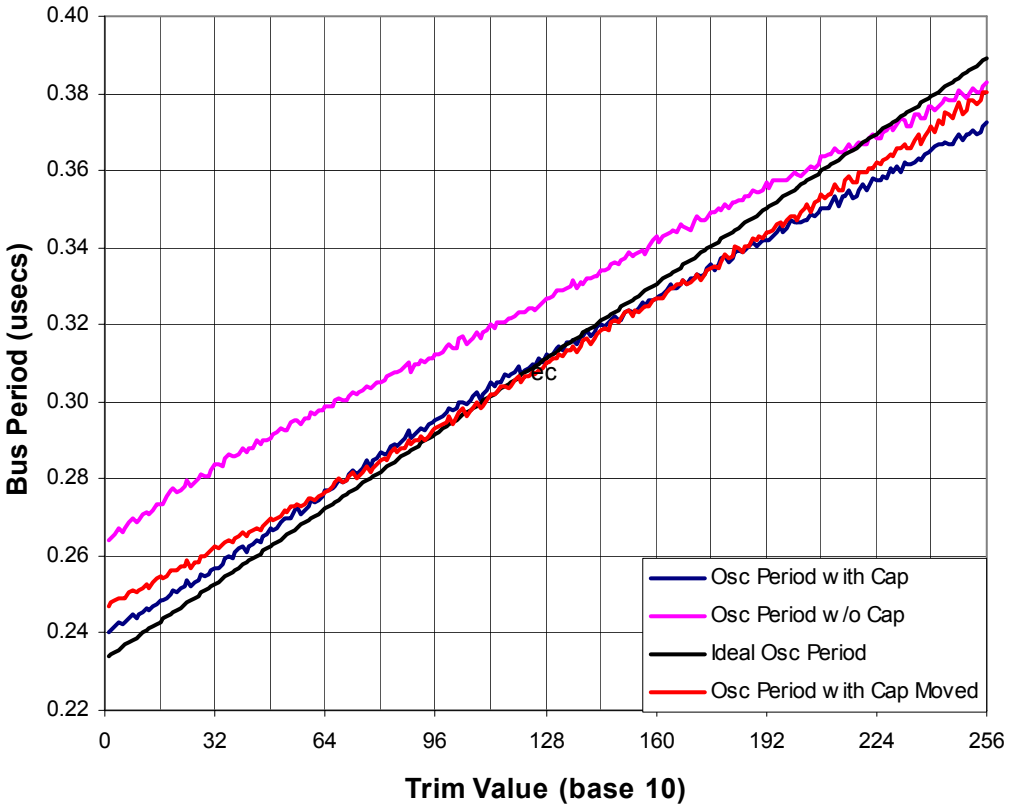


Figure 8. Bus Period versus Trim Value

Also, we see that the period curves have a slight saw-toothed characteristic. Both the saw-tooth and the non-linearity are a result of two characteristics of the internal capacitors, which are switched in and out based on the trim value. First, these internal capacitors do not match each other perfectly due to wafer fabrication irregularities. There is also parasitic capacitance internal to the chip. Both of these characteristics result in a non-ideal curve.

Due to the non-ideal period versus trim value relationship, Freescale recommends that the period be measured after each new trim value written during the trimming process, until the result is within the desired range.

Why Trim is Not Linear versus Frequency

The trim value adjusts the time constant of the oscillator. This time constant is approximately linear compared to the period; therefore, it is inversely related to frequency. **Figure 9** shows the same data as **Figure 8**, except as frequency versus trim value.

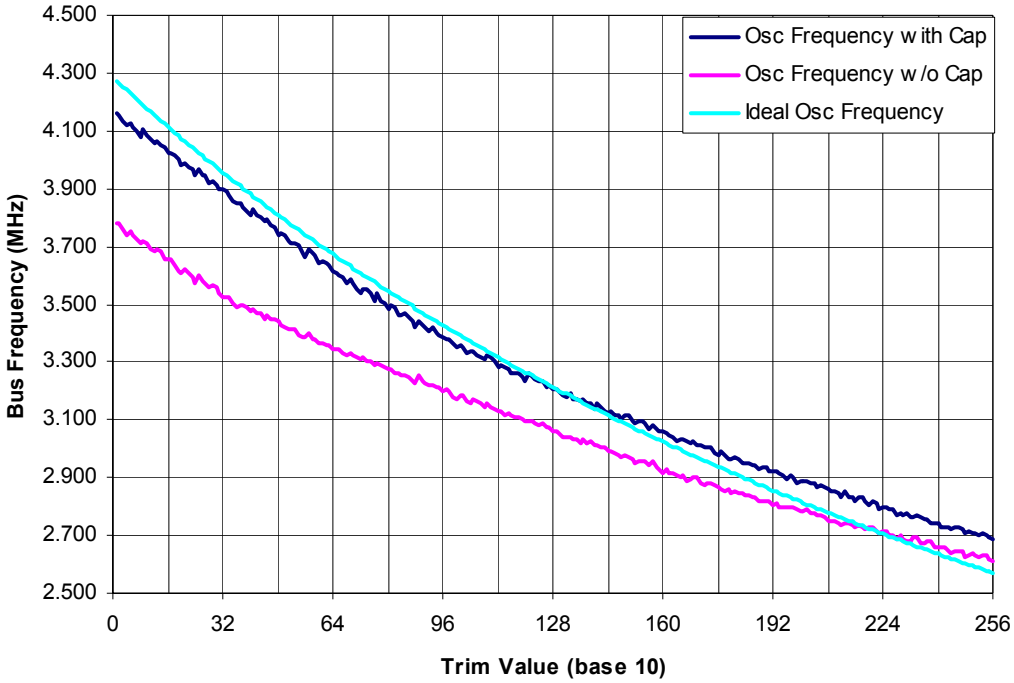


Figure 9. Bus Frequency versus Trim Value

Trimming the Internal Oscillator

The internal oscillator's period is trimmable by ±25% to compensate for wafer fabrication variations that can cause the base period to deviate ±25% from 312.5 ns (3.2 MHz). The oscillator is trimmed by writing a new value to the OSCTRIM register at memory address $38. This register is loaded with the value of $80 after any reset. Writing a higher value creates a longer period and writing a smaller value creates a shorter period. Each increment/decrement of $01 to the current trim value increases/decreases the period by approximately 0.2%.

When the parts are shipped from the factory, they will be programmed with a trim value that will set the frequency to within 5% of 3.2 MHz. The FLASH location where this value is stored is called the TRIMLOC register and is at memory address $FFC0. This location *is not* automatically loaded into the OSCTRIM register in normal user mode. It is up to the user to copy this value into OSCTRIM if needed.

This section will describe in detail a reliable method for trimming the internal oscillator to within 0.4% of 3.2 MHz. Even though the trimmable range is ±25% of the base period, the only frequency that can always be guaranteed is 3.2 MHz. This is due to wafer fabrication variations that can vary the base frequency by ±25%.

Forced Monitor Mode

If forced monitor mode (refer to the *MC68HC908QY4 Data Sheet* — Freescale document order number MC68HC908QY4/D) is entered with the internal oscillator selected, the monitor firmware will copy the value in TRIMLOC into OSCTRIM in order to establish a 9600 baud communication with the host PC. The OSCTRIM value can vary by roughly ± $0F and still maintain communication with the host PC.

Auto-Trimming Methods

Many possible ways exist to trim the internal oscillator using the MCU itself. This document will discuss four different methods and give a detailed example, including assembly code, of one. All of these trim methods require an external reference clock to be fed into either a timer channel or a general purpose input/output (I/O) port. The reference clock should be at least 100 times slower than the internal clock or 32 kHz. If the reference clock is much faster than this, the MCU will not have enough resolution to calculate an accurate trim value.

After finding a suitable trim value, it should be programmed into FLASH memory, preferably into the TRIMLOC register at location $FFC0. As part of the reset routine of the main user code, the TRIMLOC value should be copied into the OSCTRIM register.

Linear Search

The linear search assumes that the oscillator period is linear with respect to the trim value. To calculate the trim value, the period of the reference clock is measured by either using the timer input capture or by counting loop executions for one cycle of the reference. This measured value is compared to an expected value and a percentage difference between the expected and actual values is generated. Since a change of $01 to the trim value represents a change of approximately 0.2% to the period, divide the percentage difference by 0.2% to get a trim offset.

If the actual measurement of the reference was greater than the expected, then the oscillator is running too fast, so slow it down by adding the offset to the current OSCTRIM value. If the actual measurement was lower, then speed the oscillator up by subtracting the offset from the current OSCTRIM value.

This method can be simplified by setting up the timer or counting loop such that a difference of $01 in the counts is equal to a 0.2% difference in frequency.

The drawback to this method is that the oscillator period is not exactly linear to the trim value, as shown in **Frequency/Period versus Trim Value**. Therefore, the result could be off by as much as 5%.

Sequential Search

The sequential search is very straight forward. Simply set OSCTRIM to the highest or lowest value ($FF or $00), measure the period of the reference clock as shown in **Linear Search** and if the measurement does not equal the expected value, decrement or increment OSCTRIM by $01 and try again. Keep repeating until a match is found.

There are two drawbacks to this method. First, on average, it would require 128 iterations before the correct trim value is found. Second, provisions need to be in place in case a perfect match is not found. This can be accomplished by saving the smallest difference between expected and actual measurements and saving the trim value that produced this difference. Doing this requires calculating the smallest difference after each measurement.

Binary Search

The binary search is also straight forward. Starting with the reset value of OSCTRIM, $80, measure the reference clock period and compare to the expected value. If the actual value is higher, the oscillator is running too fast, so slow it down by adding $40 to OSCTRIM. Otherwise, it is too slow, so speed it up by subtracting $40 from OSCTRIM. Repeat this process seven more times, each time dividing the number to be added to/subtracted from OSCTRIM by two. After eight iterations, the trim value should be set.

The drawback to this method is related to the saw-toothed characteristic discussed in **Frequency/Period versus Trim Value**. Since the saw-toothed relationship between the trim value and period results in non-monotonic trim values, a possibility exists that the binary search could hit a peak or valley that is large enough to cause the search to branch in the wrong direction. This could

cause the final trimmed period to be off by a few percent. This can be overcome by only using the binary search for four iterations instead of eight. Then switch to a sequential search for the final four bits of the trim value.

Iterative Linear Search

The iterative linear search uses the linear search as the basis. However, instead of stopping after one measurement of the reference clock, this method iteratively measures the reference after each update of OSCTRIM and continues until the expected and actual values match.

The drawback to this method is that due to the saw-toothed irregularities in the trim curve, the search could bounce back and forth between values and never settle into the correct value. To avoid this, the iterations can be counted. After a defined number of loops, the routine switches to a sequential search to select the correct value.

Assembly Code Examples

The iterative linear search method is used for the code examples, although any of the above methods are acceptable as long as they meet the user's need. Two versions of the code are provided, the difference being the reference clock source. In the first one, an RS232 terminal is used to provide the reference clock by sending a specific character, in this case the '@' character (ASCII value $40), to either of the timer channels at 9600 baud. In the second example, a 60 Hz square wave is created from an AC power line to provide the reference clock into one of the timer channels.

The Program

This auto-trim program uses the iterative linear search to find the best trim value to set the frequency to 3.2 MHz. The timer channel to be used is selected in the assembly code by setting two EQUATE variables:

```
TSCHNL    equ   TSC0 or TSC1
TCHNLH    equ   TCH0H or TCH1H
```

TSCHNL selects which channel's status and control register to use and TCHNLH selects which timer channel register set to use.

The program starts by initializing the configuration registers, the timer, and a few random-access memory (RAM) variables. It checks the current value in TRIMLOC. If TRIMLOC is not equal to $FF (blank) then the routine assumes a trim value had previously been found and skips the search, jumping to the end of the routine where a port is toggled at 20 times the period of the bus clock. This allows the user to verify the trimmed frequency.

If TRIMLOC is blank ($FF), then the program starts the timer with interrupts enabled and goes into a loop. The input capture is setup to capture rising edges. The first timer input capture interrupt is received at the first rising edge of the reference clock. The timer channel registers are read and stored in RAM as an offset value. The interrupt is cleared and the program returns to the loop. The second rising edge of the reference triggers another input capture

interrupt. The offset that was saved after the first input capture is subtracted from timer value from the second input capture. The difference is the measured value of the reference clock's period. This value is saved in RAM as the actual value. The program then returns to the main routine.

NOTE: *The program is not concerned with a timer counter overflow when calculating the actual value. In other words, suppose the offset = $FFF0 and the second input capture = $013D due to the timer overflowing from $FFFF to $0000. Subtracting $013D – $FFF0 will still result in the correct actual value of $014D.*

Once an actual value has been captured, flags are set allowing the program to branch out of the loop and decide if the oscillator is running too fast, too slow or just right. This judgment is made by comparing the actual value to the expected value based on the reference clock.

If the oscillator is running too fast, the delta is calculated by subtracting the expected value from the actual value. Otherwise, the delta is calculated by subtracting the actual from the expected. By setting the timer prescalar value appropriately in relation to the period of the reference clock, we can use this delta value as the adjustment to the OSCTRIM value. **Trim with RS232 Terminal as Reference** and **Trim with 50/60 Hz Clock as Reference** will explain this in detail.

If the oscillator is too fast, the delta is added to the OSCTRIM value. A check is made to verify that the sum is not greater than $FF. If it is, OSCTRIM is set to $FF, otherwise it is set to the sum of OSCTRIM and delta. The program now returns to the loop to wait for the next input capture.

If the oscillator is too slow, the delta is subtracted from the OSCTRIM value. A check is made to verify the difference was not less than $00. If it is, OSCTRIM is set to $00, otherwise, it is set to the difference of OSCTRIM and delta. The program now returns to the loop to wait for the next input capture.

If the oscillator matches the desired frequency (3.2 MHz), then the program jumps out of the loop and programs the OSCTRIM value into the TRIMLOC register in flash memory.

After programming, the TRIMLOC value is moved back into OSCTRIM and PTA3 is toggled at 20 times the bus period. This allows the user to verify that the correct value was programmed into TRIMLOC.

To prevent the case where the OSCTRIM value toggles between a higher and lower trim value without ever settling into the desired trim value, the number of iterations is tracked in a RAM variable. If, after 16 iterations, the trim value has not been found, the program will change the delta to '$01' regardless of the actual delta. At this point, it should be within a few steps of the desired trim value. The program will allow another 16 iterations before quitting, using the last OSCTRIM value as the desired value.

Trim with RS232 Terminal as Reference

When using an RS232 terminal, the reference clock is provided by sending the '@' character at 9600 baud to either one of the timer channels through an RS232 terminal or PC terminal emulator. The reason for using the "@" character is explained later in this section.

Figure 10 shows a circuit for connecting an RS232 port to one of the timer channel inputs. In this example, the user will have to send "@" characters until the trim value has been found. When PTA3 starts outputting a square wave, the trim value has been found and programmed into FLASH. The amount of time between sending the characters is irrelevant, as the program is only concerned with the "@" character itself.

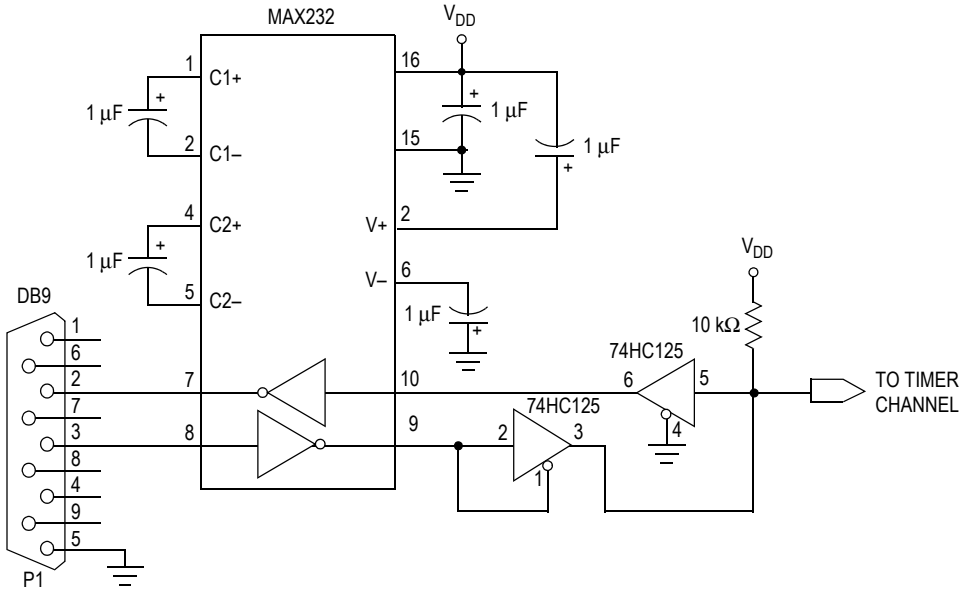


Figure 10. RS232 Communication Circuit

For the program to work best, we want to setup the timer and reference clock such that one increment in the timer count is between 0.4% and 0.2% of the expected value. Going below 0.2% is beyond the resolution of the trim capabilities of the oscillator. Going above 0.4% doesn't take full advantage of the trim capabilities of the oscillator.

To calculate the expected value, use the equation:

$$\text{Expected value} = (f_{BUS} \times t_{REF}) / PS$$

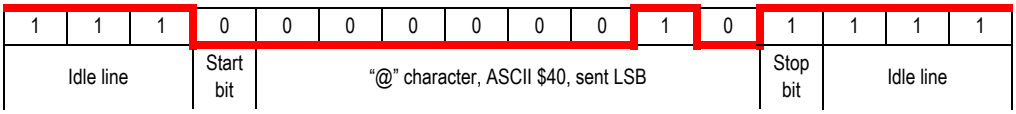
Where:

f_{BUS} = The trimmed bus frequency (3.2 MHz)

t_{REF} = The period of the reference clock

PS = The prescaler of the timer module or 1, 2, 4, 8, 16, 32, 64

At 9600 baud, each bit is 104 μs long. The '@' character is used because its hex value is $40 or %01000000. Since RS232 holds the transmit (Tx) line high between characters, the '@' character provides exactly two rising edges, exactly two bit times apart. With the start (0) and stop bits (1) added and keeping in mind data is sent LSB, the exact transmission looks like this:



The result is two rising edges at 208 μs apart. Therefore, t_{ref} = 208 μs. So, starting with a timer prescaler, PS = 1, our expected value is:

Expected value = (3.2 MHz × 208 μs)/1 = 665.6 ~ 666

However, we want the timer resolution to be between 0.2% and 0.4% of the expected value. In this example, the resolution is 1/666 = 0.15%. If we set the timer prescaler to two, then the expected value becomes 333 and the resolution is 1/333 = 0.3%. These are good values for the program.

One more trick is used to simplify the program. Since the untrimmed frequency can vary by ±25% from 3.2 MHz, calculate the extremes of the expected values:

Expected value = 333 = $14D

+25% = 1.25 × expected value = 1.25 × 333 = 416 = $1A0

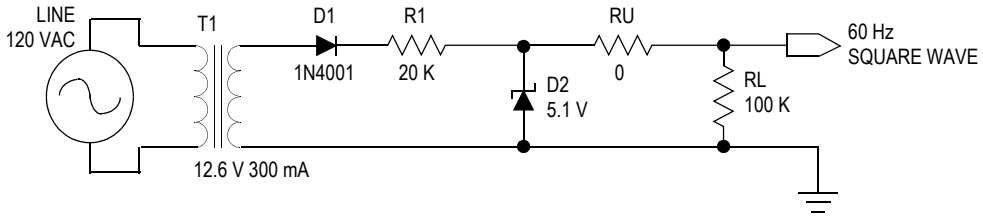
–25% = 0.75 × expected value = 0.75 × 333 = 250 = $FA

If we leave the expected value at $14D, then we could end up with an actual value of less than $100 if the oscillator is very slow. This will require us to use both the high and low bytes of the actual value to determine the delta. However, if we add an adjustment to both the expected and actual values to get the possible range of values between $100 and $1FF, then we can ignore the high byte and perform all calculations on the low byte only. In this example, an adjustment of $06 is added to both the expected and actual values, resulting in a range of possible actual values between $100 to $1A6 and an expected value of $153.

Refer to **Appendix A: HC08 Assembly Code Example for RS232**.

Trim with 50/60 Hz Clock as Reference

An AC power line can be used as a reference clock by utilizing the circuit in **Figure 11** and routing the output to the timer channel of choice. In this example, the user will not have to manually provide reference clock pulses since the square wave generator in this circuit will provide a constant reference clock as long as the AC power is turned on.



	RU	RL
5.0 V	0 K	100 K
3.3 V	33 K	68 K
3.0 V	39 K	56 K

Figure 11. 60 Hz Square Wave

We can calculate the expected value and timer prescaler setting using the same methods as in the RS232 example. In this example:

f_{BUS} = 3.2 MHz

t_{REF} = 1/60 Hz = 16.7 ms

Trying a timer prescaler PS = 64, the expected value is:

Expected value = ($f_{BUS} \times t_{REF}$) / PS

Expected value = (3.2 MHz $\times$ 16.7 ms)/64 = 833

In this case, the resolution is 1/833 = 0.12%, which is less than 0.2%. Since 64 is the maximum prescaler value, the values will have to be divided by 2 using a ROR command to get the resolution to 1/417 = 0.24%. This results in expected values of:

Expected value = 417 = $1A1

+25% = 1.25 $\times$ expected value = 1.25 $\times$ 417 = 521 = $209

–25% = 0.75 $\times$ expected value = 0.75 $\times$ 333 = 313 = $139

In order to get the range of values between $100 and $1FF, an adjustment of $0A is subtracted from all values. This puts the range of potential actual measurements between $1FF and $12F, and the expected value at $197.

Refer to **Appendix B: HC08 Assembly Code Example for 60 Hz**.

Low Power Modes

The HC08 Family of microcontrollers makes use of two special low power modes, wait and stop. In wait mode, the bus clock source is kept running, but the central processor unit (CPU) is powered off to conserve power. In stop mode, the bus clock source is powered down along with the CPU.

Wait Mode

In wait mode, the clock source remains active. Therefore, if the internal oscillator is the selected clock source, it will continue to operate normally in wait mode. Exiting wait mode with an interrupt has no effect on the clock settings. However, exiting wait mode with a reset causes the OSCTRIM value to be reset back to $80.

Stop Mode

In stop mode, the clock source is disabled. If the internal oscillator is the selected clock source, then it is powered down in stop mode. Exiting stop mode with an interrupt will activate the internal oscillator with the same trim value as before entering stop. Exiting stop mode with a reset will cause the internal oscillator to be selected as the clock source with the trim value reset to $80.

When using the internal oscillator, the short stop recovery option can be selected to provide a quicker startup time. This option is selected in the chip configuration register 1 (CONFIG1), with bit 2. With this option, instructions can be executed after 32 oscillator cycles, instead of the default 4096 oscillator cycles.

Conclusion

The internal oscillator is a new module for the HC08 Family of microcontrollers. It has the advantage of a smaller die size over the ICG module found on several other HC08 microcontrollers while requiring no external components or using valuable pins on a low pin count device, making it very cost effective. The characteristics of this oscillator make it accurate over temperature and voltage variations once it has been trimmed. This module is an ideal clock source for low pin count / low cost microcontrollers.

Appendix A: HC08 Assembly Code Example for RS232

```
Metrowerks HC08-Assembler
(c) COPYRIGHT METROWERKS 1987-2002

Loc  Obj. code   Source line
---- ---------   -----------
                 ;.base 10t                          ;change default to decimal
                 ;.pagewidth 98t
                 ;*********************************************************************
                 ;* Filename: HC908QY4_OSCTRIM_9600.asm     Copyright (c) 2002
                 ;*********************************************************************
                 ;*********************************************************************
                 ;* Oscillator Trim Routine for MC68HC908Qx Family   Scott Pape
                 ;*
                 ;* Rev: 1.0      Date: 18July2002
                 ;*
                 ;* This code, will trim the internal oscillator to 3.17 MHz. Timer
                 ;* channel 0 or 1 is used to measure the period of the ref clock. The
                 ;* reference clock is provided by a terminal or terminal emulator set
                 ;* for 9600 baud communication. The '@' (at) character is used to
                 ;* provide the proper waveform to the timer channel. The OSCTRIM value
                 ;* is modified if the measured value is higher or lower than the
                 ;* expected value. Once the propoer trim value is found, it is
                 ;* programmed into the OSCTRIM storage location in Flash, at $FFC0.
                 ;*
                 ;* Usage Notes:
                 ;* 1. This code assumes that the Tx pin of an RS-232 terminal or PC
                 ;*    terminal emulator is connected to either the TCH0 or TCH1 pin.
                 ;*    (See Monitor Mode section of the MC68HC908QY4 data book for an
                 ;*    example of this connection.)
                 ;* 2. The OSC2EN bit in the PTAPUE register is set so that the
                 ;*    internal oscillator is output on PTA4 for observation during
                 ;*    the trimming process.
                 ;* 3. Once the trim value has been found and programmed into flash,
                 ;*    a square wave is output on PTA3 at 20X the period of the busclk.
                 ;*    When this signal is output, it signals that the trimming
                 ;*    process is complete.
                 ;*********************************************************************
                 ;*********************************************************************
                 ;*
                 ;* Freescale reserves the right to make changes without further notice
                 ;* to any product herein to improve reliability, function, or design.
                 ;* Freescale does not assume any liability arising out of the
                 ;* application or use of any product, circuit, or software described
                 ;* herein; neither does it convey any license under its patent rights
                 ;* nor the rights of others.  Freescale products are not designed,
                 ;* intended, or authorized for use as components in systems intended
                 ;* for surgical implant into the body, or other applications intended
                 ;* to support life, or for any other application in which the failure
                 ;* of the Freescale product could create a situation where personal
                 ;* injury or death may occur.  Should Buyer purchase or use Freescale
```

```
                ;* products for any such intended or unauthorized application, Buyer
                ;* shall indemnify and hold Freescale and its officers, employees,
                ;* subsidiaries, affiliates, and distributors harmless against all
                ;* claims, costs, damages, and expenses, and reasonable attorney fees
                ;* arising out of, directly or indirectly, any claim of personal
                ;* injury or death associated with such unintended or unauthorized
                ;* use, even if such claim alleges that Freescale was negligent
                ;* regarding the design or manufacture of the part.
                ;*
                ;* Freescale and Freescale logo are registered trademarks of Freescale, Inc.
                ;*******************************************************************************
                ;.nolist
                             include "MC68HC908QT4.equu"
                ;.list
                ;
                ;*** Equates for registers and register values in MC68HC908Qx ********

0000 0000   initCONFIG2: equ    %00000000     ;initialization
                ;                 || ||  |       CONFIG2 is a write-once register
                ;                 || ||  +-RSTEN   - 0 disable RST pin fcn
                ;                 || ++----OSCOPT[1:0]- 0:0 use internal osc
                ;                 |+-------IRQEN   - 0 disable ext IRQ pin fcn
                ;                 +--------IRQPUE  - 0 enable pull-up on IRQ

0000 0011   initCONFIG1: equ    %00010001     ;initialization
                ;                 ||||||||       CONFIG1 is a write-once register
                ;                 |||||||+-COPD    - 1 disable COP watchdog
                ;                 ||||||+--STOP    - 0 disable STOP instruction
                ;                 |||||+---SSREC   - 0 4096 cycle STOP recovery
                ;                 ||||+----LVI5OR3 - 0 set LVI for 3v system
                ;                 |||+-----LVIPWRD - 1 disable power to LVI system
                ;                 ||+------LVIRSTD - 0 enable reset on LVI trip
                ;                 |+-------LVISTOP - 0 disable LVI in STOP mode
                ;                 +--------COPRS   - 0 long COP timeout

0000 0080   initPTAPUE:  equ    mOSC2EN       ;bit 7 enables OSC2 output

0000 0025   TSCHNL       equ    TSC0          ;Select TSC0 or TSC1, for input
0000 0026   TCHNLH       equ    TCH0H         ;Select TCH0H or TCH1H, for input
            ;resetTSC:    equ    {mTSTOP|mTRST} ;Stop and reset timer counter
0000 0030   resetTSC:    equ    (mTSTOP|mTRST) ;<CodeWarrior format>
0000 0001   startTSC:    equ    mPS0          ;Start timer with prescalar=2
            ;initTSCHNL:  equ    {mCH0IE|mELS0A} ;Setup Timer ch for input capture
0000 0044   initTSCHNL:  equ    (mCH0IE|mELS0A) ;<CodeWarrior format>

0000 0001   initFLCR     equ    mPGM          ;Set the PROG bit
            ;hvenFLCR     equ    {mPGM|mHVEN}  ;Set the HVEN and PROG bits
0000 0009   hvenFLCR     equ    (mPGM|mHVEN)  ;<CodeWarrior format>
0000 0008   clrprgFLCR   equ    mHVEN         ;Clear the PROG bit only

0000 0001   tnvs         equ    01            ;12 usec: Flash HVEN setup time
0000 0001   tpgs         equ    01            ;12 usec: Flash PROG hold time
0000 0001   tnvh         equ    01            ;12 usec: Flash HVEN hold time
0000 0003   tprog        equ    03            ;3*12 = 36 usec: Flash Prog time
0000 000D   speed        equ    13            ;cpuspd = 3.18mhz * 4 = 12.7 (~13)
```

```
      0000 280C   Delnus       equ    $280C          ;Addr of delay subr in ROM

                  ;*** equates for RAM variables and Constants ************************
      0000 0080   Sync:         equ    RamStart       ;bit0=offset flag, bit1=got value
      0000 0081   SyncOffsetH:  equ    RamStart+1     ;Timer offset hi byte
      0000 0082   SyncOffsetL:  equ    RamStart+2     ;Timer offset lo byte
      0000 0083   ActualH:      equ    RamStart+3     ;Timer count hi byte
      0000 0084   ActualL:      equ    RamStart+4     ;Timer count lo byte
      0000 0085   Delta:        equ    RamStart+5     ;Actual delta from expected
      0000 0086   LoopCnt:      equ    RamStart+6     ;Count how many measurements taken

      0000 0008   Adjust:       equ    $08            ;Adjustment to get val $100-$1FF
      0000 0001   ExpectedH:    equ    $01
      0000 0053   ExpectedL:    equ    $4B+Adjust     ;Expected Timer value if trimmed

                  ;*** Begining of Flash Memory ***************************************
                              org    FlashStart

                  ;*** Main ***********************************************************
EE00 6E 11 1F     Start:        mov    #initCONFIG1,CONFIG1
EE03 6E 00 1E                   mov    #initCONFIG2,CONFIG2
EE06 6E 80 0B                   mov    #initPTAPUE,PTAPUE ;Output OSC2 for observation
EE09 C6 FFC0                    lda    TRIMLOC        ;Get current Trim value
EE0C 43                         coma                  ;com $FF = $00
EE0D 26 6F                      bne    V_TRIMLOC      ;If Trim <> $FF, skip search
EE0F 3F 86                      clr    LoopCnt        ;Init iteration counter
EE11 6E 30 20                   mov    #resetTSC,TSC  ;Stop & reset timer
EE14 6E 44 25                   mov    #initTSCHNL,TSCHNL
EE17 6E 01 20     TryAgain:     mov    #startTSC,TSC
EE1A 3F 80                      clr    Sync
EE1C 3F 81                      clr    SyncOffsetH
EE1E 3F 82                      clr    SyncOffsetL
EE20 3F 83                      clr    ActualH
EE22 3F 84                      clr    ActualL
EE24 3F 85                      clr    Delta
EE26 3C 86                      inc    LoopCnt        ;Increment iteration count
EE28 9A                         cli
                  loop:
EE29 03 80 FD                   brclr  1,Sync,loop    ;Stay in loop until Actual flag set
EE2C B6 84                      lda    ActualL        ;Adjust the Actual value so that...
EE2E AB 08                      add    #Adjust        ;...the range fits into 1 byte
EE30 B7 84                      sta    ActualL
EE32 A1 53                      cmp    #ExpectedL     ;Compare Actual with Expected value
EE34 27 3C                      beq    GotTrim        ;If equal, osc is trimmed!
EE36 25 1D                      blo    TooSlow        ;If lower, osc is too slow
EE38 A0 53                      sub    #ExpectedL     ;Otherwise, osc is too fast
EE3A B7 85                      sta    Delta          ;Calculate Delta from Act-Exp
EE3C B6 38                      lda    OSCTRIM
EE3E A1 FF                      cmp    #$FF           ;Check if trim=$FF
EE40 27 30                      beq    GotTrim        ;if true, can't slow more
EE42 0A 86 2D                   brset  5,LoopCnt,GotTrim ;bit5=32 loops=limit
EE45 09 86 03                   brclr  4,LoopCnt,AddDelta ;bit4 = 16 loops
EE48 6E 01 85                   mov    #$01,Delta     ;after 8 loops, just add 1
EE4B BB 85        AddDelta:     add    Delta          ;acca still has OSCTRIM
```

```
EE4D 24 02                  bcc    AddTrim       ;did carry bit get set?
EE4F A6 FF                  lda    #$FF          ;if carry set, max osctrim
EE51 B7 38       AddTrim:   sta    OSCTRIM       ;Trim increases to slow freq
EE53 20 C2                  bra    TryAgain      ;Try new value
EE55 A6 53       TooSlow:   lda    #ExpectedL
EE57 B0 84                  sub    ActualL
EE59 B7 85                  sta    Delta         ;Calculate Delta from Exp-Act
EE5B B6 38                  lda    OSCTRIM
EE5D 27 13                  beq    GotTrim       ;if true, can't speed more
EE5F 0A 86 10               brset  5,LoopCnt,GotTrim ;bit5=32 loops=limit
EE62 09 86 03               brclr  4,LoopCnt,SubDelta ;bit4 = 16 loops
EE65 6E 01 85               mov    #$01,Delta    ;after 8 loops, just add 1
EE68 B0 85       SubDelta:  sub    Delta
EE6A 24 02                  bcc    SubTrim       ;did carry bit get set?
EE6C A6 FF                  lda    #$FF          ;if carry set, min osctrim
EE6E B7 38       SubTrim:   sta    OSCTRIM
EE70 20 A5                  bra    TryAgain

                 ;*** Program Trim Value to Flash ************************************
                 GotTrim:
                 ;             ldhx   #{EndPrgTrim-PrgTrim} ;Init to size of PrgTrim
EE72 45 0045                 ldhx   #(EndPrgTrim-PrgTrim) ;<CodeWarrior format>
EE75 D6 EE91     Mov2RAM:    lda    PrgTrim,x     ;Get byte from flash and...
EE78 E7 7F                   sta    RamStart-1,x  ;...put into RAM
EE7A 5B F9                   dbnzx  Mov2RAM
EE7C BD 80                   jsr    RamStart      ;Program the Trim value

                 ;*** Toggle PTA4 to check frequency **********************************
EE7E C6 FFC0     V_TRIMLOC:  lda    TRIMLOC
EE81 B7 38                   sta    OSCTRIM       ;Verify trim value in flash
EE83 16 04                   bset   3,DDRA
EE85 17 00       togPTA3:    bclr   3,PTA         ;[4]
EE87 21 FE                   brn    *             ;[3]keep 50% duty cycle on toggle
EE89 21 FE                   brn    *             ;[3]keep 50% duty cycle on toggle
EE8B 16 00                   bset   3,PTA         ;[4]
EE8D 21 FE                   brn    *             ;[3]keep 50% duty cycle on toggle
EE8F 20 F4                   bra    togPTA3       ;[3]PTA3 will toggle @ 20x bus

                 ;*** Subroutines ****************************************************
                 ; PrgTrim:
                 ; Programs the value in the OSCTRIM register into the reserved Flash
                 ; location for the trim value, TRIMLOC = $FFC0.
                 ; Entry Conditions:
                 ;  1. This routine has been copied into RAM.
                 ;  2. The TRIMLOC ($FFC0) location in flash is blank ($FF).
                 ;  3. The OSCTRIM register contains the desired trim value.
                 ;
                 ; Exit conditions:
                 ;  1. TRIMLOC is programmed with the value in OSCTRIM.
                 ;  2. OSCTRIM is not modified.
                 ;  3. A,H,X registers are saved and restored.

                 PrgTrim:
EE91 9D                      nop                  ;Sacrificial byte, not moved to RAM
EE92 87                      psha
```

```
EE93 8B                     pshh
EE94 89                     pshx                ;Save A,H,X registers

EE95 A6 01                  lda    #initFLCR    ;Programming step 1
EE97 C7 FE08                sta    FLCR

EE9A C6 FFBE                lda    FLBPR        ;Programming step 2

EE9D C7 FFC0                sta    TRIMLOC      ;Programming step 3

EEA0 AE 01                  ldx    #tnvs        ;DELAY FOR tnvs
EEA2 A6 0D                  lda    #speed
EEA4 CD 280C                jsr    Delnus       ;Programming step 4

EEA7 A6 09                  lda    #hvenFLCR    ;Programming step 5
EEA9 C7 FE08                sta    FLCR

EEAC AE 01                  ldx    #tpgs        ;DELAY FOR tpgs
EEAE A6 0D                  lda    #speed
EEB0 CD 280C                jsr    Delnus       ;Programming step 6

EEB3 B6 38                  lda    OSCTRIM      ;Programming step 7
EEB5 C7 FFC0                sta    TRIMLOC      ;Write OSCTRIM to TRIMLOC

EEB8 AE 03                  ldx    #tprog       ;DELAY FOR tprog
EEBA A6 0D                  lda    #speed
EEBC CD 280C                jsr    Delnus       ;Programming step 8

EEBF A6 08                  lda    #clrprgFLCR  ;Programming step 10
EEC1 C7 FE08                sta    FLCR

EEC4 AE 01                  ldx    #tnvh        ;DELAY FOR tnvh
EEC6 A6 0D                  lda    #speed
EEC8 CD 280C                jsr    Delnus       ;Programming step 11

EECB 4F                     clra
EECC C7 FE08                sta    FLCR         ;Programming step 12

EECF 21 FE                  brn    *
EED1 21 FE                  brn    *            ;Programming step 13

EED3 88                     pulx
EED4 8A                     pulh
EED5 86                     pula                ;Restore A,H,X registers

EED6 81          EndPrgTrim:  rts

                 ;*** Interrupt Service Routines ************************************
                 ; TIM_isr:
                 ; This interrupt service routine uses the 1st input capture as a sync
                 ; edge for the reference clock on the selected Timer channel.
                 ; Then it reads the selected timer channel's counter register. Based
                 ; on the status of the Sync flag variable in RAM, either the channel
                 ; counter data is saved as an offset or the actual count is calculated
                 ; by subtracting the offset from the current channel value.
```

```
                ;
                ; This routine MUST be shorter in duration than 1 period of the
                ; reference clock. Otherwise, the input capture interrupts will keep
                ; stacking until RAM overflows and an illegal address reset occurs.

EED7 8B         TIM_isr:    pshh                ;Save H, since it is not stacked
EED8 B6 25                  lda    TSCHNL       ;Clear the int flag...
EEDA A4 7F                  and    #$7F         ;...without modifying the...
EEDC B7 25                  sta    TSCHNL       ;...settings in Channel Register
EEDE 55 26                  ldhx   TCHNLH       ;Get the captured channel value
EEE0 00 80 06               brset  0,Sync,GetActual ;Check if Offset or Actual
EEE3 35 81                  sthx   SyncOffsetH  ;Save as Offset
EEE5 10 80                  bset   0,Sync       ;Set Offset captured flag
EEE7 20 0D                  bra    TIM_rti      ;Return from int
EEE9 12 80      GetActual:  bset   1,Sync       ;Set actual measure taken flag
EEEB 9F                     txa                 ;Move the low byte of value to A
EEEC B0 82                  sub    SyncOffsetL  ;Sub low byte of Offset
EEEE B7 84                  sta    ActualL      ;Save low byte of Actual
EEF0 8B                     pshh                ;Transfer high byte of Offset...
EEF1 86                     pula                ;...to A
EEF2 B2 81                  sbc    SyncOffsetH  ;Sub high byte of Offset & carry
EEF4 B7 83                  sta    ActualH      ;Save high byte of Actual
EEF6 8A         TIM_rti:    pulh                ;Restore H before unstacking
EEF7 80                     rti

                ;*** Vectors ***********************************************************
                            org    Vtimch1
FFF4 EED7                   fdb    TIM_isr      ;Timer ch1 int vector
FFF6 EED7                   fdb    TIM_isr      ;Timer ch0 int vector

                            org    Vreset
FFFE EE00                   fdb    Start        ;Reset vector
                END
```

Appendix B: HC08 Assembly Code Example for 60 Hz

```
Metrowerks HC08-Assembler
(c) COPYRIGHT METROWERKS 1987-2002

Loc  Obj. code   Source line
---- ---------   -----------
                 ;.base 10t                          ;change default to decimal
                 ;.pagewidth 98t
                 ;*********************************************************************
                 ;* Filename: HC908QY4_OSCTRIM_60HZ.asm     Copyright (c)  2002
                 ;*********************************************************************
                 ;*********************************************************************
                 ;* Oscillator Trim Routine for MC68HC908Qx Family   Scott Pape
                 ;*
                 ;* Rev: 1.0      Date: 18July2002
                 ;*
                 ;* This code, will trim the internal oscillator to 3.2 MHz. Timer
                 ;* channel 0 or 1 is used to measure the period of the ref clock. The
                 ;* reference clock is provided by a 60Hz signal taken from an AC
                 ;* powerline. The OSCTRIM value is modified if the measured value is
                 ;* higher or lower than the expected value. Once the propoer trim
                 ;* value is found, it is programmed into the OSCTRIM storage location
                 ;* in Flash, at $FFC0.
                 ;*
                 ;* Usage Notes:
                 ;* 1. This code assumes that the output of a 60HZ reference signal is
                 ;*    connected to either the TCH0 or TCH1 pin.
                 ;* 2. The OSC2EN bit in the PTAPUE register is set so that the
                 ;*    internal oscillator is output on PTA4 for observation during
                 ;*    the trimming process.
                 ;* 3. Once the trim value has been found and programmed into flash,
                 ;*    a square wave is output on PTA3 at 20X the period of the busclk.
                 ;*    When this signal is output, it signals that the trimming
                 ;*    process is complete.
                 ;*********************************************************************
                 ;*********************************************************************
                 ;*
                 ;* Freescale reserves the right to make changes without further notice
                 ;* to any product herein to improve reliability, function, or design.
                 ;* Freescale does not assume any liability arising out of the
                 ;* application or use of any product, circuit, or software described
                 ;* herein; neither does it convey any license under its patent rights
                 ;* nor the rights of others.  Freescale products are not designed,
                 ;* intended, or authorized for use as components in systems intended
                 ;* for surgical implant into the body, or other applications intended
                 ;* to support life, or for any other application in which the failure
                 ;* of the Freescale product could create a situation where personal
                 ;* injury or death may occur.  Should Buyer purchase or use Freescale
                 ;* products for any such intended or unauthorized application, Buyer
                 ;* shall indemnify and hold Freescale and its officers, employees,
                 ;* subsidiaries, affiliates, and distributors harmless against all
```

```
                ;* claims, costs, damages, and expenses, and reasonable attorney fees
                ;* arising out of, directly or indirectly, any claim of personal
                ;* injury or death associated with such unintended or unauthorized
                ;* use, even if such claim alleges that Freescale was negligent
                ;* regarding the design or manufacture of the part.
                ;*
                ;* Freescale and the Freescale logo are registered trademarks of Freescale, Inc.
                ;*****************************************************************************
                ;.nolist
                             include "MC68HC908QT4.equ"
                ;.list
                ;
                ;*** Equates for registers and register values in MC68HC908Qx ********

0000 0000   initCONFIG2: equ    %00000000     ;initialization
                ;                 || ||  |       CONFIG2 is a write-once register
                ;                 || ||  +-RSTEN   - 0 disable RST pin fcn
                ;                 || ++----OSCOPT[1:0]- 0:0 use internal osc
                ;                 |+-------IRQEN   - 0 disable ext IRQ pin fcn
                ;                 +--------IRQPUE  - 0 enable pull-up on IRQ

0000 0011   initCONFIG1: equ    %00010001     ;initialization
                ;                 ||||||||       CONFIG1 is a write-once register
                ;                 |||||||+-COPD    - 1 disable COP watchdog
                ;                 ||||||+--STOP    - 0 disable STOP instruction
                ;                 |||||+---SSREC   - 0 4096 cycle STOP recovery
                ;                 ||||+----LVI5OR3 - 0 set LVI for 3v system
                ;                 |||+-----LVIPWRD - 1 disable power to LVI system
                ;                 ||+------LVIRSTD - 0 enable reset on LVI trip
                ;                 |+-------LVISTOP - 0 disable LVI in STOP mode
                ;                 +--------COPRS   - 0 long COP timeout

0000 0080   initPTAPUE:  equ    mOSC2EN       ;bit 7 enables OSC2 output

0000 0028   TSCHNL       equ    TSC1          ;Select TSC0 or TSC1, for input
0000 0029   TCHNLH       equ    TCH1H         ;Select TCH0H or TCH1H, for input
            ;resetTSC:    equ    {mTSTOP|mTRST} ;Stop and reset timer counter
0000 0030   resetTSC:    equ    (mTSTOP|mTRST) ;<CodeWarrior format>
            ;startTSC:    equ    {mPS2|mPS1}   ;Start timer with prescalar=64
0000 0006   startTSC:    equ    (mPS2|mPS1)   ;<CodeWarrior format>
            ;initTSCHNL:  equ    {mCH0IE|mELS0A} ;Setup Timer ch for input capture
0000 0044   initTSCHNL:  equ    (mCH0IE|mELS0A) ;<CodeWarrior format>

0000 0001   initFLCR     equ    mPGM          ;Set the PROG bit
            ;hvenFLCR     equ    {mPGM|mHVEN}  ;Set the HVEN and PROG bits
0000 0009   hvenFLCR     equ    (mPGM|mHVEN)  ;<CodeWarrior format>
0000 0008   clrprgFLCR   equ    mHVEN         ;Clear the PROG bit only

0000 0001   tnvs         equ    01            ;12 usec: Flash HVEN setup time
0000 0001   tpgs         equ    01            ;12 usec: Flash PROG hold time
0000 0001   tnvh         equ    01            ;12 usec: Flash HVEN hold time
0000 0003   tprog        equ    03            ;3*12 = 36 usec: Flash Prog time
0000 000D   speed        equ    13            ;cpuspd = 3.18mhz * 4 = 12.7 (~13)

0000 280C   Delnus       equ    $280C         ;Addr of delay subr in ROM
```

```
                ;*** equates for RAM variables and Constants ************************
     0000 0080  Sync:        equ    RamStart      ;bit0=offset flag, bit1=got value
     0000 0081  SyncOffsetH: equ    RamStart+1    ;Timer offset hi byte
     0000 0082  SyncOffsetL: equ    RamStart+2    ;Timer offset lo byte
     0000 0083  ActualH:     equ    RamStart+3    ;Timer count hi byte
     0000 0084  ActualL:     equ    RamStart+4    ;Timer count lo byte
     0000 0085  Delta:       equ    RamStart+5    ;Actual delta from expected
     0000 0086  LoopCnt:     equ    RamStart+6    ;Count how many measurements taken

     0000 000A  Adjust:      equ    $0A           ;Adjustment to get val $100-$1FF
     0000 0001  ExpectedH:   equ    $01
     0000 0097  ExpectedL:   equ    $A1-Adjust    ;Expected Timer value if trimmed

                ;*** Begining of Flash Memory ***************************************
                        org    FlashStart

                ;*** Main ***********************************************************
EE00 6E 11 1F   Start:      mov    #initCONFIG1,CONFIG1
EE03 6E 00 1E               mov    #initCONFIG2,CONFIG2
EE06 6E 80 0B               mov    #initPTAPUE,PTAPUE ;Output OSC2 for observation
EE09 C6 FFC0                lda    TRIMLOC       ;Get current Trim value
EE0C 43                     coma                 ;com $FF = $00
EE0D 26 6F                  bne    V_TRIMLOC     ;If Trim <> $FF, skip search
EE0F 3F 86                  clr    LoopCnt       ;Init iteration counter
EE11 6E 30 20               mov    #resetTSC,TSC ;Stop & reset timer
EE14 6E 44 28               mov    #initTSCHNL,TSCHNL
EE17 6E 06 20   TryAgain:   mov    #startTSC,TSC
EE1A 3F 80                  clr    Sync
EE1C 3F 81                  clr    SyncOffsetH
EE1E 3F 82                  clr    SyncOffsetL
EE20 3F 83                  clr    ActualH
EE22 3F 84                  clr    ActualL
EE24 3F 85                  clr    Delta
EE26 3C 86                  inc    LoopCnt       ;Increment iteration count
EE28 9A                     cli
                loop:
EE29 03 80 FD               brclr  1,Sync,loop   ;Stay in loop until Actual flag set
EE2C B6 84                  lda    ActualL       ;Adjust the Actual value so that...
EE2E A0 0A                  sub    #Adjust       ;...the range fits into 1 byte
EE30 B7 84                  sta    ActualL
EE32 A1 97                  cmp    #ExpectedL    ;Compare Actual with Expected value
EE34 27 3C                  beq    GotTrim       ;If equal, osc is trimmed!
EE36 25 1D                  blo    TooSlow       ;If lower, osc is too slow
EE38 A0 97                  sub    #ExpectedL    ;Otherwise, osc is too fast
EE3A B7 85                  sta    Delta         ;Calculate Delta from Act-Exp
EE3C B6 38                  lda    OSCTRIM
EE3E A1 FF                  cmp    #$FF          ;Check if trim=$FF
EE40 27 30                  beq    GotTrim       ;if true, can't slow more
EE42 0A 86 2D               brset  5,LoopCnt,GotTrim ;bit5=32 loops=limit
EE45 09 86 03               brclr  4,LoopCnt,AddDelta ;bit4 = 16 loops
EE48 6E 01 85               mov    #$01,Delta    ;after 8 loops, just add 1
EE4B BB 85      AddDelta:   add    Delta         ;acca still has OSCTRIM
EE4D 24 02                  bcc    AddTrim       ;did carry bit get set?
EE4F A6 FF                  lda    #$FF          ;if carry set, max osctrim
```

```
EE51 B7 38       AddTrim:     sta    OSCTRIM       ;Trim increases to slow freq
EE53 20 C2                    bra    TryAgain      ;Try new value
EE55 A6 97       TooSlow:     lda    #ExpectedL
EE57 B0 84                    sub    ActualL
EE59 B7 85                    sta    Delta         ;Calculate Delta from Exp-Act
EE5B B6 38                    lda    OSCTRIM
EE5D 27 13                    beq    GotTrim       ;if true, can't speed more
EE5F 0A 86 10                 brset  5,LoopCnt,GotTrim ;bit5=32 loops=limit
EE62 09 86 03                 brclr  4,LoopCnt,SubDelta ;bit4 = 16 loops
EE65 6E 01 85                 mov    #$01,Delta    ;after 8 loops, just add 1
EE68 B0 85       SubDelta:    sub    Delta
EE6A 24 02                    bcc    SubTrim       ;did carry bit get set?
EE6C A6 FF                    lda    #$FF          ;if carry set, min osctrim
EE6E B7 38       SubTrim:     sta    OSCTRIM
EE70 20 A5                    bra    TryAgain

                 ;*** Program Trim Value to Flash ************************************
                 GotTrim:
                 ;              ldhx   #{EndPrgTrim-PrgTrim} ;Init to size of PrgTrim
EE72 45 0045                  ldhx   #(EndPrgTrim-PrgTrim) ;<CodeWarrior format>
EE75 D6 EE91     Mov2RAM:     lda    PrgTrim,x     ;Get byte from flash and...
EE78 E7 7F                    sta    RamStart-1,x  ;...put into RAM
EE7A 5B F9                    dbnzx  Mov2RAM
EE7C BD 80                    jsr    RamStart      ;Program the Trim value

                 ;*** Toggle PTA4 to check frequency **********************************
EE7E C6 FFC0     V_TRIMLOC:   lda    TRIMLOC
EE81 B7 38                    sta    OSCTRIM       ;Verify trim value in flash
EE83 16 04                    bset   3,DDRA
EE85 17 00       togPTA3:     bclr   3,PTA         ;[4]
EE87 21 FE                    brn    *             ;[3]keep 50% duty cycle on toggle
EE89 21 FE                    brn    *             ;[3]keep 50% duty cycle on toggle
EE8B 16 00                    bset   3,PTA         ;[4]
EE8D 21 FE                    brn    *             ;[3]keep 50% duty cycle on toggle
EE8F 20 F4                    bra    togPTA3       ;[3]PTA3 will toggle @ 20x bus

                 ;*** Subroutines *****************************************************
                 ; PrgTrim:
                 ; Programs the value in the OSCTRIM register into the reserved Flash
                 ; location for the trim value, TRIMLOC = $FFC0.
                 ; Entry Conditions:
                 ;  1. This routine has been copied into RAM.
                 ;  2. The TRIMLOC ($FFC0) location in flash is blank ($FF).
                 ;  3. The OSCTRIM register contains the desired trim value.
                 ;
                 ; Exit conditions:
                 ;  1. TRIMLOC is programmed with the value in OSCTRIM.
                 ;  2. OSCTRIM is not modified.
                 ;  3. A,H,X registers are saved and restored.

                 PrgTrim:
EE91 9D                       nop                  ;Sacrificial byte, not moved to RAM
EE92 87                       psha
EE93 8B                       pshh
EE94 89                       pshx                 ;Save A,H,X registers
```

```
EE95 A6 01                  lda    #initFLCR     ;Programming step 1
EE97 C7 FE08                sta    FLCR

EE9A C6 FFBE                lda    FLBPR         ;Programming step 2

EE9D C7 FFC0                sta    TRIMLOC       ;Programming step 3

EEA0 AE 01                  ldx    #tnvs         ;DELAY FOR tnvs
EEA2 A6 0D                  lda    #speed
EEA4 CD 280C                jsr    Delnus        ;Programming step 4

EEA7 A6 09                  lda    #hvenFLCR     ;Programming step 5
EEA9 C7 FE08                sta    FLCR

EEAC AE 01                  ldx    #tpgs         ;DELAY FOR tpgs
EEAE A6 0D                  lda    #speed
EEB0 CD 280C                jsr    Delnus        ;Programming step 6

EEB3 B6 38                  lda    OSCTRIM       ;Programming step 7
EEB5 C7 FFC0                sta    TRIMLOC       ;Write OSCTRIM to TRIMLOC

EEB8 AE 03                  ldx    #tprog        ;DELAY FOR tprog
EEBA A6 0D                  lda    #speed
EEBC CD 280C                jsr    Delnus        ;Programming step 8

EEBF A6 08                  lda    #clrprgFLCR   ;Programming step 10
EEC1 C7 FE08                sta    FLCR

EEC4 AE 01                  ldx    #tnvh         ;DELAY FOR tnvh
EEC6 A6 0D                  lda    #speed
EEC8 CD 280C                jsr    Delnus        ;Programming step 11

EECB 4F                     clra
EECC C7 FE08                sta    FLCR          ;Programming step 12

EECF 21 FE                  brn    *
EED1 21 FE                  brn    *             ;Programming step 13

EED3 88                     pulx
EED4 8A                     pulh
EED5 86                     pula                 ;Restore A,H,X registers

EED6 81          EndPrgTrim:  rts

                 ;*** Interrupt Service Routines ************************************
                 ; TIM_isr:
                 ; This interrupt service routine uses the 1st input capture as a sync
                 ; edge for the reference clock on the selected Timer channel.
                 ; Then it reads the selected timer channel's counter register. Based
                 ; on the status of the Sync flag variable in RAM, either the channel
                 ; counter data is saved as an offset or the actual count is calculated
                 ; by subtracting the offset from the current channel value.
                 ;
                 ; This routine MUST be shorter in duration than 1 period of the
```

```
                ; reference clock. Otherwise, the input capture interrupts will keep
                ; stacking until RAM overflows and an illegal address reset occurs.

EED7 8B         TIM_isr:    pshh                ;Save H, since it is not stacked
EED8 B6 28                  lda    TSCHNL       ;Clear the int flag...
EEDA A4 7F                  and    #$7F         ;...without modifying the...
EEDC B7 28                  sta    TSCHNL       ;...settings in Channel Register
EEDE 55 29                  ldhx   TCHNLH       ;Get the captured channel value
EEE0 00 80 06               brset  0,Sync,GetActual ;Check if Offset or Actual
EEE3 35 81                  sthx   SyncOffsetH  ;Save as Offset
EEE5 10 80                  bset   0,Sync       ;Set Offset captured flag
EEE7 20 12                  bra    TIM_rti      ;Return from int
EEE9 12 80      GetActual:  bset   1,Sync       ;Set actual measure taken flag
EEEB 9F                     txa                 ;Move the low byte of value to A
EEEC B0 82                  sub    SyncOffsetL  ;Sub low byte of Offset
EEEE B7 84                  sta    ActualL      ;Save low byte of Actual
EEF0 8B                     pshh                ;Transfer high byte of Offset...
EEF1 86                     pula                ;...to A
EEF2 B2 81                  sbc    SyncOffsetH  ;Sub high byte of Offset & carry
EEF4 B7 83                  sta    ActualH      ;Save high byte of Actual
EEF6 98                     clc                 ;Clear carry bit for divide...
EEF7 36 83                  ror    ActualH      ;...divide ActualH by 2...
EEF9 36 84                  ror    ActualL      ;...divide ActualL by 2
EEFB 8A         TIM_rti:    pulh                ;Restore H before unstacking
EEFC 80                     rti

                ;*** Vectors ***********************************************************
                            org    Vtimch1
FFF4 EED7                   fdb    TIM_isr      ;Timer ch1 int vector
FFF6 EED7                   fdb    TIM_isr      ;Timer ch0 int vector

                            org    Vreset
FFFE EE00                   fdb    Start        ;Reset vector
                END
```

This page intentionally left blank.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com