Introduction

This application note addresses several methods used to trim or calibrate the internal clock generator (ICG)’s internal reference generator (IRG) residing on the MC9S08GB/GT Family of microcontrollers. (The MC9S08GB/GT Family will be referred to as the GB/GT Family in this application note.) The IRG is a low-cost alternative to an external crystal or resonator, since using the internal reference generator requires no external components. Software is provided for one of the IRG calibration methods discussed in the document.

Features of the Internal Reference Generator

The IRG is easy to use as the reference oscillator for the ICG residing on the GB/GT Family. The IRG is designed to operate at 243 kHz, but can vary in frequency from part to part by as much as ±25% due to processing variations in manufacturing. This family’s IRG is designed such that it can be trimmed by up to ±25% to obtain the desired 243-kHz base frequency. Once trimmed, the IRG will remain calibrated to within ±5% of the trimmed frequency across the operating temperature and voltage range of the GB/GT Family microcontroller.

The GB/GT Family’s ICG has frequency-lock loop (FLL) circuitry which allows the IRG to be multiplied and divided by various integer values to obtain bus clock frequencies ranging from approximately 34.47 kHz to approximately 19.99 MHz when calibrated. When the IRG is used as the reference clock for the GB/GT Family, the ICG will be placed in mode 3: FLL engaged-internal (FEI) by the user software. FEI mode means the system will use the IRG as a reference frequency and the FLL will be used to multiply and/or divide the 243-kHz reference for use by the system to create the bus clock. Actual setup and use of the ICG is beyond the scope of this document and is discussed in AN2494, Configuring the System and Peripheral Clocks in the MC9S08GB/GT. Also, refer to the GB/GT Family data sheet (MC9S08GB60/D) for more information.
ADVB24496/D

Advantages of Using the Internal Reference Generator

The IRG clock reference source:

- Requires no external components
- Does not require any pins
- Starts up very quickly after being powered down
- Provides a reference clock to the system of sufficient accuracy to be used by its SCI (when calibrated)

Extra cost associated with purchasing, maintaining inventory, and assembling the external components is eliminated. The I/O pins normally associated with the oscillator (XTAL and EXTAL) are free for use with other chip functions such as general-purpose I/O.

Uncalibrated Operation of the Internal Reference Generator

In a system that has relatively loose timing requirements (where up to \( \pm 25\% \) bus frequency variations can be tolerated), no IRG calibration is necessary. When the IRG is not calibrated, the ICG maximum bus frequency in mode 3 must be limited to 15.55 MHz nominal frequency. Any higher frequency could exceed the ICG’s maximum bus frequency of 20 MHz.

When operating in mode 3, the higher speed bus frequency selections are 17.77 MHz and 19.99 MHz nominal. These nominal bus frequencies require a calibrated 243-kHz reference.

The 15.55-MHz maximum nominal frequency choice will assume the 243-kHz reference oscillator could be running untrimmed as fast as 125% of its nominal frequency. The next fastest frequency choice in mode 3 would be 17.77 MHz. If the untrimmed 243-kHz oscillator is operating at 125% of its nominal frequency, using any higher bus frequency than 15.55 MHz would exceed the maximum operating frequency of 20 MHz.

The 15.55-MHz selection, operating with the IRG uncalibrated at \( \pm 25\% \) of its nominal frequency will yield a bus frequency ranging from 11.66 MHz to 19.44 MHz.

Frequency/Period versus Trim Value

The IRG frequency can be trimmed by \( \pm 25\% \) by writing an 8-bit value to the trim register (ICGTRM). This trim value is set to $80 after any reset. Increasing the value of ICGTRM by 1 increases the period (decreases the frequency) of the oscillator by approximately 0.2%. Respectively, decreasing the value of ICGTRM by 1 decreases the period (increases the frequency) 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, this relationship is not perfectly linear. Also, circuit board layout and especially the placement of the \( V_{DD} \) to \( V_{SS} \) bypass capacitors in relation to the \( V_{DD}/V_{SS} \) pins can influence the resulting period due to system noise. Placing these power

For More Information On This Product, Go to: www.freescale.com
supply bypass capacitors very close to the V_DD and V_SS pins improves IRG as well as overall system performance.

**Figure 1** shows an oscillator period versus trim value plot of an ideal device. The ideal plot is calculated starting at a trim value of $80 (128 decimal). The default bus period is derived from the value at $80 (in this case, 4.11 \mu s). In an ideal device, the period would increase 0.2% for each increment of 1 in the OSCTRM register.

![Figure 1. Oscillator Period vs. Trim](image)

The ideal waveform is perfectly linear; however, the actual period curves typically have a slight saw-toothed characteristic that can result in a non-monotonic relationship. 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. Second, 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 mostly linear compared to the period, and therefore it is inversely related to frequency. Figure 2 shows the same data as Figure 2, except as oscillator frequency versus trim value.

Figure 2. Oscillator Frequency vs. Trim

A FLASH location has been reserved to store the trim value after a calibration routine has been performed. This location is called NVICGTRIM and resides at memory address $FFBE. This location is not automatically loaded into the ICGTRM register after a system reset. It is the responsibility of the user to copy this value into ICGTRM if needed. The value of OSCTRIM is used when the ICG is operated in mode 3, FLL engaged internal (FEI) by user software.
This section will describe in detail reliable methods for trimming the internal reference generator to within 0.4% of its ideal frequency of 243 kHz.

### Auto-Trimming Methods

This section will discuss and give detailed examples of these auto-trimming methods:

- Linear search
- Sequential search
- Binary search
- Iterative linear search

All of these trim methods require an external reference clock to be fed into either a timer channel or a general-purpose I/O port. This reference clock should be at least 100 times slower than the bus clock. 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, the value should be programmed into FLASH memory, NVICGTRIM located at $FFBE. As part of the reset routine of the main user code, the value in NVICGTRIM should be copied into the ICGTRM 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 1 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, the oscillator is running too fast; slow it down by adding the offset to the current ICGTRM value. If the actual measurement was lower than the expected, speed up the oscillator by subtracting the offset from the current ICGTRM value.

This method can be simplified by setting up the timer or counting loop such that a difference of 1 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. Therefore the result could be off by as much as 5%.

#### Sequential Search

Set ICGTRM 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 ICGTRM by 1 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 must 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**

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

The drawback to this method is that since the saw-toothed relationship between the trim value and period results in non-monotonic trim values, the binary search could hit a peak or valley that is severe enough to cause the search to branch in the wrong direction. This could cause the final trimmed period to be off by a couple of percent. This can be overcome by using the binary search for only five iterations instead of eight. Then switch to a sequential search for the final three bits of the trim value.

**Iterative Linear Search**

The iterative linear search uses the linear search as its basis. However, instead of stopping after one measurement of the reference clock, this method iteratively measures the reference after each update of ICGTRM 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 should be counted. After a defined number of loops, the routine switches to a sequential search to settle into 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. The two versions of the code have different reference clock sources. In the first example, an RS-232 terminal is used to provide the reference clock by sending a specific character (in this case the @ character) into a timer channel 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 the timer channel.
The Program

This auto-trim program uses the iterative linear search to find the best trim value to set the frequency of the IRG to 243 kHz. There are as many as a total of eight timer input capture/output compare channels on the GB/GT Family, depending on the particular device and package. Two timer channels are used by the calibration program. An input capture channel is used to capture the external timing reference. A timer output compare channel is used as a “user interface” to output a square wave frequency to verify completion of the calibration task. Both of these timer channels are selected by the user and can be any two of the available timer channels on the GB/GT Family. The timer channels to be used are selected in the assembly code by setting two of eight conditional assembly switches to logic 1. The rest of the switches must be set to logic 0. The input capture and output compare channels must not be the same channel. An equate file named 9S08GB_GT_Timer_Selection.inc is common to both the 9600-baud and 60-Hz reference input frequency programs. The user will select the input capture and output compare channels used by the calibration program. The beginning section of the header file appears as follows:

```
;*** Conditional assembly switches and data *************************
; NOTE INPUT AND OUTPUT CHANNEL MUST BE DIFFERENT
 ; Change the one of the following eight TIMERxCHx choices to 1 for the
 ; timer INPUT channel to be used by the program and the set rest to 0.
 ; Then save and reassemble the source.
IN_TIMER1CH0 equ 1           ;User chooses timer 1 channel 0 as input in this case
IN_TIMER1CH1 equ 0
IN_TIMER1CH2 equ 0
IN_TIMER2CH0 equ 0
IN_TIMER2CH1 equ 0
IN_TIMER2CH2 equ 0
IN_TIMER2CH3 equ 0
IN_TIMER2CH4 equ 0

; Change the one of the following eight TIMERxCHx choices to 1 for the
; timer OUTPUT channel to be used by the program and the set rest to 0.
; Then save and reassemble the source.
OUT_TIMER1CH0 equ 0
OUT_TIMER1CH1 equ 1           ;User chooses timer 1 channel 1 as output in this case
OUT_TIMER1CH2 equ 0
OUT_TIMER2CH0 equ 0
OUT_TIMER2CH1 equ 0
OUT_TIMER2CH2 equ 0
OUT_TIMER2CH3 equ 0
OUT_TIMER2CH4 equ 0
```

Note, the first conditional assembly switch reads “IN_TIMER1CH0 equ 1 ...”. That assembler directive flags the assembler to use timer 1 channel 0 as the input for the external reference clock. As an example, to use timer 2, channel 4 as the input for the external reference clock, change the first entry to “IN_TIMER1CH0 equ 0” and the eighth entry to “IN_TIMER2CH4 equ 1”. Then reassemble the program, using the CodeWarrior assembler. Now timer 2, channel 4 will be used as the input capture channel for the external reference clock.
Using the same logic, the line reading “OUT_TIMER1CH1 equ 1…” sets the output compare channel as timer 1, channel 1. As an example, to use timer 2, channel 3 as output compare channel, change the entry “OUT_TIMER1CH1 equ 0…” and the entry to “OUT_TIMER2CH3 equ 1…”. Then reassemble the program, using the CodeWarrior assembler. Now timer 2, channel 3 will be used as the output capture channel for the program’s completion output.

Figure 3 shows a flowchart of the calibration program. The program starts by initializing the microcontroller’s configuration registers, timer, and a few RAM variables used by the program. It checks the current value in NVICGTRIM. If NVICGTRIM is not erased, the program assumes a trim value had previously been programmed into NVICGTRIM. The value of NVICGTRIM is then loaded into the ICGTRM register. Control is then transferred to the end of the program where the selected timer output compare channel’s pin will output a square wave in the range of 12.4 kHz to 20.8 kHz. At this time, the bus clock is set to run at approximately 1.6663 MHz, using the trim value from NVICGTRIM. The selected timer output compare channel’s pin is now being toggled at 1% of the actual bus clock frequency. The reason for this wide range in frequency is because the 243-kHz IRG could be $\pm 25\%$ of its nominal value if the value in NVICGTRIM is not an accurate trim value.

This low output frequency on the selected timer output compare channel’s pin is an indication to the user that the FLASH memory at NVICGTRIM is not erased. The program could also output this low frequency because of a FLASH programming error during the NVICGTRIM programming sequence. This programming error is very unlikely to occur.
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

Figure 3. ICG Calibration Program Flow Chart
The program initializes the selected timer input capture and output compare channel pins. The input capture is set to capture rising edges. The first timer input capture is received at a rising edge of the reference clock. The timer channel capture registers are read and stored in RAM as an offset value. The input capture flag is cleared and the program continues to obtain the next rising edge of the external reference clock. The offset that was saved after the first input capture is subtracted from the timer value from the second input capture. The difference between the two is the measured value of the reference clock's period. This value is saved in RAM as the actual value.

Note that 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 resetting from $FFFF to $0000. Subtracting $013D – $FFF0 will still result in the correct actual value of $014D.

Once an actual value has been captured, the program will determine 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 prescaler value appropriately in relation to the period of the reference clock, this delta value can be used to adjust to the ICGTRM value. Later sections in this application note will explain this in detail.

If the oscillator is too fast, the delta is added to the ICGTRM value. A check is made to verify that the sum is not greater than $FF. If it is, ICGTRM is set to $FF, otherwise it is set to the sum of ICGTRM and delta. The program now returns to obtain another external reference clock time from the pair of input captures.

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

If the oscillator matches the desired frequency (1.6664 MHz), then the program jumps out of the loop and programs the ICGTRM value into NVICGTRIM, located in FLASH memory.

After successful completion of the calibration process, the value of NVICGTRIM is moved back into ICGTRM. A square wave with a frequency of 10% of the bus clock frequency is output to the selected timer output compare channel’s pin. The square wave frequency will be very close to 166.63 kHz (10% of the calibrated CPU bus frequency), its calibrated value. This allows the user to verify that the correct value is programmed into NVICGTRIM.
To prevent the case where the ICGTRM 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 ICGTRM value as the desired value.

**Trim RS-232 Terminal as Reference**

By using an RS-232 terminal, the reference clock is provided by repeatedly sending the @ character at 9600 baud to the timer channels through an RS-232 terminal or PC terminal emulator. Figure 4 shows two possible circuits for connecting an RS-232 port to one of the timer channel inputs. In this example, the user will have to repeatedly send characters until the trim value has been found by holding the “@” key down on the terminal. When the selected timer output compare channel’s pin begins to output a square wave of approximately 166.63 kHz, the trim value has been found and is programmed into FLASH. If a 12.4-kHz to 20.8-kHz frequency is output, there has been an error as previously discussed.

![Figure 4. RS-232 Connection Circuit](image)

For the program to work best, set up 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:

\[ \text{Expected value} = \left( f_{\text{Bus}} \times T_{\text{ref}} \right) / \text{PS} \]

Where,

- \( f_{\text{Bus}} \) = the trimmed bus frequency (1.6663 MHz)
- \( T_{\text{ref}} \) = the period of the reference clock
- \( \text{PS} \) = the prescaler of the timer module or 1, 2, 4, 8, 16, 32, 64, 128

At 9600 baud, each bit is 104.16 microseconds long. The @ character is used because its hex value is $40 or %01000000. Since RS-232 levels hold the transmit (Tx) line high between characters, the @ character provides exactly two rising edges, exactly 2 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:

```
1110000000101111
```

Idle line Start bit "@" character, ASCII $40, sent LSB Stop bit Idle line

The result is two rising edges at 208.3 \( \mu \text{s} \) apart. Therefore, \( T_{\text{ref}} = 208.3 \mu \text{s} \). So, starting with a timer prescaler, \( \text{PS} = 1 \), our expected value is:

\[ \text{Expected value} = (1.6663 \text{ MHz} \times 208 \mu \text{s})/1 = 346.5 \text{ (approximately 347)} \]

The desired timer resolution should be between 0.2% and 0.4% of the expected value. In this example, with a timer prescaler of 1, the resolution is 1/347 = 0.29%. This is a good value for the program.

One more trick is used to simplify the program. Since the untrimmed frequency can vary by ±25% from the 1.6663-MHz expected value, calculate the extremes of the expected values:

- \( +25\% = 1.25 \times \text{expected value} = 1.25 \times 347 = 434 = \$1B2 \)
- \( -25\% = 0.75 \times \text{expected value} = 0.75 \times 347 = 260 = \$104 \)

Notice the extreme values fall in a range between $100 and $1FF. This will enable math to be performed on only the low byte of the external reference value to obtain a trim value. An example of applying an offset to keep the program simpler will be shown in the 60-Hz calibration example.
Trim with 50-/60-Hz Clock as Reference

An ac power line can be used as a reference clock by using the circuit in Figure 4 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.

![Circuit Diagram](image)

**Figure 5. 60-Hz Square Wave Generator**

Calculate the expected value and timer prescaler setting using the same methods as in the RS-232 example. In this example:

\[ f_{\text{Bus}} = 1.6663 \text{ MHz} \]
\[ T_{\text{ref}} = \frac{1}{60 \text{ Hz}} = 16.7 \mu\text{s} \]

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

\[ \text{Expected value} = \left( f_{\text{Bus}} \times T_{\text{ref}} \right) / PS \]
\[ \text{Expected value} = \left( 1.6663 \text{ MHz} \times \left( \frac{1}{60 \text{ Hz}} \right) \right) / 64 = 433.9 \text{ (approximately 434)} \]

In this case, the resolution is \( 1/434 = 0.23\% \). We are using a prescaler value of 64 in this case. This results in expected values of

\[ \text{Expected value} = 434 = \$1B2 \]
\[ +25\% = 1.25 \times \text{expected value} = 1.25 \times 434 = 542 = \$21E \]
\[ -25\% = 0.75 \times \text{expected value} = 0.75 \times 434 = 325 = \$145 \]

If the maximum expected value is left at \$21E, perform math on both the high and low bytes of the captured time during the calibration calculation portion of the program to determine the delta. However, if an adjustment is subtracted from both the expected and actual values, the possible range of values between \$100 and \$1FF is limited. The high byte can be ignored and all calculations performed on the low byte only. In this example, an adjustment of \$1F is subtracted from both the expected and actual values, resulting in a range of possible actual values from \$126 to \$1FF and an expected adjusted value of \$194.
Running the Calibration Program

Both the 60-Hz or 9600-baud calibration programs are designed to execute in the RAM on the GB/GT Family microcontrollers. Both programs assume the user has erased FLASH at NVICGTRIM ($FFBE). NVICGTRIM is the FLASH location where the trim or calibration value will be stored at the conclusion of the calibration procedure.

First, the user must determined whether a 60-Hz or 9600-baud external timing reference source will be used to perform IRG calibration. That choice will determine which program to load into the microcontroller. If the default timer input capture (T1CH0) and output compare (T1CH1) channels are not selected, the user must change the statements in the header file, pick a new timer(s), and re-assemble the program before proceeding.

If the user selects the 9600-baud external reference clock, a terminal or terminal emulator set to 9600 baud, one stop bit and no parity will be needed.

To perform the IRG calibration, perform the following steps:

1. Connect the external timing reference (60-Hz or 9600-Baud) to the selected timer input capture channel.
2. Connect the P&E multi-link cable to the back ground debug port on the PC board containing the GB/GT Family microcontroller.
3. Connect an oscilloscope or frequency counter to the selected timer output compare channel pin.
4. Apply power to the system.
5. Verify NVICGTRIM at $FFBE is erased and equal to $FF, using the debugger.
6. Load the appropriate oscillator trim program with the debugger for the 60-Hz or 9600-baud external reference clock.
7. Using the debugger, type RESET (cr) and then G 80 (cr).
8. If you are not using a 9600-baud reference clock, skip to step 10
9. If you are using the 9600-baud reference clock, hold the @ key down until there is an output on the selected timer’s output compare pin.
10. There will be a square wave output to the selected timer output compare channel pin at the conclusion of the calibration procedure. It will be on one of two forms:
   - There will be a frequency very close to 166.63 kHz if the calibration is successful. The procedure is complete at this time.
   - If there is a square wave in the range of 12.4 kHz to 20.8 kHz, an error was detected. The wide range is because the IRG may not be calibrated. The most likely reason for this error is NVICGTRIM was

For More Information On This Product, Go to: www.freescale.com
not erased when the calibration program was executed. At this time, the IRG will use the value found in NVICGTRIM as a trim value. With the debugger, verify NVICGTRIM is erased and if it is not, erase it and repeat the procedure. A second and very unlikely error could result from a FLASH programming error during the FLASH programming process. If NVICGTRIM is, in fact, erased and you suspect a FLASH programming error, repeat the procedure. That will verify the presence of the unlikely programming error.

**Conclusion**

The internal clock generator is a new module on the HCS08 Family of microcontrollers. It has the advantages of requiring no external components and conserving I/O pins on the device when the internal reference generator is used. The characteristics of this oscillator make it accurate over temperature and voltage variations once it has been trimmed. The IRG is an ideal clock source for low cost systems.

---

**Downloading and Using the Accompanying Software**

Two zip files accompany this application note, AN2496SW1.zip and AN2496SW2.zip. AN2496SW1.zip contains the assembly files for both the 9600-baud and 60-Hz versions of the IRG calibration programs along with the necessary equate files. This zip file does not contain compiled versions of the programs.

AN2496SW2.zip contains two complete project folders, one for each version of the calibration program. These project directories are for use with MetroWerks CodeWarrior for CW08_V3.0. The folders are named 9S08GBGT_IRG_Cal_9600 and 9S08GBGT_IRG_Cal_60Hz. Inside the first level of each project folder is a CodeWarrior project file with a ".mcp" filename extension. Double clicking these files will open the project if CodeWarrior has been installed. Each project has been assembled, and listings (".lst" file extensions) are available in the "bin" subfolders. Also, the s-records (".s19" file extensions) are available in the same folder.

---

**For More Information On This Product,**

Go to: www.freescale.com
Appendix A: Timer Selection Header File

The following is the common header file used by both calibration programs to select the input capture and output compare channels used by the IRG calibration program.

;*********************************************************************
;* Filename: 9S08GB_GT_Timer_Selection.inc Copyright (c) 2003
;*********************************************************************
;*********************************************************************
;* Oscillator Trim Routine for 9S08GB/GT Family  Bill Lucas/Scott Pape
;*
;* Rev: 1.0   Date: 18July2002 Scott Pape for QT
;* Rev: 1.1   Date: 2January2003 Bill Lucas Ported to 9S08GB/GT
;*
;*********************************************************************
;*********************************************************************

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.

;*** Conditional assembly switches and data ***********************

; NOTE INPUT AND OUTPUT CHANNEL MUST BE DIFFERENT

; Change the one of the following eight TIMERxCHx choices to 1 for the timer INPUT channel to be used by the program and the set rest to 0. Then save and reassemble the source.
IN_TIMER1CH0 equ 1 ;User chooses timer 1 channel 0
IN_TIMER1CH1 equ 0 ; as input in this case

Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

For More Information On This Product, Go to: www.freescale.com
IN_TIMER1CH2 equ    0
IN_TIMER2CH0 equ    0
IN_TIMER2CH1 equ    0
IN_TIMER2CH2 equ    0
IN_TIMER2CH3 equ    0
IN_TIMER2CH4 equ    0

; Change the one of the following eight TIMERxCHx choices to 1 for the
; timer OUTPUT channel to be used by the program and the set rest to 0.
; Then save and reassemble the source.
OUT_TIMER1CH0 equ   0
OUT_TIMER1CH1 equ   1             ;User chooses timer 1 channel 1
OUT_TIMER1CH2 equ   0             ; as output in this case
OUT_TIMER2CH0 equ   0
OUT_TIMER2CH1 equ   0
OUT_TIMER2CH2 equ   0
OUT_TIMER2CH3 equ   0
OUT_TIMER2CH4 equ   0

; Input channel data:

IF     IN_TIMER1CH0  ;if true, choose timer 1, channel 0
IN_TPMxSC    equ    TPM1SC
IN_TPMxCnSC  equ    TPM1C0SC
IN_TPMxCnVH  equ    TPM1C0VH
ENDIF

IF     IN_TIMER1CH1  ;if true, choose timer 1, channel 1
IN_TPMxSC    equ    TPM1SC
IN_TPMxCnSC  equ    TPM1C1SC
IN_TPMxCnVH  equ    TPM1C1VH
ENDIF

IF     IN_TIMER1CH2  ;if true, choose timer 1, channel 2
IN_TPMxSC    equ    TPM1SC
IN_TPMxCnSC  equ    TPM1C2SC
IN_TPMxCnVH  equ    TPM1C2VH
ENDIF

IF     IN_TIMER2CH0  ;if true, choose timer 2, channel 0
IN_TPMxSC    equ    TPM2SC
IN_TPMxCnSC  equ    TPM2C0SC
IN_TPMxCnVH  equ    TPM2C0VH
ENDIF

IF     IN_TIMER2CH1  ;if true, choose timer 2, channel 1
IN_TPMxSC    equ    TPM2SC
IN_TPMxCnSC  equ    TPM2C1SC
IN_TPMxCnVH  equ    TPM2C1VH
ENDIF

IF     IN_TIMER2CH2  ;if true, choose timer 2, channel 2
IN_TPMxSC    equ    TPM2SC
IN_TPMxCnSC  equ    TPM2C2SC
IN_TPMxCnVH  equ    TPM2C2VH

For More Information On This Product,
Go to: www.freescale.com
ENDIF

IF IN_TIMER2CH3 ;if true, choose timer 2, channel 3
IN_TPMxSC equ TPM2SC
IN_TPMxCNSC equ TPM2C3SC
IN_TPMxCNVH equ TPM2C3VH
ENDIF

IF IN_TIMER2CH4 ;if true, choose timer 2, channel 4
IN_TPMxSC equ TPM2SC
IN_TPMxCNSC equ TPM2C4SC
IN_TPMxCNVH equ TPM2C4VH
ENDIF

; Output channel data:

IF OUT_TIMER1CH0 ;if true, choose timer 1, channel 0
OUT_TPMxSC equ TPM1SC
OUT_TPMxCNSC equ TPM1C0SC
OUT_TPMxCNVH equ TPM1C0VH
OUT_TPMxCNVL equ TPM1C0VL
OUT_TPMxMODH equ TPM1MODH
OUT_TPMxMODL equ TPM1MODL
ENDIF

IF OUT_TIMER1CH1 ;if true, choose timer 1, channel 1
OUT_TPMxSC equ TPM1SC
OUT_TPMxCNSC equ TPM1C1SC
OUT_TPMxCNVH equ TPM1C1VH
OUT_TPMxCNVL equ TPM1C1VL
OUT_TPMxMODH equ TPM1MODH
OUT_TPMxMODL equ TPM1MODL
ENDIF

IF OUT_TIMER1CH2 ;if true, choose timer 1, channel 2
OUT_TPMxSC equ TPM1SC
OUT_TPMxCNSC equ TPM1C2SC
OUT_TPMxCNVH equ TPM1C2VH
OUT_TPMxCNVL equ TPM1C2VL
OUT_TPMxMODH equ TPM1MODH
OUT_TPMxMODL equ TPM1MODL
ENDIF

IF OUT_TIMER2CH0 ;if true, choose timer 2, channel 0
OUT_TPMxSC equ TPM2SC
OUT_TPMxCNSC equ TPM2C0SC
OUT_TPMxCNVH equ TPM2C0VH
OUT_TPMxCNVL equ TPM2C0VL
OUT_TPMxMODH equ TPM2MODH
OUT_TPMxMODL equ TPM2MODL
ENDIF

IF OUT_TIMER2CH1 ;if true, choose timer 2, channel 1
OUT_TPMxSC equ TPM2SC
OUT_TPMxCNSC equ TPM2C1SC
Appendix B: HCS08 Assembly Code Example for RS-232

The following is the assembly code listing for trimming the internal oscillator to 243 kHz using a 9600-baud RS-232 terminal as the reference clock.

```assembly
OUT_TPMxCnVH equ TPM2C1VH
OUT_TPMxCnVL equ TPM2C1VL
OUT_TPMxMODH equ TPM2MODH
OUT_TPMxMODL equ TPM2MODL
ENDIF
IF     OUT_TIMER2CH2 ;if true, choose timer 2, channel 2
OUT_TPMxSC   equ TPM2SC
OUT_TPMxCnSC equ TPM2C2SC
OUT_TPMxCnVH equ TPM2C2VH
OUT_TPMxCnVL equ TPM2C2VL
OUT_TPMxMODH equ TPM2MODH
OUT_TPMxMODL equ TPM2MODL
ENDIF
IF     OUT_TIMER2CH3 ;if true, choose timer 2, channel 3
OUT_TPMxSC   equ TPM2SC
OUT_TPMxCnSC equ TPM2C3SC
OUT_TPMxCnVH equ TPM2C3VH
OUT_TPMxCnVL equ TPM2C3VL
OUT_TPMxMODH equ TPM2MODH
OUT_TPMxMODL equ TPM2MODL
ENDIF
IF     OUT_TIMER2CH4 ;if true, choose timer 2, channel 4
OUT_TPMxSC   equ TPM2SC
OUT_TPMxCnSC equ TPM2C4SC
OUT_TPMxCnVH equ TPM2C4VH
OUT_TPMxCnVL equ TPM2C4VL
OUT_TPMxMODH equ TPM2MODH
OUT_TPMxMODL equ TPM2MODL
ENDIF
```

---

**Appendix B: HCS08 Assembly Code Example for RS-232**

The following is the assembly code listing for trimming the internal oscillator to 243 kHz using a 9600-baud RS-232 terminal as the reference clock.

---

**Freescale Semiconductor, Inc.**

**Appendix B: HCS08 Assembly Code Example for RS-232**

The following is the assembly code listing for trimming the internal oscillator to 243 kHz using a 9600-baud RS-232 terminal as the reference clock.

---

**Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)**

**For More Information On This Product,**
**Go to: www.freescale.com**
This program will trim the internal 243 kHz ICG reference oscillator. The ICG is set to run from the 243 kHz internal oscillator with the FLL engaged to clock the CPU bus at 1.6663 MHz. Timer 1 channel 0 is used to measure the period of an external 9600 baud calibration reference clock. Timer 1 channel 1 is used to output 10% of the 1.6663 MHz CPU bus frequency after the calibration process.

The ICGTRM value is modified if the measured value is higher or lower than the expected value. Once the proper trim value is found, it is programmed into the NVICGTRIM (located at $FFBE) storage location in Flash. The user code MUST copy the value of NVICGTRIM into the ICG ICGTRM register at the beginning of his application code.

Usage Notes:
1. This program requires a 9600 baud reference signal connected to Timer 1, channel 0 (PTD0/TPM1CH0), or the timer of your choice.
2. The program assumes Flash at location NVICGTRIM ($FFBE) is erased.
3. Code for this program is loaded into and executes in RAM.
4. After loading the code into RAM and connecting the 9600 baud signal to timer 1 channel 0, through the debugger, execute a RESET and GO 80 command.
5. At the conclusion of successful ICG calibration, 10% of the bus clock frequency will be output to timer 1 channel 1. The frequency of timer 1 channel 1 will be very close to 166.63 kHz. If an error occurs because Flash at NVICGTRIM was not erased or a Flash programming error was detected, timer 1 channel 1 will toggle in the range from 12.4 kHz to 20.8 kHz, indicating the error.

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 for 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.
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

For More Information On This Product, Go to: www.freescale.com
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

For More Information On This Product, Go to: www.freescale.com
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

For More Information On This Product, Go to: www.freescale.com
210
211
212

??? Subroutines ProgTrim ****************************************************

?* Changed for GB/GT Flash WLL
?ProgTrim:

?Programs the value in the ICGTRM register into the reserved Flash
?location for the trim value, NVICGTRIM @ $FFBE.

?Entry Conditions:

? 1. The NVICGTRIM ($FFBE) location in flash is erased ($FF).
? 2. The ICGTRM register contains the desired trim value.

?Exit conditions:

? 1. NVICGTRIM is programmed with the value in ICGTRM.
? 2. ICGTRM is not modified.
? 3. no registers altered

? 0144 87

?PrgTrim: psha

?0145 A6 30

?lda #FPPVIOL|FACCERR ;error flag bits to clear if set

?0147 C7 1825

?sta FSTAT ;clear any error flags

?014A A6 08

?1.666 MHz/8+1 = 185Khz Flash clock

?014C C7 1820

;divide by 8+1 for 1.66 MHz bus.

?014F B6 4E

?lda ICGTRM ;trim value

?0151 C7 FFBE

;place in Flash

?0154 A6 20

?lda #byte_pgm ;page write command

?0156 C7 1826

;initiate the programming sequence

?0159 C6 1825

?sta FSTAT

?015C AA 80

;This is normally done at start-up, however we

?will initialize here to only program one byte

?014F B6 4E

?lda ICGTRM ;trim value

?0151 C7 FFBE

;place in Flash

?0154 A6 20

?lda #byte_pgm ;page write command

?0156 C7 1826

;initiate the programming sequence

?0159 C6 1825

?sta FSTAT ;clear any error flags

?015C AA 80

;launch the command

?0161 21 FE

;burn bus cycs before checking status

?0163 C6 1825

;look at status register for errors

?0166 A4 30

;wait around until programming complete

?0168 26 A9

;check for programming errors

?016A C6 1825

;wait around until programming complete

?016D A4 40

;command complete yet?

?016F 27 F9

;not done yet..loop until complete

?0171 86

; EndPrgTrim: rts ;done

?250

?*** Subroutine read_timer **********************************************

?* Changed for TB/GT device WLL

?read_timer:

?Waits for a reference clock positive going edge, reads the captured
?value into H:X, clears the input capture flag and returns the timer
?value in register H:X.

?Entry Conditions:

?None

?;
Appendix C: HCS08 Assembly Code Example for 60 Hz

The following is the assembly code listing for trimming the internal oscillator to 243 kHz using a 60-Hz square wave as the reference clock.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

For More Information On This Product, Go to: www.freescale.com
Appendix C: HCS08 Assembly Code Example for 60 Hz

Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

For More Information On This Product, Go to: www.freescale.com
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

<table>
<thead>
<tr>
<th>Rel. Loc</th>
<th>Obj. code</th>
<th>Source line</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>00BF CD 0173</td>
<td>jsr read_timer ;get the second edge</td>
</tr>
<tr>
<td>133</td>
<td>00C2 9F</td>
<td>txa ;low byte of second edge time</td>
</tr>
<tr>
<td>134</td>
<td>00C3 B0 83</td>
<td>sub SyncOffsetL ;compute the time difference</td>
</tr>
<tr>
<td>135</td>
<td>00C5 B7 85</td>
<td>sta ActualL ;low byte of Actual</td>
</tr>
<tr>
<td>136</td>
<td>00C7 8B</td>
<td>pshh ;high byte of second edge time</td>
</tr>
<tr>
<td>137</td>
<td>00C8 86</td>
<td>pula</td>
</tr>
<tr>
<td>138</td>
<td>00C9 B2 82</td>
<td>sbc SyncOffsetH</td>
</tr>
<tr>
<td>139</td>
<td>00CB B7 84</td>
<td>sta ActualH ;high byte of Actual</td>
</tr>
<tr>
<td>140</td>
<td>00CD B6 85</td>
<td>lda ActualL ;Adjust the Actual value so...</td>
</tr>
<tr>
<td>141</td>
<td>00CF A0 1F</td>
<td>sub #Adjust ;...the range fits into 1 byte</td>
</tr>
<tr>
<td>142</td>
<td>00D1 B7 85</td>
<td>sta ActualL</td>
</tr>
<tr>
<td>143</td>
<td>00D3 A1 93</td>
<td>cmp #ExpectedL ;Compare Actual with Expected value</td>
</tr>
<tr>
<td>144</td>
<td>00D5 27 52</td>
<td>beq GotTrim ;If equal, osc is trimmed!</td>
</tr>
<tr>
<td>145</td>
<td>00D7 25 1D</td>
<td>bco TooSlow ;If lower, osc is too slow</td>
</tr>
<tr>
<td>146</td>
<td>00DA A9 93</td>
<td>sub #ExpectedL ;Otherwise, osc is too fast</td>
</tr>
<tr>
<td>147</td>
<td>00DB B7 86</td>
<td>sta Delta ;Calculate Delta from Act-Exp</td>
</tr>
<tr>
<td>148</td>
<td>00DD B6 4E</td>
<td>lda ICGTRM</td>
</tr>
<tr>
<td>149</td>
<td>00DF A1 FF</td>
<td>cmp #$FF ;Check if trim=$FF</td>
</tr>
<tr>
<td>150</td>
<td>00E1 27 46</td>
<td>beq GotTrim ;if true, can't slow more</td>
</tr>
<tr>
<td>151</td>
<td>00E3 0A 87 43</td>
<td>brset 5,LoopCnt,GotTrim ;bit5=32 loops=limit</td>
</tr>
<tr>
<td>152</td>
<td>00E6 09 87 03</td>
<td>brcir 4,LoopCnt,AddDelta ;bit4 = 16 loops</td>
</tr>
<tr>
<td>153</td>
<td>00E9 6E 01 86</td>
<td>mov #$01,Delta ;after 8 loops, just add 1</td>
</tr>
<tr>
<td>154</td>
<td>00EC BB 86</td>
<td>AddDelta: add Delta ;acca still has OSCTRIM</td>
</tr>
<tr>
<td>155</td>
<td>00EE 24 02</td>
<td>bcc AddTrim ;did carry bit get set?</td>
</tr>
<tr>
<td>156</td>
<td>00F0 A6 FF</td>
<td>lda #$FF ;if carry set, max osctrim</td>
</tr>
<tr>
<td>157</td>
<td>00F2 B7 4E</td>
<td>AddTrim: sta ICGTRM ;Trim increases to slow freq</td>
</tr>
<tr>
<td>158</td>
<td>00F4 20 B2</td>
<td>bra TryAgain ;Try new value</td>
</tr>
<tr>
<td>159</td>
<td>00F6 A6 93</td>
<td>TooSlow: lda #ExpectedL</td>
</tr>
<tr>
<td>160</td>
<td>00F8 B0 85</td>
<td>sub ActualL</td>
</tr>
<tr>
<td>161</td>
<td>00FA B7 86</td>
<td>sta Delta ;Calculate Delta from Exp-Act</td>
</tr>
<tr>
<td>162</td>
<td>00FC B6 4E</td>
<td>lda ICGTRM</td>
</tr>
<tr>
<td>163</td>
<td>00FE 27 29</td>
<td>beq GotTrim ;if true, can't speed-up more</td>
</tr>
<tr>
<td>164</td>
<td>0100 0A 87 26</td>
<td>brset 5,LoopCnt,GotTrim ;bit5=32 loops=limit</td>
</tr>
<tr>
<td>165</td>
<td>0103 09 87 03</td>
<td>brcir 4,LoopCnt,SubDelta ;bit4 = 16 loops</td>
</tr>
<tr>
<td>166</td>
<td>0106 6E 01 86</td>
<td>mov #$01,Delta ;after 8 loops, just add 1</td>
</tr>
<tr>
<td>167</td>
<td>0109 B0 86</td>
<td>SubDelta: sub Delta</td>
</tr>
<tr>
<td>168</td>
<td>010B 24 02</td>
<td>bcc SubTrim ;did carry bit get set?</td>
</tr>
<tr>
<td>169</td>
<td>010D A6 FF</td>
<td>lda #$FF ;if carry set, min osctrim</td>
</tr>
<tr>
<td>170</td>
<td>010F B7 4E</td>
<td>SubTrim: sta ICGTRM</td>
</tr>
<tr>
<td>171</td>
<td>0111 20 95</td>
<td>bra TryAgain</td>
</tr>
</tbody>
</table>

173 ;******** Toggle PTD1 @ 12.4 kHz to 20.8 kHz if we have an error ******
174 ;* There are two reasons to be here:
175 ;* 1). NVICGTRIM is not erased
176 ;* 2). There was a Flash error during prog'ing. This error is rare.
177 ; The following timer code will output 1% of the CPU BUSCLOCK to
178 ; the selected output compare channel for ICG frequency monitoring
179
180 181 0113 C6 FFBE | Trim_error: lda NVICGTRIM ;Current Trim value. |
182 0116 B7 4E | sta ICGTRM ;Use the value that's there
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

**Appendix C: HCS08 Assembly Code Example for 60 Hz**

183 0118 6E 14 38         mov    #(mMS1A|mELS1A),OUT_TPMxCnSC ;Toggle on output cmp
184 011B 6E 00 33         mov    #zero,OUT_TPMxMODH
185 011E 6E 31 34         mov    #forty_nine,OUT_TPMxMODL ;Timer modulo reg = 49
186 0121 6E 00 39         mov    #zero,OUT_TPMxCnVH
187 0124 6E 01 3A         mov    #one,OUT_TPMxCnVL ;Timer output compare at 1
188 0127 20 FE           bra    $             ;Let the output TPM pin toggle
189
190 ;*** Program Trim Value to Flash ********************************************
191
192 193 0129 AD 19     GotTrim:     bsr    PrgTrim       ;Program the Trim value
194 012B C6 FFBE          lda    NVICGTRIM     ;Read value from Flash to make sure
195 012E B7 4E           sta    ICGTRM        ;Verf trim value is prog'ed in flash

; The following timer code will output 10% of the CPU BUSCLOCK to
; the selected output compare channel for ICG frequency monitoring

200 0130 6E 08 30         mov    #(mCLKSA),IN_TPMxSC ;bus clk
201 0133 6E 14 38         mov    #(mMS1A|mELS1A),OUT_TPMxCnSC ;Toggle on output cmp
202 0136 6E 00 33         mov    #zero,OUT_TPMxMODH
203 0139 6E 04 34         mov    #four,OUT_TPMxMODL ;Timer modulo register = 4
204 013C 6E 00 39         mov    #zero,OUT_TPMxCnVH
205 013F 6E 01 3A         mov    #one,OUT_TPMxCnVL ;Timer output compare at 1
206 207 0142 20 FE        bra    $             ;Let selected output TPM pin toggle
208
209
210
211 ;*** Subroutines PrgTrim ********************************************
212 ;* Changed for GB/GT Flash WLL
213 ; PrgTrim:
214 ; Programs the value in the ICGTRM register into the reserved Flash
215 ; location for the trim value, NVICGTRIM @$FFBE.
216 ; Entry Conditions:
217 ; 1. The NVICGTRIM ($FFBE) location in flash is erased ($FF).
218 ; 2. The ICGTRM register contains the desired trim value.
219 ;
220 ; Exit Conditions:
221 ; 1. NVICGTRIM is programmed with the value in ICGTRM.
222 ; 2. ICGTRM is not modified.
223 ; 3. no registers altered
224 ;
225 226 0144 87     PrgTrim:     psha
227 0145 A6 30          lda    #mFPVIOL|mFACCERR ;error flag bits to clear if set
228 0147 C7 1825        sta    FSTAT         ;clear any error flags
229 014A A6 08          lda    #mDIV3     ;1.666 MHz/8+1 = 185Khz Flash clock
230 014C C7 1820        sta    FCDIV         ;divide by 8+1 for 1.66 MHz bus.

**For More Information On This Product, Go to:** www.freescale.com
Calibrating the MC9S08GB/GT Internal Clock Generator (ICG)

This is normally done at start-up, however we will initialize here to only program one byte

```
231                  ;                         This is normally done at start-up, however we
232                  ;                         will initialize here to only program one byte

233 014F B6 4E       lda   ICGTRM   ;trim value
234 0151 C7 FFBE     sta   NVICGTRIM   ;place in Flash
235 0154 A6 20       lda   #byte_pgm   ;page write command
236 0156 C7 1826     sta   FCMD   ;initiate the programming sequence
237 0159 C6 1825     lda   FSTAT
238 015c AA 80       ora   #mFCBEF
239 015E C7 1825     sta   NVICGTRIM   ;place in Flash
240 0161 21 FE       brn * ;burn bus cycs before chking status
241 0163 C6 1825     lda   FSTAT   ;look at status register for errors
242 0166 A4 30       and   #mFPVIOL|mFACCERR ;error flag bits
243 0168 26 A9       bne    Trim_error ;check for programming errors
244 016a C6 1825     pgm_loop:   lda   FSTAT   ;wait around unil prog'ing complete
245 016D A4 40       beq    pgm_loop ;not done yet..loop until complete
246 016F 27 F9       pula
247 0171 86
248 0172 81          EndPrgTrim:  rts   ;done

249
250                  ;*** Subroutine read_timer *******************************************
251                  ;* Changed for TB/GT device WLL
252                  ; read_timer:
253                  ; Waits for a reference clock positive giong edge, reads the captured
254                  ; value into H:X, clears the input capture flag and returns the timer
255                  ; capture value in register H:X.
256                  ; Entry Conditions:
257                  ;  None
258
259                  ; Exit conditions:

260 0173 0F 35 FD    brclr  CHOF,IN_TPMxCnSC,read_timer ;loop until see edge
264 0176 55 36       ldhx   IN_TPMxCnVH ;get the counter value
265 017B B6 35       lda   IN_TPMxCnSC ;dummy read of the status register
266 017A 1F 35       brclr  CHOF,IN_TPMxCnSC ;complete the clearing process
267 017C 81          rts
```

(c) COPYRIGHT METROWERKS 1987-2003
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
+46 8 5220080 (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
Donor, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCFOrFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document. Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor 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 or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor 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 Semiconductor was negligent regarding the design or manufacture of the part.