HCS12 FAMILY

EMBEDDED C USING CODEWARRIOR

PROGRAMMING THE PULSE WIDTH MODULATOR

Version 1.0

TECHNOLOGICAL ARTS
#301-533 College St., Toronto, Ontario, Canada M6G 1A8
Toll-free: 1-877-963-8996 (USA and Canada)
Phone: (416) 963-8996 Fax: (416) 963-9179

www.technologicalarts.com

BEFORE READING THIS GUIDE . . .

The source code herein is written in Embedded C using the Metrowerks CodeWarrior 3.1 IDE.

If you are new to CodeWarrior 3.1 or embedded C, you can get a free copy of Technological Arts' *Embedded C Using CodeWarrior - Getting Started Manual* from our website. This manual will show you how to start a project, where to write your source code, compile the code, and finally how to download it to your HCS12 board!

The Embedded C code written herein applies to the 9S12C derivatives of the HCS12 family, however . . .

The PWM concepts are the same for all HCS12 derivatives. The embedded C coding structure for the PWM is similar for all HCS12s plus/minus a few registers.

Where necessary, this document will suggest where you should be concerned about your HCS12 derivative.

PULSE WIDTH WHAT? WHY DO I NEED TO LEARN THIS?

So you want to use your HCS12 module, from Technological Arts, to control the speed of a motor, or make a light flash? Well your eyes are on the right pages!

The **Pulse Width Modulator (PWM),** built into most HCS12s, is the device that allows you to control the speed of a motor, cause an LED to flash, and vary the voltage/current applied to a load.

The purpose of this guide is to:

- Introduce the concept of pulse width modulation
- Define the use and application for a pulse width modulator
- Identify the pulse width modulators (channels) in the HCS12
- Show how an HCS12 PWM channel uses it's registers to derives a pulse width modulated signal
- Demonstrate how to obtain the values for these PWM Registers
- Provide a sample source code used to program the PWM
- Analyze the source code

WHAT IS PULSE WIDTH MODULATION?

Pulse Width Modulation is the process of altering (modulating) the pulse width (duty cycle) of a TTL signal.

Duty cycle is the duration of a TTL signal's period that the signal is high. This value is given as a percentage.

Figure 1 shows a TTL waveform with period T. During the period, the signal is low for 50% of the period and high for the remaining 50% of the period. We say that the signal has a **duty cycle** of 50%.

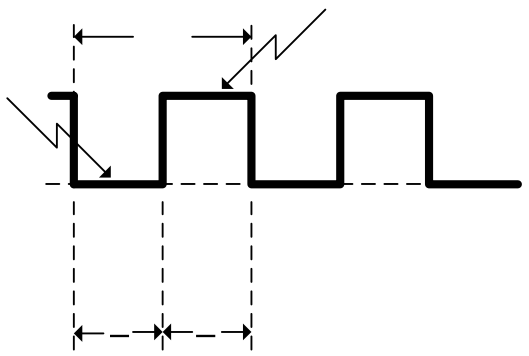

FIGURE 1

Figure 2 shows three independent TTL waveforms with duty cycles of 70%, 80%, and 100% respectively. In this case the duty cycle of the TTL signal is increasing.

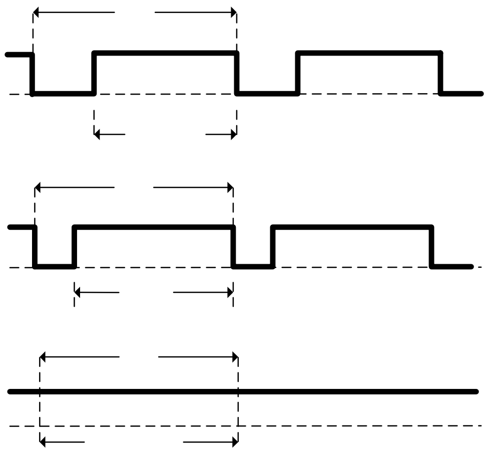

FIGURE 2

WHAT IS A PULSE WIDTH MODULATOR USED FOR?

A pulse width modulator is a device that is programmed to vary a TTL signal's duty cycle. The duty cycle produced can be used to control the amount of voltage/current applied to a load.

Consider **figure 3**. The pulse width modulator generates a signal whose duty cycle controls the power delivered to a motor.

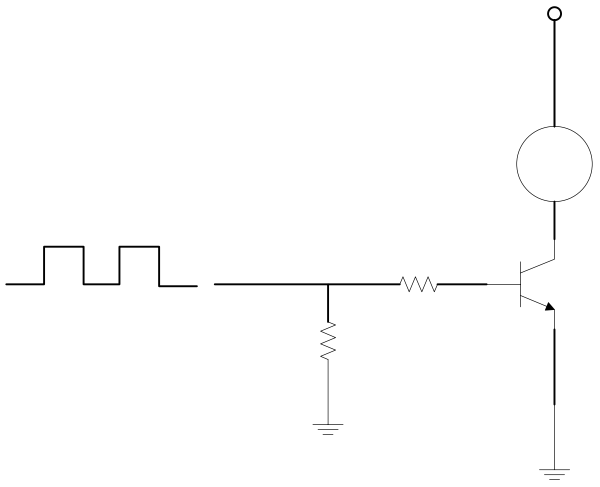

FIGURE 3

If the duty cycle produced by the PWM is 0%, then transistor Q1 is turned off. Therefore, no power is delivered to the motor M.

Table 1 shows what happens to the power delivered to the motor as the duty cycle increases.

Duty Cycle	Transistor Conducts at:	Average Power Delivered To Motor
0%	0%	0%
20%	20%	20%
50%	50%	50%
80%	80%	80%
100%	100%	FULL POWER

TABLE 1

In general, **the power delivered to a load is proportional the to the duty cycle applied to the circuit.** As a result, the PWM can be used for varying light intensity, motor speed and pulsation of light.

THE PWM CHANNELS IN THE HCS12

In the HCS12, a PWM is referred to as a **channel**. For example, an HCS12 that has four PWM channels means that the HCS12 has four PWMs.

Most HCS12 derivatives come with one or more PWM channel(s).

The 9S12C32, **Figure 4,** has five PWM channels.

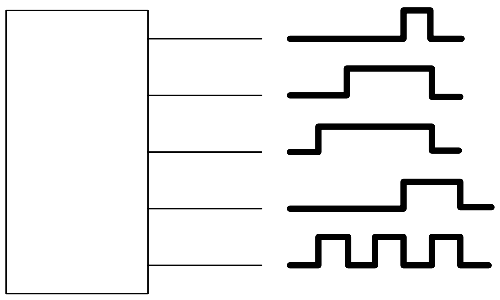

FIGURE 4

Each channel can be individually programmed to produce a signal with its own period and duty cycle without effecting the operations of neighbouring PWM channels. A channel must be enabled by the programmer in order to generate a modulated signal.

DERIVING THE PULSE WIDTH MODULATED SIGNAL

The pulse width modulated signal, produced by a PWM channel, is derived from the HCS12's bus clock. This is accomplished by programming a series of PWM registers with values that scale the bus clock down to the desired pulse width modulated frequency.

WARNING:
Altering between run and boot mode will alter the output frequency of your PWM. The HCS12, due to PLL and HCS12s that support the serial monitor, operates under one of two bus clock speeds determined by the HCS12's operating mode (run mode or load* mode). **Since the PWM frequency is derived from the bus clock you must consider the mode you are operating in.** For example, the 9S12C32 has a run mode bus clock of 4MHz and a boot mode bus clock of 24 MHz.

* On the Technological Arts 32 pin dip module, load mode is also known as boot mode. The term load mode is used for all other HCS12s supporting the serial monitor.

STEPS FOR PROGRAMMING THE PWM REGISTERS (8bit)

Each PWM channel has four programmable PWM registers used to reduce the bus clock to the desired pulse width modulated frequency. These registers are:

- PeriodValue
- DutyCycleValue
- PrescaleA
- ScaleA

The programmer selects the values for these registers. There is no set formula for determining these values.

In the four step algorithm below, each **formula** (except for the duty cycle) is to help you keep track of how close you have reduced the bus clock frequency to your desired pulse width modulated frequency.

STEP 1 – SET THE 'PeriodValue' REGISTER

PeriodValue: scales the bus clock to a smaller size.
PeriodValue: must be <= 256
FORMULA : Bus clock / (2* PeriodValue)

STEP 2 – SET THE 'DutyCycleValue' REGISTER

DutyCycleValue sets the duty cycle of the TTL signal
DutyCycleValue must be <=256
Formula: Duty Cycle = **(DutyCycleValue/ PeriodValue) x100%**

NOTE: Selecting Period and Duty Cycle Values

Since: Duty cycle = (DutyCycleValue / Period Value) x 100%

the larger the value of PeriodValue selected the larger and more precise the range of duty cycle values we can represent.

For example, assume we have Period Value =100. DutyCycle Values of 1, 2 . .100 correspond to the Duty Cycles of 1%, 2%. . . 100%.

Conversely, assume we have Period Value = 25. A 50% duty cycle, for example, cannot be precisely obtained as Duty Cycle value =12.5. Decimal values can not be sent the to the PWM's registers. The closest value we can use is either 12 or 13 which does not yield a 50% duty cycle.

STEP 3 - SET the 'PreScaleA' REGISTER

PreScaleA: further scales the value of bus clock after being reduced by the **PeriodValue**
PreScaleA: The 3 bit combinations in table 2 show the value the Bus clock (referred to as Clock A) is scaled by
Formula: **(Scaled Bus Clock)/ (2^n)** where n<=8 and 'Scaled Bus Clock' = Bus clock / PeriodValue

PCKA2	PCKA1	PCKA0	Value of Clock A
0	0	0	bus clock
0	0	1	bus clock / 2
0	1	0	bus clock / 4
0	1	1	bus clock / 8
1	0	0	bus clock / 16
1	0	1	bus clock / 32
1	1	0	bus clock / 64
1	1	1	bus clock / 128

TABLE 2

STEP 4- SET THE 'ScaleA' REGISTER

ScaleA: Further scales the clock after scaled by PeriodValue and PreScaleA
ScaleA: must be <=256
Forumla: (Further Scaled Bus Clock) /ScaleA where Further Scaled Bus Clock = Bus Clock / PeriodValue / PrescaleA

IN SUMMERIZING THE REGISTER VALUES:

Desired PWM Frequency = Bus Clock / PeriodValue / PrescaleA / ScaleA

The example problem on the next page will help solidify your understanding of determining the register values.

EXAMPLE

PROBLEM:

Determine the PWM register values:

- PeriodValue (used the reduce the clock bus)
- DutyCycleValue
- PrescaleA
- ScaleA

required to generate pulse width modulated signal with 10Hz and a duty cycle of 50%. The 9S12C32 is used and will operate in run mode.

SOLUTION:

Since we are using the 9S12C32 in run mode, the bus clock = 4MHz. We have to reduce 4MHz to 10Hz by programming the PWM registers with the appropriate values.

STEP 1: REGISTER→ PERIODVALUE

Let **PeriodValue=250**

Bus clock / (2 x Period Value) = 4MHz/(2x **250)** = 16kHZ

250 was selected to give us a nice round figure which can be further reduced by the Scale A Value and the PrescaleA

STEP 2: REGISTER → DUTYCYCLEVALUE

Let **DutyCycleValue=125**

Duty Cycle = (DutyCycleValue/ PeriodValue) x100%
Duty Cycle=(125/250) *100% = 50%

STEP 3: REGISTER → PRESCALE A

Let **PrescaleA = 4**

NOTE: 4 is the value submitted to the register. 2^4=16 is the factor that the bus clock is reduced by

(Bus clock / PeriodValue) / (2^n)
(16kHz)/ (16) = 1kHz

1Khz can be easily reduced to 10 Hz by dividing by 100.

STEP 4: REGISTER → SCALED A

Scaled A Value = 100
1kHz / Scale A Value =10 Hz

ONE NOTE BEFORE SHOWING YOU THE CODE . . .

For simplicity we determined the register values in the order of

- PeriodValue (used the reduce the clock bus)
- DutyCycleValue
- PrescaleA
- ScaleA

However, when we program the HCS12 we reverse the order. Don't worry the values for each register do not change, only the order they are programmed, in essence:

- ScaleA
- PrescaleA
- PeriodValue (used the reduce the clock bus)
- DutyCycleValue

Changing PWM signal at runtime
You can change the values of the DutyCycleValue Register and the PeriodValue Register during runtime. Each PWM channel has its own counter. Changes to the period and duty cycle do not take effect until the counter resets back to 0. Thus there is no signal overlap when the signal transitions from one frequency to another.

And with out further a due here is the embedded C code for our 10Hz 50% duty cycle PWM signal!

PROGRAMMING THE MICROCONTROLLER

The Embedded C code that programs the HCS12 PWM to generate the 10Hz 50% signal derived on the previous page is listed below. In CodeWarrior, create a new project called PWM. Open the main.c from the source window. Delete the code that is in there. Copy and paste the following code:

```
//------------------------------------ START OF CODE --------------------------------
#include <hidef.h>      /* common defines and macros */
#include <mc9s12c32.h>     /* derivative information */

#pragma LINK_INFO DERIVATIVE "mc9s12c32"
#pragma CODE_SEG DEFAULT
void main(void)
{


          //Enabling portT for PWM0
           MODRR =0x01;       /*
                                 The  line MODRR=0x01 may not be
                                 necessary if your PWM
                                 in not multiplexed with the Timer module.
                                 Check the block diagram of your circuit.
                              */
           PWMCLK=0x01;
           PWMSCLA=0x32;
           PWMPRCLK=0x04;
           PWMPER0=0xF9;
           PWMDTY0=0x7B;
           PWME=0x01;

while(1){

          }
}

//----------------------------------- END OF CODE --------------------------------------
```

<u>MAKE THE SOURCE CODE → DOWNLOAD THE CODE → TEST PWM</u>

Click MAKE on CodeWarrior, set your HCS12 module into boot mode, hit reset and then download your program to the HCS12. Once the program has downloaded, switch the module into run mode and then select reset again.

If you have done everything correctly the user LED should be flashing!

NOTE: If you do not have a user LED connected to PWM0 (PT0) you can use the circuit in figure 5 and connect it to PT0 using a jumper wire. Use the datasheet for your HCS12 board to identify the pin number corresponding to PT0.

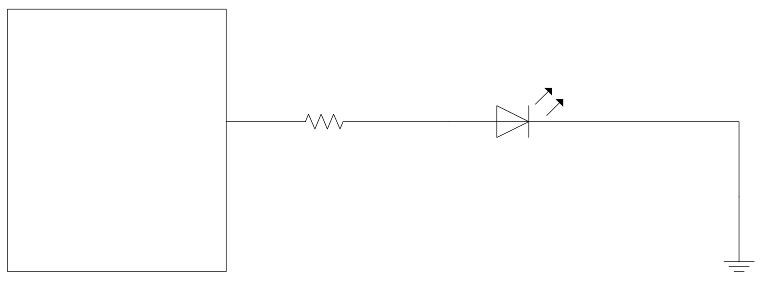

FIGURE 4

Don't worry if you don't understand the source code on the previous page. On the next page we will analyze it.

ANALYZING THE SOURCE CODE

In essence, the purpose of the code is to initialize each of the appropriate PWM registers to produce the 10Hz 50% duty cycle. This section briefly describes each register accessed in the source code.

NOTE: each bit in the MODRR, PWMCLK and PWME register corresponds to the PWM channel. For example:

PWME=0x01 Enables **PWM0** (PWM channel 0), by setting **bit 0 ,** and **PWMCLK=0x01** sets PWM0 to use the ScaleA clock

NOTE: this section refers to the registers on the 9S12C32 derivative. Consult Motorola's documentation - S12PWMxxx.pdf

REGISTER DESCRIPTIONS

MODRR =0x01;
This line may not be necessary if your PWM **is not** multiplexed with the Timer module. Check the block diagram of your HCS12. A set bit means that the corresponding PWM channel will be multiplexed to corresponding bit onPort T. A cleared bit means the corresponding timer channel will be multiplexed to the corresponding bit on Port T
In this line PWM0 is multiplex to Port T (PT0)

PWMCLK=0x01;
BIT 0 ie- PWM0 is set meaning that the Scaled A clock will be use
If BIT 0 were cleared, Clock A would be modulated.

PWMSCLA=0x32;
Sets the Scale A the Hex equivalent of 100

PWMPRCLK=0x04;
Sets the Prescale A to the Hex equivalent of 4.

PWMPER0=0xF9;
Sets the PeriodValue to the Hex equivalent of 250.

PWMDTY0=0x7B;
Sets the DutyCycleValue to the Hex equivalent of 125

PWME=0x01;
Enables PWM0. We enable the PWM channels last. The moment you enable a PWM channel, it starts outputting a signal with the values contained in the registers. We want to make sure that our correct register values are loaded before the PWM is enabled.

16 BIT PERIODVALUE AND DUTYCYCLEVALUE REGISTERS

The PeriodValue and the DutyCycleValue registers can each extend their value range up to 65536 to allow more precise values for duty cycle. This is done by concatenating two PWM channels, forming a 16bit PeriodValue Register and a 16bit DutyCycleValue.

When concatenating two PWM channels, either PWM0 and PWM1 or PWM2 and PWM3 the higher PWM channel will produce the output signal.

Refer to the Motorola documentation - S12PWMxxx.pdf for more information on concatenating under the register description chapter.

SUMMARY

- The pulse width modulator is an HCS12 device which can produce a TTL signal who's frequency and duty cycle can be programmed
- PWMs can be found in most HCS12s.
- Depending on the derivative, Most HCS12s have 1 or more PWMs known has PWM channels.
- Each PWM channel has four programmable registers programmed to derive a desired frequency and duty cycle from the bus clock
- The method for determining the values of these registers are describe in this guide
- IN SUMMERIZING THE REGISTER VALUES:
  Desired PWM Frequency = Bus Clock / PeriodValue / PrescaleA / ScaleA
- The duty cycle and frequency can be programmed to changed during runtime
- Programming the PWM in embedded C, and a description of the registers programmed, are described in this guide

HCS12 RESOURCES

Motorola Documentation

Motorola documentation for your HCS12 derivative can be found on the Technological Arts resource CD or on the web at www.technologicalarts.com

PWM Register Names, and PWM block diagrams can be found in **Motorola documentation - S12PWMxxx.pdf**