ECE263

Lab 8

Fall 2011

Enhanced Capture Timer Module
ECT

(Engine Controller)

Revised: 1 Nov 2011 DAL

(intentionally left blank)

Object:

In this lab you will be using the input capture and output compare functions of the Enhanced Capture Timer (ECT) module of the MC9S12DG256 to perform some simple timing functions. You will use the logic analyzer to verify the correct operation of your programs.

Material:

CodeWarrior software package
Dragon 12 Plus Development Board
P&E USB Multilink
CPU12 Reference Guide (CPU12RG)
Agilent 16801A Logic Analyzer
One flying lead logic analyzer probes

Background

The ECT contains eight channels each of which can be configured to perform either the input capture function or the output compare function and all eight channel share the same 16-bit free running counter. In order to conserve pins, the ECT shares the I/O pins of the MC9S12DG256 with parallel Port T.

When configured for the output compare function, the I/O pin associated with a channel is set to be an output and can be commanded to go to a high state, a low state or to toggle (i.e. take on the opposite state from the current state) at some time in the future when the contents of the free running counter match the value contained in the channel register. In this mode the channel acts like the alarm function on an alarm clock: you select the time that you want the alarm to sound (channel register) and then when the real time (free running counter) counts up to reach the value contained in the alarm register, the buzzer goes off (pin takes on commanded state). The ECT has an 8-bit status register which contains one bit for each channel. When the output event occurs on the pin, the status bit for the channel is set true so the program can determine that the time period has elapsed and the output pin has changed state. The channel can be programmed for only one event at a time which is the next event that will occur at some time in the future. If you need to generate some arbitrary waveform that consists of a series of transitions at various times (frequencies) then you program the events in series: program the channel for the first event, wait for it to occur by monitoring the status bit, when it occurs, reset the status bit and then program for the next event.

When configured for the input capture function, the I/O pin associated with the channel is set to be an input and when it detects a specific transition, either high state to low state, low state to high state or toggle (i.e. a change from either state to the opposite state), the channel register is loaded with the contents of the free running counter and the status bit for the channel is set. In this mode, the channel acts like a stopwatch: the watch is counting up in time and when the user pushes the stop button (when the commanded state is detected), the current time is captured on the display (the free running counter is loaded into the channel register). Like the output compare function, the input compare function can be programmed for only one operation at a time. If you need to detect a series of transitions, they must be programmed one at a time: program the transition desired, wait for it to occur by monitor the status bit, when it becomes set reset it, then read the channel register and save it in memory, and finally program for the next event.

Projects that require complex timing tasks can combine the operations of multiple ECT channels by programming various combinations of functions. We will examine the design of a circuit that will control the firing of the spark plugs of a 4 cylinder internal combustion engine where each spark plug must be fired a specific number of degrees of rotation of the engine after top dead center (TDC) as determined by a sensor on the crankshaft. First you need to determine the speed of rotation of the crankshaft by measuring the time between two transitions of the TDC sensor; this measurement gives the time required for 360 degrees of rotation. Simple calculations will then determine the amount of time after the detection of TDC that each individual plug must be fired ($T_{FIRE} = T_{ROTATION}$ x #_degrees/ 360_degrees). To fire a spark plug you would use the time captured by the TDC sensor channel, add the calculated offset and then set the output channel to fire at the right time. To control the amount of time the signal must be present to fire the spark plug, another transition on the output channel would be programmed to turn off the output. To implement the design, one channel would be programmed as an input capture and it would capture the contents of the free running counter every time the TDC sensor was detected. By subtracting the previous time, the time for one rotation would be determined. Four other channels would be programmed for output compare functions and would do their calculations for determining when to fire using the time captured by the input compare function.

Preparation:

For this lab you will write a program to perform the operation of a simplified version of the engine controller described above. Timer channel 0 (TC0) on Port T bit 0 (PT0) will be configured for the input capture operation and it will be connected to the output of the function generator which will simulate the crankshaft sensor. Timer channel 1 (TC1, on PT1) and timer channel 2 (TC2, on PT2) will be configured for the output compare operation and will be used to simulate the signals to fire two spark plugs. The time between transitions of the signal on the sensor channel (TC0) will represent one full revolution of 360^{O} and each spark plug channel (TC1 and TC2) will pulse high for 10 usec at 90^{O} and 180^{O} respectively after the transitions on TC0.

Your program will measure the time between two transitions of the sensor signal and then you will use this number to generate the delta time for 90^{O} and 180^{O}. After the second transition of TC0 you will use the two calculated delta times to program the time for the low to high transitions of the high going pulses on TC1 and TC2. After TC1 and TC2 have gone high you need to set them to go back to a low after a delay of 10 usec. Once you start this process it will continue forever. That is to say, the second transition of TC0 represents the end of the first cycle and the start of the second cycle. The two time delays that you calculated after the end of first cycle will be used to fire the spark plugs in the second cycle. At the end of the second cycle (third transition of TC0) you will again calculate two delta times which will be used during the third cycle.

You will test the operation of the program by varying the frequency of the function generator between 500Hz to 5KHz and observing all three of the timer channels with the logic analyzer.

Note: use status polling to service the timer channels.

Before you come to the lab, do the following:

1. Draw a timing diagram showing the operation of PT0, PT1 and PT2 for the program.
2. Draw a flow chart of the program.
3. Generate the assembly code for the program 1.

Generate the code BEFORE coming to lab. I guarantee that you will not have time to design the program, write the code, compile it and debug it all in the 1.5 hours allocated for the lab.

At the start of the lab, turn in a copy of the flow chart and the assembly code that will be your starting point.

Remember, the pre-lab is to be the combined effort of both (all) lab partners.

Procedure: Working with your lab partner(s), complete the following steps:

Using the procedures you learned in the previous labs,

1. Create a new project and enter your program as *main.asm*. Generate the executable object file using the *Make* facility and then download the program on the Dragon 12 Plus.
2. Set up the logic analyzer to monitor PT0, PT1 and PT2.
3. Set the function generator as follows:
   a. Set the frequency to approximately 1KHz
   b. in the *Utilities* window, and set SYNC ON
4. Attach the SYNC output of the function generator to PT0 on the Dragon 12 Plus board.
5. Start testing the program by triggering the analyzer on LO to HI transitions of PT0
6. Verify that your program can generate the waveform described in the write-up.
7. Set the frequency of the function generator to different values between 500Hz and 5KHz and verify that the program still operates correctly.
8. Print out some logic analyzer traces to include in the lab report to prove that your program worked correctly
9. Demonstrate the operation of the program to the instructor or the TA and get the Lab Verification sheet signed.

Lab Report:

The lab report is to contain the following sections: (see Lab Report Guideline)

1. Title sheet
2. Objective
3. Material List
4. Procedure
5. Experiment data
    a. Copy of the timing diagram you generated for Step 1 of the prelab
    b. Logic Analyzer traces from Step 8 with explanation of the data being displayed.
    c. Explanation of any problems and/or errors and how you would have corrected them.
6. What you need to know section including a brief explanation of your program and calculations
7. Appendix
    a. Lab Verification Sheet signed by instructor or TA
    b. Copy of the updated flow charts for the program
    c. Copy of final working the program
    d. Copy of the Pre-lab

NOTE: DO NOT INCLUDE A COPY OF THIS LAB WRITE-UP IN THE LAB REPORT