Code Measurements

Purpose

You will measure the performance of software during execution.

Setup

Copy the Code Measurements archive into your directory. Unzip it and open it.

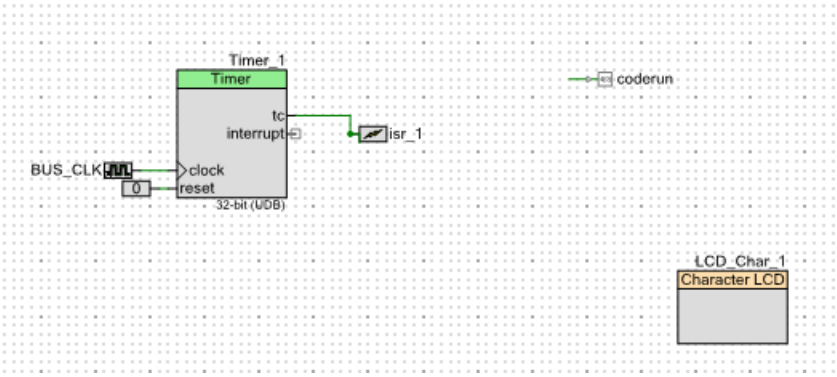
Procedure

Our goal is to measure the execution time of a block of code as accurately as possible. Look at main.c. The measurement code is repeatedly executed in a *for(;;)* loop. The code to be measured is marked off with comments; we execute the code several times (based on the value of *startmax*) to give a total execution time that is long enough to be accurately measured by the timer.

Q1: Draw a UML state diagram for the operation of the execution loop.

We will measure code performance using both internal and external techniques: we will measure execution time using an internal timer; we will also bring a signal to the pins for software execution that can be observed on an oscilloscope.

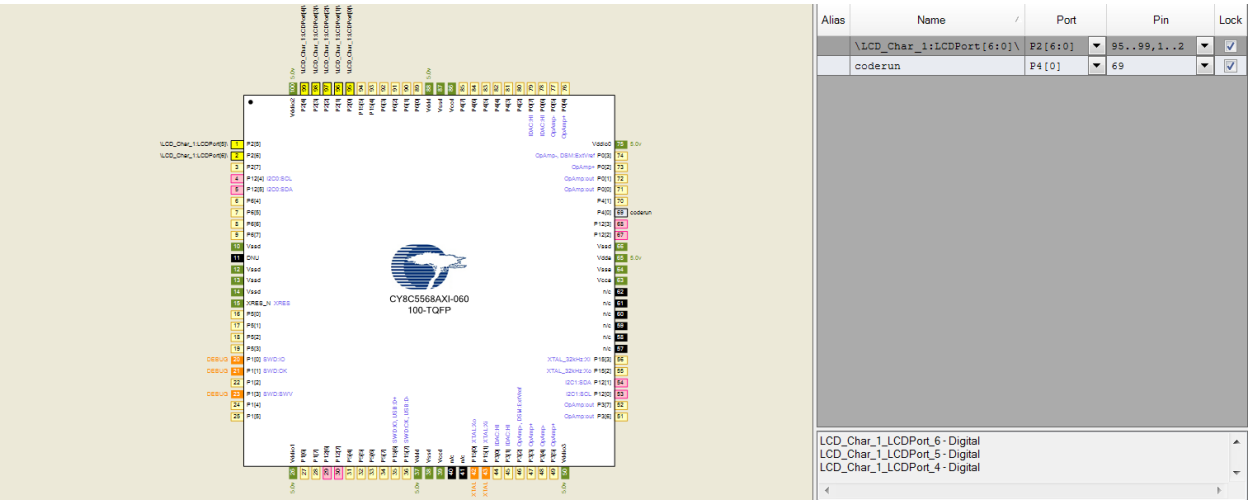
Look at the .cysch file:



The design includes several elements:

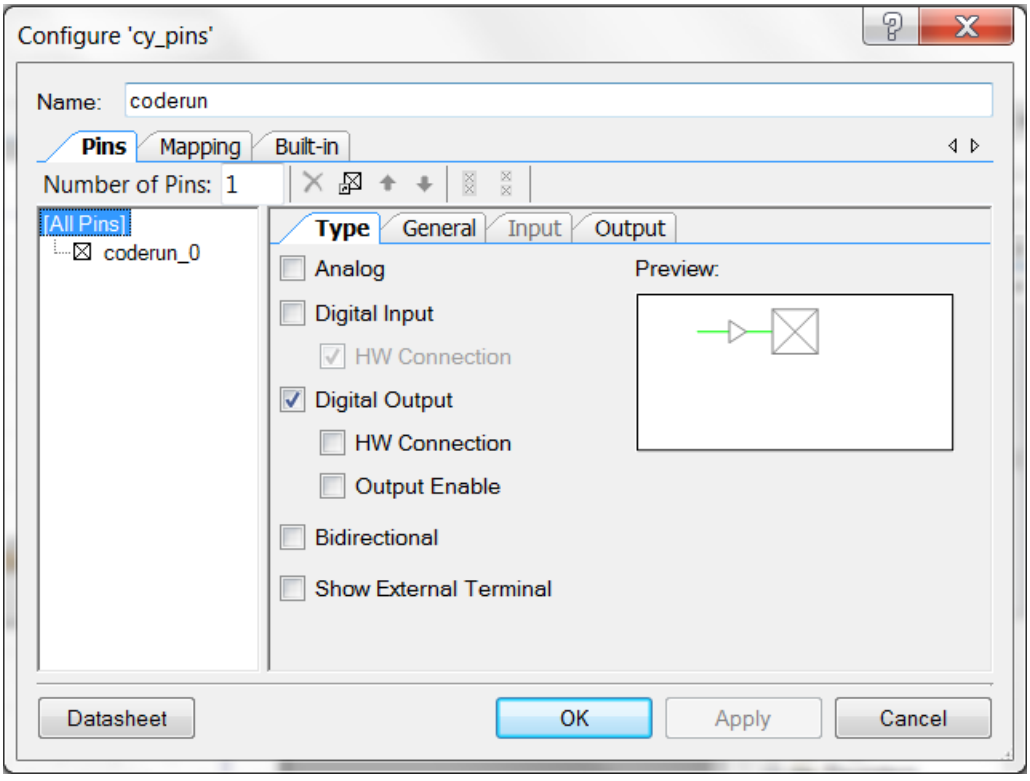
- The timer will measure the execution time of the code. It is driven by an internal clock. The timer's output is connected to an interrupt handler that simply resets the counter.
- The coderun pin is set to 1 when the code is running.
- The LCD is used to write the results of the counter.

Now look at the .cydwr file:



The left-hand side of the diagram shows the arrangement of pins around the edge of the chip. The right-hand side allows you to declare an internal signal as connected to a pin and to assign that signal to a particular pin.

Now go back to the .cysch view and double-click on the *coderun* pin to view its configuration:



Notice that it is configured as a digital output.

Compile and download the code.

Q2: What is the execution time of the code as shown on the LCD?

Q3: What is the execution time of the code as measured by an oscilloscope?

You Should Turn In

1. Answers to Q1, Q2, Q3.

As always, output snapshots are cool.

Super-duper bonus points: use the potentiometer to change the value of NDELAY on-the-fly.

