Part b) Write the C language device driver for the DAC interface. Include at least two functions that implement the DAC interface. For example, you could implement the function **DAC_Init()** to initialize the DAC, and the function **DAC_Out** to send a new data value to the DAC. Place all code that accesses the DAC in the **DAC.c** code file. The **DAC.h** header file contains the prototypes for public functions. Describe how to use a module in the comments of the header file. Describe how the module works, how to test the module, and how to change the module in the comments of the code file.

Part c) Begin with the static testing of the DAC. You should write a simple main program to test the DAC, similar in style as Program 13.2. You are free to debug this system however you wish, but you must debug the DAC module separately. You should initially debug your software in the simulator (Figure 13.2). You can single step this program, comparing digital data to the analog voltage at the DACOUT without the speaker attached (i.e., left side of Figure 13.1). Figure 13.3 shows the static testing using program 13.2 running in the simulator. When testing in the simulator, you can skip the call to the **delayms** function. However, when testing on the real board, this delay is needed to make the DACOUT wave slower than the 10 kHz sampling of the TExaS scope. You can also connect the DACOUT, through the resistor, to your headphones and hear what a sawtooth wave sounds like.

```
#include "DAC.h"
// Inputs: Number of msec to delay
// Outputs: None
void delayms(unsigned long msec){
  unsigned long count;
  while(msec > 0 ) {  // repeat while there are still delay
    count = 16000;    // about 1ms
    while (count > 0) {
      count--;  // each loop takes 5 cycles in simulation
    }
    msec--;
  }
}
int main(void){ // this main is to debug the DAC
      // you must connect PD3 to your DAC output
  TExaS_Init(SW_PIN_PE3210, DAC_PIN_PB3210,ScopeOn);
  DAC_Init(); // initialize SysTick timer and DAC
  EnableInterrupts();  // enable after all initialization are done
```

```
    while(1){unsigned long i; // static debugging
      for(i=0;i<16;i++){
        DAC_Out(i);
        delayms(10); // connect PD3 to DAC output
      }
    }
}
```

Program 13.2. A simple program that outputs all DAC values in sequence.

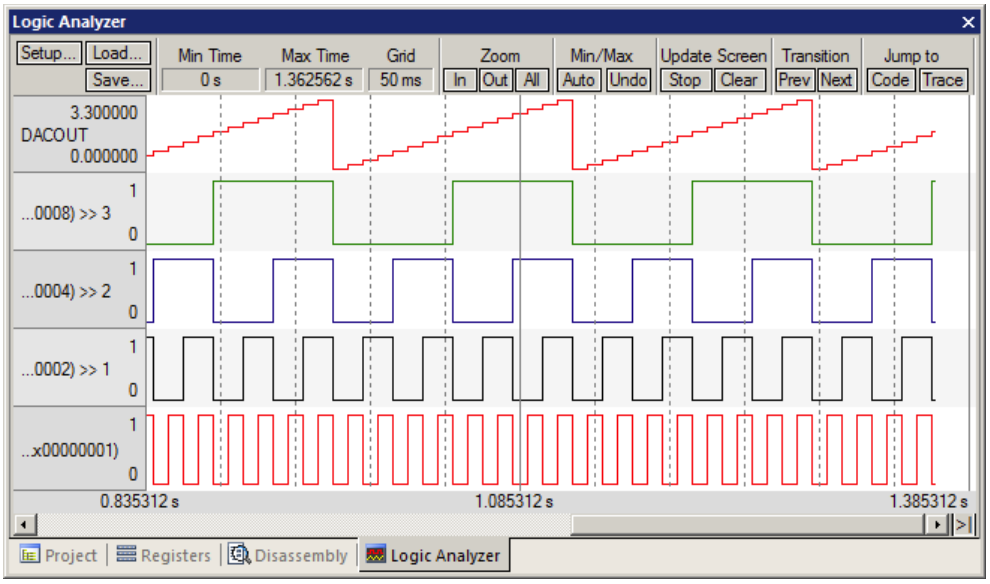

Figure 13.3. A screenshot in simulation mode showing the static testing of the DAC.