


# Debugging Techniques

- ▶ Debugging embedded HW/SW systems differs greatly from debugging software or hardware alone.
- ▶ Key steps to debugging:<sup>1</sup>
  - ▶ Make the bug repeatable
  - ▶ Observe its behavior and gather information on it
  - ▶ Create a suspect list and generate a hypothesis
  - ▶ Try your fix, unfix it, then re-fix
- ▶ Prerequisites to debugging:
  - ▶ Observability (o-scope, meter, blink an LED, print statement)
  - ▶ Controlability (swap circuitry, push a button, initiate the fault)
  - ▶ Keep notes on what you are doing

---

<sup>1</sup>adapted from Jack Gansel's "The Art of Designing Embedded Systems" 

# Debugging Techniques

- ▶ Much of debugging is about asking the right questions
  - ▶ What changed last?
    - ▶ New cable, new hardware, re-soldered a wire, new code
  - ▶ Is this a problem or a symptom?
    - ▶ A part getting hot doesn't need more cooling
  - ▶ Is the power on? Is the input voltage correct?
    - ▶ Five volts from the USB port is never 5 volts
  - ▶ Are you compiling/editing the right file?
  - ▶ Is the hardware basically OK? Run a sanity test.
  - ▶ What assumption are you making that is wrong?
  - ▶ Explain the problem to someone else.

# Debugging Techniques

- ▶ Debugging requires a correct attitude
  - ▶ Assume that things are somewhat broken
  - ▶ Assume that a few small bugs may still exist
  - ▶ Trust the laws of physics. Your project depends on them.
  - ▶ You will witness something that cannot be happening; so why is it happening? Retrace assumptions.
  - ▶ Take a break. Walk around, get a coffee. Take a nap.

# Debugging Techniques

- ▶ Debugging requires discipline
  - ▶ Resist unlikely sources of bugs:
    - ▶ bad parts, compiler error, phase of moon
  - ▶ Stay focused on one problem at a time
  - ▶ Keep to the scientific method
  - ▶ When you find something you can't explain, write it down.
  - ▶ Resist random o-scope probing
  - ▶ Don't take shots in the dark

# Debugging Techniques

- ▶ It helps to avoid bugs in the first place
  - ▶ Build a little hardware, write a little software, test
  - ▶ Write throwaway code for targeted hardware testing

# Debugging Techniques

- ▶ Perform a sanity test to find programming problems, power supply issues, shorted lines, missing libraries and 101 other silly things.

```
//  
// count.c, Binary counting pattern on PORTB LEDs.  
//  
#include <avr/io.h>  
#include <util/delay.h>  
  
int main() {  
    uint8_t i;  
    DDRB = 0xFF; //set port B to all outputs  
    while(1){  
        PORTB++;  
        for(i=0; i<=4; i++){_delay_ms(100);} //0.5 sec delay  
    }//while  
}//main
```

# Debugging Techniques

- ▶ Debugging the LED display board
- ▶ See if segments are connected, and digits are enabled correctly.
- ▶ Port D push-buttons 4-7 select the digit via the decoder. All segments should illuminate on each digit.

```
//LED board testing code
int main(){
    DDRA  = 0xFF;    //set port A to all outputs
    DDRB  = 0xF0;    //set port bits 4-7 B as outputs
    DDRD  = 0x00;    //set port D all inputs
    PORTD = 0xFF;    //set port D all pullups
    PORTA = 0x00;    //set port A to all zeros (all segments on)
    while(1){PORTB = ~PIND;} //push buttons selects digit
} //main
```

# Debugging Techniques

- ▶ More debugging of the LED display board
- ▶ Walk through each digit and each segment.

```
//LED board testing code
int main(){
    uint16_t i;    //big delay counter
    uint8_t k=0;  //digit counter
    while(1){
        k = (k % 5);        //digits vary from 0 to 4
        PORTB = (k << 4);  //walk the digits via upper nibble
        PORTA = ~0x01;    for (i=0; i<100000; i++){ //A
        PORTA = ~0x02;    for (i=0; i<100000; i++){ //B
        PORTA = ~0x04;    for (i=0; i<100000; i++){ //C
        PORTA = ~0x08;    for (i=0; i<100000; i++){ //D
        PORTA = ~0x10;    for (i=0; i<100000; i++){ //E
        PORTA = ~0x20;    for (i=0; i<100000; i++){ //F
        PORTA = ~0x40;    for (i=0; i<100000; i++){ //G
        PORTA = ~0x80;    for (i=0; i<100000; i++){ //colon
        k++; //increment digit counter
    } //while
```



# Debugging Techniques

- ▶ Debug scenarios, what to do when...
  - ▶ One of the digits are not displaying.
  - ▶ Two digits display at simultaneously
  - ▶ None of the the digits light up.
  - ▶ Segments exhibit *ghosting* when your hand is close to the board.
  - ▶ The entire display works one time through then quits.

# Debugging Techniques

- ▶ Bugs that mysteriously go away will mysteriously reappear
- ▶ "...reason back from the state of the crashed program to determine what could have caused this. Debugging involves backwards reasoning, like solving murder mysteries. Something impossible occurred, and the only solid information is that it really did occur. So we must think backwards from the result to discover the reasons."  
- Brian W. Kernighan, *The Practice of Programming*
- ▶ "The single greatest asset a designer can have is self-knowledge. Knowing when your thinking feels right and when you're trying to fool yourself...knowing your strengths and weaknesses, prowesses and prejudices...learning when to ask questions and when to believe your answers."  
- Jim Williams, Linear Technology
- ▶ "When things are acting funny, measure the amount of funny."  
- Bob Pease, National Semiconductor