CS122A: Intermediate Embedded and Real Time Operating Systems

Jeffrey McDaniel

University of California, Riverside

Debugging

- Bugs in your code are inevitable

Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually

Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually
- "More than a third of these costs ... could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects." NIST

Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually
- "More than a third of these costs ... could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects." NIST
- Available tools and techniques for debugging:

Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually
- "More than a third of these costs ... could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects." NIST
- Available tools and techniques for debugging:
  - Simulators

Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually
- "More than a third of these costs ... could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects." NIST
- Available tools and techniques for debugging:
  - Simulators
  - Output (LCD, LED, and Pin Debugging)

Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually
- "More than a third of these costs ... could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects." NIST
- Available tools and techniques for debugging:
  - Simulators
  - Output (LCD, LED, and Pin Debugging)
  - UART

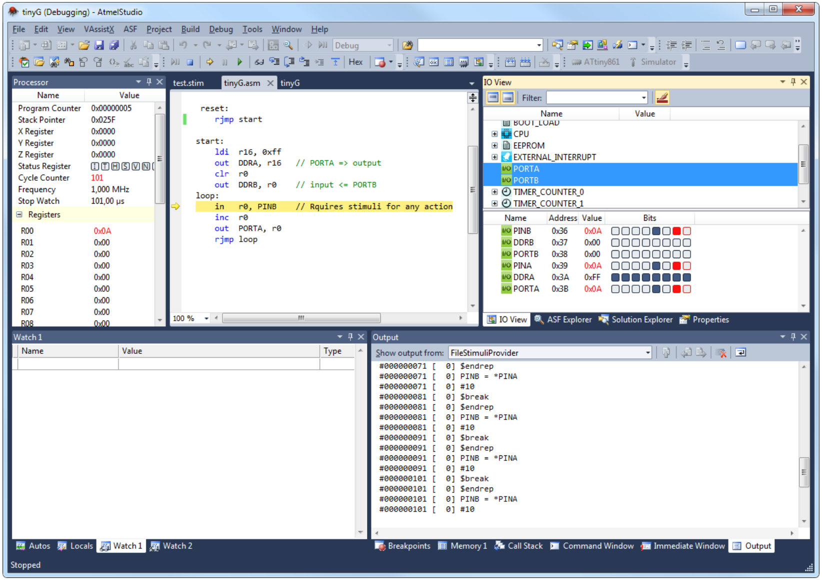
Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually
- "More than a third of these costs ... could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects." NIST
- Available tools and techniques for debugging:
  - Simulators
  - Output (LCD, LED, and Pin Debugging)
  - UART
  - Logic Analyzer

Debugging

- Bugs in your code are inevitable
- Software bugs cost the U.S. economy $59.6 billion annually
- "More than a third of these costs ... could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects." NIST
- Available tools and techniques for debugging:
  - Simulators
  - Output (LCD, LED, and Pin Debugging)
  - UART
  - Logic Analyzer
  - On-Chip-Debuggers (OCD)

Simulators


tinyG (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Hex
ATtiny861 Simulator
Processor
test.stim tinyG.asm tinyG
IO View
Filter:
100 %
IO View ASF Explorer Solution Explorer Properties
Watch 1
Output
Autos Locals Watch 1 Watch 2
Breakpoints Memory 1 Call Stack Command Window Immediate Window Output
Stopped

Name	Value
Program Counter	0x00000005
Stack Pointer	0x025F
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N
Cycle Counter	101
Frequency	1,000 MHz
Stop Watch	101,00 μs
Registers	
R00	0x0A
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00

```
 reset:
    rjmp start

start:
    ldi  r16, 0xff
    out  DDRA, r16    // PORTA => output
    clr  r0
    out  DDRB, r0     // input <= PORTB
loop:
    in   r0, PINB     // Rquires stimuli for any action
    inc  r0
    out  PORTA, r0
    rjmp loop
```

Name	Value
BOOT_LOAD	
CPU	
EEPROM	
EXTERNAL_INTERRUPT	
PORTA	
PORTB	
TIMER_COUNTER_0	
TIMER_COUNTER_1	

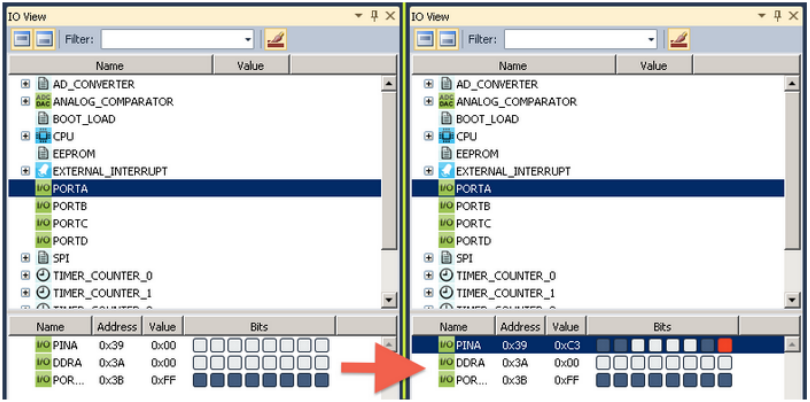
Name	Address	Value	Bits
PINB	0x36	0x0A	☐☐☐☐■☐🟥☐
DDRB	0x37	0x00	☐☐☐☐☐☐☐☐
PORTB	0x38	0x00	☐☐☐☐☐☐☐☐
PINA	0x39	0x0A	☐☐☐☐■☐🟥☐
DDRA	0x3A	0xFF	■■■■■■■■
PORTA	0x3B	0x0A	☐☐☐☐■☐🟥☐

Name	Value	Type

Show output from: FileStimuliProvider

```
#000000071 [  0] $endrep
#000000071 [  0] PINB = *PINA
#000000071 [  0] #10
#000000081 [  0] $break
#000000081 [  0] $endrep
#000000081 [  0] PINB = *PINA
#000000081 [  0] #10
#000000091 [  0] $break
#000000091 [  0] $endrep
#000000091 [  0] PINB = *PINA
#000000091 [  0] #10
#000000101 [  0] $break
#000000101 [  0] $endrep
#000000101 [  0] PINB = *PINA
#000000101 [  0] #10
```

Simulators


IO View

Filter:

Name	Value
AD_CONVERTER	
ANALOG_COMPARATOR	
BOOT_LOAD	
CPU	
EEPROM	
EXTERNAL_INTERRUPT	
PORTA	
PORTB	
PORTC	
PORTD	
SPI	
TIMER_COUNTER_0	
TIMER_COUNTER_1	

Name	Address	Value	Bits
PINA	0x39	0x00	□□□□□□□□
DDRA	0x3A	0x00	□□□□□□□□
POR...	0x3B	0xFF	■■■■■■■■

IO View

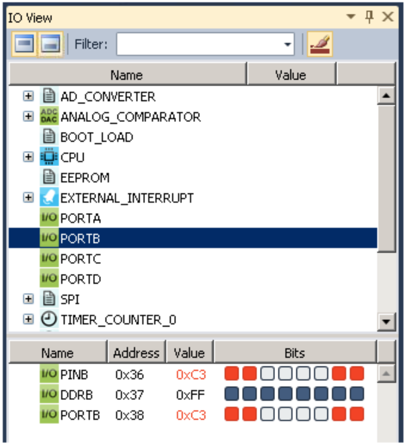
Filter:

Name	Value
AD_CONVERTER	
ANALOG_COMPARATOR	
BOOT_LOAD	
CPU	
EEPROM	
EXTERNAL_INTERRUPT	
PORTA	
PORTB	
PORTC	
PORTD	
SPI	
TIMER_COUNTER_0	
TIMER_COUNTER_1	

Name	Address	Value	Bits
PINA	0x39	0xC3	■■□□□□■🟥
DDRA	0x3A	0x00	□□□□□□□□
POR...	0x3B	0xFF	■■■■■■■■

- Simulators model the internal state of the device
- Great for testing logic and tracking values of variables

Simulators



IO View

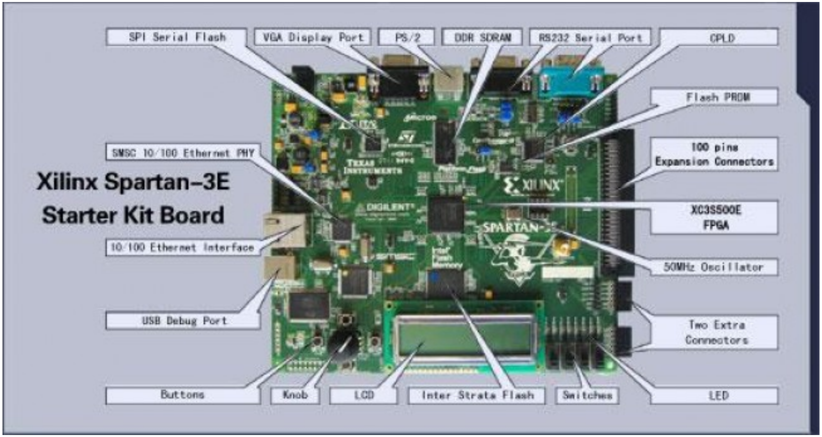
Filter:

Name	Value
AD_CONVERTER	
ANALOG_COMPARATOR	
BOOT_LOAD	
CPU	
EEPROM	
EXTERNAL_INTERRUPT	
PORTA	
PORTB	
PORTC	
PORTD	
SPI	
TIMER_COUNTER_0	

Name	Address	Value	Bits
PINB	0x36	0xC3	🟥🟥□□□□🟥🟥
DDRB	0x37	0xFF	■■■■■■■■
PORTB	0x38	0xC3	🟥🟥□□□□🟥🟥

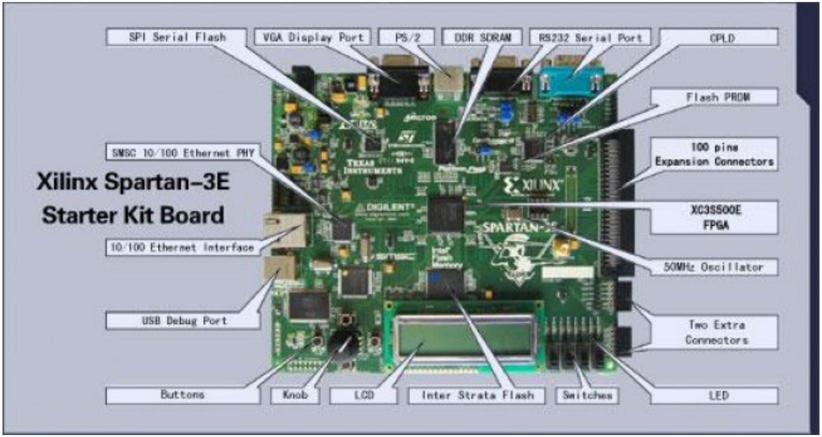
- Simulators model the internal state of the device
- Great for testing logic and tracking values of variables
- Simulator is not running on the hardware and so not all bugs can be caught

Using Output



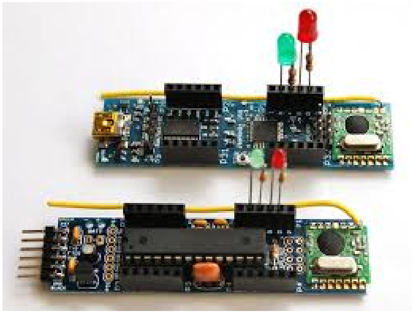
- Simulators are not always able to catch every bug

Using Output



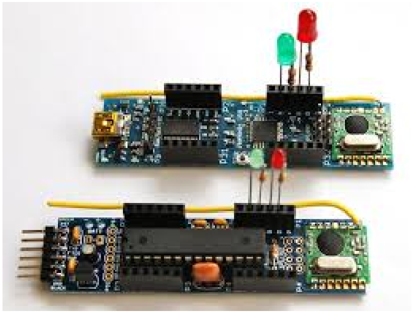
- Simulators are not always able to catch every bug
- Sometimes you need to see what is happening on the hardware itself

Using LEDs



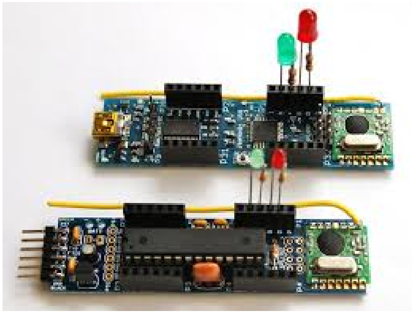
- LED's are the simplest way to debug

Using LEDs



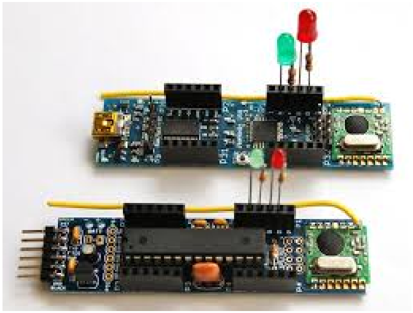
- LED's are the simplest way to debug
- Test to see if a port is getting the output (1/0) that it is supposed to

Using LEDs



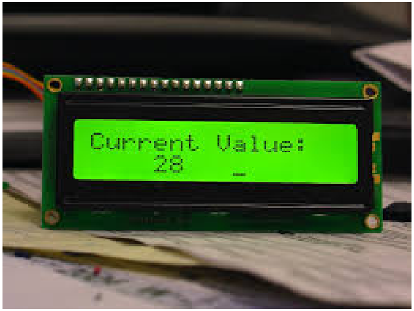
- LED's are the simplest way to debug
- Test to see if a port is getting the output (1/0) that it is supposed to
- Output the binary value of a variable

Using LEDs



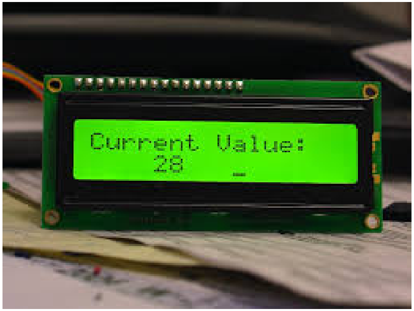
- LED's are the simplest way to debug
- Test to see if a port is getting the output (1/0) that it is supposed to
- Output the binary value of a variable
- Output the binary value of the state that an SM is in

Using an LCD screen



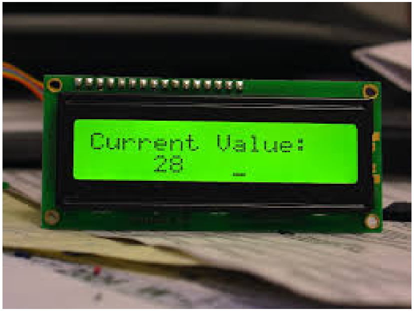
- The LCD screen allows you to display more information

Using an LCD screen



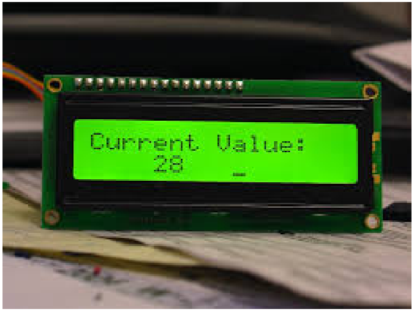
- The LCD screen allows you to display more information
- The integration is slightly more complex however

Using an LCD screen



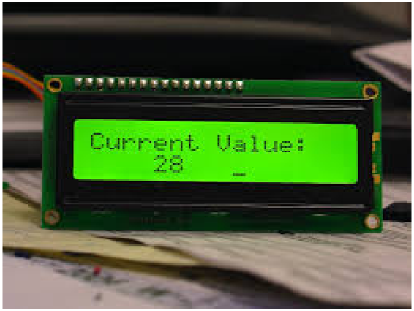
- The LCD screen allows you to display more information
- The integration is slightly more complex however
- Display the value of variables

Using an LCD screen



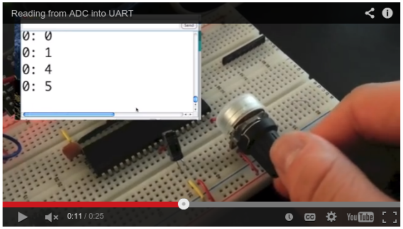
- The LCD screen allows you to display more information
- The integration is slightly more complex however
- Display the value of variables
- Display the state of each state machine

Using an LCD screen



- The LCD screen allows you to display more information
- The integration is slightly more complex however
- Display the value of variables
- Display the state of each state machine
- Create a more complex on chip debug environment

UART Debugging

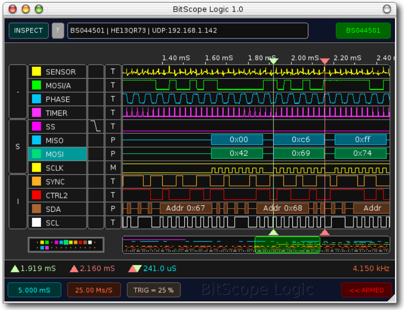


```
0: 0
0: 1
0: 4
0: 5
```

ieee.ucr.edu/parts.cs120b/atmega/

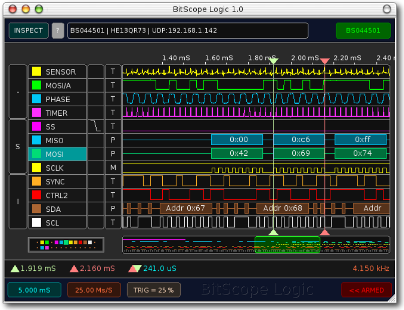
- UART allows you to send messages to your computer to help debug
- More information is able to be displayed this way
- The integration process is more difficult
- If you are already using your UART ports it is more difficult

Logic Analyzers



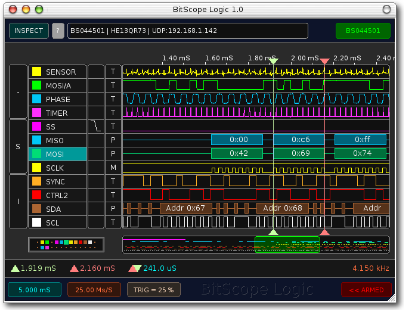
- Captures and displays multiple signals

Logic Analyzers



- Captures and displays multiple signals
- Display uses timing diagrams, SM traces, raw signal, or other formats

Logic Analyzers



- Captures and displays multiple signals
- Display uses timing diagrams, SM traces, raw signal, or other formats
- Useful for seeing exactly what signals are coming for analyzing sensor data

On-Chip-Debugger

- Mechanisms for monitoring and controlling execution on the device

On-Chip-Debugger

- Mechanisms for monitoring and controlling execution on the device
- Application is not being emulated/simulated but actually running on the target hardware

On-Chip-Debugger

- Mechanisms for monitoring and controlling execution on the device
- Application is not being emulated/simulated but actually running on the target hardware
- Not available on all microcontrollers