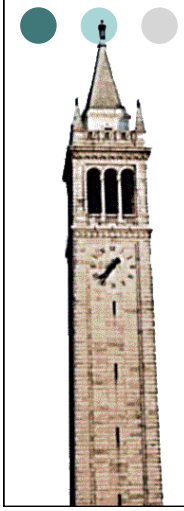


Introduction to Embedded Systems

Edward A. Lee & Sanjit Seshia
With contributions from Isaac Liu and Winthrop Williams
UC Berkeley
EECS 149
Spring 2012

Interfacing to Sensors and Actuators

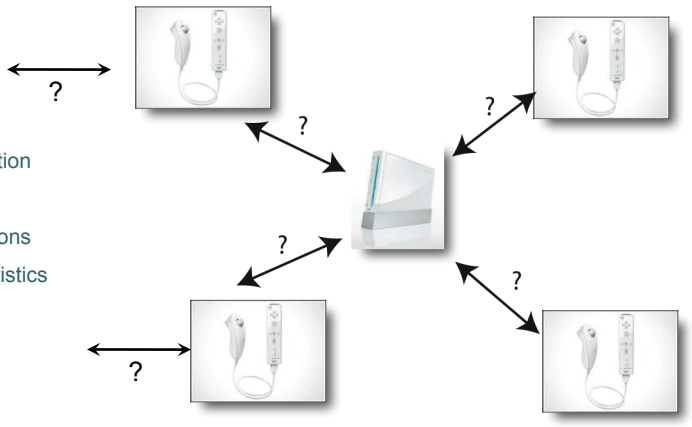
Connecting the Analog and Digital Worlds

Cyber:
- Digital
- Discrete in time
- Sequential

Physical:
- Continuum
- Continuous in time
- Concurrent

Interfaces

- Physical Connection
- Handshake
- Multiple connections
- Timing Characteristics



A Typical Microcomputer Board

This board has analog and digital inputs and outputs. What are they? How do they work?

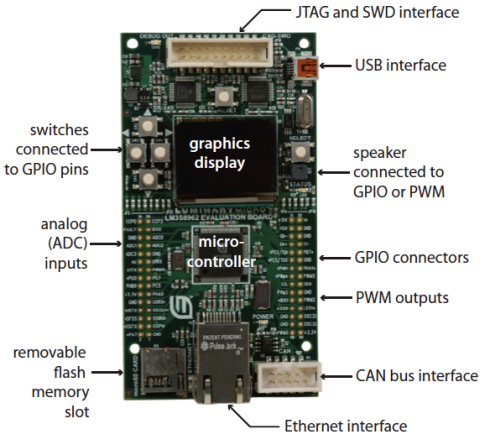
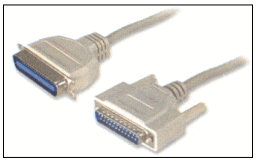


Figure 8.1: Stellaris®LM3S8962 evaluation board (Luminary Micro®, 2008a).

Parallel vs. Serial Digital Interfaces

- Parallel
  - Multiple data lines transmitting data
  - Speed
  - Ex: PCI, ATA, CF cards, Bus



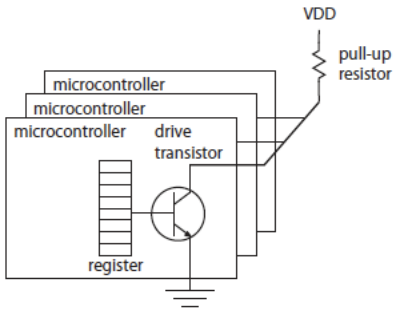
- Serial
  - Single data line transmitting data
  - Low Power, length
  - Ex: USB, SATA, SD cards, PCI-Express

Simple Digital Output: GPIO

Open collector circuits are often used on GPIO (general-purpose I/O) pins of a microcontroller.

The same pin can be used for input and output. And multiple users can connect to the same bus.

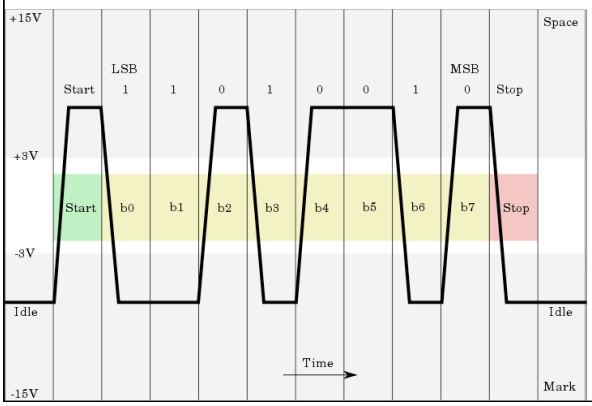
Why is the current limited?



Serial Interfaces

The old but persistent RS-232 standard supports asynchronous serial connections (no common clock).

How does it work?

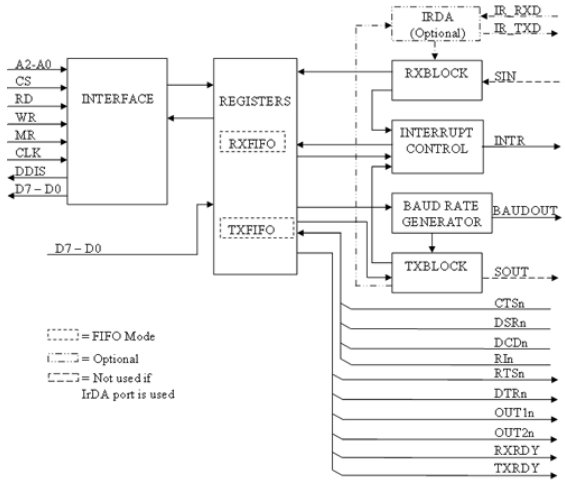


Many but not all uses of RS-232 are being replaced by USB, which is electrical simpler but with a more complex protocol.

Uppercase ASCII "K" character (0x4b) with 1 start bit, 8 data bits, 1 stop bit.
Image license: Creative Commons ShareAlike 1.0 License

UART: Universal Asynchronous Receiver-Transmitter

- Convert serial data to parallel data, and vice versa.
- Uses shift registers to load store data
- Can raise interrupt when data is ready
- Commonly used with RS-232 interface



Variant: USART: Universal Synchronous/Asynchronous Receiver-Transmitter

Example Using a Serial Interface

In an Atmel AVR 8-bit microcontroller, to send a byte over a serial port, the following C code will do:

```
while(!(UCSR0A & 0x20));
UDR0 = x;
```

- x is a variable of type uint8.
- UCSR0A and UDR0 are variables defined in header.
- They refer to memory-mapped registers in the UART.

Send a Sequence of Bytes

```
for(i = 0; i < 8; i++) {
    while(!(UCSR0A & 0x20));
    UDR0 = x[i];
}
```

How long will this take to execute? Assume:

- 57600 baud serial speed.
- 8/57600 =139 microseconds.
- Processor operates at 18 MHz.

Each while loop will consume 2500 cycles.

Receiving via UART

Again, on an Atmel AVR:

```
while(!(UCSR0A & 0x80));
return UDR0;
```

- Wait until the UART has received an incoming byte.
- *The programmer must ensure there will be one!*
- If reading a sequence of bytes, how long will this take?

Under the same assumptions as before, it will take 2500 cycles to receive each byte.

Your Lab Hardware (2012)

Source: National Instruments

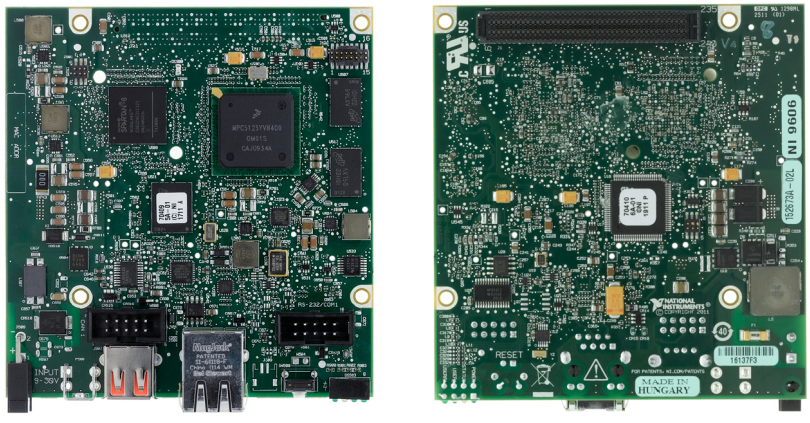
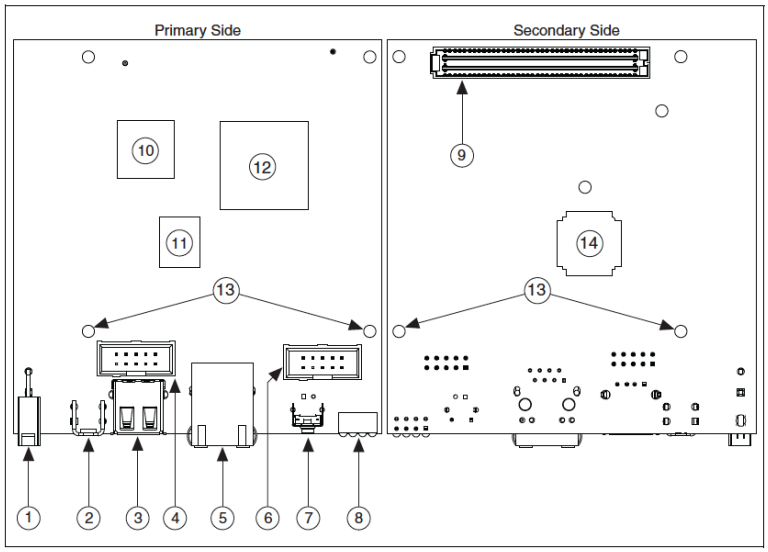


Figure 1. Top and bottom views of sbRIO-9606 featuring a RIO Mezzanine Card connector.

sbRIO

Source:
National
Instruments



1	Power Connector	8	LEDs
2	Chassis Ground Bracket	9	RIO Mezzanine Card Connector
3	USB Host Port (sbRIO-9606 only)	10	FPGA
4	CAN IDC Header (CAN0) (sbRIO-9606 only)	11	NAND Flash
5	RJ-45 Ethernet Port	12	Processor
6	RS-232 Serial IDC Header (COM1)	13	Mounting Holes Connected to Chassis Ground
7	Reset Switch	14	CPLD

Figure 2. NI sbRIO-960x Component Location Diagram

Microblaze I/O Architecture

Source:
Xilinx

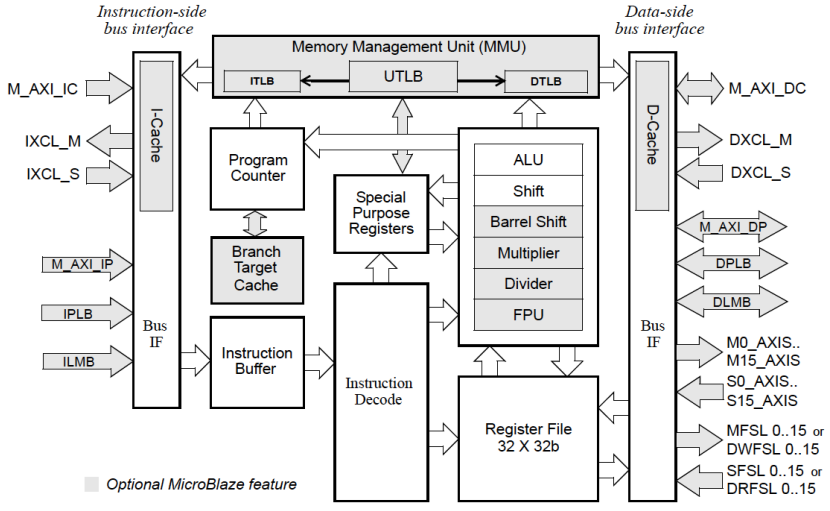


Figure 2-1: **MicroBlaze Core Block Diagram**

Microblaze I/O Architecture

Source:
Xilinx

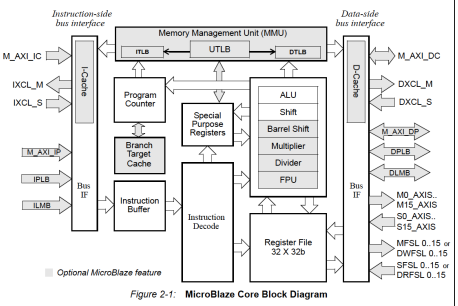


Figure 2-1: **MicroBlaze Core Block Diagram**

M_AXI_DP: Peripheral Data Interface, AXI4-Lite or AXI4 interface
DPLB: Data interface, Processor Local Bus
DLMB: Data interface, Local Memory Bus (BRAM only)
M_AXI_IP: Peripheral Instruction interface, AXI4-Lite interface
IPLB: Instruction interface, Processor Local Bus
ILMB: Instruction interface, Local Memory Bus (BRAM only)
M0_AXIS..M15_AXIS: AXI4-Stream interface master direct connection interfaces
S0_AXIS..S15_AXIS: AXI4-Stream interface slave direct connection interfaces
MFSL 0..15: FSL master interfaces
DWFSL 0..15: FSL master direct connection interfaces
SFSL 0..15: FSL slave interfaces
DRFSL 0..15: FSL slave direct connection interfaces
DXCL: Data side Xilinx CacheLink interface (FSL master/slave pair)
M_AXI_DC: Data side cache AXI4 interface
IXCL: Instruction side Xilinx CacheLink interface (FSL master/slave pair)
M_AXI_IC: Instruction side cache AXI4 interface
Core: Miscellaneous signals for: clock, reset, debug, and trace

Standards

- Serial:
  - Synchronous:
    - SPI, I2C, JTAG, USB
  - Asynchronous:
    - RS232
- Parallel:
  - Bus protocols
    - Advanced Technology Attachment (ATA)
    - Peripheral Component Interface (PCI)
    - …

An I/O View of a Microcontroller

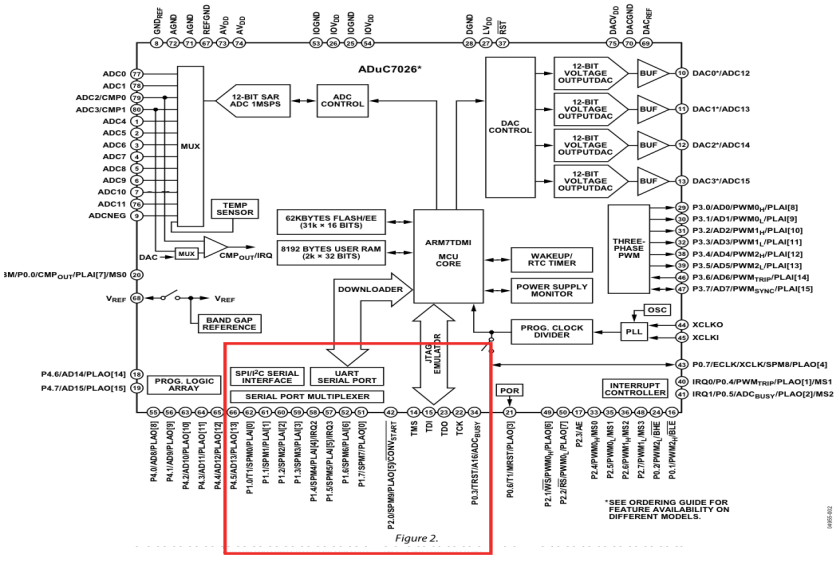
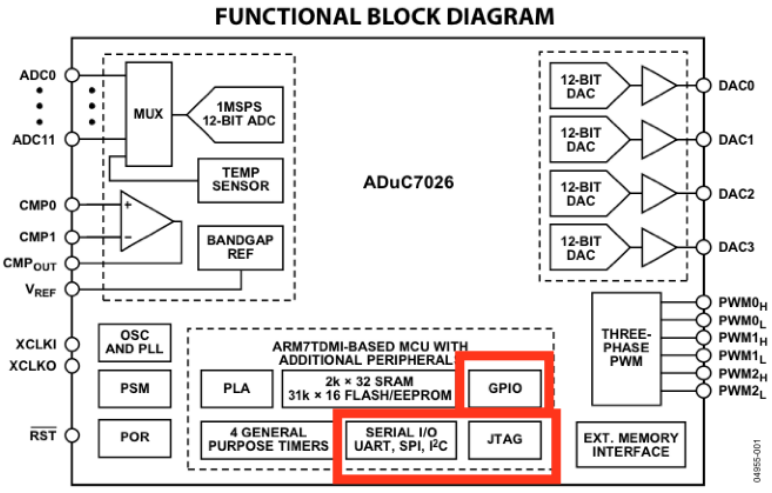


Figure 2.

Another I/O View of a Microcontroller



Interrupt example: Pitfalls

```
static int iTemperatures[2];        Use "volatile" keyword here!
void interrupt vReadTemperatures (void) {
        iTemperatures[0] = //Read value from hardware 1
        iTemperatures[1] = //Read value from hardware 2
}
void main(void) {
        int iTemp0, iTemp1;
        // Setup code
        iTemperatures[0] = 0;
        iTemperatures[1] = 0;
        while(TRUE) {
                iTemp0 = iTemperatures[0];
                iTemp1 = iTemperatures[1];
                if ( iTemp0 != iTemp1 ) {
                        // Set off alarm!
                }
        }
}
```

What if interrupt updates both values here?

What if compiler optimizes this to if false { set off alarm}

Interrupt example revisited

```
static volatile int iTemperatures[2];
void interrupt vReadTemperatures (void) {
        iTemperatures[0] = //Read value from hardware 1
        iTemperatures[1] = //Read value from hardware 2
}

void main(void) {
        int iTemp0, iTemp1;
        //Setup code

        while(TRUE) {
                disableInterrupts();
                iTemp0 = iTemperatures[0];
                iTemp1 = iTemperatures[1];
                enableInterrupts();
                if ( iTemp0 != iTemp1 )
                        // Set off alarm!
        }
}
```

Shared Data

- Data consistency
  - Critical Section
    - Need to protect portion of the code from other access
    - Disable interrupts
  - Compiler Optimizations
    - Various optimization techniques
    - Volatile keyword, or turn off optimizations

iRobot Drive example

```
SIGNAL(SIG_USART_RECV) {
  uint8_t temp;
  temp = UDR0;

  if(sensors_flag) {
    sensors_in[sensors_index++] = temp;
    if(sensors_index >= Sen6Size)
      sensors_flag = 0;
  }
}
```

```
void delayAndUpdateSensors(uint16_t time_ms){
  uint8_t temp;

  timer_on = 1;
  timer_cnt = time_ms;
  while(timer_on) {
    if(!sensors_flag){
      for(temp = 0; temp < Sen6Size; temp++) {
        sensors[temp] = sensors_in[temp];
      }
      // Update running totals of distance and angle

      byteTx(CmdSensors);
      byteTx(6);
      sensors_index = 0;
      sensors_flag = 1;
    }
  }
}
```

```
for(;;) {
    delayAndUpdateSensors(10);
    if(UserButtonPressed)
    {
      // Drive around until a button or unsafe condition is detected
      while(!(UserButtonPressed) && (!sensors[SenCliffL])
&& (!sensors[SenCliffFL]) && (!sensors[SenCliffFR])
&& (!sensors[SenCliffR])&& (!sensors[SenChAvailable])){
       // Keep turning until the specified angle is reached
       if(turning) {
        // Code to continue turning }

       // Check for a bump
       } else if(sensors[SenBumpDrop] & BumpEither) {
        // Set the turn parameters and reset the angle
        if(sensors[SenBumpDrop] & BumpLeft) {
          turn_dir = 0;
        } else {
          turn_dir = 1;
         //Command to turn iRobot }
       } else {
        // Otherwise, drive straight
        drive(300, RadStraight); }

       // Flash the leds in sequence
       // Update LED State
       // wait a little more than one robot tick for sensors to update
       delayAndUpdateSensors(20);
      } //End while loop
      // Stop driving
      drive(0, RadStraight);
    }
}
```