MICROCONTROLLERS
Overview

• A microcontroller (uC) is a small, lightweight CPU which is usually combined with on-board memory and peripherals
  – Compact and low power (relatively)
• Often used as a simple hardware to software interface as well as for in-situ processing
  – Analog to digital gateway
  – Allows for real-time feedback based on data
• A microcontroller may often be used in an embedded system
Types

There are a huge variety of microcontrollers:
- CPUs
  - 4bit, 8bit, 16bit, 32bit...
  - CISC, RISC
  - Range of possible clock frequencies
- Memory
  - FLASH, EEPROM, SRAM
- Integrated Peripherals
  - Number of I/O Ports
  - Number and type of the Timers
  - ADC
  - Other I/O Interfaces like USART, SPI or CAN
  - Wireless (Bluetooth, Wifi, etc.)
- Other features, like sleep mode for low energy consumption
Ex: embedded system

Sensor/Actuators

Digital

Analog
Embedded Controller

An embedded microcontroller is a microprocessor that is used in an embedded system. These processors are usually smaller, use a surface mount form factor and consume less power. Embedded processors can be divided into two categories: ordinary microprocessors and microcontrollers.

It is also a microcontroller in computers that handles various system tasks that the operating system does not handle.

Microcontrollers Vs. DSPs

• DSPs are optimized for math [multiply and accumulate]
• Embedded controller may not be a microcontroller per se but is used for special purpose control application
• Typical applications: temperature control, smart instrument, GPS, digital lock, cell phone, etc.
Examples of embedded systems

Personal information products: Cell phone, pager, watch, pocket recorder, calculator

- Laptop components: mouse, keyboard, modem, fax, card, sound card, battery charger

- Home appliances: door lock, alarm clock, thermostat, air conditioner, tv remote, hair dryer,

- DVR, small refrigerator, exercise equipment, washer/dryer, microwave oven

- Toys; video games, cars, dolls, etc.
Features

• Processor speed: Fundamental measure of processing rate of device
  – Value of interest is in MIPS, not MHz

• Supply voltage/current: Measure of the amount of power required to run the device
  – Multiple modes (sleep, idle, etc)

• It is possible to adjust the voltage and frequency of some devices in real time, thereby trading off speed and power usage

• Internal memory: Sometimes divided between program and data memory, the amount of information that can be stored on board

• Can sometimes be supplemented by external memory

• I/O Pins: Individual points for communication between the uC and the rest of the world

• Can be digital or analog, general or special purpose

• Interrupts: Non-linear program flow based on event triggers from peripheral or pins
Peripherals

• Timers: Internal registers (any size) in the uC that increment at the clock rate
  – May have prescaler
  – May be combined with range testing for interrupt
  – Watchdog timers reset processor if it hangs.
• Comparators: Input that effectively functions as a 1-bit ADC with an adjustable threshold
  – Used for real-time data monitoring
• ADC: Most ADCs used in sensor data collection are integrated with uController
• Watch for number of channels vs number of inputs
• Sampling speed does not include input switch time
• Very fast ADCs often combined with DMA
• DAC: Digital to analog converters are also include in some data collection driven uC
• Mostly used for feedback and control
Communication

- UART: Basic hardware module which mediates serial communication (RS232)
  - Simplest form of communication but limited by speed
  - Most modules are full duplex

- USB: High bandwidth serial communication between uC and a computer or an embedded host
  - Usually requires chips with specialized hardware and firmware
  - Host side issues

- SPI: Full duplex master-slave 4-wire protocol for data transfer between uCs
  - Mbit transfer rates, but somewhat quirky protocol
  - Unlimited (almost) nodes, can change master

- I2C: Half duplex master-slave 2-wire protocol for data transfer between uCs
  - kbit transfer rates
  - Tx/Rx based on slave addressing
  - Can invert protocol with sensors as masters

- RF: Radio frequency (>100 MHz) EM transmission of data
  - Built in to some newer special-purpose uC
  - Wireless spherical transmission
  - More later
Memory Map: a massive table, in effect a database, that comprises complete information about how the memory is structured in a computer system. Thorough understanding of this map is essential when programming microcontrollers.
Example microcontrollers

RISC / Harvard

• Licensed to Atmel (http://www.atmel.com)

• Powerful microcontroller designed for small applications

• Very low power operation

• 118 instructions

• 1 instruction per clock cycle (pipelined)

• Register-to-register operation

• Intended as a “single chip” solution
- **RISC core with ~100 instructions**
- Modest clock speeds (4-16 MHz)
- 8-bit bus, 32 GP 8-bit registers
- Intended as “single chip” solutions
- In-circuit programmable Flash (~1000 cycles)
- Small amount of EEPROM and SRAM
- Single-cycle execution of most instructions
- Several on-chip peripherals (UART, SPI, ADC, PWM, WDT)
Example microcontrollers

ATmega128 Architecture
Atmel AVR Registers

- 32 general-purpose 8-bit registers (r0 to r31).
- The upper six can also act as three 16-bit index registers (X, Y and Z).
The AVR has separate program and data spaces, and supports an address space of up to 8M.
Example microcontrollers

TI MSP430

• Proprietary TI low-power low-cost RISC chips
  – Well supported by TI with good program chain
  – Designed for intermittent sampling and fast startup

• General specs
  – Very low power (flexible)
  – Max 32KHz / 8 MIPS
  – Max 50K program space / 10K RAM
  – Max 16 bit ADC
  – UART/SPI/DAC/LCD/PWM/Comparators

• [http://www.msp430.com](http://www.msp430.com)
Example microcontrollers

• 8051 derivate uC with high reconfigurability
  – Many programming environments available
  – Vary from 3mm² to 100 pin packages

• General specs
  – Medium power
  – Max 100 MHz / 100 MIPS
  – Max 128K program space / 8K RAM
  – Max 16 bit ADC
  – UART/USB/SPI/CAN/PWM/Comparators

Example microcontrollers

Atmel ARM7 (AT91SAM7S series)

• 32-bit ARM microcontroller
  – Low power (for 32-bit machines)
  – Can run in 16-bit mode if needed

• General specs
  – Lots of memory (8-64KB RAM, 32-256KB flash)
  – Variable speed up to 55MHz
  – Packed with peripherals (USB, ADC, SPI, etc.)
  – Comes in LQFP 48 and 64 packages
  – Not suitable for beginners

• http://www.at91.com/
Example microcontrollers

• Analog Devices ADUC8xx:
  – More of an ADC with a uC attached
  – Some models include 24 bit sigma-delta converter
  – Useful with IEEE 1451 (see later)
  – [http://www.analog.com/IST/SelectionTable/?selection_table_id=212](http://www.analog.com/IST/SelectionTable/?selection_table_id=212)

• Chipcon CCxxxx:
  – More of an RF transceiver with a uC attached
  – Variety of frequency ranges and modulation schemes

• Link to top microcontroller companies in the world:
Example microcontrollers

Motorola M68HC05

Von Neumann Architecture
Device Memory Map
Example microcontrollers

Microchip PIC

Internal Structure
Harvard Architecture
PIC Microcontrollers

Memory Map
PIC RISC instructions

<table>
<thead>
<tr>
<th>Instruction-Binary (Hex)</th>
<th>Name</th>
<th>Mnemonic, Operands</th>
<th>Operation</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001 1df ffff 3cf</td>
<td>Add W and f</td>
<td>ADDWF f, d</td>
<td>W + f → d</td>
<td>C, D, Z</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>0001 1df ffff 3af</td>
<td>AND W and f</td>
<td>ANDWF d, W &amp; f</td>
<td>Z</td>
<td>2, 4</td>
<td></td>
</tr>
<tr>
<td>0000 01ff ffff 0cf</td>
<td>Clear f</td>
<td>CLR f</td>
<td>0 → f</td>
<td>Z</td>
<td>4</td>
</tr>
<tr>
<td>0010 01df ffff 0cf</td>
<td>Clear W</td>
<td>CLRW</td>
<td>0 → W</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>0000 1df ffff 2cf</td>
<td>Decrement f</td>
<td>DECF f, d</td>
<td>1 → d</td>
<td>Z</td>
<td>2, 4</td>
</tr>
<tr>
<td>0010 1df ffff 2cf</td>
<td>Decrement f, Skip if Zero</td>
<td>DECFSZ f, d</td>
<td>1 → d, skip if zero</td>
<td>None</td>
<td>2, 4</td>
</tr>
<tr>
<td>0101 1df ffff 28cf</td>
<td>Increment f</td>
<td>INCF f, d</td>
<td>1 → d</td>
<td>Z</td>
<td>2, 4</td>
</tr>
<tr>
<td>0111 1df ffff 3cf</td>
<td>Increment f, Skip if Zero</td>
<td>INCFSZ f, d</td>
<td>1 → d, skip if zero</td>
<td>None</td>
<td>2, 4</td>
</tr>
<tr>
<td>0001 0df ffff 10cf</td>
<td>Set OR W and f</td>
<td>IORWF d, W v → d</td>
<td>Z</td>
<td>2, 4</td>
<td></td>
</tr>
<tr>
<td>0001 0df ffff 20cf</td>
<td>Move f</td>
<td>MOVF f, d</td>
<td>d → f</td>
<td>Z</td>
<td>2, 4</td>
</tr>
<tr>
<td>0000 01ff ffff 02cf</td>
<td>Move W to f</td>
<td>MOVWF f, d</td>
<td>f → f</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>0006 0000 0000 0cf</td>
<td>No Operation</td>
<td>NOP</td>
<td></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>0101 0df ffff 34cf</td>
<td>Rotate Left f</td>
<td>RLf, d</td>
<td>f → (f+n)</td>
<td>C</td>
<td>2, 4</td>
</tr>
<tr>
<td>0110 0df ffff 30cf</td>
<td>Rotate Right f</td>
<td>Rlf, d</td>
<td>f → (f-n)</td>
<td>C</td>
<td>2, 4</td>
</tr>
<tr>
<td>0000 1df ffff 08cf</td>
<td>Subtract W from f</td>
<td>SUBWF f, d</td>
<td>d → (d-W)</td>
<td>C, D, Z</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>0001 1df ffff 38cf</td>
<td>Swap Halves f</td>
<td>SWAP f, d</td>
<td>(d-0) ↔ (d-7)</td>
<td>None</td>
<td>2, 4</td>
</tr>
<tr>
<td>0001 1df ffff 18cf</td>
<td>Exclusive OR W and f</td>
<td>XORWF f, d</td>
<td>d → f → d</td>
<td>C</td>
<td>2, 4</td>
</tr>
</tbody>
</table>

BIT-ORIENTED FILE REGISTER OPERATIONS

<table>
<thead>
<tr>
<th>Instruction-Binary (Hex)</th>
<th>Name</th>
<th>Mnemonic, Operands</th>
<th>Operation</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100 bbb ffff 8bf</td>
<td>Bit Clear f</td>
<td>BCF f, b</td>
<td>0 → (f)</td>
<td>None</td>
<td>2, 4</td>
</tr>
<tr>
<td>0100 bbb ffff 8bf</td>
<td>Bit Set f</td>
<td>BSF f, b</td>
<td>1 → (f)</td>
<td>None</td>
<td>2, 4</td>
</tr>
<tr>
<td>0110 bbb bbb ffff 8bf</td>
<td>Bit Test f, Skip if Clear</td>
<td>BTFS f, b</td>
<td>Test bit (b) in file (f), skip if clear</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>0111 bbb bbb ffff 7bf</td>
<td>Bit Test f, Skip if Set</td>
<td>BTFT f, b</td>
<td>Test bit (b) in file (f), skip if set</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

LITERAL AND CONTROL OPERATIONS

<table>
<thead>
<tr>
<th>Instruction-Binary (Hex)</th>
<th>Name</th>
<th>Mnemonic, Operands</th>
<th>Operation</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>AND Literal and W</td>
<td>ANDLW k, w</td>
<td>k &amp; W → W</td>
<td>Z</td>
<td>1</td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Call subroutine</td>
<td>CALL k</td>
<td>PC → Stack, k → PC</td>
<td>None</td>
<td>1</td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Clear Watchdog timer</td>
<td>CLRWD T</td>
<td>0 → WD T (and prescaler, if assigned)</td>
<td>TO, PD</td>
<td></td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Go To address (k is 8 bits)</td>
<td>GOTOF k</td>
<td>k → PC (9 bits)</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Indirect OR Literal and W</td>
<td>IORLW k, w</td>
<td>k &amp; W → W</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Move Literal to W</td>
<td>MOVLW k</td>
<td>k → W</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Return address</td>
<td>RETLW k</td>
<td>k → Stack, PC</td>
<td>TO, PD</td>
<td></td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Return to standby mode</td>
<td>SLEEP</td>
<td>0 → WD T, stop oscillator</td>
<td>TO, PD</td>
<td></td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Treat as peripheral</td>
<td>TRIS f, w</td>
<td>W → I/O control register (f)</td>
<td>None</td>
<td>3</td>
</tr>
<tr>
<td>0000 0000 0000 0000 kkkk</td>
<td>Exclusive OR Literal and W</td>
<td>XORLW k, w</td>
<td>k &amp; W → W</td>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>
Programming Languages

• C / C++
  – Most used
  – HiTech C
  – Microchip C
  – CCS PIC C
  – Arduino

• Java
  - typically compiled to bytecode that can run on any Java virtual machine

• ASM
  – Low level
  – Full Control

• BASIC, Forth, LOGO
  – Interpreted
  – Easy to use
  – Slow
Program Example: Loop

/* pulses pin PORTB<3> eight times */
pulse:
  movlw 0x08
  movwf counter

pulse_lp0:
  bsf PORTB, 3
  bcf PORTB, 3
  decfsz counter, F
  goto pulse_lp0
return

/* pulses pin PORTB<3> eight times */
void pulse()
{
  int i;
  for (i=0; i<8; i++)
  {
    output_high(PIN_B3);
    output_low(PIN_B3);
  }
  return;
}
Compilers’ Inefficiency

/* pulses pin
PORTB<3> eight times */

0000: movlw 0x8
0001: movwf 0x20
0002: bsf 0x6,0x3
0003: bcf 0x6,0x3
0004: decfsz 0x20
0005: clrf 21
0006: movf 21,W
0007: sublw 07
0008: btfss 03,0
0009: goto 014
000A: bsf 03,5
000B: bcf 06,3
000C: bcf 03,5
000D: bsf 06,3
000E: bsf 03,5
000F: bcf 06,3
0010: bcf 03,5
0011: bcf 06,3
0012: incf 21,F
0013: goto 006

Handwritten Code

Compiler’s ASM Code

See http://www.ccsinfo.com/picc.shtml for compiler’s info
TI TMS370Cx Series Microcontrollers

functional block diagram

Interrupts
Oscillator Clock
System Control

CPU
RAM (256 Bytes Usable as Registers)
Program Memory 4K or 8K Bytes (ROM or EPROM)
Data EEPROM 256 Bytes

A-to-D Converter
(40 - Pin: 4 CH
44 - Pin: 8 CH)

Serial Communications Interface
Timer 2
Timer 1
Watchdog

Port A
Port B
Port D

40-PIN DIP: AN2, AN3, AN6, AN7
44-PIN PLCC: AN0–AN7

Von Neumann Architecture
TI TMS Internal Registers

Figure 4. CPU Registers
Memory Map

0000h
00FFh
0100h
0FFFh
1000h
10FFh
1100h
1EFFh
1F00h
1FFFh
2000h
5FFFh
6000h
6FFFh
7000h
7FFFh
8000h
FFFFh

Reserved
System Control
Digital Port Control
Reserved
Timer 1 Peripheral Control
SCI Peripheral Control
Timer 2 Peripheral Control
ADC Peripheral Control
Reserved

Vectors
Trap 15-0
Reserved
A-D Converter
Timer 2
Serial Comm I/F Tx
Serial Comm I/F Rx
Timer 1
Interrupt 3
Interrupt 2
Interrupt 1
Reset

1000h - 100Fh
1010h - 101Fh
1020h - 102Fh
1030h - 103Fh
1040h - 104Fh
1050h - 105Fh
1060h - 106Fh
1070h - 107Fh
1080h - 10FFh
7FC0h - 7FDFh
7FE0h - 7FEBh
7FECb - 7FEDh
7FEeh - 7FEFh
7FF0h - 7FF1h
7FF2h - 7FF3h
7FF4h - 7FF5h
7FF8h - 7FF9h
7FFAh - 7FFBh
7FFCh - 7FFDh
7FFEh - 7FFFh
# Intel 8051 Microcontrollers

## Table 1. 8051 Family of Microcontrollers

<table>
<thead>
<tr>
<th>Device Name</th>
<th>ROMless Version</th>
<th>EPROM Version</th>
<th>ROM Bytes</th>
<th>RAM Bytes</th>
<th>16-Bit Timers</th>
<th>Ckt Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>8051AH</td>
<td>8031AH</td>
<td>–</td>
<td>4K</td>
<td>128</td>
<td>2</td>
<td>HMOS</td>
</tr>
<tr>
<td>8052AH</td>
<td>8032AH</td>
<td>–</td>
<td>8K</td>
<td>256</td>
<td>3</td>
<td>HMOS</td>
</tr>
<tr>
<td>80C51BH</td>
<td>80C31BH</td>
<td>87C51</td>
<td>4K</td>
<td>128</td>
<td>2</td>
<td>CHMOS</td>
</tr>
<tr>
<td>83C451</td>
<td>80C451</td>
<td>87C451</td>
<td>4K</td>
<td>128</td>
<td>2</td>
<td>CHMOS</td>
</tr>
<tr>
<td>83C751</td>
<td>–</td>
<td>87C751</td>
<td>2K</td>
<td>64</td>
<td>1</td>
<td>CHMOS</td>
</tr>
<tr>
<td>83C652</td>
<td>80C652</td>
<td>87C652</td>
<td>8K</td>
<td>256</td>
<td>2</td>
<td>CHMOS</td>
</tr>
<tr>
<td>83C552</td>
<td>80C552</td>
<td>87C552</td>
<td>8K</td>
<td>256</td>
<td>3</td>
<td>CHMOS</td>
</tr>
</tbody>
</table>

## Table 2. 80C51 Derivative Comparisons

<table>
<thead>
<tr>
<th>Device</th>
<th>A/D</th>
<th>Ports</th>
<th>PWM</th>
<th>Timers</th>
<th>UART</th>
</tr>
</thead>
<tbody>
<tr>
<td>80C51</td>
<td>–</td>
<td>4</td>
<td>–</td>
<td>2 std</td>
<td>std</td>
</tr>
<tr>
<td>83C451</td>
<td>–</td>
<td>7</td>
<td>–</td>
<td>2 std</td>
<td>std</td>
</tr>
<tr>
<td>83C751</td>
<td>–</td>
<td>2–3/8</td>
<td>–</td>
<td>1 std, timer 1</td>
<td>I²C</td>
</tr>
<tr>
<td>83C652</td>
<td>–</td>
<td>4</td>
<td>–</td>
<td>2 std</td>
<td>std, I²C</td>
</tr>
<tr>
<td>83C552</td>
<td>8 chan 10 bit</td>
<td>5</td>
<td>2</td>
<td>2 std, timer 3, watchdog</td>
<td>std, I²C</td>
</tr>
</tbody>
</table>
8051 Internal Structure

Von Neumann Architecture
### 8051 Memory Map

<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Bit Addressable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>F8</td>
<td></td>
<td>FF</td>
</tr>
<tr>
<td>F0</td>
<td></td>
<td>F7</td>
</tr>
<tr>
<td>E8</td>
<td></td>
<td>EF</td>
</tr>
<tr>
<td>E0</td>
<td></td>
<td>E7</td>
</tr>
<tr>
<td>D8</td>
<td></td>
<td>DF</td>
</tr>
<tr>
<td>D0</td>
<td></td>
<td>D7</td>
</tr>
<tr>
<td>C8</td>
<td></td>
<td>CF</td>
</tr>
<tr>
<td>(T2CON)</td>
<td></td>
<td>(RCAP2L)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(RCAP2H)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(TL2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(TH2)</td>
</tr>
<tr>
<td>CO</td>
<td></td>
<td>C7</td>
</tr>
<tr>
<td>B8</td>
<td></td>
<td>BF</td>
</tr>
<tr>
<td>B0</td>
<td></td>
<td>B7</td>
</tr>
<tr>
<td>A8</td>
<td></td>
<td>AF</td>
</tr>
<tr>
<td>A0</td>
<td></td>
<td>A7</td>
</tr>
<tr>
<td>98</td>
<td></td>
<td>9F</td>
</tr>
<tr>
<td>90</td>
<td></td>
<td>97</td>
</tr>
<tr>
<td>88</td>
<td></td>
<td>8F</td>
</tr>
<tr>
<td>(TCON)</td>
<td></td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCON</td>
</tr>
</tbody>
</table>

Legend:
- FB: B
- E8: ACC
- E0: PSW
- D8: IP
- D0: P3
- C0: IE
- B8: P2
- 98: SCON
- 90: SBUF
- 88: TCON
- 80: TMOD
- P0: SP
- DPL
- DPH
- PCON
General Design rules - outputs

Many logic chips, output pins, circuits use open-collector (open-drain)

- Simple logic-glue, signal bus
- Wired-OR circuits
- Interface different voltage levels.

Interface between 5V CMOS logic and a higher voltage
Interrupt operation

- A way to respond to an external event (i.e., flag being set) without polling

How it works:
- H/W senses flag being set
- Automatically transfers control to s/w that “services” the interrupt
- When done, H/W returns control wherever it left off

Advantages:
- Transparent to user
- Cleaner code
- μC doesn’t waste time polling
Shorthand for an “Inter-integrated circuit” bus

- I2C devices include EEPROMs, thermal sensors, and real-time clocks
- Used as a control interface to signal processing devices that have separate data interfaces, e.g. RF tuners, video decoders and encoders, and audio processors.
- I2C bus has three speeds:
  - Slow (under 100 Kbps)
  - Fast (400 Kbps)
  - High-speed (3.4 Mbps) – I2C v.2.0
- Limited to about 10 feet for moderate speeds
I²C bus configuration

- 2-wire serial bus – Serial data (SDA) and Serial clock (SCL)
- Half-duplex, synchronous, multi-master bus
- No chip select or arbitration logic required
- Lines pulled high via resistors, pulled down via open-drain drivers (wired-AND)
I²C Features

- “Clock stretching” – when the slave (receiver) needs more time to process a bit, it can pull SCL low. The master waits until the slave has released SCL before sending the next bit.

- “General call” broadcast – addresses every device on the bus

- 10-bit extended addressing for new designs. 7-bit addresses all exhausted
Microcontroller serial communication

Shorthand for “Serial Peripheral Interface”

- Defined by Motorola on the MC68HCxx line of microcontrollers
- Generally faster than I2C, capable of several Mbps

Applications:

- Like I2C, used in EEPROM, Flash, and real time clocks
- Better suited for “data streams”, i.e. ADC converters
- Full duplex capability, i.e. communication between a codec and digital signal processor
SPI bus configuration

- Synchronous serial data link operating at full duplex
- Master/slave relationship
- 2 data signals:
  - MOSI – master data output, slave data input
  - MISO – master data input, slave data output
- 2 control signals:
  - SCLK – clock
  - /SS – slave select (no addressing)
As the register transmits the byte to the slave on the MOSI signal line, the slave transfers the contents of its shift register back to the master on the MISO signal line, exchanging the contents of the two shift registers.
Microcontroller serial communication

- For point-to-point, SPI is simple and efficient
  - Less overhead than \( I^2C \) due to lack of addressing, plus SPI is full duplex.

- For multiple slaves, each slave needs separate slave select signal
  - More effort and more hardware than \( I^2C \)
Shorthand for “Universal Asynchronous Receiver-Transmitter “

• A UART’s transmitter is essentially just a parallel-to-serial converter with extra features.

• The essence of the UART transmitter is a shift register that is loaded in parallel, and then each bit is sequentially shifted out of the device on each pulse of the serial clock.

• One of the problems associated with serial transmission is reconstructing the data at the receiving end.

• Difficulties arise in detecting boundaries between bits.
UART

Asynchronous serial devices, such as UARTs, do not share a common clock

- Each device has its own, local clock.
- The devices must operate at exactly the same frequency.
- Logic (within the UART) is required to detect the phase of the transmitted data and phase lock the receiver’s clock to this.
The transmission format uses one start bit at the beginning and one or two stop bits end of each character.
Analog to digital conversion

Discretization of time

\[ V_e \text{ is a mapping } \mathbb{R} \rightarrow \mathbb{R} \]

Restriction to digital information processing.
Known digital computers can only process discrete time series
Sample and hold-devices.
Ideally: width of clock pulse \( \rightarrow 0 \)

- \( V_x \) is a **sequence** of values or a mapping \( \mathbb{Z} \rightarrow \mathbb{R} \)
Digital computers require digital form of physical values — A/D-conversion. An Analog to Digital Converter (ADC) converts an analog input voltage to a digital number. Many methods with different speeds. Example: 1. Flash A/D converter:

- Parallel comparison with reference voltage
- Speed: $O(1)$
- Hardware complexity: $O(n)$
- with $n$ = # of distinguished voltage levels
ADC successive approximation

Key idea: binary search:
- Set MSB='1'
- if too large: reset MSB
- Set MSB-1='1'
- if too large: reset MSB-1

Speed: $O(ld(n))$

Hardware complexity: $O(ld(n))$

with $n=$ # of distinguished voltage levels;
slow, but high precision possible.
Pulse width modulation (PWM)

\( T_{\text{on}} \) is the time for which the output is high and \( T_{\text{off}} \) is time for which output is low. Let \( T_{\text{total}} \) be time period of the wave such that,

\[
T_{\text{total}} = T_{\text{on}} + T_{\text{off}}
\]

Duty cycle of a square wave is defined as

\[
D = \frac{T_{\text{on}}}{(T_{\text{on}} + T_{\text{off}})} = \frac{T_{\text{on}}}{T_{\text{total}}}
\]

The output voltage varies with duty cycle as...

\[
V_{\text{out}} = D \times V_{\text{in}}
\]

\[
V_{\text{out}} = \frac{T_{\text{on}}}{T_{\text{total}}} \times V_{\text{in}}
\]
Summary

Microcontrollers are self contained processing systems that may be configured using Harvard or Von Neumann architectures.

They may be RISC or CISC machines.

Developers should be intimately familiar with a particular memory map for the processor they choose to design with.

Real time applications require the use of interrupt based programming.

Many microcontrollers use serial interfaces such as UARTs, SPI and I²C.

There are many types of analog to digital converters available for microcontrollers so a developer must select the best one for his application.

Pulse width modulation outputs enable the microcontroller to produce various signals ranging from simple clocks to tone bursts for audio frequencies.