Hardware Architectures For Embedded Systems Design

Prepared By: Hind Alsalem;  
Supervised By: Dr. Lo’ai Tawalbeh  
Jordan University of Science and Technology
Embedded System Definition:

- Any electronic system that uses a CPU chip, but that not a general-purpose workstation, desktop or laptop computer.

- Such systems generally use microprocessors, or they may use custom-designed chips or both.

- They are used in automobiles, planes, trains, space vehicles, machine tools, cameras, consumer and office appliances, cellphones, PDAs and other handhelds as well as robots and toys.
From the previous definition, (Such systems generally use microprocessors, or they may use custom-designed chips or both), Embedded Systems can be classified based on the core hardware used to implement the system into:

- Microprocessor- based EMS
- Microcontroller- based EMS
- DSP- based EMS
- FPGA- based EMS
- ASIC- based EMS

Let’s study each type
For each type, the following terms will be discussed:

- The core hardware architecture.
- The Embedded Language.
- A case study. (To discuss the hardware architecture).
- Real Examples. (Real life applications)
- Comparison with other types.

Many Hardware design issues will be discussed, so feel free for any question, comment or discussion.
Discussion

Are embedded systems usually real-time systems?
Are real-time systems usually embedded systems?

- Real-time does not mean super fast but merely meet the requirements, most embedded systems are in the category "as fast as possible" systems.

- It is rare to find an embedded system that doesn't have some hard real-time deadlines, such as processing an interrupt before the next one occurs. Most also have soft real-time deadlines.

- It is also rare to find a real-time system that isn't embedded. If the system has hard real-time deadlines and is important, then you put it on its own circuit board and not on a desktop PC with Windows.
Microprocessor-based Embedded Systems (MPU)
1. Microprocessor-based Embedded Systems (MPU)

Microprocessors in general:

- A microprocessor is a digital logic circuit manufactured using VLSI (very large scale integration) technology.

A Microprocessor is able to perform the following functions:

- The ability to execute a stored set of instructions to carry out user defined tasks.
- The ability to access external memory chips to both read and write data from and to the memory.
- The ability to access I/O devices.
So, any microprocessor based-system takes the following hardware architecture:

Von-Neumann Architecture

Program & Data Memory
- Dynamic RAM (DRAM)
- Static RAM (SRAM)
- Cache
- ROM
- Flash Memory
- EEPROM
- SDRAM
- RAMBUS

CPU (MPU)
- (ALU, Registers, Control)

Buses
- Control Bus, Address Bus, Data Bus

Input & Output (I/O)
- Intel
- AMD
- Motorola
- IBM

Separate Data and instruction buses are called Harvard Architecture

Printer
- Serial Communication
- Hard disk drive
- Mouse
- CD-ROM
- Plotter
- Keyboard
- Monitor
- Tape back up
- Scanner
- DVD

Microprocessor – based system Architecture
Microprocessors are classified based on the systems that are used in into:

- **Desktop (Laptop):** Desktop processors market tend to be driven to:
  - optimize *price-performance*, as a result, desktop systems are where the newest, highest-performance microprocessors appear.

- **Servers:** server systems tend to achieve availability, reliability, scalability.

- **Embedded:** Embedded processors are lodged in devices where the presence of the computer is not immediately obvious. These systems need to optimize:
  - performance at a minimum price, minimize memory, minimize power.

*To be discussed*
Microprocessors in Embedded Systems:

Available Embedded processors by: 75%

- ARM, Motorola M680X0, Motorola PowerPC, Motorola M88K, MIPS, Hitachi Super-H (SH3/SH4), Intel i386 and above, others.

Case Study: ARM Embedded Processor

Cortex-M3 (Architecture/Features):

1. Higher performance through better efficiency:

   - Processors can either work hard or work smart.
   - Higher clock frequencies → increase performance → higher power consumption and design complexity.
   - Higher compute efficiency at slower clock speeds → simpler and lower power designs that can perform the same tasks.
   - 3-stage pipeline core
   - New powerful features → branch speculation, single cycle multiply and hardware divide
2. Ease of use for quick and efficient application development:
   - Fast and easy to program
   - Thumb-2 Instruction Set Architecture (ISA)

3. Reduced costs and lower power for sensitive markets:
   - Just 33,000 gates in the central core
   - Tightly coupled system components in the processor
   - Thumb-2 instruction set that reduces instruction memory requirements
   - A power consumption of just 4.5mW through extensive clock gating and integrated sleep modes.

4. Integrated debug and trace for faster time to market:
   - Implements debug technology in the hardware itself
   - High level of visibility into the system through a traditional JTAG port
Cortex-M3 Processor Architecture:

1. The Cortex-M3 Core:

- **Harvard Architecture** by being able to read both an instruction and data from memory at the same time, the Cortex-M3 processor can perform many **operations in parallel**, speeding application execution.

- **Pipeline has 3 stages**: Instruction Fetch, Instruction Decode and Instruction Execute. With speculative branch prediction.

- **Decoder** for traditional Thumb and new Thumb-2 instructions.

- **Advanced ALU**.

- The Cortex-M3 processor is a **32-bit processor** with a 32-bit wide data path, register bank and memory interface. There are 13 general-purpose registers, two stack pointers, a link register, program counter and a number of special registers including a program status register.

- Supports 32-bit **multiply** operations in a **single cycle** and also supports signed and unsigned **divide** operations that take between 2 and **12 cycles**.

- The Cortex-M3 processor is a **memory mapped** system with a simple, fixed memory map for up to 4 gigabytes.

*In the next slide*
The Memory Map

- Vendor Specific
- Private Peripheral Bus - External
- Private Peripheral Bus - Internal
- External Device (1GB)
- External RAM (1GB)
- Peripheral (0.5GB)
- SRAM (0.5GB)
- Code (0.5GB)
2. Thumb-2 Instruction Set Architecture

- Thumb-2 technology is a blend of 16 and 32-bit instructions
- delivers the performance of 32-bit ARM instructions,
- matches the code density and compatible with the original 16-bit Thumb instruction set.
3. The Nested Vectored Interrupt Controller (NVIC)

- The NVIC supports nesting (stacking) of interrupts, allowing an interrupt to be serviced earlier by exerting higher priority.
- It also supports dynamic reprioritization of interrupts. Priority levels can be changed by software during run time.
- Interrupts that are being serviced are blocked from further activation until the interrupt service routine is completed, so their priority can be changed without risk of accidental re-entry.
- Supports tail chaining.
4. The Memory Protection Unit (MPU)

The MPU is an optional component of the Cortex-M3 processor that protects critical data used by the operating system from user applications.

5. Debug and Trace (DAP Debug Access port)

- Debug actions can be triggered by various events like breakpoints, watchpoints, fault conditions, or external debug requests.

- When a debug event takes place, the Cortex-M3 processor can either enter the halt mode or the debug monitor mode.
Real applications that use Cortex-M3 Processor:

**W950**
The W950i Walkman® is a slim and stylish mobile phone with an advanced digital music player and large touchscreen for optimal ease of use.

**N93**
Boasting a 3.2 megapixel camera with Carl Zeiss optics, 3x optical zoom and DVD-like video capture, the Nokia N93 also offers advanced connectivity and editing options.

**Blu-Ray BD-P1000 DVD Player**
The Samsung BD-P1000 boasts the highest-definition playback in the world. With the BD-P1000 you can be among the first to view Blu-ray format discs on your HDTV.

**PIN 570 GPS PDA**
The Navman PIN 570 GPS PDA uses the very latest GPS technology combined with Navman’s navigation software.
Microcontroller- based Embedded Systems (MCU)
• **A Microcontroller** is a device which integrates a number of the components of a microprocessor system onto a single microchip.

• The CPU core (Microprocessor ranging from simple 8 bit to sophisticated 64 bit processor), memory (both ROM and RAM), I/O ports, serial and parallel ports, Timers, A/D & D/A and PWM are integrated within one chip.

![A Single Chip Microcontroller](image)

**Microcontroller Features:**

• Microcontrollers have Harvard architecture,

• long word instructions,

• single word instructions,

• single cycle instruction,

• Instruction pipelining,

• RISC (Reduced Instruction set),

• Register file architecture.
Available Microcontrollers by:

- **Intel**: 8051, 8096, 80960
- **Microchip**: PIC 12Cxxx/12Fxxx, PIC 16C5X, PIC 16Cxxx/16Fxxx, PIC 17Cxxx, PIC 18Cxxx / 18Fxxx
- **Texas Instruments**: TMS370
- **Motorola**: 68HC11, 68HC16, MPC500

**Case Study: PIC Microcontroller**

**PIC 16F877 – 40-pin 8-Bit CMOS FLASH Microcontrollers**

(Architecture/Features)
Microcontroller Core Features:

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
- Up to 368 x 8 bytes of Data Memory (RAM)
- Up to 256 x 8 bytes of EEPROM data memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
- Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
**Microcontroller Core Features:**

- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS FLASH/EEPROM technology
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption:

**Peripheral Features:**

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) • Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)
PIC 16F877 Architecture

How this Architecture fits for an EMS ?????
PIC 16F877 Program Memory Organization
## Data Memory Organization

<table>
<thead>
<tr>
<th>Indirect addr.</th>
<th>Indirect addr.</th>
<th>Indirect addr.</th>
<th>Indirect addr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>01h</td>
<td>08h</td>
<td>09h</td>
</tr>
<tr>
<td>TMR0</td>
<td>OPTION_REG</td>
<td>TMR0</td>
<td>OPTION_REG</td>
</tr>
<tr>
<td>01h</td>
<td>02h</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>PCL</td>
<td>PCL</td>
<td>PCL</td>
<td>PCL</td>
</tr>
<tr>
<td>02h</td>
<td>03h</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>STATUS</td>
<td>STATUS</td>
<td>INTCON</td>
<td>INTCON</td>
</tr>
<tr>
<td>03h</td>
<td>04h</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>FSR</td>
<td>FSR</td>
<td>INTCON</td>
<td>INTCON</td>
</tr>
<tr>
<td>04h</td>
<td>05h</td>
<td>0Ch</td>
<td>0Ch</td>
</tr>
<tr>
<td>PORTA</td>
<td>TRISA</td>
<td>PIE1</td>
<td>EEDATA</td>
</tr>
<tr>
<td>05h</td>
<td>06h</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>PORTB</td>
<td>TRISB</td>
<td>PIE2</td>
<td>EEADR</td>
</tr>
<tr>
<td>06h</td>
<td>07h</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>PORTC</td>
<td>TRISC</td>
<td>PCON</td>
<td>EEDATH</td>
</tr>
<tr>
<td>07h</td>
<td>08h</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>PORTD</td>
<td>TRISD (n)</td>
<td>8Ah</td>
<td>8Ah</td>
</tr>
<tr>
<td>08h</td>
<td>09h</td>
<td>8Ah</td>
<td>8Ah</td>
</tr>
<tr>
<td>PORTE (n)</td>
<td>TRISE (n)</td>
<td>8Ah</td>
<td>8Ah</td>
</tr>
<tr>
<td>09h</td>
<td>0Ah</td>
<td>8Ah</td>
<td>8Ah</td>
</tr>
<tr>
<td>PCLATH</td>
<td>PCLATH</td>
<td>EEDATA</td>
<td>EECON1</td>
</tr>
<tr>
<td>0Ah</td>
<td>0Bh</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>INTCON</td>
<td>INTCON</td>
<td>EEADR</td>
<td>EECON2</td>
</tr>
<tr>
<td>0Bh</td>
<td>0Ch</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>PIR1</td>
<td>PIE1</td>
<td>EEADRH</td>
<td>Reserved (n)</td>
</tr>
<tr>
<td>0Ch</td>
<td>0Dh</td>
<td>0Ah</td>
<td>0Ah</td>
</tr>
<tr>
<td>PIR2</td>
<td>PIE2</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>0Dh</td>
<td>0Eh</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>TMR1L</td>
<td>PCON</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>0Eh</td>
<td>0Fh</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>TMR1H</td>
<td>EEADRH</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>0Fh</td>
<td>10h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>T1CON</td>
<td>11h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>10h</td>
<td>11h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>TMR2</td>
<td>SSPCON2</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>11h</td>
<td>12h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>T2CON</td>
<td>PR2</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>12h</td>
<td>13h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>SSPBUF</td>
<td>SSPADD</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>13h</td>
<td>14h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>SSPCON</td>
<td>SSPSTAT</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>14h</td>
<td>15h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>CCPR1L</td>
<td>16h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>15h</td>
<td>16h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>CCPR1H</td>
<td>17h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>16h</td>
<td>17h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>CCP1CON</td>
<td>18h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>17h</td>
<td>19h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>RCSTA</td>
<td>1Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>18h</td>
<td>19h</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>TXREG</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>19h</td>
<td>1Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>RCREG</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>1Ah</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>CCPR2L</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>1Bh</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>CCPR2H</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>1Ch</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>CCP2CON</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>1Dh</td>
<td>9Ah</td>
<td>10Ah</td>
<td>18Ah</td>
</tr>
<tr>
<td>ADRESH</td>
<td>1Eh</td>
<td>11Ah</td>
<td>19Ah</td>
</tr>
<tr>
<td>1Eh</td>
<td>11Ah</td>
<td>120h</td>
<td>19Fh</td>
</tr>
<tr>
<td>ADCON0</td>
<td>11Fh</td>
<td>120h</td>
<td>19Fh</td>
</tr>
<tr>
<td>1Fh</td>
<td>120h</td>
<td>120h</td>
<td>19Fh</td>
</tr>
<tr>
<td>ADCCON1</td>
<td>120h</td>
<td>120h</td>
<td>19Fh</td>
</tr>
</tbody>
</table>

**General Purpose Register**

- Bank 0: 00h-07h
- Bank 1: 08h-0Fh
- Bank 2: 10h-17h
- Bank 3: 18h-1Fh

**What’s the benefit of a banked memory?**

- **Banking** allows for efficient memory management by dividing the memory into smaller blocks, which can be independently accessed or protected.
- **Confidentiality** can be maintained by securing different banks with different access permissions.
- **Flexibility** allows for dynamic allocation and deallocation of memory resources as needed by the application.
### TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on:</th>
<th>Value on all other resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h[4]</td>
<td>INDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>01h</td>
<td>TMRO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>02h[4]</td>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>03h[4]</td>
<td>STATUS</td>
<td>IRP</td>
<td>RP1</td>
<td>RP0</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC</td>
<td>C</td>
<td>0001 1xxx</td>
<td>0000 0000</td>
</tr>
<tr>
<td>04h[4]</td>
<td>FSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>05h</td>
<td>PORTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RE2</td>
<td>RE1</td>
<td>RE0</td>
<td></td>
</tr>
<tr>
<td>06h</td>
<td>PORTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-0x 0000</td>
<td>-0x 0000</td>
</tr>
<tr>
<td>07h</td>
<td>PORTC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>08h[6]</td>
<td>PORTD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>09h[6]</td>
<td>PORTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>0Ah[4,4]</td>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-0 0000</td>
<td>-0 0000</td>
</tr>
<tr>
<td>0Bh[4]</td>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>T0IE</td>
<td>INTE</td>
<td>RBIE</td>
<td>T0IF</td>
<td>INTF</td>
<td>RBIF</td>
<td>0000 000x</td>
<td>0000 000x</td>
</tr>
<tr>
<td>0Ch</td>
<td>PIR1</td>
<td>pS/C/F[9]</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSIIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>0Dh</td>
<td>PIR2</td>
<td>(6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CDP2IF</td>
<td>-x- 0 0 -0 -0</td>
</tr>
<tr>
<td>0Eh</td>
<td>TMR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>0Fh</td>
<td>TMR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>10h</td>
<td>T1CON</td>
<td></td>
<td></td>
<td></td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCN</td>
<td>T1SYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
<td>--0 0000</td>
</tr>
<tr>
<td>11h</td>
<td>TMR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>12h</td>
<td>T2CON</td>
<td></td>
<td>TOUTPS3</td>
<td>TOUTPS2</td>
<td>TOUTPS1</td>
<td>TOUTPS0</td>
<td>TMR2ON</td>
<td>T2CKPS1</td>
<td>T2CKPS0</td>
<td>-00 0000</td>
<td>-00 0000</td>
</tr>
<tr>
<td>13h</td>
<td>SSPBUF</td>
<td>Synchronous Serial Port Receive Buffer/Transmit Register</td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14h</td>
<td>SSPCON</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM3</td>
<td>SSPM2</td>
<td>SSPM1</td>
<td>SSPM0</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>15h</td>
<td>CCP1R1L</td>
<td>Capture/Compare/PWM Register1 (LSB)</td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16h</td>
<td>CCP1R1H</td>
<td>Capture/Compare/PWM Register1 (MSB)</td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17h</td>
<td>CCP1CON</td>
<td></td>
<td></td>
<td></td>
<td>CCP1X</td>
<td>CCP1Y</td>
<td>CCP1M3</td>
<td>CCP1M2</td>
<td>CCP1M1</td>
<td>CCP1M0</td>
<td>--0 0000</td>
</tr>
<tr>
<td>18h</td>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>0000 003x</td>
<td>0000 003x</td>
</tr>
<tr>
<td>19h</td>
<td>TXREG</td>
<td>USARF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>1Ah</td>
<td>RCReg</td>
<td>USARF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>1Bh</td>
<td>CCRP2L</td>
<td>Capture/Compare/PWM Register2 (LSB)</td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1Ch</td>
<td>CCRP2H</td>
<td>Capture/Compare/PWM Register2 (MSB)</td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1Dh</td>
<td>CCP2RCON</td>
<td></td>
<td></td>
<td></td>
<td>CCP2X</td>
<td>CCP2Y</td>
<td>CCP2M3</td>
<td>CCP2M2</td>
<td>CCP2M1</td>
<td>CCP2M0</td>
<td>--0 0000</td>
</tr>
<tr>
<td>1Eh</td>
<td>ADRF</td>
<td>A/D Result Register High Byte</td>
<td>xxxxx xxxx</td>
<td>uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1Fh</td>
<td>ADCON0</td>
<td>ADCS1</td>
<td>ADCS0</td>
<td>CHS2</td>
<td>CHS1</td>
<td>CHS0</td>
<td>GO/DONE</td>
<td></td>
<td>ADON</td>
<td>0000 00-0</td>
<td>0000 00-0</td>
</tr>
<tr>
<td>Mnemonic, Operands</td>
<td>Description</td>
<td>Cycles</td>
<td>14-Bit Opcode</td>
<td>Status Affected</td>
<td>Notes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>--------</td>
<td>---------------</td>
<td>----------------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDWF, f, d</td>
<td>Add W and f</td>
<td>1</td>
<td>00 0111 dfff ffff</td>
<td>C,DC,Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANDWF, f, d</td>
<td>AND W with f</td>
<td>1</td>
<td>00 0101 dfff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLRF, f</td>
<td>Clear f</td>
<td>1</td>
<td>00 0001 1fff ffff</td>
<td>Z</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLRWF -</td>
<td>Clear W</td>
<td>1</td>
<td>00 0001 0xxx xxxx</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMF, f, d</td>
<td>Complement f</td>
<td>1</td>
<td>00 1001 1fff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC, f, d</td>
<td>Decrement f</td>
<td>1</td>
<td>00 0011 dfff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFSZ, f, d</td>
<td>Decrement f, Skip if 0</td>
<td>1(2)</td>
<td>00 1011 dfff ffff</td>
<td>Z</td>
<td>1,2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INCF, f, d</td>
<td>Increment f</td>
<td>1</td>
<td>00 1010 dfff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INCFSZ, f, d</td>
<td>Increment f, Skip if 0</td>
<td>1(2)</td>
<td>00 1111 dfff ffff</td>
<td>Z</td>
<td>1,2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IORWF, f, d</td>
<td>Inclusive OR W with f</td>
<td>1</td>
<td>00 0100 dfff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVF, f, d</td>
<td>Move f</td>
<td>1</td>
<td>00 1000 dfff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVWF, f</td>
<td>Move W to f</td>
<td>1</td>
<td>00 0000 1fff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOP -</td>
<td>No Operation</td>
<td>1</td>
<td>00 0000 0xxx 0xxx 0xxx</td>
<td>C</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RLF, f, d</td>
<td>Rotate Left f through Carry</td>
<td>1</td>
<td>00 1101 dfff ffff</td>
<td>C</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RRF, f, d</td>
<td>Rotate Right f through Carry</td>
<td>1</td>
<td>00 1100 dfff ffff</td>
<td>C</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBWF, f, d</td>
<td>Subtract W from f</td>
<td>1</td>
<td>00 0100 dfff ffff</td>
<td>C</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWAPF, f, d</td>
<td>Swap nibbles in f</td>
<td>1</td>
<td>00 1110 dfff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XORWF, f, d</td>
<td>Exclusive OR W with f</td>
<td>1</td>
<td>00 0110 dfff ffff</td>
<td>Z</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### BIT-ORIENTED FILE REGISTER OPERATIONS

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>14-Bit Opcode</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCF, f, b</td>
<td>Bit Clear f</td>
<td>1</td>
<td>00 00bb bfff ffff</td>
<td></td>
<td>1,2</td>
</tr>
<tr>
<td>BSF, f, b</td>
<td>Bit Set f</td>
<td>1</td>
<td>01 01bb bfff ffff</td>
<td></td>
<td>1,2</td>
</tr>
<tr>
<td>BTFSC, f, b</td>
<td>Bit Test f, Skip if Clear</td>
<td>1 (2)</td>
<td>01 10bb bfff ffff</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>BTFSS, f, b</td>
<td>Bit Test f, Skip if Set</td>
<td>1 (2)</td>
<td>01 11bb bfff ffff</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

### LITERAL AND CONTROL OPERATIONS

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>14-Bit Opcode</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDLW, k</td>
<td>Add literal and W</td>
<td>1</td>
<td>11 111x kkkk kkkk</td>
<td>C,DC,Z</td>
<td>Z</td>
</tr>
<tr>
<td>ANDLW, k</td>
<td>AND literal with W</td>
<td>1</td>
<td>11 1001 kkkk kkkk</td>
<td>C,DC,Z</td>
<td>Z</td>
</tr>
<tr>
<td>CALL, k</td>
<td>Call subroutine</td>
<td>1</td>
<td>11 0000 0110 0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLRWDT -</td>
<td>Clear Watchdog Timer</td>
<td>1</td>
<td>10 0kkk kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOTO, k</td>
<td>Go to address</td>
<td>1</td>
<td>10 1kkk kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IORLW, k</td>
<td>Inclusive OR literal with W</td>
<td>1</td>
<td>11 1000 kkkk kkkk</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>MOVLW, k</td>
<td>Move literal to W</td>
<td>1</td>
<td>11 00xx kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETFIE -</td>
<td>Return from interrupt</td>
<td>2</td>
<td>00 0000 0000 1001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETLW, k</td>
<td>Return with literal in W</td>
<td>2</td>
<td>11 01xx kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETURN -</td>
<td>Return from Subroutine</td>
<td>2</td>
<td>00 0000 0000 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLEEP -</td>
<td>Go into standby mode</td>
<td>1</td>
<td>00 0000 0110 0011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBLW, k</td>
<td>Subtract W from literal</td>
<td>1</td>
<td>11 110x kkkk kkkk</td>
<td>C,DC,Z</td>
<td>Z</td>
</tr>
<tr>
<td>XORLW, k</td>
<td>Exclusive OR literal with W</td>
<td>1</td>
<td>11 1010 kkkk kkkk</td>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>
Real Example:

```
:********** Four Ways Traffic Lights Control
: PORTA, PORTB, PORTC, PORTD and PORTE are Outputs
: RA0 - First Walker
: RA1 - Second Walker
: RA2 - Third Walker
: RA3 - Fourth Walker
: RA5 - Fifth Walker
: RB0 - First Red
: RB1 - First Yellow
: RB2 - First Green
: RB5 - Second Red
: RB6 - Second Yellow
: RB7 - Main
RC0 - BSF PORTD, 3 ; First sequence
RC1 - BSF PORTB, 2
RC2 - BCF PORTB, 0
RC3 - BSF PORTA, 3
RC4 - BCF PORTA, 1
RC5 - BCF PORTA, 5
RD0 - BCF PORTE, 0
RD1 - BCF PORTE, 1
RD2 - MOVWL 0x04
RD3 - MOVWF FACTOR
RD4 - L1 WAITX h'ff', b'11000111'
RD5 - DECFSZ FACTOR, 1
RE0 - GOTO L1
RE1 - MOVWL 0x05
RE2 - MOVWF FLASH
Loop1
BCF PORTB, 2
WAITX h'5', b'11000111'
BSF PORTB, 2
WAITX h'5', b'11000111'
DECFSZ FLASH, 1
GOTO Loop1
BCF PORTB, 2
BCF PORTB, 5
BSF PORTB, 1
BSF PORTB, 6
WAITX h'20', b'11000111'
BCF PORTB, 1
BCF PORTB, 6
```

Figure 106: Four Ways Traffic Light Control
Project #2: Technique to Calculate Day of Week

Basically, there are two kinds of electronic systems that come with a built-in calendar. The first kind of system is used mainly to display a calendar for a user’s convenience. Examples of these systems are digital watches, computers, VCRs and TVs with on-screen display features. The second kind of system is required to know whether a given date is a weekend or weekday. Examples of such a system are multi-rate meters (such as a phone bill meter) and the weekday rate are different. In design, the designer needs to write a piece of code (Monday, ….Saturday) of the week based on the information of the basic component of an electronics system to find the exact day for a given date.

The day of week is a straightforward algorithm. For example, the year 1989 as a reference point. The year 1989 was a leap year because it was on the last day of the month, and the number of days in the year was 365. It makes it easy to calculate the number of days after this date was Monday, and so on.

```
YearHi   equ 10h   ; Store the Upper byte of Year
YearLw   equ 11h   ; Store the Lower byte of year
Month    equ 12h   ; Store the Month (1 for January, 2 for February and so on)
Day      equ 13h   ; Store the Day
AcoValue equ 14h
TempA    equ 15h
TempB    equ 16h
TempYearHi equ 15h
TempYearLw equ 16h
Lbyte    equ 17h
Hbyte    equ 18h
Ltemp    equ 19h
Htemp    equ 1Ah
Temp     equ 1Bh

org 0x00

; Test program for GetDayOfWeek. The End Result will be stored in AccValue
main
movlw 19h ; Set Date as 21 September 1998
movwf TempA
movlw 98h
movwf TempB
movlw 9
movwf Month
movlw D21'
movwf Day

call BCDtoBin ; Convert the 4-digit BCD Year to 16bit int value
movfw Lbyte
movwf YearLw
movfw Hbyte
movwf YearHi
```
Program #12: Keypad Interfacing

![Keypad Image]

Figure 104: 4 x 3 Keypad

The program will use a scan algorithm from PORTB and sends the number to PORTA. It uses a PIC16F84 running at 4 MHz.

- pad Pin 1, Row 1
- pad Pin 2, Row 2
- pad Pin 3, Row 3
- pad Pin 4, Row 4
- pad Pin 5, Col 1
- pad Pin 6, Col 2
- pad Pin 7, Col 3

```asm
; PIC16F84A Runs at 4 MHz
WDT_OFF & _XT_OSC & _PWRT_ON
; Note that the WatchDog Timer is OFF!
```
Discussion ??????

Let’s summarize the differences between MPU-based EMS and MCU- based EMS Together..
DSP-based Embedded Systems
• **Digital Signal Processing (DSP)** is the arithmetic processing of discrete-time signals.

• A signal is a physical quantity that varies with time, frequency or space.

• Instead of using op-amps and other analog electronics to process an analog signal, DSP processor can be used to perform **mathematical operations on digital signals** to achieve the same (or better) effect.

• A/D is needed for analog signals.

• DSP chips are specialized microprocessors (programmable devices with its own instruction code). **With architectures designed to reduce the number of instructions and operations necessary for efficient signal processing.**
DSP chips are used to perform computationally efficient and fast algorithms, such as:

- Digital filtering
- Spectral analysis
- Parameter estimation
- Data compression

DSP chips are available by:

- Texas Instruments
- Motorola
### DSP Architecture Features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-cycle instructions</td>
<td>Executes advanced control systems in real-time</td>
</tr>
<tr>
<td>Pipelined architecture</td>
<td>Controls high-bandwidth systems</td>
</tr>
<tr>
<td>Harvard architecture</td>
<td>Access data and instructions simultaneously and therefore increase speed</td>
</tr>
<tr>
<td>Hardware multiplier</td>
<td>Minimizes computational delays</td>
</tr>
<tr>
<td>Hardware shifter</td>
<td>Have large dynamic range</td>
</tr>
<tr>
<td>Hardware stack</td>
<td>Support fast interrupt processing</td>
</tr>
<tr>
<td>Multi-bus pipeline</td>
<td>Enables the controller to read and write data values in a single cycle, compared to traditional MCUs which can only work with a single data value at a time.</td>
</tr>
</tbody>
</table>
**DSP Architecture Advantages:**

- **Common features for DSP:**
  - Use a lot of mathematics (multiplying and adding signals)
  - Deal with signals that come from the real world
  - Requires a response in a certain time. (signals are measured from real world and usually need a reaction in real time).

- **Application Features of DSP:**
  - High speed processing applications such as real-time control
  - Accurate and complex control systems
  - Frequency based applications.
  - Iterative algorithms
  - Matrix operations.
A Case Study: TMS320C6201 Fixed Point DSP

Architecture / Features:

- High-Performance Fixed-Point Digital Signal Processor (DSP) TMS320C6201
  - 5-ns Instruction Cycle Time
  - 200-MHz Clock Rate
  - Eight 32-Bit Instructions/Cycle
  - 1600 MIPS
- VelociTI Advanced Very Long Instruction Word (VLIW) TMS320C62x DSP CPU Core
  - Eight Independent Functional Units:
    - Six ALUs (32-/40-Bit)
    - Two 16-Bit Multipliers (32-Bit Results)
  - Load-Store Architecture With 32 32-Bit General-Purpose Registers
  - Instruction Packing Reduces Code Size
  - All Instructions Conditional

Has the problem of quantization error
1M-Bit On-Chip SRAM
- 512K-Bit Internal Program/Cache (16K 32-Bit Instructions)
- 512K-Bit Dual-Access Internal Data
(64K Bytes) Organized as Two Blocks for Improved Concurrency

32-Bit External Memory Interface (EMIF)

Four-Channel Bootloading Direct-Memory-Access (DMA) Controller with an Auxiliary Channel

Two Multichannel Buffered Serial Ports (McBSPs)
Two 32-Bit General-Purpose Timers
Flexible Phase-Locked Loop (PLL) Clock Generator

IEEE-1149.1 (JTAG†) Boundary-Scan Compatible
352-Pin BGA Package (GJC Suffix)
352-Pin BGA Package (GJL Suffix)
CMOS Technology
- 0.18-\(\mu\)m/5-Level Metal Process
3.3-V I/Os, 1.8-V Internal
DSP Applications:

- Industrial Drives
- Appliances
- Optical Networking
- Power management
- Automotive
- Fuel Pumps
- Intelligent sensors
- Video and Mobile.
Recommendations:

• The first step in Embedded system design is to distinguish between the hardware architectures, differences must be clear so when the task of the EMS is specified the best HW is chosen according to the requirements of the system. (optimization).

• Data sheets are the confident resource to study the architecture and features of any electronic device.

• EMS designer must have the ability for self learning / hardworking.

• Development Kits are available for educational purposes.

• Detailed real applications can be studied in the next presentations
The End

Hind Salem