**- A Microcontroller** is a “computer on a chip”—that is a computer whose functional modules are all contained within a single integrated circuit (IC).

- An **embedded system** is simply a system that is controlled by one or more computers that are internal to the system.

- **Microchip:** PICs, **Intel:** 8051, **Freescale:** 68HC9, 68HC11, and 68HC12, **Atmel:** AVR, **Arm Limited, Infineon:** xx166, **Zilog:** ZNEO, **ST Micro**, **NXP**, **Texas Instrument:** MSP430 – low power family.

**Characteristics:** Real-Time Exe, Physical Size and environment, Power consumption, UI, Memory needs, Cost, Hardware vs Software, Multi-rate Op (parallel tasks), Registers, busses, reserved addresses and port addresses.

- **PIC32:** Microchip Technology was founded in **1987** by General Instruments and became its own company in 1989. Instruction sets can be 12, 14, 16, or 24 bits long. **32-bit** processor, 1 instruction per clock cycle up to 80MIPS, up to 256kB FLASH. **PIC32MX150F128D** in lab

- **68HC12:** Introduced in **the mid 1990’s**, the **68HC12** is **a 16-bit** microcontroller with 16-bit addresses. 8MHz bus and up to 128kB FLASH. has **two 8-bit accumulators A and B** which can be cascaded to allow for operations involving 16 bits, two 16-bit registers X and Y, a 16-bit program counter, a 16-bit stack pointer and an 8-bit status (condition code) register.

- A **cross compiler** is a compiler which creates executable code for a platform (processor) other than the one on which the compiler runs. -> assembler -> Linker

- **S19 file format: Motorola (Big endian) -** Start code: S, Record (data) type 1 digit 0-9, Byte count 2 hex digits num of bytes, Address 4, 6, or 8 hex digits for first data byte, Data 2n bytes for n bytes, Checksum 1’s compliment of least significant byte of sum of values of hex digit pairs for byte count, address and data.

- **Hex file: Microchip –** “:” Start of line, BB number of bytes on line, AAAA address (in bytes), TT record type where 00 – data, 01 EOF, 04 is extended address, DD… data bytes depends on value of BB, CC checksum **2’s** compliment of bytes + address + data.

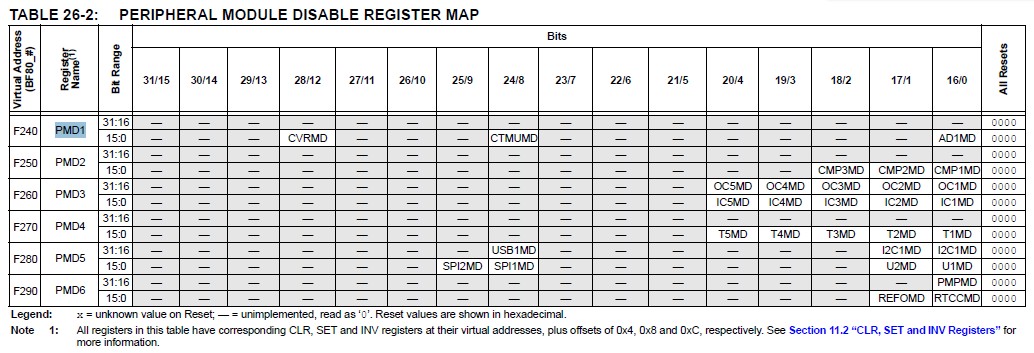
- Header file for MC9S12: **MCS9S12dt256.h**

**-** Header file for PIC32: **plib.h**

**-** JTAG, “Joint Test Action Group”, created the industry IEEE Standard 1149.1-1990 entitled *Standard Test Access Port and Boundary-Scan Architecture* but usually referred to by its creators’ name, JTAG.

- **Breadboard components:** LCD module 32 char (2x16), Keypad (hex or 4x4), Potentiometers, LEDs, DIP switches, Pushbutton switches, Opto-isolators, Relays, 7Seg display, Buzzer, Op amp/ comparator, Expansion ports, DACs, Triacs and SCRs, Temp sensors, Heat source, LED bar graphs, CAN ports, Solid state relays.

- **Mode of operation: Normal** – power can be saved by reducing clock frequency using internal clock. **Idle –** CPU halted but SYSCLK and peripherals still operate. **Sleep -** CPU halted but peripherals can wake CPU.

- **PIC32 memory:**  **RAM** 4kB - 64kB, **FLASH EEPROM** 16kB – 256kB

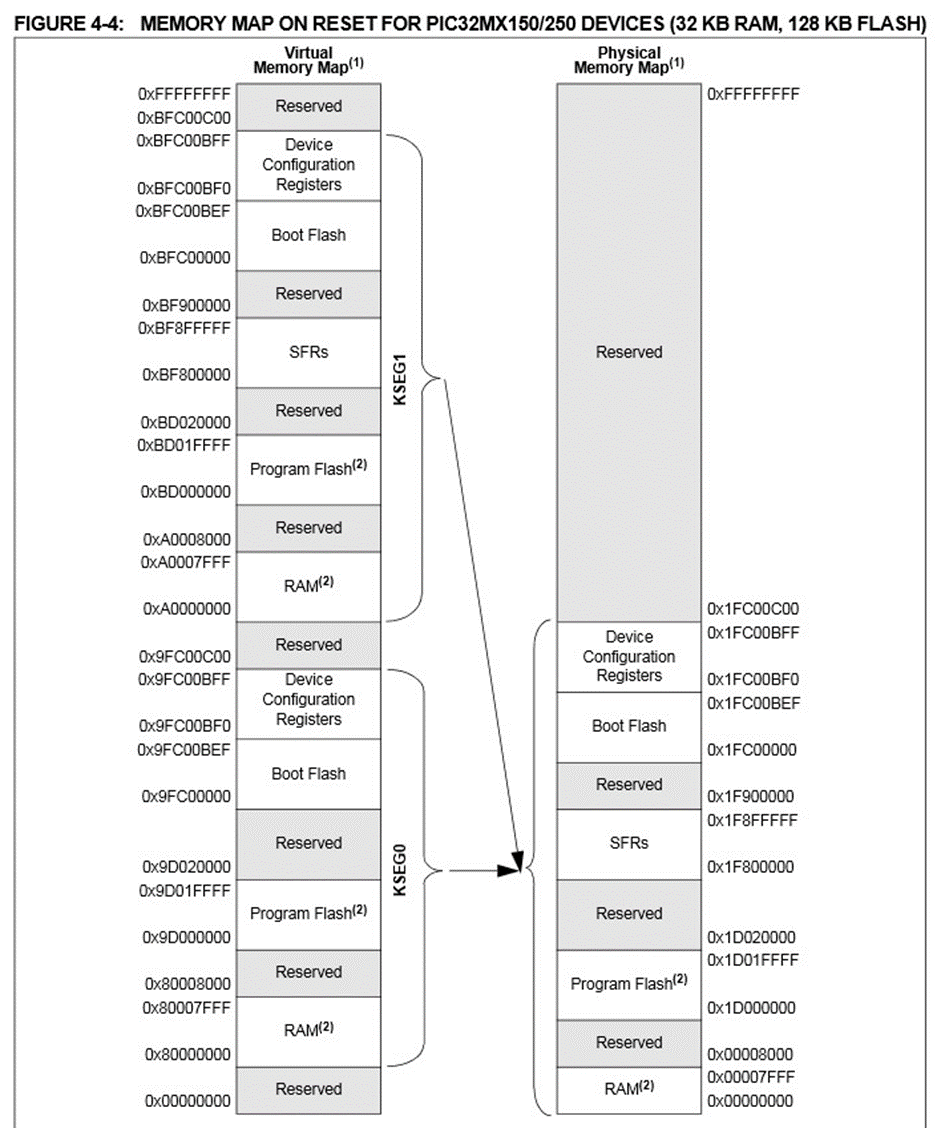
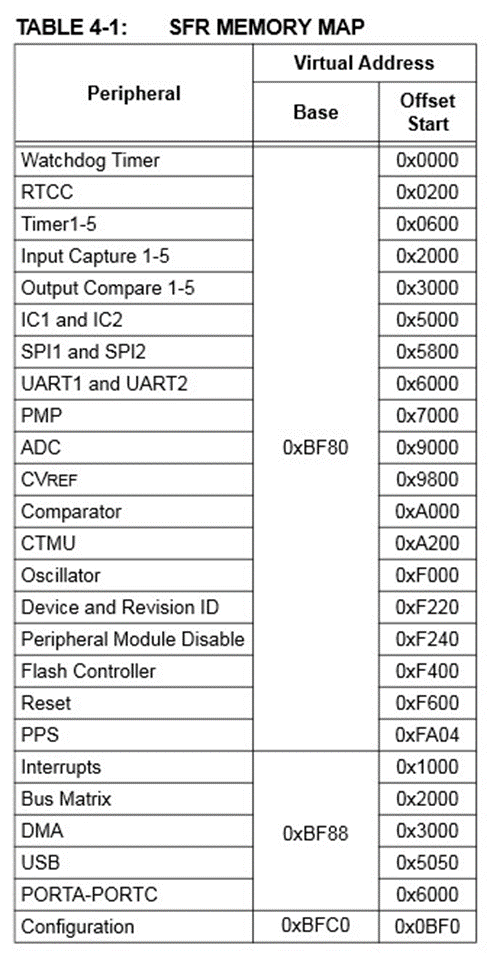
- CFGCONbits.PMDLOCK = 0; to set peripherals

- The ROMCTL signal allows the setting of the ROMON bit in the MISC register thus controlling whether the internal Flash is visible in the memory map. ROMON = 1 means the Flash is visible in the memory map. The state of the ROMCTL pin is latched into the ROMON bit in the MISC register on the rising edge of the reset signal.

- **Single chip mode: C = 1, B = 0, A = 0:** no external addresses/data bus

- **Expanded wide mode: C = 1, B = 1, A = 1:** 16 bit address and data. PortA takes high bits, PortB takes low

- **Expanded Narrow mode: C = 1, B = 0, A = 1:** 16 bit address, 8 bit data. Address split like above and data all on B.



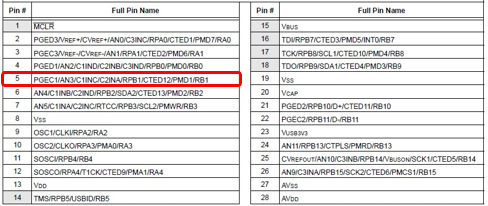
- **Volatile -** writeablility and speed pros. **SRAM and DRAM** used for variable, stack and some code

- **Non Volatile -** Data retention and cost pros. **Masked ROM, EEPROM, OTP, FLASH** program code, constants (tables, images.), system settings, last state.

- **SRAM** – lower power at rest, faster (expensive), simpler, registers, clock, cache, real time clock. Main mem in PIC32

- **DRAM –** lower power when active, slower, ¼ the size, much cheaper, main memory in PC or MC if need lots.

**28 pin not QFN**

- **Masked ROM –** final stage of production mass prod, cheap.

- **PROM –** Programmable Read Only Memory – reprogram once.

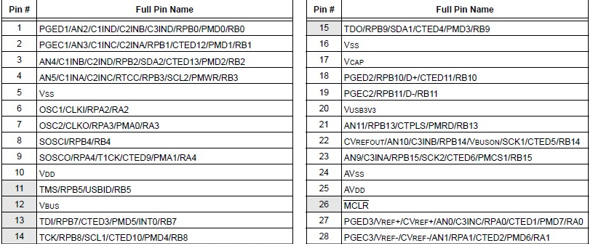
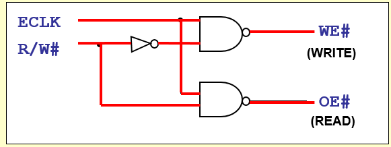
- **EPROM –** Erasable Programmable Read Only Memory, UV erases few times to reprogram

- **EEPROM** – erased in circuit one byte at a time.

- **FLASH -** EEPROM block erasable.

- **NVRAM –** SRAM with battery for backup on power down

**28 pin QFN**

- **EEPROM** reprogram 10 times before giving up

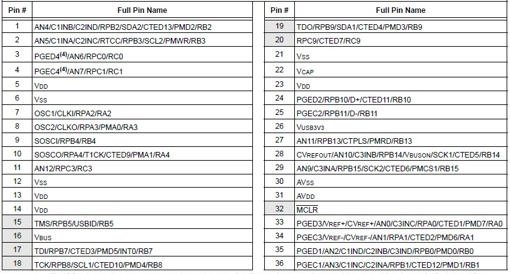
- Glue logic for MC9S12

- **PIC32** all 1s are critical

0’s are assumed non critical until

Found to be critical.

**36 pin VTLA**

- **Port B is the lower address (Odd)**

- **Port A is higher (even)**

- **ELCK** triggers latching

------------------------------------

- **Big Endian:**

- LSTRB = odd addresses

- A0 = even addresses

- Chip 0 gets high order data and A0

- Chip 1 gets low order data and LSTRB

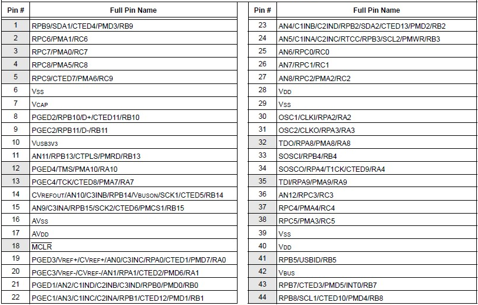
- **Little Endian:**

- Chip 0 gets high and LSTRB

- Chip 1 gets low and A0

- **Also can flip order of high/low to flip endianness.**

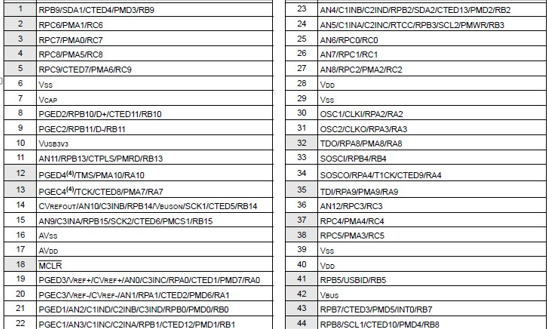
**44 pin QFN**

- IO is parallel by default because its byte addressable. For serial use shift registers and then do byte wise.

- **Isolated I/O** – additional pins needed for address, etc. I/O bus smaller than memory. Additional I/O instructions needed to access I/O bus. IOW, IOR (IO read/write). Simultaneous I/O and memory and I/O can be slower than memory.

- **Memory mapper I/O** – memory address range assigned to IO. Same memory instruction used for I/O MEMR and MEMW. No additional pins required, I/O gets richer instruction sets and addressing modes mem has to wait for IO and less mem can be mapped.

**44 pin TQFP**

- #define LATA \*(volatile unsigned int \*) 0xBF886030

- | or SET to set. & or CLR to clear

- **PPS: IN** RPB4 (0010) to INT4

INT4bits.INT4R = 2; or INT4R = 2

- **PPS: OUT** U1TX (1) to RPC7 RPC7Rbits.RPC7R = 1; or RPC7R = 1;

- CFGCONbits.IOLOCK = 0; unlock

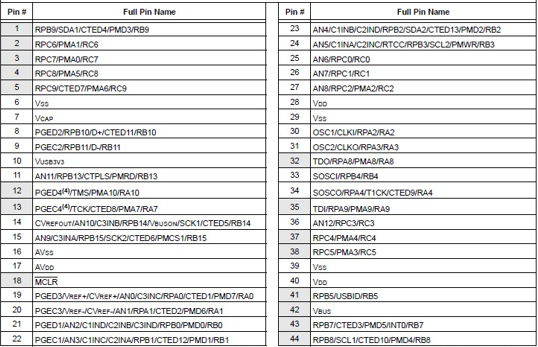
- CFGCONbits.IOLOCK = 1; lock

-

-

-

**44 pin VTLA**

- **Registers:**

- **ANSELx**: 1 – analog, 0 - digital

- **TRISx**: 1 – input, 0 - output

- read from **PORTx**

- write to **LATx**

- **ODCx**: 1 – open drain, 0 – push-pull

- **CNCONx**: 1 – on, 0 - off

- **CNPUx**: 1 – pull up, 0 – no pull up

- **CNPDx**: 1- pull down, 0 – disabled

- **CNENx**: 1 – interrupt enable, 0 - off

- **CNSTATx**: Read 1 - change, 0 - no

- **SET, CLR and INV** used like PORTBSET = 0x0001; sets bit 0

- enum{x, y, z } means x = 0, y = 1,

z = 2

-

# delays = ((sections \* time \* frequency)/(clock ticks/ instruction)) – 1 : round up

- **VIH –** Minimum voltage at logic high.

- **VIL –** Maximum voltage at logic low.

- **VOH** – Minimum voltage at logic high.

- **VOL –** Maximum voltage at logic low.

- **VIH ≤ VOH**

- **VIL ≥ VOL**

- **IIH –** Current at logic high

- **IIL –** Current at logic low

- **IOH –** Maximum current at logic high (VOH)

- **IOL –** Maximum current at logic low (VOL )

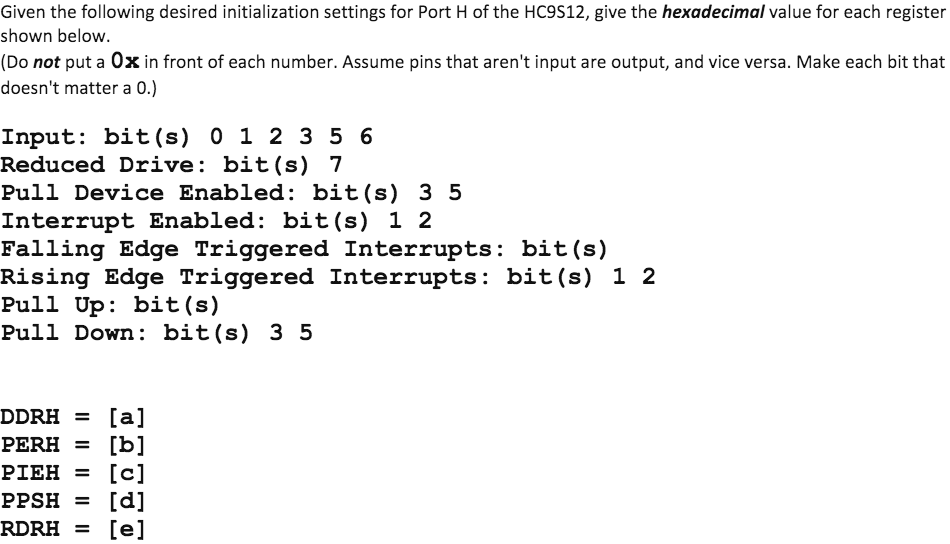
- **IIH ≤ IOH**

- **IIL ≤ IOL**

- **FanoutH (logic high) = IOH / IIH.**

- **FanoutL (logic low) = IOL / IIL.**

- **Fanout = min[FanoutL, FanoutH].**



- **DDRH**=Either output or inverted input bits

- **PERH**=Pull Device Enabled Bits

**- PIEH**=Interrupt Enabled Bits

- **PPSH**=Rising Edge Triggered Interrupt bits and pull down bits

- **RDRH**=Reduced Drive Bits

- Switches: normally opened/closed push button switch

SPDT – single pole double throw also SPST etc.

- Which of the following is/are not found on the MC9S12? **Piezo Electric Buzzer, Digital to Analog Converter, Temperature Sensor**

- What type of data/information is stored at the upper end of the MC9S12 memory map? **Interrupt Vector Table**

- The [a] memory in the chip is used to hold code instructions for the chip. **Flash**

**-** Timing diagram, at what point in time does the EPROM programmer read the DATA to be verified from the bus? **The second rising edge of CE#**

- Timing diagram, how long must the DATA be stable before the EPROM latches the data into memory? **tDS**

- Timing diagram, how long must the ADDRESS be stable before the EPROM latches data into memory? **tAS**

- Timing diagram, what signal tells the EPROM to output the cell data which was written to it? **CE#**

- Timing diagram, at what point in time does the EPROM read the DATA it is to program into its cell? **The first rising edge of CE#**

- Timing diagram, how long must the programmer leave the ADDRESS on the bus after they have read the data to be verified? **tAH**

**-** Timing diagram, what signal determines when the EPROM reads the data to be programmed into the cell? **CE#**

- What signal determines whether input data or output data is on the databus? **OE**

**-**  Give an advantage and disadvantage of using “linear select addressing of I/O. **Pro - Simple selection logic and one port per address-line; Con – Wasteful Addressing**

-This register can be used to detect overloads or shorts at port pins. **PTln**

-This register can be used to internally connect a resistor to form a port pin to either Vcc or Vdd if the corresponding pin is set to input. **PERn**

- In order to know which pin an interrupt has occurred on, this register must be read. **PIFn**

- A [a] resistor is generally used to make an input pin’s default value 1. **Pull up**

- This register can determine whether the pull resistor is “pull-up” or “pull-down”. **PPSn**

- This register can determine whether the interrupt is detected on either the rising edge or falling edge of a signal. **PPSn**

- Which register is written to in order to change the values of a certain port’s pins? **PTn**

- What Port only has 4 pins instead of 8? **J**

- Which register needs to be configured before ever outputting a value to a certain port pin? **DDRn**

- This register determines whether or not a transition on a port pin will produce an interrupt. **PIEn**

- The ISA level directly supports serial instructions. **False**

- Which register determines if port pins will be used as input or output? **DDRn**

- Which register determines if port pins will act as an OR gate when its pins are tied together, or not? **WOMn**

- This register reads the state of the register buffer if reading a pin is set to output. **PTn**

- This register tells whether or not an interrupt has occurred a specific pin of a port. **PIFn**

- This register can be used to limit the amount of current supplied to a circuit connected to the corresponding port. **RDRn**

- This Register ignores data written to it if its pins are set up for input. **PTn**