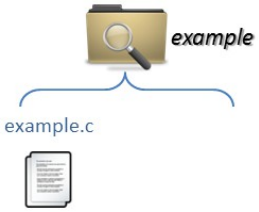
1. CREATION DE PROJET

*Nous allons développer nos programmes sous MPLABX, qui est un environnement de développement ou IDE proposé par Microchip pour toute sa gamme de MCU. Il faut savoir que MPLAB n'est pas le seul IDE existant pour travailler sur MCU Microchip (MikroC ...). Afin de générer des fichiers binaires de sortie, MPLABX doit être associé à un compilateur. Celui utilisé en TP est C32, proposé également par Microchip pour toute sa gamme de MCU 32bits PIC32MX. MPLABX et C32 (ou XC32) sont des logiciels libres et sont librement téléchargeables sur le site de Microchip (**http://www.microchip.com/**). Vous pouvez d'ailleurs développer vos codes chez vous du moment que vous restez au stade de la compilation.*

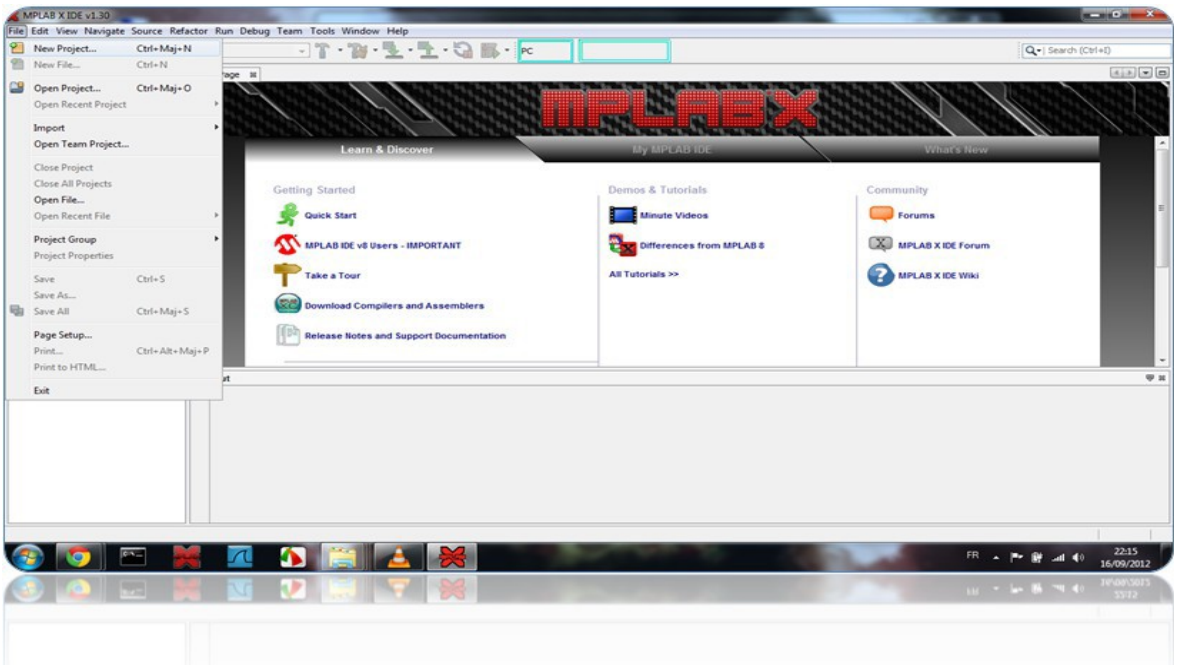
- *Durant les enseignements de Systèmes Embarqués, vous pouvez utiliser librement le répertoire **Z :**.*

- *Afin d'optimiser les temps de compilation, il peut néanmoins être intéressant de travailler localement dans le répertoire **Documents**. Penser à ré-importer vos répertoires de travail sous Z : en fin de séance.*

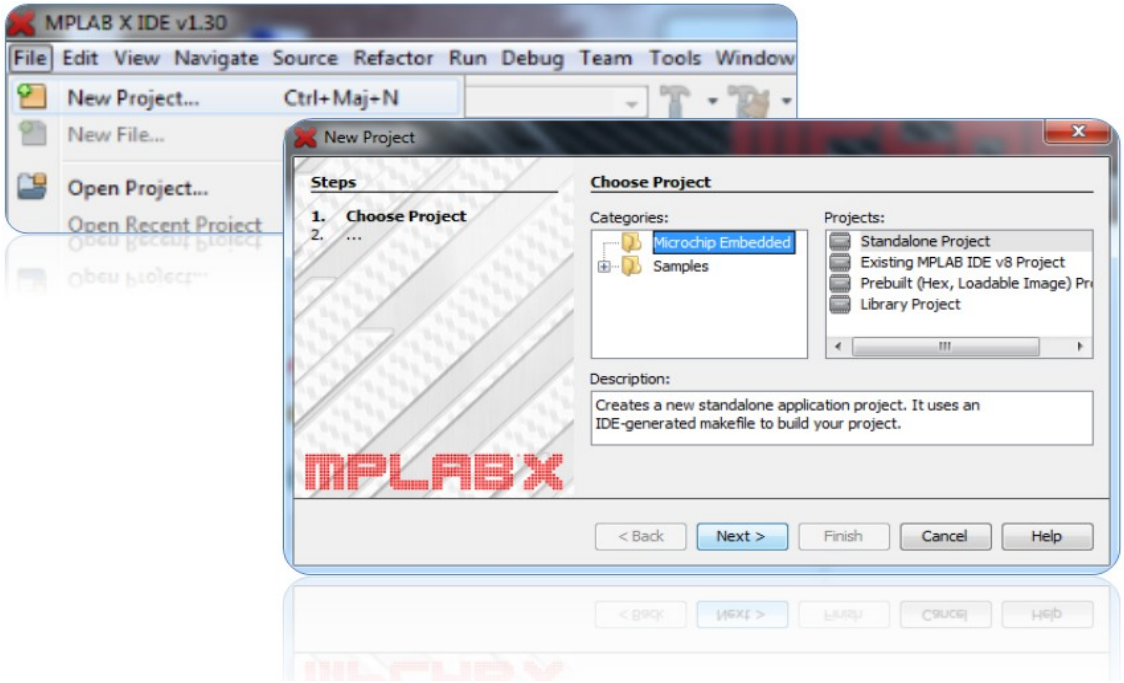
- *Créer un répertoire au nom de votre projet et importer les sources élèves déjà pré-remplis. Dans les captures d'écrans suivantes, le répertoire du projet se nomme **example** ainsi que le fichier source principal.*



- *Ouvrir MPLABX.*

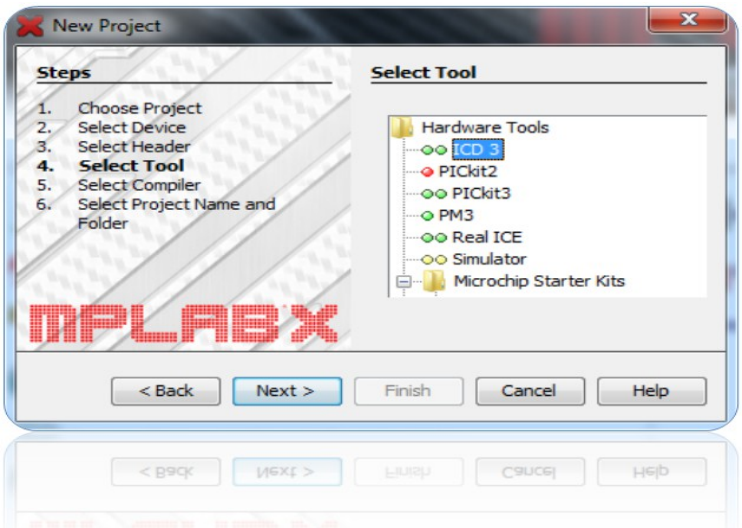


- *Créer un nouveau projet.* ***File >> New Project >> Next***

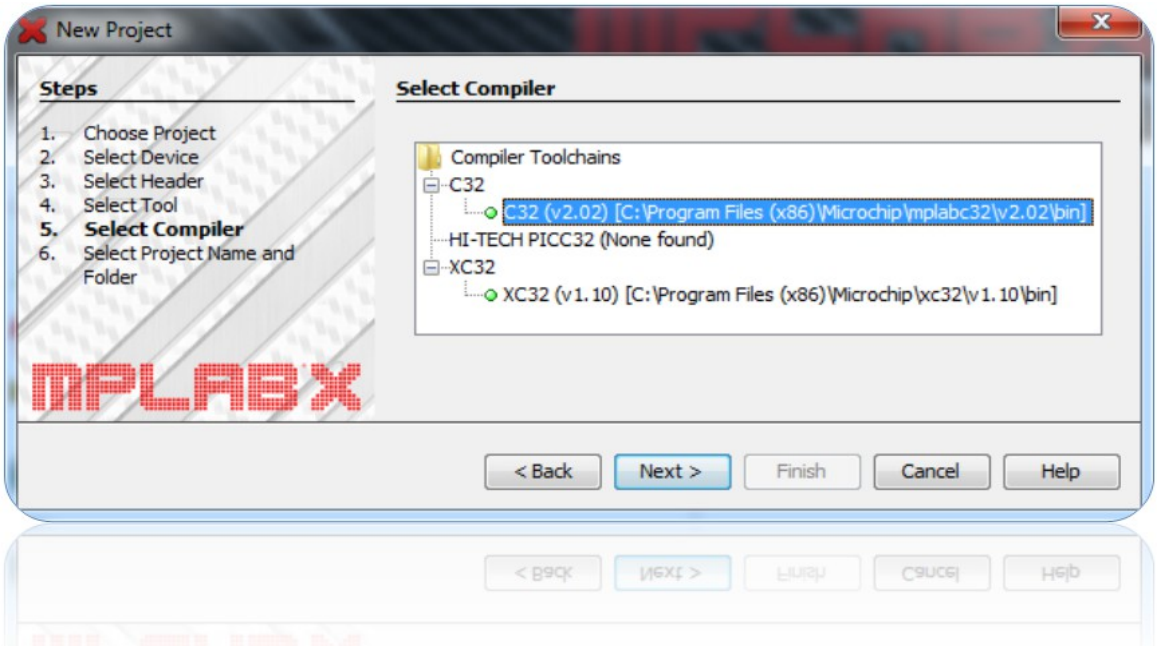


- *Sélectionner le MCU cible. A adapter en fonction de vos projets :*

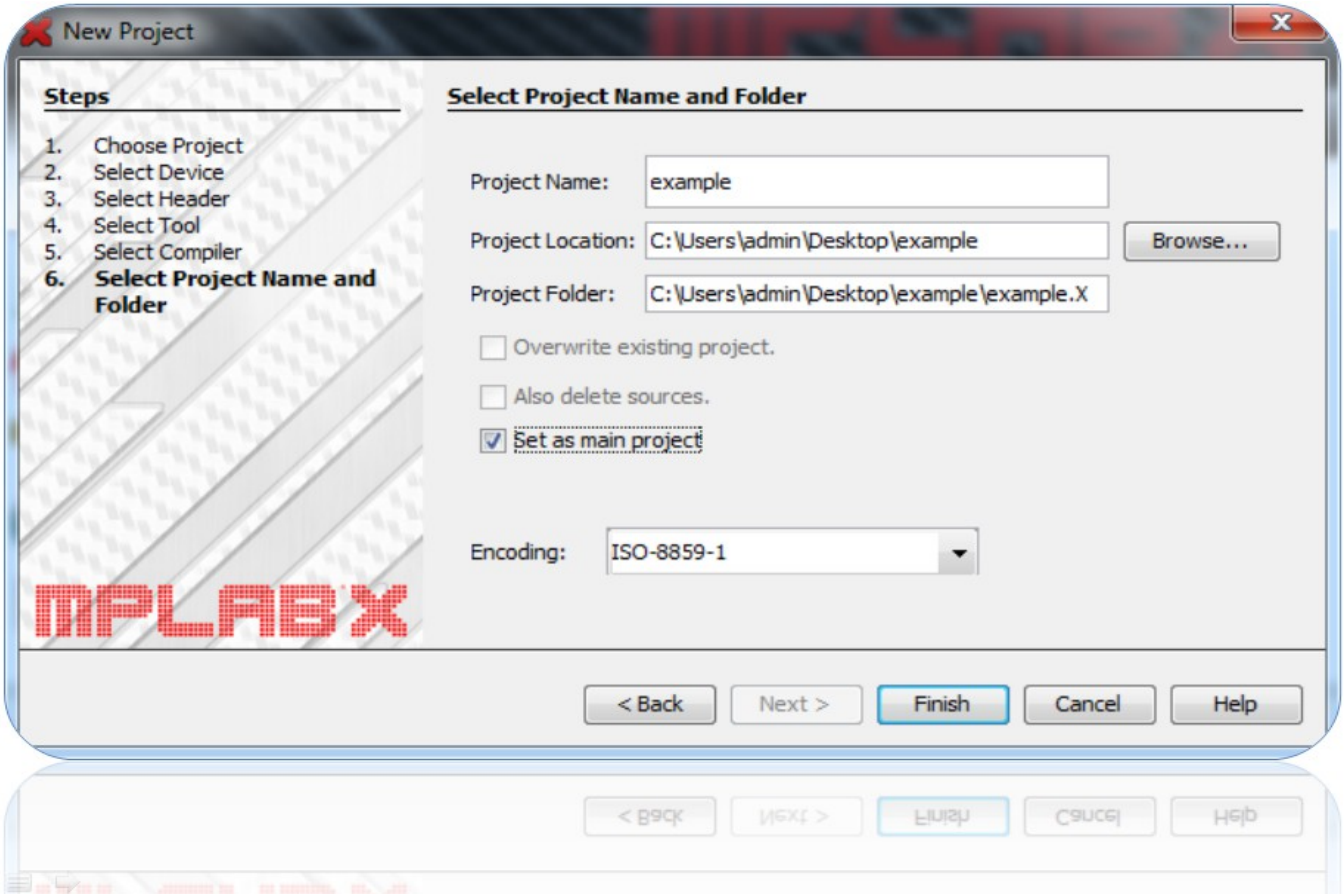
- *Sélectionner la sonde de programmation utilisée :*



- *Sélectionner la chaîne de compilation appelée par l'IDE :*

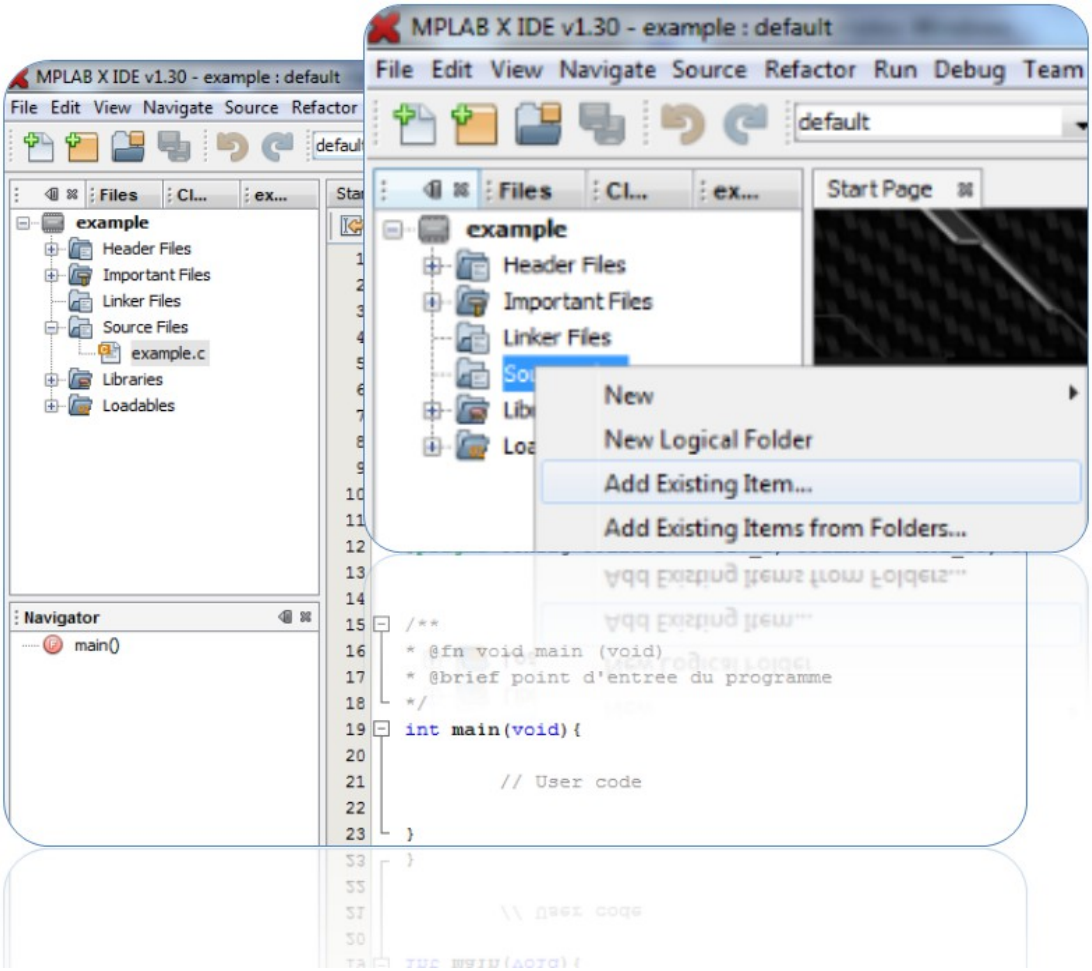


- *Donner un nom au projet. Attention au répertoire de travail du projet. Vous n'avez accès qu'au répertoire* **Z :** *sur le réseau ou au répertoire* ***Documents*** *localement sur la machine.*



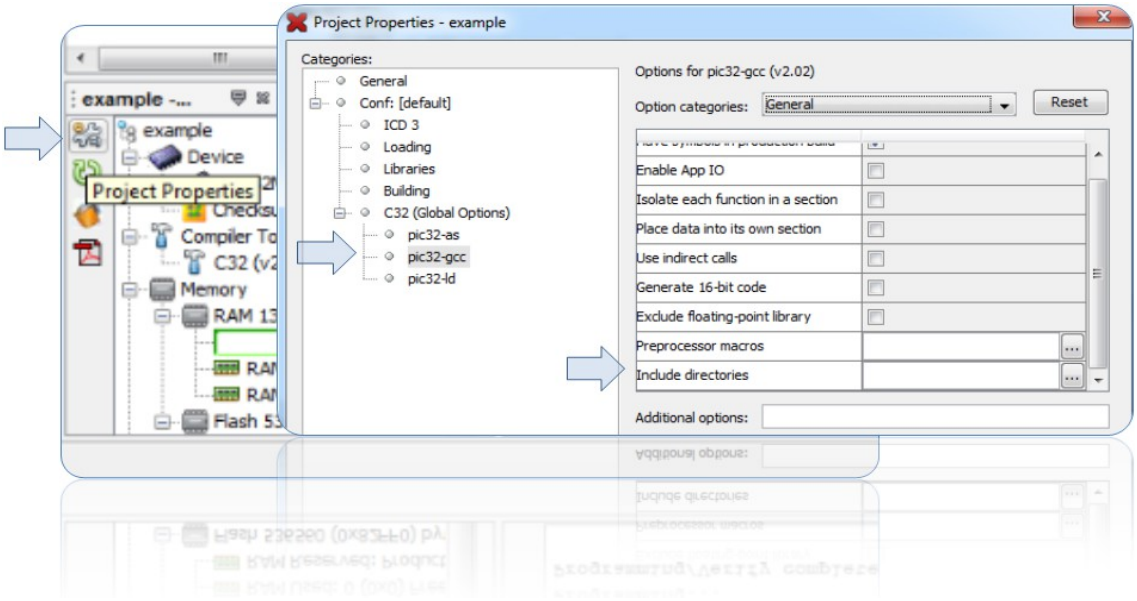
- *Cliquer sur* ***Finish****, votre projet est maintenant créé.*

- *Ajouter maintenant les fichiers sources au projet :* ***Clic droit sur Sources Files >> Add Existing Item ...***



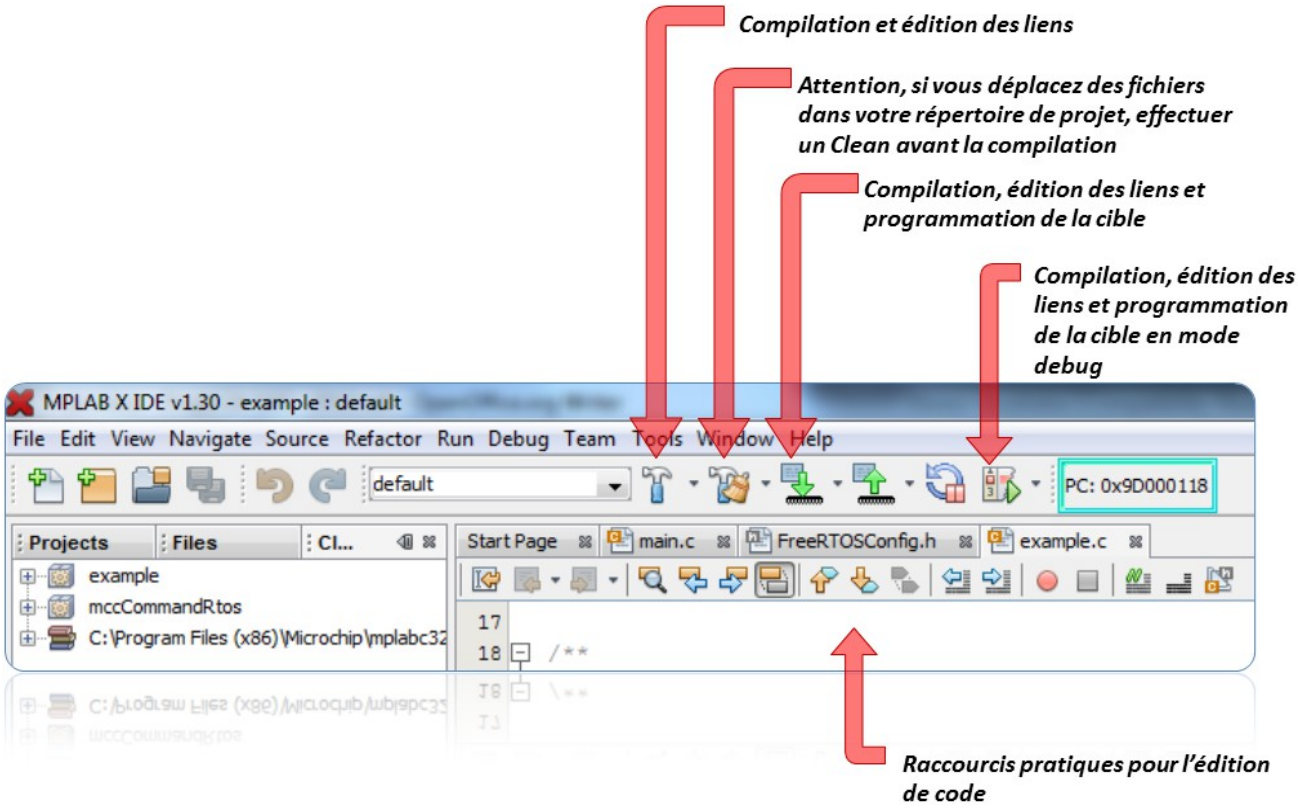
- *Votre projet est maintenant prêt à être compilé.*

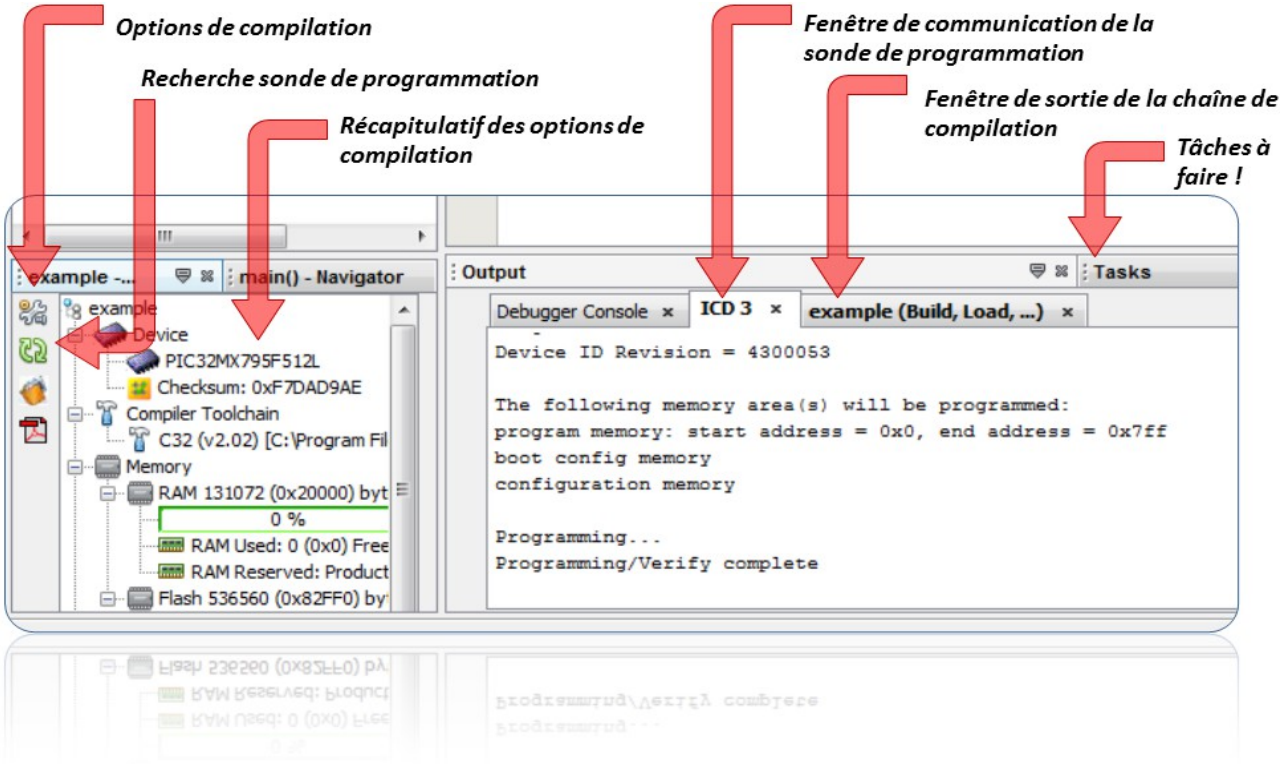
- *Si nécessaire, n'oubliez pas de configurer les chemins ou path de la chaîne de compilation si votre projet est dépend de fichiers d'en-tête :*

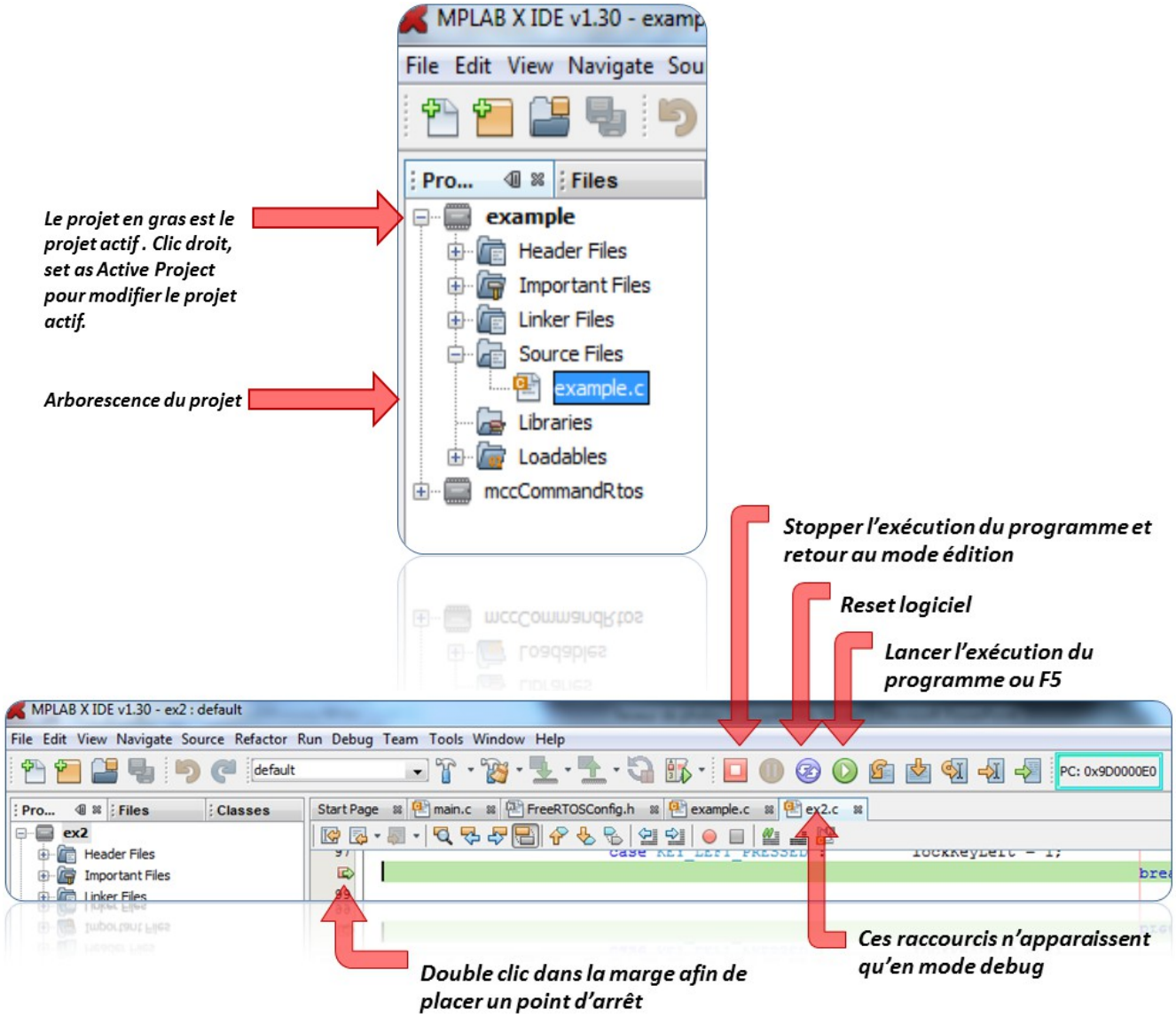


2. IDE MPLABX

Présentation de quelques truc et astuces à savoir sur MPLABX et la chaîne de compilation :







- *Taille des types de variables sous la chaîne de compilation C32 :*

Type	Bits	Min	Max
char, signed char	8	-128	127
unsigned char	8	0	255
short, signed short	16	-32768	32767
unsigned short	16	0	65535
int, signed int, long, signed long	32	-2^{31}	$2^{31}-1$
unsigned int, unsigned long	32	0	$2^{32}-1$
long long, signed long long	64	-2^{63}	$2^{63}-1$
unsigned long long	64	0	$2^{64}-1$

Type	Bits
float	32
double	64
long double	64

3. DATASHEET PIC18F4550

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-chip USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μA typical
- Sleep mode currents down to 0.1 μA typical
- Timer1 oscillator: 1.1 μA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal oscillator block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Dual oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

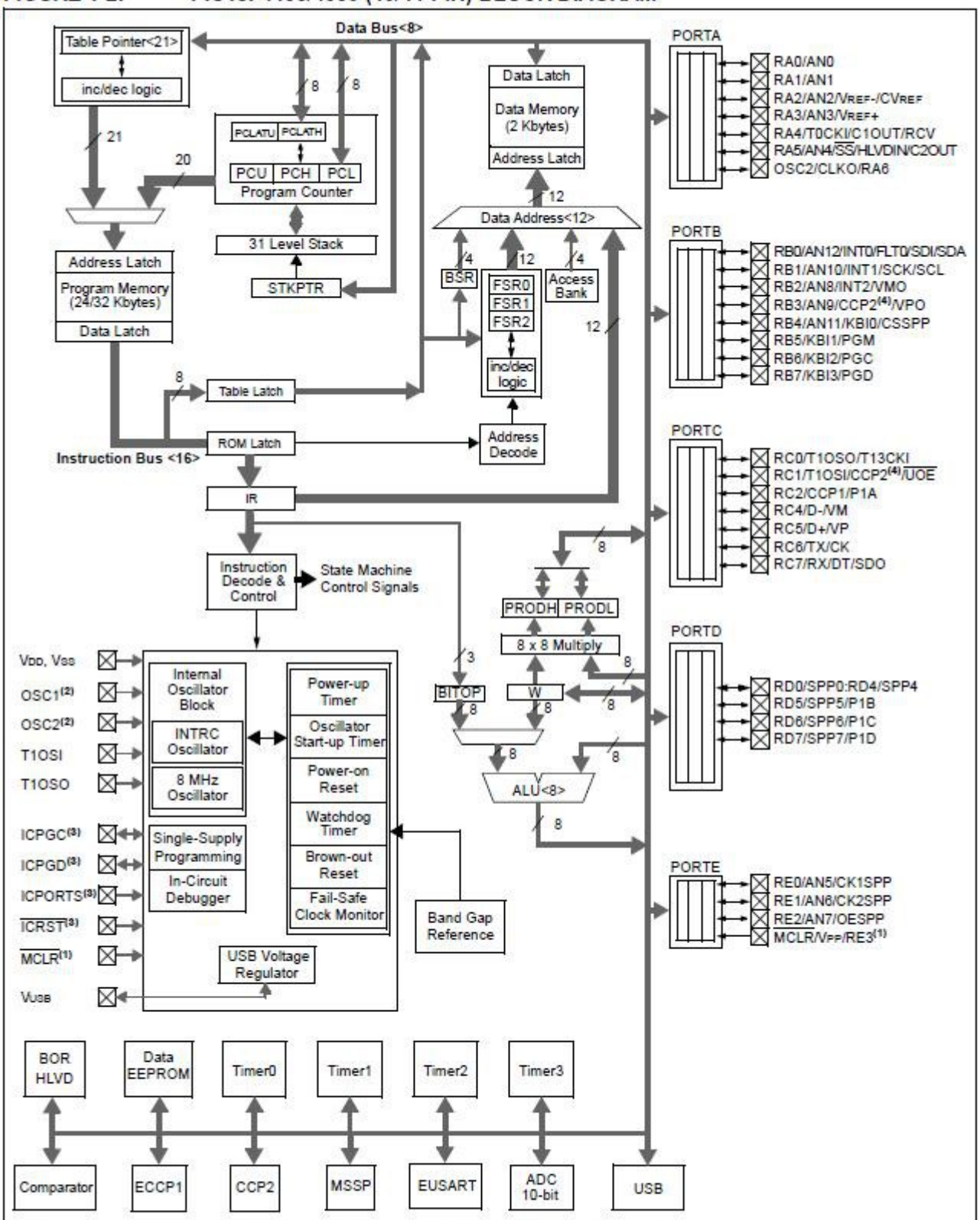
Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I²C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

PIC18F2455/2550/4455/4550

FIGURE 1-2: PIC18F4455/4550 (40/44-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with MCLR and is only available when the MCLR Resets are disabled.

2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

3: These pins are only available on 44-pin TQFP under certain conditions. Refer to **Section 25.9 "Special ICPORT Features (Designated Packages Only)"** for additional information.

4: RB3 is the alternate pin for CCP2 multiplexing.

PIC18F2455/2550/4455/4550

5.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 enhanced microcontroller devices:

- Program Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate busses; this allows for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Flash program memory is provided in **Section 6.0 "Flash Program Memory"**. Data EEPROM is discussed separately in **Section 7.0 "Data EEPROM Memory"**.

5.1 Program Memory Organization

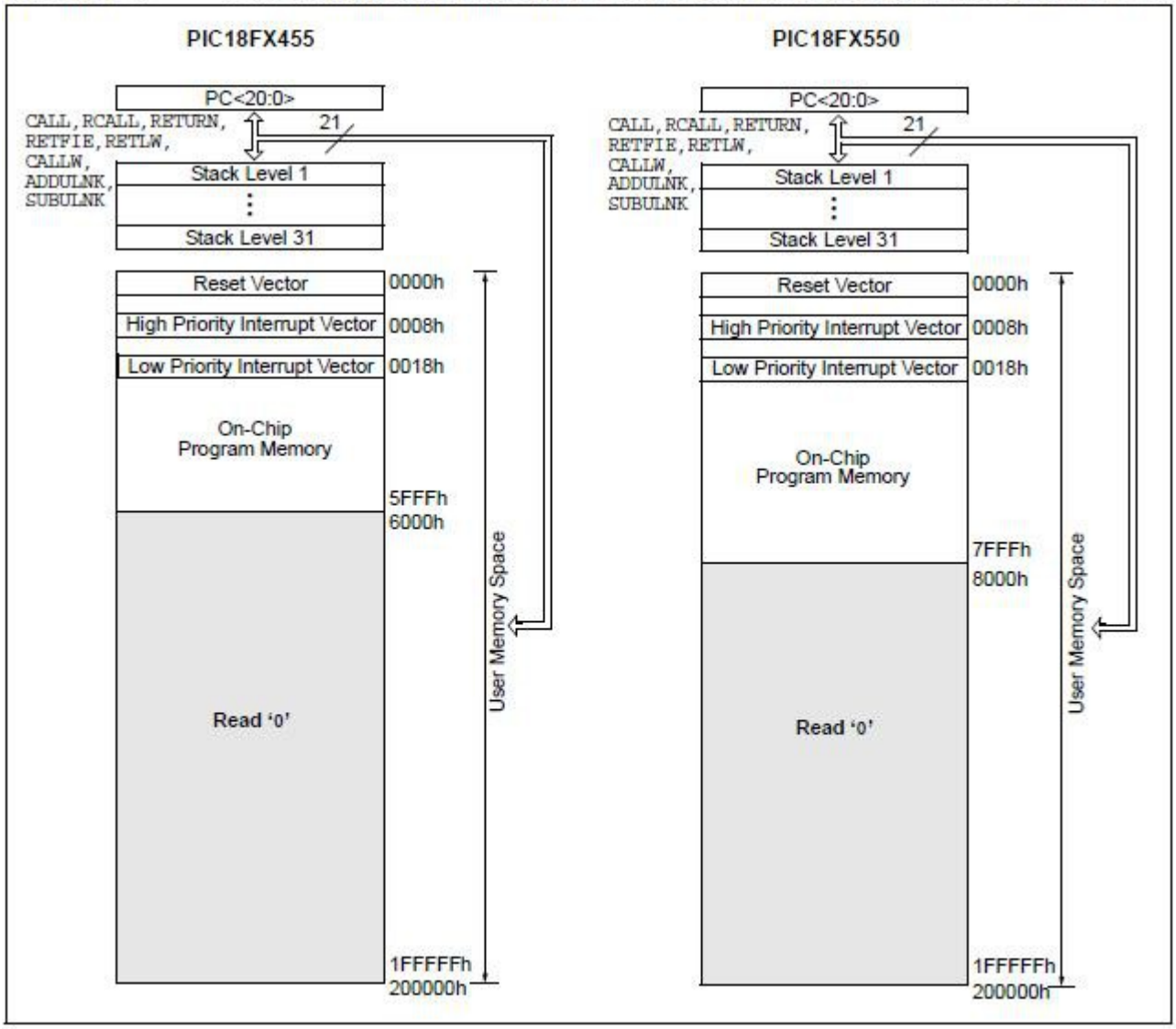
PIC18 microcontrollers implement a 21-bit program counter which is capable of addressing a 2-Mbyte program memory space. Accessing a location between the upper boundary of the physically implemented memory and the 2-Mbyte address will return all '0's (a NOP instruction).

The PIC18F2455 and PIC18F4455 each have 24 Kbytes of Flash memory and can store up to 12,288 single-word instructions. The PIC18F2550 and PIC18F4550 each have 32 Kbytes of Flash memory and can store up to 16,384 single-word instructions.

PIC18 devices have two interrupt vectors. The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

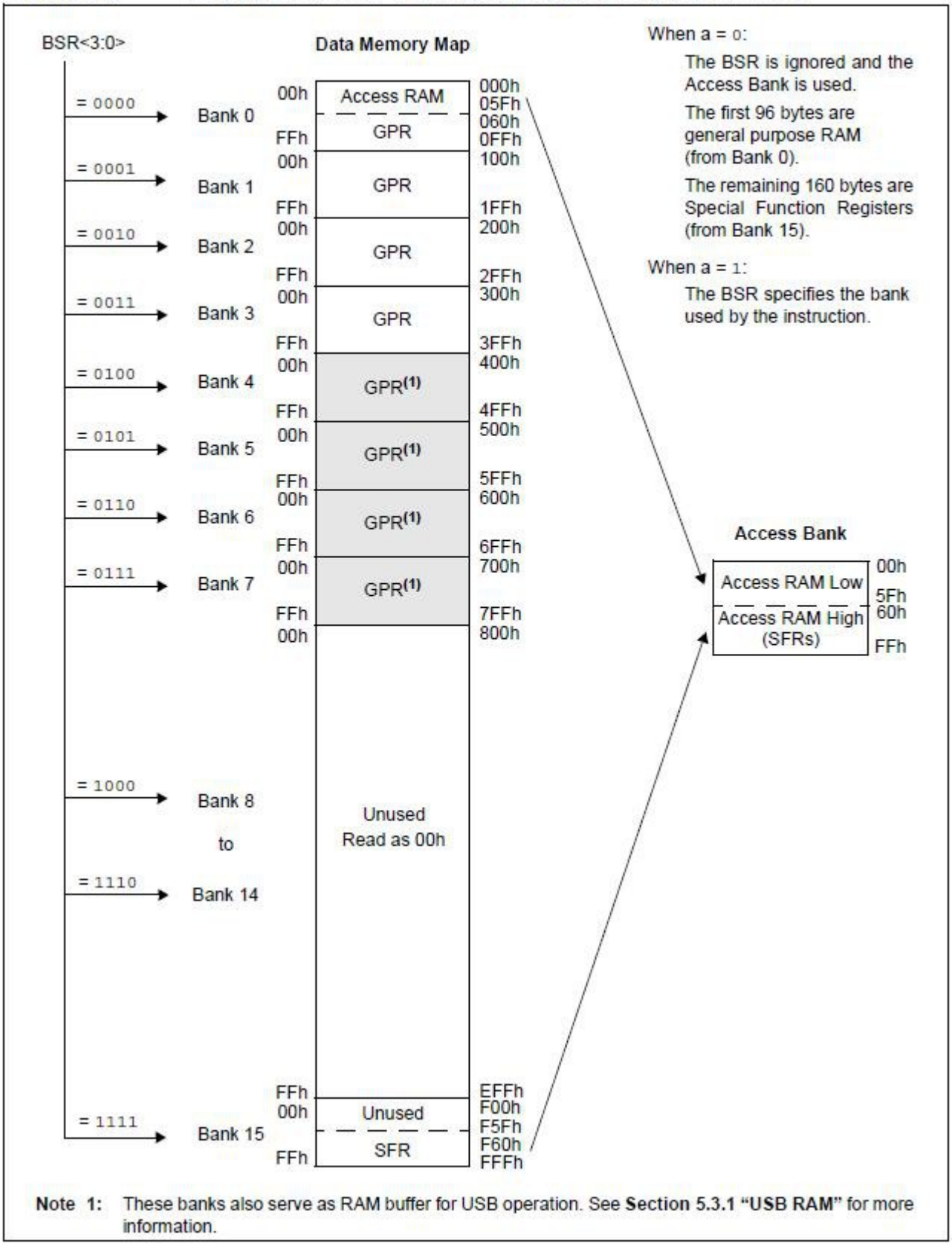
The program memory maps for PIC18FX455 and PIC18FX550 devices are shown in Figure 5-1.

FIGURE 5-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F2455/2550/4455/4550 DEVICES



PIC18F2455/2550/4455/4550

FIGURE 5-5: DATA MEMORY MAP FOR PIC18F2455/2550/4455/4550 DEVICES



Note 1: These banks also serve as RAM buffer for USB operation. See **Section 5.3.1 "USB RAM"** for more information.

PIC18F2455/2550/4455/4550

9.0 INTERRUPTS

The PIC18F2455/2550/4455/4550 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will interrupt any low priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/ compiler to automatically take care of the placement of these bits within the specified register.

Each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PICmicro® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit which enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The "return from interrupt" instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding enable bit or the GIE bit.

Note:	Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

9.1 USB Interrupts

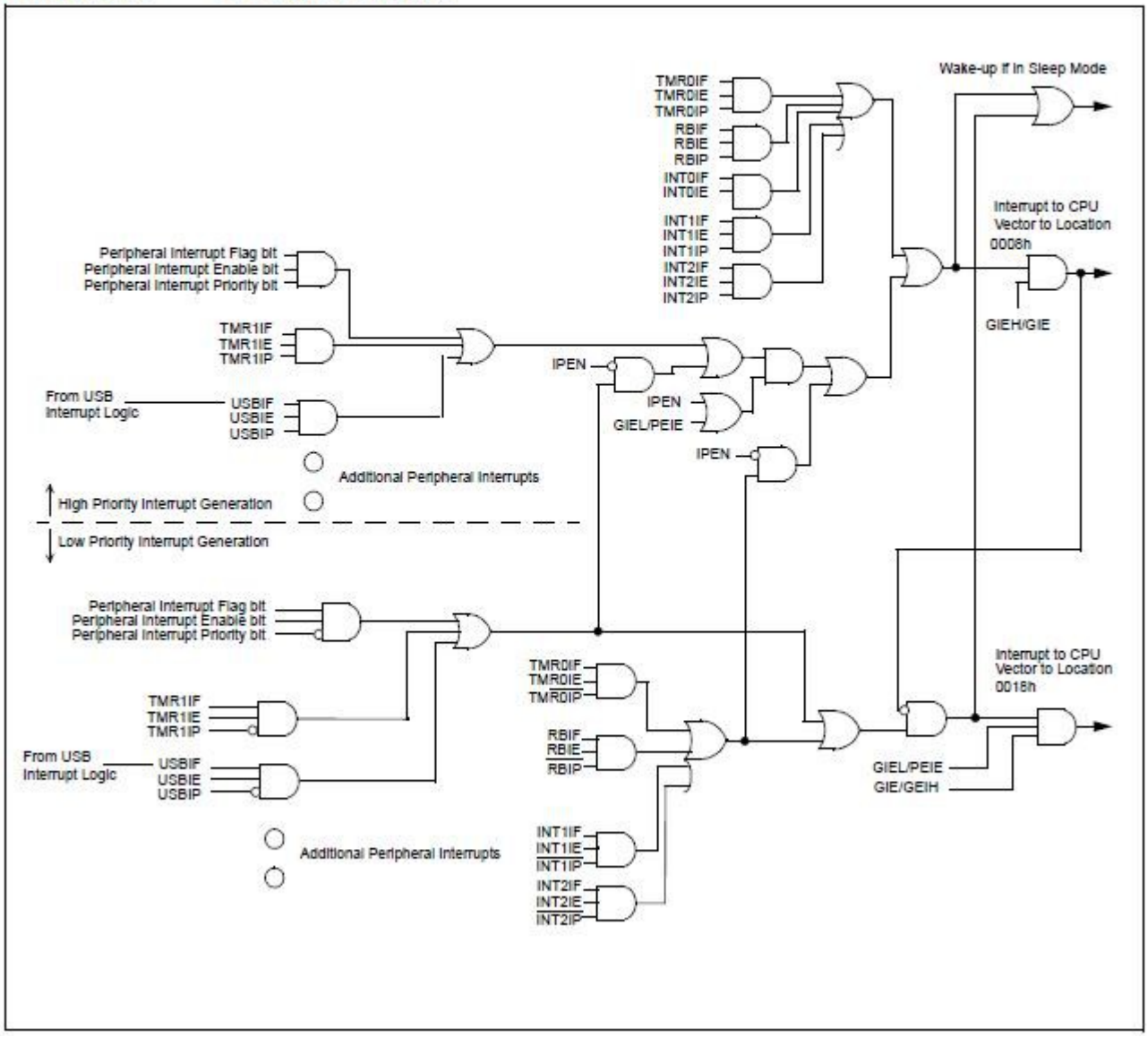
Unlike other peripherals, the USB module is capable of generating a wide range of interrupts for many types of events. These include several types of normal communication and status events and several module level error events.

To handle these events, the USB module is equipped with its own interrupt logic. The logic functions in a manner similar to the microcontroller level interrupt funnel, with each interrupt source having separate flag and enable bits. All events are funneled to a single device level interrupt, USBIF (PIR2<5>). Unlike the device level interrupt logic, the individual USB interrupt events cannot be individually assigned their own priority. This is determined at the device level interrupt funnel for all USB events by the USBIP bit.

For additional details on USB interrupt logic, refer to **Section 17.5 "USB Interrupts"**.

PIC18F2455/2550/4455/4550

FIGURE 9-1: INTERRUPT LOGIC



PIC18F2455/2550/4455/4550

9.2 INTCON Registers

The INTCON registers are readable and writable registers which contain various enable, priority and flag bits.

Note: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all high priority interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

REGISTER 9-2: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 3 **Unimplemented:** Read as '0'

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as '0'

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

PIC18F2455/2550/4455/4550

9.3 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Request (Flag) registers (PIR1 and PIR2).

Note 1: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE (INTCON<7>).

2: User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **SPPIF:** Streaming Parallel Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

Note 1: This bit is reserved on 28-pin devices; always maintain this bit clear.

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF:** EUSART Receive Interrupt Flag bit
1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The EUSART receive buffer is empty

bit 4 **TXIF:** EUSART Transmit Interrupt Flag bit
1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The EUSART transmit buffer is full

bit 3 **SSPIF:** Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

9.4 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Enable registers (PIE1 and PIE2). When IPEN = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 9-6: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **SPPIE:** Streaming Parallel Port Read/Write Interrupt Enable bit(1)
1 = Enables the SPP read/write interrupt
0 = Disables the SPP read/write interrupt

Note 1: This bit is reserved on 28-pin devices; always maintain this bit clear.

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE:** EUSART Receive Interrupt Enable bit
1 = Enables the EUSART receive interrupt
0 = Disables the EUSART receive interrupt

bit 4 **TXIE:** EUSART Transmit Interrupt Enable bit
1 = Enables the EUSART transmit interrupt
0 = Disables the EUSART transmit interrupt

bit 3 **SSPIE:** Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

9.5 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Priority registers (IPR1 and IPR2). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 9-8: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
SPPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

bit 7 **SPPIP:** Streaming Parallel Port Read/Write Interrupt Priority bit(1)
1 = High priority
0 = Low priority

Note 1: This bit is reserved on 28-pin devices; always maintain this bit set.

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP:** EUSART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP:** EUSART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP:** Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

9.6 RCON Register

The RCON register contains flag bits which are used to determine the cause of the last Reset or wake-up from Idle or Sleep modes. RCON also contains the IPEN bit which enables interrupt priorities.

REGISTER 9-10: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6 **SBOREN:** BOR Software Enable bit
For details of bit operation, see Register 4-1.

bit 5 **Unimplemented:** Read as '0'

bit 4 $\overline{RI}$**:** RESET Instruction Flag bit
For details of bit operation, see Register 4-1.

bit 3 $\overline{TO}$**:** Watchdog Time-out Flag bit
For details of bit operation, see Register 4-1.

bit 2 $\overline{PD}$**:** Power-Down Detection Flag bit
For details of bit operation, see Register 4-1.

bit 1 $\overline{POR}$**:** Power-on Reset Status bit
For details of bit operation, see Register 4-1.

bit 0 $\overline{BOR}$**:** Brown-out Reset Status bit
For details of bit operation, see Register 4-1.

Note: Actual Reset values are determined by device configuration and the nature of the device Reset. See Register 4-1 for additional information.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

10.0 I/O PORTS

Depending on the device selected and features enabled, there are up to five ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

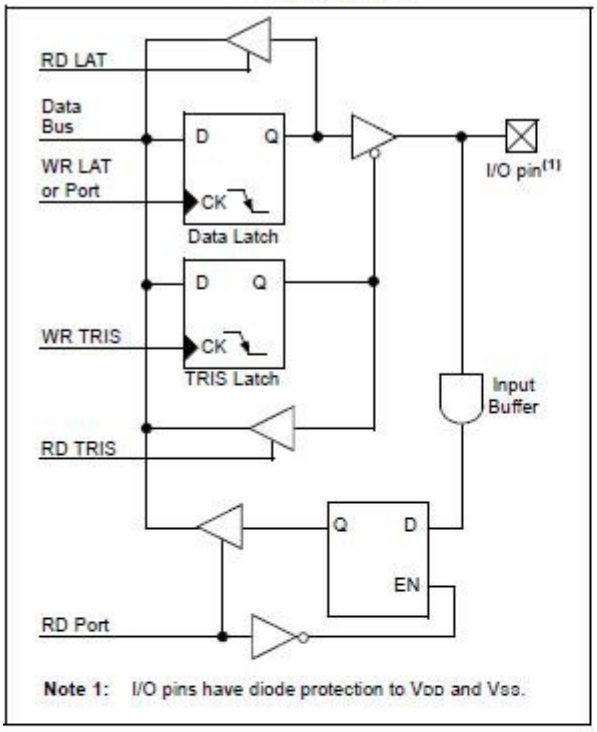
Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- Port register (reads the levels on the pins of the device)
- LAT register (output latch)

The Data Latch register (LATA) is useful for read-modify-write operations on the value that the I/O pins are driving.

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 10-1.

FIGURE 10-1: GENERIC I/O PORT OPERATION



Note 1: I/O pins have diode protection to VDD and VSS.

10.1 PORTA, TRISA and LATA Registers

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins; writing to it will write to the port latch.

The Data Latch register (LATA) is also memory mapped. Read-modify-write operations on the LATA register read and write the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA6 pin is multiplexed with the main oscillator pin; it is enabled as an oscillator or I/O pin by the selection of the main oscillator in Configuration Register 1H (see **Section 25.1 "Configuration Bits"** for details). When not used as a port pin, RA6 and its associated TRIS and LAT bits are read as '0'.

RA4 is also multiplexed with the USB module; it serves as a receiver input from an external USB transceiver. For details on configuration of the USB module, see **Section 17.2 "USB Status and Control"**.

Several PORTA pins are multiplexed with analog inputs, the analog VREF+ and VREF- inputs and the comparator voltage reference output. The operation of pins RA3:RA0 and RA5 as A/D converter inputs is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register 1).

Note:	On a Power-on Reset, RA5 and RA3:RA0 are configured as analog inputs and read as '0'. RA4 is configured as a digital input.

All other PORTA pins have TTL input levels and full CMOS output drivers.

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 10-1: INITIALIZING PORTA

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF    LATA    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0Fh     ; Configure A/D
MOVWF   ADCON1  ; for digital inputs
MOVLW   07h     ; Configure comparators
MOVWF   CMCON   ; for digital input
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISA   ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```

PIC18F2455/2550/4455/4550

TABLE 10-1: PORTA I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RA0/AN0	RA0	0	OUT	DIG	LATA<0> data output; not affected by analog input.
		1	IN	TTL	PORTA<0> data input; disabled when analog input enabled.
	AN0	1	IN	ANA	A/D input channel 0 and Comparator C1- input. Default configuration on POR; does not affect digital output.
RA1/AN1	RA1	0	OUT	DIG	LATA<1> data output; not affected by analog input.
		1	IN	TTL	PORTA<1> data input; reads '0' on POR.
	AN1	1	IN	ANA	A/D input channel 1 and Comparator C2- input. Default configuration on POR; does not affect digital output.
RA2/AN2/ VREF-/CVREF	RA2	0	OUT	DIG	LATA<2> data output; not affected by analog input. Disabled when CVREF output enabled.
		1	IN	TTL	PORTA<2> data input. Disabled when analog functions enabled; disabled when CVREF output enabled.
	AN2	1	IN	ANA	A/D input channel 2 and Comparator C2+ input. Default configuration on POR; not affected by analog output.
	VREF-	1	IN	ANA	A/D and comparator voltage reference low input.
	CVREF	x	OUT	ANA	Comparator voltage reference output. Enabling this feature disables digital I/O.
RA3/AN3/ VREF+	RA3	0	OUT	DIG	LATA<3> data output; not affected by analog input.
		1	IN	TTL	PORTA<3> data input; disabled when analog input enabled.
	AN3	1	IN	ANA	A/D input channel 3 and Comparator C1+ input. Default configuration on POR.
	VREF+	1	IN	ANA	A/D and comparator voltage reference high input.
RA4/T0CKI/ C1OUT/RCV	RA4	0	OUT	DIG	LATA<4> data output; not affected by analog input.
		1	IN	ST	PORTA<4> data input; disabled when analog input enabled.
	T0CKI	1	IN	ST	Timer0 clock input.
	C1OUT	0	OUT	DIG	Comparator 1 output; takes priority over port data.
	RCV	x	IN	TTL	External USB transceiver RCV input.
RA5/AN4/$\overline{SS}$/ HLVDIN/C2OUT	RA5	0	OUT	DIG	LATA<5> data output; not affected by analog input.
		1	IN	TTL	PORTA<5> data input; disabled when analog input enabled.
	AN4	1	IN	ANA	A/D input channel 4. Default configuration on POR.
	$\overline{SS}$	1	IN	TTL	Slave select input for SSP (MSSP module).
	HLVDIN	1	IN	ANA	High/Low-Voltage Detect external trip point input.
	C2OUT	0	OUT	DIG	Comparator 2 output; takes priority over port data.
OSC2/CLKO/ RA6	RA6	0	OUT	DIG	LATA<6> data output. Available only in ECIO, ECPIO and INTIO modes; otherwise, reads as '0'.
		1	IN	TTL	PORTA<6> data input. Available only in ECIO, ECPIO and INTIO modes; otherwise, reads as '0'.
	OSC2	x	OUT	ANA	Main oscillator feedback output connection (all XT and HS modes).
	CLKO	x	OUT	DIG	System cycle clock output (FOSC/4); available in EC, ECPLL and INTCKO modes.

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

PIC18F2455/2550/4455/4550

TABLE 10-3: PORTB I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB0/AN12/ INT0/FLT0/ SDI/SDA	RB0	0	OUT	DIG	LATB<0> data output; not affected by analog input.
		1	IN	TTL	PORTB<0> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled(1).
	AN12	1	IN	ANA	A/D input channel 12(1).
	INT0	1	IN	ST	External Interrupt 0 input.
	FLT0	1	IN	ST	Enhanced PWM Fault input (ECCP1 module); enabled in software.
	SDI	1	IN	ST	SPI™ data input (MSSP module).
	SDA	1	OUT	DIG	I^2C™ data output (MSSP module); takes priority over port data.
		1	IN	I^2C/SMB	I^2C data input (MSSP module); input type depends on module setting.
RB1/AN10/ INT1/SCK/ SCL	RB1	0	OUT	DIG	LATB<1> data output; not affected by analog input.
		1	IN	TTL	PORTB<1> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled(1).
	AN10	1	IN	ANA	A/D input channel 10(1).
	INT1	1	IN	ST	External interrupt 1 input.
	SCK	0	OUT	DIG	SPI clock output (MSSP module); takes priority over port data.
		1	IN	ST	SPI clock input (MSSP module).
	SCL	0	OUT	DIG	I^2C clock output (MSSP module); takes priority over port data.
		1	IN	I^2C/SMB	I^2C clock input (MSSP module); input type depends on module setting.
RB2/AN8/ INT2/VMO	RB2	0	OUT	DIG	LATB<2> data output; not affected by analog input.
		1	IN	TTL	PORTB<2> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled(1).
	AN8	1	IN	ANA	A/D input channel 8(1).
	INT2	1	IN	ST	External Interrupt 2 input.
	VMO	0	OUT	DIG	External USB transceiver VMO data output.
RB3/AN9/ CCP2/VPO	RB3	0	OUT	DIG	LATB<3> data output; not affected by analog input.
		1	IN	TTL	PORTB<3> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled(1).
	AN9	1	IN	ANA	A/D input channel 9(1).
	CCP2(2)	0	OUT	DIG	CCP2 Compare and PWM output.
		1	IN	ST	CCP2 Capture input.
	VPO	0	OUT	DIG	External USB transceiver VPO data output.
RB4/AN11/ KBI0/CSSPP	RB4	0	OUT	DIG	LATB<4> data output; not affected by analog input.
		1	IN	TTL	PORTB<4> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input enabled(1).
	AN11	1	IN	ANA	A/D input channel 11(1).
	KBI0	1	IN	TTL	Interrupt on pin change.
	CSSPP(4)	0	IN	DIG	SPP chip select control output.
RB5/KBI1/ PGM	RB5	0	OUT	DIG	LATB<5> data output.
		1	IN	TTL	PORTB<5> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	KBI1	1	IN	TTL	Interrupt on pin change.
	PGM	x	IN	ST	Single-Supply Programming mode entry (ICSP™). Enabled by LVP configuration bit; all other pin functions disabled.

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, I^2C/SMB = I^2C/SMBus input buffer, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

PIC18F2455/2550/4455/4550

TABLE 10-3: PORTB I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB6/KBI2/ PGC	RB6	0	OUT	DIG	LATB<6> data output.
		1	IN	TTL	PORTB<6> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	KBI2	1	IN	TTL	Interrupt on pin change.
	PGC	x	IN	ST	Serial execution (ICSP) clock input for ICSP and ICD operation(3).
RB7/KBI3/ PGD	RB7	0	OUT	DIG	LATB<7> data output.
		1	IN	TTL	PORTB<7> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	KBI3	1	IN	TTL	Interrupt on pin change.
	PGD	x	OUT	DIG	Serial execution data output for ICSP and ICD operation(3).
		x	IN	ST	Serial execution data input for ICSP and ICD operation(3).

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, I^2C/SMB = I^2C/SMBus input buffer, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

Note 1: Configuration on POR is determined by PBADEN configuration bit. Pins are configured as analog inputs when PBADEN is set and digital inputs when PBADEN is cleared.

2: Alternate pin assignment for CCP2 when CCP2MX = 0. Default assignment is RC1.

3: All other pin functions are disabled when ICSP™ or ICD operation is enabled.

4: 40/44-pin devices only.

PIC18F2455/2550/4455/4550

TABLE 10-5: PORTC I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RC0/T1OSO/ T13CKI	RC0	0	OUT	DIG	LATC<0> data output.
		1	IN	ST	PORTC<0> data input.
	T1OSO	x	OUT	ANA	Timer1 oscillator output; enabled when Timer1 oscillator enabled. Disables digital I/O.
	T13CKI	1	IN	ST	Timer1/Timer3 counter input.
RC1/T1OSI/ CCP2/UOE	RC1	0	OUT	DIG	LATC<1> data output.
		1	IN	ST	PORTC<1> data input.
	T1OSI	x	IN	ANA	Timer1 oscillator input; enabled when Timer1 oscillator enabled. Disables digital I/O.
	CCP2(1)	0	OUT	DIG	CCP2 Compare and PWM output; takes priority over port data.
		1	IN	ST	CCP2 Capture input.
	$\overline{UOE}$	0	OUT	DIG	External USB transceiver $\overline{OE}$ output.
RC2/CCP1/ P1A	RC2	0	OUT	DIG	LATC<2> data output.
		1	IN	ST	PORTC<2> data input.
	CCP1	0	OUT	DIG	ECCP1 Compare and PWM output; takes priority over port data.
		1	IN	ST	ECCP1 Capture input.
	P1A(3)	0	OUT	DIG	ECCP1 Enhanced PWM output, channel A; takes priority over port data. May be configured for tri-state during Enhanced PWM shutdown events.
RC4/D-/VM	RC4	—(2)	IN	TTL	PORTC<4> data input; disabled when USB enabled.
	D-	—(2)	OUT	XCVR	USB bus differential minus line output (internal transceiver).
		—(2)	IN	XCVR	USB bus differential minus line input (internal transceiver).
	VM	—(2)	IN	TTL	External USB transceiver VM input.
RC5/D+/VP	RC5	—(2)	IN	TTL	PORTC<5> data input; disabled when USB enabled.
	D+	—(2)	OUT	XCVR	USB bus differential plus line output (internal transceiver).
		—(2)	IN	XCVR	USB bus differential plus line input (internal transceiver).
	VP	—(2)	IN	TTL	External USB transceiver VP input.
RC6/TX/CK	RC6	0	OUT	DIG	LATC<6> data output.
		1	IN	ST	PORTC<6> data input.
	TX	0	OUT	DIG	Asynchronous serial transmit data output (EUSART module); takes priority over port data. User must configure as output.
	CK	0	OUT	DIG	Synchronous serial clock output (EUSART module); takes priority over port data.
		1	IN	ST	Synchronous serial clock input (EUSART module).
RC7/RX/DT/ SDO	RC7	0	OUT	DIG	LATC<7> data output.
		1	IN	ST	PORTC<7> data input.
	RX	1	IN	ST	Asynchronous serial receive data input (EUSART module).
	DT	1	OUT	DIG	Synchronous serial data output (EUSART module); takes priority over SPI™ and port data.
		1	IN	ST	Synchronous serial data input (EUSART module). User must configure as an input.
	SDO	0	OUT	DIG	SPI data output (MSSP module); takes priority over port data.

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, XCVR = USB transceiver, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

PIC18F2455/2550/4455/4550

TABLE 10-7: PORTD I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RD0/SPP0	RD0	0	OUT	DIG	LATD<0> data output.
		1	IN	ST	PORTD<0> data input.
	SPP0	1	OUT	DIG	SPP<0> output data; takes priority over port data.
		1	IN	TTL	SPP<0> input data.
RD1/SPP1	RD1	0	OUT	DIG	LATD<1> data output.
		1	IN	ST	PORTD<1> data input.
	SPP1	1	OUT	DIG	SPP<1> output data; takes priority over port data.
		1	IN	TTL	SPP<1> input data.
RD2/SPP2	RD2	0	OUT	DIG	LATD<2> data output.
		1	IN	ST	PORTD<2> data input.
	SPP2	1	OUT	DIG	SPP<2> output data; takes priority over port data.
		1	IN	TTL	SPP<2> input data.
RD3/SPP3	RD3	0	OUT	DIG	LATD<3> data output.
		1	IN	ST	PORTD<3> data input.
	SPP3	1	OUT	DIG	SPP<3> output data; takes priority over port data.
		1	IN	TTL	SPP<3> input data.
RD4/SPP4	RD4	0	OUT	DIG	LATD<4> data output.
		1	IN	ST	PORTD<4> data input.
	SPP4	1	OUT	DIG	SPP<4> output data; takes priority over port data.
		1	IN	TTL	SPP<4> input data.
RD5/SPP5/P1B	RD5	0	OUT	DIG	LATD<5> data output
		1	IN	ST	PORTD<5> data input
	SPP5	1	OUT	DIG	SPP<5> output data; takes priority over port data.
		1	IN	TTL	SPP<5> input data.
	P1B	0	OUT	DIG	ECCP1 Enhanced PWM output, channel B; takes priority over port and SPP data(1).
RD6/SPP6/P1C	RD6	0	OUT	DIG	LATD<6> data output.
		1	IN	ST	PORTD<6> data input.
	SPP6	1	OUT	DIG	SPP<6> output data; takes priority over port data.
		1	IN	TTL	SPP<6> input data.
	P1C	0	OUT	DIG	ECCP1 Enhanced PWM output, channel C; takes priority over port and SPP data(1).
RD7/SPP7/P1D	RD7	0	OUT	DIG	LATD<7> data output.
		1	IN	ST	PORTD<7> data input.
	SPP7	1	OUT	DIG	SPP<7> output data; takes priority over port data.
		1	IN	TTL	SPP<7> input data.
	P1D	0	OUT	DIG	ECCP1 Enhanced PWM output, channel D; takes priority over port and SPP data(1).

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input.
Note 1: May be configured for tri-state during Enhanced PWM shutdown events.

PIC18F2455/2550/4455/4550

TABLE 10-9: PORTE I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RE0/AN5/ CK1SPP	RE0	0	OUT	DIG	LATE<0> data output; not affected by analog input.
		1	IN	ST	PORTE<0> data input; disabled when analog input enabled.
	AN5	1	IN	ANA	A/D input channel 5; default configuration on POR.
	CK1SPP	0	OUT	DIG	SPP clock 1 output (SPP enabled).
RE1/AN6/ CK2SPP	RE1	0	OUT	DIG	LATE<1> data output; not affected by analog input.
		1	IN	ST	PORTE<1> data input; disabled when analog input enabled.
	AN6	1	IN	ANA	A/D input channel 6; default configuration on POR.
	CK2SPP	0	OUT	DIG	SPP clock 2 output (SPP enabled).
RE2/AN7/ OESPP	RE2	0	OUT	DIG	LATE<2> data output; not affected by analog input.
		1	IN	ST	PORTE<2> data input; disabled when analog input enabled.
	AN7	1	IN	ANA	A/D input channel 7; default configuration on POR.
	OESPP	0	OUT	DIG	SPP enable output (SPP enabled).
$\overline{MCLR}$/VPP/ RE3	RE3	— (1)	IN	ST	PORTE<3> data input; enabled when MCLRE configuration bit is clear.
	$\overline{MCLR}$	— (1)	IN	ST	External Master Clear input; enabled when MCLRE configuration bit is set.
	VPP	— (1)	IN	ANA	High-voltage detection, used for ICSP™ mode entry detection. Always available regardless of pin mode.

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input.
Note 1: RE3 does not have a corresponding TRISE<3> bit. This pin is always an input regardless of mode.

PIC18F2455/2550/4455/4550

11.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 11-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 11-1. Figure 11-2 shows a simplified block diagram of the timer module in 16-bit mode.

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

11.1 Timer0 Operation

Timer0 can operate as either a timer or a counter; the mode is selected by clearing the T0CS bit (T0CON<5>). In Timer mode, the module increments on every clock by default unless a different prescaler value is selected (see **Section 11.3 "Prescaler"**). If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

The Counter mode is selected by setting the T0CS bit (= 1). In Counter mode, Timer0 increments either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

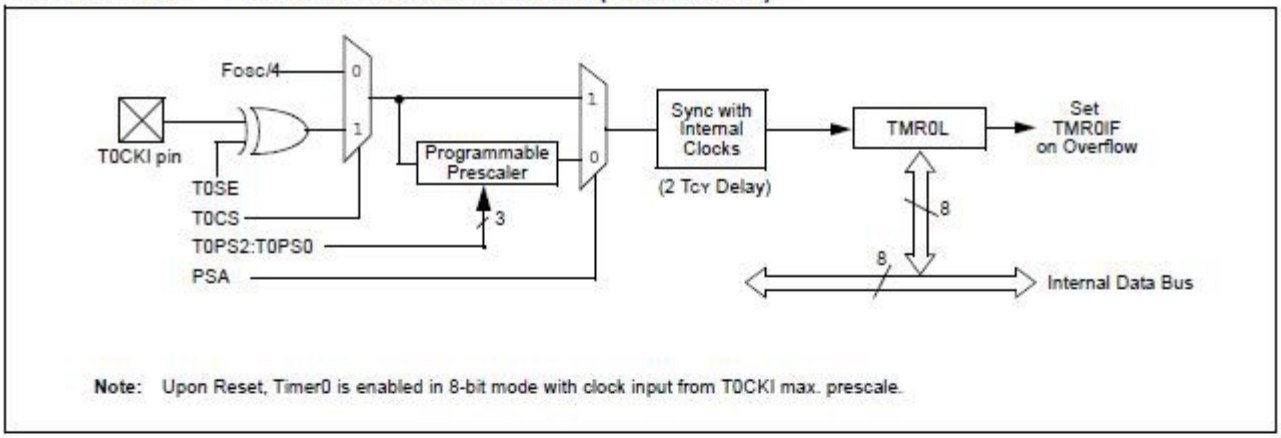
An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (Tosc). There is a delay between synchronization and the onset of incrementing the timer/counter.

11.2 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode; it is actually a buffered version of the real high byte of Timer0, which is not directly readable nor writable (refer to Figure 11-2). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid, due to a rollover between successive reads of the high and low byte.

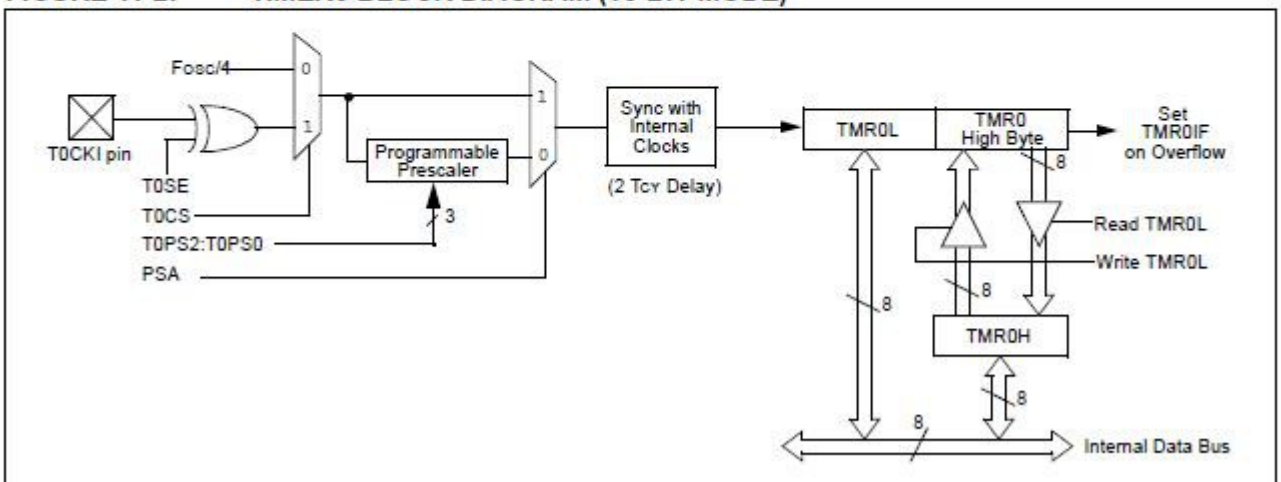
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

FIGURE 11-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

PIC18F2455/2550/4455/4550

11.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable; its value is set by the PSA and T0PS2:T0PS0 bits (T0CON<3:0>) which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When it is assigned, prescale values from 1:2 through 1:256, in power-of-2 increments, are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0`, etc.) clear the prescaler count.

Note: Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count but will not change the prescaler assignment.

11.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed "on-the-fly" during program execution.

11.4 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMR0IE bit (INTCON<5>). Before re-enabling the interrupt, the TMR0IF bit must be cleared in software by the Interrupt Service Routine.

Since Timer0 is shut down in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

TABLE 11-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TMR0L	Timer0 Module Low Byte Register								52
TMR0H	Timer0 Module High Byte Register								52
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	51
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	52
TRISA	—	TRISA6(1)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	54

Legend: — = unimplemented locations, read as '0'. Shaded cells are not used by Timer0.

Note 1: RA6 is configured as a port pin based on various primary oscillator modes. When the port pin is disabled, all of the associated bits read '0'.

PIC18F2455/2550/4455/4550

20.0 ENHANCED UNIVERSAL SYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs and so on.

The Enhanced Universal Synchronous Receiver Transmitter (EUSART) module implements additional features, including Automatic Baud Rate Detection (ABD) and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These features make it ideally suited for use in Local Interconnect Network bus (LIN bus) systems.

The EUSART can be configured in the following modes:

- Asynchronous (full-duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit Break character transmission
- Synchronous – Master (half-duplex) with selectable clock polarity
- Synchronous – Slave (half-duplex) with selectable clock polarity

The pins of the Enhanced USART are multiplexed with PORTC. In order to configure RC6/TX/CK and RC7/RX/DT/SDO as a USART:

- bit SPEN (RCSTA<7>) must be set (= 1)
- bit TRISC<7> must be set (= 1)
- bit TRISC<6> must be cleared (= 0) for Asynchronous and Synchronous Master modes or set (= 1) for Synchronous Slave mode

Note:	The EUSART control will automatically reconfigure the pin from input to output as needed.

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

These are detailed on the following pages in Register 20-1, Register 20-2 and Register 20-3, respectively.

PIC18F2455/2550/4455/4550

REGISTER 20-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit

Asynchronous mode:
Don't care.

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in Sync mode, with the exception that SREN has no effect in Synchronous Slave mode.

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit

Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed

Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** Ninth bit of Transmit Data
Can be address/data bit or a parity bit.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

REGISTER 20-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** Ninth bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

REGISTER 20-3: BAUDCON: BAUD RATE CONTROL REGISTER

R/W-0	R-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

bit 7 **ABDOVF**: Auto-Baud Acquisition Rollover Status bit
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL**: Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

bit 5 **Unimplemented:** Read as '0'

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
Unused in this mode.
Synchronous mode:
1 = Idle state for clock (CK) is a high level
0 = Idle state for clock (CK) is a low level

bit 3 **BRG16:** 16-bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

20.1 Baud Rate Generator (BRG)

The BRG is a dedicated 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCON<3>) selects 16-bit mode.

The SPBRGH:SPBRG register pair controls the period of a free running timer. In Asynchronous mode, bits BRGH (TXSTA<2>) and BRG16 (BAUDCON<3>) also control the baud rate. In Synchronous mode, BRGH is ignored. Table 20-1 shows the formula for computation of the baud rate for different EUSART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 20-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 20-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 20-2. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGH:SPBRG registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

20.1.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRG register pair.

20.1.2 SAMPLING

The data on the RX pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 20-1: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

EXAMPLE 20-1: CALCULATING BAUD RATE ERROR

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

Desired Baud Rate = Fosc/(64 ([SPBRGH:SPBRG] + 1)

Solving for SPBRGH:SPBRG:

X = ((Fosc/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615

Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%

TABLE 20-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	53
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	53
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	53
SPBRGH	EUSART Baud Rate Generator Register High Byte								53
SPBRG	EUSART Baud Rate Generator Register Low Byte								53

PIC18F2455/2550/4455/4550

TABLE 20-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.04	832	300	-0.16	415	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117647	-2.12	16

PIC18F2455/2550/4455/4550

20.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA<4>). In this mode, the EUSART uses the standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is eight bits. An on-chip dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate depending on the BRGH and BRG16 bits (TXSTA<2> and BAUDCON<3>). Parity is not supported by the hardware but can be implemented in software and stored as the ninth data bit.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-bit Break Character Transmit
- Auto-Baud Rate Detection

20.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 20-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available).

Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and the TXIF flag bit (PIR1<4>) is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF will be set regardless of the state of TXIE; it cannot be cleared in software. TXIF is also not cleared immediately upon loading TXREG but becomes valid in the second instruction cycle following the load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

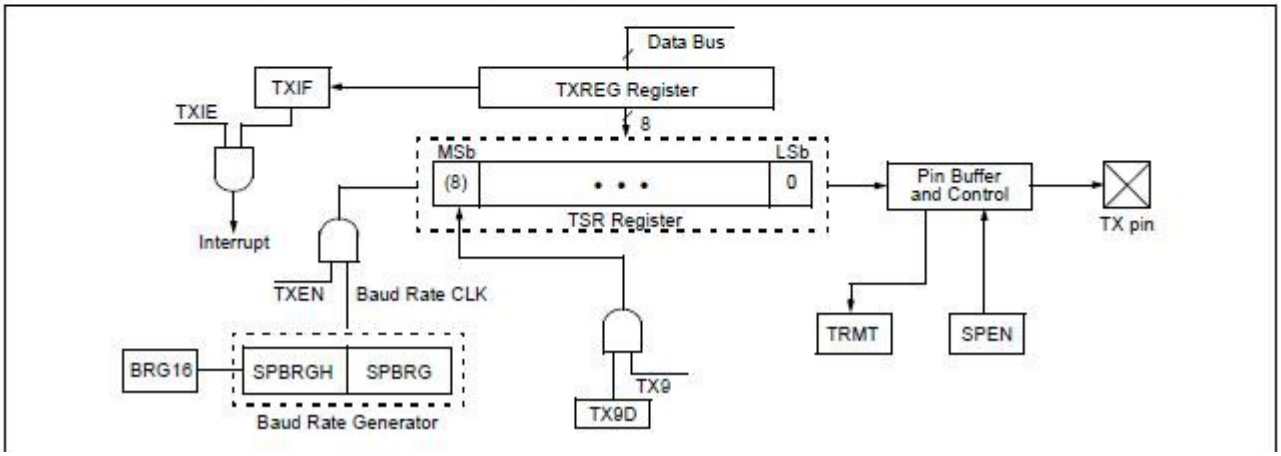
While TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty.

Note 1:	The TSR register is not mapped in data memory so it is not available to the user.
2:	Flag bit TXIF is set when enable bit TXEN is set.

To set up an Asynchronous Transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 20-3: EUSART TRANSMIT BLOCK DIAGRAM



PIC18F2455/2550/4455/4550

20.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 20-6. The data is received on the RX pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at Fosc. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

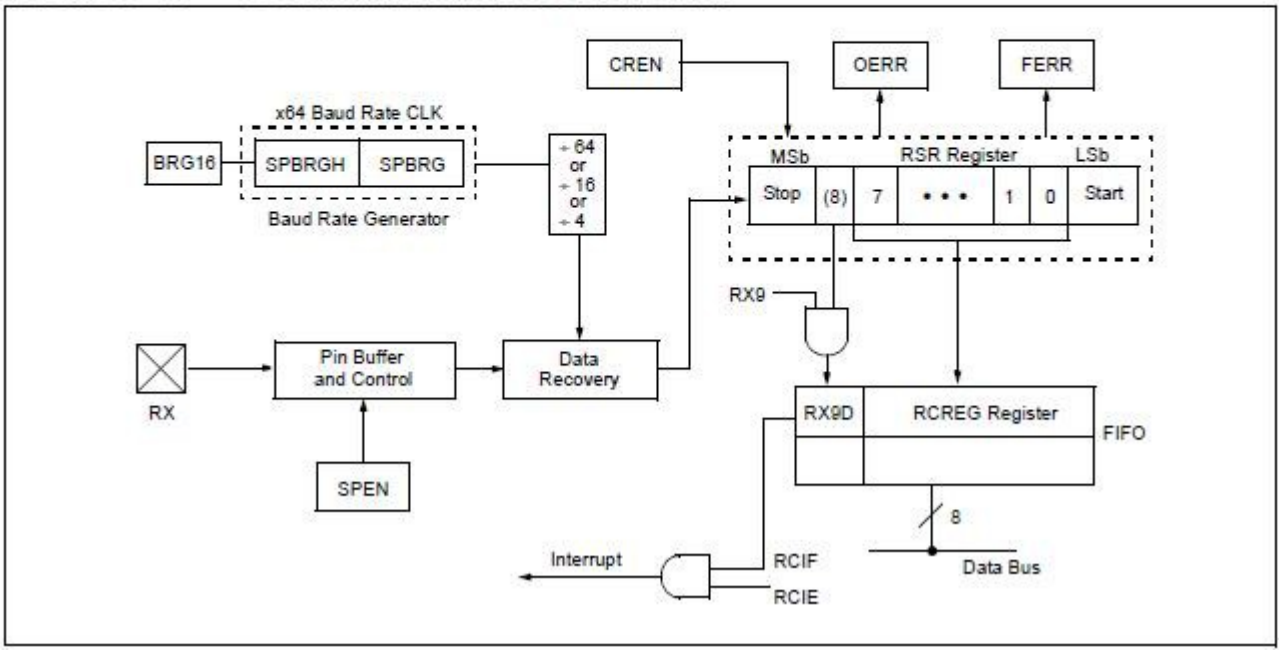
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCIE.
4. If 9-bit reception is desired, set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

20.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCIE and GIE bits are set.
8. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 20-6: EUSART RECEIVE BLOCK DIAGRAM



PIC18F2455/2550/4455/4550

21.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) converter module has 10 inputs for the 28-pin devices and 13 for the 40/44-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 21-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 21-2, configures the functions of the port pins. The ADCON2 register, shown in Register 21-3, configures the A/D clock source, programmed acquisition time and justification.

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits

0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

Note 1: These channels are not implemented on 28-pin devices.

2: Performing a conversion on unimplemented channels will return a floating input measurement.

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit

1 = A/D converter module is enabled
0 = A/D converter module is disabled

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit (V_{REF-} source)
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Voltage Reference Configuration bit (V_{REF+} source)
1 = V_{REF+} (AN3)
0 = V_{DD}

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Note 1: The POR value of the PCFG bits depends on the value of the PBADEN configuration bit. When PBADEN = 1, PCFG<3:0> = 0000; when PBADEN = 0, PCFG<3:0> = 0111.

2: AN5 through AN7 are available only on 40/44-pin devices.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as '0'

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC18F2455/2550/4455/4550

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (VDD and VSS) or the voltage level on the RA3/AN3/VREF+ and RA2/AN2/VREF-/CVREF pins.

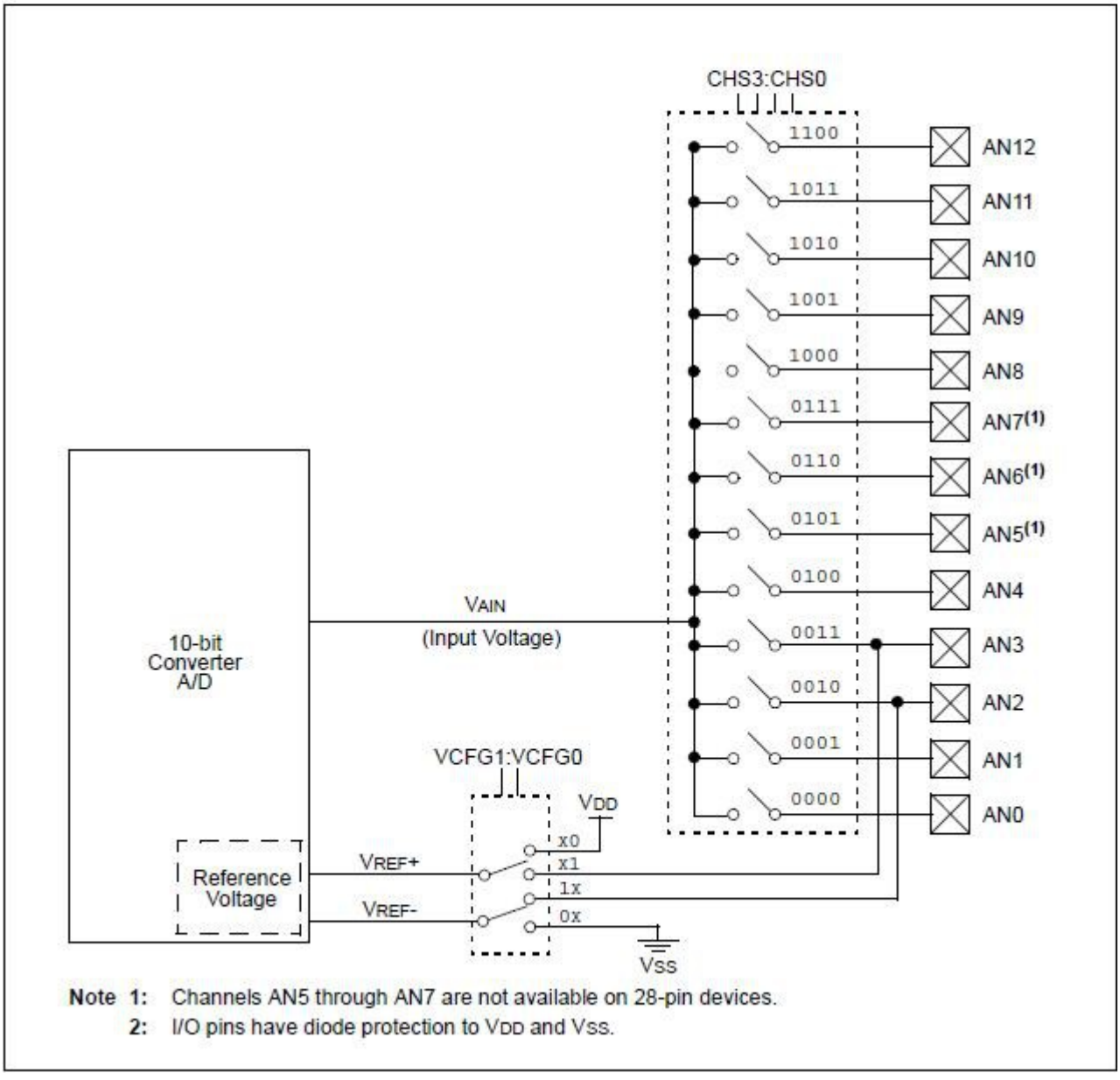
The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted.

Each port pin associated with the A/D converter can be configured as an analog input or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/$\overline{\text{DONE}}$ bit (ADCON0 register) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 21-1.

FIGURE 21-1: A/D BLOCK DIAGRAM



Note 1: Channels AN5 through AN7 are not available on 28-pin devices.
2: I/O pins have diode protection to VDD and VSS.

PIC18F2455/2550/4455/4550

The value in the ADRESH:ADRESL registers is not modified for a Power-on Reset. The ADRESH:ADRESL registers will contain unknown data after a Power-on Reset.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see **Section 21.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/DONE bit and the actual start of the conversion.

The following steps should be followed to perform an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/DONE bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/DONE bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 3 TAD is required before the next acquisition starts.

FIGURE 21-2: A/D TRANSFER FUNCTION

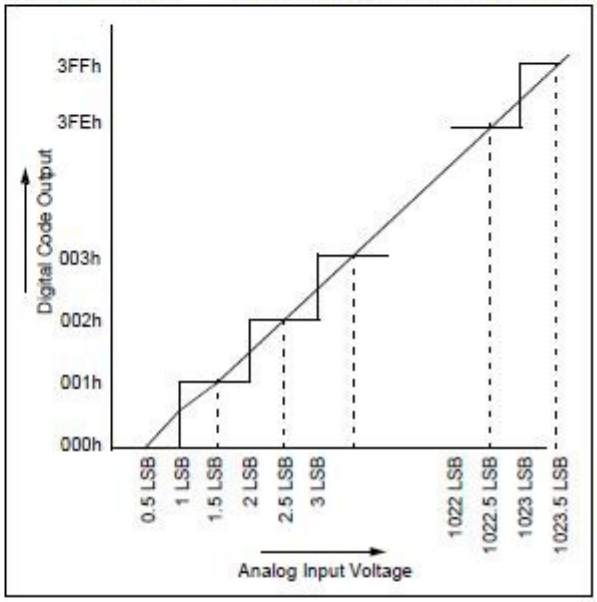
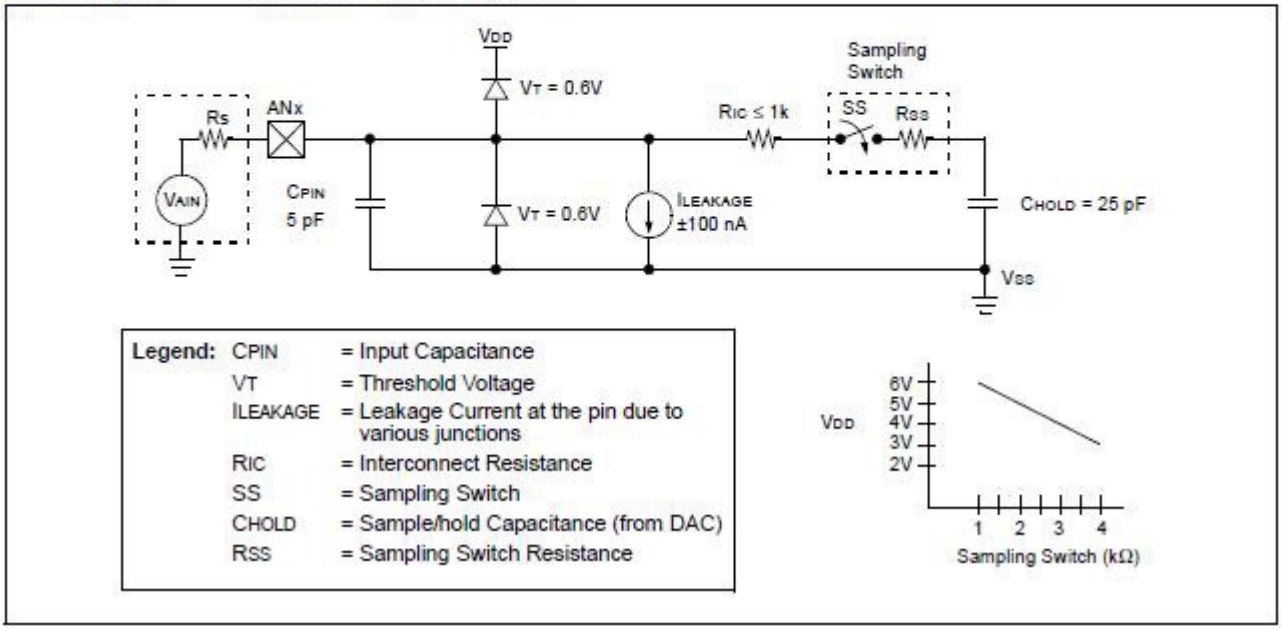


FIGURE 21-3: ANALOG INPUT MODEL



PIC18F2455/2550/4455/4550

21.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 21-3. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD). The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is 2.5 kΩ.** After the analog input channel is selected (changed), the channel must be sampled for at least the minimum acquisition time before starting a conversion.

Note:	When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 21-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Example 21-3 shows the calculation of the minimum required acquisition time TACQ. This calculation is based on the following application system assumptions:

CHOLD	=	25 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSb
VDD	=	5V → Rss = 2 kΩ
Temperature	=	85°C (system max.)

EQUATION 21-1: ACQUISITION TIME

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$

EQUATION 21-2: A/D MINIMUM CHARGING TIME

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$
or
$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2048)$$

EQUATION 21-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

$T_{AMP} = 0.2\ \mu s$

$T_{COFF} = (\text{Temp} - 25°C)(0.02\ \mu s/°C)$
$(50°C - 25°C)(0.02\ \mu s/°C)$
$1.2\ \mu s$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, $T_{COFF} = 0$ ms.

$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2047)\ \mu s$
$-(25\ pF)(1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega) \ln(0.0004883)\ \mu s$
$5.03\ \mu s$

$T_{ACQ} = 0.2\ \mu s + 5\ \mu s + 1.2\ \mu s$
$6.4\ \mu s$

PIC18F2455/2550/4455/4550

21.2 Selecting and Configuring Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/DONE bit is set. It also gives users the option to use an automatically determined acquisition time.

Acquisition time may be set with the ACQT2:ACQT0 bits (ADCON2<5:3>), which provide a range of 2 to 20 TAD. When the GO/DONE bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there may be no need to wait for an acquisition time between selecting a channel and setting the GO/DONE bit.

Manual acquisition is selected when ACQT2:ACQT0 = 000. When the GO/DONE bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/DONE bit. This option is also the default Reset state of the ACQT2:ACQT0 bits and is compatible with devices that do not offer programmable acquisition times.

In either case, when the conversion is completed, the GO/DONE bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

21.3 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. There are seven possible options for TAD:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal RC Oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be as short as possible but greater than the minimum TAD (see parameter 130 in Table 28-29 for more information).

Table 21-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

TABLE 21-1: TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18FXXXX	PIC18LFXXXX(4)
2 TOSC	000	2.86 MHz	1.43 kHz
4 TOSC	100	5.71 MHz	2.86 MHz
8 TOSC	001	11.43 MHz	5.72 MHz
16 TOSC	101	22.86 MHz	11.43 MHz
32 TOSC	010	40.0 MHz	22.86 MHz
64 TOSC	110	40.0 MHz	22.86 MHz
RC(3)	x11	1.00 MHz(1)	1.00 MHz(2)

Note 1: The RC source has a typical TAD time of 4 ms.
2: The RC source has a typical TAD time of 6 ms.
3: For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.
4: Low-power devices only.

PIC18F2455/2550/4455/4550

21.6 A/D Conversions

Figure 21-4 shows the operation of the A/D converter after the GO/$\overline{\text{DONE}}$ bit has been set and the ACQT2:ACQT0 bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 21-5 shows the operation of the A/D converter after the GO/$\overline{\text{DONE}}$ bit has been set, the ACQT2:ACQT0 bits are set to '010' and selecting a 4 TAD acquisition time before the conversion starts.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. This means the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

Note: The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

21.7 Discharge

The discharge phase is used to initialize the value of the capacitor array. The array is discharged before every sample. This feature helps to optimize the unity-gain amplifier as the circuit always needs to charge the capacitor array, rather than charge/discharge based on previous measurement values.

FIGURE 21-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

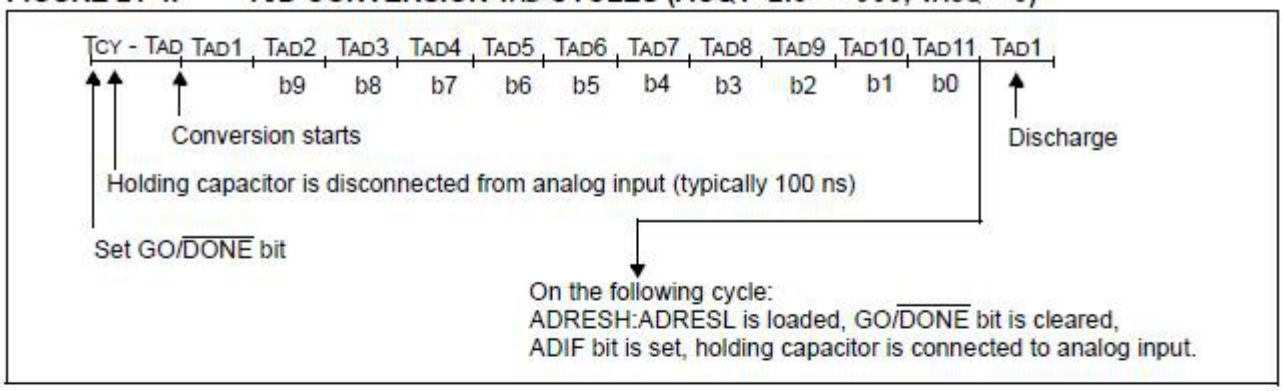
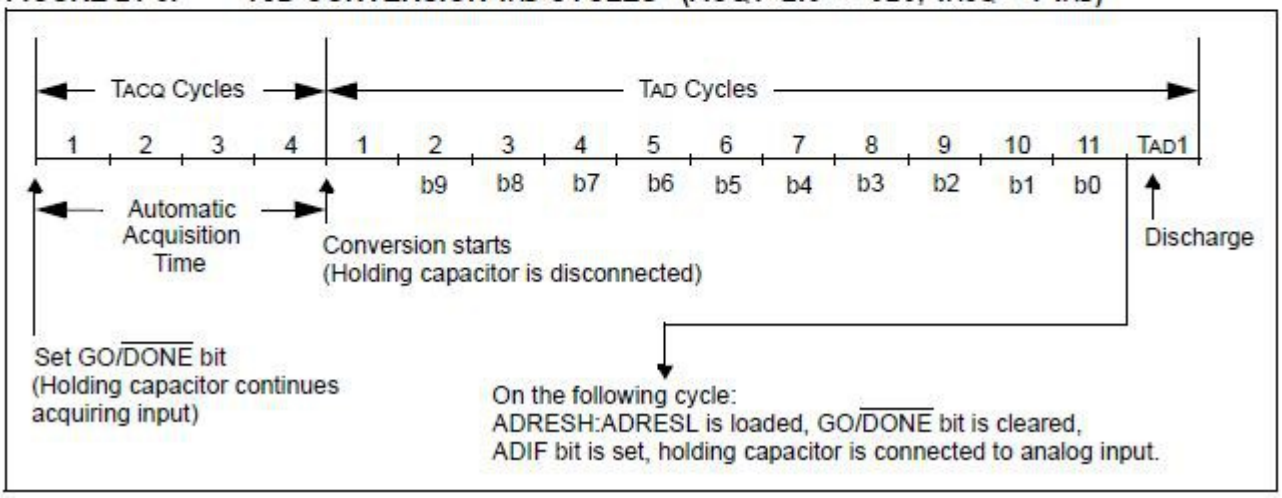


FIGURE 21-5: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)



4. INSTRUCTIONS SET

Mnemonic, Operands		Description	Cycles	14-bit Opcode MSb	14-bit Opcode LSb	Status Affected	Notes
Byte-Oriented File Register Operations							
ADDWF	f,d,a	Add WREG and f	1	0010 01da	ffff ffff	C,DC,Z,OV,N	1,2
ADDWFC	f,d,a	Add WREG and C bit to f	1	0010 00da	ffff ffff	C,DC,Z,OV,N	1,2
ANDWF	f,d,a	AND WREG and f	1	0001 01da	ffff ffff	Z,N	1,2
CLRF	f,a	Clear f	1	0110 101a	ffff ffff	Z	2
COMF	f,d,a	Complement f	1	0001 11da	ffff ffff	Z,N	1,2
CPFSEQ	f,d,a	Compare f with WREG, Skip =	1 (2 or 3)	0110 001a	ffff ffff		4
CPFSGT	f,d,a	Compare f with WREG, Skip >	1 (2 or 3)	0110 010a	ffff ffff		4
CPFSLT	f,d,a	Compare f with WREG, Skip <	1 (2 or 3)	0110 000a	ffff ffff		4
DECF	f,d,a	Decrement f	1	0000 01da	ffff ffff	C,DC,Z,OV,N	1,2,3,4
DECFSZ	f,d,a	Decrement f, Skip if 0	1 (2 or 3)	0010 11da	ffff ffff		1,2,3,4
DCFSNZ	f,d,a	Decrement f, Skip if not 0	1 (2 or 3)	0100 11da	ffff ffff		1,2
INCF	f,d,a	Increment f	1	0010 10da	ffff ffff	C,DC,Z,OV,N	1,2,3,4
INCFSZ	f,d,a	Increment f, Skip if 0	1 (2 or 3)	0011 11da	ffff ffff		4
INCFSNZ	f,d,a	Increment f, Skip if not 0	1 (2 or 3)	0100 10da	ffff ffff		1,2
IORWF	f,d,a	Inclusive OR W with f	1	0001 00da	ffff ffff	Z,N	1,2
MOVF	f,d,a	Move f	1	0101 00da	ffff ffff	Z,N	1
MOVFF	f_s,f_d	Move f_s (source – 1st word) to f_d (destination – 2nd word)	2	1100 ffff	ffff ffff 1111 ffff	ffff ffff	
MOVWF	f,a	Move WREG to f	1	0110 111a	ffff ffff		
MULWF	f,a	Multiply WREG with f	1	0000 001a	ffff ffff		
NEGF	f,d,a	Negate f	1	0110 110a	ffff ffff	C,DC,Z,OV,N	1,2
RLCF	f,d,a	Rotate Left f through Carry	1	0011 01da	ffff ffff	C,Z,N	
RLNCF	f,d,a	Rotate Left f (no Carry)	1	0100 01da	ffff ffff	Z,N	1,2
RRCF	f,d,a	Rotate Right f through Carry	1	0011 00da	ffff ffff	C,Z,N	
RRNCF	f,d,a	Rotate Right f (no Carry)	1	0100 00da	ffff ffff	Z,N	1,2
SETF	f,a	Set f	1	0110 100a	ffff ffff		
SUBFWB	f,d,a	Subtract f from WREG with Borrow	1	0101 01da	ffff ffff	C,DC,Z,OV,N	1,2
SUBWF	f,d,a	Subtract WREG from f	1	0101 11da	ffff ffff	C,DC,Z,OV,N	
SUBWFB	f,d,a	Subtract WREG from f with Borrow	1	0101 10da	ffff ffff	C,DC,Z,OV,N	1,2
SWAPF	f,d,a	Swap nibbles in f	1	0011 10da	ffff ffff		4
TSTFSZ	f,d,a	Test f, skip if 0	1 (2 or 3)	0110 011a	ffff ffff		1,2
XORWF	f,d,a	Exclusive OR W with f	1	0001 10da	ffff ffff	Z,N	
Bit-Oriented File Register Operations							
BCF	f,b,a	Bit Clear f	1	1001 bbba	ffff ffff		1,2
BSF	f,b,a	Bit Set f	1	1000 bbba	ffff ffff		1,2
BTFSC	f,b,a	Bit Test f, Skip if Clear	1 (2 or 3)	1011 bbba	ffff ffff		3,4
BTFSS	f,b,a	Bit Test f, Skip if Set	1 (2 or 3)	1010 bbba	ffff ffff		3,4
BTG	f,b,a	Bit Toggle f	1	0111 bbba	ffff ffff		1,2

Note:
1: When a PORT register is modified as a function of itself (e.g. MOVF PORTB 1, 0) the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
2: If this instruction is executed on the TMR0 register (and, where applicable, d=1) the prescaler will be cleared if assigned.
3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a nop.
4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instructions.
5: If the Table Write starts te write cycle to internal memory, the write will continue until terminated.

Mnemonic, Operands		Description	Cycles	14-bit Opcode MSb LSb	Status Affected	Notes
Literal and Control Operations						
BC	n	Branch if Carry	1 (2)	1110 0010 nnnn nnnn		
BN	n	Branch if Negative	1 (2)	1110 0110 nnnn nnnn		
BNC	n	Branch if Not Carry	1 (2)	1110 0011 nnnn nnnn		
BNN	n	Branch if Not Negative	1 (2)	1110 0111 nnnn nnnn		
BNOV	n	Branch if Not Overflow	1 (2)	1110 0101 nnnn nnnn		
BNZ	n	Branch if Not Zero	1 (2)	1110 0001 nnnn nnnn		
BOV	n	Branch if Overflow	1 (2)	1110 0100 nnnn nnnn		
BRA	n	Branch Unconditionally	1 (2)	1101 0nnn nnnn nnnn		
BZ	n	Branch if Zero	1 (2)	1110 0000 nnnn nnnn		
CALL	k,s	Call subroutine	2	1110 110s kkkk kkkk 1111 kkkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	0000 0000 0000 0100	$\overline{TO}$,$\overline{PD}$	
DAW	-	Decimal Adjust WREG	1	0000 0000 0000 0111	C	
GOTO	k	Go to address	2	1110 1111 kkkk kkkk 1111 kkkk kkkk kkkk		
NOP	-	No Operation	1	0000 0000 0000 0000		
NOP	-	No Operation	1	1111 xxxx xxxx xxxx		
POP	-	Pop top of return stack (TOS)	1	0000 0000 0000 0110		
PUSH	-	Push top of return stack (TOS)	1	0000 0000 0000 0101		
RCALL	n	Relative Call	2	1101 1nnn nnnn nnnn		
RESET	-	Software device RESET	1	0000 0000 1111 1111	All	
RETFIE	s	Return from Interrupt	2	0000 0000 0001 000s	GIE,PEIE	
RETLW	k	Return with literal in W	2	0000 1100 kkkk kkkk		
RETURN		Return from Subroutine	2	0000 0000 0001 001s		
SLEEP	-	Go into Standby mode	1	0000 0000 0000 0011	$\overline{TO}$,$\overline{PD}$	
Literal Operations						
ADDLW	k	Add literal to WREG	1	0000 1111 kkkk kkkk	C,DC,Z,OV,N	
ANDLW	k	AND literal with WREG	1	0000 1011 kkkk kkkk	Z,N	
IORLW	k	Inclusive OR literal with WREG	1	0000 1001 kkkk kkkk	Z,N	
LFSR	n,k	Move literal (12-bit) to FSRn	1	1110 1110 00nn kkkk 1111 0000 kkkk kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000 0001 0000 kkkk		
MOVLW	k	Move literal to WREG	1	0000 1110 kkkk kkkk		
MULLW	k	Multiply literal with W	1	0000 1101 kkkk kkkk		
SUBLW	k	Subtract W from literal	1	0000 1000 kkkk kkkk	C,DC,Z,OV,N	
XORLW	k	Exclusive OR literal with W	1	0000 1010 kkkk kkkk	Z	
Data Memory ⇔ Program Memory Operations						
TBLRD*		Table Read	2	0000 0000 0000 1000		
TBLRD*+		Table Read with Post-Increment	2	0000 0000 0000 1001		
TBLRD*-		Table Read with Post-Decrement	2	0000 0000 0000 1010		
TBLRD+*		Table Read with Pre-Increment	2	0000 0000 0000 1011		
TBLWT*		Table Write	2 (5)	0000 0000 0000 1100		
TBLWT*+		Table Write with Post-Increment	2 (5)	0000 0000 0000 1101		
TBLWT*-		Table Write with Post-Decrement	2 (5)	0000 0000 0000 1110		
TBLWT+*		Table Write with Pre-Increment	2 (5)	0000 0000 0000 1111		

OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU status bits: Carry, Digit Carry, Zero, Overflow, Negative.
d	Destination select bit d = 0: store result in WREG d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
f_s	12-bit register file address (000h to FFFh). This is the source address.
f_d	12-bit register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm * *+ *- +*	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions: No change to register (such as TBLPTR with table reads and writes) Post-Increment register (such as TBLPTR with table reads and writes) Post-Decrement register (such as TBLPTR with table reads and writes) Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for Call/Branch and Return instructions.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
$\overline{PD}$	Power-Down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a program memory location).
TABLAT	8-bit Table Latch.
$\overline{TO}$	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
x	Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
z_s	7-bit offset value for indirect addressing of register files (source).
z_d	7-bit offset value for indirect addressing of register files (destination).
{ }	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr]<n>	Specifies bit n of the register indicated by the pointer expr.
→	Assigned to.
< >	Register bit field.
∈	In the set of.
italics	User defined term (font is Courier).

5. INSTRUCTIONS SET DESCRIPTIONS

ADDLW — **ADD Literal to W**

Syntax: ADDLW k

Operands: $0 \le k \le 255$

Operation: $(W) + k \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `ADDLW 15h`

Before Instruction
W = 10h
After Instruction
W = 25h

BCF — **Bit Clear f**

Syntax: BCF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

ANDLW — **AND Literal with W**

Syntax: ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. $k \rightarrow W$

Status Affected: N, Z

Encoding:

0000	1011	kkkk	kkkk

Description: The contents of W are ANDed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `ANDLW 05Fh`

Before Instruction
W = A3h
After Instruction
W = 03h

BRA — **Unconditional Branch**

Syntax: BRA n

Operands: $-1024 \le n \le 1023$

Operation: $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1101	0nnn	nnnn	nnnn

Description: Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE BRA Jump`

Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

BSF **Bit Set f**

Syntax: BSF f, b {,a}

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 1 → f<b>

Status Affected: None

Encoding:

1000	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is set.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BSF FLAG_REG, 7, 1`

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

CLRF **Clear f**

Syntax: CLRF f {,a}

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: 000h → f
1 → Z

Status Affected: Z

Encoding:

0110	101a	ffff	ffff

Description: Clears the contents of the specified register.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `CLRF FLAG_REG,1`

Before Instruction
FLAG_REG = 5Ah
After Instruction
FLAG_REG = 00h

CALL **Subroutine Call**

Syntax: CALL k {,s}

Operands: 0 ≤ k ≤ 1048575
s ∈ [0,1]

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(W) → WS,
(Status) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	110s	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	Push PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE,1`

Before Instruction
PC = address (HERE)
After Instruction
PC = address (THERE)
TOS = address (HERE + 4)
WS = W
BSRS = BSR
STATUSS = Status

DECFSZ — **Decrement f, Skip if 0**

Syntax: DECFSZ f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0010	11da	ffff	ffff

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    DECFSZ  CNT, 1, 1
        GOTO    LOOP
CONTINUE
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT – 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE + 2)

GOTO — **Unconditional Branch**

Syntax: GOTO k

Operands: 0 ≤ k ≤ 1048575

Operation: k → PC<20:1>

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	1111	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description: GOTO allows an unconditional branch anywhere within the entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example:

```
GOTO THERE
```

After Instruction
PC = Address (THERE)

MOVF — **Move f**

Syntax: MOVF f {,d {,a}}

Operands: $0 \le f \le 255$, $d \in [0,1]$, $a \in [0,1]$

Operation: f → dest

Status Affected: N, Z

Encoding:

0101	00da	ffff	ffff

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write W

Example: `MOVF REG, 0, 0`

Before Instruction
REG = 22h
W = FFh
After Instruction
REG = 22h
W = 22h

MOVFF — **Move f to f**

Syntax: MOVFF f_s,f_d

Operands: $0 \le f_s \le 4095$, $0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	1100	ffff	ffff	$ffff_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh.
Either source or destination can be W (a useful special situation).
`MOVFF` is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).
The `MOVFF` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: `MOVFF REG1, REG2`

Before Instruction
REG1 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

MOVLW **Move literal to W**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk

Description: The eight-bit literal 'k' is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `MOVLW 5Ah`

After Instruction
W = 5Ah

MULLW **Multiply Literal with W**

Syntax: MULLW k

Operands: $0 \le k \le 255$

Operation: (W) x k → PRODH:PRODL

Status Affected: None

Encoding:

0000	1101	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte.
W is unchanged.
None of the Status flags are affected.
Note that neither Overflow nor Carry is possible in this operation. A zero result is possible but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: `MULLW 0C4h`

Before Instruction
W = E2h
PRODH = ?
PRODL = ?
After Instruction
W = E2h
PRODH = ADh
PRODL = 08h

MOVWF **Move W to f**

Syntax: MOVWF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (W) → f

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `MOVWF REG, 0`

Before Instruction
W = 4Fh
REG = FFh
After Instruction
W = 4Fh
REG = 4Fh

NOP **No Operation**

Syntax: NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

0000 1111	0000 xxxx	0000 xxxx	0000 xxxx

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

POP — **Pop Top of Return Stack**

Syntax: POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

0000	0000	0000	0110

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Pop TOS value	No operation

Example:

```
POP
GOTO    NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

PUSH — **Push Top of Return Stack**

Syntax: PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding:

0000	0000	0000	0101

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Push PC + 2 onto return stack	No operation	No operation

Example:

```
PUSH
```

Before Instruction
TOS = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah

RETFIE — **Return from Interrupt**

Syntax: RETFIE {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

0000	0000	0001	000s

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	Pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example:

```
RETFIE  1
```

After Interrupt
PC = TOS
W = WS
BSR = BSRS
Status = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETURN — **Return from Subroutine**

Syntax: RETURN {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	0000	0001	001s

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Pop PC from stack
No operation	No operation	No operation	No operation

Example:
```
RETURN
```
After Instruction:
PC = TOS

SETF — **Set f**

Syntax: SETF f {,a}

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example:
```
SETF    REG,1
```
Before Instruction
REG = 5Ah
After Instruction
REG = FFh

RRNCF — **Rotate Right f (no carry)**

Syntax: RRNCF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:
```
RRNCF   REG, 1, 0
```
Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2:
```
RRNCF   REG, 0, 0
```
Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

6. DATASHEET HD44780U

HD44780U (LCD-II)

(Dot Matrix Liquid Crystal Display Controller/Driver)

HITACHI

Description

The HD44780U dot-matrix liquid crystal display controller and driver LSI displays alphanumerics, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.

A single HD44780U can display up to one 8-character line or two 8-character lines.

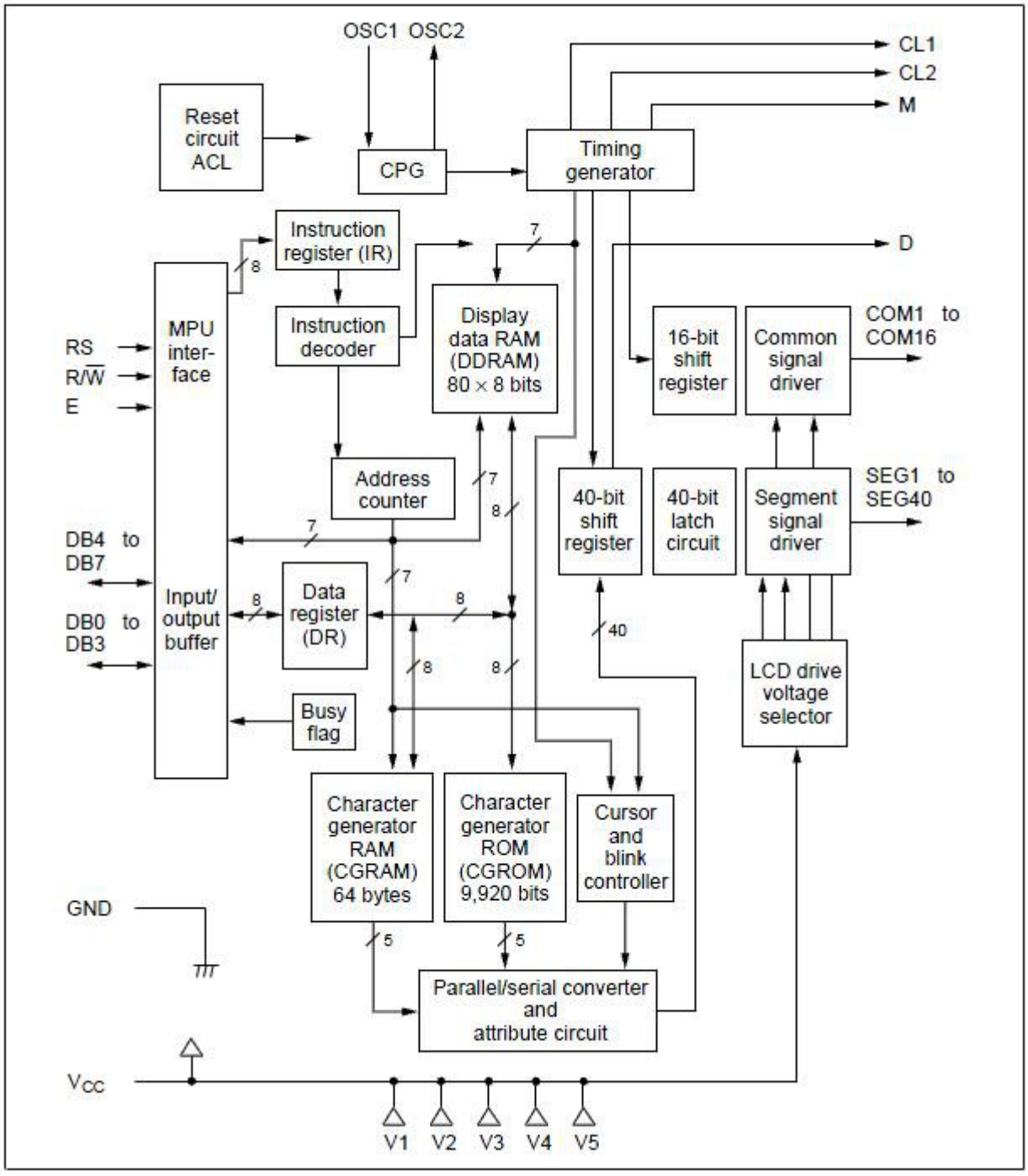
The HD44780U has pin function compatibility with the HD44780S which allows the user to easily replace an LCD-II with an HD44780U. The HD44780U character generator ROM is extended to generate 208 5 × 8 dot character fonts and 32 5 × 10 dot character fonts for a total of 240 different character fonts.

The low power supply (2.7V to 5.5V) of the HD44780U is suitable for any portable battery-driven product requiring low power dissipation.

Features

- 5 × 8 and 5 × 10 dot matrix possible
- Low power operation support:
  — 2.7 to 5.5V
- Wide range of liquid crystal display driver power
  — 3.0 to 11V
- Liquid crystal drive waveform
  — A (One line frequency AC waveform)
- Correspond to high speed MPU bus interface
  — 2 MHz (when V_{CC} = 5V)
- 4-bit or 8-bit MPU interface enabled
- 80 × 8-bit display RAM (80 characters max.)
- 9,920-bit character generator ROM for a total of 240 character fonts
  — 208 character fonts (5 × 8 dot)
  — 32 character fonts (5 × 10 dot)

HD44780U Block Diagram



Reset Function

Initializing by Internal Reset Circuit

An internal reset circuit automatically initializes the HD44780U when the power is turned on. The following instructions are executed during the initialization. The busy flag (BF) is kept in the busy state until the initialization ends (BF = 1). The busy state lasts for 10 ms after V_{CC} rises to 4.5 V.

1. Display clear
2. Function set:
   DL = 1; 8-bit interface data
   N = 0; 1-line display
   F = 0; 5 × 8 dot character font
3. Display on/off control:
   D = 0; Display off
   C = 0; Cursor off
   B = 0; Blinking off
4. Entry mode set:
   I/D = 1; Increment by 1
   S = 0; No shift

Note: If the electrical characteristics conditions listed under the table Power Supply Conditions Using Internal Reset Circuit are not met, the internal reset circuit will not operate normally and will fail to initialize the HD44780U. For such a case, initial-ization must be performed by the MPU as explained in the section, Initializing by Instruction.

Instructions

Outline

Only the instruction register (IR) and the data register (DR) of the HD44780U can be controlled by the MPU. Before starting the internal operation of the HD44780U, control information is temporarily stored into these registers to allow interfacing with various MPUs, which operate at different speeds, or various peripheral control devices. The internal operation of the HD44780U is determined by signals sent from the MPU. These signals, which include register selection signal (RS), read/

write signal (R/$\overline{W}$), and the data bus (DB0 to DB7), make up the HD44780U instructions (Table 6). There are four categories of instructions that:

- Designate HD44780U functions, such as display format, data length, etc.
- Set internal RAM addresses
- Perform data transfer with internal RAM
- Perform miscellaneous functions

HD44780U

Normally, instructions that perform data transfer with internal RAM are used the most. However, auto-incrementation by 1 (or auto-decrementation by 1) of internal HD44780U RAM addresses after each data write can lighten the program load of the MPU. Since the display shift instruction (Table 11) can perform concurrently with display data write, the user can minimize system development time with maximum programming efficiency.

When an instruction is being executed for internal operation, no instruction other than the busy flag/address read instruction can be executed.

Because the busy flag is set to 1 while an instruction is being executed, check it to make sure it is 0 before sending another instruction from the MPU.

Note: Be sure the HD44780U is not in the busy state (BF = 0) before sending an instruction from the MPU to the HD44780U. If an instruction is sent without checking the busy flag, the time between the first instruction and next instruction will take much longer than the instruction time itself. Refer to Table 6 for the list of each instruc-tion execution time.

Table 6 Instructions

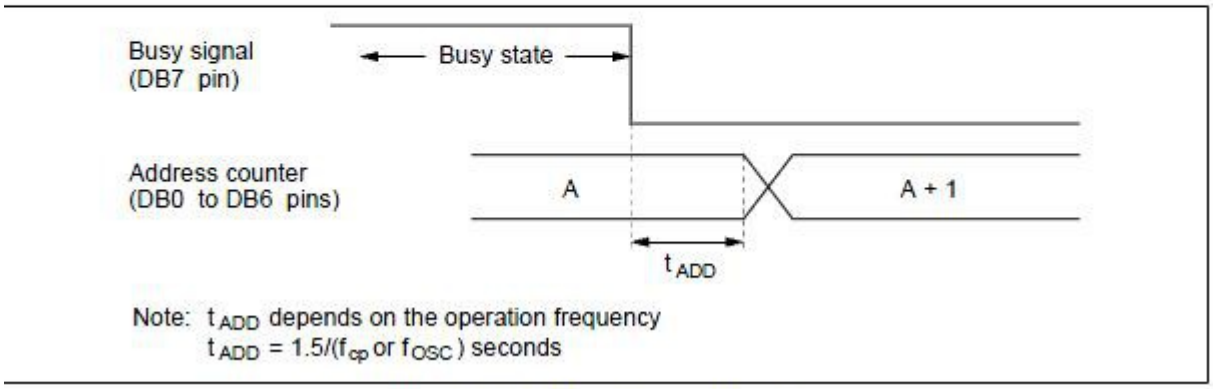
	Code											Execution Time (max) (when f_{op} or f_{osc} is 270 kHz)
Instruction	RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 µs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 µs
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 µs
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs

Table 6 Instructions (cont)

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{cp} or f_{OSC} is 270 kHz)
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 × 10 dots, F = 0: 5 × 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable										DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses	Execution time changes when frequency changes Example: When f_{cp} or f_{OSC} is 250 kHz, $37\ \mu s \times \frac{270}{250} = 40\ \mu s$

Note: — indicates no effect.

* After execution of the CGRAM/DDRAM data write or read instruction, the RAM address counter is incremented or decremented by 1. The RAM address counter is updated after the busy flag turns off. In Figure 10, t_{ADD} is the time elapsed after the busy flag turns off until the address counter is updated.



Note: t_{ADD} depends on the operation frequency
t_{ADD} = 1.5/(f_{cp} or f_{OSC}) seconds

Figure 10 Address Counter Update

HD44780U

Figure 11 Instruction (1)

- Interfacing to a 4-bit MPU

  The HD44780U can be connected to the I/O port of a 4-bit MPU. If the I/O port has enough bits, 8-bit data can be transferred. Otherwise, one data transfer must be made in two operations for 4-bit data. In this case, the timing sequence becomes somewhat complex. (See Figure 17.)

  See Figure 18 for an interface example to the HMCS4019R.

  Note that two cycles are needed for the busy flag check as well as for the data transfer. The 4-bit operation is selected by the program.

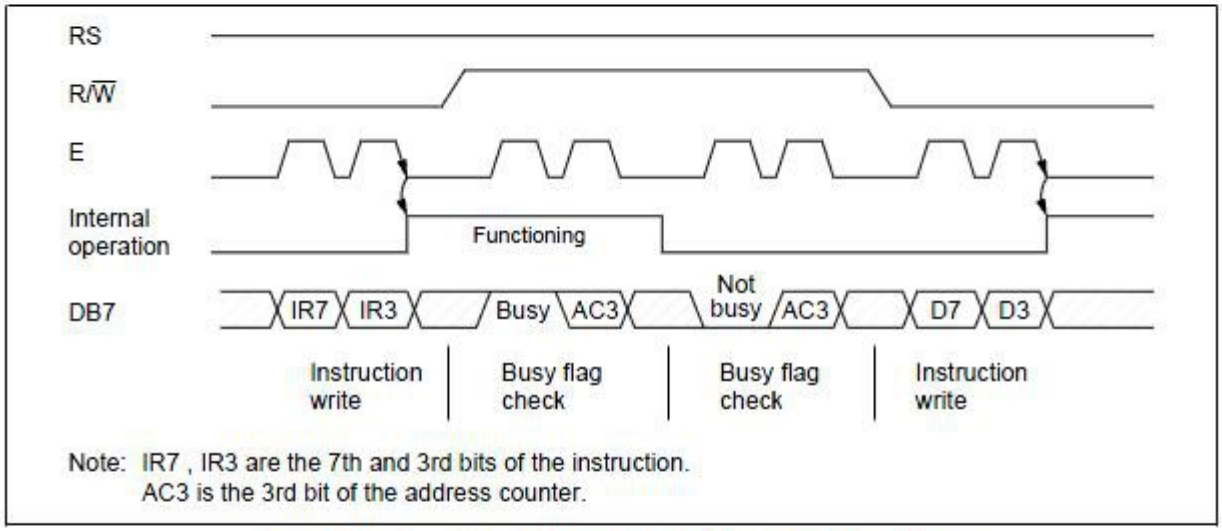


Figure 17 Example of 4-Bit Data Transfer Timing Sequence

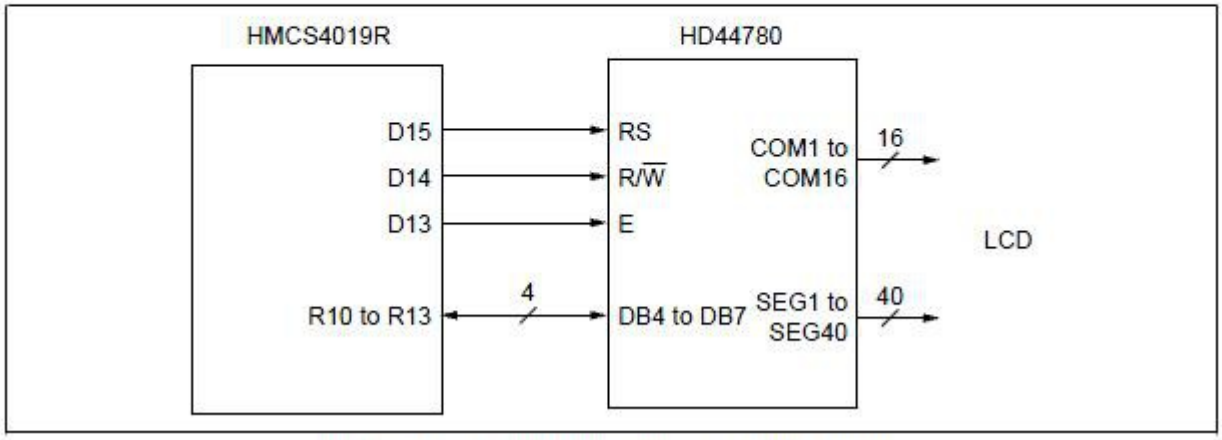


Figure 18 Example of Interface to HMCS4019R

HD44780U

Table 12 4-Bit Operation, 8-Digit × 1-Line Display Example with Internal Reset

Step No.	Instruction RS	R/W	DB7	DB6	DB5	DB4	Display	Operation
1	Power supply on (the HD44780U is initialized by the internal reset circuit)							Initialized. No display.
2	Function set							Sets to 4-bit operation. In this case, operation is handled as 8 bits by initialization, and only this instruction completes with one write.
	0	0	0	0	1	0		
3	Function set							Sets 4-bit operation and selects 1-line display and 5 × 8 dot character font. 4-bit operation starts from this step and resetting is necessary. (Number of display lines and character fonts cannot be changed after step #3.)
	0	0	0	0	1	0		
	0	0	0	0	*	*		
4	Display on/off control						_	Turns on display and cursor. Entire display is in space mode because of initialization.
	0	0	0	0	0	0		
	0	0	1	1	1	0		
5	Entry mode set						_	Sets mode to increment the address by one and to shift the cursor to the right at the time of write to the DD/CGRAM. Display is not shifted.
	0	0	0	0	0	0		
	0	0	0	1	1	0		
6	Write data to CGRAM/DDRAM						H_	Writes H. The cursor is incremented by one and shifts to the right.
	1	0	0	1	0	0		
	1	0	1	0	0	0		

Note: The control is the same as for 8-bit operation beyond step #6.

Initializing by Instruction

If the power supply conditions for correctly operating the internal reset circuit are not met, initialization by instructions becomes necessary.

Refer to Figures 23 and 24 for the procedures on 8-bit and 4-bit initializations, respectively.

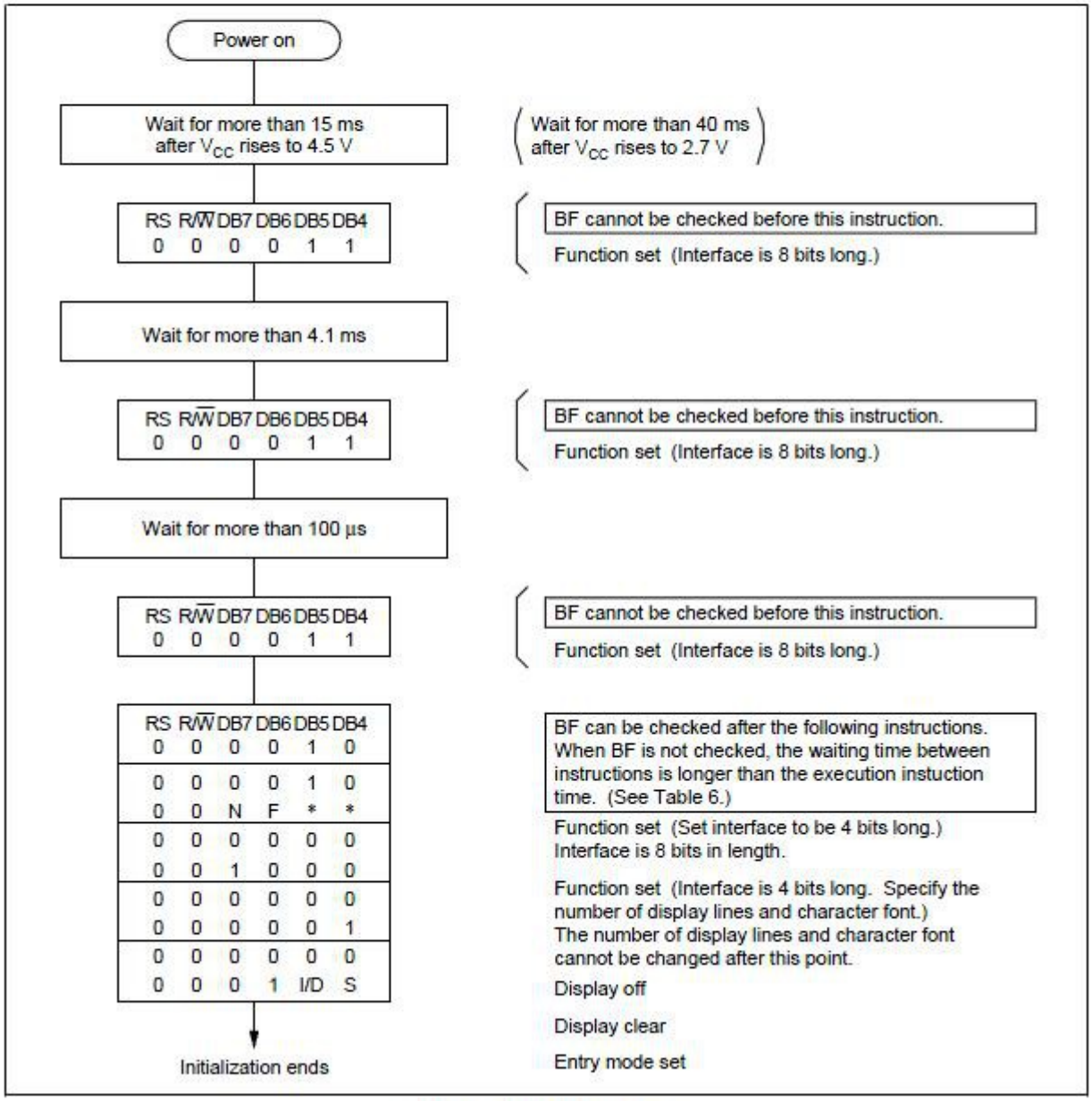


Figure 24 4-Bit Interface

Timing Characteristics

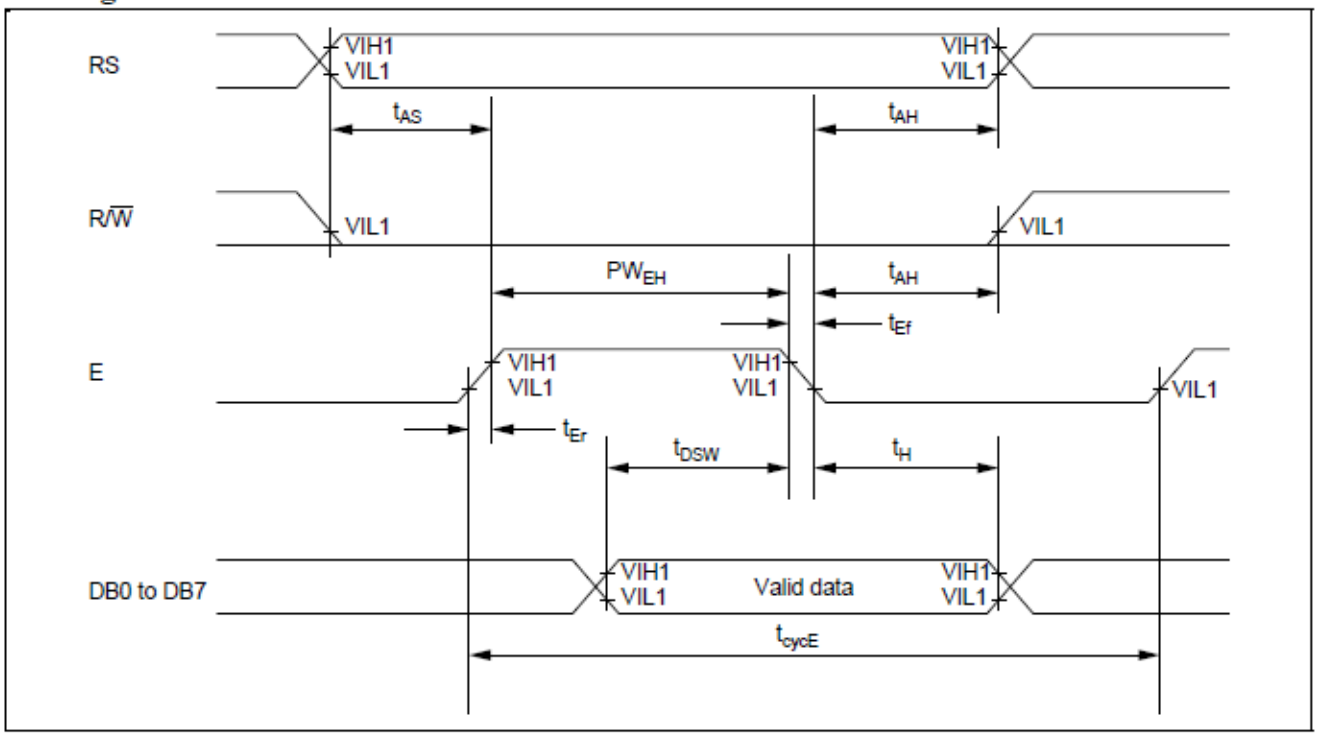


Figure 25 Write Operation

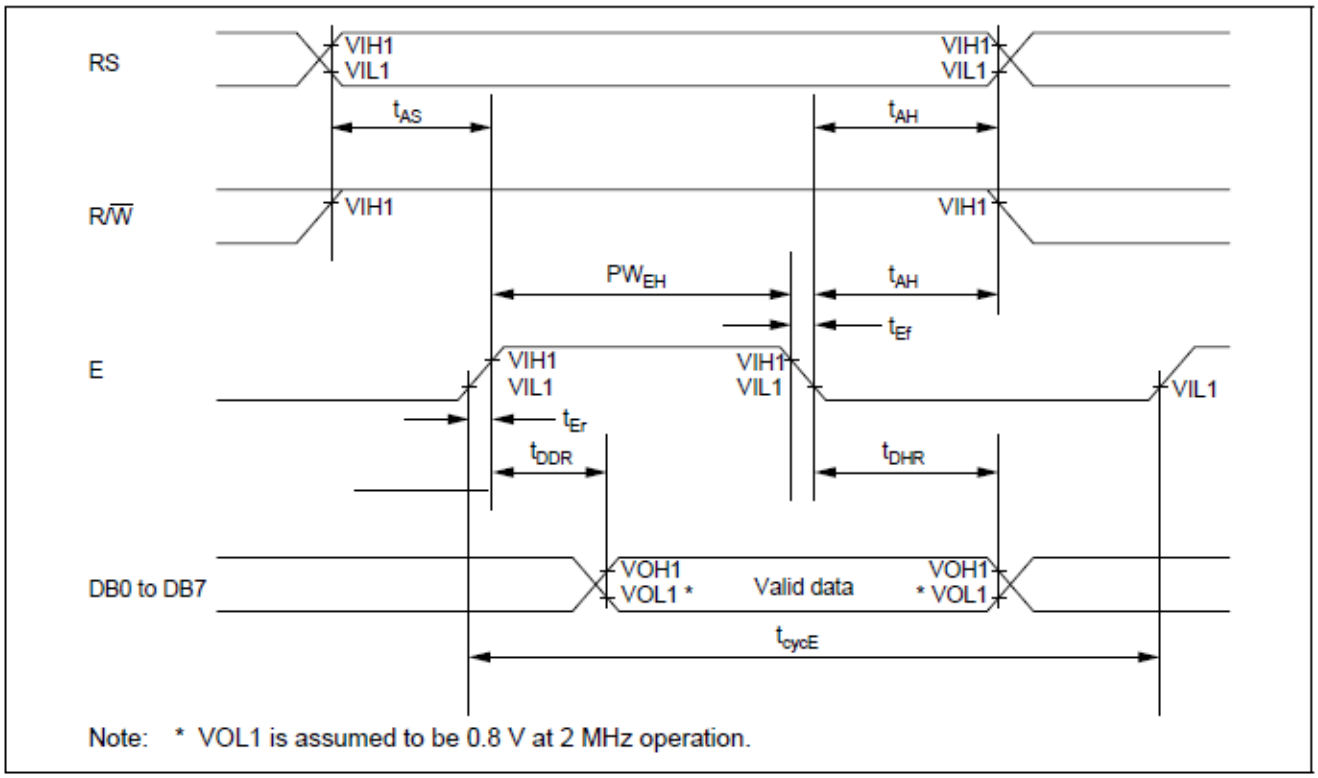


Figure 26 Read Operation

HD44780U

AC Characteristics (V_{CC} = 4.5 to 5.5 V, T_a = –30 to +75°C*[3])

Clock Characteristics

Item		Symbol	Min	Typ	Max	Unit	Test Condition	Notes*
External clock operation	External clock frequency	f_{cp}	125	250	350	kHz		11
	External clock duty	Duty	45	50	55	%		11
	External clock rise time	t_{rcp}	—	—	0.2	µs		11
	External clock fall time	t_{fcp}	—	—	0.2	µs		11
R_f oscillation	Clock oscillation frequency	f_{OSC}	190	270	350	kHz	R_f = 91 kΩ V_{CC} = 5.0 V	12

Note: * Refer to the Electrical Characteristics Notes section following these tables.

Bus Timing Characteristics

Write Operation

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Enable cycle time	t_{cycE}	500	—	—	ns	Figure 25
Enable pulse width (high level)	PW_{EH}	230	—	—		
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	20		
Address set-up time (RS, R/$\overline{W}$ to E)	t_{AS}	40	—	—		
Address hold time	t_{AH}	10	—	—		
Data set-up time	t_{DSW}	80	—	—		
Data hold time	t_H	10	—	—		

Read Operation

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Enable cycle time	t_{cycE}	500	—	—	ns	Figure 26
Enable pulse width (high level)	PW_{EH}	230	—	—		
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	20		
Address set-up time (RS, R/$\overline{W}$ to E)	t_{AS}	40	—	—		
Address hold time	t_{AH}	10	—	—		
Data delay time	t_{DDR}	—	—	160		
Data hold time	t_{DHR}	5	—	—		

7. TABLE ASCII

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC		‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	Š 0160	‹ 2039	Œ 0152		Ž 017D	
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	˜ 02DC	™ 2122	š 0161	› 203A	œ 0153		ž 017E	Ÿ 0178
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

REGLES DE CODAGE

Ce document à pour objectif de fixer un cadre et des règles de codage en langage C pour les enseignements de Systèmes Embarqués à l'ENSICAEN. Cette démarche reflète les contraintes que vous aurez d'ici quelques années à suivre dans le monde de l'entreprise. La plupart des entreprises travaillant dans le domaine du développement logiciel, dont l'embarqué fait parti, imposent à leurs développeurs des règles semblables à celles-ci (beaucoup plus exhaustives en général). L'objectif étant de standardiser, clarifier, cadrer et faciliter le partage de codes sources au sein d'une équipe voir de l'entreprise. Les règles de codage imposées sont inspirées du ''Linux Kernel Coding Style'' et du ''GNU Coding Standard''. Voici le sommaire de ce document :

INDENTATION

- ***Tabulation :*** 8 caractères (à régler dans les préférences de l'éditeur de texte).

```
switch (data) {
case 'A':
case 'B':
	/* comment */
	data <<= 20;
	break;
default:
	break;
}
```

- ***Nombres de niveaux d'indentation :*** Éviter plus de 3 niveaux d'indentation imbriqués. Facilite la lisibilité du code.

- ***Nombres de colonnes par page :*** 80 caractères/colonnes par page (à régler dans les préférences de l'éditeur de texte).

- ***procédures :*** cas spécial pour le fonctions, ouvrir l'accolade au début de la ligne suivante

```
int function (int x, int y)
{
	int z ;
	...
	return 0;
}
```

- ***Structures de contrôle :*** règles spécifiant l'indentation ainsi que les espaces à respecter pour l'écriture de structures de contrôle.

```
do {
    if (a == b) {
        ...
    } else if (a > b) {
        ...
    } else {
        ...
    }
    /* single statement */
    if (condition)
        action1();
    else
        action2();

} while (condition);
```

DENOMINATION

- ***Fonctions :*** Toujours débuter par une minuscule. Pour donner un nom complexe, utiliser comme séparateur une Majuscule ou un ''_''.

- ***Variables locales :*** Toujours débuter par une minuscule. Les variables servant de compteur de boucle peuvent avoir un nom sans sens précis, par exemple ''i''. Sinon, toujours donner à une variable un nom court permettant d'identifier clairement son rôle. Déclarez vos variables locales en début de fonction (portabilité de code).

- ***Variables globales :*** Toujours débuter par une minuscule. Toujours donner à une variable globale un nom permettant d'identifier clairement son rôle (séparateurs ''_'' ou Majuscules). Bien évidemment, éviter tant que faire se peut les variables globales (ressources partagées).

```
int tempProcessA, temp_process_B;

void lm4567_codec_ init (void)
{
    temp_process_B++;
    …
}
```

- ***Définition de type :*** Toujours débuter par une Majuscule. Toujours définir un nom permettant d'identifier clairement le type des variables déclarés.

```
HandleSpiModule  a;      // Bien !
Hmod  a;                 // Pas Bien !
```

- ***Macros :*** Toujours en Majuscule. Utiliser comme séparateur un ''_'' pour les noms complexes. Pour les macros fonctions, appliquer les même règles de dénomination que pour les fonctions.

```
#define  UART_BAUDRATE   0xFFFF
#define  mul (x, y)      x * y
```

COMMENTAIRES

- ***Toujours penser à expliquer ce que votre code fait et non comment il le fait !***
- ***C99 style :*** Ne pas utiliser ''**//**'' (problème de portabilité)
- ***C89 style :*** Toujours utiliser ''**/* ... */**'' (solution portable)
- ***Balise pour la documentation de code (exemple de Doxygen) :*** Insérer dans vos cartouches quelques tags standards, par exemple ceux de Doxygen. Attention, seuls les fichiers d'en-tête doivent contenir des balises pour la génération de documentation. Doxygen permet la génération automatique de documentation vers différents formats (HTML, PDF, CHM ...). Ne pas utiliser de caractères accentués dans vos commentaires.

```
/**
 * @file example.h
 * @brief demo header file
 * @author
 */

int temp_conv;       /*  converted data from temperature sensor */

/**
 * @struct  Str_data_buffer
 * @brief Objet latch converted data from temperature sensor
 */
typedef struct {
      int buffSize
      int* allocCnvTab
} Str_data_buffer;

/**
 * @brief read data codec LM4567 controller
 * @param resetVal imput value
 * @param  cnvResult converted value
 * @return null if data conversion error
 */
Handle_spi_module lm4567_codec_read (char resetVal, float cnvResult);
```

- Quelques tags supplémentaires (cf. ***www.doxygen.org***):

```
/**
 * @example example insertion
 * @warning warning insertion
 * @li bullet point insertion
 * @mainpage main page comments insertion
 * @image  picture insertion
 * @include code insertion
 */
```

DIVERS

- ***Valeur de retour :*** Essayer de faire en sorte que la valeur de retour d'une fonction soit représentative de son bon traitement. Par exemple, retourne zéro (ou pointeur nul) en cas d'échec et différent de zéro en cas de succès. Les résultats des procédures seront retournés par pointeur via les paramètres d'entrée.

GLOSSAIRE

A

- ***ABI :*** Application Binary Interface
- ***ADC :*** Analog to Digital Converter
- ***ALU :*** Arithmetic and Logical Unit
- ***AMD :*** Advanced Micro Devices
- ***ANSI :*** American National Standards Institute
- ***API :*** Application Programming Interface
- ***APU :*** Accelerrated Processor Unit
- ***ARM :*** société anglaise proposant des architectures CPU RISC 32bits
- ***ASCII :*** American Standar Code for Information Interchange

B

- ***BP :*** *Base Pointer*
- ***BSL :*** Board Support Library

C

- ***CCS :*** Code Composer Studio
- ***CEM :*** Compatibilité ElectroMagnétique
- ***CISC :*** Complex Instruction Set Computer
- ***CPU :*** Central Processing Unit
- ***CSL :*** Chip Support Library

D

- ***DAC :*** Digital to Analog Converter
- ***DDR :*** Double Data Rate
- ***DDR SDRAM:*** Double Data Rate Synchronous Dynamic Random Access Memory
- ***DMA :*** Direct Memory Access
- ***DSP :*** Digital Signal Processor
- ***DSP :*** Digital Signal Processing

E

- ***EDMA :*** Enhanced Direct Memory Access
- ***EUSART :*** Enhanced Universal Synchronous Asynchronous Receiver Transmitter
- ***EMIF :*** External Memory Interface
- ***EPIC :*** Explicitly Parallel Instruction Computing

F

- ***FPU :*** Floating Point Unit
- ***FLOPS :*** Floating-Point Operations Per Second
- ***FMA:*** Fused Multiply-Add

G

- ***GCC :*** Gnu Collection Compiler
- ***GLCD :*** Graphical Liquid Crytal Display
- ***GNU :*** GNU's Not UNIX
- ***GPIO :*** General Purpose Input Output
- ***GPP :*** General Purpose Processor
- ***GPU :*** Graphical Processing Unit

I

- ***IA-64 :*** Intel Architecture 64bits
- ***I2C :*** Inter Integrated Circuit
- ***ICC :*** Intel C++ Compiler
- ***IDE :*** Integrated Development Environment
- ***IDMA :*** Internal Direct memory Access
- ***IRQ :*** Interrupt ReQuest
- ***ISR :*** Interrupt Software Routine
- ***ISR :*** Interrupt Service Routine

L

- ***L1D :*** Level 1 Data Memory
- ***L1I :*** Level 1 Instruction Memory (idem L1P)
- ***L1P :*** Level 1 Program Memory (idem L1I)
- ***Lx :*** Level x Memory
- ***LCD :*** Liquid Crytal Display
- ***LRU :*** Least Recently Used

M

- ***MAC:*** Multiply Accumulate

- ***MCU :*** Micro Controller Unit
- ***MIMD :*** Multiple Instructions on Multiple Data
- ***MIPS :*** Mega Instructions Per Second
- ***MMU :*** Memory Managment Unit
- ***MPLABX :*** MicrochiP LABoratory 10, IDE Microchip
- ***MPU :*** Micro Processor Unit ou GPP

- ***MPU :*** Memory Protect Unit

O

- ***OS :*** Operating System

P

- ***PC :*** Program Counter
- ***PC :*** Personal Computer
- ***PIC18 :*** Famille MCU 8bits Microchip
- ***PLD :*** Programmable Logic Device
- ***POSIX :*** Portable Operating System Interface, héritage d'UNIX (norme IEEE 1003)
- ***PPC :*** Power PC

R

- ***RAM :*** Random Access Memory
- ***RISC :*** Reduced Instruction Set Computer
- ***RS232 :*** Norme standardisant un protocole de communication série asynchrone
- ***RTOS :*** Real Time Operating System

S

- ***SDK :*** Software Development Kit
- ***SIMD :*** Single Instruction Multiple Date
- ***SOB :*** System On Board
- ***SOC :*** System On Chip
- ***SOP :*** Sums of products
- ***SP :*** Stack Pointer
- ***SP :*** Serial Port
- ***SPI :*** Serial Peripheral Interface
- ***SRAM :*** Static Random Access Memory
- ***SSE :*** Streaming SIMD Extensions
- ***STM32 :*** STMicroelectronics 32bits MCU

T

- ***TI :*** Texas Instruments
- ***TNS :*** Traitement Numérique du Signal
- ***TSC :*** Time Stamp Counter
- ***TTM :*** Time To Market

U

- ***UART :*** Universal Asynchronous Receiver Transmitter
- ***USB :*** Universal Serial Bus

V

- ***VHDL :*** VHSIC Hardware Description langage
- ***VHSIC :*** Very High Speed Integrated Circuit
- ***VLIW :*** Very Long Intruction Word