CENG 336
INT. TO EMBEDDED SYSTEMS DEVELOPMENT

Spring 2006
Recitation 10

OUTLINE

- More about PIC18 Series
- Introduction to MCC18
- Introductıon to PICos18

PIC18 Series

Features:

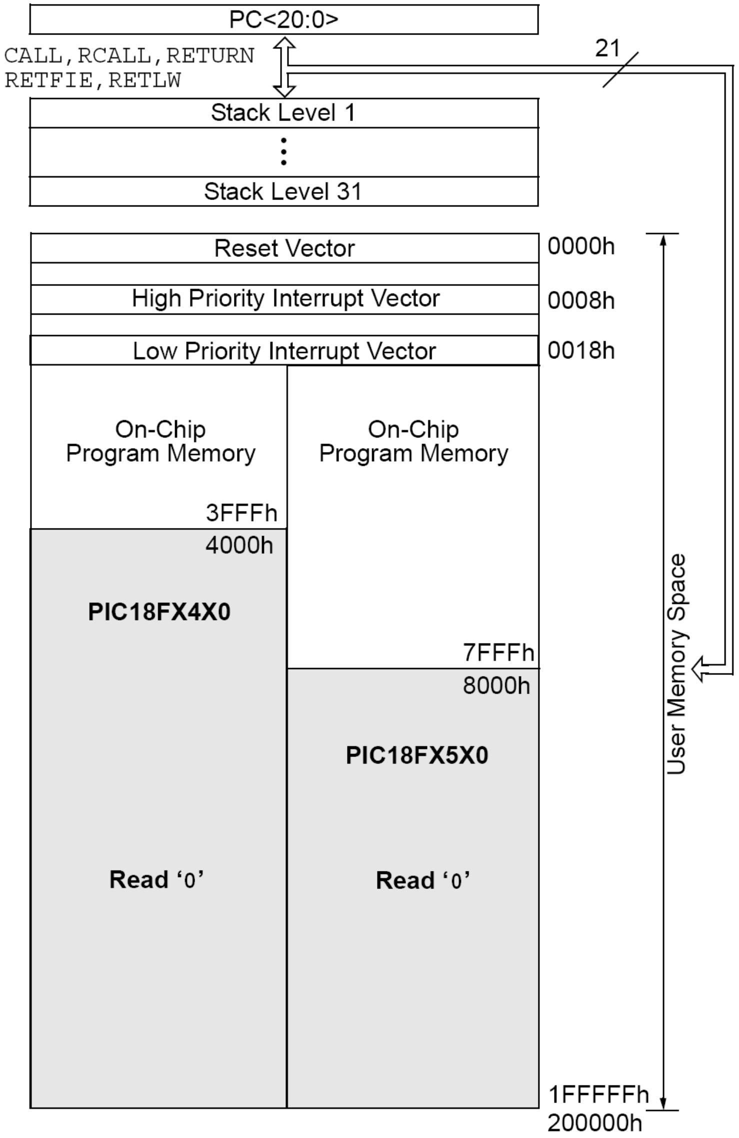
- Increased flash (16k words for the PIC18F4520)
- Increased RAM (1536 bytes for the PIC18F4520)
- Extended configuration bits
- Four Crystal modes, up to 40 MHz
- Much deeper call stack (31 levels deep)
- The call stack may be read and written
- Two levels of interrupt priority
- Enhanced memory management (can acess the SFRs without switching banks; but not the GPRs)

PIC18 Series

Features:

- New instructions
- 8x8 hardware multiplier
- Indexed addressing mode (PLUSW)
- Extending the FSR registers to 12 bits, allowing them to linearly address the entire data address space
- C has overtaken assembly
- We will use MPLAB C18 Student Edition.

Program Memory Structure



Program Memory Structure

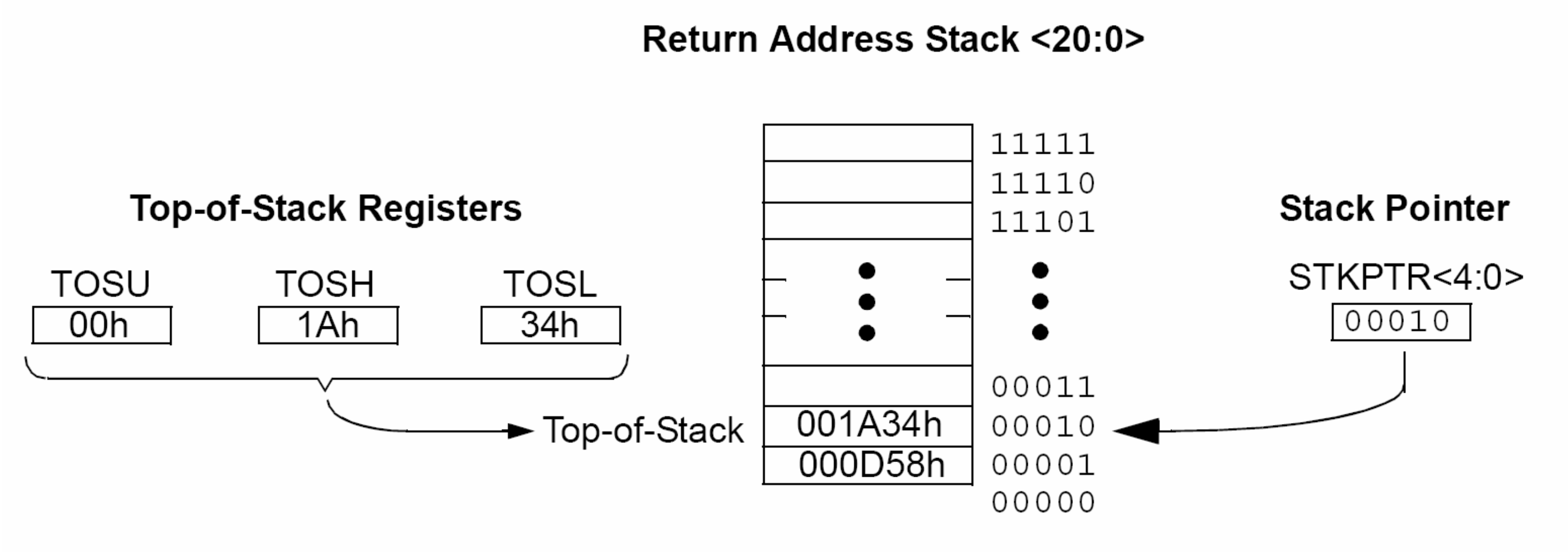
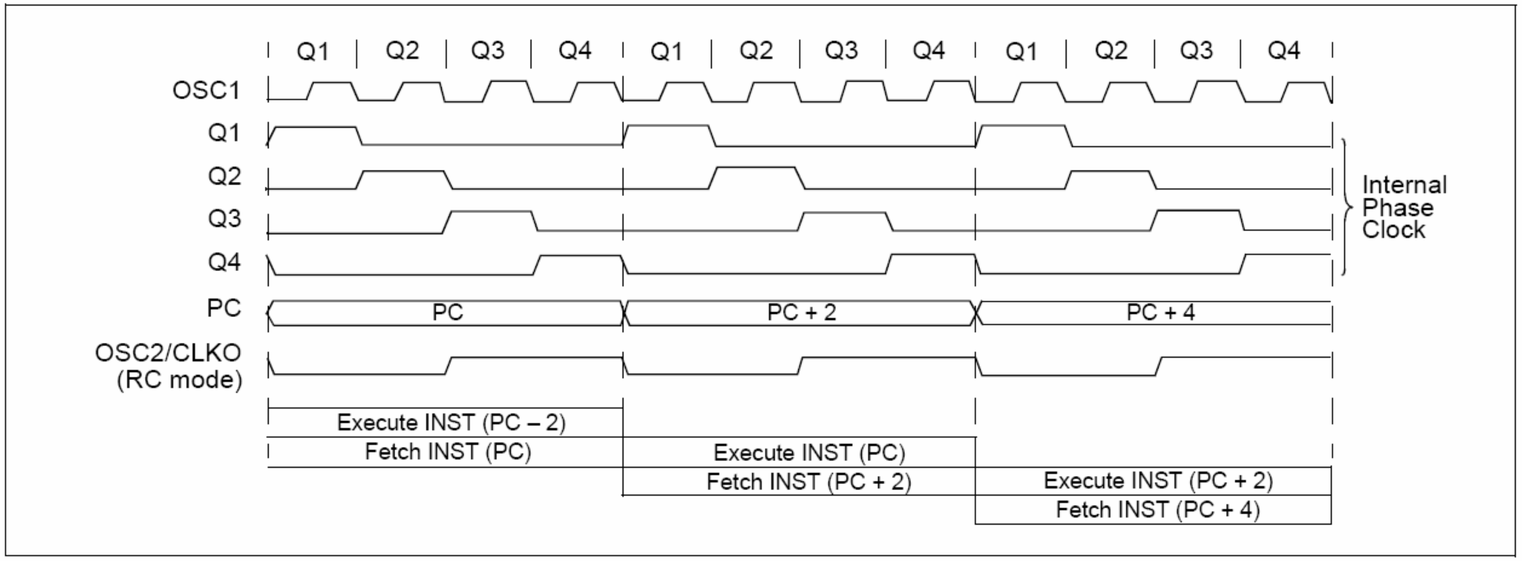
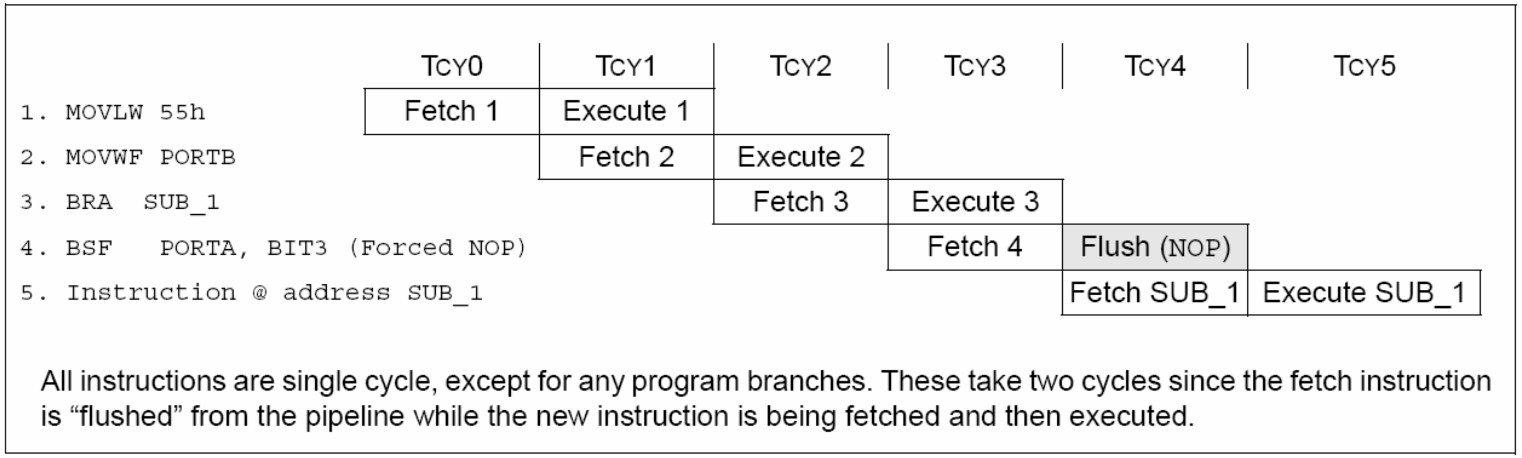


FIGURE 5-3: CLOCK/INSTRUCTION CYCLE

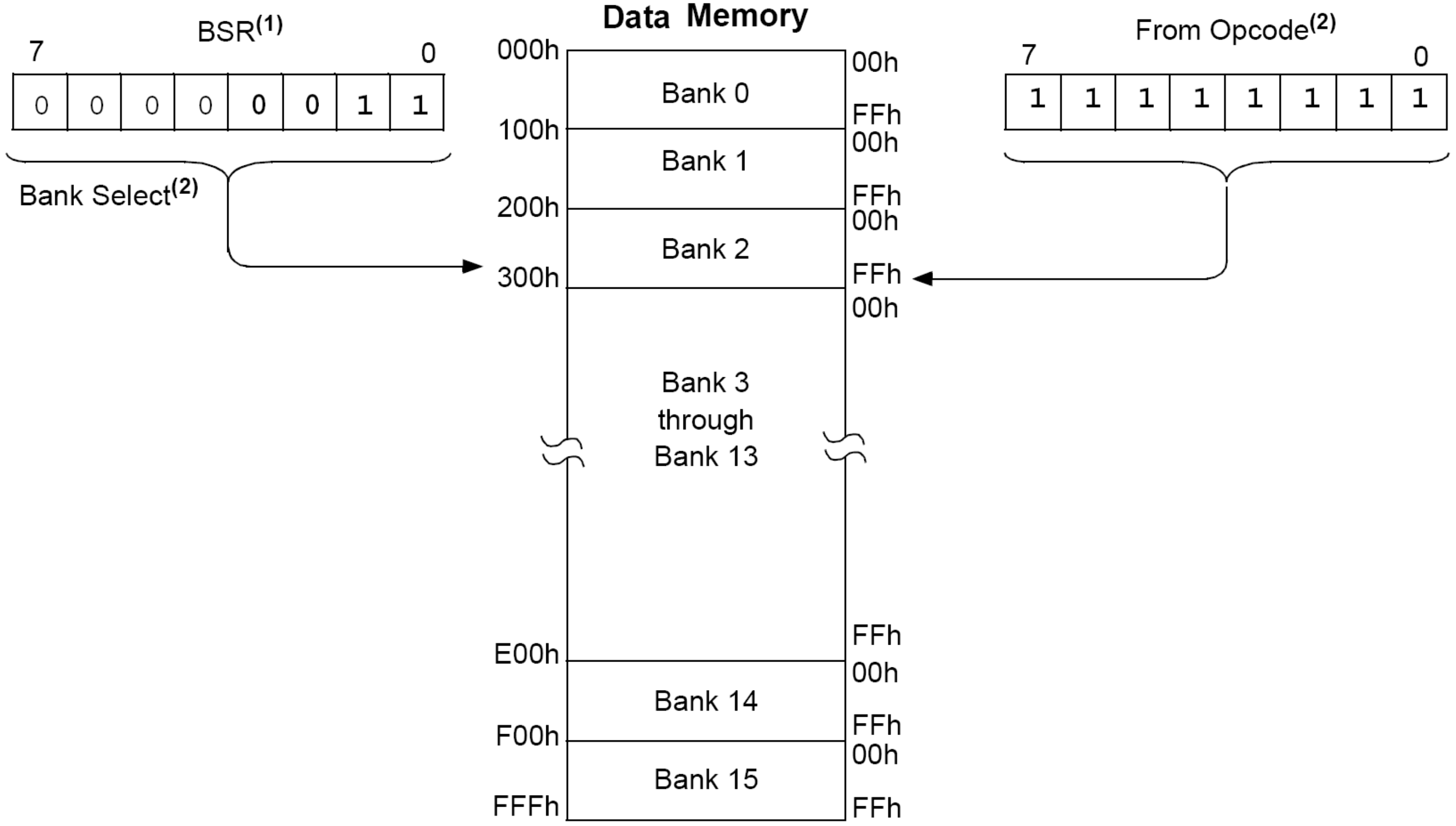


EXAMPLE 5-3: INSTRUCTION PIPELINE FLOW



	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOVLW 55h`	Fetch 1	Execute 1				
`2. MOVWF PORTB`		Fetch 2	Execute 2			
`3. BRA SUB_1`			Fetch 3	Execute 3		
`4. BSF PORTA, BIT3 (Forced NOP)`				Fetch 4	Flush (`NOP`)	
`5. Instruction @ address SUB_1`					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.



Note 1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

2: The `MOVFF` instruction embeds the entire 12-bit address in the instruction.

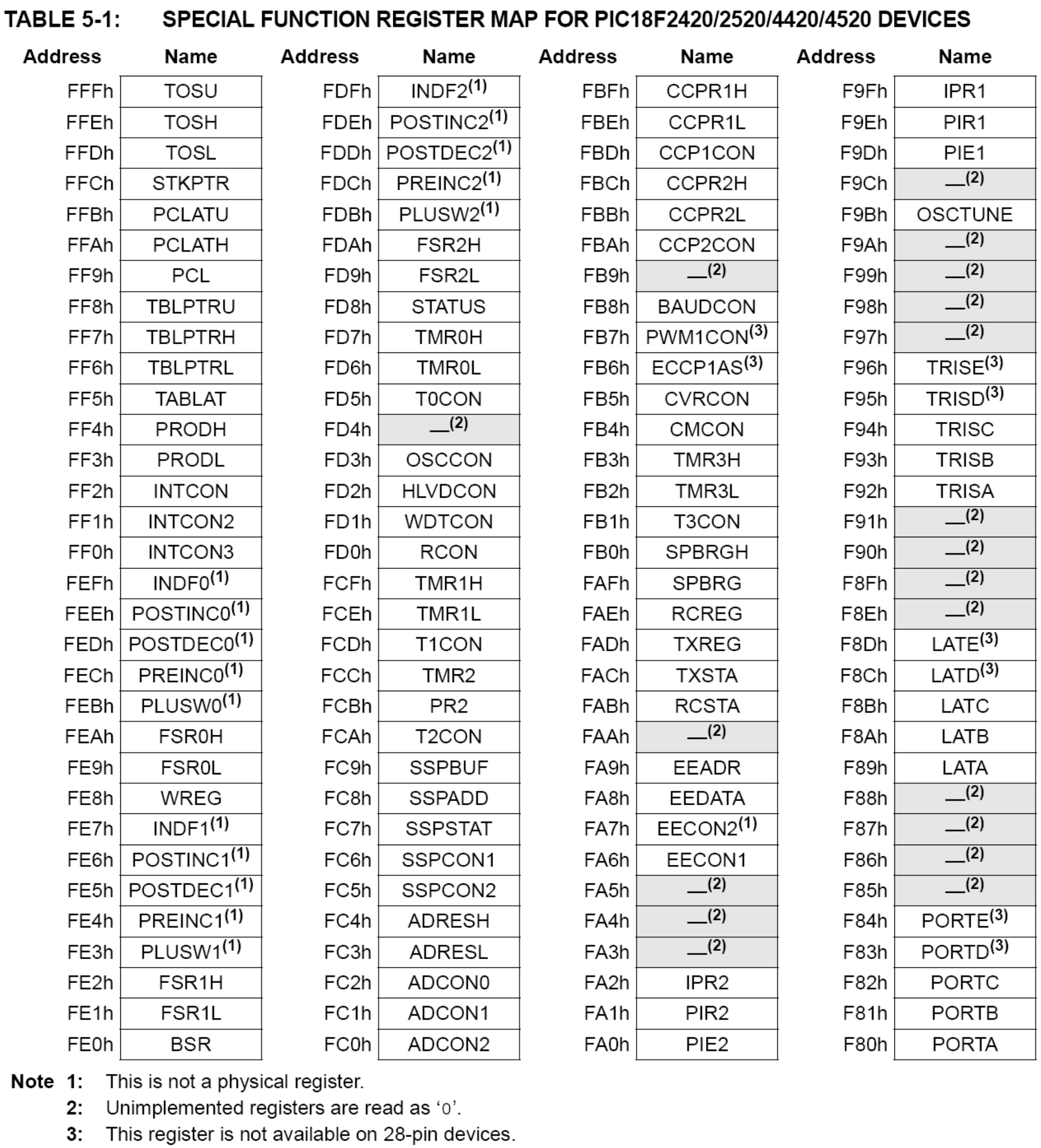
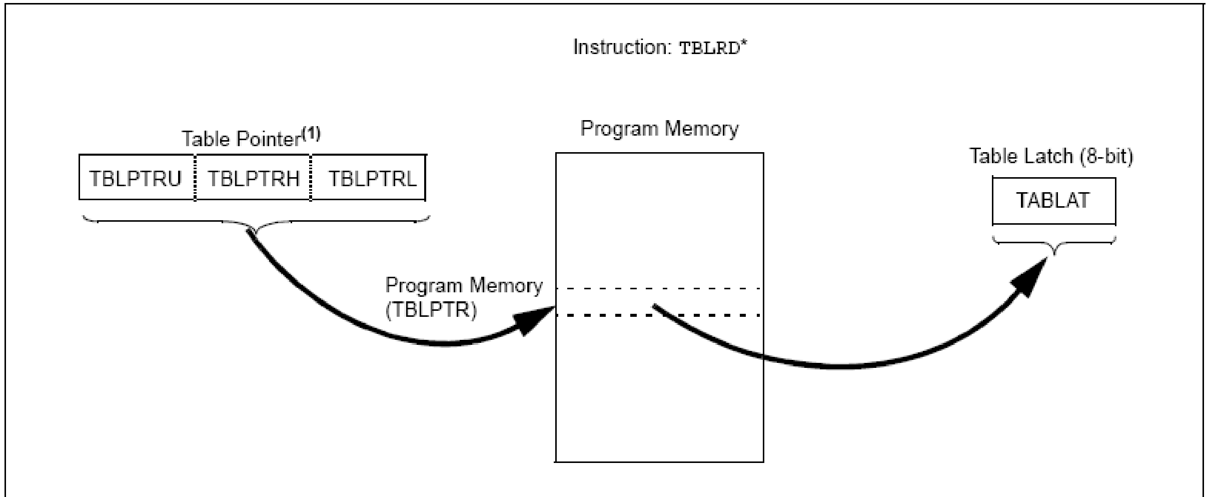


TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2420/2520/4420/4520 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2[1]	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2[1]	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2[1]	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2[1]	FBCh	CCPR2H	F9Ch	—[2]
FFBh	PCLATU	FDBh	PLUSW2[1]	FBBh	CCPR2L	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—[2]
FF9h	PCL	FD9h	FSR2L	FB9h	—[2]	F99h	—[2]
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	—[2]
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PWM1CON[3]	F97h	—[2]
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS[3]	F96h	TRISE[3]
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD[3]
FF4h	PRODH	FD4h	—[2]	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—[2]
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	—[2]
FEFh	INDF0[1]	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—[2]
FEEh	POSTINC0[1]	FCEh	TMR1L	FAEh	RCREG	F8Eh	—[2]
FEDh	POSTDEC0[1]	FCDh	T1CON	FADh	TXREG	F8Dh	LATE[3]
FECh	PREINC0[1]	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD[3]
FEBh	PLUSW0[1]	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—[2]	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—[2]
FE7h	INDF1[1]	FC7h	SSPSTAT	FA7h	EECON2[1]	F87h	—[2]
FE6h	POSTINC1[1]	FC6h	SSPCON1	FA6h	EECON1	F86h	—[2]
FE5h	POSTDEC1[1]	FC5h	SSPCON2	FA5h	—[2]	F85h	—[2]
FE4h	PREINC1[1]	FC4h	ADRESH	FA4h	—[2]	F84h	PORTE[3]
FE3h	PLUSW1[1]	FC3h	ADRESL	FA3h	—[2]	F83h	PORTD[3]
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

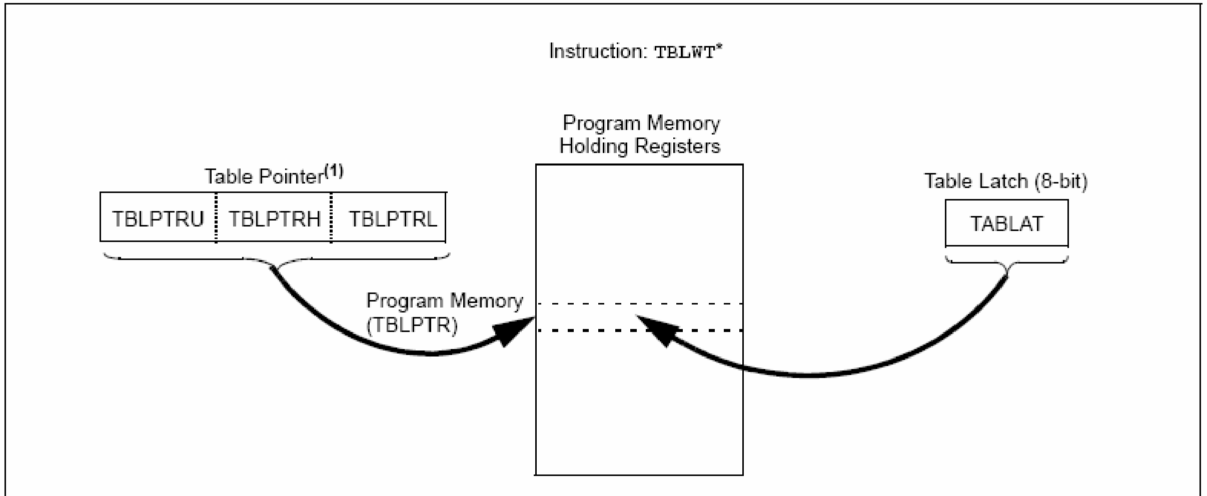
Note 1: This is not a physical register.
2: Unimplemented registers are read as ‘0’.
3: This register is not available on 28-pin devices.

FIGURE 6-1: TABLE READ OPERATION



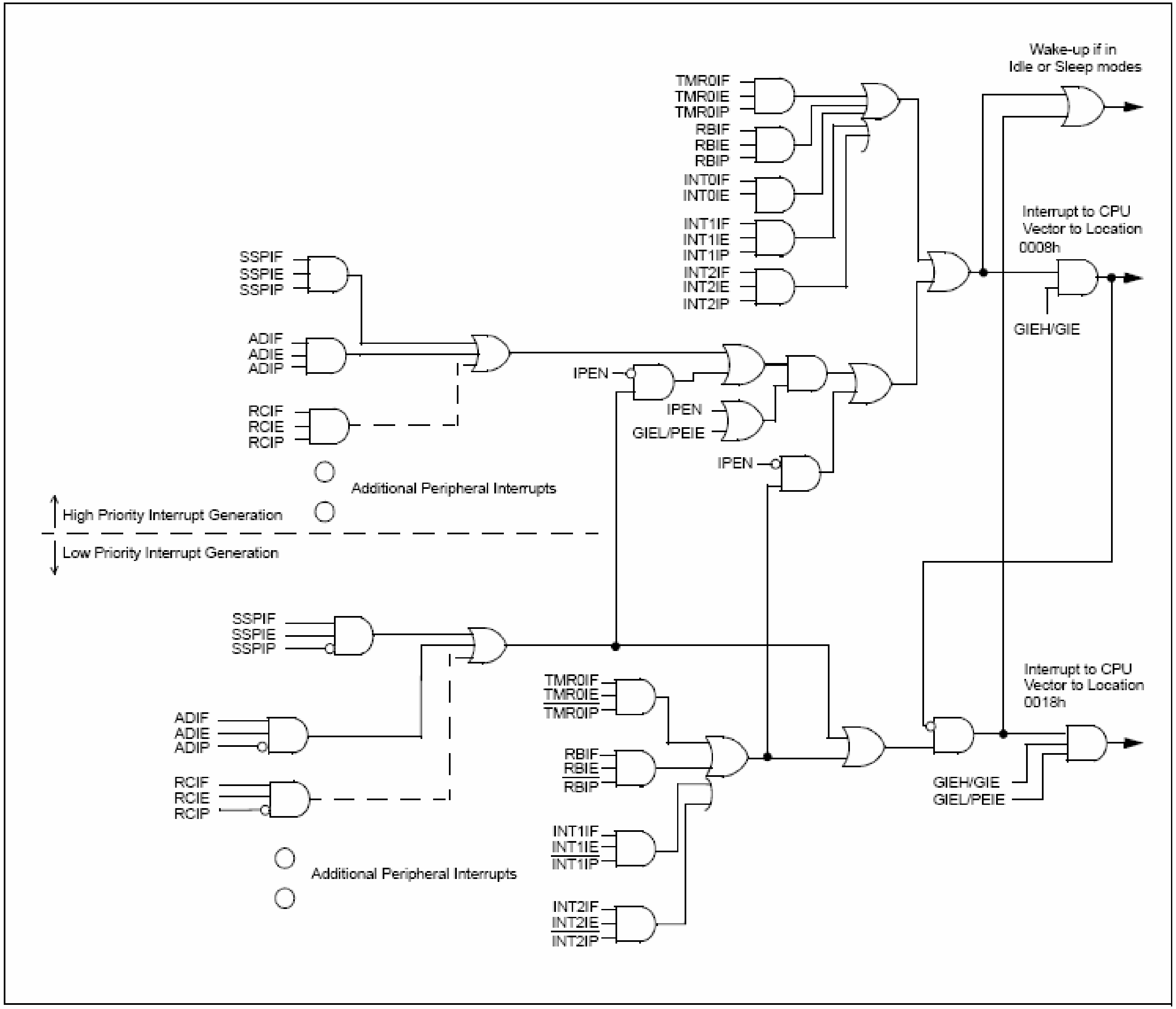
Note 1: Table Pointer register points to a byte in program memory.

FIGURE 6-2: TABLE WRITE OPERATION



Note 1: Table Pointer actually points to one of 64 holding registers, the address of which is determined by TBLPTRL<5:0>. The process for physically writing data to the program memory array is discussed in **Section 6.5 "Writing to Flash Program Memory"**.

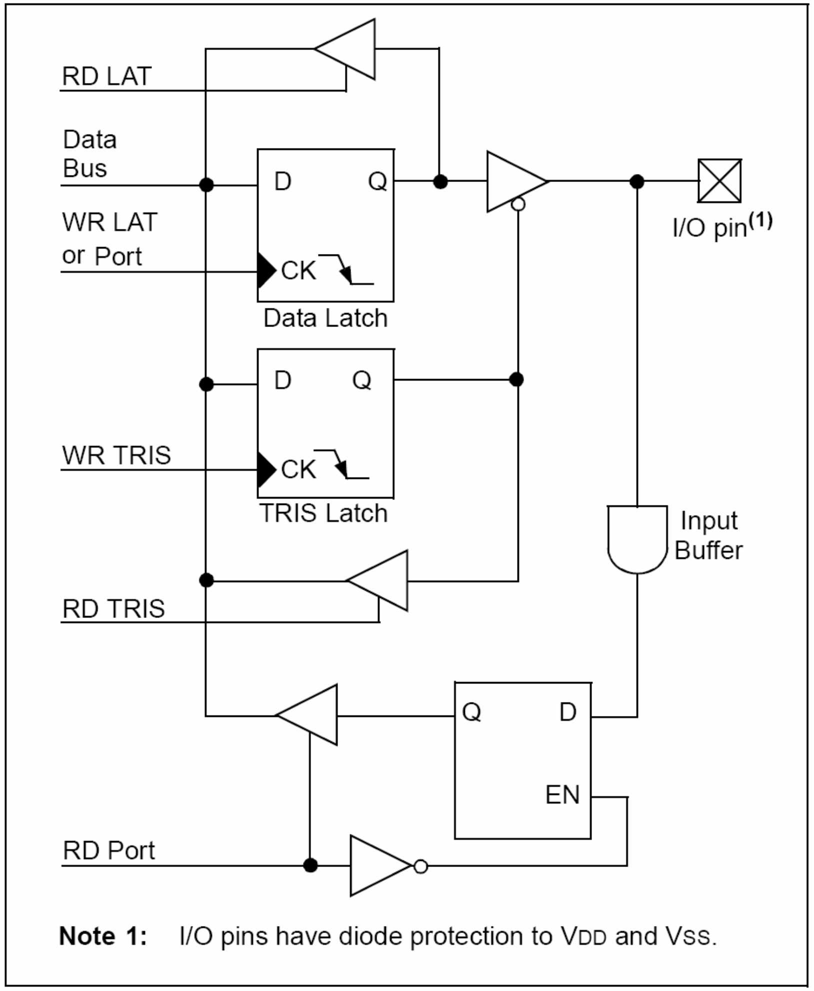
FIGURE 9-1: PIC18 INTERRUPT LOGIC



Some difference in I/O ports

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

In case of bidirectional pins, any read-modify-write instructions intended to change ***other*** pins of the port should use **LATi *not* PORTi.**



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

TIMERS

Timer0:

- 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

Timer1:

- 16 bit

Timer2, Timer3 ?

ENHANCED UNIVERSAL SYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

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

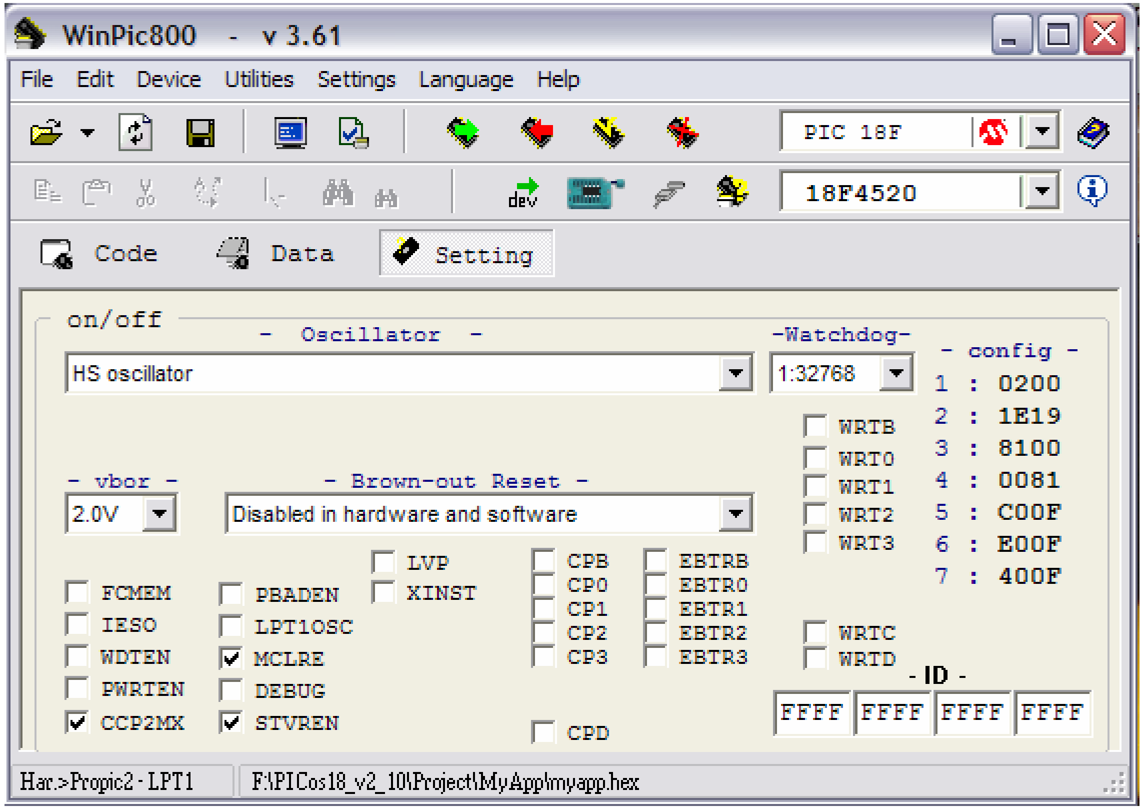
ADC MODULE

- 10-bit 13 channel
- ADCON0, ADCON1, ADCON2 ?
- An acquisition time can be programmed to occur between setting the GO/DONE bit and the actual start of the conversion.

Ex.

ADCON1 (PCFG3:PCFG0) ← 1111 ; makes all inputs digital i/o

Configuration bits



PIC 18F

18F4520

on/off

- Oscillator - HS oscillator

-Watchdog- 1:32768

- vbor - 2.0V

- Brown-out Reset - Disabled in hardware and software

□ WRTB
□ WRT0
□ WRT1
□ WRT2
□ WRT3

□ FCMEM
□ IESO
□ WDTEN
□ PWRTEN
☑ CCP2MX

□ PBADEN
□ LPT1OSC
☑ MCLRE
□ DEBUG
☑ STVREN

□ LVP
□ XINST

□ CPB
□ CP0
□ CP1
□ CP2
□ CP3

□ EBTRB
□ EBTR0
□ EBTR1
□ EBTR2
□ EBTR3

□ WRTC
□ WRTD

□ CPD

- config -
1 : 0200
2 : 1E19
3 : 8100
4 : 0081
5 : C00F
6 : E00F
7 : 400F

- ID -
FFFF FFFF FFFF FFFF

Har.>Propic2 - LPT1 F:\PICos18_v2_10\Project\MyApp\myapp.hex

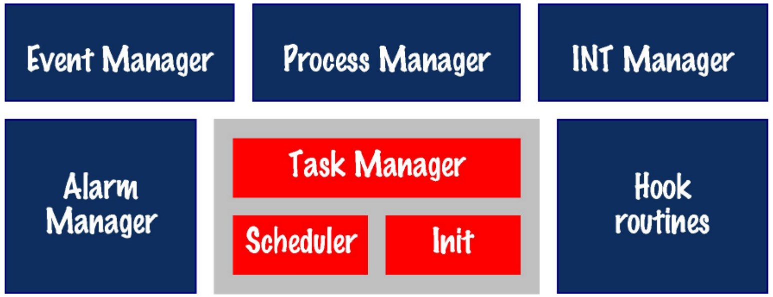
PICos18

A multi-task real-time kernel for PIC18 based on OSEK/VDX™ standart.

- **a kernel**
  - a set of functionalities, services.
- **multi-tasking**
  - multi-task and preemptive (PICos18)
  - multi-task and cooperative
- **real-time**
  - priorities
  - latency time (determinism)

  latency time for PICos: 50 us

PICos18 Kernel



- ✓ The **kernel core** (Init + Scheduler + Task Manager) in charge of managing the tasks present in the application and so decide the next task to activate function to the state and the priority oif the tasks.
- ✓ The **alarm and counter manager** (Alarm Manager). Attached to the kernel core it uses the TIMER0 interrupt in order to update periodically the alarms and counters used by the application.
- ✓ The **Hook routines** are partially included in the kernel core and let the developer to jump into additional land personal routines. Doing so it is possible to take the control of the kernel process during a short time to debug the application for instance.
- ✓ The **task manager** (Process Manager) is a set of kernel services which proposes the necessary functions to manage the task state (to change the state of a task, chain two tasks, activate a task...).
- ✓ The **event manager** (Event Manager) is a set of kernel services which proposes the necessary function to manage the events waited or posted by/to a task (to wait for an event, to post an event, to clear an event, to read the set of events received..).
- ✓ The **interrupt manager** (INT Manager) is a set of kernel services to enable or disable the interrupts (high and low interrupts).