Panel 1

Prior to Le10

Instruction Cycles and Special Function Registers

ME430 Mechatronics

1

Panel 2

Instruction Cycles and Special Function Registers

Special Function Registers

General Pin Input/Output (GPIO)
- ADCON1
- TRISx
- PORTx

Instruction Cycles
- Clock Frequency
- Instruction cycle frequency
- Delays

2

Panel 3

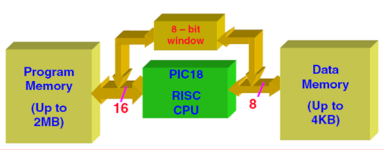
What is a Special Function Register (SFR)?

Table 1: SFRs vs. Variables

Common variable	SFR
char x;	(defined within the p18f4520.h file)
x = 0x5A;	PORTB = 0x5A;

Harvard Architecture

- 8-bit microcontroller
- 16-bit Instruction width
- Data Transfer Mechanism between PM and DM



3

Panel 4

Examples of SFRs that we'll use

ADCON1

TRISA	PORTA
TRISB	PORTB
TRISC	PORTC
TRISD	PORTD
TRISE	PORTE

OSCCON

(Plus a TON more, but they are behind the scenes used by the complier)

4

Panel 5

MASTERs 2007

SFR Access: Bits

- Bits may be referenced using the syntax:

  SFRbits.bitname

For example:

```
PORTBbits.RB0 = 1;
INTCONbits.GIEH = 0;
```

5

Panel 6

c:\MCC18\h\p18f4520.h

```
extern volatile near unsigned char       PORTB;
extern volatile near union {
  struct {
    unsigned RB0:1;
    unsigned RB1:1;
    unsigned RB2:1;
    unsigned RB3:1;
    unsigned RB4:1;
    unsigned RB5:1;
    unsigned RB6:1;
    unsigned RB7:1;
  };
  struct {
    unsigned INT0:1;
    unsigned INT1:1;
    unsigned INT2:1;
    unsigned CCP2:1;
    unsigned KBI0:1;
    unsigned KBI1:1;
    unsigned KBI2:1;
    unsigned KBI3:1;
  };
  struct {
    unsigned AN12:1;
    unsigned AN10:1;
    unsigned AN8:1;
    unsigned AN9:1;
    unsigned AN11:1;
    unsigned PGM:1;
    unsigned PGC:1;
    unsigned PGD:1;
  };
} PORTBbits;
```

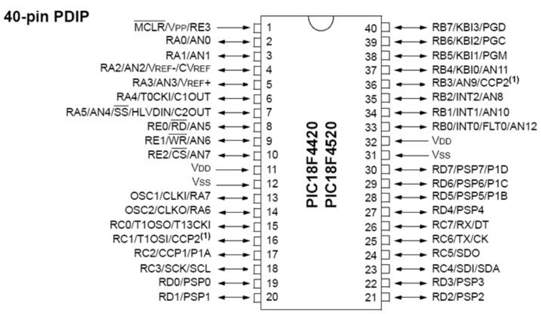
PORTB = 0 b ___ ___ ___ ___ ___ ___ ___ ___

RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0

6

Panel 7

Pin output diagram

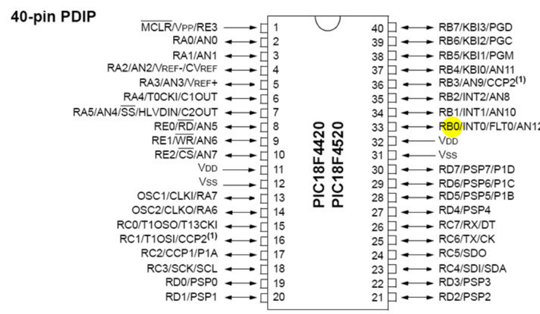


AN stands for Analog

7

Panel 8

Pin output diagram



What's different about RB5, RB6, & RB7?

AN stands for Analog

8

Panel 9

REGISTER 19-2: ADCON1 REGISTER

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 (VREF- source)
1 = VREF- (AN2)
0 = VSS

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = VDD

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

9

Panel 10

Setting your program to use digital outs

Step 1: Setup the pins as analog or digital as needed

```
ADCON1 = 0x0F;      // Everybody is digital
```

Step 2: Setup the pins as inputs or outputs as needed

```
TRISA = 0b00000011;        // Bottom 2 pins input
TRISB = 0x0F;              // Bottom 4 pins input
TRISC = 0xFF;              // All inputs
TRISD = 0x00;              // All outputs
TRISEbits.TRISE0 = 1;      // RE0 input
TRISEbits.TRISE1 = 0;      // RE1 output
TRISEbits.TRISE2 = 1;      // RE2 input
```

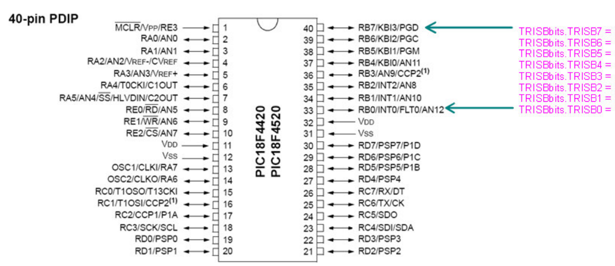
Step 3: Use the pins

```
Set output lines       high (1 => 5 volts) or low (0 => 0 volts)
   PORTBbits.RB1 = 1;      // Force RB1 high
Read input pins        high (5 volts => 1) or low (0 volts => 0)
   if(PORTC == 0x0A)       // Read PORTC
```

10

Panel 11

Example: There are 8 bits in TRISB and 8 pins in PORTB
Setting a bit in TRISB to a 1 makes that pin in PORTB an Input
Setting a bit in TRISB to a 0 makes that pin in PORTB an Output



Option 1: By Register (binary)

```
TRISB = 0b00000011;
```

Option 2: By Register (hex)

```
TRISB = 0x03;
```

Option 3: By bits

```
TRISBbits.TRISB7 = 0;
TRISBbits.TRISB6 = 0;
TRISBbits.TRISB5 = 0;
TRISBbits.TRISB4 = 0;
TRISBbits.TRISB3 = 0;
TRISBbits.TRISB2 = 0;
TRISBbits.TRISB1 = 1;
TRISBbits.TRISB0 = 1;
```

11

Panel 12

Example SFRs within a program

```
void main (void)
{
     ADCON1 = 0b00001111;        // Sets all the pins to digital
     TRISB = 0b00000000;         // Sets all the dital PORTB pins to outputs

    while (1)
    {
        PORTB = 0b0001;          // Light RB0
        Delay10KTCYx(200);       // Delay for 2 seconds
        PORTB = 0b1000;          // Light RB3
        Delay10KTCYx(200);       // Delay for 2 seconds
    }
}
```

Each of these SFR serves a specific purpose.

TRISA	= Settings make PORTA pins either inputs or outputs
TRISB	= Settings make PORTB pins either inputs or outputs
PORTA	= If A is an input you can read the values of the pins here If A is an output you can set the values of the pins here
PORTB	= If B is an input you can read the values of the pins here If B is an output you can set the values of the pins here

12

Panel 13

Practice: Write the statements needed to make all pins digital (not analog). Next, make the bottom 4 pins of PORTA (RA3-RA0) outputs (top 4 pins inputs). Finally set the bottom two pins high (RA0,RA1 high RA2,RA3 low)

13

Panel 14

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

14

Panel 15

Instruction Cycle

- Clock Frequency
  - EC - External Canned Oscillator
  - INTIO67 - Internal Oscillator
- Setting the OSCCON Special Function Register
- Instruction cycle frequency
  - Assembly
- Delays Library

15

Panel 16

Clock sources

External Canned Oscillator 4 MHz Only on PICDEM board

Internal oscillator
- Officially less exact (still penty exact for your needs)
- Only option when we move off the PICDEM later
- Range of values from 8 MHz to 31.25 kHz

16

Panel 17

Setting the configuration bit to choose the oscillator

```
#pragma config OSC = EC  // External 4MHz crystal for PICDEM board only

#pragma config OSC = INTIO67  // Internal oscillator
```

Pick the one you want, never use both at the same time

17

Panel 18

Setting the internal oscillator

REGISTER 2-2: OSCCON REGISTER

R/W-0	R/W-1	R/W-0	R/W-0	R[(1)]	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0
bit 7							bit 0

bit 7 **IDLEN:** Idle Enable bit
1 = Device enters Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF2:IRCF0:** Internal Oscillator Frequency Select bits
111 = 8 MHz (INTOSC drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz[(3)]
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (from either INTOSC/256 or INTRC directly)[(2)]

Example code

```
void main (void)
{
        OSCCONbits.IRCF2 = 1;
        OSCCONbits.IRCF1 = 1;
        OSCCONbits.IRCF0 = 0;

        TRISB = 0;

        while (1)
        {
                PORTB = 0b00000001;
                Delay10KTCYx(delayTime);

                PORTB = 0b00001000;
                Delay10KTCYx(delayTime);
        }
}
```

18

Panel 19

Modify the OSCCON bits to set the internal oscillator to 2 MHz

OSCCONbits.IRCF2 = __ ;

OSCCONbits.IRCF1 = __ ;

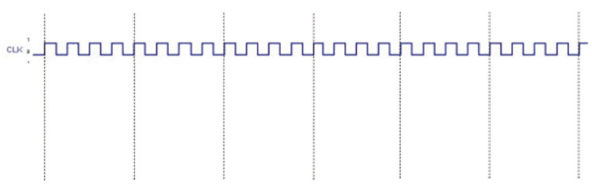
OSCCONbits.IRCF0 = __ ;

19

Panel 20

Instruction cycle

The Instruction cycle frequency is always ________ the clock frequency




```
  ADCON1 = 0b00001111;        // Sets all the pins to digital
MOVLW 0xf
MOVWF 0xfc1, ACCESS
  TRISB = 0b00000000;         // Sets all the dital PORTB pins to outputs
CLRF 0xf93, ACCESS
```

20

Panel 21

Delay Functions:

#include <delays.h>

TABLE 4-4: DELAY FUNCTIONS

Function	Description
Delay1TCY	Delay one instruction cycle.
Delay10TCYx	Delay in multiples of 10 instruction cycles.
Delay100TCYx	Delay in multiples of 100 instruction cycles.
Delay1KTCYx	Delay in multiples of 1,000 instruction cycles.
Delay10KTCYx	Delay in multiples of 10,000 instruction cycles.

```
/** Header Files ****************************************/
#include <p18f4520.h>
#include <delays.h>

#pragma config OSC = EC  // External 4MHz crystal for PICDEM board only


        PORTB = 0b0001;         // Light RB0
        Delay10KTCYx(200);      // Delay for 2 seconds
        PORTB = 0b1000;         // Light RB3
        Delay10KTCYx(200);      // Delay for 2 seconds
```

21

Panel 22

In the code it says this command delays for 2 seconds. Show the math:

Delay10KTCYx(200); // Delay for 2 seconds

Clock Frequency =

Instruction Cycle Frequency =

Period of the Instruction Cycle =
(aka the time needed for 1 instruction)

Number of Instruction generated by this command =

Resulting delay time = ________ * _________ = 2 seconds

22