



# **MICROPROCESSADORES II**

---

Professor: Patric Janner Marques

Aula : Utilização da PLIB, LCD e teclado

## Legacy Peripheral Libraries

- Nas versões mais recentes do compilador XC8, a parte que se refere as bibliotecas de interfaceamento com os periféricos dos microcontroladores PIC não vem junto com o compilador;
- Sendo assim, as bibliotecas para acessar os periféricos (PLIB), como ADC, LCD, SPI, I<sup>2</sup>C e outros, devem ser instaladas separadamente, iniciando pela download do “PIC18F Legacy Peripheral Libraries v2.0 - Windows” na aba Downloads do link: <<http://www.microchip.com/mplab/compilers>>

# Legacy Peripheral Libraries

## Peripheral Libraries (PLIBS)



### Current Peripheral Libraries

8-bit MCUs: MPLAB Code Configurator

16-bit PIC24F MCUs: MPLAB Code Configurator

16-bit dsPIC33, PIC24E, PIC24H MCUs: Legacy Peripheral Libraries

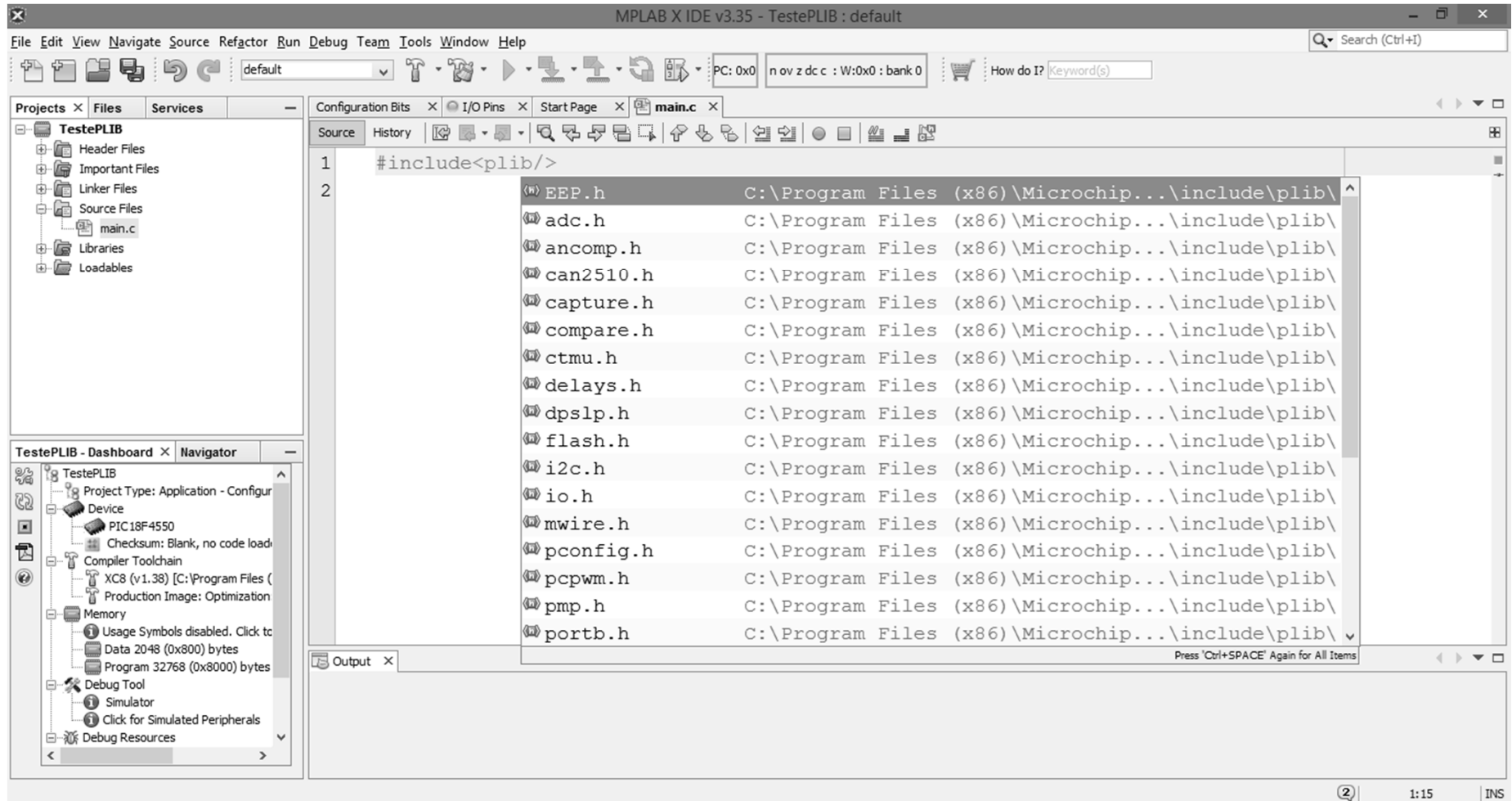
32-Bit MCUs: MPLAB Harmony

Legacy Peripheral Libraries	Date Published	Size	D/L
<b>PIC18F</b>			
PIC18F Legacy Peripheral Libraries v2.0 - Linux	7/28/15	87.5 MB	
PIC18F Legacy Peripheral Libraries v2.0 - MAC OS	7/28/15	93.0 MB	
PIC18F Legacy Peripheral Libraries v2.0 - Windows	7/28/15	85.5 MB	

# Legacy Peripheral Libraries

- Para ativar o uso da PLIB, dois passos são importantes:
  - Deve-se incluir a biblioteca desejada no projeto:
    - Exemplo:  
`#include <plib/xlcd.h>`
  - Para verificar as bibliotecas disponíveis, entre no MPLAB e digite `#include <plib/`
  - O sistema de autocompletar do MPLAB mostrará as opções.

# Legacy Peripheral Libraries



PARA TUDO.....

QUE QUERO DESCER.....

The PIC18 peripheral libraries have been not been supported or shipped with the compiler for several versions. Now, the option to indicate the use of these libraries has also been removed.



**O QUE FAZER AGORA!?!?!?**

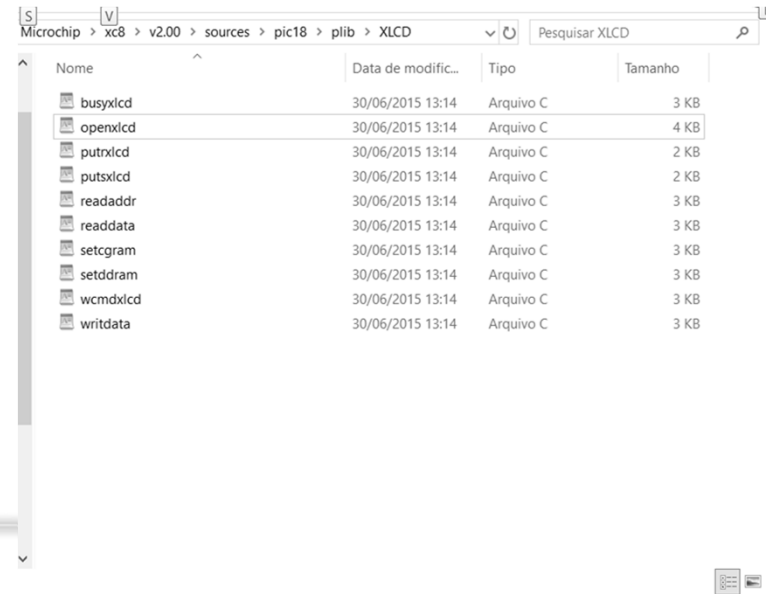
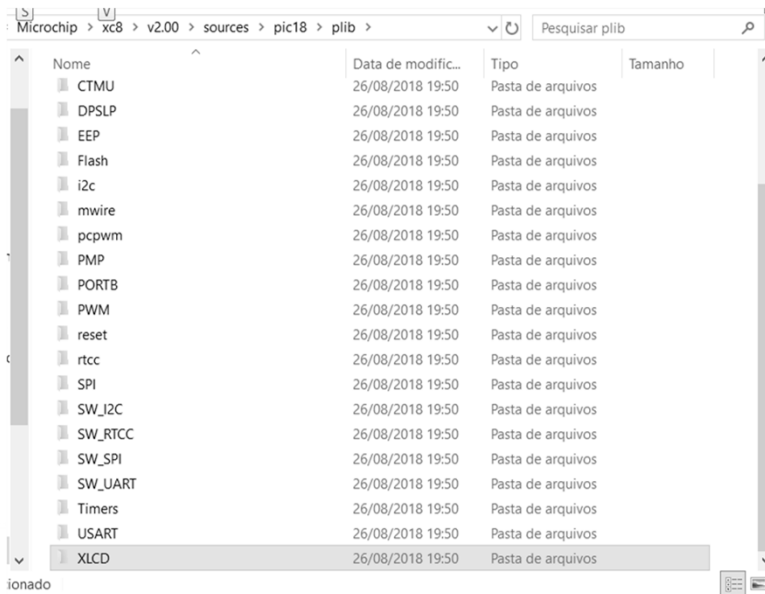


## Indo a origem..... Manobra RAIZ!!!

- Teremos que ir até a fonte da PLIB e pegar os códigos-fontes;
- Normalmente, eles estão nos diretórios:

C:\Program Files (x86)\Microchip\xc8\v2.00\sources\pic18\plib\XLCD

C:\Program Files (x86)\Microchip\xc8\v2.00\include\plib



## Indo a origem..... Manobra RAIZ!!!

- Depois disso, teremos que criar a nossa própria biblioteca a partir das funções implementadas nestes arquivos;
- Copie tudo do arquivo .h (xc8\v2.00\include\plib) e cole no teu arquivo .h;

- Exceto:

```
#ifndef __XLCD_H
#define __XLCD_H
#endif
```

- Dos arquivos .c (xc8\v2.00\sources\pic18\plib\XLCD), copie tudo e cole no teu arquivo .c. Exceto os *#include*, deixando apenas um *#include "p18cxxx.h"*



Indo a origem..... Manobra RAIZ!!!

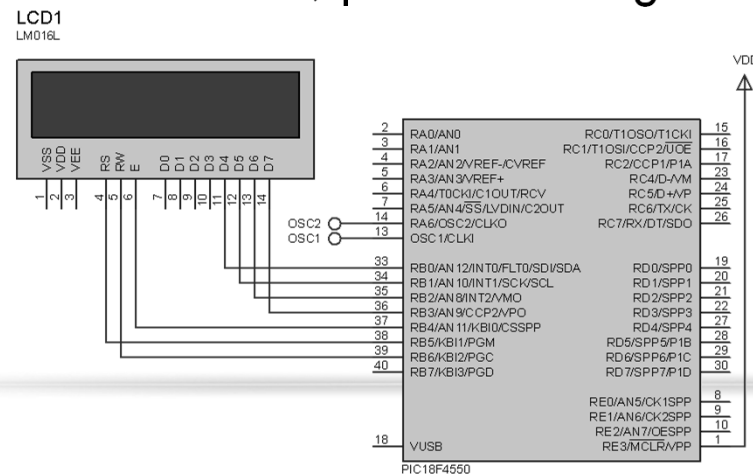
- Ainda no `xlcd.c`:
  - Busque por *Delay10KTCYx(0x05);*
    - Troque por *\_delay(500);*
  - Busque por *Delay10KTCYx(0x01);*
    - Troque por *\_delay(100);*
- Por fim, inclua no teu `main.c` e `xlcd.c`:

`#include "xlcd.h"`

**Problemas com  
a biblioteca  
*delays.h***

# Biblioteca xlcd.h

- Vamos conhecer um pouco da biblioteca xlcd.h para utilização do display LCD;
- Alguns informações extras podem ser vistas em:  
<<http://www.studentcompanion.co.za/interfacing-lcd-display-with-pic-microcontroller-xc8/>>
- Seja prudente na leitura, pois notei algumas incoerências.



## Biblioteca xlcd.h

- Uma observação inicial:
  - A biblioteca foi criada com base em um popular LCD industrial, que utiliza o controlador HITACHI HD44780U;
  - No Proteus, é possível encontrar o controlador propriamente dito, ou utilizar o conjunto completo (Display LCD + controlador) buscando por LM016L, um produto comercial da HITACHI.

## Biblioteca xlcd.h – Funções

- BusyXLCD (): Verifica se o controlador está ocupado, retornando 1 quando ocupado;
- OpenXLCD(): Configura o modo de operação do LCD, pode ser:
  - FOUR\_BIT /\* comunicação por 4-bit \*/
  - EIGHT\_BIT /\* comunicação por 8-bit \*/
  - LINE\_5X7 /\* 5x7 pontos por caractere, linha simples\*/
  - LINE\_5X10 /\* 5x10 pontos por caractere \*/
  - LINES\_5X7 /\* 5x7 pontos por caractere, múltiplas linhas\*/
- Exemplo: OpenXLCD(FOUR\_BIT&LINES\_5X7)

## Biblioteca xlcd.h – Funções

- putcXLCD (): Escreve um byte no LCD, a partir da tabela ASCII.
- putsXLCD (): Escreve uma string **constante** no LCD, parando quando detectar o caractere NULL ('\0').
  - Exemplo: putsXLCD("Hello World");
- putrsXLCD (): Escreve uma string **guardada em uma variável** no LCD, parando quando detectar o caractere NULL ('\0').
  - Exemplo:           char teste[20] = "Hello mundo";  
                          putrsXLCD(Teste);

Obs.: **Nos testes que fiz, não teve diferença!**

## Biblioteca xlcd.h – Funções

- Estes comandos devem ser executados quando o controlador do LCD não está ocupado. Logo, sempre execute antes o comando `while(BusyXLCD())` para depois execute-os;
- `WriteCmdXLCD ()`: Escreve um comando para o controlador LCD, pode ser:
  - DON : Ligar display
  - DOFF : Desligar display
  - CURSOR\_ON : Display ligado e aparecendo o cursor
  - CURSOR\_OFF : Display ligado, mas sem aparecer o cursor
  - BLINK\_ON : Display ligado e o cursor piscando
  - BLINK\_OFF : Display ligado, mas cursor não piscando
  - SHIFT\_CUR\_LEFT : Cursor se desloca para a esquerda
  - SHIFT\_CUR\_RIGHT : Cursor se desloca para a direita
  - SHIFT\_DISP\_LEFT : Display se desloca para a esquerda
  - SHIFT\_DISP\_RIGHT : Display se desloca para a direita

## Biblioteca xlcd.h – Funções

- Mais comandos na página 24 do Manual do controlador HITACHI em: <<http://www.studentcompanion.co.za/wp-content/uploads/2013/05/HD44780U.pdf>>

Instruction	Code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
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.
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.
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).

## Biblioteca xlcd.h – Funções

- SetDDRamAddr (): Defini o endereço dentro DDRAM do controlador, posicionando o início da escrita no display;
- Exemplo (olhar pg. 12 do HITACHI):
  - SetDDRamAddr(0x40);

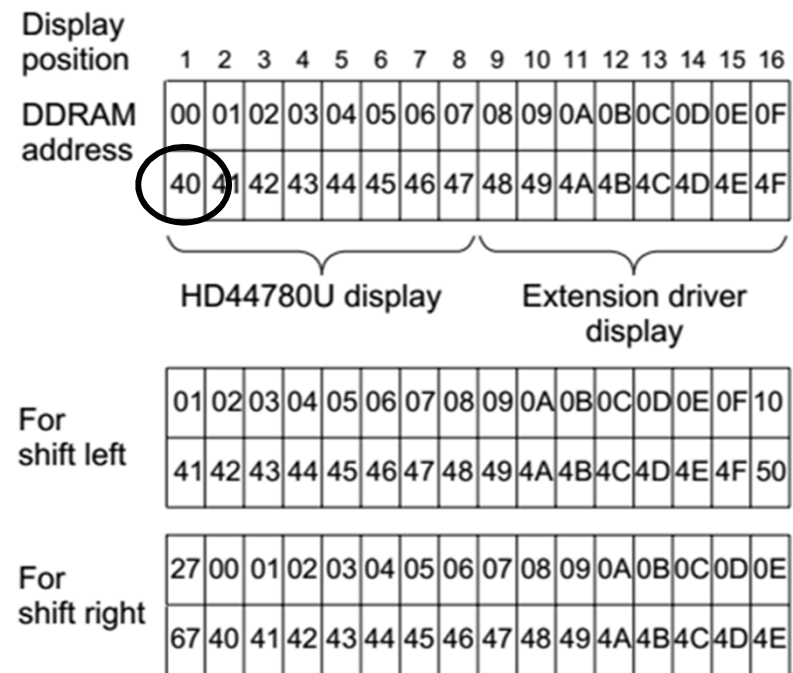


Figure 6 2-Line by 16-Character Display Example



# IMPORTANTE

- A biblioteca `xlcd.h` exige que sejam criadas 3 funções no seu projeto, com propósitos temporais, baseadas no *clock* do sistema:
- `void DelayFor18TCY(void);` // Delay para 18 ciclos de máquina
- `void DelayPORXLCD(void);` // Delay para 15 ms
- `void DelayXLCD(void);` // Delay para 5 ms

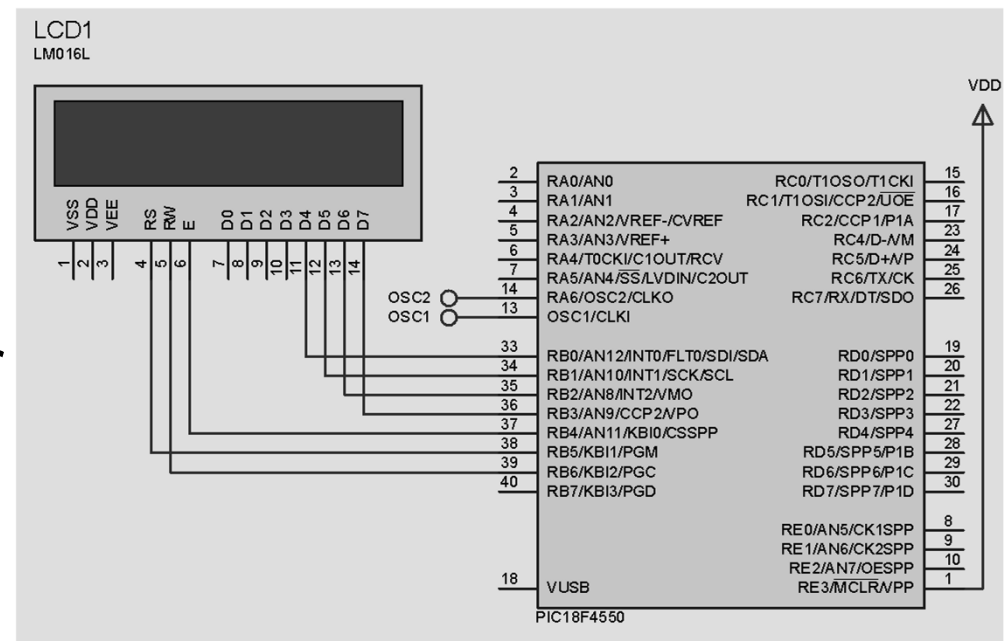
# IMPORTANTE

- Exemplo para um *clock* de 8MHz:

```
void DelayFor18TCY(void)
{
    __delay_us(36);
}
void DelayPORXLCD(void)
{
    __delay_ms(15);
}
void DelayXLCD(void)
{
    __delay_ms(5);
}
```

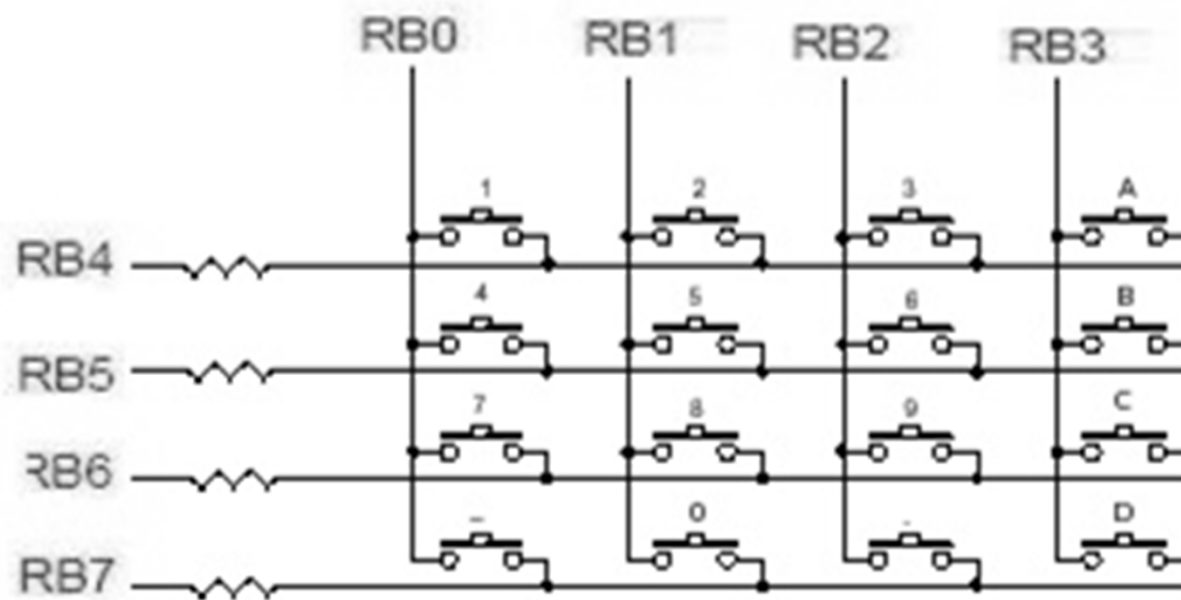
# Melhorando a biblioteca

- Criando funções para:
  - Inicializar o LCD;
  - Limpar a tela do LCD;
  - Escrever qualquer mensagem em qualquer posição do LCD.
- Testar usando o Proteus.



# Teclado Matricial

- Princípio de funcionamento:



**Pensando nisso, como seria o programa para controlar um teclado???**

# Teclado Matricial

- Exemplo de algoritmo para teclado:

```
char teclado() {  
    while (1) {  
        LATA = 0b00000001;  
        Mydelay(50);  
        if (RA3) {  
            while (RA3);  
            return 1;  
        } else if (RA4) {  
            while (RA4);  
            return 4;  
        } else if (RA5) {  
            while (RA5);  
            return 7;  
        }  
    }  
}
```

```
LATA = 0b00000010;  
Mydelay(50);  
if (RA3) {  
    while (RA3);  
    return 2;  
} else if (RA4) {  
    while (RA4);  
    return 5;  
} else if (RA5) {  
    while (RA5);  
    return 8;  
}
```

```
LATA = 0b00000100;  
Mydelay(50);  
if (RA3) {  
    while (RA3);  
    return 3;  
} else if (RA4) {  
    while (RA4);  
    return 6;  
} else if (RA5) {  
    while (RA5);  
    return 9;  
}
```