Trabajo final de Electrónica Aplicada III 5° año

"Control de Acceso por RF"

Curso: 5R2

Grupo: 3

Alumnos:

LOPEZ RICCI, Mariano Legajo: 46.717
RYAN, Emilio Tomás Legajo: 46.641
MUÑOZ TREJO, Daniel Angel Legajo: 50.022

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN.

Una vez un integrante de este grupo fue a su casa (en Paraguay) con un vehículo comprado aquí en Argentina. Y cuando, allá, le puso la alarma al auto (con el control remoto) se abrió el portón del vecino, que también era a control remoto. El guardia salió preocupado a ver que pasaba. Resultó que al poner la alarma al vehículo se abría el portón, y al sacarle la alarma se cerraba.

Ese incidente impulsó a los integrantes de este grupo a buscar un mando a distancia más seguro. Un mando cuya señal no pueda ser sintonizada por un receptor furtivo y luego imitada para entrar a robar (como en la película *60 segundos*).

Este desarrollo pretende crear una técnica segura de mando a distancia para un portón y aplicarla en un prototipo.

CAPÍTULO 1: PROTOCOLO DESARROLLADO.

Desarrollamos un protocolo simple pero seguro para el mando a distancia.

EL PROTOCOLO PARA ABRIR EL PORTÓN.

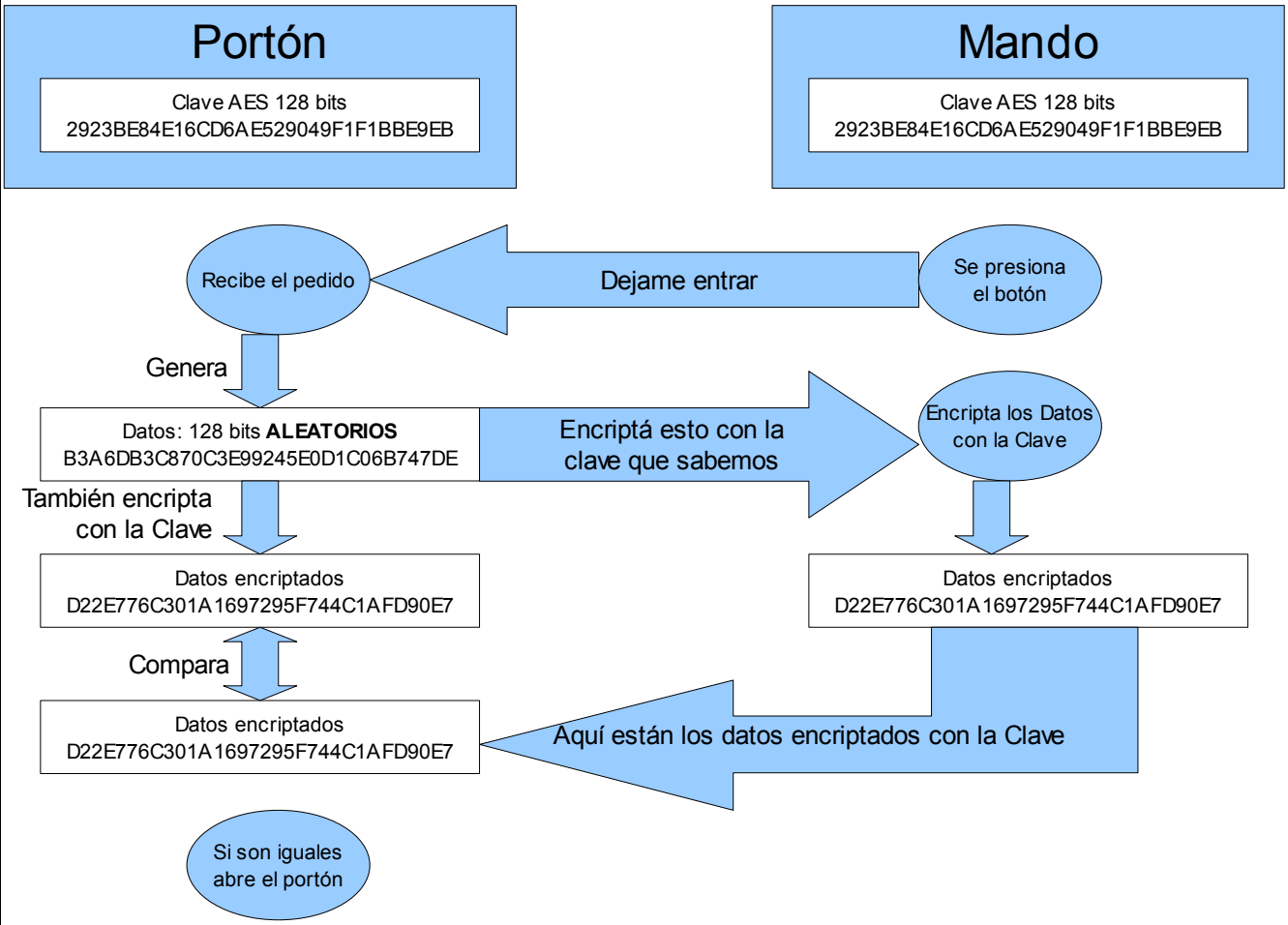


Ilustración 1.1: Protocolo para abrir/cerrar

La Ilustración 1.1 muestra el protocolo. Cuando el mando a distancia quiere abrir el portón (porque se presionó el botón del control remoto), envía un código (0xFF) al portón. El portón recibe el pedido y genera un bloque de 128 bits que envía (precedido por 0x55) al mando y luego los encripta para comparar posteriormente. El mando encripta esos datos con una clave de 128 bits que ambos conocen y se los devuelve encriptados (precedidos por 0x55). El portón compara los datos recibidos con los datos que él mismo encriptó. Si coinciden, abre el portón.

EL ALGORITMO DE ENCRIPTACIÓN AES.

Advanced Encryption Standard (AES), también conocido como Rijndael, es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. Se espera que sea usado en el mundo entero y analizado exhaustivamente, como fue el caso de su predecesor, el Data Encryption Standard (DES). El AES fue anunciado por el Instituto Nacional de Estándares y Tecnología (NIST) como FIPS PUB 197 de los Estados Unidos (FIPS 197) el 26 de noviembre de 2001 después de un proceso de estandarización que duró 5 años (véase proceso de Advanced Encryption

Standard para más detalles). Se transformó en un estándar efectivo el 26 de mayo de 2002. Desde 2006, el AES es uno de los algoritmos más populares usados en criptografía simétrica.

El cifrador fue desarrollado por dos criptólogos belgas, Joan Daemen y Vincent Rijmen, ambos estudiantes de la Katholieke Universiteit Leuven, y enviado al proceso de selección AES bajo el nombre "Rijndael".

Hasta 2005, no se ha encontrado ningún ataque exitoso contra el AES. La Agencia de Seguridad Nacional de los Estados Unidos (NSA) revisó todos los finalistas candidatos al AES, incluyendo el Rijndael, y declaró que todos ellos eran suficientemente seguros para su empleo en información no clasificada del gobierno de los Estados Unidos. En junio de 2003, el gobierno de los Estados Unidos anunció que el AES podía ser usado para información clasificada.

El algoritmo AES tiene la característica de que aunque uno conozca los datos sin encriptar y los datos encriptados, no puede descubrir la clave. Por lo tanto es idóneo para esta aplicación.

LA TRANSMISIÓN DE LA CLAVE.

Si uno sospecha de que alguien se pudo apropiar de la clave, o simplemente por seguridad, puede cambiar la clave. Para cambiar la clave se mantiene presionado el botón del controlador del portón por 500 ms o más. Luego, para difundir la clave a los demás mandos a distancia, se presiona el mismo botón, pero sin dejarlo presionado. Esto hace un “broadcast” de la clave y los mandos a distancia se sincronizan. La clave se envía precedida por 0x00.

Si no se desea transmitir la clave inalámbricamente se puede desactivar la transmisión cambiando un jumper en el controlador del portón y se conecta físicamente cada mando a distancia que se quiera sincronizar. Al presionar el botón, estando conectado mediante un cable, se sincroniza (recibe la nueva clave).

CAPÍTULO 2: EL HARDWARE.

Se realizaron dos prototipos. Uno del controlador del portón y otro del mando a distancia. Ambos en plaquetas multipropósito como muestra la Ilustración 2.1.

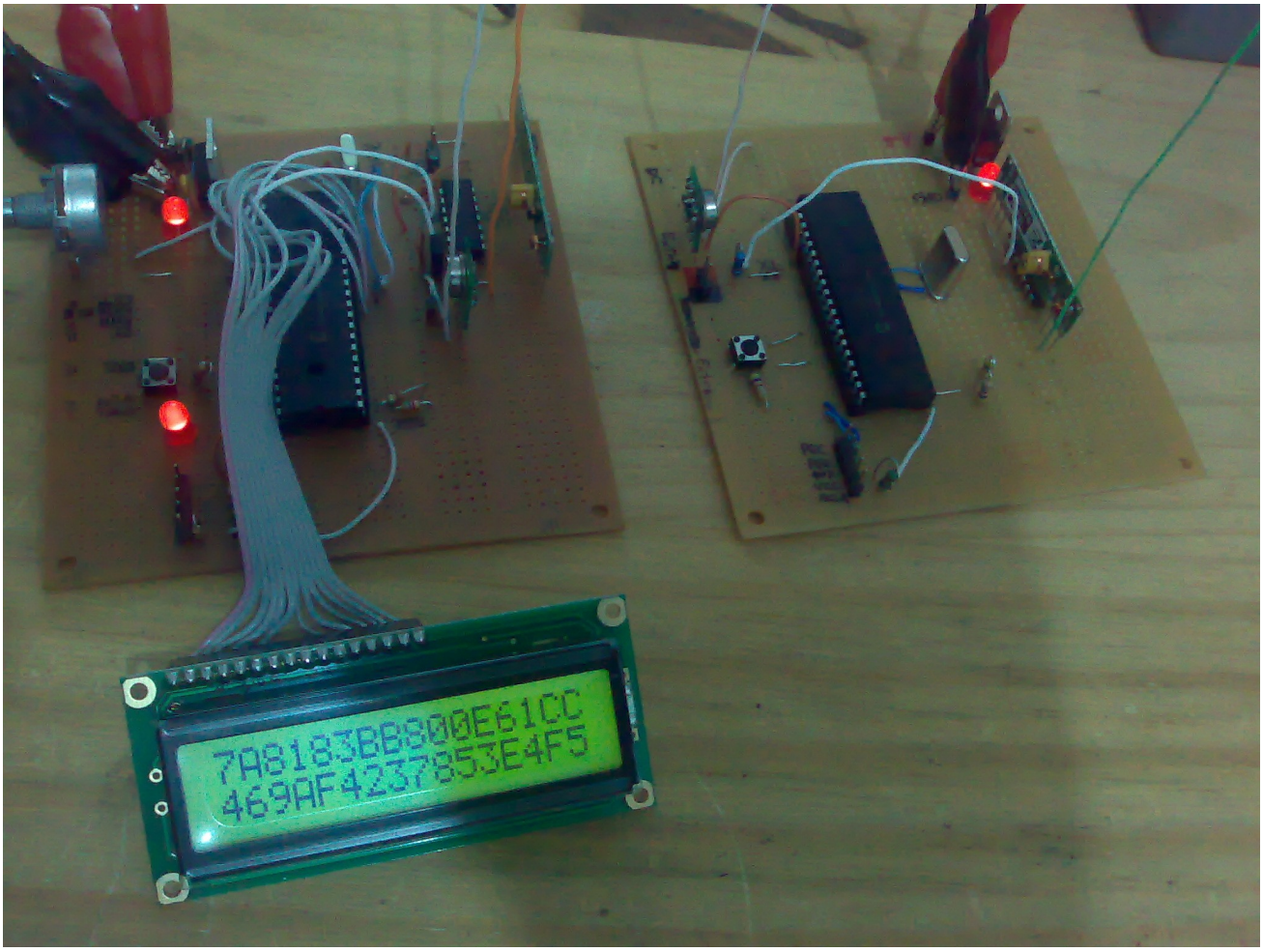


Ilustración 2.1: Portón y mando

EL PROTOTIPO DEL CONTROLADOR DEL PORTÓN.

La Ilustración 2.2 muestra el hardware utilizado para el prototipo del controlador del portón.

Hay un divisor resistivo (10K y 10K) entre VCC y masa para generar ruido a la entrada del conversor AD (AN0). Este conversor se utiliza para generar datos aleatorios. Se realizan tantas conversiones como bits aleatorios se necesiten. Se toma el bit menos significativo de cada conversión y se lo agrega al buffer de bits aleatorios. El código a continuación muestra como se generan los datos.

```
// Genera 16 bytes aleatorios y los escribe a partir de addr
void generarAleatorio(unsigned char *addr){
      unsigned char i,j,byte;
      ClearWDT();
      for(i=0;i<16;i++){
            byte=0;
            for(j=0;j<8;j++){
                  ConvertADC();
                  while(BusyADC());
                  byte |= ((ADRESL & 1)<<j);
```

```
            }
            *addr=byte;
            addr++;
        }
}
```

Esta función se llama cuando se quiere generar una nueva clave y cada vez que se generan los datos aleatorios que se deben enviar al mando a distancia para ser encriptados y devueltos.

El LCD de 16x2 caracteres se utiliza simplemente para mostrar la clave que está manejando el sistema actualmente. No es necesario para el producto final y su fin en este prototipo es simplemente mostrar cuan aleatoria es realmente la clave, y también los datos, que son generados con la misma función.

El LED simula un portón abierto o cerrado para este prototipo. No es el objetivo de este trabajo crear el sistema mecánico del portón. El objetivo es crear un sistema electrónico inalámbrico seguro de acceso que después pueda ser implementado en diferentes tipos de portones.

El botón tiene dos funciones. Si se presiona por más de medio segundo genera una nueva clave y guardarla en la memoria no volátil (EEPROM) del PIC. Es bueno cambiar la clave del sistema cada tanto. Si simplemente se presiona y suelta cumple la función de enviar la clave a los mandos para que estos la guarden en su memoria no volátil.

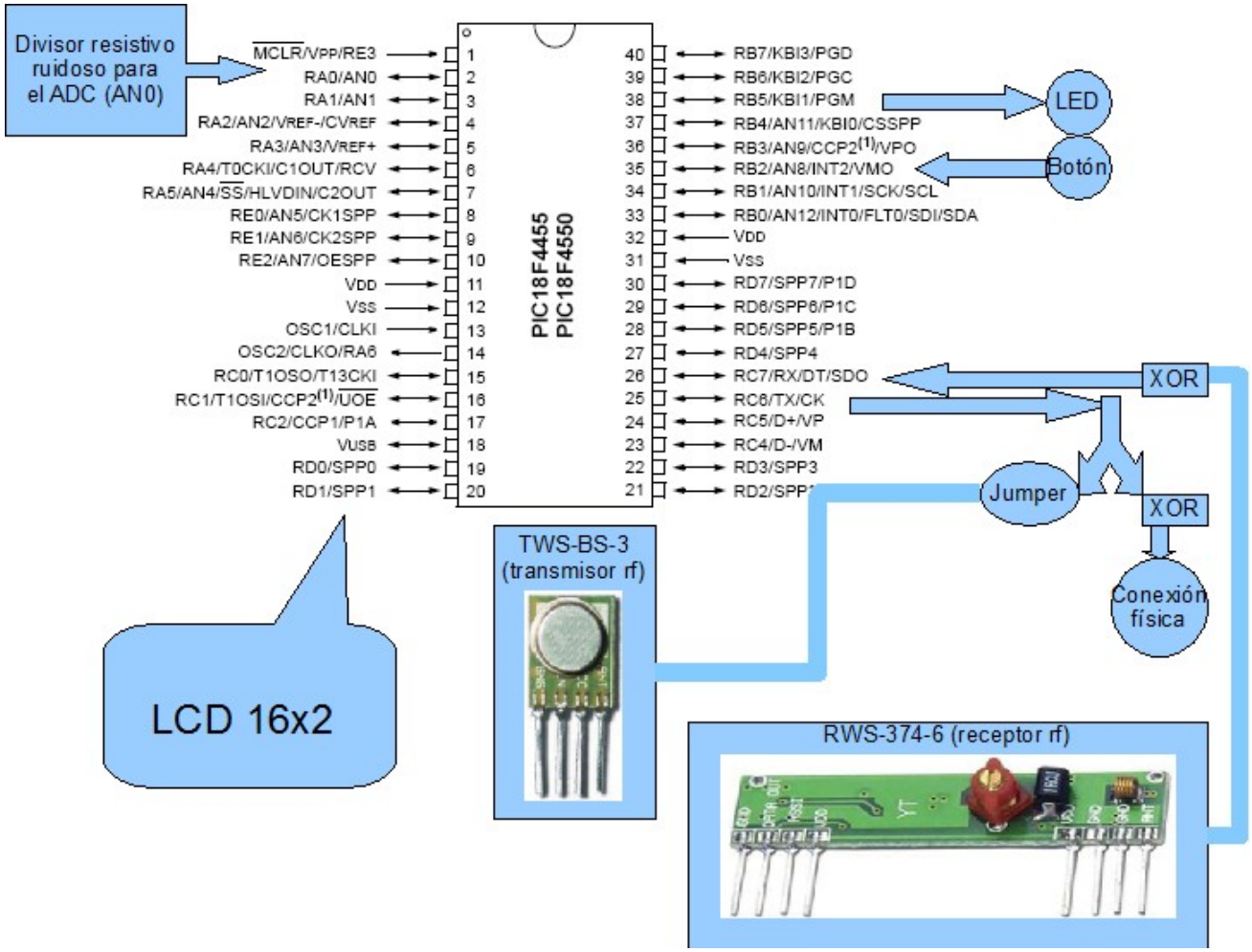


Ilustración 2.2: Controlador del portón

Si no se desea que la clave se emita inalámbricamente se puede desconectar el jumper que une el PIC y el transmisor y unir físicamente (mediante la “conexión física”, que no es otra cosa que un cable de dos conductores) este dispositivo y el mando a distancia.

El proceso se debe repetir para cada mando. O sea, se conecta el mando, se presiona el botón, y se lo desconecta. Luego se conecta el siguiente, se presiona el botón, y se lo desconecta. Etc.

El transmisor y el receptor están conectados al puerto USART del pic y utilizan un protocolo asíncrono (a 3.33 Kbps) como el RS-232 para comunicarse.

Las compuertas XOR sirven de inversor controlado. Están siempre en una configuración inversora (la otra entrada a VCC), pero si se quiere probar una configuración no inversora simplemente hay que soldar la pata a masa. ¿Porqué inversora? Se determinó experimentalmente que los transmisores/receptores utilizados no pueden mantener una tensión en alto por mucho tiempo. Así que el TX del PIC se configuró para enviar con polaridad inversa (así en estado “idle” TX queda en nivel bajo). Por lo tanto era necesario invertir nuevamente al recibir. Se podía hacer por software. Pero se optó por hacer por hardware porque la amplitud a la salida del receptor no era suficiente para la entrada RX del PIC (que es trigger schmitt). Por lo tanto se utilizó un XOR CMOS.

La Ilustración 2.3 muestra la señal antes y después de la compuerta XOR. Antes tiene 3.7 V, y después 4.96 V. Con 3.7 V la señal a veces no llegaba bien, llegaba con errores.

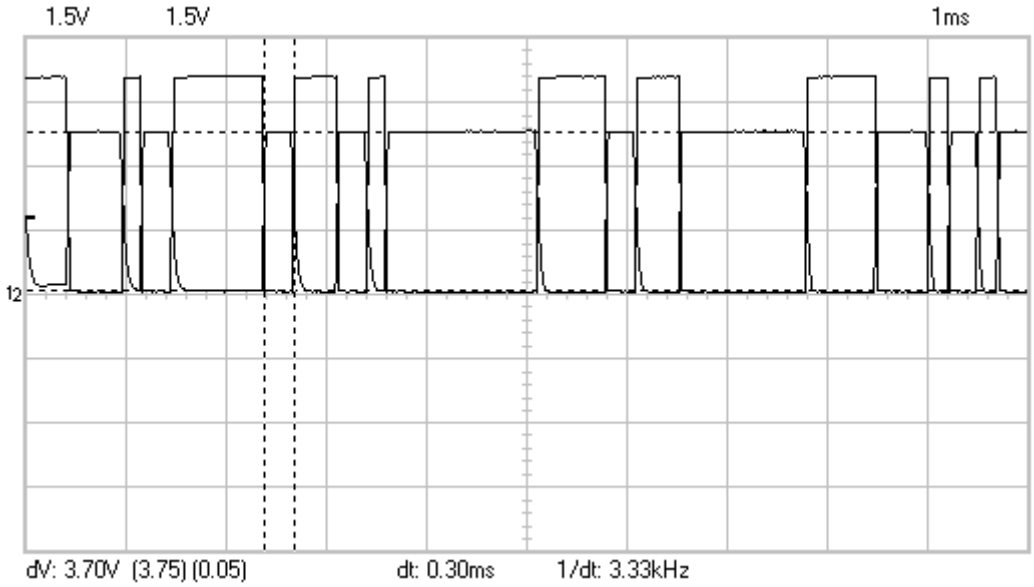


Ilustración 2.3: Antes y despues del XOR

EL PROTOTIPO DEL MANDO A DISTANCIA.

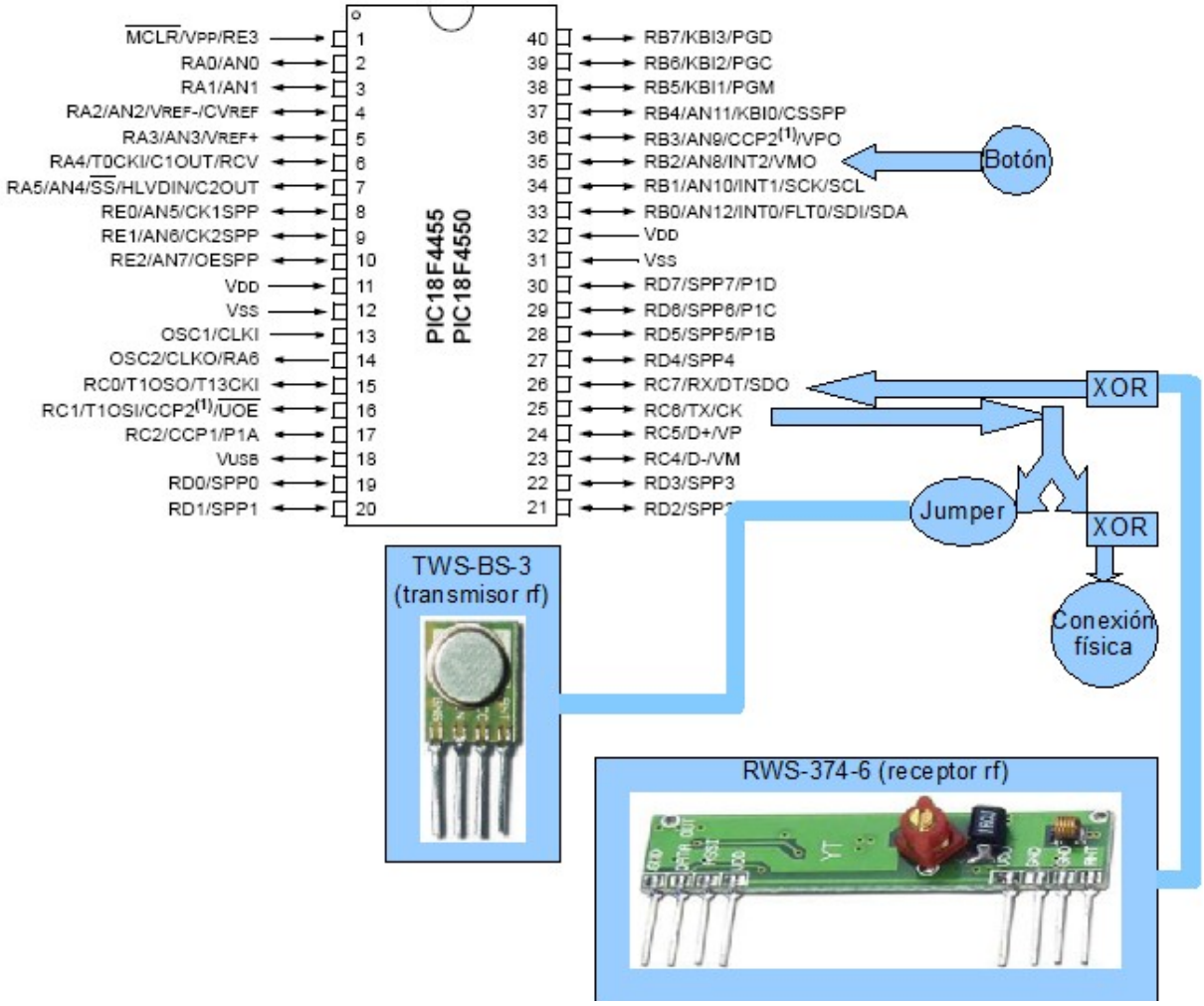


Ilustración 2.4: Mando a distancia

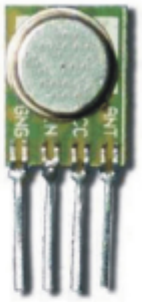
La Ilustración 2.4 muestra un diagrama del mando a distancia. Es igual al controlador del portón pero no tiene LCD, ni LED ni utiliza el conversor AD. Simplemente al detectar que se presionó el botón envía la petición de datos al controlador del portón. Cuando los recibe los encripta y los devuelve. Así abre el portón. También está a la escucha por si se transmite una nueva clave.

EL TRANSMISOR TWS-BS-3.

El transmisor utilizado fue el TWS-BS-3, que modula en ASK a 433.92 Mhz y puede transmitir a 8Kbps. La Tabla 2.1 muestra sus especificaciones.

Transmitter Module : TWS-BS3 (433.92 MHz)

*Frequency Range: 433.92 MHz
*Modulate Mode: ASK
*Circuit Shape: SAW
*Date Rate: 8kbps
*Supply Voltage: 3~ 12 V
*Power Supply and All Input / Output Pins: -0.3 to +12.0 V
*Non-Operating Case Temperature: -20 to +85 ℃
*Soldering Temperature (10 Seconds) : 230 ℃ (10 Seconds)



Absolute Maximum Ratings

Rating	Value	Unit
Power Supply and All Input/Output Pins	-0.3 to +12.0	V
Non-Operating Case Temperature	-20 to +85	℃
Soldering Temperature(10 seconds)	230	℃

Electrical Characteristics, T=25℃, Vcc=3.6v, Freq=433.92MHz

Characteristic	Sym	Min.	Typ.	Max.	Uint
Operating Frequency (200KHz)	Vcc		433.92		MHz
Data Rate	ASK			8K	Kbps
Transmitter Performance(OOK@2.4kbps)					
Peak Input Current,12 Vdc Supply	ITP			45	mA
Peak Output Power	PO		10		mW
Tum On/ Tum Off Time	T ON/T OFF			1	US
Power Supply Vo ltage Range	Vcc	3		12	VDC
Operating Ambient Temperature	TA	-20		+85	℃
Tx Antenna Out (3V) +2.4dB	Vcc				mA

Tabla 2.1: Especificaciones del TWS-BS-3

EL RECEPTOR RWS-374-6.

El receptor utilizado fue el RWS-374-6, que demodula lo que el TWS moduló. Las tablas 2.2 y 2.3 muestran sus características y especificaciones.

➢ Model: RWS-374-6(433.92MHz)

*Frequency Range: 433.92MHz

*Modulate Mode: ASK

*Circuit Shape: LC

*Date Rate: 4800 bps

*Selectivity: -108 dBm

*Channel Spacing: ±500KHz

*Supply Voltage: 5V

* High sensitivity passive design.

*Simple to apply with low external count.

Tabla 2.2: Características del RWS-374-6

➢ Electrical Characteristics :

Characteristics	Sym	Min	Typ	Max	Units
Operating Radio Frequency	FC	433.42	433.92	434.42	MHz
Sensitivity	Pref.	-106	-108	-110	dBm
Channel Width		-500		+ 500	KHz
Noise Equivalent BW	NEB		5	4	
Baseboard Data Rate				3	KB/S
Receiver Tum On Time				3	ms

➢ DC Characteristics :

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Vcc	Operating Supply Voltage		4.9	5	5.1	
I Tot	Operating Supply Voltage			4.5		
V Data	Data Out	1 Data = -10 uA (Low)	Vcc -0.5	Vcc		V
		1 Data = -10 uA (Low)			0.3	V

Tabla 2.3: Especificaciones del RWS-374-6

CAPÍTULO 3: EL SOFTWARE.

El software se desarrolló con el MPLAB IDE v8.36 de Microchip, utilizando la suite C18, también de microchip, que permite programar los PIC18 en lenguaje C.

La Ilustración 3.1 muestra los archivos utilizados para el proyecto del mando a distancia. Se llama camión porque se supone que el mando a distancia es para un camión que quiere entrar a un área restringida.

La Ilustración 3.2 muestra los archivos para el proyecto del portón.

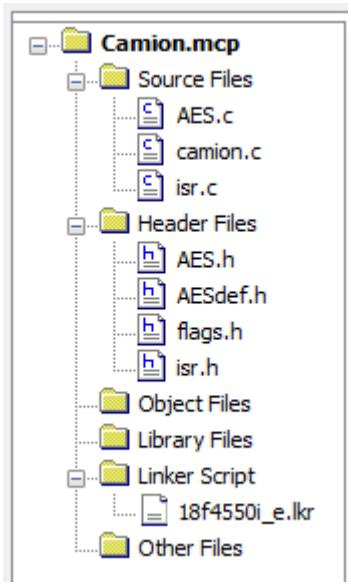


Ilustración 3.1: Proyecto del mando a distancia

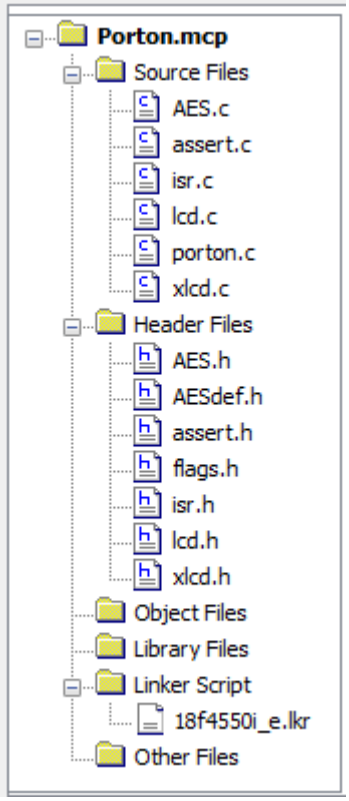


Ilustración 3.2: Proyecto del portón

ARCHIVOS COMUNES A AMBOS PROYECTOS.

A continuación se vertirá el contenido de los archivos fuentes que son comunes a ambos proyectos.

AES.H

```
void AESEncode(unsigned char* block, unsigned char* key);
void AESDecode(unsigned char* block, unsigned char* key);
void AESCalcDecodeKey(unsigned char* key);
```

AESDEF.H

```
#define ENCODE
#define NODECODE
#define NOCALCKEY
```

AES.C

El contenido del archivo está protegido por copyright y no puede ser reproducido. Puede ser obtenido de microchip por 5 dólares estadounidenses. A continuación la nota de copyright.

```
/*********************************************************************
 *
 *              Module for encrypting/decrypting data using
 *              the AES (Rijndael) algorithm.
 *
 *********************************************************************
 * FileName:        AES.c
 * Dependencies:    none
 * Processor:       18FXXX/18CXXX
 * Assembler:       MPASMWIN 02.70.02 or higher
 * Linker:          MPLINK 2.33.00 or higher
 * Company:         Microchip Technology, Inc.
 *
 * Software License Agreement
 *
 * The software supplied herewith by Microchip Technology Incorporated
 * (the "Company") for its PICmicro(r) Microcontroller is intended and
 * supplied to you, the Company's customer, for use solely and
 * exclusively on Microchip PICmicro Microcontroller products. The
 * software is owned by the Company and/or its supplier, and is
 * protected under applicable copyright laws. All rights are reserved.
 * Any use in violation of the foregoing restrictions may subject the
 * user to criminal sanctions under applicable laws, as well as to
 * civil liability for the breach of the terms and conditions of this
 * license.
 *
 * Microchip Technology Inc. ("Microchip") licenses this software to
 * you solely for use with Microchip products.  The software is owned
 * by Microchip and is protected under applicable copyright laws.
 * All rights reserved.
 *
 * You may not export or re-export Software, technical data, direct
 * products thereof or any other items which would violate any applicable
 * export control laws and regulations including, but not limited to,
 * those of the United States or United Kingdom.  You agree that it is
 * your responsibility to obtain copies of and to familiarize yourself
 * fully with these laws and regulations to avoid violation.
 *
 * SOFTWARE IS PROVIDED "AS IS."  MICROCHIP EXPRESSLY DISCLAIM ANY
 * WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL MICROCHIP
 * BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES,
 * LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF PROCUREMENT
 * OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS BY THIRD PARTIES
 * (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), ANY CLAIMS FOR
 * INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.
 *
 *
 *
 *
 * Author               Date        Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
 * David Flowers        05/28/04    Initial Revision
 ************************************************************************/
```

ASSERT.H

```
#ifndef __ASSERT_H
#define __ASSERT_H

#ifdef __DEBUG

#include <stdio.h>

void assertImpl(const near rom char *file,int linea);

     // Usar "debug" como un printf para enviar datos al LCD.
     // Cuando no se define la etiqueta DEBUG esa linea no ocupa ningun
espacio.
     // Ej:
     // debug("%d",ZonaHoraria);
#define debug(x) printf("\ndebug:%S:%d: ",__FILE__,__LINE__);printf(x)

     // Usar "assert" para comprobar si una condicion es valida. Algo que se
sepa que si no es así el programa no debería seguir.
     // Ej:
     // assert((CambiarWO>=0) && (CambiarWO<=1));
#define assert(cond) ((cond)?(void)0:assertImpl(__FILE__,__LINE__))

#else

     #define debug(x) ((void)0)
     #define assert(cond) ((void)0)

#endif

#endif
```

ASSERT.C

```
#include "assert.h"
#include "lcd.h"

void assertImpl(const near rom char *file,int linea){
     LCDClear();
     printf("Fallo:%S:%d",file,linea);
     while(1);
}
```

ISR.H

```
#ifndef __ISR_H
#define __ISR_H

#include <p18cxxx.h>

void low_isr(void);
void high_isr(void);

#endif
```

ISR.C

```
#include "isr.h"

extern void procesarUSART(void);
extern void procesarTecla(void);

#pragma code low_vector=0x18
void interrupt_at_low_vector(void){
     _asm GOTO low_isr _endasm
}
#pragma code

#pragma interruptlow low_isr
void low_isr(void){ // Aquí manejar baja prioridad.
}

#pragma code high_vector=0x08
void interrupt_at_high_vector(void){
     _asm GOTO high_isr _endasm
}
#pragma code

#pragma interrupt high_isr
void high_isr(void){ // Aquí manejar alta prioridad.
     if(PIR1bits.RCIF){ // Recibo por UART. Se limpia sola al leer.
          procesarUSART();
     }
     if(INTCON3bits.INT2IF){ // Tecla pulsada.
          procesarTecla();
          INTCON3bits.INT2IF=0; // Ya la procesé. Limpio por soft.
     }
}
```

18F4550I_E.LKR

```
// File: 18f4550i_e.lkr
// Sample ICD2 linker script for the PIC18F4550 processor

LIBPATH .

FILES c018i_e.o
FILES clib_e.lib
FILES p18f4550_e.lib

CODEPAGE   NAME=vectors    START=0x0           END=0x29           PROTECTED
CODEPAGE   NAME=page       START=0x2A          END=0x7DBF
CODEPAGE   NAME=debug      START=0x7DC0        END=0x7FFF         PROTECTED
CODEPAGE   NAME=idlocs     START=0x200000      END=0x200007       PROTECTED
CODEPAGE   NAME=config     START=0x300000      END=0x30000D       PROTECTED
CODEPAGE   NAME=devid      START=0x3FFFFE      END=0x3FFFFF       PROTECTED
CODEPAGE   NAME=eedata     START=0xF00000      END=0xF000FF       PROTECTED

DATABANK   NAME=gpr0       START=0x0           END=0xFF
DATABANK   NAME=gpr1       START=0x100         END=0x1FF
DATABANK   NAME=gpr2       START=0x200         END=0x2FF
DATABANK   NAME=gpr3       START=0x300         END=0x3FF
DATABANK   NAME=gpr4       START=0x400         END=0x4FF
DATABANK   NAME=gpr5       START=0x500         END=0x5FF
```

```
DATABANK   NAME=gpr6       START=0x600          END=0x6FF
DATABANK   NAME=gpr7       START=0x700          END=0x7F3
DATABANK   NAME=dbgspr     START=0x7F4          END=0x7FF          PROTECTED
ACCESSBANK NAME=accesssfr  START=0xF60          END=0xFFF          PROTECTED

SECTION    NAME=CONFIG     ROM=config

STACK SIZE=0x100 RAM=gpr2
```

ARCHIVOS ÚNICOS DE PORTÓN.

Los siguientes archivos fuente pertenecen únicamente al proyecto “portón”.

LCD.H

```
#ifndef __LCD_H
#define __LCD_H

#include "xlcd.h"
#include <delays.h>

#define LCDCOL 16

#define XLCDCursorOnBlinkOn()         XLCDCommand(0x0F)
#define XLCDCursorOnBlinkOff()        XLCDCommand(0x0E)
#define XLCDCursorOffBlinkOff()   XLCDCommand(0b00001100)
#define XLCDDisplayOnCursorOff()      XLCDCommand(0x0C)
#define XLCDDisplayOff()              XLCDCommand(0x08)
#define XLCDCursorMoveLeft()          XLCDCommand(0x10)
#define XLCDCursorMoveRight()         XLCDCommand(0x14)
#define XLCDDisplayMoveLeft()         XLCDCommand(0x18)
#define XLCDDisplayMoveRight()        XLCDCommand(0x1C)

void LCDClear(void);
int _user_putc(char c);

#endif //__LCD_H
```

LCD.C

```
#include "lcd.h"

void XLCDDelay15ms(void);
void XLCDDelay4ms(void);
void XLCDDelay100us(void);
void XLCD_Delay500ns(void);
void XLCDDelay(void);
void LCDEnter(void);

char fila=0,columna=0;

void LCDEnter(void)
{
     char i;
     columna=0;
     if(fila==0)
     {
```

```
            fila++;
            XLCDCommand(0b11000000);
            for(i=0;i<LCDCOL;i++)
                  XLCDPut(' ');
            XLCDCommand(0b11000000);
      }
      else
      {
            fila--;
            XLCDCommand(0b10000000); // XLCDL2home()
            for(i=0;i<LCDCOL;i++)
                  XLCDPut(' ');
            XLCDCommand(0b10000000); // XLCDL2home()
      }
}

void LCDClear(void)
{
      fila=0;
      columna=0;
      XLCDCommand(0x01);
}

int _user_putc(char c){
      switch(c)
      {
            case '\n':
                  LCDEnter();
                  return c;
            case '\r':
                  LCDEnter();
                  return c;
            case '\b':
                  if(columna==0)
                  {
                        columna=LCDCOL;
                        if(fila==1)
                        {
                              fila=0;
                              XLCDCommand(0b10010100);
                        }
                        else
                        {
                              fila=1;
                              XLCDCommand(0b11010100);
                        }
                  }
                  XLCDCommand(0x10);
                  XLCDPut(' ');
                  XLCDCommand(0x10);
                  columna--;
                  return c;
            default:
                  if(columna>=LCDCOL)
                        LCDEnter();
                  XLCDPut(c);
                  columna++;
                  return c;
```

```
      }
}

void XLCDDelay15ms(void){
      Delay10KTCYx(18);
}

void XLCDDelay4ms(void){
      Delay100TCYx(12);
}

void XLCDDelay100us(void){
      Delay10TCYx(12);
}

void XLCD_Delay500ns(void){
      Nop();Nop();Nop();Nop();Nop();Nop();
}

void  XLCDDelay(void){
      Delay1KTCYx(36);
}
```

XLCD.H

```
/*********************************************************************
 *
 *                  External LCD access routines defs
 *
 *********************************************************************
 * FileName:        XLCD.h
 * Dependencies:    compiler.h
 * Processor:       PIC18
 * Complier:        MCC18 v1.00.50 or higher
 *                  HITECH PICC-18 V8.10PL1 or higher
 * Company:         Microchip Technology, Inc.
 *
 * Software License Agreement
 *
 * The software supplied herewith by Microchip Technology Incorporated
 * (the "Company") for its PICmicro® Microcontroller is intended and
 * supplied to you, the Company's customer, for use solely and
 * exclusively on Microchip PICmicro Microcontroller products. The
 * software is owned by the Company and/or its supplier, and is
 * protected under applicable copyright laws. All rights are reserved.
 * Any use in violation of the foregoing restrictions may subject the
 * user to criminal sanctions under applicable laws, as well as to
 * civil liability for the breach of the terms and conditions of this
 * license.
 *
 * THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
 * WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
 * TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
 * IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 *
 * Author               Date    Comment
```

```
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Naveen Raj     6/9/03  Original        (Rev 1.0)
 ************************************************************************/
#ifndef __XLCD_H
#define __XLCD_H
#define AddFile ///ADD_PROC_INC_FILE
#include "p18cxxx.h"
#include "XLCD.Def"

/* DATA_PORT defines the port to which the LCD data lines are connected */


#if ((XLCD_DATA_PORT ^ 0) == 0)
#define XLCD_DATAPORT       PORTA
#define XLCD_DATAPORT_TRIS  TRISA
#endif

#if ((XLCD_DATA_PORT ^ 1) == 0)
#define XLCD_DATAPORT       PORTB
#define XLCD_DATAPORT_TRIS  TRISB
#endif

#if ((XLCD_DATA_PORT ^ 2) == 0)
#define XLCD_DATAPORT       PORTC
#define XLCD_DATAPORT_TRIS  TRISC
#endif

#if ((XLCD_DATA_PORT ^ 3) == 0)
#define XLCD_DATAPORT       PORTD
#define XLCD_DATAPORT_TRIS  TRISD
#endif

#if ((XLCD_DATA_PORT ^ 4) == 0)
#define XLCD_DATAPORT       PORTE
#define XLCD_DATAPORT_TRIS  TRISE
#endif

#if ((XLCD_DATA_PORT ^ 5) == 0)
#define XLCD_DATAPORT       PORTF
#define XLCD_DATAPORT_TRIS  TRISF
#endif

#if ((XLCD_DATA_PORT ^ 6) == 0)
#define XLCD_DATAPORT       PORTG
#define XLCD_DATAPORT_TRIS  TRISG
#endif

#if ((XLCD_DATA_PORT ^ 7) == 0)
#define XLCD_DATAPORT       PORTH
#define XLCD_DATAPORT_TRIS  TRISH
#endif

#if ((XLCD_DATA_PORT ^ 8) == 0)
#define XLCD_DATAPORT       PORTJ
#define XLCD_DATAPORT_TRIS  TRISJ
#endif

/* CTRL_PORT defines the port where the control lines are connected.
```

```
 * These are just samples, change to match your application.
 */

//ReadWrite Pin
#if ((XLCD_RW_PIN ^ 0x00) == 0)
#define XLCD_RWPIN   PORTAbits.RA0
#define XLCD_RWPIN_TRIS  TRISAbits.TRISA0
#endif
#if ((XLCD_RW_PIN ^ 0x01) == 0)
#define XLCD_RWPIN   PORTAbits.RA1
#define XLCD_RWPIN_TRIS  TRISAbits.TRISA1
#endif
#if ((XLCD_RW_PIN ^ 0x02) == 0)
#define XLCD_RWPIN   PORTAbits.RA2
#define XLCD_RWPIN_TRIS  TRISAbits.TRISA2
#endif
#if ((XLCD_RW_PIN ^ 0x03) == 0)
#define XLCD_RWPIN   PORTAbits.RA3
#define XLCD_RWPIN_TRIS  TRISAbits.TRISA3
#endif
#if ((XLCD_RW_PIN ^ 0x04) == 0)
#define XLCD_RWPIN   PORTAbits.RA4
#define XLCD_RWPIN_TRIS  TRISAbits.TRISA4
#endif
#if ((XLCD_RW_PIN ^ 0x05) == 0)
#define XLCD_RWPIN   PORTAbits.RA5
#define XLCD_RWPIN_TRIS  TRISAbits.TRISA5
#endif

#if ((XLCD_RW_PIN ^ 0x10) == 0)
#define XLCD_RWPIN   PORTBbits.RB0
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB0
#endif
#if ((XLCD_RW_PIN ^ 0x11) == 0)
#define XLCD_RWPIN   PORTBbits.RB1
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB1
#endif
#if ((XLCD_RW_PIN ^ 0x12) == 0)
#define XLCD_RWPIN   PORTBbits.RB2
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB2
#endif
#if ((XLCD_RW_PIN ^ 0x13) == 0)
#define XLCD_RWPIN   PORTBbits.RB3
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB3
#endif
#if ((XLCD_RW_PIN ^ 0x14) == 0)
#define XLCD_RWPIN   PORTBbits.RB4
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB4
#endif
#if ((XLCD_RW_PIN ^ 0x15) == 0)
#define XLCD_RWPIN   PORTBbits.RB5
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB5
#endif
#if ((XLCD_RW_PIN ^ 0x16) == 0)
#define XLCD_RWPIN   PORTBbits.RB6
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB6
#endif
#if ((XLCD_RW_PIN ^ 0x17) == 0)
```

```
#define XLCD_RWPIN   PORTBbits.RB7
#define XLCD_RWPIN_TRIS  TRISBbits.TRISB7
#endif

#if ((XLCD_RW_PIN ^ 0x20) == 0)
#define XLCD_RWPIN   PORTCbits.RC0
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC0
#endif
#if ((XLCD_RW_PIN ^ 0x21) == 0)
#define XLCD_RWPIN   PORTCbits.RC1
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC1
#endif
#if ((XLCD_RW_PIN ^ 0x22) == 0)
#define XLCD_RWPIN   PORTCbits.RC2
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC2
#endif
#if ((XLCD_RW_PIN ^ 0x23) == 0)
#define XLCD_RWPIN   PORTCbits.RC3
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC3
#endif
#if ((XLCD_RW_PIN ^ 0x24) == 0)
#define XLCD_RWPIN   PORTCbits.RC4
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC4
#endif
#if ((XLCD_RW_PIN ^ 0x25) == 0)
#define XLCD_RWPIN   PORTCbits.RC5
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC5
#endif
#if ((XLCD_RW_PIN ^ 0x26) == 0)
#define XLCD_RWPIN   PORTCbits.RC6
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC6
#endif
#if ((XLCD_RW_PIN ^ 0x27) == 0)
#define XLCD_RWPIN   PORTCbits.RC7
#define XLCD_RWPIN_TRIS  TRISCbits.TRISC7
#endif

#if ((XLCD_RW_PIN ^ 0x30) == 0)
#define XLCD_RWPIN   PORTDbits.RB0
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD0
#endif
#if ((XLCD_RW_PIN ^ 0x31) == 0)
#define XLCD_RWPIN   PORTDbits.RD1
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD1
#endif
#if ((XLCD_RW_PIN ^ 0x32) == 0)
#define XLCD_RWPIN   PORTDbits.RD2
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD2
#endif
#if ((XLCD_RW_PIN ^ 0x33) == 0)
#define XLCD_RWPIN   PORTDbits.RD3
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD3
#endif
#if ((XLCD_RW_PIN ^ 0x34) == 0)
#define XLCD_RWPIN   PORTDbits.RD4
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD4
#endif
#if ((XLCD_RW_PIN ^ 0x35) == 0)
```

```
#define XLCD_RWPIN   PORTDbits.RD5
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD5
#endif
#if ((XLCD_RW_PIN ^ 0x36) == 0)
#define XLCD_RWPIN   PORTDbits.RD6
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD6
#endif
#if ((XLCD_RW_PIN ^ 0x37) == 0)
#define XLCD_RWPIN   PORTDbits.RD7
#define XLCD_RWPIN_TRIS  TRISDbits.TRISD7
#endif

#if ((XLCD_RW_PIN ^ 0x40) == 0)
#define XLCD_RWPIN   PORTEbits.RE0
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE0
#endif
#if ((XLCD_RW_PIN ^ 0x41) == 0)
#define XLCD_RWPIN   PORTEbits.RE1
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE1
#endif
#if ((XLCD_RW_PIN ^ 0x42) == 0)
#define XLCD_RWPIN   PORTEbits.RE2
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE2
#endif
#if ((XLCD_RW_PIN ^ 0x43) == 0)
#define XLCD_RWPIN   PORTEbits.RE3
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE3
#endif
#if ((XLCD_RW_PIN ^ 0x44) == 0)
#define XLCD_RWPIN   PORTEbits.RE4
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE4
#endif
#if ((XLCD_RW_PIN ^ 0x45) == 0)
#define XLCD_RWPIN   PORTEbits.RE5
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE5
#endif
#if ((XLCD_RW_PIN ^ 0x46) == 0)
#define XLCD_RWPIN   PORTEbits.RE6
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE6
#endif
#if ((XLCD_RW_PIN ^ 0x47) == 0)
#define XLCD_RWPIN   PORTEbits.RE7
#define XLCD_RWPIN_TRIS  TRISEbits.TRISE7
#endif

#if ((XLCD_RW_PIN ^ 0x50) == 0)
#define XLCD_RWPIN   PORTFbits.RF0
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF0
#endif
#if ((XLCD_RW_PIN ^ 0x51) == 0)
#define XLCD_RWPIN   PORTFbits.RF1
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF1
#endif
#if ((XLCD_RW_PIN ^ 0x52) == 0)
#define XLCD_RWPIN   PORTFbits.RF2
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF2
#endif
#if ((XLCD_RW_PIN ^ 0x53) == 0)
```

```
#define XLCD_RWPIN   PORTFbits.RF3
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF3
#endif
#if ((XLCD_RW_PIN ^ 0x54) == 0)
#define XLCD_RWPIN   PORTFbits.RF4
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF4
#endif
#if ((XLCD_RW_PIN ^ 0x55) == 0)
#define XLCD_RWPIN   PORTFbits.RF5
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF5
#endif
#if ((XLCD_RW_PIN ^ 0x56) == 0)
#define XLCD_RWPIN   PORTFbits.RF6
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF6
#endif
#if ((XLCD_RW_PIN ^ 0x57) == 0)
#define XLCD_RWPIN   PORTFbits.RF7
#define XLCD_RWPIN_TRIS  TRISFbits.TRISF7
#endif

#if ((XLCD_RW_PIN ^ 0x60) == 0)
#define XLCD_RWPIN   PORTGbits.RG0
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG0
#endif
#if ((XLCD_RW_PIN ^ 0x61) == 0)
#define XLCD_RWPIN   PORTGbits.RG1
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG1
#endif
#if ((XLCD_RW_PIN ^ 0x62) == 0)
#define XLCD_RWPIN   PORTGbits.RG2
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG2
#endif
#if ((XLCD_RW_PIN ^ 0x63) == 0)
#define XLCD_RWPIN   PORTGbits.RG3
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG3
#endif
#if ((XLCD_RW_PIN ^ 0x64) == 0)
#define XLCD_RWPIN   PORTGbits.RG4
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG4
#endif
#if ((XLCD_RW_PIN ^ 0x65) == 0)
#define XLCD_RWPIN   PORTGbits.RG5
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG5
#endif
#if ((XLCD_RW_PIN ^ 0x66) == 0)
#define XLCD_RWPIN   PORTGbits.RG6
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG6
#endif
#if ((XLCD_RW_PIN ^ 0x67) == 0)
#define XLCD_RWPIN   PORTGbits.RG7
#define XLCD_RWPIN_TRIS  TRISGbits.TRISG7
#endif

#if ((XLCD_RW_PIN ^ 0x70) == 0)
#define XLCD_RWPIN   PORTHbits.RH0
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH0
#endif
#if ((XLCD_RW_PIN ^ 0x71) == 0)
```

```
#define XLCD_RWPIN   PORTHbits.RH1
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH1
#endif
#if ((XLCD_RW_PIN ^ 0x72) == 0)
#define XLCD_RWPIN   PORTHbits.RH2
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH2
#endif
#if ((XLCD_RW_PIN ^ 0x73) == 0)
#define XLCD_RWPIN   PORTHbits.RH3
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH3
#endif
#if ((XLCD_RW_PIN ^ 0x74) == 0)
#define XLCD_RWPIN   PORTHbits.RH4
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH4
#endif
#if ((XLCD_RW_PIN ^ 0x75) == 0)
#define XLCD_RWPIN   PORTHbits.RH5
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH5
#endif
#if ((XLCD_RW_PIN ^ 0x76) == 0)
#define XLCD_RWPIN   PORTHbits.RH6
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH6
#endif
#if ((XLCD_RW_PIN ^ 0x77) == 0)
#define XLCD_RWPIN   PORTHbits.RH7
#define XLCD_RWPIN_TRIS  TRISHbits.TRISH7
#endif

#if ((XLCD_RW_PIN ^ 0x80) == 0)
#define XLCD_RWPIN   PORTJbits.RJ0
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ0
#endif
#if ((XLCD_RW_PIN ^ 0x81) == 0)
#define XLCD_RWPIN   PORTJbits.RJ1
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ1
#endif
#if ((XLCD_RW_PIN ^ 0x82) == 0)
#define XLCD_RWPIN   PORTJbits.RJ2
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ2
#endif
#if ((XLCD_RW_PIN ^ 0x83) == 0)
#define XLCD_RWPIN   PORTJbits.RJ3
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ3
#endif
#if ((XLCD_RW_PIN ^ 0x84) == 0)
#define XLCD_RWPIN   PORTJbits.RJ4
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ4
#endif
#if ((XLCD_RW_PIN ^ 0x85) == 0)
#define XLCD_RWPIN   PORTJbits.RJ5
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ5
#endif
#if ((XLCD_RW_PIN ^ 0x86) == 0)
#define XLCD_RWPIN   PORTJbits.RJ6
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ6
#endif
#if ((XLCD_RW_PIN ^ 0x87) == 0)
#define XLCD_RWPIN   PORTJbits.RJ7
```

```
#define XLCD_RWPIN_TRIS  TRISJbits.TRISJ7
#endif

////////////////////////////////////////////////////////////////////////////
//

//RS Pin
////////////////////////////////////////////////////////////////////////////
//

#if ((XLCD_RS_PIN ^ 0x00) == 0)
#define XLCD_RSPIN   PORTAbits.RA0
#define XLCD_RSPIN_TRIS  TRISAbits.TRISA0
#endif
#if ((XLCD_RS_PIN ^ 0x01) == 0)
#define XLCD_RSPIN   PORTAbits.RA1
#define XLCD_RSPIN_TRIS  TRISAbits.TRISA1
#endif
#if ((XLCD_RS_PIN ^ 0x02) == 0)
#define XLCD_RSPIN   PORTAbits.RA2
#define XLCD_RSPIN_TRIS  TRISAbits.TRISA2
#endif
#if ((XLCD_RS_PIN ^ 0x03) == 0)
#define XLCD_RSPIN   PORTAbits.RA3
#define XLCD_RSPIN_TRIS  TRISAbits.TRISA3
#endif
#if ((XLCD_RS_PIN ^ 0x04) == 0)
#define XLCD_RSPIN   PORTAbits.RA4
#define XLCD_RSPIN_TRIS  TRISAbits.TRISA4
#endif
#if ((XLCD_RS_PIN ^ 0x05) == 0)
#define XLCD_RSPIN   PORTAbits.RA5
#define XLCD_RSPIN_TRIS  TRISAbits.TRISA5
#endif

#if ((XLCD_RS_PIN ^ 0x10) == 0)
#define XLCD_RSPIN   PORTBbits.RB0
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB0
#endif
#if ((XLCD_RS_PIN ^ 0x11) == 0)
#define XLCD_RSPIN   PORTBbits.RB1
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB1
#endif
#if ((XLCD_RS_PIN ^ 0x12) == 0)
#define XLCD_RSPIN   PORTBbits.RB2
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB2
#endif
#if ((XLCD_RS_PIN ^ 0x13) == 0)
#define XLCD_RSPIN   PORTBbits.RB3
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB3
#endif
#if ((XLCD_RS_PIN ^ 0x14) == 0)
#define XLCD_RSPIN   PORTBbits.RB4
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB4
#endif
#if ((XLCD_RS_PIN ^ 0x15) == 0)
#define XLCD_RSPIN   PORTBbits.RB5
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB5
```

```
#endif
#if ((XLCD_RS_PIN ^ 0x16) == 0)
#define XLCD_RSPIN   PORTBbits.RB6
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB6
#endif
#if ((XLCD_RS_PIN ^ 0x17) == 0)
#define XLCD_RSPIN   PORTBbits.RB7
#define XLCD_RSPIN_TRIS  TRISBbits.TRISB7
#endif

#if ((XLCD_RS_PIN ^ 0x20) == 0)
#define XLCD_RSPIN   PORTCbits.RC0
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC0
#endif
#if ((XLCD_RS_PIN ^ 0x21) == 0)
#define XLCD_RSPIN   PORTCbits.RC1
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC1
#endif
#if ((XLCD_RS_PIN ^ 0x22) == 0)
#define XLCD_RSPIN   PORTCbits.RC2
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC2
#endif
#if ((XLCD_RS_PIN ^ 0x23) == 0)
#define XLCD_RSPIN   PORTCbits.RC3
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC3
#endif
#if ((XLCD_RS_PIN ^ 0x24) == 0)
#define XLCD_RSPIN   PORTCbits.RC4
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC4
#endif
#if ((XLCD_RS_PIN ^ 0x25) == 0)
#define XLCD_RSPIN   PORTCbits.RC5
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC5
#endif
#if ((XLCD_RS_PIN ^ 0x26) == 0)
#define XLCD_RSPIN   PORTCbits.RC6
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC6
#endif
#if ((XLCD_RS_PIN ^ 0x27) == 0)
#define XLCD_RSPIN   PORTCbits.RC7
#define XLCD_RSPIN_TRIS  TRISCbits.TRISC7
#endif

#if ((XLCD_RS_PIN ^ 0x30) == 0)
#define XLCD_RSPIN   PORTDbits.RB0
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD0
#endif
#if ((XLCD_RS_PIN ^ 0x31) == 0)
#define XLCD_RSPIN   PORTDbits.RD1
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD1
#endif
#if ((XLCD_RS_PIN ^ 0x32) == 0)
#define XLCD_RSPIN   PORTDbits.RD2
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD2
#endif
#if ((XLCD_RS_PIN ^ 0x33) == 0)
#define XLCD_RSPIN   PORTDbits.RD3
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD3
```

```
#endif
#if ((XLCD_RS_PIN ^ 0x34) == 0)
#define XLCD_RSPIN   PORTDbits.RD4
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD4
#endif
#if ((XLCD_RS_PIN ^ 0x35) == 0)
#define XLCD_RSPIN   PORTDbits.RD5
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD5
#endif
#if ((XLCD_RS_PIN ^ 0x36) == 0)
#define XLCD_RSPIN   PORTDbits.RD6
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD6
#endif
#if ((XLCD_RS_PIN ^ 0x37) == 0)
#define XLCD_RSPIN   PORTDbits.RD7
#define XLCD_RSPIN_TRIS  TRISDbits.TRISD7
#endif

#if ((XLCD_RS_PIN ^ 0x40) == 0)
#define XLCD_RSPIN   PORTEbits.RE0
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE0
#endif
#if ((XLCD_RS_PIN ^ 0x41) == 0)
#define XLCD_RSPIN   PORTEbits.RE1
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE1
#endif
#if ((XLCD_RS_PIN ^ 0x42) == 0)
#define XLCD_RSPIN   PORTEbits.RE2
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE2
#endif
#if ((XLCD_RS_PIN ^ 0x43) == 0)
#define XLCD_RSPIN   PORTEbits.RE3
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE3
#endif
#if ((XLCD_RS_PIN ^ 0x44) == 0)
#define XLCD_RSPIN   PORTEbits.RE4
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE4
#endif
#if ((XLCD_RS_PIN ^ 0x45) == 0)
#define XLCD_RSPIN   PORTEbits.RE5
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE5
#endif
#if ((XLCD_RS_PIN ^ 0x46) == 0)
#define XLCD_RSPIN   PORTEbits.RE6
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE6
#endif
#if ((XLCD_RS_PIN ^ 0x47) == 0)
#define XLCD_RSPIN   PORTEbits.RE7
#define XLCD_RSPIN_TRIS  TRISEbits.TRISE7
#endif

#if ((XLCD_RS_PIN ^ 0x50) == 0)
#define XLCD_RSPIN   PORTFbits.RF0
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF0
#endif
#if ((XLCD_RS_PIN ^ 0x51) == 0)
#define XLCD_RSPIN   PORTFbits.RF1
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF1
```

```
#endif
#if ((XLCD_RS_PIN ^ 0x52) == 0)
#define XLCD_RSPIN   PORTFbits.RF2
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF2
#endif
#if ((XLCD_RS_PIN ^ 0x53) == 0)
#define XLCD_RSPIN   PORTFbits.RF3
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF3
#endif
#if ((XLCD_RS_PIN ^ 0x54) == 0)
#define XLCD_RSPIN   PORTFbits.RF4
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF4
#endif
#if ((XLCD_RS_PIN ^ 0x55) == 0)
#define XLCD_RSPIN   PORTFbits.RF5
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF5
#endif
#if ((XLCD_RS_PIN ^ 0x56) == 0)
#define XLCD_RSPIN   PORTFbits.RF6
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF6
#endif
#if ((XLCD_RS_PIN ^ 0x57) == 0)
#define XLCD_RSPIN   PORTFbits.RF7
#define XLCD_RSPIN_TRIS  TRISFbits.TRISF7
#endif

#if ((XLCD_RS_PIN ^ 0x60) == 0)
#define XLCD_RSPIN   PORTGbits.RG0
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG0
#endif
#if ((XLCD_RS_PIN ^ 0x61) == 0)
#define XLCD_RSPIN   PORTGbits.RG1
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG1
#endif
#if ((XLCD_RS_PIN ^ 0x62) == 0)
#define XLCD_RSPIN   PORTGbits.RG2
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG2
#endif
#if ((XLCD_RS_PIN ^ 0x63) == 0)
#define XLCD_RSPIN   PORTGbits.RG3
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG3
#endif
#if ((XLCD_RS_PIN ^ 0x64) == 0)
#define XLCD_RSPIN   PORTGbits.RG4
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG4
#endif
#if ((XLCD_RS_PIN ^ 0x65) == 0)
#define XLCD_RSPIN   PORTGbits.RG5
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG5
#endif
#if ((XLCD_RS_PIN ^ 0x66) == 0)
#define XLCD_RSPIN   PORTGbits.RG6
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG6
#endif
#if ((XLCD_RS_PIN ^ 0x67) == 0)
#define XLCD_RSPIN   PORTGbits.RG7
#define XLCD_RSPIN_TRIS  TRISGbits.TRISG7
#endif
```

```
#if ((XLCD_RS_PIN ^ 0x70) == 0)
#define XLCD_RSPIN   PORTHbits.RH0
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH0
#endif
#if ((XLCD_RS_PIN ^ 0x71) == 0)
#define XLCD_RSPIN   PORTHbits.RH1
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH1
#endif
#if ((XLCD_RS_PIN ^ 0x72) == 0)
#define XLCD_RSPIN   PORTHbits.RH2
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH2
#endif
#if ((XLCD_RS_PIN ^ 0x73) == 0)
#define XLCD_RSPIN   PORTHbits.RH3
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH3
#endif
#if ((XLCD_RS_PIN ^ 0x74) == 0)
#define XLCD_RSPIN   PORTHbits.RH4
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH4
#endif
#if ((XLCD_RS_PIN ^ 0x75) == 0)
#define XLCD_RSPIN   PORTHbits.RH5
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH5
#endif
#if ((XLCD_RS_PIN ^ 0x76) == 0)
#define XLCD_RSPIN   PORTHbits.RH6
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH6
#endif
#if ((XLCD_RS_PIN ^ 0x77) == 0)
#define XLCD_RSPIN   PORTHbits.RH7
#define XLCD_RSPIN_TRIS  TRISHbits.TRISH7
#endif

#if ((XLCD_RS_PIN ^ 0x80) == 0)
#define XLCD_RSPIN   PORTJbits.RJ0
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ0
#endif
#if ((XLCD_RS_PIN ^ 0x81) == 0)
#define XLCD_RSPIN   PORTJbits.RJ1
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ1
#endif
#if ((XLCD_RS_PIN ^ 0x82) == 0)
#define XLCD_RSPIN   PORTJbits.RJ2
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ2
#endif
#if ((XLCD_RS_PIN ^ 0x83) == 0)
#define XLCD_RSPIN   PORTJbits.RJ3
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ3
#endif
#if ((XLCD_RS_PIN ^ 0x84) == 0)
#define XLCD_RSPIN   PORTJbits.RJ4
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ4
#endif
#if ((XLCD_RS_PIN ^ 0x85) == 0)
#define XLCD_RSPIN   PORTJbits.RJ5
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ5
#endif
```

```
#if ((XLCD_RS_PIN ^ 0x86) == 0)
#define XLCD_RSPIN   PORTJbits.RJ6
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ6
#endif
#if ((XLCD_RS_PIN ^ 0x87) == 0)
#define XLCD_RSPIN   PORTJbits.RJ7
#define XLCD_RSPIN_TRIS  TRISJbits.TRISJ7
#endif

////////////////////////////////////////////////////////////////////////////
//

//Enable Pin
////////////////////////////////////////////////////////////////////////////
//
#if ((XLCD_EN_PIN ^ 0x00) == 0)
#define XLCD_ENPIN   PORTAbits.RA0
#define XLCD_ENPIN_TRIS  TRISAbits.TRISA0
#endif
#if ((XLCD_EN_PIN ^ 0x01) == 0)
#define XLCD_ENPIN   PORTAbits.RA1
#define XLCD_ENPIN_TRIS  TRISAbits.TRISA1
#endif
#if ((XLCD_EN_PIN ^ 0x02) == 0)
#define XLCD_ENPIN   PORTAbits.RA2
#define XLCD_ENPIN_TRIS  TRISAbits.TRISA2
#endif
#if ((XLCD_EN_PIN ^ 0x03) == 0)
#define XLCD_ENPIN   PORTAbits.RA3
#define XLCD_ENPIN_TRIS  TRISAbits.TRISA3
#endif
#if ((XLCD_EN_PIN ^ 0x04) == 0)
#define XLCD_ENPIN   PORTAbits.RA4
#define XLCD_ENPIN_TRIS  TRISAbits.TRISA4
#endif
#if ((XLCD_EN_PIN ^ 0x05) == 0)
#define XLCD_ENPIN   PORTAbits.RA5
#define XLCD_ENPIN_TRIS  TRISAbits.TRISA5
#endif

#if ((XLCD_EN_PIN ^ 0x10) == 0)
#define XLCD_ENPIN   PORTBbits.RB0
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB0
#endif
#if ((XLCD_EN_PIN ^ 0x11) == 0)
#define XLCD_ENPIN   PORTBbits.RB1
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB1
#endif
#if ((XLCD_EN_PIN ^ 0x12) == 0)
#define XLCD_ENPIN   PORTBbits.RB2
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB2
#endif
#if ((XLCD_EN_PIN ^ 0x13) == 0)
#define XLCD_ENPIN   PORTBbits.RB3
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB3
#endif
#if ((XLCD_EN_PIN ^ 0x14) == 0)
#define XLCD_ENPIN   PORTBbits.RB4
```

```
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB4
#endif
#if ((XLCD_EN_PIN ^ 0x15) == 0)
#define XLCD_ENPIN   PORTBbits.RB5
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB5
#endif
#if ((XLCD_EN_PIN ^ 0x16) == 0)
#define XLCD_ENPIN   PORTBbits.RB6
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB6
#endif
#if ((XLCD_EN_PIN ^ 0x17) == 0)
#define XLCD_ENPIN   PORTBbits.RB7
#define XLCD_ENPIN_TRIS  TRISBbits.TRISB7
#endif

#if ((XLCD_EN_PIN ^ 0x20) == 0)
#define XLCD_ENPIN   PORTCbits.RC0
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC0
#endif
#if ((XLCD_EN_PIN ^ 0x21) == 0)
#define XLCD_ENPIN   PORTCbits.RC1
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC1
#endif
#if ((XLCD_EN_PIN ^ 0x22) == 0)
#define XLCD_ENPIN   PORTCbits.RC2
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC2
#endif
#if ((XLCD_EN_PIN ^ 0x23) == 0)
#define XLCD_ENPIN   PORTCbits.RC3
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC3
#endif
#if ((XLCD_EN_PIN ^ 0x24) == 0)
#define XLCD_ENPIN   PORTCbits.RC4
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC4
#endif
#if ((XLCD_EN_PIN ^ 0x25) == 0)
#define XLCD_ENPIN   PORTCbits.RC5
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC5
#endif
#if ((XLCD_EN_PIN ^ 0x26) == 0)
#define XLCD_ENPIN   PORTCbits.RC6
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC6
#endif
#if ((XLCD_EN_PIN ^ 0x27) == 0)
#define XLCD_ENPIN   PORTCbits.RC7
#define XLCD_ENPIN_TRIS  TRISCbits.TRISC7
#endif

#if ((XLCD_EN_PIN ^ 0x30) == 0)
#define XLCD_ENPIN   PORTDbits.RB0
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD0
#endif
#if ((XLCD_EN_PIN ^ 0x31) == 0)
#define XLCD_ENPIN   PORTDbits.RD1
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD1
#endif
#if ((XLCD_EN_PIN ^ 0x32) == 0)
#define XLCD_ENPIN   PORTDbits.RD2
```

```
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD2
#endif
#if ((XLCD_EN_PIN ^ 0x33) == 0)
#define XLCD_ENPIN   PORTDbits.RD3
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD3
#endif
#if ((XLCD_EN_PIN ^ 0x34) == 0)
#define XLCD_ENPIN   PORTDbits.RD4
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD4
#endif
#if ((XLCD_EN_PIN ^ 0x35) == 0)
#define XLCD_ENPIN   PORTDbits.RD5
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD5
#endif
#if ((XLCD_EN_PIN ^ 0x36) == 0)
#define XLCD_ENPIN   PORTDbits.RD6
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD6
#endif
#if ((XLCD_EN_PIN ^ 0x37) == 0)
#define XLCD_ENPIN   PORTDbits.RD7
#define XLCD_ENPIN_TRIS  TRISDbits.TRISD7
#endif

#if ((XLCD_EN_PIN ^ 0x40) == 0)
#define XLCD_ENPIN   PORTEbits.RE0
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE0
#endif
#if ((XLCD_EN_PIN ^ 0x41) == 0)
#define XLCD_ENPIN   PORTEbits.RE1
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE1
#endif
#if ((XLCD_EN_PIN ^ 0x42) == 0)
#define XLCD_ENPIN   PORTEbits.RE2
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE2
#endif
#if ((XLCD_EN_PIN ^ 0x43) == 0)
#define XLCD_ENPIN   PORTEbits.RE3
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE3
#endif
#if ((XLCD_EN_PIN ^ 0x44) == 0)
#define XLCD_ENPIN   PORTEbits.RE4
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE4
#endif
#if ((XLCD_EN_PIN ^ 0x45) == 0)
#define XLCD_ENPIN   PORTEbits.RE5
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE5
#endif
#if ((XLCD_EN_PIN ^ 0x46) == 0)
#define XLCD_ENPIN   PORTEbits.RE6
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE6
#endif
#if ((XLCD_EN_PIN ^ 0x47) == 0)
#define XLCD_ENPIN   PORTEbits.RE7
#define XLCD_ENPIN_TRIS  TRISEbits.TRISE7
#endif

#if ((XLCD_EN_PIN ^ 0x50) == 0)
#define XLCD_ENPIN   PORTFbits.RF0
```

```
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF0
#endif
#if ((XLCD_EN_PIN ^ 0x51) == 0)
#define XLCD_ENPIN   PORTFbits.RF1
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF1
#endif
#if ((XLCD_EN_PIN ^ 0x52) == 0)
#define XLCD_ENPIN   PORTFbits.RF2
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF2
#endif
#if ((XLCD_EN_PIN ^ 0x53) == 0)
#define XLCD_ENPIN   PORTFbits.RF3
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF3
#endif
#if ((XLCD_EN_PIN ^ 0x54) == 0)
#define XLCD_ENPIN   PORTFbits.RF4
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF4
#endif
#if ((XLCD_EN_PIN ^ 0x55) == 0)
#define XLCD_ENPIN   PORTFbits.RF5
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF5
#endif
#if ((XLCD_EN_PIN ^ 0x56) == 0)
#define XLCD_ENPIN   PORTFbits.RF6
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF6
#endif
#if ((XLCD_EN_PIN ^ 0x57) == 0)
#define XLCD_ENPIN   PORTFbits.RF7
#define XLCD_ENPIN_TRIS  TRISFbits.TRISF7
#endif

#if ((XLCD_EN_PIN ^ 0x60) == 0)
#define XLCD_ENPIN   PORTGbits.RG0
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG0
#endif
#if ((XLCD_EN_PIN ^ 0x61) == 0)
#define XLCD_ENPIN   PORTGbits.RG1
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG1
#endif
#if ((XLCD_EN_PIN ^ 0x62) == 0)
#define XLCD_ENPIN   PORTGbits.RG2
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG2
#endif
#if ((XLCD_EN_PIN ^ 0x63) == 0)
#define XLCD_ENPIN   PORTGbits.RG3
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG3
#endif
#if ((XLCD_EN_PIN ^ 0x64) == 0)
#define XLCD_ENPIN   PORTGbits.RG4
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG4
#endif
#if ((XLCD_EN_PIN ^ 0x65) == 0)
#define XLCD_ENPIN   PORTGbits.RG5
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG5
#endif
#if ((XLCD_EN_PIN ^ 0x66) == 0)
#define XLCD_ENPIN   PORTGbits.RG6
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG6
```

```
#endif
#if ((XLCD_EN_PIN ^ 0x67) == 0)
#define XLCD_ENPIN   PORTGbits.RG7
#define XLCD_ENPIN_TRIS  TRISGbits.TRISG7
#endif

#if ((XLCD_EN_PIN ^ 0x70) == 0)
#define XLCD_ENPIN   PORTHbits.RH0
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH0
#endif
#if ((XLCD_EN_PIN ^ 0x71) == 0)
#define XLCD_ENPIN   PORTHbits.RH1
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH1
#endif
#if ((XLCD_EN_PIN ^ 0x72) == 0)
#define XLCD_ENPIN   PORTHbits.RH2
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH2
#endif
#if ((XLCD_EN_PIN ^ 0x73) == 0)
#define XLCD_ENPIN   PORTHbits.RH3
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH3
#endif
#if ((XLCD_EN_PIN ^ 0x74) == 0)
#define XLCD_ENPIN   PORTHbits.RH4
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH4
#endif
#if ((XLCD_EN_PIN ^ 0x75) == 0)
#define XLCD_ENPIN   PORTHbits.RH5
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH5
#endif
#if ((XLCD_EN_PIN ^ 0x76) == 0)
#define XLCD_ENPIN   PORTHbits.RH6
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH6
#endif
#if ((XLCD_EN_PIN ^ 0x77) == 0)
#define XLCD_ENPIN   PORTHbits.RH7
#define XLCD_ENPIN_TRIS  TRISHbits.TRISH7
#endif

#if ((XLCD_EN_PIN ^ 0x80) == 0)
#define XLCD_ENPIN   PORTJbits.RJ0
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ0
#endif
#if ((XLCD_EN_PIN ^ 0x81) == 0)
#define XLCD_ENPIN   PORTJbits.RJ1
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ1
#endif
#if ((XLCD_EN_PIN ^ 0x82) == 0)
#define XLCD_ENPIN   PORTJbits.RJ2
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ2
#endif
#if ((XLCD_EN_PIN ^ 0x83) == 0)
#define XLCD_ENPIN   PORTJbits.RJ3
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ3
#endif
#if ((XLCD_EN_PIN ^ 0x84) == 0)
#define XLCD_ENPIN   PORTJbits.RJ4
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ4
```

```
#endif
#if ((XLCD_EN_PIN ^ 0x85) == 0)
#define XLCD_ENPIN   PORTJbits.RJ5
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ5
#endif
#if ((XLCD_EN_PIN ^ 0x86) == 0)
#define XLCD_ENPIN   PORTJbits.RJ6
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ6
#endif
#if ((XLCD_EN_PIN ^ 0x87) == 0)
#define XLCD_ENPIN   PORTJbits.RJ7
#define XLCD_ENPIN_TRIS  TRISJbits.TRISJ7
#endif




void XLCDInit(void);                                    //to initialise the LCD
void XLCDPut(char data);                                //to put dtat to be
displayed
void XLCDPutRamString(char *string);                    //to display data string
in RAM
void XLCDPutRomString(rom char *string);                //to display data
stringin ROM
char XLCDIsBusy(void);                                  //to check Busy flag
void XLCDCommand(unsigned char cmd);                    //to send commands to
LCD
unsigned char XLCDGetAddr(void);
char XLCDGet(void);

#define XLCDL1home()    XLCDCommand(0x80)
#define XLCDL2home()    XLCDCommand(0xC0)
#define XLCDClear()     XLCDCommand(0x01)
#define XLCDReturnHome() XLCDCommand(0x02)

void XLCDDelay15ms(void);
void XLCDDelay4ms(void);
void XLCDDelay100us(void);
void XLCD_Delay500ns(void);
void XLCDDelay(void);

#endif
```

XLCD.C

```
/*********************************************************************
 *
 *                  External LCD access routines
 *
 *********************************************************************
 * FileName:        XLCD.c
 * Dependencies:    xlcd.h
 * Processor:       PIC18
 * Complier:        MCC18 v1.00.50 or higher
 *                  HITECH PICC-18 V8.10PL1 or higher
 * Company:         Microchip Technology, Inc.
 *
```

```
 * Software License Agreement
 * The software supplied herewith by Microchip Technology Incorporated
 * (the "Company") for its PICmicro® Microcontroller is intended and
 * supplied to you, the Company's customer, for use solely and
 * exclusively on Microchip PICmicro Microcontroller products. The
 * software is owned by the Company and/or its supplier, and is
 * protected under applicable copyright laws. All rights are reserved.
 * Any use in violation of the foregoing restrictions may subject the
 * user to criminal sanctions under applicable laws, as well as to
 * civil liability for the breach of the terms and conditions of this
 * license.
 *
 * THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
 * WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
 * TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
 * IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 *
 * HiTech PICC18 Compiler Options excluding device selection:
 *                  -FAKELOCAL -G -E -C
 *
 * Author               Date    Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Naveen Raj          6/9/03         Original        (Rev 1.0)
 ********************************************************************/
#include "xlcd.h"
char _vXLCDreg =0;          //Used as a flag to check if from XLCDInit()

/*********************************************************************
 * Function         : void XLCDInit(void)
 * PreCondition     : None
 * Input            : None
 * Output           : None
 * Side Effects     : None
 * Overview         : LCD is intialized
 * Note             : This function will work with all Hitachi HD447780
 *                    LCD controller.
 ********************************************************************/
void XLCDInit(void)
{
/*This par of the code is initialization by instruction*/
_vXLCDreg=1;

//PORT initialization
#ifdef  XLCD_8BIT                           //8-bit mode, use whole port
    XLCD_DATAPORT_TRIS  = 0x00;             //make DATAPORT output
    XLCD_DATAPORT = 0;
#endif

#ifdef  XLCD_4BIT                           //4bit mode
    #ifdef XLCD_UPPER                       //Upper 4-bits of the DATAPORT
output
    XLCD_DATAPORT_TRIS  &= 0x0f;
    XLCD_DATAPORT &= 0x0f;
    #else                                   //Lower 4-bits of the DATAPORT
output
    XLCD_DATAPORT_TRIS  &= 0xf0;
```

```
    XLCD_DATAPORT &= 0xf0;
    #endif
#endif
//end of data port initialization


//control port initialization
XLCD_RSPIN_TRIS =0;                         //make control ports output
XLCD_ENPIN_TRIS =0;
#ifndef XLCD_RW_GROUND
XLCD_RWPIN_TRIS =0;                         //if RW pin grounded
#endif

XLCD_RSPIN  =0;                             //clear control ports
XLCD_ENPIN  =0;
#ifndef XLCD_RW_GROUND
XLCD_RWPIN=0;                               //if RW pin grounded
#endif



//initialization by instruction

XLCDDelay15ms();
#ifdef XLCD_8BIT                        // 8-bit mode interface
    XLCD_DATAPORT   = 0b00110000;   // Function set cmd(8-bit interface)
#endif

#ifdef XLCD_4BIT
    #ifdef XLCD_UPPER                   // Upper nibble interface
    XLCD_DATAPORT   &= 0x0f;        // Clear upper port
    XLCD_DATAPORT   |= 0b00110000;
    #else                               // Lower nibble interface
    XLCD_DATAPORT       &= 0xf0;    // Clear lower port
    XLCD_DATAPORT   |= 0b00000011;  // Function set cmd(4-bit interface)
    #endif
#endif
    XLCD_ENPIN = 1;                 // Clock the cmd in
    XLCD_Delay500ns();
    XLCD_ENPIN = 0;
////////////////////////////////////////////////////////////////////////
    XLCDDelay4ms();
#ifdef XLCD_8BIT                        // 8-bit mode interface
    XLCD_DATAPORT   = 0b00110000;   // Function set cmd(8-bit interface)
#endif

#ifdef XLCD_4BIT
    #ifdef XLCD_UPPER                   // Upper nibble interface
    XLCD_DATAPORT   &= 0x0f;        // Clear upper port
    XLCD_DATAPORT   |= 0b00110000;
    #else                               // Lower nibble interface
    XLCD_DATAPORT       &= 0xf0;    // Clear lower port
    XLCD_DATAPORT   |= 0b00000011;  // Function set cmd(4-bit interface)
    #endif
#endif
    XLCD_ENPIN = 1;                 // Clock the cmd in
    XLCD_Delay500ns();
    XLCD_ENPIN = 0;
```

```
////////////////////////////////////////////////////////////////////////
 //       XLCDDelay100us();
    XLCDDelay4ms();
#ifdef XLCD_8BIT                    // 8-bit mode interface
    XLCD_DATAPORT   = 0b00110000;   // Function set cmd(8-bit interface)
#endif

#ifdef XLCD_4BIT
    #ifdef XLCD_UPPER               // Upper nibble interface
    XLCD_DATAPORT   &= 0x0f;        // Clear upper port
    XLCD_DATAPORT   |= 0b00110000;
    #else                           // Lower nibble interface
    XLCD_DATAPORT       &= 0xf0;    // Clear lower port
    XLCD_DATAPORT   |= 0b00000011;  // Function set cmd(4-bit interface)
    #endif
#endif

    XLCD_ENPIN = 1;                 // Clock the cmd in
    XLCD_Delay500ns();
    XLCD_ENPIN = 0;


//required only for 4 bit interface as per LCDdatasheet
#ifdef XLCD_4BIT
    XLCDDelay4ms();
    #ifdef XLCD_UPPER               // Upper nibble interface
    XLCD_DATAPORT   &= 0x0f;        // Clear upper port
    XLCD_DATAPORT   |= 0b00100000;
    #else                           // Lower nibble interface
    XLCD_DATAPORT       &= 0xf0;    // Clear lower port
    XLCD_DATAPORT   |= 0b00000010;  // Function set cmd(4-bit interface)
    #endif

    XLCD_ENPIN = 1;                 // Clock the cmd in
    XLCD_Delay500ns();
    XLCD_ENPIN = 0;
#endif
//---------------------------------------------------------------------
//function set command "0 0 1 DL N F X X"
//---------------------------------------------------------------------

#ifdef XLCD_8BIT                    // if 8bit

    #ifdef XLCD_1LINE
        #ifdef XLCD_FONT5x8
        XLCDCommand(0b00110000);    //if 1Line 5x8
        #else
        XLCDCommand(0b00110100);    //if 1Line 5x10
        #endif
    #endif

    #ifdef XLCD_2LINE
        #ifdef XLCD_FONT5x8
        XLCDCommand(0b00111000);    //if 2Line 5x8
        #else
        XLCDCommand(0b00111100);    //if 2Line 5x10
        #endif
    #endif
```

```
#endif

#ifdef XLCD_4BIT                    //if 4bit
    #ifdef XLCD_1LINE
        #ifdef XLCD_FONT5x8
        XLCDCommand(0b00100000);    //if 1Line 5x8
        #else
        XLCDCommand(0b00100100);    //if 1Line 5x10
        #endif
    #else
        #ifdef XLCD_FONT5x8
        XLCDCommand(0b00101000);    //if 2Line 5x8
        #else
        XLCDCommand(0b00101100);    //if 2Line 5x10
        #endif
    #endif
#endif
    XLCDCommand(0b00001000);        //display off
    XLCDCommand(0b00000001);        //display clear
////////////////////////////////////////////////////////////////////////////
/////
//Entry mode setting
////////////////////////////////////////////////////////////////////////////
////
//Entry mode command " 0 0 0 0 0 1 ID S "
//ID =0 no cursor increment during read and write
//ID =1 cursor increment during read and write
//S =0 no display during read and write
//S =1 display shift

#ifdef XLCD_CURSOR_INCREMENT
    #ifdef XLCD_DISPLAY_SHIFT
    XLCDCommand(0b00000111);    //if cursor inc and display shift
    #endif
    #ifdef XLCD_DISPLAY_NOSHIFT
    XLCDCommand(0b00000110);    //if cursor inc and no display shift
    #endif
#endif

#ifdef  XLCD_CURSOR_NOINCREMENT
    #ifdef XLCD_DISPLAY_SHIFT
    XLCDCommand(0b00000101);    //if no cursor increment, but with display
shift
    #endif
    #ifdef XLCD_DISPLAY_NOSHIFT
    XLCDCommand(0b00000100);    //if no cursor increment, and no display
shift
    #endif
#endif
////////////////////////////////////////////////////////////////////////////
///////
//Display on off ,Blink ,cursor command set
// ////////////////////////////////////////////////////////////////////////
//////
//"0 0 0 0 1 D C B "
//D=1 dislay on, C=1 cursor on, B=1 blink on

#ifdef  XLCD_DISPLAYON
```

```
    #ifdef XLCD_CURSORON
        #ifdef XLCD_BLINKON
        XLCDCommand(0b00001111);    //display on cursor on blink on
        #else
        XLCDCommand(0b00001110);    //display on cursor on blink off
        #endif
    #endif

    #ifdef XLCD_CURSOROFF
        #ifdef XLCD_BLINKON
        //XLCDCommand(0b00001001);    //display on cursor off blink on
        #else
        XLCDCommand(0b00001100);    // display on cursor off blink off
        #endif
    #endif
#endif


#ifdef  XLCD_DISPLAYOFF
    XLCDCommand(0b00001000);        //display off
#endif
_vXLCDreg=0;
// end of initialization
      return;
}

/*********************************************************************
 * Function         : void XLCDCommand(unsigned char cmd)
 * PreCondition     : None
 * Input            : cmd - Command to be set to LCD.
 * Output           : None
 * Side Effects     : None
 * Overview         : None
 * Note             : None
 ********************************************************************/
void XLCDCommand(unsigned char cmd)
{
if(_vXLCDreg==1)                            //if  called from XLCDinit
routine is always Blocking
{
    #ifdef  XLCD_DELAYMODE
    XLCDDelay();
    #endif
    #ifdef  XLCD_READBFMODE
    XLCDIsBusy();
    #endif
}


if(_vXLCDreg==0)                            //if not called from XLCDinit
routine
{                                           //if NON Block the user need to
call XLCDIsBusy
#ifdef  XLCD_BLOCK                          //and check the w reg status to
check if the
    #ifdef  XLCD_DELAYMODE                  //module is free
    XLCDDelay();
    #endif
```

```
    #ifdef  XLCD_READBFMODE
    XLCDIsBusy();
    #endif
#endif
}

XLCD_RSPIN=0;
XLCD_ENPIN=0;
#ifndef XLCD_RW_GROUND
XLCD_RWPIN=0;
#endif

#ifdef XLCD_8BIT
    XLCD_DATAPORT = cmd;                    // Write command to data port
    XLCD_ENPIN = 1;                         // Clock the cmd in
    XLCD_Delay500ns();
    XLCD_ENPIN = 0;
#endif

#ifdef XLCD_4BIT
    #ifdef XLCD_UPPER
        XLCD_DATAPORT &=0x0f;               //clear port
        XLCD_DATAPORT |= cmd&0xf0;          //write upper nibble to port
        XLCD_ENPIN = 1;                     // Clock the cmd in
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;

        XLCD_DATAPORT &= 0x0f;              //clear port
        XLCD_DATAPORT |= (cmd<<4)&0xf0;        //shift left 4 times
        XLCD_ENPIN = 1;
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;
    #endif

    #ifdef XLCD_LOWER
        XLCD_DATAPORT &=0xF0;               //clear port
        XLCD_DATAPORT |=((cmd>>4)&0x0f);
        XLCD_ENPIN = 1;                     // Clock the cmd in
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;

        XLCD_DATAPORT &= 0xF0;              //clear port
        XLCD_DATAPORT |= cmd&0x0f ;          //shift left 4 times
        XLCD_ENPIN = 1;
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;
    #endif

#endif


    return;
}
/*********************************************************************
 * Function         :XLCDPut()
 * PreCondition     :None
 * Input            :cmd - Command to be set to LCD.
 * Output           :None
```

```
 * Side Effects     :None
 * Overview         :None
 * Note             :None
 ********************************************************************/
void XLCDPut(char data)
{
#ifdef  XLCD_BLOCK
    #ifdef  XLCD_DELAYMODE
    XLCDDelay();
    #endif
    #ifdef  XLCD_READBFMODE
    XLCDIsBusy();
    #endif
#endif


#ifndef XLCD_RW_GROUND
XLCD_RWPIN=0;
#endif
XLCD_RSPIN=1;
XLCD_ENPIN=0;
#ifdef XLCD_8BIT
    XLCD_DATAPORT=data;
    XLCD_ENPIN = 1;
    XLCD_Delay500ns();
    XLCD_ENPIN = 0;
#endif
#ifdef XLCD_4BIT
    #ifdef XLCD_UPPER
        XLCD_DATAPORT &=0x0f;               //clear port
        XLCD_DATAPORT |= data&0xf0;         //write upper nibble to port
        XLCD_ENPIN = 1;                     // Clock the cmd in
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;

        XLCD_DATAPORT &= 0x0f;              //clear port
        XLCD_DATAPORT |= (data<<4)&0xf0;   //shift left 4 times
        XLCD_ENPIN = 1;
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;
    #endif
    #ifdef XLCD_LOWER
        XLCD_DATAPORT &=0xF0;               //clear port
        XLCD_DATAPORT |=((data>>4)&0x0f);
        XLCD_ENPIN = 1;                     // Clock the cmd in
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;

        XLCD_DATAPORT &= 0xF0;              //clear port
        XLCD_DATAPORT |= data&0x0f ;      //shift left 4 times
        XLCD_ENPIN = 1;
        XLCD_Delay500ns();
        XLCD_ENPIN = 0;
    #endif



#endif
```

```
   return;
}



#ifndef XLCD_RW_GROUND    //need not compile any read command if RWpin
grounded

/*********************************************************************
 * Function         :char XLCDIsBusy(void)
 * PreCondition     :None
 * Input            :None
 * Output           :non-zero if LCD controller is ready to accept new
 *                   data or commandzero otherwise.
 * Side Effects     :None
 * Overview         :None
 * Note             :None
 ********************************************************************/
char XLCDIsBusy(void)
{
XLCD_RSPIN=0;
XLCD_RWPIN=1;
XLCD_ENPIN=0;

#ifdef XLCD_8BIT
    XLCD_DATAPORT_TRIS=0xFF;                //make port input
    XLCD_DATAPORT=0;
    XLCD_ENPIN=1;
    XLCD_Delay500ns();

    if(_vXLCDreg==1)                        //will execute only if  called
from XLCDInit
    {
    while(XLCD_DATAPORT&0x80);
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS=0x00;                //make port output
    return;
    }

    #ifdef  XLCD_BLOCK
    if(_vXLCDreg==0)                        // will execute only if not
called from XLCDInit
    {
    while(XLCD_DATAPORT&0x80);
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS=0x00;                //make port input
    return;
    }
    #endif


    #ifdef  XLCD_NONBLOCK
    if(_vXLCDreg==0)                        //will execute only if not
called from XLCDInit
    {
        if(XLCD_DATAPORT&0x80)              //Read bit 7 (busy bit)
        {
```

```
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS=0x00;            //make port op
        return 1;                           //Return TRUE
        }
        else
        {
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS=0x00;            //make port op
        return 0;                           //Return FALSE
        }
    }
    #endif
#endif


#ifdef XLCD_4BIT

    #ifdef  XLCD_UPPER
    XLCD_DATAPORT_TRIS|=0xF0;               //make upper port input
    XLCD_DATAPORT   &=0x0F;
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    if(_vXLCDreg==1)                        // will execute only if  called
from XLCDInit
    {
        while(XLCD_DATAPORT&0x80);
        XLCD_ENPIN=0;
        XLCD_Delay500ns();
        XLCD_ENPIN=1;
        XLCD_Delay500ns();
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS&=0x0F;           //make upper port output
        return;
    }

    #ifdef  XLCD_BLOCK
    if(_vXLCDreg==0)                        // When not called from the
XLCDInit
    {
        while(XLCD_DATAPORT&0x80);
        XLCD_ENPIN=0;
        XLCD_Delay500ns();
        XLCD_ENPIN=1;
        XLCD_Delay500ns();
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS&=0x0F;           //make upper port output
        return;
    }
    #endif

    #ifdef  XLCD_NONBLOCK
    if(_vXLCDreg==0)                        // will execute only if not
called from XLCDInit
    {
    if(XLCD_DATAPORT&0x80)
        {
        XLCD_ENPIN=0;
        XLCD_Delay500ns();
```

```
        XLCD_ENPIN=1;
        XLCD_Delay500ns();
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS&=0x0F;           //make port output
        return 1;                           //Return TRUE
        }
    else
        {
        XLCD_ENPIN=0;
        XLCD_Delay500ns();                  // If high
        XLCD_ENPIN=1;
        XLCD_Delay500ns();
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS&=0x0F;           //make port input
        return 0;                           // Return FALSE
        }

    }
    #endif
    #endif


    #ifdef  XLCD_LOWER
    XLCD_DATAPORT_TRIS|=0x0F;               //make lower port input
    XLCD_DATAPORT   &=0xF0;
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    if(_vXLCDreg==1)                        // will execute only if  called
from XLCDInit
        {
        while(XLCD_DATAPORT&0x08);
        XLCD_ENPIN=0;
        XLCD_Delay500ns();
        XLCD_ENPIN=1;
        XLCD_Delay500ns();
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS&=0xF0;           //make port output
        return;
        }
    #ifdef XLCD_BLOCK
    if(_vXLCDreg==0)                        //will execute only if  called
from XLCDInit
        {
        while(XLCD_DATAPORT&0x08);
        XLCD_ENPIN=0;
        XLCD_Delay500ns();
        XLCD_ENPIN=1;
        XLCD_Delay500ns();
        XLCD_ENPIN=0;
        XLCD_DATAPORT_TRIS&=0xF0;           //make port output
        return 0;
        }
    #endif

    #ifdef XLCD_NONBLOCK
    if(_vXLCDreg==0)                        // will execute only if  called
from XLCDInit
        {
```

```
            if(XLCD_DATAPORT&0x08)
            {
            XLCD_ENPIN=0;
            XLCD_Delay500ns();
            XLCD_ENPIN=1;
            XLCD_Delay500ns();
            XLCD_ENPIN=0;
            XLCD_DATAPORT_TRIS=0x00;        //make port input
            return 1;                       // Return TRUE
            }
        else
            {
            XLCD_ENPIN=0;
            XLCD_Delay500ns();
            XLCD_ENPIN=1;
            XLCD_Delay500ns();
            XLCD_ENPIN=0;
            XLCD_DATAPORT_TRIS=0x00;        //make port input
            return 0;                       // Return FALSE
            }

        }
    #endif
    #endif
#endif

}

/*********************************************************************
 * Function         :unsigned char XLCDGetAddr(void)
 * PreCondition     :None
 * Input            :None
 * Output           :Current address byte from LCD
 * Side Effects     :None
 * Overview         :None
 * Note             :The address is read from the character generator
 *                   RAM or display RAM depending on current setup.
 ********************************************************************/

unsigned char XLCDGetAddr(void)

{
char addr =0;
#ifdef  XLCD_BLOCK
    #ifdef  XLCD_DELAYMODE
    XLCDDelay();
    #endif
    #ifdef  XLCD_READBFMODE
    XLCDIsBusy();
    #endif
#endif

XLCD_RSPIN=0;
XLCD_RWPIN=1;
XLCD_ENPIN=0;

#ifdef XLCD_8BIT
    XLCD_DATAPORT_TRIS=0xFF;                //make port input
```

```
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    addr=XLCD_DATAPORT;
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS=0x00;                //make port input
    return(addr&0x7F);
#endif
#ifdef XLCD_4BIT
    #ifdef  XLCD_UPPER
    XLCD_DATAPORT_TRIS|=0xF0;               //make upper port input
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    addr=XLCD_DATAPORT&0xF0;
    XLCD_ENPIN=0;;
    XLCD_Delay500ns();
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    addr|=(XLCD_DATAPORT>>4)&0x0F;
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS&=0x0F;               //make upper port output
    return(addr&0x7F);
    #endif

    #ifdef  XLCD_LOWER
    XLCD_DATAPORT_TRIS|=0x0F;               //make lower port input
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    addr=(XLCD_DATAPORT<<4)&0xF0;
    XLCD_ENPIN=0;
    XLCD_Delay500ns();
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    addr|=XLCD_DATAPORT&0x0F;
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS&=0xF0;               //make port output
    return(addr&0x7F);
    #endif

#endif

}


/*****************************************************************
 * Function         :char XLCDGet(void)
 * PreCondition     :None
 * Input            :None
 * Output           :Current data byte from LCD
 * Side Effects     :None
 * Overview         :None
 * Note             :The data is read from the character generator
 *                   RAM or display RAM depending on current setup.
 *****************************************************************/
char XLCDGet(void)
{
    char data=0;
#ifdef  XLCD_BLOCK
    #ifdef  XLCD_DELAYMODE
```

```
    XLCDDelay();
    #endif
    #ifdef  XLCD_READBFMODE
    XLCDIsBusy();
    #endif
#endif
    XLCD_RSPIN=1;
    XLCD_RWPIN=1;
    XLCD_ENPIN=0;

#ifdef  XLCD_8BIT
    XLCD_DATAPORT_TRIS=0xFF;
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    data=XLCD_DATAPORT;
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS=0x00;
    return(data);
#endif
#ifdef  XLCD_4BIT
    #ifdef  XLCD_UPPER
    XLCD_DATAPORT_TRIS |=0xf0;              //make upper input
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    data = XLCD_DATAPORT&0xf0;              //Read the upper nibble of data
    XLCD_ENPIN=0;
    XLCD_Delay500ns();
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    data |= ((XLCD_DATAPORT>>4)&0x0f);      //Read the upper nibble of data
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS &=0x0f;              //make output
    return(data);
    #endif

    #ifdef  XLCD_LOWER
    XLCD_DATAPORT_TRIS |=0x0F;              //make  input
    XLCD_ENPIN=1;
    XLCD_Delay500ns();
    data = (XLCD_DATAPORT<<4)&0xf0;         //Read the upper nibble of data
    XLCD_ENPIN=0;
    XLCD_Delay500ns();
    XLCD_ENPIN=1;
    XLCD_Delay500ns();;
    data |= XLCD_DATAPORT&0x0f;             //Read the upper nibble of data
    XLCD_ENPIN=0;
    XLCD_DATAPORT_TRIS &=0xf0;              //make output
    return(data);
    #endif
#endif


}

#endif          //end of #ifndef XLCD_RW_GROUND(all read commands)
```

```
/*********************************************************************
 * Function         :XLCDPutRomString(rom char *string)
 * PreCondition     :None
 * Input            :None
 * Output           :Displays string in Program memory
 * Side Effects     :None
 * Overview         :None
 * Note             :is lways blocking till the string is written fully
 ********************************************************************/

void XLCDPutRomString(rom char *string)
{
     while(*string)                         // Write data to LCD up to null
    {
        #ifdef  XLCD_NONBLOCK
        while(XLCDIsBusy());
        #endif
        XLCDPut(*string);                   // Write character to LCD
        string++;                           // Increment buffer
    }
    return;
}
/*********************************************************************
 * Function         :XLCDPutRomString(rom char *string)
 * PreCondition     :None
 * Input            :None
 * Output           :Displays string in Program memory
 * Side Effects     :None
 * Overview         :None
 * Note             :is lways blocking till the string is written fully
 ********************************************************************/
void XLCDPutRamString(char *string)
{
    while(*string)                          // Write data to LCD up to null
    {
        #ifdef  XLCD_NONBLOCK
        while(XLCDIsBusy());
        #endif

        XLCDPut(*string);                   // Write character to LCD
        string++;                           // Increment buffer
    }
    return;
}
```

PORTON.C

```
#include <p18f4550.h>
#include <delays.h>
#include <stdio.h>
#include <usart.h>
#include <adc.h>
#include "xlcd.h"
#include "lcd.h"
#include "assert.h"
#include "flags.h"
#include "AESdef.h"
#include "AES.h"
```

```
#define LED LATBbits.LATB5
#define TECLA PORTBbits.RB2

#define PEDIDO_DATOS 0xFF
#define ENVIO_DATOS 0x55
#define ENVIO_CLAVE 0x00

void __init (void);
void enviar(unsigned char flag, unsigned char *addr, unsigned char n);
void recibir(unsigned char *addr, unsigned char n);
void generarAleatorio(unsigned char *addr);
void Ram2Eeprom(unsigned char *addrRAM, unsigned char addrEEPROM, unsigned
char n);
void Eeprom2Ram(unsigned char addrEEPROM, unsigned char *addrRAM, unsigned
char n);
void procesarUSART(void);
char coincide(void);
void procesarTecla(void);
unsigned char EERead(unsigned char addr);
void EEWrite(unsigned char addr, unsigned char c);

unsigned char datos[32]; // datos sin codificar +  datos codificados.
unsigned char key[16];

void main(){
      char i;
      ClearWDT();
      LCDClear();
      Eeprom2Ram(0,key,16);
      ClearWDT();
      for(i=0;i<16;i++){
            printf("%2.2X",key[i]);
      }
      ClearWDT();
      while(1){
            ClearWDT();
      }
}

void __init(void){
      ClearWDT();

      stdout=_H_USER; // Para que printf escriba al LCD.

      XLCDInit(); // Inicia el LCD.
      LCDClear(); // Lo limpia.

      OpenUSART(USART_TX_INT_OFF&USART_RX_INT_ON&USART_ASYNCH_MODE&
      USART_EIGHT_BIT&USART_CONT_RX&USART_BRGH_LOW&USART_ADDEN_OFF,60);
      BAUDCONbits.TXCKP=1; // Invierte la polaridad de TX.
      IPR1bits.RCIP=1; // De alta prioridad.

      OpenADC(ADC_FOSC_64 & ADC_RIGHT_JUST & ADC_4_TAD,
      ADC_CH0 & ADC_INT_OFF & ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS,
      14); // Para generar una clave aleatoria.

      LED=0; // Apago el LED.
      TRISBbits.TRISB5 = 0; // Y lo habilito a su pin como salida.
```

```
	TRISBbits.TRISB2=1; // La tecla es entrada.
	INTCON3bits.INT2IP=1; // Alta prioridad.
	INTCON3bits.INT2IE=1; // Habilito su interrupcion.

	RCONbits.IPEN=1; // Habilito la prioridad de interrupciones.
	INTCONbits.GIEH=1; // Habilito las interrupciones de alta prioridad.
	//INTCONbits.GIEL=1; // Habilito int de baja prioridad.

	ClearWDT();
}

// Envia datos.
// flag: El tipo de transmisión (PEDIDO_DATOS, ENVIO_DATOS o ENVIO_CLAVE
0x00)
// addr: Puntero a los datos.
// n: cuantos bytes enviar de datos.
void enviar(unsigned char flag, unsigned char *addr, unsigned char n){
	unsigned char i;
	while(BusyUSART()); // Me aseguro de no estar transmitiendo todavía.
	Delay10TCYx(60); // Me aseguro de que el otro ya haya terminado de
transmitir.
	ClearWDT();
	putcUSART(flag);
	ReadUSART();
	for(i=0;i<n;i++){
		while(BusyUSART());
		ClearWDT();
		putcUSART(*addr++);
		ReadUSART();
	}
	while(!DataRdyUSART());
	ReadUSART();
}

// Recibe n datos y los guarda en addr.
void recibir(unsigned char *addr, unsigned char n){
	unsigned char i;
	while(!DataRdyUSART());
	ClearWDT();
	for(i=0;i<n;i++){
		while(!DataRdyUSART());
		*addr++=ReadUSART();
		ClearWDT();
	}
}

// Genera 16 bytes aleatorios y los escribe a partir de addr
void generarAleatorio(unsigned char *addr){
	unsigned char i,j,byte;
	ClearWDT();
	for(i=0;i<16;i++){
		byte=0;
		for(j=0;j<8;j++){
			ConvertADC();
			while(BusyADC());
			byte |= ((ADRESL & 1)<<j);
		}
```

```
            *addr=byte;
            addr++;
      }
}

// Guarda n bytes de la RAM a la memoria EEPROM.
void Ram2Eeprom(unsigned char *addrRAM, unsigned char addrEEPROM, unsigned
char n){
      unsigned char i;
      for(i=0;i<n;i++)
            EEWrite(addrEEPROM++,*addrRAM++);
}

// Levanta n bytes de la EEPROM a la RAM.
void Eeprom2Ram(unsigned char addrEEPROM, unsigned char *addrRAM, unsigned
char n){
      unsigned char i;
      for(i=0;i<n;i++){
            *addrRAM=EERead(addrEEPROM++);
            addrRAM++;
      }
}

// Llamada desde interrupcion.
void procesarUSART(void){
      while(!DataRdyUSART());
      ClearWDT();
      switch(ReadUSART()){
            case PEDIDO_DATOS:
                  generarAleatorio(datos);
                  enviar(ENVIO_DATOS,datos,16);
                  break;
            case ENVIO_DATOS:
                  recibir(&datos[16],16);
                  Eeprom2Ram(0,key,16);
                  AESEncode(datos,key);
                  if(coincide())
                        LED=~LED;
                  break;
            default:
                  return;
      }
}

char coincide(void){
      char i;
      for(i=0;i<16;i++){
            if(datos[i]!=datos[i+16])
                  return 0;
      }
      return 1;
}

void procesarTecla(void){ // Llamada desde interrupcion.
      char i,largo=1;
      unsigned long j;
      ClearWDT();
      Delay10KTCYx(12); // 10 ms hasta que paren los rebotes.
```

```
      ClearWDT();
      if(!TECLA)
            return;
      for(j=0;j<222222;j++){ // Esperar unos 500 ms para ver si es clic
largo.
            if(!TECLA){
                  largo=0;
                  break;
            }
            ClearWDT();
      }
      if(largo){
            ClearWDT();
            generarAleatorio(key);
            Ram2Eeprom(key,0,16);
            LCDClear();
            for(i=0;i<16;i++)
                  printf("%2.2X",key[i]);
      }
      else{
            ClearWDT();
            Eeprom2Ram(0,key,16);
            enviar(ENVIO_CLAVE,key,16);
      }
}

unsigned char EERead(unsigned char addr){
      EEADR = addr;
      EECON1bits.EEPGD = 0;
      EECON1bits.CFGS = 0;
      EECON1bits.RD = 1;
      return EEDATA;
}

void EEWrite(unsigned char addr, unsigned char c){
      unsigned char gie;
      EEADR=addr;
      EEDATA=c;
      EECON1bits.EEPGD = 0;
      EECON1bits.CFGS = 0;
      EECON1bits.WREN = 1;
      gie=INTCONbits.GIE;
      INTCONbits.GIE = 0;
      EECON2 = 0x55;
      EECON2 = 0xAA;
      EECON1bits.WR = 1;
      INTCONbits.GIE = gie;
      while(!PIR2bits.EEIF);
      PIR2bits.EEIF = 0;
      EECON1bits.WREN = 0;
}
```

FLAGS.H

```
#ifndef __FLAGS_H
#define __FLAGS_H

#pragma config PLLDIV = 1
#pragma config CPUDIV = OSC1_PLL2
```

```
#pragma config USBDIV = 1
#pragma config FOSC = HSPLL_HS
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config PWRT = OFF
#pragma config BOR = OFF
#pragma config BORV = 3
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config PBADEN = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = ON
#pragma config LVP = OFF
#pragma config ICPRT = OFF
#pragma config XINST = ON
#ifdef __DEBUG
#pragma config DEBUG = ON
#pragma config WDT = OFF
#define ClearWDT() ((void)0)
#else
#pragma config DEBUG = OFF
#pragma config WDT = ON, WDTPS = 128
#define ClearWDT() ClrWdt()
#endif

#endif
```

ARCHIVOS ÚNICOS DE CAMIÓN.

Los siguientes archivos fuente pertenecen únicamente al proyecto “camión”. O sea, al mando a distancia.

CAMION.C

```
#include <p18f4550.h>
#include <delays.h>
#include <stdio.h>
#include <usart.h>
#include <adc.h>
#include <timers.h>
#include "flags.h"
#include "AESdef.h"
#include "AES.h"

#define TECLA PORTBbits.RB2

#define PEDIDO_DATOS 0xFF
#define ENVIO_DATOS 0x55
#define ENVIO_CLAVE 0x00

void __init (void);
void enviar(unsigned char flag, unsigned char *addr, unsigned char n);
void recibir(unsigned char *addr, unsigned char n);
void Ram2Eeprom(unsigned char *addrRAM, unsigned char addrEEPROM, unsigned
char n);
void Eeprom2Ram(unsigned char addrEEPROM, unsigned char *addrRAM, unsigned
```

```
char n);
void procesarUSART(void);
void procesarTecla(void);
unsigned char EERead(unsigned char addr);
void EEWrite(unsigned char addr, unsigned char c);

unsigned char datos[16];
unsigned char key[16];

void main(){
     while(1){
          ClearWDT();
     }
}

void __init(void){
     ClearWDT();

     OpenUSART(USART_TX_INT_OFF&USART_RX_INT_ON&USART_ASYNCH_MODE&
     USART_EIGHT_BIT&USART_CONT_RX&USART_BRGH_LOW&USART_ADDEN_OFF,60);
     BAUDCONbits.TXCKP=1; // Invierte la polaridad de TX.
     IPR1bits.RCIP=1; // De alta prioridad.

     TRISBbits.TRISB2=1; // La tecla es entrada.
     INTCON3bits.INT2IP=1; // Alta prioridad.
     INTCON3bits.INT2IE=1; // Habilito su interrupcion.

     RCONbits.IPEN=1; // Habilito la prioridad de interrupciones.
     INTCONbits.GIEH=1; // Habilito las interrupciones de alta prioridad.
     //INTCONbits.GIEL=1; // Habilito int de baja prioridad.

     ClearWDT();
}

// Envia datos.
// flag: El tipo de transmisión (PEDIDO_DATOS, ENVIO_DATOS o ENVIO_CLAVE
0x00)
// addr: Puntero a los datos.
// n: cuantos bytes enviar de datos.
void enviar(unsigned char flag, unsigned char *addr, unsigned char n){
     unsigned char i;
     while(BusyUSART()); // Me aseguro de no estar transmitiendo todavía.
     Delay10TCYx(60); // Me aseguro de que el otro ya haya terminado de
transmitir.
     ClearWDT();
     putcUSART(flag);
     ReadUSART();
     for(i=0;i<n;i++){
          while(BusyUSART());
          ClearWDT();
          putcUSART(*addr++);
          ReadUSART();
     }
     while(!DataRdyUSART());
     ReadUSART();
}

// Recibe n datos y los guarda en addr.
```

```
void recibir(unsigned char *addr, unsigned char n){
      unsigned char i;
      while(!DataRdyUSART());
      ClearWDT();
      for(i=0;i<n;i++){
            while(!DataRdyUSART());
            *addr++=ReadUSART();
            ClearWDT();
      }
}

// Guarda n bytes de la RAM a la memoria EEPROM.
void Ram2Eeprom(unsigned char *addrRAM, unsigned char addrEEPROM, unsigned
char n){
      unsigned char i;
      for(i=0;i<n;i++)
            EEWrite(addrEEPROM++,*addrRAM++);
}

// Levanta n bytes de la EEPROM a la RAM.
void Eeprom2Ram(unsigned char addrEEPROM, unsigned char *addrRAM, unsigned
char n){
      unsigned char i;
      for(i=0;i<n;i++){
            *addrRAM=EERead(addrEEPROM++);
            addrRAM++;
      }
}

// Llamada desde interrupcion.
void procesarUSART(void){
      while(!DataRdyUSART());
      ClearWDT();
      switch(ReadUSART()){
            case ENVIO_CLAVE:
                  recibir(key,16);
                  Ram2Eeprom(key,0,16);
                  break;
            case ENVIO_DATOS:
                  recibir(datos,16);
                  Eeprom2Ram(0,key,16);
                  AESEncode(datos,key);
                  enviar(ENVIO_DATOS,datos,16);
                  break;
            default:
                  return;
      }
}

void procesarTecla(void){ // Llamada desde interrupcion.
      char i;
      ClearWDT();
      Delay10KTCYx(12); // 10 ms hasta que paren los rebotes.
      ClearWDT();
      if(!TECLA)
            return;
      while(TECLA)
            ClearWDT();
```

```
      enviar(PEDIDO_DATOS,0,0);
      ClearWDT();
}

unsigned char EERead(unsigned char addr){
      EEADR = addr;
      EECON1bits.EEPGD = 0;
      EECON1bits.CFGS = 0;
      EECON1bits.RD = 1;
      return EEDATA;
}

void EEWrite(unsigned char addr, unsigned char c){
      unsigned char gie;
      EEADR=addr;
      EEDATA=c;
      EECON1bits.EEPGD = 0;
      EECON1bits.CFGS = 0;
      EECON1bits.WREN = 1;
      gie=INTCONbits.GIE;
      INTCONbits.GIE = 0;
      EECON2 = 0x55;
      EECON2 = 0xAA;
      EECON1bits.WR = 1;
      INTCONbits.GIE = gie;
      while(!PIR2bits.EEIF);
      PIR2bits.EEIF = 0;
      EECON1bits.WREN = 0;
}
```

FLAGS.H

```
#ifndef __FLAGS_H
#define __FLAGS_H

#pragma config PLLDIV = 5
#pragma config CPUDIV = OSC1_PLL2
#pragma config USBDIV = 1
#pragma config FOSC = HSPLL_HS
#pragma config FCMEN = OFF
#pragma config IESO = OFF
#pragma config PWRT = OFF
#pragma config BOR = OFF
#pragma config BORV = 3
#pragma config VREGEN = OFF
#pragma config MCLRE = ON
#pragma config LPT1OSC = OFF
#pragma config PBADEN = OFF
#pragma config CCP2MX = ON
#pragma config STVREN = ON
#pragma config LVP = OFF
#pragma config ICPRT = OFF
#pragma config XINST = ON
#ifdef __DEBUG
#pragma config DEBUG = ON
#pragma config WDT = OFF
#define ClearWDT() ((void)0)
#else
#pragma config DEBUG = OFF
```

```
#pragma config WDT = ON, WDTPS = 128
#define ClearWDT() ClrWdt()
#endif

#endif
```

CONCLUSIÓN.

Al ser nuestra primera experiencia con transmisión/recepción en rf tuvimos muchas dificultades.

Primero hicimos la comunicación mediante cables para que, una vez que estuviera funcionando, empezar a probar con los módulos de rf. Cuando probamos los módulos no funcionaba, no lograbamos establecer la comunicación. Probamos con circuitos encoders y decoders como sugería el fabricante de los módulos pero éstos eran muy lentos, podíamos transmitir a unos 10 bps.

Luego empezamos a probar con el osciloscopio como transmitía el TWS y como recibía el RWS. Notamos que si dejábamos en alto el Data In del transmisor el receptor primero lo recibía bien pero después bajaba el nivel y ya salía simplemente un cero. Supusimos que era porque la modulación es ASK, que detecta variaciones en la amplitud, y al no detectar variaciones creía que venía simplemente la portadora (con más potencia) y entendía que venía un “cero”.

Así que invertimos la polaridad del TX del pic, para que en estado “idle” estuviera bajo. Ahí por fin funcionó. Transmitía bien. Buscamos un límite de kbps al que la onda no sufriera deformaciones y adoptamos 3.3 Kbps.

Pero el PIC no recibía los datos. Se perdían bytes. Resultó que la tensión de la salida del receptor no era suficiente para la entrada tipo trigger schmitt del RX del PIC. Así que pusimos una compuerta XOR CMOS y la configuramos como inversora. Por fin se recibían los datos.

Todavía no era todo, quedaba por solucionar el siguiente problema (que por lo menos ya habíamos anticipado): Al tener cada dispositivo tanto un transmisor como receptor, al transmitir recibe al mismo tiempo. Había que manejar esa situación. Lo hicimos por software. En cada transmisión, después de enviar cada byte, leemos uno también. Y al terminar de enviar el último esperamos a que llegue un byte y lo leemos. Ese es el último byte.

Aprendimos que estos módulos son muy prácticos porque sirven para comunicarse como lo hicimos en este trabajo por ejemplo, pero también con circuitos más sencillos, sin ningún microcontrolador. Simplemente utilizando un un encoder y un decoder (dip 18) en conjunto con el TWS y el RWS se puede crear muy facilmente un control remoto de 4 canales. Y cada control puede controlar 256 dispositivos diferentes (4 canales cada uno) utilizando unos jumpers que seleccionen el address del dispositivo.