#define TRUE 1

**#define MAX 256**

extern int contadorEjes; //Contador de ejes general

unsigned int masDeDosEjes; //Contador para autos con mas de dos ejes

extern int velocidad;

unsigned int nAuto;

**typedef struct{**

unsigned int hora;

unsigned int minutos;

unsigned int segundos;

} time;

**unsigned int time;**

**typedef struct{**

struct time timeStamp ;

unsigned int velocidad;

unsigned int cantEjes;

} dato;

**dato dataLogger[MAX];**

int counterTog = 0;

int terminoRecepcion;

unsigned int mensajeRecepcion[MAX];

unsigned int mensajeTransmicion[MAX];

int msEnviar;

**void UpdateClock (void)**

{

counterTog = 0;

/\* Actualizar Hora \*/

seconds=seconds+1;

if (seconds > 59)

{

seconds=0;

minutes=minutes+1;

}

if (minutes > 59)

{

minutes=0;

hours=hours+1;

}

if (hours == 24)

{

hours=0;

}

}

**void llenarArreglo()**

{

unsigned int i=0;

while(i<2){

i++;

dataLogger[nAuto].velocidad= 50;

dataLogger[nAuto].timeStamp.hora=21;

dataLogger[nAuto].timeStamp.minutos=30;

dataLogger[nAuto].timeStamp.segundos=10\*i;

dataLogger[nAuto].cantEjes= 1+i;

nAuto++; //dos vehiculos

masDeDosEjes = 1; //uno tiene mas de dos ejes

}

}

**void agregarDatos(){**

dataLogger[nAuto].velocidad= velocidad;

dataLogger[nAuto].timeStamp.hora=hours;

dataLogger[nAuto].timeStamp.minutos=minutes;

dataLogger[nAuto].timeStamp.segundos=seconds;

dataLogger[nAuto].cantEjes= contadorEjes;

nAuto++;

if (contadorEjes > 2){

masDeDosEjes++;

}

}

**/\*---------------------------------------------------------------------**

**Function Name: calcularChecksum**

**Description: calcula el checksum**

**Precondiciones: cant = C; mensaje[0..C]**

**Poscondiciones: checksum = acumulador = A; return A;**

**-----------------------------------------------------------------------\*/**

**unsigned int calcularChecksum(unsigned int cant, unsigned int mensaje[]){**

unsigned int acumulador =0;

unsigned int var ;

unsigned int i= 0;

while (i < (cant-3)){

var = mensaje[i];//Depositamos lo que esta en la posicion[0,2,4..n] del arreglo

var = var <<8; //Dezplazamos hacia izquierda; ejemplo: 0x00FE quedaria 0xFE00

var = var + mensaje[i+1];//sumamos; ejemplo: lo posicionado en [0]+[1] = 0xFE00 + 0x0080 = 0xFE08

i+=2;

acumulador+= var;//sumamos y guardamos en acumulador; ejemplo: 0x0000 + 0xFE08 = 0xFE08

}

if(cant%2 == 1){//Si el mensaje tiene 9 argumentos va a entrar

var = mensaje[6];

var = var << 8;

acumulador += var;

}

return acumulador;

}

**/\*---------------------------------------------------------------------**

**Function Name: verificarMensaje**

**Description: verifica el mensaje recibido**

**Precondiciones: --**

**Poscondiciones: msEnviar = MS; return MS; MS puede ser(41,42,43,44,45,46 o 47 si hay algun tipo de error)**

**-----------------------------------------------------------------------\*/**

**void verificarMensaje(){**

unsigned int check;

unsigned int valor;

check = calcularChecksum(mensajeRecepcion[1],mensajeRecepcion);

if (mensajeRecepcion[1] == 9){

valor = mensajeRecepcion[7];//Parte alta del checksum recibido

valor = valor << 8;//Desplaza 8 hacia izquierda; ejemplo:0x0081 quedaria 0x8100

valor = valor + mensajeRecepcion[8]; //sumamos parte alta mas parte baja; ejemplo: 0x8000 + 0x004B = 0x084B

}

else{

valor = mensajeRecepcion[6];//Parte alta del checksum recibido

valor = valor << 8;//Desplaza 8 hacia izquierda; ejemplo:0x0081 quedaria 0x8100

valor = valor + mensajeRecepcion[7]; //sumamos parte alta mas parte baja; ejemplo: 0x8000 + 0x004B = 0x084B

}

if (check == valor){

msEnviar = mensajeRecepcion[5]; //Si es correcto enviamos el argumento

}

else{

msEnviar = 0x0047;//si no enviamos NACK

}

}

**void armarMensajeD() {**

unsigned int bccl,bcch= 0;

unsigned int checksum;

unsigned int i = 0;

unsigned int j = 0;

mensajeTransmicion[0] = mensajeRecepcion[0];//SOF; 0x00FE

mensajeTransmicion[2] = mensajeRecepcion[3];//Dts

mensajeTransmicion[3] = mensajeRecepcion[2];//Src

mensajeTransmicion[4]= 0x0080;//Sec

mensajeTransmicion[5]= mensajeRecepcion[5];//Argumentos

while(i<nAuto){

if(mensajeRecepcion[6] == dataLogger[i].timeStamp.hora){

mensajeTransmicion[6+j] = dataLogger[i].timeStamp.hora;

mensajeTransmicion[7+j] = dataLogger[i].timeStamp.minutos;

mensajeTransmicion[8+j] = dataLogger[i].timeStamp.segundos;

mensajeTransmicion[9+j] = dataLogger[i].velocidad;

mensajeTransmicion[10+j] = dataLogger[i].cantEjes;

j = j + 5;

}

i++;

}

mensajeTransmicion[1] = 8 + j;//QTY

checksum = calcularChecksum(mensajeTransmicion[1],mensajeTransmicion);

bcch = checksum >>8;

bccl = checksum << 8;

bccl = bccl >> 8;

mensajeTransmicion[mensajeTransmicion[1]-2] = bcch;//BCCH

mensajeTransmicion[mensajeTransmicion[1]-1] = bccl;//BCCL

}

**/\*---------------------------------------------------------------------**

**Function Name: armarMensaje**

**Description: arma el mensaje a transmitir**

**Precondiciones: cant = C; msEnviar = MS;**

**Poscondiciones: mensajeTransmicion = MT; MT[0..C]**

**-----------------------------------------------------------------------\*/**

**void armarMensaje( unsigned int cant , unsigned int msEnviar) {**

unsigned int bccl,bcch= 0;

unsigned int checksum;

mensajeTransmicion[0] = mensajeRecepcion[0];//SOF; 0x0FE

mensajeTransmicion[1] = cant;//QTY; 8 o 9

mensajeTransmicion[2] = mensajeRecepcion[3];//Dts

mensajeTransmicion[3] = mensajeRecepcion[2];//Src

mensajeTransmicion[4]= 0x0080;//Sec

if(cant == 9){

mensajeTransmicion[5]= mensajeRecepcion[5];//Argumentos

mensajeTransmicion[6] = msEnviar;//Datos

checksum = calcularChecksum(9,mensajeTransmicion);

bcch = checksum >>8;

bccl = checksum << 8;

bccl = bccl >> 8;

mensajeTransmicion[7] = bcch;//BCCH

mensajeTransmicion[8] = bccl;//BCCL

}

else {

mensajeTransmicion[5]= msEnviar;//Argumentos

checksum = calcularChecksum(8,mensajeTransmicion);

bccl = checksum << 8;

bccl = bccl >> 8;

checksum = checksum >>8;

bcch = checksum ;

mensajeTransmicion[6] = bcch;//BCCH

mensajeTransmicion[7] = bccl;//BCCL

}

}

**/\*---------------------------------------------------------------------**

**Function Name: encenderCamara**

**Description: enciende camara por un milisegundo y la apaga**

**Precondiciones: --**

**Poscondiciones: --**

**-----------------------------------------------------------------------\*/**

**void encenderCamara(){**

unsigned int k=0;

PORTAbits.RA3 = 1;

while (k<5000){

k++; //tarda un milisegundo

}

PORTAbits.RA3 = 0;

}

**/\*---------------------------------------------------------------------**

**Function Name: resetear**

**Description: Resetea la cantidad de vehículos a 0 y borrar todos los registros.**

**Precondiciones: --**

**Poscondiciones: --**

**-----------------------------------------------------------------------\*/**

**void resetear(){**

unsigned int i = 0;

while(i < nAuto){

dataLogger[i].velocidad= 0;

dataLogger[i].timeStamp.hora= 0;

dataLogger[i].timeStamp.minutos = 0;

dataLogger[i].timeStamp.segundos = 0;

dataLogger[i].cantEjes = 0;

i++;

}

masDeDosEjes = 0;

nAuto = 0;

}

**/\*---------------------------------------------------------------------**

**Function Name: procesarMensaje**

**Description: procesa el mensaje y segun el caso ejecuta una funcion**

**Precondiciones: msEnviar = MS; MS (41..47)**

**Poscondiciones: --**

**-----------------------------------------------------------------------\*/**

**void procesarMensaje(){**

switch(msEnviar) {

case 'A':

armarMensaje(9,nAuto);//Consultar cantidad de vehículos hasta el momento.

IFS1bits.U2TXIF = 1; // Interrupt request has occurred

break;

case 'B':

resetear();//Resetear la cantidad de vehículos a 0 y borrar todos los registros.

armarMensaje(8,46);//ACK

IFS1bits.U2TXIF = 1; // Interrupt request has occurred

break;

case 'C':

armarMensaje(9,masDeDosEjes);//Consultar cantidad de vehículos con más de dos ejes.

IFS1bits.U2TXIF = 1; // Interrupt request has occurred

break;

case 'D':

armarMensajeD();//Consulta detallada de vehículos que pasaron en una determinada hora.

IFS1bits.U2TXIF = 1; // Interrupt request has occurred

break;

case 'E':

encenderCamara();//Accionar la cámara fotográfica.

armarMensaje(8,46);//ACK

IFS1bits.U2TXIF = 1; // Interrupt request has occurred

break;

case 'F':

armarMensaje(8,46);//ACK

IFS1bits.U2TXIF = 1; // Interrupt request has occurred

break;

case 'G':

armarMensaje(8,msEnviar);//NACK

IFS1bits.U2TXIF = 1; // Interrupt request has occurred

break;

}

}

**int main (void)**

{

config();

llenarArreglo();

/\* Loop infinito \*/

while (TRUE)

{

if (terminoRecepcion){

terminoRecepcion = 0;

verificarMensaje();

procesarMensaje();

}

if(counterTog > 3)

UpdateClock(); //Actualizar hora del dispositivo

}

}

**CONFIG.C**

#include "p33FJ256GP710.h"

#include "common.h"

#include "config.h"

**void config( void )**

{

//Variables para clock del sistema

hours, minutes, seconds = 0;

//Inicialización de PORTS I/O

/\* (RA0-RA7) como salidas - LEDs aa\*/

TRISA = 0xFF00;

/\* Initializar UART2\*/

InitUART2();

/\* Inicializar Timers necesarios \*/

Init\_Timer4();

Init\_Timer6();

/\*Todos los pines de todos los puertos como digitales\*/

AD1PCFGH = 0XFFFF;

AD1PCFGL = 0XFFFF;

AD2PCFGL = 0XFFFF;

IFS1bits.CNIF = 0; // Reset change notice interrupt flag

IPC4bits.CNIP = 6; //Set Change Notice interrupt priority to 6

CNEN1bits.CN15IE = 1; // pulsador para RD6

CNEN2bits.CN19IE = 1; // pulsador para RD13

CNEN2bits.CN16IE = 1; // pulsador para RD7

IEC1bits.CNIE = 1; //Enable CN interrupts

TRISD = 0xFFFF; // PORTD como pulsadores

TRISA = 0x0000; // PORTA como salida

}

**ISR\_TIMER46.C**

#define ValPR4 39062

#define ValPR6 5000 //1 ms

volatile unsigned char hours;

volatile unsigned char minutes;

volatile unsigned char seconds;

extern int counterTog;

//#define MAX 240

unsigned int contadorEjes = 0; //Contador de ejes general

//unsigned int masDeDosEjes; //Contador para autos con mas de dos ejes

unsigned int velocidad;

//unsigned int nAuto;

unsigned int quantum;

void prenderTimmer(){

TMR6 = 0;

quantum = 1;

T6CONbits.TON=1;

}

void detenerTimmer(){

T6CONbits.TON=0;

}

/\*---------------------------------------------------------------------

Function Name: \_CNInterrupt

Description: CN Interrupt Handler

-----------------------------------------------------------------------\*/

**void \_\_attribute\_\_((interrupt, auto\_psv)) \_CNInterrupt( void ) {**

IFS1bits.CNIF=0;

if (( PORTDbits.RD13)&& (PORTDbits.RD6) && (!PORTDbits.RD7)&&(contadorEjes < 1)){

prenderTimmer();

}

if((PORTDbits.RD13)&& (!PORTDbits.RD6)&&(PORTDbits.RD7)&&(contadorEjes == 1)){

detenerTimmer();

velocidad = 36 \* (30/quantum);

if (quantum < 18){

unsigned int k = 0;

PORTAbits.RA0 = 1;

while (k<5000){

k++; //tarda un milisegundo

}

PORTAbits.RA0 = 0;

}

}

if((PORTDbits.RD13)&&(PORTDbits.RD6)&&(!PORTDbits.RD7)){

contadorEjes++;

}

if((!PORTDbits.RD13)&&(contadorEjes > 0)){

agregarDatos();

contadorEjes = 0;//listo para recibir a un nuevo vehiculo

}

}

**/\*---------------------------------------------------------------------**

**Function Name: \_T4Interrupt**

**Description: Timer4 Interrupt Handler**

**Comments: 62500**

**-----------------------------------------------------------------------\*/**

**void \_\_attribute\_\_((interrupt, auto\_psv)) \_T4Interrupt( void )**

{

/\* reset Timer 4 interrupt flag \*/

IFS1bits.T4IF = 0;

counterTog ++;

}

**/\*---------------------------------------------------------------------**

**Function Name: Init\_Timer4**

**Description: Initialize Timer4**

**-----------------------------------------------------------------------\*/**

**void Init\_Timer4( void )**

{

/\* ensure Timer 4 is in reset state \*/

T4CON = 0;

T4CONbits.TCKPS = 3; //Prescaler 256

/\* reset Timer 4 interrupt flag \*/

IFS1bits.T4IF = 0;

/\* set Timer interrupt priority level \*/

IPC6bits.T4IP = 5;

/\* enable Timer interrupt \*/

IEC1bits.T4IE = 1;

/\* set Timer period register \*/

PR4 = ValPR4;

T4CONbits.TON = 1; //habilito Timer

}

**/\*---------------------------------------------------------------------**

**Function Name: \_T6Interrupt**

**Description: Timer6 Interrupt Handler**

**-----------------------------------------------------------------------\*/**

**void \_\_attribute\_\_((interrupt, auto\_psv)) \_T6Interrupt( void )**

{

/\* reset Timer 6 interrupt flag \*/

IFS2bits.T6IF = 0;

quantum++;

}

**/\*---------------------------------------------------------------------**

**Function Name: Init\_Timer6**

**Description: Initialize Timer6**

**-----------------------------------------------------------------------\*/**

**void Init\_Timer6( void )**

{

/\* ensure Timer 6 is in reset state \*/

T6CON = 0;

T6CONbits.TCKPS = 1; //Prescaler 8

/\* reset Timer 6 interrupt flag \*/

IFS2bits.T6IF = 0;

/\* set Timer interrupt priority level \*/

IPC11bits.T6IP = 5;

/\* enable Timer interrupt \*/

IEC2bits.T6IE = 1;

/\* set Timer period register \*/

PR6 = ValPR6;

T6CONbits.TON = 0; //deshabilito Timer

}

**ISR\_UART2.C**

#include "p33FJ256GP710.h"

//UART

#define FCY 40000000

#define BAUDRATE 19200

#define BRGVAL ((FCY / BAUDRATE) / 16) -1

volatile unsigned char caracter;

int cant , contador = 0; //Cantidad de caracteres a enviar

int esPrimero = 1;

int estado = 0;

unsigned int valor;

volatile unsigned char uart\_update;

extern int mensajeRecepcion[];

extern int mensajeTransmicion[];

extern int terminoRecepcion;

**/\*---------------------------------------------------------------------**

**Function Name: UART2Interrupt**

**Description: UART2 Interrupt Handler**

**Inputs: None**

**Returns: None**

**-----------------------------------------------------------------------\*/**

**//Interrupciones de la UART2**

**void \_\_attribute\_\_((interrupt, auto\_psv)) \_U2RXInterrupt( void )**

{

IFS1bits.U2RXIF = 0;

if (!estado){

mensajeRecepcion[contador] = U2RXREG;

if (mensajeRecepcion[contador] == 0x00FE){

estado = 1;

}

}

if (estado){

if ((esPrimero) && (mensajeRecepcion[contador] != 0x00FE)){

mensajeRecepcion[contador] = U2RXREG; //meto segundo valor al registro

esPrimero = 0; //bajo bandera

cant = mensajeRecepcion[contador]; //tomo la cantidad de datos total

} else {

mensajeRecepcion[contador] = U2RXREG;

}

contador++;

if(contador == cant){

contador = 0; //resetea valores

esPrimero = 1; //para el proximo

estado = 0; //mensaje

terminoRecepcion = 1;

}

}

}

**//Rutina de INT para transmisión (no hace nada)**

**void \_\_attribute\_\_((interrupt, auto\_psv)) \_U2TXInterrupt(void)**

{

//Atención. Se debe modificar para que no emita reiteradamente

//caracteres

IFS1bits.U2TXIF = 0;

cant = mensajeTransmicion[1];//El segundo valor del arreglo trae la cantidad a transmitir

if(contador < cant) // de 0 hasta cantidad de datos

{

U2TXREG = mensajeTransmicion[contador]; //Simple ECO

contador++;

} else {

contador = 0;

}

}

**/\*---------------------------------------------------------------------**

**Function Name: InitUART2**

**Description: Inicializar UART2**

**Inputs: None**

**Returns: None**

**-----------------------------------------------------------------------\*/**

**void InitUART2(void)**

{

// The HPC16 board has a DB9 connector wired to UART2,

// so we will be configuring this port only

// configure U2MODE

U2MODEbits.UARTEN = 0; // Bit15 TX, RX DISABLED, ENABLE at end of func

U2MODEbits.RTSMD = 1; // Bit11 Simplex Mode

U2MODEbits.PDSEL = 0; // 8 bits no parity

// Load a value into Baud Rate Generator. Example is for 19200.

U2BRG = BRGVAL; // 40Mhz osc, 19200 Baud

IPC7 = 0x4400; // Mid Range Interrupt Priority level, no urgent reason

IFS1bits.U2RXIF = 0; // Clear the Recieve Interrupt Flag

IEC1bits.U2RXIE = 1; // Enable Recieve Interrupts

U2MODEbits.UARTEN = 1; // And turn the peripheral on

U2STAbits.UTXEN = 1; // Empieza a transmitir. Se dispara el Flag TXIF

IFS1bits.U2TXIF = 0; // Clear the Transmit Interrupt Flag

IEC1bits.U2TXIE = 1; // Enable Transmit Interrupts

}