Escola Universitària Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Grau Enginyeria industrial.

DISSENY D'UN CONTROLADOR UNIVERSAL DE BAIX COST PER APLICACIONS INDUSTRIALS.

Annexos.

David Vega

GP12A

PONENT: Julian Horrillo.

PRIMAVERA 2015

TecnoCampus Mataró-Maresme

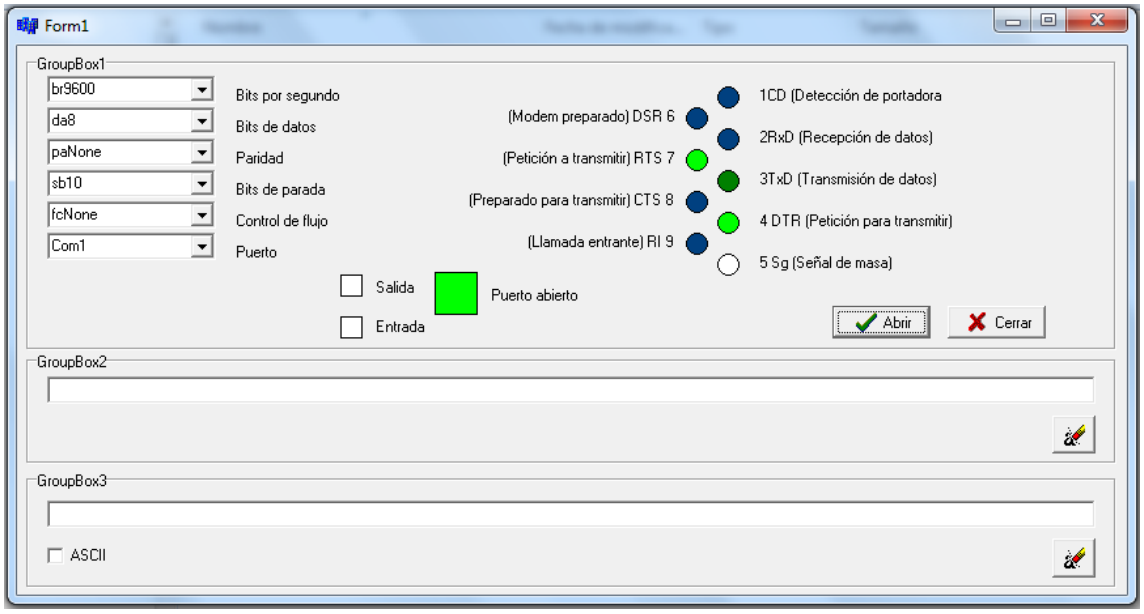
Índex de continguts.

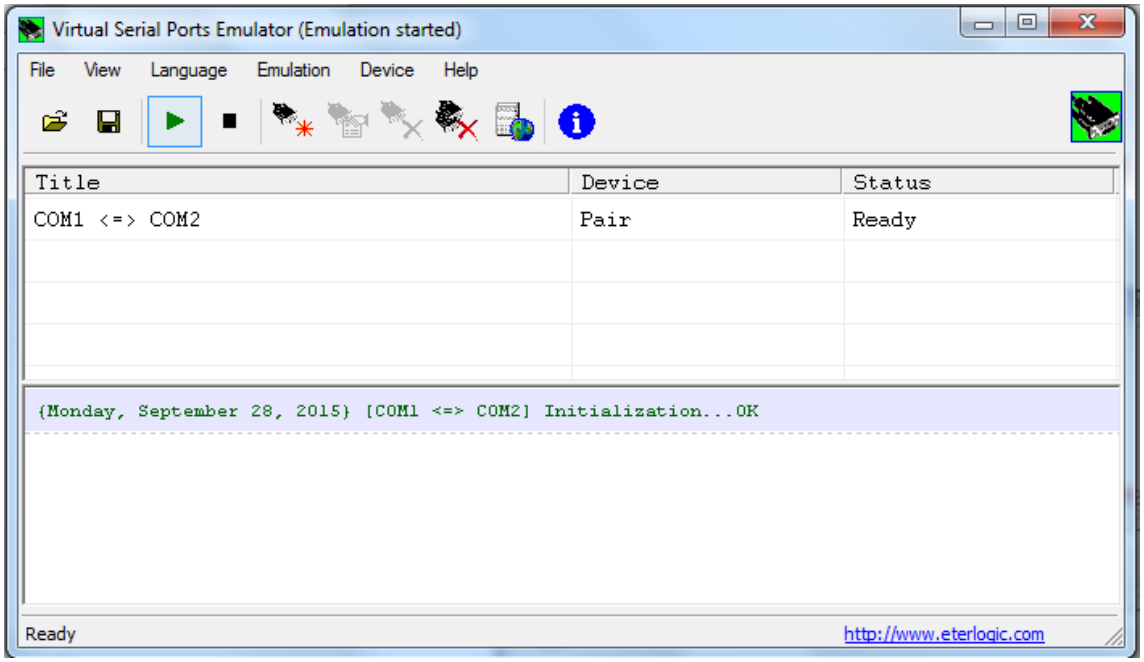
Annex1: Contingut del CD-ROM.

- Documentació del projecte (format PDF).
    - Memòria.
    - Plànols.
    - Estudi Econòmic.
    - Avantprojecte.
    - Annexos.
- Arxius adjunts.
    - Aplicacions informàtiques en Borland Builder C++ 6.
    - Firmwares per al microcontrolador PIC18F97J60-I/PF.
    - Esquemes electrònics en ORCAD Capture.
    - Excel amb la distribució dels PINS del microcontrolador.

Annex2:Programes en Borland Builder C++ 6.

Annex2.1: Comunicació PC→PC via RS232.





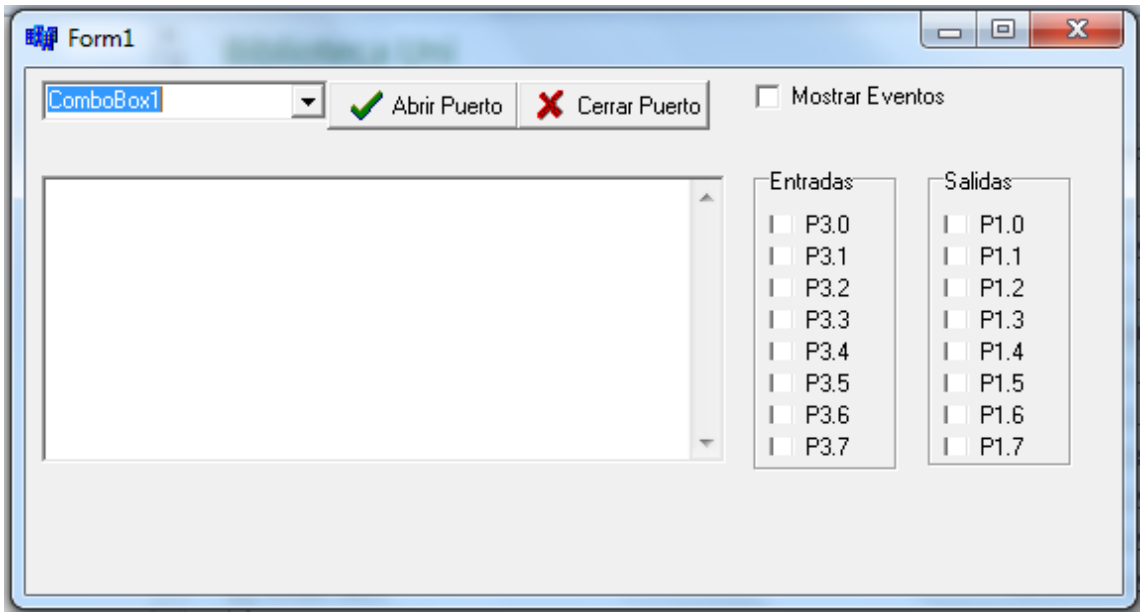
```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "Comm"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
//Establecer los valores por defecto de comunicaciones
    ComboBox1->ItemIndex = br9600; //Bits por segundo
/*      0:  br110
        1:  br300
        2:  br600
        3:  br1200
        4:  br2400
        5:  br4800
        6:  br9600
        7:  br14400
        8:  br19200
        9:  br38400
        10: br56000
        11: br57600
        12: br115200 */
    ComboBox2->ItemIndex = da8; //Bits de datos
/*      0:  da4
        1:  da5
        2:  da6
        3:  da7
        4:  da8 */
    ComboBox3->ItemIndex = paNone; //Paridad
/*      0:  paNone
        1:  paOdd (Par)
        2:  paEven (impar)
        3:  paMark (Marca)
        4:  paSpace (Espacio) */
    ComboBox4->ItemIndex = sb10; //Bits de parada
/*      0:  sb10
        1:  sb15
        2:  sb20 */
    ComboBox5->ItemIndex = fcNone; //Control de flujo
/*      0:  fcNone
        1:  fcCTS
        2:  fcDTR
        3:  fcSottware
        4:  fcDefault */
    ComboBox6->ItemIndex = 0; //Puerto
}
//---------------------------------------------------------------------------
  //Verde Salidas OFF clGreen ON clLime
  //Azul Entradas OFF 0x00804000 ON 0x00FF8000
//---------------------------------------------------------------------------
void __fastcall TForm1::Comm1RxChar(TObject *Sender, DWORD Count)
{
int Estado;
int NumRec;
String Dat;
char BufRec[101];

Shape3->Brush->Color = clLime;
NumRec=Comm1->InQueCount();             //Número de caracteres recibidos
Estado = Comm1->Read(BufRec, Count);    //Leer el Buffer
if(Estado==-1)
    {
    ShowMessage("Error en la recepción");
    }
else
    {
    for(int n=0;n<NumRec;n++)
        {
        if(!CheckBox1->Checked)
            {
            Dat=Dat+IntToStr(BufRec[n])+',';
            }
        else
            {
            Dat=Dat+BufRec[n];
            }
        }
    Edit2->Text=Edit2->Text+Dat;
    }
Shape2->Brush->Color = 0x00FF8000;
Timer1->Enabled=true;
}
//---------------------------------------------------------------------------
void TForm1::EstadoLineas (void)
```

```
 97: {
 98: if(Comm1->CTS)
 99:     {
100:     Shape8->Brush->Color = 0x00FF8000;
101:     }
102: else
103:     {
104:     Shape8->Brush->Color = 0x00804000;
105:     }
106: if(Comm1->DSR)
107:     {
108:     Shape6->Brush->Color = 0x00FF8000;
109:     }
110: else
111:     {
112:     Shape6->Brush->Color = 0x00804000;
113:     }
114: if(Comm1->RING)
115:     {
116:     Shape9->Brush->Color = 0x00FF8000;
117:     }
118: else
119:     {
120:     Shape9->Brush->Color = 0x00804000;
121:     }
122: if(Comm1->RLSD)
123:     {
124:     Shape1->Brush->Color = 0x00FF8000;
125:     }
126: else
127:     {
128:     Shape1->Brush->Color = 0x00804000;
129:     }
130:
131: }
132: //---------------------------------------------------------------------------
133: void __fastcall TForm1::Comm1Cts(TObject *Sender)
134: {
135: //Mirar el estado de la líneas CTS y DSR
136: EstadoLineas();
137: }
138: //---------------------------------------------------------------------------
139: void __fastcall TForm1::Comm1Dsr(TObject *Sender)
140: {
141: //Mirar el estado de la líneas CTS y DSR
142: EstadoLineas();
143: }
144: //---------------------------------------------------------------------------
145: void __fastcall TForm1::BitBtn1Click(TObject *Sender)
146: {
147: if(!Comm1->Enabled())
148:     {
149:     //Establecer los parámetros
150:     Comm1->DeviceName = ComboBox6->Items->Strings[ComboBox6->ItemIndex];//Puerto
151:     Comm1->BaudRate = ComboBox1->ItemIndex;     //Bits por segundo
152:     Comm1->DataBits = ComboBox2->ItemIndex;     //Bits de datos
153:     Comm1->Parity = ComboBox3->ItemIndex;       //Paridad
154:     Comm1->StopBits = ComboBox4->ItemIndex;     //Bits de parada
155:     Comm1->FlowControl = ComboBox5->ItemIndex;  //Control de flujo
156:     //Abrir el puerto de comunicaciones
157:     Comm1->Open();
158:     //Comm1->SetRTSState(true); //Activar la línea RTS
159:     Shape7->Brush->Color = clLime;
160:     //Comm1->SetDTRState(true); //Activar la línea DTR
161:     Shape4->Brush->Color = clLime;
162:     //Activar los indicadores del estado del puerto abierto
163:     Label7->Caption = "Puerto abierto";
164:     Shape10->Brush->Color = clLime;
165:     //Mirar el estado de la líneas CTS y DSR
166:     EstadoLineas();
167:
168:     }
169: }
170: //---------------------------------------------------------------------------
171: void __fastcall TForm1::BitBtn2Click(TObject *Sender)
172: {
173: if(Comm1->Enabled())
174:     {
175:     //Cerrar el puerto de comunicaciones
176:     Comm1->Close();
177:     //Activar los indicadores del estado del puerto abierto
178:     Label7->Caption = "Puerto cerrado";
179:     Shape10->Brush->Color = clRed;
180:     Shape7->Brush->Color = clGreen;
181:     Shape4->Brush->Color = clGreen;
182:     //Mirar el estado de la líneas CTS y DSR
183:     EstadoLineas();
184:     }
185: }
186: //---------------------------------------------------------------------------
187: void __fastcall TForm1::BitBtn3Click(TObject *Sender)
188: {
189:     Edit1->Text="";
190: }
191: //---------------------------------------------------------------------------
192: void __fastcall TForm1::BitBtn4Click(TObject *Sender)
```

```
193: {
194:     Edit2->Text="";
195: }
196: //---------------------------------------------------------------------------
197: void __fastcall TForm1::Comm1Ring(TObject *Sender)
198: {
199: //Mirar el estado de la líneas CTS y DSR
200: EstadoLineas();
201: }
202: //---------------------------------------------------------------------------
203: void __fastcall TForm1::Comm1Rlsd(TObject *Sender)
204: {
205: //Mirar el estado de la líneas CTS y DSR
206: EstadoLineas();
207: }
208: //---------------------------------------------------------------------------
209: void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
210: {
211: char Dat[2];
212:
213: Dat[0]=Key;
214: Comm1->Write(Dat,1);
215:
216: Shape3->Brush->Color = clLime;
217: Timer2->Enabled=true;
218:
219: }
220: //---------------------------------------------------------------------------
221: void __fastcall TForm1::Timer1Timer(TObject *Sender)
222: {
223: Timer1->Enabled=false;
224: Shape2->Brush->Color = 0x00804000;
225: }
226: //---------------------------------------------------------------------------
227: void __fastcall TForm1::Timer2Timer(TObject *Sender)
228: {
229: Timer2->Enabled=false;
230: Shape3->Brush->Color = clGreen;
231: }
232: //---------------------------------------------------------------------------
```

Annex2:Programes en Borland Builder C++ 6.

Annex2.2: Comunicació PC→CUBC via RS232.



```
1: //---------------------------------------------------------------------------
2:
3: #include <vcl.h>
4: #pragma hdrstop
5:
6: #include "Unit1.h"
7: //---------------------------------------------------------------------------
8: #pragma package(smart_init)
9: #pragma link "Comm"
10: #pragma resource "*.dfm"
11: TForm1 *Form1;
12: //---------------------------------------------------------------------------
13: __fastcall TForm1::TForm1(TComponent* Owner)
14:         : TForm(Owner)
15: {
16: }
17: //---------------------------------------------------------------------------
18: void __fastcall TForm1::FormCreate(TObject *Sender)
19: {
20: //Establecer los valores por defecto de comunicaciones
21:     /*ComboBox1->ItemIndex = 0; //Puerto
22:     Comm1->BaudRate=br9600;
23:     Comm1->DataBits=da8;
24:     Comm1->Parity=paNone;
25:     Comm1->StopBits=sb10;
26:     Comm1->FlowControl=fcNone;  */
27:     cond1=0;
28:     Memo1->Clear();
29: }
30: //---------------------------------------------------------------------------
31:
32: void __fastcall TForm1::Comm1RxChar(TObject *Sender, DWORD Count)
33: {
34: int Estado;
35: int NumRec;
36: String Dat;
37: char BufRec[101];
38: NumRec=Comm1->InQueCount();             //Número de caracteres recibidos
39: Estado = Comm1->Read(BufRec, Count);    //Leer el Buffer
40: if(Estado==-1)
41:     {
42:     ShowMessage("Error en la recepción");
43:     }
44: else
45:     {
46:     for(int n=0;n<NumRec;n++)
47:         {
48:             Dat=Dat+BufRec[n];
49: //LLama a la función RecDat con 1 caracter de los recividos 1 vez por cada caracter.
50:             RecDat(BufRec[n]);
51:         }
52:     Memo1->Lines->Add("Se Ha recivido:  "+Dat);
53:
54:     }
55: }
56: //---------------------------------------------------------------------------
57:
58: void __fastcall TForm1::BitBtn1Click(TObject *Sender)
59: {
60: if(!Comm1->Enabled())
61:     {
62:     //Abrir el puerto de comunicaciones
63:     Comm1->Open();
64:     }
65: if(Comm1->Enabled())
66:         {
67:         Memo1->Lines->Add("Puerto Abierto:    "+ComboBox1->Text);
68:         }
69: else
70:         {
71:         Memo1->Lines->Add("Error Abrir Puerto:   "+ComboBox1->Text);
72:         }
73: }
74: //---------------------------------------------------------------------------
75:
76: void __fastcall TForm1::BitBtn2Click(TObject *Sender)
77: {
78: if(Comm1->Enabled())
79:     {
80:     //Cerrar el puerto de comunicaciones
81:     Comm1->Close();
82:     Memo1->Lines->Add("Cerrar puerto");
83:     }
84: }
85: //---------------------------------------------------------------------------
86: void TForm1::RecDat(char dat)
87: {
88: //Esta función consta de dos pasos Esperar letra y esperar numero de checkbox, Esto lo controlamos
     con la variable Cond1
89: //Si estamos en cond1=0;
90:         if (cond1==0)
91:         {
92:         //Si recivimos F o f cargamos false en la variable boleana y pasoamso al paso 2 cargando 1
     end cond 1
93:         //Si recivimos N o n lo mismo pero cargamos true;
94:         switch(dat){
```

```
 95:                 case 'F':
 96:                         aux=false;
 97:                         cond1=1;
 98:                         break;
 99:                 case 'f':
100:                         aux=false;
101:                         cond1=1;        //Bit P1.1 a 0
102:                         break;
103:                 case 'N':
104:                         aux=true;
105:                         cond1=1;        //Bit P1.2 a 0
106:                         break;
107:                 case 'n':
108:                         aux=true;
109:                         cond1=1;        //Bit P1.3 a 0
110:                         break;
111:         }
112:         }
113: // Si estamos en cond1=1 comprovamos que recivimos un numero de 0 a 7.
114: //Si es true segun se de 0-7 cambiamos el estado del check box corriespondente.
115: //Si es fols pero recivimos M o m o N o p volvemos a cargar un 1 en cond1
116: //Si no volvemos a cond1=0;
117:                 else if (cond1==1)
118:         {
119:                 if
     (dat=='0'||dat=='1'||dat=='2'||dat=='3'||dat=='4'||dat=='5'||dat=='6'||dat=='7')
120:                 {
121:                 switch(dat){
122:                 // Poner a cero los bits de puerto 1
123:                 case '0':
124:                         CheckBox2->Checked = aux;       //Bit P1.0 a 0
125:                         cond1=0;
126:                         break;
127:                 case '1':
128:                         CheckBox3->Checked = aux;       //Bit P1.1 a 0
129:                         cond1=0;
130:                         break;
131:                 case '2':
132:                         CheckBox4->Checked = aux;
133:                         cond1=0;        //Bit P1.2 a 0
134:                         break;
135:                 case '3':
136:                         CheckBox5->Checked = aux;
137:                         cond1=0;        //Bit P1.3 a 0
138:                         break;
139:                 case '4':
140:                         CheckBox6->Checked = aux;
141:                         cond1=0;        //Bit P1.4 a 0
142:                         break;
143:                 case '5':
144:                         CheckBox7->Checked = aux;
145:                         cond1=0;        //Bit P1.5 a 0
146:                         break;
147:                 case '6':
148:                         CheckBox8->Checked = aux;
149:                         cond1=0;        //Bit P1.6 a 0
150:                         break;
151:                 case '7':
152:                         CheckBox9->Checked = aux;
153:                         cond1=0;        //Bit P1.7 a 0
154:                         break;
155:                         }
156:                         }
157:
158:                 else if(dat=='P'||dat=='p'||dat=='M'||dat=='m' )
159:                 {
160:                 cond1=1;
161:                 }
162:                 else
163:                 {
164:                 cond1=0;
165:                 }
166:
167: }
168: }
169: void __fastcall TForm1::CheckBox10Click(TObject *Sender)
170: {
171: //Comprovamos el estado del checkbox y lanzamos por la uart M y el nuemro indicado si es true o P
      y lo mismo si es false
172: if (CheckBox10->Checked==true)
173:         {
174:                 dat[0]='M';
175:                 dat[1]='0';
176:                 Comm1->Write(dat,2);
177:                 if (Comm1->Enabled()==true)
178:                 {
179:                 Memo1->Lines->Add("Se a enviado:        M0");
180:                 }
181:         }
182: else if(CheckBox10->Checked==false)
183:         {
184:                 dat[0]='P';
185:                 dat[1]='0';
186:                 Comm1->Write(dat,2);
187:                 if (Comm1->Enabled()==true)
188:                 {
```

```
189:                 Memo1->Lines->Add("Se a enviado:        P0");
190:                 }
191:         }
192: }
193: //---------------------------------------------------------------------------
194:
195: void __fastcall TForm1::CheckBox11Click(TObject *Sender)
196: {
197: if (CheckBox11->Checked==true)
198:         {
199:                 dat[0]='M';
200:                 dat[1]='1';
201:                 Comm1->Write(dat,2);
202:                 if (Comm1->Enabled()==true)
203:                 {
204:                 Memo1->Lines->Add("Se a enviado:        M1");
205:                 }
206:         }
207: else if(CheckBox11->Checked==false)
208:         {
209:                 dat[0]='P';
210:                 dat[1]='1';
211:                 Comm1->Write(dat,2);
212:                 if (Comm1->Enabled()==true)
213:                 {
214:                 Memo1->Lines->Add("Se a enviado:        P1");
215:                 }
216:         }
217: }
218: //---------------------------------------------------------------------------
219:
220: void __fastcall TForm1::CheckBox12Click(TObject *Sender)
221: {
222: if (CheckBox12->Checked==true)
223:         {
224:                 dat[0]='M';
225:                 dat[1]='2';
226:                 Comm1->Write(dat,2);
227:                 if (Comm1->Enabled())
228:                 {
229:                 Memo1->Lines->Add("Se a enviado:        M2");
230:                 }
231:         }
232: else if(CheckBox12->Checked==false)
233:         {
234:                 dat[0]='P';
235:                 dat[1]='2';
236:                 Comm1->Write(dat,2);
237:                 if (Comm1->Enabled())
238:                 {
239:                 Memo1->Lines->Add("Se a enviado:        P2");
240:                 }
241:         }
242: }
243: //---------------------------------------------------------------------------
244:
245: void __fastcall TForm1::CheckBox13Click(TObject *Sender)
246: {
247: if (CheckBox13->Checked==true)
248:         {
249:                 dat[0]='M';
250:                 dat[1]='3';
251:                 Comm1->Write(dat,2);
252:                 if (Comm1->Enabled())
253:                 {
254:                 Memo1->Lines->Add("Se a enviado:        M3");
255:                 }
256:         }
257: else if(CheckBox13->Checked==false)
258:         {
259:                 dat[0]='P';
260:                 dat[1]='3';
261:                 Comm1->Write(dat,2);
262:                 if (Comm1->Enabled())
263:                 {
264:                 Memo1->Lines->Add("Se a enviado:        P3");
265:                 }
266:         }
267: }
268: //---------------------------------------------------------------------------
269:
270: void __fastcall TForm1::CheckBox15Click(TObject *Sender)
271: {
272: if (CheckBox15->Checked==true)
273:         {
274:                 dat[0]='M';
275:                 dat[1]='5';
276:                 Comm1->Write(dat,2);
277:                 if (Comm1->Enabled())
278:                 {
279:                 Memo1->Lines->Add("Se a enviado:        M5");
280:                 }
281:         }
282: else if(CheckBox15->Checked==false)
283:         {
284:                 dat[0]='P';
```

```
285:                 dat[1]='5';
286:                 Comm1->Write(dat,2);
287:                 if (Comm1->Enabled())
288:                 {
289:                 Memo1->Lines->Add("Se a enviado:        P5");
290:                 }
291:         }
292: }
293: //---------------------------------------------------------------------------
294:
295: void __fastcall TForm1::CheckBox14Click(TObject *Sender)
296: {
297: if (CheckBox14->Checked==true)
298:         {
299:                 dat[0]='M';
300:                 dat[1]='4';
301:                 Comm1->Write(dat,2);
302:                 if (Comm1->Enabled())
303:                 {
304:                 Memo1->Lines->Add("Se a enviado:        M4");
305:                 }
306:         }
307: else if(CheckBox14->Checked==false)
308:         {
309:                 dat[0]='P';
310:                 dat[1]='4';
311:                 Comm1->Write(dat,2);
312:                 if (Comm1->Enabled())
313:                 {
314:                 Memo1->Lines->Add("Se a enviado:        P4");
315:                 }
316:         }
317: }
318: //---------------------------------------------------------------------------
319:
320: void __fastcall TForm1::CheckBox16Click(TObject *Sender)
321: {
322: if (CheckBox16->Checked==true)
323:         {
324:                 dat[0]='M';
325:                 dat[1]='6';
326:                 Comm1->Write(dat,2);
327:                 if (Comm1->Enabled())
328:                 {
329:                 Memo1->Lines->Add("Se a enviado:        M6");
330:                 }
331:         }
332: else if(CheckBox16->Checked==false)
333:         {
334:                 dat[0]='P';
335:                 dat[1]='6';
336:                 Comm1->Write(dat,2);
337:                 if (Comm1->Enabled())
338:                 {
339:                 Memo1->Lines->Add("Se a enviado:        P6");
340:                 }
341:         }
342: }
343: //---------------------------------------------------------------------------
344:
345: void __fastcall TForm1::CheckBox17Click(TObject *Sender)
346: {
347: if (CheckBox17->Checked==true)
348:         {
349:                 dat[0]='M';
350:                 dat[1]='7';
351:                 Comm1->Write(dat,2);
352:                 if (Comm1->Enabled())
353:                 {
354:                 Memo1->Lines->Add("Se a enviado:        M7");
355:                 }
356:         }
357: else if(CheckBox17->Checked==false)
358:         {
359:                 dat[0]='P';
360:                 dat[1]='7';
361:                 Comm1->Write(dat,2);
362:                 if (Comm1->Enabled())
363:                 {
364:                 Memo1->Lines->Add("Se a enviado:        P7");
365:                 }
366:         }
367: }
368: //---------------------------------------------------------------------------
369:
```

Annex3:Firmwares pel microcontrolador.

Annex3.1: P1→ Parpadeig Ledprog.

```
;  Titulo: Parpadeig Ledprog.
;   Autor: David Vega.
;   Comentarios
; Configuración del PIC18F97J60, parpadeo del puerto RC5 a través
; de un retardo por bucle de 0.1 s
    list p=18F97J60 ;indica el tipo de procesador.
    INCLUDE "P18F97J60.INC"; Incluye las definiciones del pic



; CONFIG1L
  CONFIG  DEBUG = ON         ; permite depurar el programa.
  CONFIG  WDT = OFF          ; Watchdog Timer Enable bit (WDT enabled)
  CONFIG  STVR = ON          ; Stack Overflow/Underflow Reset Enable bit (Reset
                             ; on stack overflow/underflow enabled)
  CONFIG  XINST = OFF        ; Extended Instruction Set Enable bit (Instruction
                             ; set extension and Indexed Addressing mode enabled)

; CONFIG1H
  CONFIG  CP0 = OFF          ; Code Protection bit (Program memory is not
                             ; code-protected)

; CONFIG2L
  CONFIG  FOSC = ECPLL       ; Oscillator Selection bits (EC oscillator, PLL
                             ; enabled and under software control, CLKO function
                             ; on OSC2)
  CONFIG  FOSC2 = ON         ; Default/Reset System Clock Select bit (Clock
                             ; selected by FOSC1:FOSC0 as system clock is enabled
                             ; when OSCCON<1:0> = 00)
  CONFIG  FCMEN = ON         ; Fail-Safe Clock Monitor Enable (Fail-Safe Clock
                             ; Monitor enabled)
  CONFIG  IESO = ON          ; Two-Speed Start-up (Internal/External Oscillator
                             ; Switchover) Control bit (Two-Speed Start-up
                             ; enabled)
; CONFIG2H
  CONFIG  WDTPS = 32768      ; Watchdog Timer Postscaler Select bits (1:32768)

; CONFIG3L
  CONFIG  EASHFT = OFF       ; External Address Bus Shift Enable bit (Address
                             ;shifting enabled; address on external bus is offset
                             ; to start at 000000h)
  CONFIG  MODE = MM          ; External Memory Bus (Microcontroller mode,
                             ; external bus disabled)
  CONFIG  BW = 16            ; Data Bus Width Select bit (16-Bit Data Width mode)
  CONFIG  WAIT = OFF         ; External Bus Wait Enable bit (Wait states for
                             ; operations on external memory bus disabled)

; CONFIG3H
  CONFIG  CCP2MX = OFF       ; ECCP2 MUX bit (ECCP2/P2A is multiplexed with RE7
                             ; in Microcontroller mode (80-pin and 100-pin
```

```
                            ; devices) or with RB3 in Extended Microcontroller
                            ; mode (100-pin devices only))
  CONFIG  ECCPMX = OFF      ; ECCP MUX bit (ECCP1 outputs (P1B/P1C) are
                            ; multiplexed with RH7 and RH6; ECCP3 outputs
                            ;(P3B/P3C) are multiplexed with RH5 and RH4)
  CONFIG  ETHLED = OFF      ; Ethernet LED Enable bit (RA0/RA1 function as
                            ;I/O regardless of Ethernet module status)

        ORG     00;
        CLRF    PORTC;      ; Configuració del port a utilitzar com sortida.
        CLRF    LATC;
        MOVLW   B'00001111';
        MOVWF   TRISC;
inicio:

        Call tempo1;        ; Cridar a la funció de temporització.

        BTG     PORTC,5,0;  ; Complement al port 5.

        Goto    inicio;     ; reinici del codi.

;*********************************************************************************
tempo1: MOVLW   0xFF;  Funció de retard.
        MOVWF   0x01;
        MOVLW   0xFF;
buc11:  MOVWF   0x00;
bucle1: DECFSZ  0x00,1,1;
        goto bucle1;
        DECFSZ  0x01,1,1;
        goto  buc11;
        RETURN;
;*********************************************************************************
        END; Fin del codi.
```

Annex3:Firmwares pel microcontrolador.

Annex3.2: P2→ Prova Sortides digitals.

Consisteix en un programa que fa intermitència a les sortides analògiques, activen alternativament les sortides senars del CUBC i per altre banda els parells.

```
/*
 * File:   Header file Proves Sortides Digitals
 * Author: David
 *
 * Created on 2 de mayo de 2015, 18:25
 */



// PIC18F97J60 Configuration Bit Settings

// 'C' source line config statements

#include <xc.h>

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

// CONFIG1L
#pragma config WDT = OFF
// Watchdog Timer Enable bit (WDT disabled (control is placed on SWDTEN bit))
#pragma config STVR = ON
// Stack Overflow/Underflow Reset Enable bit
#pragma config XINST = OFF
// Extended Instruction Set Enable bit

// CONFIG1H
#pragma config CP0 = OFF
// Code Protection bit (Program memory is not code-protected)

// CONFIG2L
#pragma config FOSC = EC
// Oscillator Selection bits (EC oscillator, CLKO function on OSC2)
#pragma config FOSC2 = ON
// Default/Reset System Clock Select bit
#pragma config FCMEN = ON
// Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor enabled)
#pragma config IESO = OFF
// Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit

// CONFIG2H
#pragma config WDTPS = 32768
// Watchdog Timer Postscaler Select bits (1:32768)

// CONFIG3L
#pragma config EASHFT = OFF
// External Address Bus Shift Enable bit (Address shifting disabled;
#pragma config MODE = MM
// External Memory Bus (Microcontroller mode, external bus disabled)
#pragma config BW = 8
```

```
// Data Bus Width Select bit (8-Bit Data Width mode)
#pragma config WAIT = OFF
// External Bus Wait Enable bit

// CONFIG3H
#pragma config CCP2MX = OFF
// ECCP2 MUX bit (ECCP2/P2A is multiplexed with RC1)
#pragma config ECCPMX = OFF
// ECCP MUX bit (ECCP1 outputs (P1B/P1C) are multiplexed with RE6 and RE5;)
#pragma config ETHLED = OFF
// Ethernet LED Enable bit

//oscilator frequency
#define _XTAL_FREQ 25000000
```

```
/*
 * File:   Proves sortides digitals.
 * Author: David
 * Parpadeo de 0.5s en les 8 sortides optoacoblades.
 * Created on 2 de mayo de 2015, 18:31
 */

#include <stdio.h>
#include <stdlib.h>
#include "config.h"
#include <xc8debug.h>
#include <pic18f97j60.h>

int main(int argc, char** argv) {

    LATC=0x00;
    TRISC=0x0F;

    LATB=0x00;
    TRISB=0x3F;

    LATD=0x00;
    TRISD=0xF9;

    RC5=0;
    RB0=1;
    RB1=0;
    RB2=1;
    RB3=0;
    RB4=1;
    RB5=0;
    RD2=1;
    RD3=0;

    while(1)
    {
        RC5=~RC5;
        RB0=~RB0;
        RB1=~RB1;
        RB2=~RB2;
        RB3=~RB3;
        RB4=~RB4;
        RB5=~RB5;
        RD2=~RD2;
        RD3=~RD3;

        for (int count_Delay=0; count_Delay <100;count_Delay++)
        {
           __delay_ms(5);
        }
```

```
    }

    return (EXIT_SUCCESS);
}
```

Annex3:Firmwares pel microcontrolador.

Annex3.3: P3→ Prova Entrades digitals.

Consisteix en un programa que interpreta l'estat de les entrades i iguala la sortida corresponent.

```c
/*
 * File:   Prova entradesDigitals.
 * Author: David
 * Transeferencia de l'estat de les entrades a les sortides.
 * Created on 2 de mayo de 2015, 18:31
 */

#include <stdio.h>
#include <stdlib.h>
#include "config.h"
#include <xc8debug.h>
#include <pic18f97j60.h>

int main(int argc, char** argv) {
    PORTC = 0x00;
    LATC = 0x00;
    TRISC = 0x00;

    PORTB = 0x00;
    LATB = 0x00;
    TRISB = 0x00;

    PORTD = 0x00;
    LATD = 0x00;
    TRISD = 0x00;

    PORTJ = 0x00;
    LATJ = 0x00;
    TRISJ = 0x1F;

    PORTH = 0x00;
    LATH = 0x00;
    TRISF = 0x0F;

    PORTG = 0x00;
    LATG = 0x00;
    TRISG = 0x00;
    while (1) {
        RB0=RH0;
        RB1 = RH1;
        RB2 = RH2;
        RB3 = RH3;
        RB4 = RJ4;
        RB5 = RJ1;
        RD2= RJ2;
        RD3 = RJ3;
    }
    return (EXIT_SUCCESS);
}
```

Annex3:Firmwares pel microcontrolador.

Annex3.4: P4→ Sortides analogiques.

Consisteix en un programa que realitza una conversió analògica digital de 8 bits interpretant les entrades digitals i dona una sortida en conseqüència.

```
;*****************************************************************************
;                                                                            *
;    Filename: Prova sortidas analògiques.                                                                  *
;    Date:                                                                   *
;    File Version: `1 de Setembre de 2015                                    *
;                                                                            *
;    Author:David Vega                                                       *
;    Company:  Escola Universitaria Politècnica de Matar                     *
;                                                                            *
;*****************************************************************************
;                                                                            *
;    Files Required: P18F97J60.INC                                           *
;                                                                            *
;*****************************************************************************

    LIST P=18F97J60     ;directive to define processor
    #include <P18F97J60.INC>    ;processor specific variable definitions


;*****************************************************************************
;Configuration bits

;configuration bits
; CONFIG1L
  CONFIG  DEBUG = ON        ; permite depurar el programa.
  CONFIG  WDT = OFF         ; Watchdog Timer Enable bit (WDT enabled)
  CONFIG  STVR = ON         ; Stack Overflow/Underflow Reset Enable bit
  CONFIG  XINST = OFF       ; Extended Instruction Set Enable bit

; CONFIG1H
  CONFIG  CP0 = ON          ; (Program memory is not code-protected)

; CONFIG2L
  CONFIG  FOSC = ECPLL      ; Oscillator Selection bits
  CONFIG  FOSC2 = ON        ; Default/Reset System Clock Select bit
  CONFIG  FCMEN = ON        ; Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor enabled)
  CONFIG  IESO = ON         ; Two-Speed Start-up

; CONFIG2H
  CONFIG  WDTPS = 32768     ; Watchdog Timer Postscaler Select bits (1:32768)

; CONFIG3L
  CONFIG  EASHFT = OFF      ; External Address Bus Shift Enable bit
  CONFIG  MODE = MM         ; (Microcontroller mode, external bus disabled)
  CONFIG  BW = 16           ; Data Bus Width Select bit (16-Bit Data Width mode)
  CONFIG  WAIT = OFF        ; External Bus Wait Enable bit

; CONFIG3H
  CONFIG  CCP2MX = OFF      ; ECCP2 MUX bit

  CONFIG  ECCPMX = OFF      ; ECCP MUX bit
  CONFIG  ETHLED = OFF      ; Ethernet LED Enable bit

;*****************************************************************************
;Variable definitions
; More variables may be needed to store other special function registers used

       EEADDR  equ 0x00    ; Address register
       EESLAVE equ 0x04    ; Device address (10011[A1][A0]X)

       EECTRLB equ 0x01    ; data to read/write (Control byte)
       EEMSB   equ 0x02   ; data to read/write (MSB)
       EELSB   equ 0x03   ; data to read/write (LSB)
       HSMAST equ 0x0E   ; data to send HS mode
       CTRLDACA    equ 0x10    ; Literal control byte to update DAC A
       SlaveAddr   equ 0x9A    ; slave address literal (1001 110z)
```

```
        CBLOCK  0x060
        WREG_TEMP   ;variable used for context saving
        STATUS_TEMP ;variable used for context saving
        BSR_TEMP    ;variable used for context saving
        ENDC



;*******************************************************************************
;Reset vector
; This code will start executing when a reset occurs.

        ORG 0x0000

        goto    Main        ;go to start of main code

;*******************************************************************************
;High priority interrupt vector
; This code will start executing when a high priority interrupt occurs or
; when any interrupt occurs if interrupt priorities are not enabled.

        ORG 0x0008

        bra HighInt     ;go to high priority interrupt routine

;*******************************************************************************
;Low priority interrupt vector and routine
; This code will start executing when a low priority interrupt occurs.
; This code can be removed if low priority interrupts are not used.

        ORG 0x0018

        movff   STATUS,STATUS_TEMP  ;save STATUS register

        movff   WREG,WREG_TEMP      ;save working register
        movff   BSR,BSR_TEMP        ;save BSR register

;   *** low priority interrupt code goes here ***


        movff   BSR_TEMP,BSR        ;restore BSR register
        movff   WREG_TEMP,WREG      ;restore working register
        movff   STATUS_TEMP,STATUS  ;restore STATUS register
        retfie

;*******************************************************************************
;High priority interrupt routine
; The high priority interrupt code is placed here to avoid conflicting with
; the low priority interrupt vector.

HighInt:

;   *** high priority interrupt code goes here ***


        retfie  FAST

;*******************************************************************************
;Start of main program
; The main program code is placed here.

Main:

;   *** main code goes here ***

    ;Configuración MSSP2
bcf SSP2STAT,SMP    ; I2C slew rate IN HS mode.
```

```
bsf SSP2CON1,SSPM3  ;MASTER MODE ENABLED
bcf SSP2CON1,SSPM2  ;
bcf SSP2CON1,SSPM1  ;
bcf SSP2CON1,SSPM0  ;

bsf SSP2CON1,SSPEN  ; enable SSP module

    movlw   0x0F        ; set I2C clock rate to 400kHz
    movwf   SSP2ADD

    bsf TRISC,3     ; I2C SCL pin is input
    bsf PORTC,3     ; (will be controlled by SSP)
    bsf TRISC,4     ; I2C SDA pin is input
    bsf PORTC,4     ; (will be controlled by SSP)

    bsf TRISH,0     ; configure INPUTS

    bsf PORTH,0     ;
    bsf TRISH,1     ; configure INPUTS
    bsf PORTH,1     ;
    bsf TRISH,2     ; configure INPUTS
    bsf PORTH,2     ;
    bsf TRISH,3     ; configure INPUTS
    bsf PORTH,3     ;

    bsf TRISJ,4     ; configure INPUTS
    bsf PORTJ,4     ;
    bsf TRISJ,1     ; configure INPUTS
    bsf PORTJ,1     ;
    bsf TRISJ,2     ; configure INPUTS
    bsf PORTJ,2     ;
    bsf TRISJ,3     ; configure INPUTS
    bsf PORTJ,3     ;



    bcf LATC,5      ;Clear por C5
    bcf TRISC,5     ; por C5 as Output

    bcf LATD,5      ;Clear port D5
    bcf TRISC,5     ;Configura as input, Importan for SDA2.
    bcf LATD,6      ;Clear port D5
    bcf TRISC,6     ;Configura as input, Importan for SCL2.


    movlw   SlaveAddr   ; DAC I2C address
    movwf   EESLAVE

    movlw   CTRLDACA    ; DAC A Control byte for update.
    movwf   EECTRLB     ;



Loop:

        btfsc   PORTH,0;
        bsf     EEMSB,0;
        btfsc   PORTH,1;
        bsf     EEMSB,1;
        btfsc   PORTH,2;
        bsf     EEMSB,2;
        btfsc   PORTH,3;
        bsf     EEMSB,3;
        btfsc   PORTJ,4; Mal contacto en PIC no funciona bien
        bsf     EEMSB,4;
        btfsc   PORTJ,1;
        bsf     EEMSB,5;
```

```
        btfsc   PORTJ,2;
        bsf     EEMSB,6;
        btfsc   PORTJ,3;
        bsf     EEMSB,7;

    rcall   WakeSlave   ;
    rcall   WrHSMODE;
    rcall   WrADDR  ;
    rcall   WrCTRLB ;
    rcall   WrEEMSB ;
    rcall   WrEELSB ;
    rcall   Stop    ;









    goto Loop


;*************************************************************
; sends start bit, slave address
; if ACK not recieved, sends restart, tries again
; execution can get stuck in this loop if slave not present
; can be used to poll slave status (retries until slave responds)

wakeSlave
    bsf SSP2CON2,SEN    ; Send start bit
    btfsc   SSP2CON2,SEN    ; Has SEN cleared yet?
    goto    $-2     ; No, loop back to test.
    return

rWakeSlave
    bcf PIR3,SSP2IF ; clear interrupt flag
    nop
    movf    EESLAVE,W
    movwf   SSP2BUF     ; move slave address to SSPBUF
    btfss   PIR3,SSP2IF     ; has SSP completed sending SLAVE Address?
    goto    $-2     ; no, loop back to test

    btfss   SSP2CON2,ACKSTAT    ; was ACK received from slave?
    return          ; yes, return to calling routine

    bsf SSP2CON2,RSEN   ; send repeated start bit
    btfsc   SSP2CON2,RSEN   ; has repeated start been sent yet?
    goto    $-2     ; no, loop back to test

    bra rWakeSlave  ; send slave address again


;*************************************************************

;*************************************************************
; writes DAC memory address, hangs if no ACK

wrHSMODE
    bcf PIR3,SSP2IF ; clear interrupt flag

    movff   HSMAST,SSPBUF   ; move DAC address to SSPBUF
    btfss   PIR3,SSP2IF ; has SSP completed sending DAC Address?
    goto    $-2     ; no, loop back to test

    btfsc   SSP2CON2,ACKSTAT    ; has slave sent ACK?
    goto    $-2     ; no, try again

    return
```

```
;*********************************************************************

;***********************************************************
; writes DAC memory address, hangs if no ACK

wrADDR
    bcf PIR3,SSP2IF ; clear interrupt flag

    movff   EEADDR,SSPBUF   ; move DAC address to SSPBUF
    btfss   PIR3,SSP2IF ; has SSP completed sending DAC Address?
    goto    $-2     ; no, loop back to test

    btfsc   SSP2CON2,ACKSTAT    ; has slave sent ACK?
    goto    $-2     ; no, try again

    return
;*********************************************************************


;***********************************************************
; writes DAC memory address, hangs if no ACK

wrCTRLB
    bcf PIR3,SSP2IF ; clear interrupt flag

    movff   EECTRLB,SSPBUF  ; move DAC address to SSPBUF
    btfss   PIR3,SSP2IF ; has SSP completed sending DAC Address?
    goto    $-2     ; no, loop back to test

    btfsc   SSP2CON2,ACKSTAT    ; has slave sent ACK?
    goto    $-2     ; no, try again

    return

;*********************************************************************
;***********************************************************
; writes DAC memory address, hangs if no ACK

wrEEMSB
    bcf PIR3,SSP2IF ; clear interrupt flag

    movff   EEMSB,SSPBUF    ; move DAC address to SSPBUF
    btfss   PIR3,SSP2IF ; has SSP completed sending DAC Address?
    goto    $-2     ; no, loop back to test

    btfsc   SSP2CON2,ACKSTAT    ; has slave sent ACK?
    goto    $-2     ; no, try again

    return
;*********************************************************************
;***********************************************************
; writes DAC memory address, hangs if no ACK

wrEELSB
    bcf PIR3,SSP2IF ; clear interrupt flag

    movff   EELSB,SSPBUF    ; move DAC address to SSPBUF
    btfss   PIR3,SSP2IF ; has SSP completed sending DAC Address?
    goto    $-2     ; no, loop back to test

    btfsc   SSP2CON2,ACKSTAT    ; has slave sent ACK?
    goto    $-2     ; no, try again

    return
;*********************************************************************
;***********************************************************
; Sends stop bit, waits until sent
```

```
Stop
    bsf SSP2CON2,PEN    ; send stop bit
    btfsc   SSP2CON2,PEN    ; has stop bit been sent?
    goto    $-2     ; no, loop back to test

    return

;*********************************************************



;End of program

        END
```

Annex3:Firmwares pel microcontrolador.

Annex3.5: P5→ Entrades analògiques.

El programa consisteix en ADC de 8 bits, interpreta l'estat de l'entrada analògica i la transforma en un sortida digital de 8 bits pel port de sortides.

```c
/*
 * File:   Prova Entrades Analogiques
 * Author: David

 Aquest programa es un conversor A/D de 8 bits utilitzan el ADC de 10 bits
 del PIC18F97J60
 *
 * Created on 2 de mayo de 2015, 18:31
 */

#include <stdio.h>
#include <stdlib.h>
#include "config.h"
#include <xc8debug.h>
#include <pic18f97j60.h>


//*****************************************************************************
// Aquesta subrutina fa la conversió i retorna el valor.
//*****************************************************************************
unsigned int Read_ADC_Value(void)
{
    unsigned int ADCValue;

    ADCON0bits.GO = 1;              // Comenzar la conversió
    while (ADCON0bits.GO);          // Esperar que la conversió termini.
    ADCValue = ADRESH << 8;         /* Extreu els dos bits de mes pes i els
                                     rota a la dreta per obtenir el seupes real
                                     */
    ADCValue = ADCValue + ADRESL;   // Ara suma el valor del byte de mensys pes
    return (ADCValue);              // Retorna el valor de 10 bits
}


int main(int argc, char** argv) {
    while (1) {

        unsigned int AnalogValue;       // used to store ADC result after capture
        unsigned int VAR;

        //Configuració dels ports.
    LATA = 0x00;
    TRISA=0x1F;

    LATC = 0x00;
    TRISC = 0x00;

    LATB = 0x00;
    TRISB = 0x00;


    LATD = 0x00;
    TRISD = 0x00;

    LATJ = 0x00;
    TRISJ = 0x1F;

    LATH = 0x00;
    TRISF = 0x0F;

    LATG = 0x00;
    TRISG = 0x00;

        ADCON1= 0x07;           //Configurar AN(0-3) com entrades analógiques
        ADCON0bits.CHS=0x00;    //Seleció del canal
        ADCON0bits.ADON=1;              // ADC is on
        ADCON2bits.ADCS=0x01;   // select ADC conversion clock select as 25M/8
        ADCON2bits.ADFM=1;    //    Activa el AC
```

```
    while (1)
    {

        AnalogValue=Read_ADC_Value(); //Trucada a Subrutina de lectura.
        RB0=AnalogValue.3;          //
        RB1=AnalogValue.4;
        RB2=AnalogValue.5;
        RB3=AnalogValue.6;
        RB4=AnalogValue.7;
        RB5=AnalogValue.8;
        RD2=AnalogValue.9;
        RD3=AnalogValue.10;

    }
    }
    return (EXIT_SUCCESS);
}
```

Annex3:Firmwares pel microcontrolador.

Annex3.6: P6→ Comunicació RS232.

Programa que envia l'estat de les entrades al PC. Després interpreta la trama rebuda pel PC per actuar en conseqüència a les sortides.

```
/*
 * File:   Prova RS232 PC
 * Author: David

 Aquest programa es un conversor A/D de 8 bits utilitzan el ADC de 10 bits
 del PIC18F97J60
 *
 * Created on 2 de mayo de 2015, 18:31
 */

#include <stdio.h>
#include <stdlib.h>
#include "config.h"
#include <xc8debug.h>
#include <pic18f97j60.h>

/* Modificació al ficher del PIC18F97J60
Asiganció dels ports de sortida del PIC
bit Led_0 = RB0;        // Asignación del bit P1.0 al símbolo Led_0
bit Led_1 = RB1;        // Asignación del bit P1.1 al símbolo Led_1
bit Led_2 = RB2;        // Asignación del bit P1.2 al símbolo Led_2
bit Led_3 = RB3;        // Asignación del bit P1.3 al símbolo Led_3
bit Led_4 = RB4;        // Asignación del bit P1.4 al símbolo Led_4
bit Led_5 = RB5;        // Asignación del bit P1.5 al símbolo Led_5
bit Led_6 = RD2;        // Asignación del bit P1.6 al símbolo Led_6
bit Led_7 = RD3;        // Asignación del bit P1.7 al símbolo Led_7
asignacion puerto de entradas para placa electronica
bit Led1_0 = RH0;       // Asignación del bit P3.0 al símbolo Led_0
bit Led1_1 = RH1;       // Asignación del bit P3.1 al símbolo Led_1
bit Led1_2 = RH2;       // Asignación del bit P3.2 al símbolo Led_2
bit Led1_3 = RH3;       // Asignación del bit P3.3 al símbolo Led_3
bit Led1_4 = RJ0;       // Asignación del bit P3.4 al símbolo Led_4
bit Led1_5 = RJ1;       // Asignación del bit P3.5 al símbolo Led_5
bit Led1_6 = RJ2;       // Asignación del bit P3.6 al símbolo Led_6
bit Led1_7 = RJ3;       // Asignación del bit P3.7 al símbolo Led_7
 *
 */
        // Declaración de las funciones prototipo
void InitSerial(void);          // Iniciar el puerto serie
void RecDat(void);              // Función Recepción de datos
void LanDat (void);     // Funcior de lanzar datos
void InitPort(void);            //

// Declaración de Variables globales
unsigned char cond1,aux;
unsigned char a,b,c,d,e,f,g,h;

//-------------------------------------------------------------
// Programa principal
//-------------------------------------------------------------

void main(void){

// Tareas iníciales de preparación
    PORTC = 0x00;
    LATC = 0x00;
    TRISC = 0x80;

    PORTB = 0x00;
    LATB = 0x00;
    TRISB = 0x00;

    PORTD = 0x00;
    LATD = 0x00;
    TRISD = 0x00;

    PORTJ = 0x00;
    LATJ = 0x00;
```

```
    TRISJ = 0x1F;

    PORTH = 0x00;
    LATH = 0x00;
    TRISF = 0x0F;

    PORTG = 0x00;
    LATG = 0x00;
    TRISG = 0x00;

    InitSerial();       // Iniciar las comunicaciones
    InitPort();
    // Igualar el valor de la entrada al valor de la entrada anterior.


                // Cuerpo del programa en un bucle infinito
    while(1){
// Muestrear el indicador RI hasta la llegada de un nuevo Byte
        if(RC1IF){
                RecDat();
        }
                LanDat();
        }
    }

//-----------------------------------------------------------
// Función Recepción de datos
//-----------------------------------------------------------
void RecDat(void){
    unsigned char dat;

    RC1IF = 0;          // Borrar el flag de recepción
    dat = RCREG1;       // Recuperar el byte del buffer de recepción


/*Analiza el primer bit recibido en la uart.
Comparando si es paro o marcha pone a 1 o a 0
la variable aux que es la que se encargara de
poner a 1 o 0 el puerto seleccionado posteriormente*/
        if (cond1==0)
        {
        switch(dat){
        // Poner a cero los bits de puerto 1
        case 'p':
                aux=0;
                cond1=1;
                break;
                case 'P':
                aux=0;
                cond1=1;        //Bit P1.1 a 0
                break;
        case 'M':
                aux=1;
                cond1=1;        //Bit P1.2 a 0
                break;
        case 'm':
                aux=1;
                cond1=1;        //Bit P1.3 a 0
                break;
    }
    }
/*Aqui le el dato de la uart que entre despues de un p
o una m que deve ser un numero, este numero indica el bit
del puerto de salida seleccionado, en caso aux 1 pondra on
en caso de aux 0 lo pondra of*/
        if (cond1==1)
        {
        if (dat=='0'||dat=='1'||dat=='2'||dat=='3'||dat=='4'||dat=='5'||dat=='6'||dat=='7')
```

```
        {
        switch(dat){
        // Poner a cero los bits de puerto 1
        case '0':
                RB0 = aux;      //Bit P1.0 a 0
                cond1=0;
                break;
        case '1':
                RB1 = aux;      //Bit P1.1 a 0
                cond1=0;
                break;
        case '2':
                RB2 = aux;
                cond1=0;        //Bit P1.2 a 0
                break;
        case '3':

                RB3 = aux;
                cond1=0;        //Bit P1.3 a 0
                break;
        case '4':
                RB4 = aux;
                cond1=0;        //Bit P1.4 a 0
                break;
        case '5':
                RB5 = aux;
                cond1=0;        //Bit P1.5 a 0
                break;
        case '6':
                RD2 = aux;
                cond1=0;        //Bit P1.6 a 0
                break;
        case '7':
                RD3 = aux;
                cond1=0;        //Bit P1.7 a 0
                break;
        }
    }
/*Este fragmento de codigo orienta al programa.
si el dato en la uart es una p o una m nos dirigira
al segundo switch de seleccion del bit del puerto
si no nos dirigira al primero esto lo realiza medaiante
la variable cond1
este algoritmo permite discriminar los datos que
no sean validos para el programa o no esperados
por lo tanto cuando llega un dato no esperado
hay que volver a recivir los dos bits seguidos
para realizar la accion*/
        else if(dat=='P'||dat=='p'||dat=='M'||dat=='m' )
        {
        cond1=1;
        }
        else
        {
        cond1=0;
        }
        }//
     }

//-------------------------------------------------------------
// Iniciar el puerto serie
//-------------------------------------------------------------
void LanDat(void){
/*En este fragmento verificamos el estado de las entradas
con el estado anterior de las entradas, cuando se produce
un cambio y lanzamos
ej: RH0 0->1 lanzamos "N0"
```

```
RH0 1-> lanzamos "F0"
a->h => estados anteriores
RH0 => estados actuales
Una vez lanzado el dato memorizamos el estado de las entradas
llamando a init port*/
if (a!=RH0)
        {
        if(RH0==0){
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='F';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='0';
        }
                if(RH0==1){
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='N';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='0';
                }
        }
else if (b!=RH1)
        {
                if(RH1==0){
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='F';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='1';
                }
        if(RH1==1){
                //Wait for TXREG Buffer to become available
                        while(!TX1IF);
                //Write data
                TXREG='N';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='1';
                }

               InitPort();
        }
        else if (c!=RH2)
        {
                if(RH2==0){
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='F';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='2';
                        }
        if(RH2==1){
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
```

```
                //Write data
                TXREG='N';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='2';
                }
        InitPort();
        }
else if (d!=RH3)
        {
        if(RH3==0){
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='F';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='3';
                }
        if(RH3==1){
                while(!TX1IF);
                //Write data
                TXREG='N';
                //Wait for TXREG Buffer to become available
                while(!TX1IF);
                //Write data
                TXREG='2';
        }
        InitPort();
}
else if (e!=RJ0)

        {
        if(RJ0==0){
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='F';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='4';
        }
        if(RJ0==1){
        while(!TX1IF);
        //Write data
        TXREG='N';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='4';
        }
        InitPort();
        }
else if (f!=RJ1)
        {
        if(RJ1==0){
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='F';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='5';
        }
```

```
        if(RJ1==1){
        while(!TX1IF);
        //Write data
        TXREG='N';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='5';
        }
        InitPort();
        }
else if (g!=RJ2)
        {
        if(RJ2==0){
        //Wait for TXREG Buffer to become available
        while(!TX1IF);

        //Write data
        TXREG='F';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='6';
        }
        if(RJ2==1){
        while(!TX1IF);
        //Write data
        TXREG='N';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='6';
        }
        InitPort();
}
else if (h!=RJ3)
        {
        if(RH1==0){
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='F';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='7';
        }
        if(RJ3==1){
        while(!TX1IF);
        //Write data
        TXREG='N';
        //Wait for TXREG Buffer to become available
        while(!TX1IF);
        //Write data
        TXREG='7';
        }
        InitPort();
}
}
//-----------------------------------------------------------
// Iniciar el puerto serie
//-----------------------------------------------------------
/*Como se mencionava anteriormente una vez lanzado el dato
igualamos el estado entareior de las entradas con el actual*/

void InitPort(void){
        a=RH0;
```

```
        b=RH1;
        c=RH2;
        d=RH3;
        e=RJ0;
        f=RJ1;
        g=RJ2;
        h=RJ3;
}
//------------------------------------------------------------
// Iniciar el puerto serie
//------------------------------------------------------------
void InitSerial(void){

//Baud Rate = 9600 Bits per Second
//*** Note: Valid On 25MHz Crystal ONLY ***
//For other crystal freq calculate new values for SPBRG
    SPBRGH1=0x02;
    SPBRG1=0x8B;
    TXEN1=1;
    BRGH1=1;
    SPEN1=1;
    RCEN1=1;
    BRG161=1;

}
```