```
1: //              ++++       IDENTIFICADOR DE PRODUCTOS  +++++++
2: /*
3:  * Descripción:
4:      El presente código corresponde a la aplicación que permite tarifar
5:      productos en base a los códigos de barras a través de un lector tipo
6:      pistola con una comunicación inalámbrica a una base de datos externa.
7:      Además, opera con una pantalla GLCD basada en un controlador TOSHIBA
8:      T6963C y una comunicación inalámbrica a 2.4 GHz. por medio de los
9:      módulos XBEE basados en el protocolo de comunicación IEEE 802.15.4.
10:     Para la comunicación se utilizan  dos puertos seriales UART, uno
11:     para el lector de barras y el otro para el XBEE. Los mandos de función
12:     se efectúan a través de las interrupciones externas.
13:
14:  * Configuración:
15:     MCU:             dsPIC30F4013
16:     Oscilador:       XT-PLL8, 12.000MHz
17:     Modulos Ext.:    mE T6369C board,pantalla GLCD T6369C 240x128 pixels,
18:                      XBEE PRO, Lector de barras tipo pistola
19:     Software:        mikroC for dsPIC30/33 and PIC24 v4.0.0.0
20:  */
21:
22: #include "T6963C.h"             // Librería  LCD gráfica
23: /*
24:  * Imágenes .bmp monocromáticas almacenadas en la ROM
25:  */
26: extern  const   char coche1[] ;
27: extern  const   char coche2[] ;
28: extern  const   char coche3[] ;     // Definición matriz imágenes
29: extern  const   char oferta[] ;
30: extern  const   char tar_fact[] ;
31: extern  const   char ofer_special[] ;
32: extern  const   char ofer_compras[] ;
33:
34: char Trama_Central[70];          // buffer para recepción de datos de la BDD
35: unsigned Trama_Pistola[31];     // buffer para recibir código de barras
36: char rx1;
37: char rx2;                       // variables para UART
38: char dir[3];                    // vector Dirección Coche
39: int indice_code=0;
40: int indice_pistola=5;        // referencia de posición para recolecion de datos
41: int band_uart2, band_uart1;  // banderas activación función interrupciones UART
42:
43: int ini_Name, ini_Precio, ini_Total, ini_Dir;
44: int fin_Name, fin_Precio, fin_Total, fin_Dir;  // Auxiliares detección tamaño
45: int tam_Name, tam_Precio, tam_Total, tam_Dir;
46:
47: int cont_Dir=0;
48: int lenght_Dir=0;
49: int cont_Name=0;
50: int lenght_name=0;
51: int cont_Precio=0;             // variables para escribir datos en GLCD
52: int lenght_precio=0;
53: int cont_Total=0;
54: int lenght_total=0;
55:
56: int cont_regProducto=9;        // registro de productos en la GLCD
57: int ini_Fact;                  // variable inicializar factura o Venta BDD
58: int ofer=0;                    // auxiliar activación imágenes ofertas
59:
60:  /// PANTALLA GLCD
61:     unsigned char   panel ;         // panel actual
62:     unsigned int    i ;             // registro de propósito general
```

```
63:      unsigned char   curs ;          // visibilidad del cursor
64:      unsigned int    cposx, cposy ;  // posición  x-y cursor
65: //////
66: void Ini_GLCD ()
67: {
68:      /*
69:      * Inicializar pantalla de 240 pixel de ancho y 128 pixel de alto
70:      * 8 bits de longitud bus de datos
71:      * Bus de datos en PORTB
72:      * Bus de control en PORTD
73:      * !WR es bit 0
74:      * !RD es bit 1
75:      * !CD es bit 2
76:      * RST es bit 3
77:      */
78:      T6963C_cntlwr=PORTD.F0;
79:      T6963C_cntlrd=PORTD.F1;
80:      T6963C_cntlcd=PORTD.F2;
81:      T6963C_cntlrst=PORTD.F3;
82:      T6963C_init(240, 128, 8, &PORTB, &PORTD, 0, 1, 2, 3) ;
83:
84:      T6963C_graphics(1) ;
85:      T6963C_text(1) ;      // habilitar gráficos y texto al mismo tiempo
86:      panel = 0 ;
87:      i = 0 ;
88:      curs = 0 ;
89:      cposx = cposy = 0 ;
90: }
91:
92: void Ini_Factura ()  // Subrutina de diseño pantalla para descripción producto
93: {
94:       T6963C_rectangle(0, 0, 239, 127, T6963C_WHITE);
95:       T6963C_sprite(0, 0, tar_fact, 240, 63) ;
96:       T6963C_write_text("DETALLE", 6,  8, T6963C_ROM_MODE_XOR) ;
97:       T6963C_write_text("PRECIO",  21, 8, T6963C_ROM_MODE_XOR) ;
98:       T6963C_write_text(" TOTAL:", 8, 15, T6963C_ROM_MODE_XOR) ;
99: }
100:  /*
101:  * Configuración de los puertos seriales UART
102:  */
103: void Ini_UART()
104: {
105:      UART1_Init(2400);              // inicializa la Com. UART a 2400 bps
106:      UART2_Init(2400);
107:
108:      //////UART1
109:      U1STAbits.URXISEL = 0;
110:      INTCON1bits.NSTDIS = 1;       // no anidación de interrupciones
111:      IFS0bits.U1RXIF = 0;          // asegurar interrupciones no pendientes
112:      IEC0bits.U1RXIE = 1;          // habilitar interrupción
113:      U1MODEbits.UARTEN = 1;        // habilitar UART
114:      U1STAbits.UTXEN = 1;          // habilitar UART Tx
115:
116:      //////UART2
117:      U2STAbits.URXISEL = 0;
118:      INTCON1bits.NSTDIS = 1;       // no anidación de interrupciones
119:      IFS1bits.U2RXIF = 0;          // asegurar interrupciones no pendientes
120:      IEC1bits.U2RXIE = 1;          // habilitar interrupción
121:      U2MODEbits.UARTEN = 1;        // habilitar UART
122:      U2STAbits.UTXEN = 1;          // habilitar UART Tx
123:
124:      Delay_ms(100);            // Retardo para estabilizar el modulo UART
```

```
125:      band_uart2=0;             // Inicialización bandera función interrupción
126: }
127:  /*
128:  * Conformación trama de datos Coche para Código de barras.
129:  */
130: void Ini_Payload()   {
131:
132:      Trama_Pistola[0]='^';        // caracter inicio trama
133:      Trama_Pistola[1]='A';        // DIRECCIÓN C/COCHE PARA MODIFICAR
134:      Trama_Pistola[2]='0';
135:      Trama_Pistola[3]='2';
136:      Trama_Pistola[4]='^';        // caracter separación
137:      Trama_Pistola[5]=' ';        // Campos de datos
138:      Trama_Pistola[6]=' ';
139:      Trama_Pistola[7]=' ';
140:      Trama_Pistola[8]='^';
141:      Trama_Pistola[9]='^';
142:      Trama_Pistola[10]='^';
143:      Trama_Pistola[11]='^';
144:      Trama_Pistola[12]='^';
145:      Trama_Pistola[13]='^';
146:      Trama_Pistola[14]='^';
147:      Trama_Pistola[15]='^';
148:      Trama_Pistola[16]='^';
149:      Trama_Pistola[17]='^';
150:      Trama_Pistola[18]='^';
151:      Trama_Pistola[19]='^';
152:      Trama_Pistola[20]='^';
153:      Trama_Pistola[21]='^';
154:      Trama_Pistola[22]='^';
155:      Trama_Pistola[23]='^';
156:      Trama_Pistola[24]='^';
157:      Trama_Pistola[25]='^';
158:      Trama_Pistola[26]='^';
159:      Trama_Pistola[27]='^';
160:      Trama_Pistola[28]='^';
161:      Trama_Pistola[29]='^';
162:      Trama_Pistola[30]='^';     // Caracter "^" delimitador y/ó fin de trama
163: }
164: void Ini_TramaCentral()   // Función para limpieza datos de la PC
165: {
166:      for(i=0;i<71;i++) {
167:         Trama_Central[i]=' ';
168:      }
169: }
170: void Publicar()           // Función encargada de hacer petición de oferta
171: {
172:       Uart2_Write_Char('P');     // Caracter inicio factura
173:       Uart2_Write_Char(Trama_Pistola[0]);
174: }
175: void Ofertas()            // Función encargada de graficar una oferta en GLCD
176: {
177: int i;
178:          for(i=0;i<28;i++)
179:          {
180:            T6963C_write_char(Trama_Central[i+1], i+1, 4, T6963C_ROM_MODE_XOR);
181:            // Campo correspondiente al mensaje desde la plataforma de BDD.
182:          }
183:          /*
184:          * Publicación de imágenes pregabadas en el uC. para ofertas.
185:          */
186:          ofer=ofer+1;
```

```
187:          if( ofer==1) {
188:              T6963C_sprite(0, 0, ofer_special, 240, 63) ;
189:              }
190:         if( ofer==2) {
191:              T6963C_sprite(0, 0, ofer_compras, 240, 63) ;
192:              }
193:
194:          if( ofer==3){
195:              T6963C_sprite(0, 0, tar_fact, 240, 63) ;
196:              ofer=0;
197:              }
198:           indice_code=0;          // Inicializar posición trama plataforma BDD
199:           Ini_TramaCentral();     // Borrar Trama plataforma BDD
200: }
201:
202: void Inexistencia() {       // Función para indicar que no existe un producto.
203:
204: if (dir[0] == Trama_Pistola[1]&& dir[1] == Trama_Pistola[2] && dir[2] ==
     Trama_Pistola[3])
205:      {
206:          for(i=0;i<28;i++)
207:          {
208:            T6963C_write_char(Trama_Central[i+1], i+3, 4, T6963C_ROM_MODE_XOR);
209:            // escritura mensaje GLCD desde la plataforma de BDD.
210:          }
211:      }
212: }
213:
214: //-------------- Subrutina Interrupción UART1 Lector barras coche
215: void interrupt_uart1() org 0x26 {
216:
217:   Trama_Pistola[indice_pistola]=Uart1_Read_Char();  // Adquisición datos puerto
218:   if (ini_Fact==1)          // validación inicio venta
219:   {
220:   if (Trama_Pistola[indice_pistola]==0x0A) //Verificación caracter fin de datos
221:      {
222:          band_uart1=1;  // Activación bandera cuando se ha leído todo el puerto
223:      }
224:    indice_pistola++;
225:    }
226:   IFS0bits.U1RXIF = 0;    // Asegurar que la interrupción no esté pendiente
227: }
228:
229: //------------- Subrutina Interrupción UART2 XBEE-Central
230: void interrupt_uart2() org 0x44 {
231:
232:   Trama_Central[indice_code]=Uart2_Read_Char();  // Adquisición datos puerto
233:   if(Trama_Central[indice_code]==']')     // "]" Caracter fin de datos
234:   {
235:      band_uart2=1;    // Activación bandera cuando se ha leído todo el puerto
236:   }
237:   if( Trama_Central[indice_code]==';' && ini_Fact==1)
238:                                        // Verificación trama publicidad
239:   {
240:      Ofertas();           // llamado subrutina Ofertas
241:   }
242:   if( Trama_Central[indice_code]=='X' && ini_Fact==1)
243:                                        // Verificación trama publicidad
244:   {
245:      //Inexistencia();   // llamado a función para indicar que no existe
     producto
246:   }
```

```
247:   indice_code++;
248:   IFS1bits.U2RXIF = 0;    // Asegurar que la interrupción no esté pendiente
249: }
250:
251: void Direccion() {      // Subrutina cálculo tamaño Dirección Coche.
252:     int i=0;
253:      for (i=0;i<indice_code;i++)
254:      {
255:          if(Trama_Central[i]=='+')
256:          {
257:            ini_Dir=i;
258:          }
259:          else if(Trama_Central[i]=='^')
260:          {
261:            fin_Dir=i;
262:          }
263:      }
264:      tam_Dir=fin_Dir-ini_Dir-1;
265: }
266:
267: void Nombre()  {        // Subrutina cálculo tamaño Detalle producto
268:     int i=0;
269:      for (i=0;i<indice_code;i++)
270:      {
271:          if(Trama_Central[i]=='^')
272:          {
273:            ini_Name=i;
274:          }
275:          else if(Trama_Central[i]=='_')
276:          {
277:            fin_Name=i;
278:          }
279:      }
280:      tam_Name=fin_Name-ini_Name-1;
281: }
282:
283: void Precio() {         // Subrutina cálculo tamaño Precio producto
284:     int i=0;
285:      for (i=0;i<indice_code;i++)
286:      {
287:          if(Trama_Central[i]=='_')
288:          {
289:            ini_Precio=i;
290:          }
291:          else if(Trama_Central[i]=='[')
292:          {
293:            fin_Precio=i;
294:          }
295:      }
296:      tam_Precio=fin_Precio-ini_Precio-1;        // -1 x espacio libre q se
    visualiza como basura
297: }
298:
299: void Pre_Total()   {    // Subrutina cálculo tamaño valor Total compra
300:     int i=0;
301:      for (i=0;i<indice_code;i++)
302:      {
303:          if(Trama_Central[i]=='[')
304:          {
305:            ini_Total=i;
306:          }
307:          else if(Trama_Central[i]==']')
```

```
308:          {
309:            fin_Total=i;
310:          }
311:      }
312:      tam_Total=fin_Total-ini_Total-1;
313:      T6963C_write_Text("               ",46,14, T6963C_ROM_MODE_XOR) ;
314: }                                  //Permite limpiar el espacio GLCD P_Total
315:
316: void Venta(){   // Subrutina que inicializa una compra cuando se activa la INT1
317:
318:       ini_GLCD();                // Requerido para una nueva escritura GLCD
319:       Ini_Factura();             // Llamado a subrutina para diseño en GLCD
320:       Uart2_Write_Char('@');     // Caracter inicio factura
321:       Uart2_Write_Char(Trama_Pistola[0]);
322:       Uart2_Write_Char(Trama_Pistola[1]);   // Dirección Coche
323:       Uart2_Write_Char(Trama_Pistola[2]);
324:       Uart2_Write_Char(Trama_Pistola[3]);
325:       Uart2_Write_Char(Trama_Pistola[4]);
326:       T6963C_write_text("  Gracias por preferirnos  ", 1, 4,
    T6963C_ROM_MODE_XOR) ;
327:       ini_Fact=1;
328:       cont_regProducto=9;
329:       Ini_TramaCentral();       // Borrar Trama plataforma BDD
330: }
331:
332: void Eliminar() { //Subrutina que desglosa un producto cuando se activa la INT0
333:    if (ini_Fact==1)             // Validación de que se haya iniciado la compra
334:    {
335:    T6963C_write_text("Deslizar producto a eliminar", 1, 4,T6963C_ROM_MODE_XOR);
336:      Uart2_Write_Char('*');     // Caracter para desglose de compra
337:      Uart2_Write_Char(Trama_Pistola[0]);
338:      Uart2_Write_Char(Trama_Pistola[1]);   // Dirección Coche
339:      Uart2_Write_Char(Trama_Pistola[2]);
340:      Uart2_Write_Char(Trama_Pistola[3]);
341:      Uart2_Write_Char(Trama_Pistola[4]);
342:     }
343: }
344: ////
345: void Interrupt0_DOWN() org 0x0014{ // Interrupción Externa en INT0
346:    LATB.F10=1;          // Activar pin Led
347:    DELAY_MS(200);
348:
349:    Eliminar();         // LLamado subrutina para desglosar un producto
350:    IFS0.F0 = 0;        // bandera de interrupción despejada
351:
352: }
353: void Interrupt1_UP() org 0x0034{  // Interrupción Externa en INT1
354:      LATB.F10=1;       // Activar pin Led
355:      DELAY_MS(300);
356:      Venta();          // llamado subrutina para iniciar una compra
357:      IFS1.F0 = 0;      // bandera de interrupción despejada
358: }
359: void Interrupt2_CAN() org 0x0042{ // Interrupción Externa en INT2
360:      LATB.F10=1;       // Activar pin Led
361:      DELAY_MS(200);
362:      Publicar();       // llamado subrutina para visualizar una oferta
363:      IFS1.F7 = 0;      // bandera de interrupción despejada
364: }
365:
366: void main(void)             // PROGRAMA PRINCIPAL
367:         {
368:         ADPCFG = 0xFFFF;           // inicializa los pines AN como digitales
```

```
369:          TRISD = 0x0F;
370:          TRISF = 0 ;                 // Config. PORTF como salida
371:          PORTF = 0b00000000 ;        // habilitar bits puerto
372:          TRISB = 0;                  // Config. PORTB como salida
373:          /////
374:          IFS0 = 0;        // bandera interrup despejada
375:          IFS1 = 0;
376:          IEC0 = 1;        // habilitar INT0
377:          IEC1.F0 = 1;     // habilitar INT1
378:          IEC1.F7 = 1;     // habilitar INT2
379:
380:          INTCON2.F2  = 0;      // SELECCIONAR FLANCO L_TO_H PARA INT2
381:          INTCON2.F1  = 0;      // SELECCIONAR FLANCO L_TO_H PARA INT1
382:          IPC4.F5 = 0b00000001;      //  PRIORIDAD INTERRUPCION INT2 ES 1
383:          IPC4.F1 = 0b00000011;      //  PRIORIDAD INTERRUPCION INT1 ES 3
384:          IPC0.F0 = 0b00000111;      //  PRIORIDAD INTERRUPCION INT0 ES 7
385:
386:          Ini_GLCD();       // Inicializar GLCD
387:          Ini_UART();       // inicializar puertos seriales UART
388:          Ini_Payload();    // inicializar trama de datos
389:          ini_Fact=0;
390:
391:                  //      GLCD Presentación proyecto inicio
392:          T6963C_write_text(" TRABAJO DE GRADO ", 5, 5, T6963C_ROM_MODE_XOR) ;
393:          T6963C_write_text(" ELECTRONICA Y REDES ", 4, 7, T6963C_ROM_MODE_XOR) ;
394:          T6963C_write_text(" FICA - UTN ", 8, 9, T6963C_ROM_MODE_XOR) ;
395:          T6963C_box(0, 30, 239, 88, T6963C_WHITE) ;
396:          delay_ms(600);
397:          Ini_GLCD();
398:          T6963C_image(coche3) ;    // Transición de imagenes
399:          delay_ms(700);
400:          T6963C_image(coche1) ;
401:          delay_ms(700);
402:          T6963C_image(coche2) ;
403:
404:        while (1)             // BUCLE INFINITO
405:        {
406:           LATB.F10=0;        // Desactivar pin Led
407:        if(band_uart1==1)     // Bandera correspondiente a la UART1
408:          {                   // TRANSMISIÓN TRAMA DEL COCHE por UART2
409:               Uart2_Write_Char(Trama_Pistola[0]);
410:               Uart2_Write_Char(Trama_Pistola[1]);
411:               Uart2_Write_Char(Trama_Pistola[2]);
412:               Uart2_Write_Char(Trama_Pistola[3]);
413:               Uart2_Write_Char(Trama_Pistola[4]);
414:               Uart2_Write_Char(Trama_Pistola[5]);
415:               Uart2_Write_Char(Trama_Pistola[6]);
416:               Uart2_Write_Char(Trama_Pistola[7]);
417:               Uart2_Write_Char(Trama_Pistola[8]);
418:               Uart2_Write_Char(Trama_Pistola[9]);
419:               Uart2_Write_Char(Trama_Pistola[10]);
420:               Uart2_Write_Char(Trama_Pistola[11]);
421:               Uart2_Write_Char(Trama_Pistola[12]);
422:               Uart2_Write_Char(Trama_Pistola[13]);
423:               Uart2_Write_Char(Trama_Pistola[14]);
424:               Uart2_Write_Char(Trama_Pistola[15]);
425:               Uart2_Write_Char(Trama_Pistola[16]);
426:               Uart2_Write_Char(Trama_Pistola[17]);
427:               Uart2_Write_Char(Trama_Pistola[18]);
428:               Uart2_Write_Char(Trama_Pistola[19]);
429:               Uart2_Write_Char(Trama_Pistola[20]);
430:               Uart2_Write_Char(Trama_Pistola[21]);
```

```
431:              Uart2_Write_Char(Trama_Pistola[22]);
432:              Uart2_Write_Char(Trama_Pistola[23]);
433:              Uart2_Write_Char(Trama_Pistola[24]);
434:              Uart2_Write_Char(Trama_Pistola[25]);
435:              Uart2_Write_Char(Trama_Pistola[26]);
436:              Uart2_Write_Char(Trama_Pistola[27]);
437:              Uart2_Write_Char(Trama_Pistola[28]);
438:              Uart2_Write_Char(Trama_Pistola[29]);
439:              Uart2_Write_Char(Trama_Pistola[30]);
440:         band_uart1=0;          // Reseteo bandera
441:         indice_pistola=5;      // Posición para escritura de datos en la Trama
442:         Ini_Payload();         // Borrar Trama coche
443:
444:         }
445:            if(band_uart2==1)   // Bandera correspondiente a la UART2
446:            {
447:             if(cont_regProducto==14) // Comprueba la última posición
448:                 {
449:                 Publicar();
450:                 ini_GLCD();          // Limpieza y reinicio GLCD
451:                 Ini_Factura();
452:                 cont_regProducto=9;
453:                 }
454:                 Direccion();         // Subrutina para obtener tamaño Dirección
455:                 i=0;
456:                 for(cont_Dir=ini_Dir+1;cont_Dir<=ini_Dir+tam_Dir;cont_Dir++)
457:                 {
458:                     dir[i]=  Trama_Central[cont_Dir];
459:                     lenght_dir++;    // Deducción Dirección Coche
460:                     i++;
461:                 }
462:              if (dir[0] == Trama_Pistola[1]&& dir[1] == Trama_Pistola[2] &&
     dir[2] == Trama_Pistola[3])
463:                {
464:                 Nombre();            // Subrutina para obtener tamaño Detalle
465:
     for(cont_Name=ini_Name+1;cont_Name<ini_Name+tam_Name;cont_Name++)
466:                 {   // Escritura Detalle en la GLCD
467:                     T6963C_write_char(Trama_Central[cont_Name],1+lenght_name,
     cont_regProducto, T6963C_ROM_MODE_OR) ;
468:                     lenght_name++;
469:                 }
470:
471:                 Precio();            // Subrutina para obtener tamaño Precio
472:
     for(cont_Precio=ini_Precio+1;cont_Precio<ini_Precio+tam_Precio;cont_Precio++)
473:                 {   // Escritura Precio en la GLCD
474:                     T6963C_write_char(Trama_Central[cont_Precio],
     22+lenght_precio,cont_regProducto, T6963C_ROM_MODE_XOR) ;
475:                     lenght_precio++;
476:                 }
477:
478:                 Pre_Total();   // Subrutina para obtener tamaño Total de compra
479:
     for(cont_Total=ini_Total+1;cont_Total<ini_Total+tam_Total;cont_Total++)
480:                 {   // Escritura Valor Total compra en la GLCD
481:                     T6963C_write_char(Trama_Central[cont_Total],46+lenght_total,
     14, T6963C_ROM_MODE_XOR) ;
482:                     lenght_total++;
483:                 }
484:                 lenght_name=0;
485:                 lenght_precio=0;        // Reinicio variables
```

```
486:                 lenght_total=0;
487:                 cont_regProducto= cont_regProducto + 1;  // Incremento registro
488:               }
489:                 indice_code=0;       // Reinicio variables
490:                 band_uart2=0;
491:                 Ini_TramaCentral();  // Borra la Trama de la plataforma de BDD
492:             }
493:       }
494: }
495:
496:
```