UNIVERSIDAD CATÓLICA ANDRÉS BELLO
FACULTAD DE INGENIERÍA
ESCUELA DE TELECOMUNICACIONES

"DISEÑO DE UN SEMÁFORO INALÁMBRICO, SOPORTADO POR UN SUBSISTEMA DE LA PLATAFORMA DE CONTROL DE TRÁFICO PISACOTA"

ANEXOS

TRABAJO ESPECIAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN TELECOMUNICACIONES

REALIZADO POR Marian Gómez
Carolina Mas

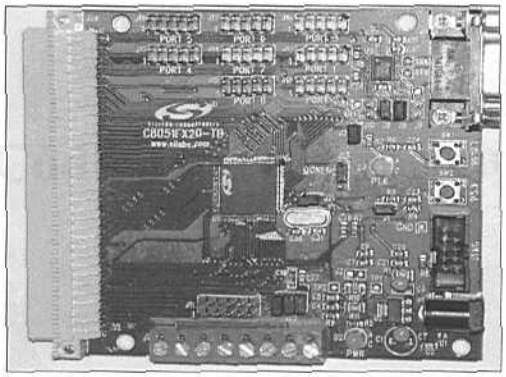
PROFESOR GUÍA Iñaki Mendizabal

FECHA Caracas, 21 de febrero del 2007

Apéndice A
kit C8051F12x-DK

El Kit C8051F12x-DK de la empresa *Siliton Laboratories* está compuesto por:

- Un circuito circuito impreso, con un dispositivo C8051F120 incluido, interfaces que conectan los puertos, para conexiones externas. (Figura 1(a))
- Las herramientas necesarias para realizar la inclusión del software y la emulación del dispositivo cuando esté operando, mostradas en la figura 1 (b)
- El ambiente de desarrollo *Silicon Laboratories Integrated Development Environment (IDE),* cuya pantalla principal se observa *en* la Figura 2.



(a)

(b)

Figura 1.- Elementos físicos del kit C8051F12x-DK: (a) Tarjeta para la programación del microcontrolador C8051F120, (b) Conector JTAG con adaptador USB.

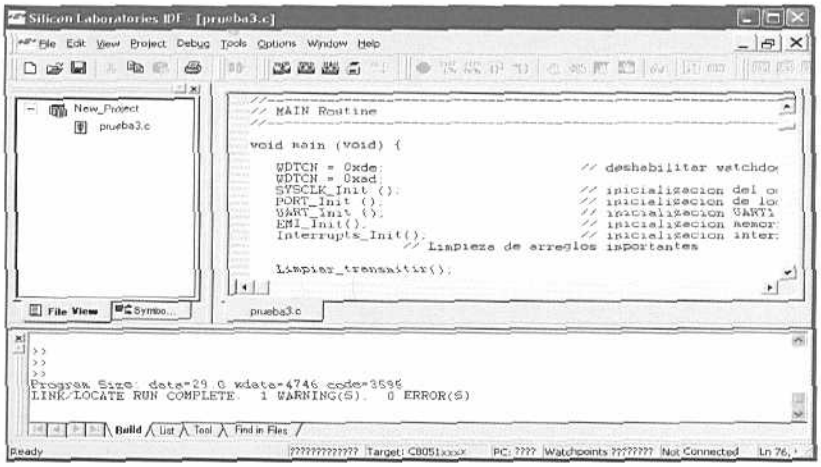


Figura 2.- Ambiente de desarrollo Silicon Labs Integrated Development (IDE).

Apéndice B
Código del Microcontrolador C8051F120

```
//-----------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------
#include <c8051f120.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//-----------------------------------------------------------------------
// CONSTANTES Globales
//-----------------------------------------------------------------------
#define BAUDRATE    115200      // Baud rate of UART in bps
#define INTCLK      24500000    // Internal oscillator frequency en Hz
#define SYSCLK      49000000    // Salida del PLL (INTCLK*2)
sbit at 0x96   LED='1';         // green LED: '1' = ON; '0' = OFF
sbit at 0x97   DISC='0';        //sbit at 0x

//-----------------------------------------------------------------------
// Funciones
//-----------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void UART_Init (void);
void EMI_Init(void);
void Interrupts_Init(void);
void Limpiar_transmitir(void);
void Limpiar_monitoreo(void);
void Limpiar_alarma(void);
void Limpiar_peaton(void);
void UART1_ISR (void) interrupt 20;
void Timer3_ISR (void) interrupt 14;
void UART0_ISR (void) interrupt 4;
void wait_ms (int ms);
void Limpiar_receptor(void);
void Limpiar_resp(void);
void Monitorear(void);
void Conectarse(void);
void Tramas_UCS(void);
void Desconectar(void);

//-----------------------------------------------------------------------
// VARIABLES Globales
//-----------------------------------------------------------------------
 unsigned char xdata malloc_mempool [0x1000];
 int i;int p=0;  int xdata tipo[12];  int xdata vel[12];  int xdata tot[12];
```

```
 int xdata cont[12];// la variable receptor
 int xdata receptor[7000]; int xdata transmitir[18]; long tim5min=0; long
tim1hora=0;
 int datos; int xdata acum2=0; int acum=0;int xdata b;int xdata k=0;int mandar=0;
int xdata resp[76];
  int xdata alarma[9];int xdata peaton[9];  int xdata monitoreo[9]; int xdata ack[9];
  int xdata sema1[20];int xdata sema2[20];int xdata sema3[20];int xdata
sema4[20];
  //Banderas
int xdata m=0;int h=0;int xdata y;int xdata q;int xdata d=1;int xdata c=1; int xdata
s=0;
int xdata check=1;  long xdata a=0; int xdata estado=1;int xdata intento=0;
//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------

void main (void) {

   WDTCN = 0xde;          // deshabilitar watchdog timer
   WDTCN = 0xad;
   SYSCLK_Init ();        // inicializacion del oscilador
   PORT_Init ();          // inicializacion de los puertos
   UART_Init ();          // inicializacion UART1
   Interrupts_Init();
   m=1;
   Tramas_UCS();          // inicializacion interrupciones
   // Limpieza de arreglos importantes
  Limpiar_transmitir();
 Limpiar _alarma ();
 Limpiar_monitoreo();
  Limpiar_peaton();
  Limpiar_resp();
  Limpiar_receptor();

  EA = 1;  // Habilitar las interrupciones globalmente

DISC=!DISC;
while (check=1)   //verificar si han activado la señal de alarma o paso de peatones
{
wait_ms(3000);
if
((alarma[3]==0xFF)||(peaton[3]=='H')||(peaton[4]=='I')||(peaton[5]=='J')||(peaton[6
]=='K'))
{
acum=0;
Conectarse();
```

```
tim5min=0;
Desconectar();
Tramas_UCS();
Limpiar_peaton();
Limpiar _alarma ();
}
}
}
//-----------------------------------------------------------------------
// Initialization Subroutines
//-----------------------------------------------------------------------

//-----------------------------------------------------------------
//Conectarse
//-----------------------------------------------------------------

void Conectarse (void)
{

//while (1) {

 LED = !LED; // cambio del estado del LED

  Limpiar_transmitir();
  Limpiar_receptor();
  Limpiar_monitoreo();
  Monitorear();
  wait_ms(400);
  if      ((monitoreo[3]=='128')&       (monitoreo[4]=='64')&(monitoreo[5]=='32')
&(monitoreo[6]=='16'))
 { m=0;
  }
 else
 { m=1;
 }
  //---------------------- Fin del Monitoreo--------------------------------//

  LED ='1'; // LED encendido cuando se establezca la secuencia de envio
  tim5min=0;

  Limpiar_transmitir();
   p=0;
        EA=0;  SFRPAGE = UART0_PAGE;
        //Secuencia de envio del comando AT+CRM=130
         TI0=1;SBUF0=0x41;TI0=0;wait_ms(1);TI0=1;SBUF0=0x54;TI0=0;
         wait_ms(1);
```

```
  TI0=1;SBUF0=0x2B;TI0=0;wait_ms(1); TI0=1;SBUF0=0x43;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x52;TI0=0;wait_ms(1);TI0=1;SBUF0=0x4D;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x3D;TI0=0;wait_ms(1);TI0=1;SBUF0=0x31;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x33;TI0=0;wait_ms(1);TI0=1;SBUF0=0x30;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x0D; TI0=0;wait_ms(1);TI0=1;SBUF0=0x0A;TI0=0;
wait_ms(1);

SFRPAGE = CONFIG_PAGE;
  TI0=0;EA=1;wait_ms(100);
if ((resp[0]==0x0D) & (resp[1]==0x0A) & (resp[2]==0x4F) &
(resp[3]==0x4B))  //respuesta OK
{

p=0;EA=0;SFRPAGE = UART0_PAGE;
//Secuencia de envio del comando AT*PID="5176360@cantv.net"
TI0=1;SBUF0=0x41;TI0=0;wait_ms(1);TI0=1;SBUF0=0x54;TI0=0;
 wait_ms(1);
TI0=1;SBUF0=0x2A;TI0=0; wait_ms(1);TI0=1;SBUF0=0x50;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x49;TI0=0;wait_ms(1);TI0=1;SBUF0=0x44;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x3D;TI0=0;wait_ms(1);TI0=1;SBUF0=0x22;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x35;TI0=0;wait_ms(1);TI0=1;SBUF0=0x31;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x37;TI0=0;wait_ms(1);TI0=1;SBUF0=0x36;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x33;TI0=0;wait_ms(1); TI0=1;SBUF0=0x36;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x30; TI0=0;wait_ms(1);TI0=1;SBUF0=0x40; TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x63; TI0=0;wait_ms(1);TI0=1;SBUF0=0x61;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x6E;TI0=0;wait_ms(1);TI0=1;SBUF0=0x74;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x76;TI0=0;wait_ms(1);TI0=1;SBUF0=0x2E;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x6E;TI0=0;wait_ms(1);TI0=1;SBUF0=0x65;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x74;TI0=0;wait_ms(1);TI0=1;SBUF0=0x22;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x0D; TI0=0;wait_ms(1);TI0=1;SBUF0=0x0A;TI0=0;
```

```
wait_ms(1);

SFRPAGE = CONFIG_PAGE;
TI0=0; EA=1; wait_ms(100);
p=0;
if ((resp[0]==0x0D) & (resp[1]==0x0A) & (resp[2]==0x4F) &
(resp[3]==0x4B))//resp OK
{
p=0;   EA=0;  SFRPAGE = UART0_PAGE;
//Secuencia de envio del comando AT*PPW="23562"
  TI0=1;SBUF0=0x41;TI0=0;wait_ms(1);TI0=1;SBUF0=0x54;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x2A; TI0=0;wait_ms(1);TI0=1;SBUF0=0x50;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x50;TI0=0;wait_ms(1);TI0=1;SBUF0=0x57;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x3D; TI0=0;wait_ms(1);TI0=1;SBUF0=0x22;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x32;TI0=0;wait_ms(1);TI0=1;SBUF0=0x33;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x35;TI0=0;wait_ms(1);TI0=1;SBUF0=0x36; TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x32;TI0=0;wait_ms(1);TI0=1;SBUF0=0x22;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x0D; TI0=0;wait_ms(1);TI0=1;SBUF0=0x0A;TI0=0;
wait_ms(1);

SFRPAGE = CONFIG_PAGE;
TI0=0; EA=1; wait_ms(100);
p=0;
if ((resp[0]==0x0D) & (resp[1]==0x0A) & (resp[2]==0x4F) &
(resp[3]==0x4B))//resp OK
{
p=0;EA=0; SFRPAGE = UART0_PAGE;
//Secuencia de envio del comando AT+DIP= "xxx.xxx.xxx.xxx"
  TI0=1;SBUF0=0x41;TI0=0; wait_ms(1);TI0=1;SBUF0=0x54;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x2B;TI0=0;wait_ms(1);TI0=1;SBUF0=0x44;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x49;TI0=0;wait_ms(1);TI0=1;SBUF0=0x50;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x3D; TI0=0;wait_ms(1);TI0=1;SBUF0=0x22;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x32;TI0=0;wait_ms(1);TI0=1;SBUF0=0x30;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x30;TI0=0;wait_ms(1);TI0=1;SBUF0=0x2E; TI0=0;
```

```
wait_ms(1);
  TI0=1;SBUF0=0x30;TI0=0;wait_ms(1);TI0=1;SBUF0=0x34;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x34;TI0=0;wait_ms(1);TI0=1;SBUF0=0x2E; TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x30;TI0=0;wait_ms(1);TI0=1;SBUF0=0x38;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x37;TI0=0;wait_ms(1);TI0=1;SBUF0=0x2E;TI0=0;
 wait_ms(1);
   TI0=1;SBUF0=0x30;TI0=0;wait_ms(1);TI0=1;SBUF0=0x33;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x32;TI0=0;wait_ms(1); TI0=1;SBUF0=0x22;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x0D;TI0=0;wait_ms(1);TI0=1;SBUF0=0x0A; TI0=0;
wait_ms(1);

SFRPAGE = CONFIG_PAGE;
TI0=0; EA=1; wait_ms(100);p=0;
if ((resp[0]==0x0D) & (resp[1]==0x0A) & (resp[2]==0x4F) &
(resp[3]==0x4B))//ok
{
p=0;EA=0;SFRPAGE = UART0_PAGE;
//Secuencia de envio del comando AT+DPORT= "4000"

  TI0=1;SBUF0=0x41;TI0=0;wait_ms(1);TI0=1;SBUF0=0x54;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x2B;TI0=0;wait_ms(1);TI0=1;SBUF0=0x44;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x50;TI0=0;wait_ms(1);TI0=1;SBUF0=0x4F;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x52;TI0=0;wait_ms(1);TI0=1;SBUF0=0x54;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x3D;TI0=0; wait_ms(1);TI0=1;SBUF0=0x22;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x34;TI0=0;wait_ms(1);TI0=1;SBUF0=0x30;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x30;TI0=0;wait_ms(1);TI0=1;SBUF0=0x30;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x22;TI0=0;wait_ms(1);TI0=1;SBUF0=0x0D;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x0A; TI0=0;wait_ms(1);
SFRPAGE = CONFIG_PAGE;
TI0=0; EA=1; wait_ms(100);
p=0;
if ((resp[0]==0x0D) & (resp[1]==0x0A) & (resp[2]==0x4F) &
(resp[3]==0x4B))//ok
```

```
{
p=0;EA=0; SFRPAGE = UART0_PAGE;
// Secuencia de envio del comando ATDT123
  TI0=1;SBUF0=0x41;TI0=0;wait_ms(1); TI0=1;SBUF0=0x54;TI0=0;
 wait_ms(1);
  TI0=1;SBUF0=0x44; TI0=0;wait_ms(1);TI0=1;SBUF0=0x54;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x31;TI0=0;wait_ms(1);TI0=1;SBUF0=0x32;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x33;TI0=0;wait_ms(1);TI0=1;SBUF0=0x0D;TI0=0;
wait_ms(1);
  TI0=1;SBUF0=0x0A;TI0=0;wait_ms(1);
SFRPAGE = CONFIG_PAGE;
TI0=0; EA=1; wait_ms(15000);p=0;
if ((resp[0]==0x43) & (resp[1]==0x41) & (resp[2]==0x4C) &
(resp[3]==0x4C)
& (resp[4]==0x0D) & (resp[5]==0x0A) & (resp[6]==0x50) &
(resp[7]==0x50)
& (resp[8]==0x50) & (resp[9]==0x0D) & (resp[10]==0x0A) &
(resp[11]==0x43)
& (resp[12]==0x4F) & (resp[13]==0x4E) &(resp[14]==0x4E) &
(resp[15]==0x45) & (resp[16]==0x43) & (resp[17]==0x54))
//CALL PPP CONNECT
{
p=0;SFRPAGE = CONFIG_PAGE;EA=0;SFRPAGE = UART0_PAGE;
Limpiar_resp();

//*************** Tramas de subida *************//

monitoreo[0]='#';monitoreo[1]='1';monitoreo[2]='0';monitoreo[7]='#';h=0;
while(h<8)
{SBUF0=monitoreo[h];//Enviar trama monitoreo a la Central
TI0=0; wait_ms(100);h=h+1;p=0;
}
alarma[0]='#'; alarma[1]='1'; alarma[2]='1'; alarma[4]='#'; h=0;
while(h<4)
{SBUF0=alarma[h];//Enviar trama alarma a la Central
TI0=0; wait_ms(100);h=h+1;p=0;
}
peaton[0]='#'; peaton[1]='0'; peaton[2]='2'; peaton[7]='#'; h=0;
while(h<8)
{SBUF0=peaton[h];//Enviar trama peaton a la Central
TI0=0; wait_ms(100);h=h+1;p=0;
}
p=0;
Limpiar_receptor();
```

```
        Limpiar_resp();
        SFRPAGE = CONFIG_PAGE;
        TI0=0; EA=1; wait_ms(30000);
        }
        else
        {p=0;  DISC=!DISC; wait_ms(1000);
        DISC=!DISC; LED ='0';wait_ms(4000); intento=intento+1;
        }
        }
        else
        {p=0;LED ='0';wait_ms(4000); intento=intento+1;
        }
        }
        else
        {p=0;  LED ='0';wait_ms(4000); intento=intento+1;
        }
        }
        else
        {p=0;LED ='0';wait_ms(4000); intento=intento+1;
        }
}
else
{p=0;LED ='0';wait_ms(4000); intento=intento+1;
}
}
if (intento==3){
intento=0;
}
}
//-------------------------------------------------------
/// Enviar a la Central byte de cerrar la cesión
//-------------------------------------------------------
void Desconectar (void)
{       SFRPAGE = CONFIG_PAGE; EA=0;SFRPAGE = UART0_PAGE;
        SBUF0='x';//Enviar solicitud de desconexión
        TI0=0; wait_ms(100);
        SFRPAGE = CONFIG_PAGE;
        TI0=0; EA=1; wait_ms(4000);

        DISC=!DISC;//cerrar la comunicación con el modem
        wait_ms(1000);

        DISC=!DISC; EA=0;
         SFRPAGE = UART1_PAGE;
}
//-------------------------------------------
```

```
// Tramas_UCS
//-----------------------------------------
void Tramas_UCS (void)
{
//*** Verificar que lleguen los datos con el formato correcto ******//
c=3;
  sema1[0]=0x55;sema2[0]=0x55;sema3[0]=0x55;sema4[0]=0x55;
  sema1[1]=resp[2];sema2[1]=resp[20];sema3[1]=resp[39];sema4[1]=resp[57];
  sema1[2]=0x01;sema2[2]=0x01;sema3[2]=0x01;sema4[2]=0x01;
  sema1[17]='a';sema2[17]='a';sema3[17]='a';sema4[17]='a';
 if (m==0)  //si monitoreo fue exitoso
 {
   while (c<17)
   { sema1[c]=resp[c+1];sema2[c]=resp[c+18];
     sema3[c]=resp[c+38];sema4[c]=resp[c+57]; c=c+1;
        }
        }
        else{
        sema1[c]=0x02;sema1[c+1]=0x00;sema1[c+7]=0x03;sema1[c+8]=0x03;
        sema2[c]=0x02;sema2[c+1]=0x00;sema2[c+7]=0x03;sema2[c+8]=0x03;
        sema1[c]=0x02;sema3[c+1]=0x00;sema3[c+7]=0x03;sema3[c+8]=0x03;
        sema1[c]=0x02;sema4[c+1]=0x00;sema4[c+7]=0x03;sema4[c+8]=0x03;
        c=4;
        while (c<10)
        { sema1[c]=0x255;sema2[c]=0x255;
          sema3[c]=0x255;sema4[c]=0x255;c=c+1;
        }
        c=12;
        while (c<17)
        { sema1[c]=0x255;sema2[c]=0x255;
          sema3[c]=0x255;sema4[c]=0x255; c=c+1;
   }
        }
//***** Enviar secuencias a los semáforos *******//
        y=0;
        while(y < 18)
        {SBUF1=sema1[y];// secuencia semaforo1
        TI1=0; wait_ms(100);y=y+1;
        }
        wait_ms(400);y=0;
        while(y < 18)
        {SBUF1=sema2[y];// secuencia semaforo2
        TI1=0; wait_ms(100);y=y+1;
        }
        wait_ms(400); y=0;
        while(y < 18)
```

```
        {SBUF1=sema3[y];// secuencia semaforo3
        TI1=0; wait_ms(100);y=y+1;
        }
        wait_ms(400);y=0;
        while(y < 18)
        {SBUF1=sema4[y];// secuencia semaforo4
        TI1=0; wait_ms(100);y=y+1;
        }
        SFRPAGE = CONFIG_PAGE;
        TI1=0;EA=1;
        wait_ms(400);
        if ((receptor[0]=='A')& (receptor[1]=='B')&(receptor[2]=='C')
        &(receptor[3]=='D'))
        {
    SFRPAGE = 0x0F;
    P7MDOUT |= 0xFF;        // Activar señal de sincronismo
    wait_ms(200);
    P7MDOUT |= 0x00;
    SFRPAGE = CONFIG_PAGE;
    }
}
//-----------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------
// This routine initializes the system clock to use the internal oscillator
// at 24.5 MHz multiplied by two using the PLL.

void SYSCLK_Init (void)
{
  int i;                    // software timer
  char SFRPAGE_SAVE = SFRPAGE;      // Save Current SFR page
  SFRPAGE = CONFIG_PAGE;            // set SFR page
  OSCICN = 0x83;                    // set internal oscillator to run
                            // at its maximum frequency

  CLKSEL = 0x00;                    // Select the internal osc. as
                            // the SYSCLK source
  //Turn on the PLL and increase the system clock by a factor of M/N = 2
  SFRPAGE = CONFIG_PAGE;
  PLL0CN  = 0x00;                   // Set internal osc. as PLL source
  SFRPAGE = LEGACY_PAGE;
  FLSCL   = 0x10;                   // Set FLASH read time for 50MHz clk
                            // or less
  SFRPAGE = CONFIG_PAGE;
  PLL0CN |= 0x01;                   // Enable Power to PLL
  PLL0DIV = 0x01;                   // Set Pre-divide value to N (N = 1)
```

```
  PLL0FLT = 0x01;              // Set the PLL filter register for
                               // a reference clock from 19 - 30 MHz
                               // and an output clock from 45 - 80 MHz
  PLL0MUL = 0x02;              // Multiply SYSCLK by M (M = 2)

  for (i=0; i < 256; i++) ;    // Wait at least 5us
  PLL0CN |= 0x02;              // Enable the PLL
  while(!(PLL0CN & 0x10));     // Wait until PLL frequency is locked
  CLKSEL  = 0x02;              // Select PLL as SYSCLK source

  SFRPAGE = SFRPAGE_SAVE;      // Restore SFR page
}

//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// This routine configures the crossbar and GPIO ports.
//
void PORT_Init (void)
{
  char SFRPAGE_SAVE = SFRPAGE;     // Save Current SFR page

  SFRPAGE = CONFIG_PAGE;           // set SFR page

  XBR0   =0x04;
  XBR1   =0x00;
  XBR2   = 0x46;               // Enable crossbar and weak pull-up,e indicar que
la EMI es en el puerto 0
                               // Enable UART1

      P0MDOUT |= 0x05;         // Set TX1 pin to push-pull
  //P1MDOUT |= 0x40;           // Set P1.6(LED) to push-pull
      P1MDOUT |= 0xC0;

  SFRPAGE = 0x0F;              // set SFR page
  P5MDOUT  |= 0xFF;
  P6MDOUT  |= 0xFF;
      P7MDOUT  |= 0X00;
      SFRPAGE = SFRPAGE_SAVE;  // Restore SFR page
}
//-----------------------------------------------------------------------------
// UART1_Init
//-----------------------------------------------------------------------------
//
// Configure the UART1 using Timer1, for <baudrate> and 8-N-1.
```

```
//
void UART_Init (void)
{
  char SFRPAGE_SAVE = SFRPAGE;        // Save Current SFR page
  SFRPAGE = UART1_PAGE;

  SCON1  = 0x50;                  // SCON1: mode 0, 8-bit UART, enable RX

 SFRPAGE = UART0_PAGE;
  SCON0  = 0x50;
  SSTA0  = 0x1F;

 SFRPAGE = TIMER01_PAGE;
  TMOD  &= ~0xF0;
  TMOD  |= 0x20;                  // TMOD: timer 1, mode 2, 8-bit reload

  CKCON  |= 0x01;
  TH1    = 0x96;
  SFRPAGE  = TIMER01_PAGE;
  TMOD    = 0x20;
  CKCON   = 0x01;
  TH1    = 0x96;

  TL1 = TH1;                      // initialize Timer1
  TR1 = 1;                        // start Timer1

  SFRPAGE  = TMR3_PAGE;
  TMR3CF   = 0x08;
  RCAP3L   = 0xFF;
  RCAP3H   = 0xFF;
  TR3 = 1;

 SFRPAGE  = TMR4_PAGE;

   TMR4CF   = 0x08;
   RCAP4L   = 0xE5;
   RCAP4H   = 0xFF;
        TR4 = 1;
  SFRPAGE = UART1_PAGE;
// TI1 = 1;
// RI1 = 1;

  SFRPAGE = UART0_PAGE;
// TI0 = 1;
// RI0 = 1;                         // Indicate TX1 ready
```

```
  SFRPAGE = SFRPAGE_SAVE;          // Restore SFR page

}
void EMI_Init (void)

{
char SFRPAGE_SAVE = SFRPAGE;       // Save Current SFR page

  SFRPAGE  = EMI0_PAGE;
//  EMI0CF   = 0x07;
      SFRPAGE = SFRPAGE_SAVE;          // Restore SFR page
}
void Interrupts_Init (void)
{char SFRPAGE_SAVE = SFRPAGE;       // Save Current SFR page
  IE      = 0x90;
  IP      = 0x10;
  EIE2    = 0x41;
      EIP2 = 0x40;
SFRPAGE = SFRPAGE_SAVE;          // Restore SFR page
}
//-----------------------------------------------------------------------
// Subrutinas de soporte
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
// wait_ms
//-----------------------------------------------------------------------
//
// Esta rutina genera un retraso de tiempo de <ms> millisegundos.
//
void wait_ms(int ms)
{
 char SFRPAGE_SAVE = SFRPAGE;       //

 SFRPAGE = TMR2_PAGE;

 TMR2CN = 0x00;                  // Stop Timer3; Clear TF3;
 TMR2CF = 0x00;                  // use SYSCLK/12 as timebase
RCAP2H = (-(SYSCLK/1000/12)) >> 8;
RCAP2L = (-(SYSCLK/1000/12)) ;        // Timer 2 overflows at 1 kHz


 ET2 = 0;                        // Disable Timer 2 interrupts

 TR2 = 1;                        // Start Timer 2

 while(ms){
```

```
  TF2 = 0;
  while(!TF2);              // wait until timer overflows
  ms--;                     // decrement ms
 }

 TR2 = 0;                   // Stop Timer 2

 SFRPAGE = SFRPAGE_SAVE;          // Restore SFRPAGE/
}

//-----------------------------------------------------------------------
// Limpieza de arreglos de memoria importanta
//-----------------------------------------------------------------------
//
void Limpiar_transmitir(void) //Limpienza del arreglo de transmision
{int xdata w=0;
while(w < 11)
{transmitir[w]=0;w=w+1;
}
}
void Limpiar_monitoreo(void) //Limpienza del arreglo de monitoreo
{int xdata w=3;
while(w < 7)
{monitoreo[w]=0;w=w+1;
}
}
void Limpiar_alarma(void) //Limpienza del arreglo de alarma
{int xdata w=0;
while(w < 10)
{alarma[w]=0;w=w+1;
}
}
void Limpiar_peaton(void) //Limpienza del arreglo de peaton
{int xdata w=0;
while(w < 10)
{peaton[w]=0;w=w+1;
}
}
void Limpiar_receptor(void)//Limpienza del arreglo de recepción (UART1)
{int xdata t=0;
while(t < 31)
{cont[t]=0;t=t+1;
}
}
void Limpiar_resp(void)//Limpienza del arreglo de recepción (UART0)
{int xdata v=0;
```

```
while(v < 12)
{resp[v]=0;v=v+1;
}
}
//-----------------------------------------------------------
//Monitorear conexión
//-----------------------------------------------------------
void Monitorear (void)
{
transmitir[0]='U'   ;//inicio de trama
transmitir[2]='0'   ;//tipo de trama;
transmitir[17]='a'   ;//fin de trama;
y=3;
while (y<17){
 transmitir[y]=0x255; y=y+1;
}
q=1;
while (q<5)
{EA=0;SFRPAGE = UART1_PAGE;y=0;transmitir[1]=q;
 while(y < 18)
 {SBUF1=transmitir[y];//Eviar a las UCS trama de monitoreo
  TI1=0;  wait_ms(100);  y=y+1;
 }
  q=q+1;
}
}
//-----------------------------------------------------------------------------
// Interrupcion del Timer 3
//-----------------------------------------------------------------------------
//

void Timer3_ISR (void) interrupt 14//interrupcion de timer3

{ TF3=0;
 tim5min=tim5min+1;
 if (tim5min== 300000)        //Condicional que indica en el momento que se
cumplan 5 min
 {
 Conectarse();
 Desconectar();
 Tramas_UCS();
 tim5min=0;
 }
}
//-----------------------------------------------------------------------------
// Interrupcion de UART 1
```

```
//-----------------------------------------------------------------------
//
void UART1_ISR (void) interrupt 20 //proceso de recepcion
{
if (TI1==1)
{
TI1=0;
}
else
{
if (RI1==1)
{ SFRPAGE = UART1_PAGE;
RI1=0;
SCON1=0x50;
receptor[acum] = SBUF1; // Se acumulan todos los datos recibidos desde el sensor
en un arreglo de memoria
acum = acum + 1;
switch(receptor[acum-1])
{
 case '255':
 alarma[3]='1'; alarma[4]='1'; alarma[5]='1'; alarma[6]='1';
 break;
 case '129':
 peaton[3]='1'; break;
 case '65':
 peaton[4]='1'; break;
 case '33':
 peaton[5]='1'; break;
 case '17':
 peaton[6]='1'; break;
 case '128':
 monitoreo[3]='1'; break;
 case '64':
 monitoreo[4]='1'; break;
 case '32':
 monitoreo[5]='1'; break;
 case '16':
 monitoreo[6]='1'; break;
 case '130';
 ack[3]='1'; break;
 case '66':
 ack[4]='1'; break;
 case '34':
 ack[5]='1'; break;
 case '18':
 ack[6]='1'; break;
```

```
}
}
}
}
//-----------------------------------------------------------------------------
// Interrupcion de UART0
//-----------------------------------------------------------------------------
void UART0_ISR (void) interrupt 4 //Recibe y envia los datos hacia el modulo
{
if (TI0==1)
{
TI0=0;
 }
else
{
if (RI0==1)
{RI0=0;resp[p]=SBUF0;
p=p+1;SCON0=0x50;
}
}
}
```

Apéndice C
Código del Microcontrolador F16877

```
LIST  P=16F877
            RADIX       HEX
            INCLUDE     <P16F877.inc>
UNO                 EQU  20h
DOS                 EQU  21h
TRES                EQU  22h
CUATRO              EQU  23h
CINCO               EQU  24h
SEIS                EQU  25h
SIETE               EQU  26h
T1                  EQU  27h
T2                  EQU  28h
T3                  EQU  29h
T4                  EQU  2Ah
T5                  EQU  2Bh
T6                  EQU  2Ch
T7                  EQU  2Dh
R_ContA             EQU  2Eh
R_ContB             EQU  2Fh
R_ContC             EQU  30h
CONTADOR            EQU  31h
BUFFER              EQU  33h
TIEMPO              EQU  34h
ID                  EQU  35h
UNOA                EQU  36h
DOSA                EQU  37h
TRESA               EQU  38h
CUATROA             EQU  39h
CINCOA              EQU  3Ah
SEISA               EQU  3Bh
SIETEA              EQU  3Ch
T1A                 EQU  3Dh
T2A                 EQU  3Eh
T3A                 EQU  3Fh
T4A                 EQU  40h
T5A                 EQU  41h
T6A                 EQU  42h
T7A                 EQU  43h
ID_A                EQU  44h
ID_P                EQU  45h

            ORG  0x00
            goto RS232
            ORG  0x04
            goto INTERR              ;Rutina de interrupción
```

```
;ESTA ES LA PARTE INICIAL DE LA RUTINA DEL PIC...LO PRIMERO Q
HACE ;ES CONFIGURAR LOS PARAMETROS NECESARIOS PARA LA
;COMUNICACIÓN

RS232
        call    BANCO0          ;paso al banco 0 para cambiar ciertas
variables
        clrf    PORTB
        clrf    PORTA
        clrf    PORTD
        clrf    PORTC           ;Limpia salidas y entradas
        call    BANCO1          ;paso al banco 1
        movlw b'00000000'
        movwf TRISA
        movlw b'11111111'
        movwf TRISD
        movlw b'00000111'
        movwf TRISB             ;Configura los pines del puerto
                movlw b'10110011'       ;RC7/Rx y RC6/Tx:
        movwf TRISC             ;RC5-RC0:Entradas digitales adicionales
        movlw b'00100100'
        movwf TXSTA;Configuracion USART: activacion de transmision, 8 bits
de tx
        movlw .003              ;velocidad cercana a 57600
        movwf SPBRG
        movlw b'00100000'
        movwf PIE1
        movlw 0x07
        movwf ADCON1
        call    BANCO0
        movlw b'10010000'
        movwf RCSTA ;Habilita el puerto serial para recepcion continua, 8 bits
para rx
        clrf    RCREG
        movlw b'11111111'
        movwf OPTION_REG
        movlw b'11010000'
        movwf INTCON            ;habilita interrupciones globales y de
perifericos
        movlw b'00000000'
        movwf CONTADOR
        movlw b'00000000'
        movwf TIEMPO
        movf    PORTD,0         ; Muevo el Puerto D al registro W
        movwf ID                ; Muevo el registro W a ID
        movf    PORTD,0
```

```
        addlw d'64'             ; Sumo 64 al Registro W
        movwf ID_A
        movf  PORTD,0
        addlw d'128'
        movwf ID_P
        goto  ESPERA            ;Voy al estado de espera de inicio del PIC
BANCO1
              bsf       STATUS,RP0
              bcf       STATUS,RP1
              return

BANCO0
              bcf       STATUS,RP0
              bcf       STATUS,RP1
              return
ESPERA
              bsf       PORTA,1
              call      Retardo_1s
              bcf       PORTA,1
              call      Retardo_1s
              goto      ESPERA
INTERR                                  ; Verifica el tipo de interrupción
              btfss     INTCON,INTF
              goto      INTERR_A
              btfsc     PORTB,1
              call      ENVIAR_ALARMA
              btfsc     PORTB,2
              goto      ENVIAR_PEATON
              goto      INTERR_A
ENVIAR_ALARMA                           ; prende el pin de alarma y envía el
byte
              bcf       INTCON,INTF
              bsf       PORTC,2
              movlw     b'11111111'
              movwf     TXREG
              call      BANCO1
TX?_ALARMA                              ; Verifica la transmisión
              btfss     TXSTA,TRMT
              goto      TX?_ALARMA
              call      BANCO0
              return
ENVIAR_PEATON                           ; prende el pin de peatón y envía el
byte
              bcf       INTCON,INTF
              bsf       PORTC,3
              movf      ID_P
```

```
            movwf   TXREG
            call    BANCO1
TX?_PEATON
            btfss   TXSTA,TRMT
            goto    TX?_PEATON
            call    BANCO0
            goto    INTERR_A
INTERR_A
            call    BANCO0
            btfss   PIR1,RCIF     ;Chequea la bandera de rx
            goto    EXIT_INTERR
            clrf    BUFFER
            movf    RCREG,W
            movwf   BUFFER


LISTO
            movlw d'0'            ; Suma hasta que llegue a 15 registros
guardados
            addlw d'1'
            addwf CONTADOR,1
            movf  CONTADOR,0
            xorlw d'37'
            btfss STATUS,Z
            goto  VERIFICAR_U
            goto  EXIT_INTERR
VERIFICAR_U                       ; Verifica inicio de trama
            movf  CONTADOR,0
            xorlw d'1'
            btfss STATUS,Z
            goto  VERIFICAR
            goto  COMPARAR_U
COMPARAR_U
            movf  BUFFER,W
            xorlw 'U'
            btfss STATUS,Z
            goto  REINICIAR
            goto  EXIT_INTERR
EXIT_INTERR                       ; regresa a la espera o al ciclo si no hay
recepción
            retfie
VERIFICAR
            movf  CONTADOR,0
            xorlw d'2'
            btfss STATUS,Z
```

```
            goto    VERIFICAR_TIPO
            goto    COMPARAR_ID
COMPARAR_ID
            movf    BUFFER,W
            xorwf   PORTD
            btfss   STATUS,Z
            goto    REINICIAR
            goto    EXIT_INTERR
REINICIAR                           ; verifica que la trama sea para el
            movlw d'0'
            movwfCONTADOR
            goto    EXIT_INTERR
VERIFICAR_TIPO                      ; chequea el tipo de trama
            movf    CONTADOR,0
            xorlw   d'3'
            btfss   STATUS,Z
            goto    VERIFICAR1
            goto    COMPARAR_TIPO_0
COMPARAR_TIPO_0
            movf    BUFFER,W
            xorlw   '0'
            btfss   STATUS,Z
            goto    COMPARAR_TIPO_1
            goto    EXIT_INTERR
COMPARAR_TIPO_1
            movf    BUFFER,W
            xorlw   '1'
            btfss   STATUS,Z
            goto    REINICIAR
            goto    ENVIAR_A
ENVIAR_A
            bsf     PORTB,6
            movf    ID
            movwfTXREG
            call    BANCO1
TX?_A
            btfss   TXSTA,TRMT
            goto    TX?_A
            call    BANCO0
            goto    REINICIAR
VERIFICAR1
; comienza el ciclo de verificación del contador y almacenamiento de las variable
            movf    CONTADOR,0
            xorlw   d'4'
            btfss   STATUS,Z
            goto    VERIFICAR2
```

```
            goto    GUARDAR1
GUARDAR1
            clrf    UNO
            movf    BUFFER,W
            movwfUNO
            goto    EXIT_INTERR
VERIFICAR2
            movf    CONTADOR,0
            xorlw   d'5'
            btfss   STATUS,Z
            goto    VERIFICAR3
            goto    GUARDAR2
GUARDAR2
            clrf    DOS
            movf    BUFFER,W
            movwfDOS
            goto    EXIT_INTERR
VERIFICAR3
            movf    CONTADOR,0
            xorlw   d'6'
            btfss   STATUS,Z
            goto    VERIFICAR4
            goto    GUARDAR3
GUARDAR3
            clrf    TRES
            movf    BUFFER,W
            movwfTRES
            goto    EXIT_INTERR
VERIFICAR4
            movf    CONTADOR,0
            xorlw   d'7'
            btfss   STATUS,Z
            goto    VERIFICAR5
            goto    GUARDAR4
GUARDAR4
            clrf    CUATRO
            movf    BUFFER,W
            movwfCUATRO
            goto    EXIT_INTERR
VERIFICAR5
            movf    CONTADOR,0
            xorlw   d'8'
            btfss   STATUS,Z
            goto    VERIFICAR6
            goto    GUARDAR5
GUARDAR5
```

```
            clrf    CINCO
            movf    BUFFER,W
            movwfCINCO
            goto    EXIT_INTERR
VERIFICAR6
            movf    CONTADOR,0
            xorlw   d'9'
            btfss   STATUS,Z
            goto    VERIFICAR7
            goto    GUARDAR6
GUARDAR6
            clrf    SEIS
            movf    BUFFER,W
            movwfSEIS
            goto    EXIT_INTERR
VERIFICAR7
            movf    CONTADOR,0
            xorlw   d'10'
            btfss   STATUS,Z
            goto    VERIFICAR8
            goto    GUARDAR7
GUARDAR7
            clrf    SIETE
            movf    BUFFER,W
            movwfSIETE
            goto    EXIT_INTERR
VERIFICAR8
            movf    CONTADOR,0
            xorlw   d'11'
            btfss   STATUS,Z
            goto    VERIFICAR9
            goto    GUARDAR8
GUARDAR8
            clrf    T1
            movf    BUFFER,W
            movwfT1
            goto    EXIT_INTERR
VERIFICAR9
            movf    CONTADOR,0
            xorlw   d'12'
            btfss   STATUS,Z
            goto    VERIFICAR10
            goto    GUARDAR9
GUARDAR9
            clrf    T2
            movf    BUFFER,W
```

```
        movwf T2
        goto  EXIT_INTERR
VERIFICAR10
        movf  CONTADOR,0
        xorlw d'13'
        btfss STATUS,Z
        goto  VERIFICAR11
        goto  GUARDAR10
GUARDAR10
        clrf  T3
        movf  BUFFER,W
        movwf T3
        goto  EXIT_INTERR
VERIFICAR11
        movf  CONTADOR,0
        xorlw d'14'
        btfss STATUS,Z
        goto  VERIFICAR12
        goto  GUARDAR11
GUARDAR11
        clrf  T4
        movf  BUFFER,W
        movwf T4
        goto  EXIT_INTERR
VERIFICAR12
        movf  CONTADOR,0
        xorlw d'15'
        btfss STATUS,Z
        goto  VERIFICAR13
        goto  GUARDAR12
GUARDAR12
        clrf  T5
        movf  BUFFER,W
        movwf T5
        goto  EXIT_INTERR

VERIFICAR13
        movf  CONTADOR,0
        xorlw d'16'
        btfss STATUS,Z
        goto  VERIFICAR14
        goto  GUARDAR13
GUARDAR13
        clrf  T6
        movf  BUFFER,W
        movwf T6
```

```
            goto    EXIT_INTERR
VERIFICAR14
            movf    CONTADOR,0
            xorlw   d'17'
            btfss   STATUS,Z
            goto    VERIFICAR15
            goto    GUARDAR14
GUARDAR14
            clrf    T7
            movf    BUFFER,W
            movwfT7
            goto    EXIT_INTERR
VERIFICAR15
            movf    CONTADOR,0
            xorlw   d'18'
            btfss   STATUS,Z
            goto    VERIFICAR_UA
            goto    COMPARAR_FINAL
COMPARAR_FINAL
            movf    BUFFER,W
            xorlw   'a'
            btfss   STATUS,Z
            goto    REINICIAR
            goto    ENVIAR_SINCRONISMO
ENVIAR_SINCRONISMO  ;reporta la llegada completa de la trama
            bsf             PORTB,7
            bsf             PORTB,6
            movf            ID_A
            movwfTXREG
            call    BANCO1
TX?_SINCRONISMO     ; verifica la señal de sincronismo y espera para iniciar
            btfss   TXSTA,TRMT
            goto    TX?_SINCRONISMO
            call    BANCO0
            call    EXIT_INTERR
ESPERA_INICIO
            btfss   PORTC,4
            goto    ESPERA_INICIO
            goto    PRENDER_UNO

PRENDER_UNO     ; comienza la salida de los valores por el puerto ya la rutina
de espera para cada secuencia.
            movf    CONTADOR,W
            xorlw   d'36'
            btfsc   STATUS,Z
            call    CAMBIO
```

```
            goto    PRENDER_UNOA
PRENDER_UNOA ; este es el ciclo para cuando ya han recibido nuevas variables
            bcf             PORTB,4;Apago Verde
            bsf             PORTB,3; Prendo Rojo
            movf    UNO,0
            xorlw   d'255'
            btfsc   STATUS,Z
            goto    APAGAR
            movlw d'0'
            movwfTIEMPO
            movf    UNO,0
            movwfPORTA
            movf    UNO,0
            xorlw   b'00000001'
            btfss   STATUS,Z
            goto    TIEMPO_UNO
            bcf             PORTB,3; Apago Rojo
            bsf             PORTB,4; Prendo Verde
TIEMPO_UNO
            call    Retardo_1s
            movf    TIEMPO,0
            addlw   d'1'
            movwfTIEMPO
            xorwf   T1,0
            btfss   STATUS,Z
            goto    TIEMPO_UNO
            goto    PRENDER_DOS
PRENDER_DOS
            bcf             PORTB,4;Apago Verde
            bsf             PORTB,3; Prendo Rojo
            movf    DOS,0
            xorlw   d'255'
            btfsc   STATUS,Z
            goto    APAGAR
            movlw d'0'
            movwfTIEMPO
            movf    DOS,0
            movwfPORTA
            movf    DOS,0
            movwfPORTA
            movf    DOS,0
            xorlw   b'00000001'
            btfss   STATUS,Z
            goto    TIEMPO_DOS
            bcf             PORTB,3; Apago Rojo
            bsf             PORTB,4; Prendo Verde
```

```
TIEMPO_DOS
        call    Retardo_1s
        movf    TIEMPO,0
        addlw   d'1'
        movwf   TIEMPO
        xorwf   T2,0
        btfss   STATUS,Z
        goto    TIEMPO_DOS
        goto    PRENDER_TRES
PRENDER_TRES
        bcf             PORTB,4;Apago Verde
        bsf             PORTB,3; Prendo Rojo
        movf    TRES,0
        xorlw   d'255'
        btfsc   STATUS,Z
        goto    APAGAR
        movlw   d'0'
        movwf   TIEMPO
        movf    TRES,0
        movwf   PORTA
        movf    TRES,0
        xorlw   b'00000001'
        btfss   STATUS,Z
        goto    TIEMPO_TRES
        bcf             PORTB,3; Apago Rojo
        bsf             PORTB,4; Prendo Verde
TIEMPO_TRES
        call    Retardo_1s
        movf    TIEMPO,0
        addlw   d'1'
        movwf   TIEMPO
        xorwf   T3,0
        btfss   STATUS,Z
        goto    TIEMPO_TRES
        goto    PRENDER_CUATRO
PRENDER_CUATRO
        bcf             PORTB,4;Apago Verde
        bsf             PORTB,3; Prendo Rojo
        movf    CUATRO,0
        xorlw   d'255'
        btfsc   STATUS,Z
        goto    APAGAR
        movlw   d'0'
        movwf   TIEMPO
        movf    CUATRO,0
        movwf   PORTA
```

```
        movf   CUATRO,0
        xorlw  b'00000001'
        btfss  STATUS,Z
        goto   TIEMPO_CUATRO
        bcf            PORTB,3; Apago Rojo
        bsf            PORTB,4; Prendo Verde
TIEMPO_CUATRO
        call   Retardo_1s
        movf   TIEMPO,0
        addlw  d'1'
        movwfTIEMPO
        xorwf  T4,0
        btfss  STATUS,Z
        goto   TIEMPO_CUATRO
        goto   PRENDER_CINCO
PRENDER_CINCO
        bcf            PORTB,4;Apago Verde
        bsf            PORTB,3; Prendo Rojo
        movf   CINCO,0
        xorlw  d'255'
        btfsc  STATUS,Z
        goto   APAGAR
        movlw d'0'
        movwfTIEMPO
        movf   CINCO,0
        movwfPORTA
        movf   CINCO,0
        xorlw  b'00000001'
        btfss  STATUS,Z
        goto   TIEMPO_CINCO
        bcf            PORTB,3; Apago Rojo
        bsf            PORTB,4; Prendo Verde
TIEMPO_CINCO
        call   Retardo_1s
        movf   TIEMPO,0
        addlw  d'1'
        movwfTIEMPO
        xorwf  T5,0
        btfss  STATUS,Z
        goto   TIEMPO_CINCO
        goto   PRENDER_SEIS
PRENDER_SEIS
        bcf            PORTB,4;Apago Verde
        bsf            PORTB,3; Prendo Rojo
        movf   SEIS,0
        xorlw  d'255'
```

```
        btfsc   STATUS,Z
        goto    APAGAR
        movlw d'0'
        movwfTIEMPO
        movf    SEIS,0
        movwfPORTA
        movf    SEIS,0
        xorlw   b'00000001'
        btfss   STATUS,Z
        goto    TIEMPO_SEIS
        bcf             PORTB,3; Apago Rojo
        bsf             PORTB,4; Prendo Verde
TIEMPO_SEIS
        call    Retardo_1s
        movf    TIEMPO,0
        addlw   d'1'
        movwfTIEMPO
        xorwf   T6,0
        btfss   STATUS,Z
        goto    TIEMPO_SEIS
        goto    PRENDER_SIETE
PRENDER_SIETE
        bcf             PORTB,4;Apago Verde
        bsf             PORTB,3; Prendo Rojo
        movf    SIETE,0
        xorlw   d'255'
        btfsc   STATUS,Z
        goto    APAGAR
        movlw d'0'
        movwfTIEMPO
        movf    SIETE,0
        movwfPORTA
        movf    SIETE,0
        xorlw   b'00000001'
        btfss   STATUS,Z
        goto    TIEMPO_SIETE
        bcf             PORTB,3; Apago Rojo
        bsf             PORTB,4; Prendo Verde
TIEMPO_SIETE
        call    Retardo_1s
        movf    TIEMPO,0
        addlw   d'1'
        movwfTIEMPO
        xorwf   T7,0
        btfss   STATUS,Z
        goto    TIEMPO_SIETE
```

```
            bcf     PORTB,6
            bcf     PORTC,2
            bcf     PORTC,3
            bcf     PORTB,7
            goto    PRENDER_UNO
APAGAR
            bcf     PORTB,6
            bcf     PORTC,2
            bcf     PORTC,3
            bcf     PORTB,7
            goto    PRENDER_UNO

VERIFICAR_UA
            movf    CONTADOR,0
            xorlw   d'19'
            btfss   STATUS,Z
            goto    VERIFICARAA
            goto    COMPARAR_UA
COMPARAR_UA
            movf    BUFFER,W
            xorlw   'U'
            btfss   STATUS,Z
            goto    REINICIAR_CONTADOR
            goto    EXIT_INTERR
VERIFICARAA
            movf    CONTADOR,0
            xorlw   d'20'
            btfss   STATUS,Z
            goto    VERIFICAR_TIPOA
            goto    COMPARAR_IDA
COMPARAR_IDA
            movf    BUFFER,W
            xorwf   PORTD
            btfss   STATUS,Z
            goto    REINICIAR_CONTADOR
            goto    EXIT_INTERR
REINICIAR_CONTADOR
            movlw d'18'
            movwfCONTADOR
            goto    EXIT_INTERR
VERIFICAR_TIPOA
            movf    CONTADOR,0
            xorlw   d'21'
            btfss   STATUS,Z
            goto    VERIFICAR1A
            goto    COMPARAR_TIPO_0A
```

```
COMPARAR_TIPO_0A
        movf    BUFFER,W
        xorlw   '0'
        btfss   STATUS,Z
        goto    COMPARAR_TIPO_1A
        goto    EXIT_INTERR
COMPARAR_TIPO_1A
        movf    BUFFER,W
        xorlw   '1'
        btfss   STATUS,Z
        goto    REINICIAR_CONTADOR
        goto    ENVIAR_AA
ENVIAR_AA
        bsf             PORTB,6
        movlw 'M'
        movwfTXREG
        call    BANCO1
TX?_AA
        btfss   TXSTA,TRMT
        goto    TX?_AA
        call    BANCO0
        goto    REINICIAR_CONTADOR
VERIFICAR1A
        movf    CONTADOR,0
        xorlw   d'22'
        btfss   STATUS,Z
        goto    VERIFICAR2A
        goto    GUARDAR1A
GUARDAR1A
        clrf    UNOA
        movf    BUFFER,W
        movwfUNOA
        goto    EXIT_INTERR
VERIFICAR2A
        movf    CONTADOR,0
        xorlw   d'23'
        btfss   STATUS,Z
        goto    VERIFICAR3A
        goto    GUARDAR2A
GUARDAR2A
        clrf    DOSA
        movf    BUFFER,W
        movwfDOSA
        goto    EXIT_INTERR
VERIFICAR3A
        movf    CONTADOR,0
```

```
        xorlw  d'24'
        btfss  STATUS,Z
        goto   VERIFICAR4A
        goto   GUARDAR3A
GUARDAR3A
        clrf   TRESA
        movf   BUFFER,W
        movwf  TRESA
        goto   EXIT_INTERR
VERIFICAR4A
        movf   CONTADOR,0
        xorlw  d'25'
        btfss  STATUS,Z
        goto   VERIFICAR5A
        goto   GUARDAR4A
GUARDAR4A
        clrf   CUATROA
        movf   BUFFER,W
        movwf  CUATROA
        goto   EXIT_INTERR
VERIFICAR5A
        movf   CONTADOR,0
        xorlw  d'26'
        btfss  STATUS,Z
        goto   VERIFICAR6A
        goto   GUARDAR5A
GUARDAR5A
        clrf   CINCOA
        movf   BUFFER,W
        movwf  CINCOA
        goto   EXIT_INTERR
VERIFICAR6A
        movf   CONTADOR,0
        xorlw  d'27'
        btfss  STATUS,Z
        goto   VERIFICAR7A
        goto   GUARDAR6A
GUARDAR6A
        clrf   SEISA
        movf   BUFFER,W
        movwf  SEISA
        goto   EXIT_INTERR
VERIFICAR7A
        movf   CONTADOR,0
        xorlw  d'28'
        btfss  STATUS,Z
```

```
            goto    VERIFICAR8A
            goto    GUARDAR7A
GUARDAR7A
            clrf    SIETEA
            movf    BUFFER,W
            movwf   SIETEA
            goto    EXIT_INTERR
VERIFICAR8A
            movf    CONTADOR,0
            xorlw   d'29'
            btfss   STATUS,Z
            goto    VERIFICAR9A
            goto    GUARDAR8A
GUARDAR8A
            clrf    T1A
            movf    BUFFER,W
            movwf   T1A
            goto    EXIT_INTERR
VERIFICAR9A
            movf    CONTADOR,0
            xorlw   d'30'
            btfss   STATUS,Z
            goto    VERIFICAR10A
            goto    GUARDAR9A
GUARDAR9A
            clrf    T2A
            movf    BUFFER,W
            movwf   T2A
            goto    EXIT_INTERR
VERIFICAR10A
            movf    CONTADOR,0
            xorlw   d'31'
            btfss   STATUS,Z
            goto    VERIFICAR11A
            goto    GUARDAR10A
GUARDAR10A
            clrf    T3A
            movf    BUFFER,W
            movwf   T3A
            goto    EXIT_INTERR
VERIFICAR11A
            movf    CONTADOR,0
            xorlw   d'32'
            btfss   STATUS,Z
            goto    VERIFICAR12A
            goto    GUARDAR11A
```

```
GUARDAR11A
            clrf    T4A
            movf    BUFFER,W
            movwfT4A
            goto    EXIT_INTERR
VERIFICAR12A
            movf    CONTADOR,0
            xorlw   d'33'
            btfss   STATUS,Z
            goto    VERIFICAR13A
            goto    GUARDAR12A
GUARDAR12A
            clrf    T5A
            movf    BUFFER,W
            movwfT5A
            goto    EXIT_INTERR
VERIFICAR13A
            movf    CONTADOR,0
            xorlw   d'34'
            btfss   STATUS,Z
            goto    VERIFICAR14A
            goto    GUARDAR13A
GUARDAR13A
            clrf    T6A
            movf    BUFFER,W
            movwfT6A
            goto    EXIT_INTERR
VERIFICAR14A
            movf    CONTADOR,0
            xorlw   d'35'
            btfss   STATUS,Z
            goto    VERIFICAR15A
            goto    GUARDAR14A
GUARDAR14A
            clrf    T7A
            movf    BUFFER,W
            movwfT7A
            goto    EXIT_INTERR
VERIFICAR15A
            movf    CONTADOR,0
            xorlw   d'36'
            btfss   STATUS,Z
            goto    EXIT_INTERR
            goto    COMPARAR_FINAL_A
COMPARAR_FINAL_A
            movf    BUFFER,W
```

```
            xorlw   'a'
            btfss   STATUS,Z
            goto    REINICIAR_CONTADOR
            goto    ENVIAR_SINCRONISMO_A
ENVIAR_SINCRONISMO_A
            bsf             PORTB,7
            bsf             PORTB,6
            movlw 'R'
            movwfTXREG
            call    BANCO1
TX?_SINCRONISMO_A
            btfss   TXSTA,TRMT
            goto    TX?_SINCRONISMO_A
            call    BANCO0
            goto    EXIT_INTERR
CAMBIO
            btfss   PORTC,4
            goto    PRENDER_UNOA
            movlw d'18'
            movwfCONTADOR
            movf    UNOA,W
            movwfUNO
            movf    DOSA,W
            movwfDOS
            movf    TRESA,W
            movwfTRES
            movf    CUATROA,W
            movwfCUATRO
            movf    CINCOA,W
            movwfCINCO
            movf    SEISA,W
            movwfSEIS
            movf    SIETEA,W
            movwfSIETE
            movf    T1A,W
            movwfT1
            movf    T2A,W
            movwfT2
            movf    T3A,W
            movwfT3
            movf    T4A,W
            movwfT4
            movf    T5A,W
            movwfT5
            movf    T6A,W
            movwfT6
```

```
        movf    T7A,W
        movwf   T7
        clrf    UNOA
        clrf    DOSA
        clrf    TRESA
        clrf    CUATROA
        clrf    CINCOA
        clrf    SEISA
        clrf    SIETEA
        clrf    T1A
        clrf    T2A
        clrf    T3A
        clrf    T4A
        clrf    T5A
        clrf    T6A
        clrf    T7A
        return

; RETARDO de 1segundo ----------------------------------------------;

Retardo_1s                      ; La llamada "call" aporta 2 ciclos máquina.
        movlw d'10'             ; Aporta 1 ciclo máquina. Este es el valor de
"N".
        goto    Retardo_1Decima         ; Aporta 2 ciclos máquina.

; El próximo bloque "Retardo_1Decima" tarda:
; 1 + N + N + MxN + MxN + KxMxN + (K-1)xMxN + MxNx2 + (K-1)xMxNx2
+
;  + (M-1)xN + Nx2 + (M-1)xNx2 + (N-1) + 2 + (N-1)x2 + 2 =
; = (2 + 4M + 4MN + 4KM) ciclos máquina. Para K=249, M=100 y N=1 supone
100011
; ciclos máquina que a 4 MHz son 100011 µs = 100 ms = 0,1 s = 1 décima de
segundo.
;
Retardo_1Decima
        movwf R_ContC                   ; Aporta 1 ciclo máquina.
R1Decima_BucleExterno2
        movlw d'100'            ; Aporta Nx1 ciclos máquina. Este es el valor
de "M".
        movwf R_ContB                   ; Aporta Nx1 ciclos máquina.
R1Decima_BucleExterno
        movlw d'249'            ; Aporta MxNx1 ciclos máquina. Este es el
valor de "K".
        movwf R_ContA                   ; Aporta MxNx1 ciclos máquina.
R1Decima_BucleInterno
        nop                     ; Aporta KxMxNx1 ciclos máquina.
```

```
        decfsz R_ContA,F            ; (K-1)xMxNx1 cm (si no salta) + MxNx2
cm (al saltar).
        goto   R1Decima_BucleInterno    ;  Aporta  (K-1)xMxNx2  ciclos
máquina.
        decfsz R_ContB,F            ; (M-1)xNx1 cm (cuando no salta) + Nx2 cm
(al saltar).
        goto   R1Decima_BucleExterno    ; Aporta (M-1)xNx2 ciclos máquina.
        decfsz R_ContC,F            ; (N-1)x1 cm (cuando no salta) + 2 cm (al
saltar).
        goto   R1Decima_BucleExterno2   ; Aporta (N-1)x2 ciclos máquina.
        return                      ; El salto del retorno aporta 2 ciclos máquina.

;En total estas subrutinas tardan:
;                    (N= 20, M=100 y K=249).
; - Retardo_1s:              2 + 1 + 2 + (2 + 4N + 4MN + 4KMN) =  1000047
cm = 1

END
```

Apéndice D
Construcción de Circuitos impresos

Para la creación de las tarjetas de circuito impreso se utilizan láminas de fibra de vidrio recubiertas con capas de cobre adheridas sobre todo el sustrato, por una o ambas capas según las necesidades del circuito, después de realizado el diseño de distribución de los componentes electrónicos de los diferentes modelos de tarjetas se siguen los siguientes pasos:

1. Se imprimen los diversos modelos de tarjetas en papel común para verificar las separaciones de los orificios de los componentes y realizar los ajustes necesarios, hasta obtener las medidas y separaciones requeridas para cada uno de los dispositivos.
2. Los diagramas circuitales se imprimen en material termo sensible (papel transfer), se verifica cuidadosamente que todas las líneas de conducción tengan el espesor deseado y se confirma la posición y espaciamiento de cada uno de los componentes.
3. Unas vez obtenidos los diagramas impresos se fijaron cuidadosamente con cinta adhesiva a cada uno de las láminas de fibra de vidrio, previamente lijadas y se procede a aplicar calor, uniformemente con el uso de una plancha, con la intención de estampar las líneas y puntos de conexión en cada tarjeta. Cuando se observe que el diagrama circuital se encuentra adherido a la superficie de la lámina, se introducen las tarjetas en un recipiente con agua, para eliminar el exceso de papel.

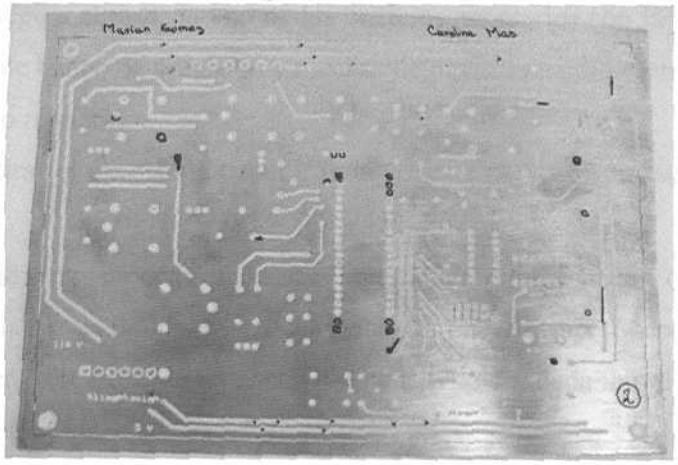


Figura 1.- Tarjeta de UCS después de retirado el exceso de papel.

4. El proceso de atacado químico, para eliminar el excedente de cobre y lograr solo la permanencia del patrón deseado, es llevado a cabo, sumergiendo cada uno de las tarjetas en recipientes con de Cloruro Férrico. Con anterioridad se deben chequear las líneas y puntos impresos y completar de ser necesario las fallas encontradas con el uso de marcadores para acetato, que de igual forma impiden el paso del ácido a utilizar.

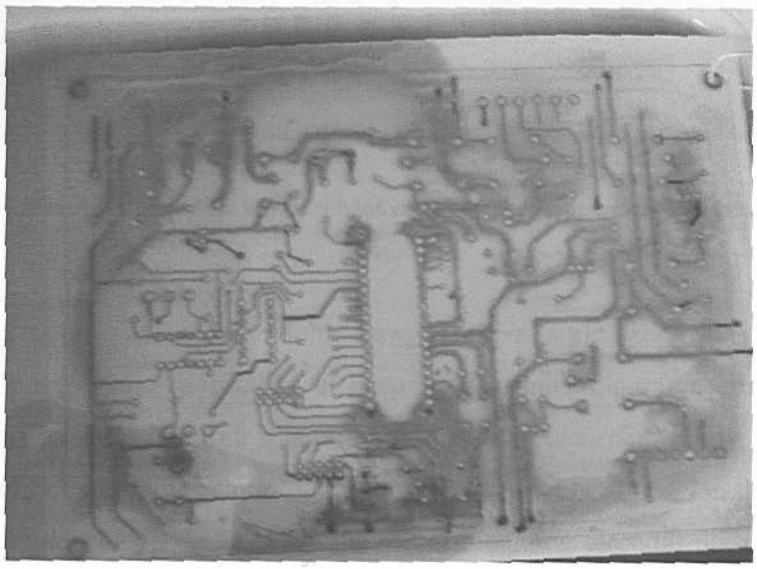


Figura 2.- Tarjeta de UCS introducida en el cloruro ferrico (a mitad del proceso de corrosión)

5. Cuando se observa que toda la capa de cobre sobrante ha desaparecido se retiran las láminas, se sumergen en agua limpia y se dejan secar. Para pasar luego a abrir cada uno de los orificios necesarios con la ayuda de un taladro de banco de precisión y mechas de distintas medidas en función del diámetro requerido.
6. Para retirar el exceso de papel y tinta adherido a las pistas, se utiliza una lija 300
7. Por último se procede a colocar los cables en las líneas de conexión necesarias y cada uno de los componentes en su respectiva posición para proceder con el uso de un cautín y estaño a soldar cada uno de los puntos de conexión.

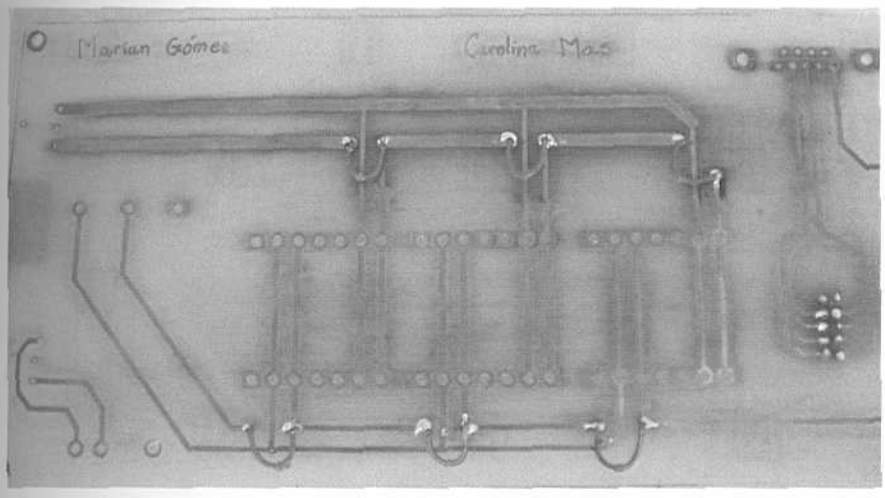


Figura 3.-Parte posterior de la tarjeta con los cables de puente de conexión necesarios ya soldados

Para las tarjetas que requieran doble cara, se utiliza lámina con capas de cobre por ambos lados; luego de realizar el paso 3 del primer diagrama circuital, se abrien algunos orificios centrales, tanto en la baquelita como en la hoja de papel transfer con la otra cara del circuito, para lograr la correcta alineación del par de caras de la tarjeta, luego de tener las líneas del diagrama circuital plasmadas en ambos lados se continua con el resto de los pasos.

Apéndice E
Layout de circuitos impresos

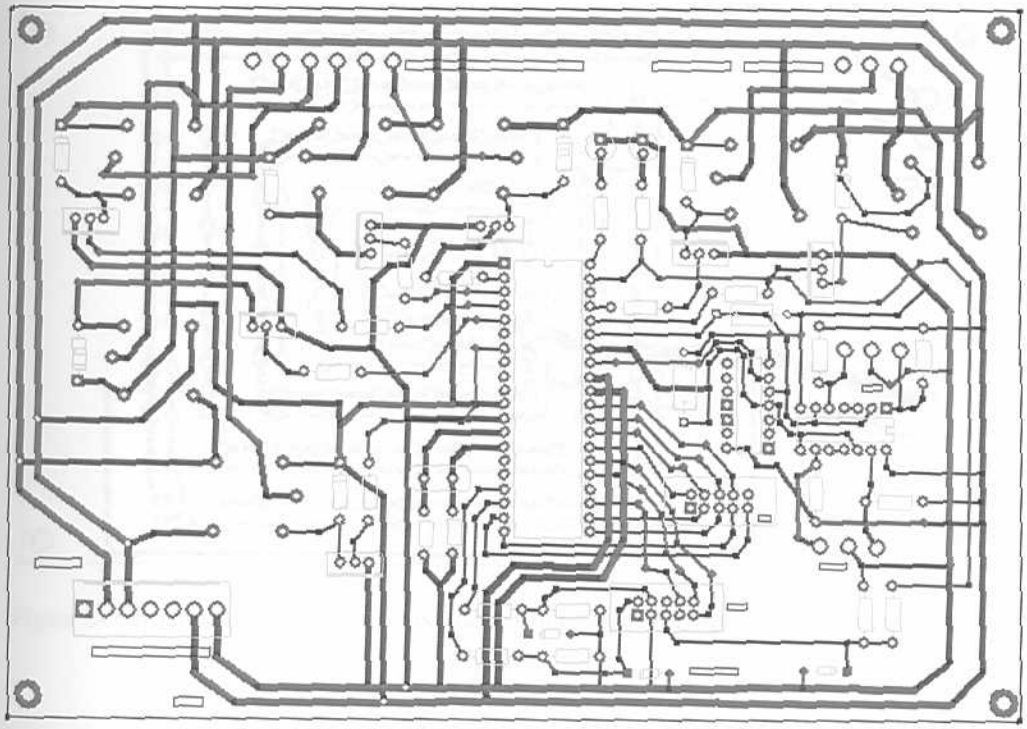


Figura 1.- Layout de la tarjeta de la Unidad de Control de Semáforo

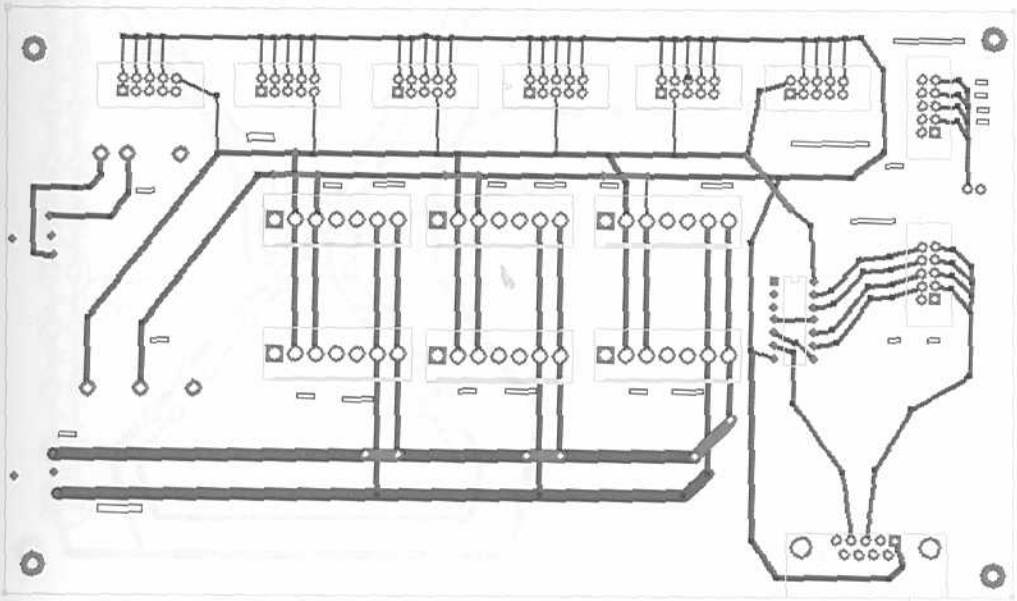


Figura 2.- Layout de la tarjeta de Alimentación

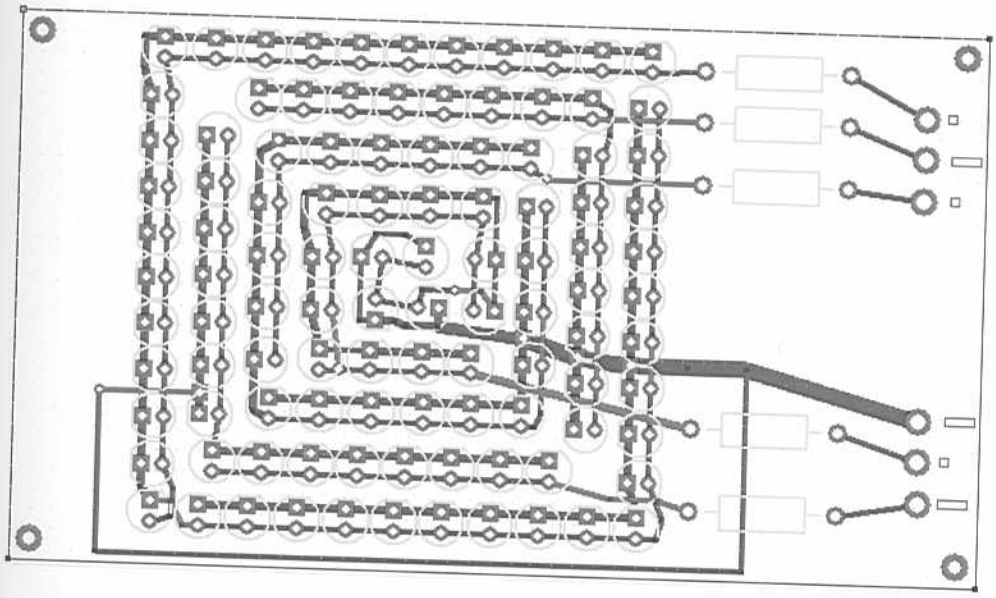


Figura 3.- Layout de la tarjeta de prototipos de semáforos, para luces principales y luces de cruces

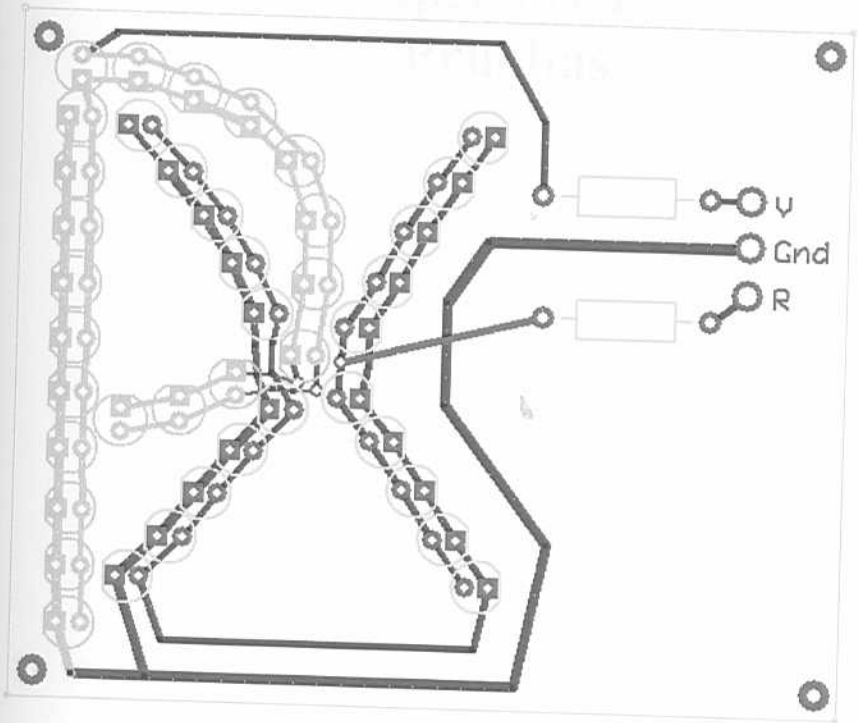


Figura 4.- Layout de la tarjeta de prototipos de semáforos

Apéndice F
Pruebas

1. **Unidad de Simulación de la Central:** en la figura se puede observa el interfaz empleado para verificar el correcto funcionamiento de la USC.

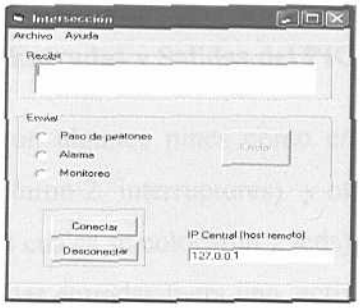


Figura 1.- Interfaz Intersección

2. **Unidad de Transmisión Inalámbrica:** a continuación se presentan los comandos AT necesarios para establecer la conexión entre la USC y la UTI.

   Color negro: indica comandos introducidos al MODEM a través de Hyperterminal, para solicitar conexión

   Color rojo: indica la respuesta recibida

AT+CAD?
OK
AT+CRM=130
OK
AT*PID=5176360@cantv.net
OK
AT*PPW="*****" (Contraseña secreta propocionada por Movilnet)
OK
AT+DIP= "201.211.132.159"
OK
AT+DPORT= "4000"
OK
ATDT123
Opción 1 (Cuando no logra conectarse a la red)
CALL
NO CARRIER

Opción 2 (Cuando existe una conexión exitosa)
CALL
PPP
CONNECT

3. **Pruebas de compilación, inclusión y funcionamiento de software:** Entre las pruebas realizadas como ayuda para lograr la implementación de la Tarjeta para la unidad de Control de Semáforos (UCS), se encuentran:

Manejo de las Entradas y Salidas del PIC

Se programaron algunos pines como entradas digitales (en los cuales se conectaron 2 interruptores) y otros como salidas también digitales (en los cuales se colocaron 2 leds) de manera que cuando el valor lógico de las entradas fuera uno, activado por los interruptores, en las salidas del PIC se encendieran los leds.

Comunicación vía serial con la PC

Se programó el PIC para que se comunicara con el computador y con la ayuda de la herramienta desarrollada en Visual Basic, observar los resultados. Se envió un carácter desde el computador, el PIC lo visualizaba en forma binaria sobre leds y lo volviese a transmitir a la PC a modo de eco, para lograr observar el carácter enviado. La recepción del carácter por parte del PIC se controló mediante la habilitación de las interrupciones periféricas.

Comparación de caracteres

Se programó el PIC de manera que comparase el caracter enviado por la PC con un caracter específico, por ejemplo la letra "A". Básicamente, si el caracter enviado desde la PC es una "A", el PIC enciende un pin, para verificar la recepción y que se efectúen comandos solo en caso de recibir los caracteres correctos correspondientes al inicio de trama e identificador del semáforo.

4. **Interconexión entre la UCS y el Computador**: A continuación en la figura 2 se puede observa el interfaz empleado para verificar el envío y recepción de los caracteres enviados en forma serial a la UCS

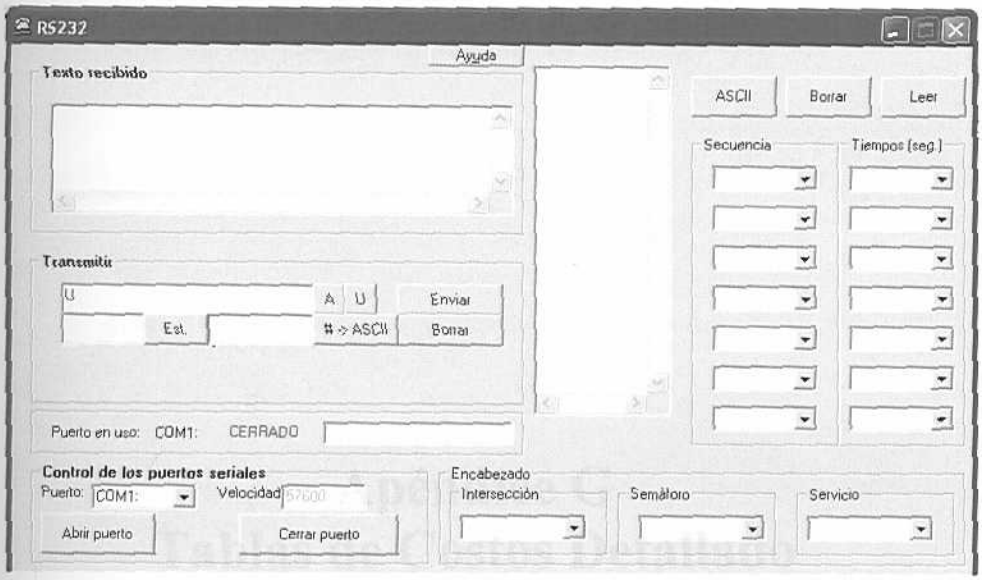


Figura 2.- Interfaz RS232

Interconexión entre la UCS la UCI y el Computador: dada la limitación de trabajar con direcciones IP públicas, en los laboratorios de la universidad, fue necesario realizar un código sencillo para la UCI, que permitiera enviar y transmitir caracteres, con la finalidad de realizar pruebas, que facilitaran observar el comportamiento de la UCS ante los comandos enviados por la UCI y poder percibir mediante el interfaz del Mscomm, las respuestas enviadas por la UCS, obteniendo resultados satisfactorios al recibir los caracteres esperados para cada acción, permitiendo con esto pasar a la prueba final de integración del sistema.

Apéndice G
Tablas de Costos Detallado

Unidad de Transmisión Inalámbrica

Cantidad	Descripción	Precio Unitario (U$$)	Total (Bs.)
1	Módem AnyData EMII-800 tecnología 1xRTT CDMA2000	200	569.400,00
		Total Bs.	569.400,00

Tabla 1.- Costo de materiales para Unidad de Transmisión Inalámbrica

Unidad de Control de Intersección

Cantidad	Descripción	Precio Unitario(U$$)	Total (Bs.)
1	KIT para desarrollo para Microcontrolador	150,00	427.050,00
		Total Bs.	427.050,00

Tabla 2.- Costo de materiales para Unidad de Control de Intersección

En la tabla 3 se está considerando el costo de materiales de todos los circuitos impresos que fueron necesarios fabricar.

Circuitos impresos

Cantidad	Descripción	Precio Unitario	Total (Bs.)
8	Papel Transfer	6.000,00	48.000,00
2	Cloruro Férrico (Lt.)	15.904,00	31.808,00
2	Mecha 0,80'	1.000,00	2.000,00
2	Mecha 0,130'	1.000,00	2.000,00
1	Marcador de acetato	2.800,00	2.800,00
1	Transparencia	500,00	500,00
20	Estaño (m)	1.000,00	20.000,00
5824	Lamina de fibra de vidrio recubierta con cobre (1 cara, cm.)	12,00	69.888,00
4800	Lamina de fibra de vidrio recubierta con cobre (doble cara, cm.)	12,75	61.200,00
		Total Bs.	238.196,00

Tabla 3.- Costo de materiales para la construcción de circuitos impresos

Tarjeta de Alimentación

Cantidad	Descripción	Precio Unitario	Total (Bs.)
6	Conector molex una hilera 0,156'	3.150,00	18.900,00
8	Conector molex doble hilera 0,100'	1.500,00	12.000,00
2	Jack DC	1.243,86	2.487,72
2	Plug DC	513,16	1.026,32
1	Fuente alimentación DC/DC (U$$)	40,00	113.880,00
		Total Bs.	148.294,04

Tabla 4.- Costo de materiales para Tarjeta de Alimentación

Unidad de Control de Semáforo

Cantidad	Descripción	Precio Unitario	Total (Bs.)
1	Base P/IC 40 PIN	1.000,00	1.000,00
5	Bloques terminales de 3 contactos	1.200,00	6.000,00
2	Capacitor de 22pf	131,58	263,16
2	Conector molex doble hilera 0,100'	1.500,00	3.000,00
1	Conector molex una hilera 0,156'	3.150,00	3.150,00
1	Cristal3.579 Mhz	3.730,70	3.730,70
7	Diodo IN4148	220,00	1.540,00
2	Led	418,00	836,00
1	Pic 16F877A Microchip	25.000,00	25.000,00
7	Rele 5 Vdc	2.631,58	18.421,06
7	Resistencia	200,00	1.400,00
		Total Bs.	64.340,92

Tabla 5.- Costo de materiales para Unidad de Control de Semáforo

Prototipo de semáforo vehicular

Cantidad	Descripción	Precio Unitario	Total (Bs.)
1	Bloques terminales de 3 contactos	1.200,00	1.200,00
32	Led Rojo	500,00	16.000,00
20	Led Amarillo	418,00	8.360,00
39	Led Verde	300,00	11.700,00
5	Resistencia	671,89	3.359,45
		Total Bs.	40.619,45

Tabla 6.- Costo de materiales para Prototipo de semáforo vehicular

Prototipo de semáforo peatonal

Cantidad	Descripción	Precio Unitario	Total (Bs.)
1	Bloques terminales de 3 contactos	1.200,00	1.200,00
20	Led Rojo	500,00	10.000,00
21	Led Verde	300,00	6.300,00
2	Resistencia	671,89	1.343,78
		Total Bs.	18.843,78

Tabla 7.- Costo de materiales para Prototipo de semáforo peatonal

Apéndice H
Glosario

A

Accionamiento: Acción de un vehículo o peatón que ocasiona que un detector produzca una llamada en una fase.

B

Bit: (Binary Digit), dígito binario que adquiere el valor de 0 o 1

Byte: combinación consecutiva de 8 bits

Botón para Peatones: Dispositivo conectado a un semáforo mediante el cual los peatones pueden solicitar la indicación para cruzar una vía.

Botón para Alarmas: Dispositivo conectado a un semáforo mediante el cual brinda la posibilidad de notificar a la central sobre alguna situación de emergencia en la via.

C

Canal de Bajada: Las señales de información se originan principalmente desde la central teniendo como destino la Unidad de Control de Semáforo (UCS).

Canal de Subida: Las señales de información se originan desde la unidad de control de semáforo (UCS), por medio de los dos botones manuales para cada semáforo, teniendo como destino la USC.

CECOTA: Centro de Control Inteligente de Tránsito Inteligente

Central: computadora o grupo de estas, considerado como el cerebro de PISACOTA encargado de las simulación, recepción, transmisión y procesamiento de datos recolectados por otros subsistemas, así como de ejecutar algoritmos de análisis estadísticos. Siendo así la encargada de generar y enviar las secuencias a cada intersección.

CDMA2000: Es un método de transmisión móvil celular de espectro extendido que permite a varios usuarios compartir el mismo espectro de radiofrecuencia por asignación de un código único a cada usuario activo

CIDI: Centro de Investigación y Desarrollo de Ingeniería

Circuito Impreso: Circuito eléctrico fabricado depositando material conductor sobre la superficie de una base aislante.

Conectores: Acoplador para unir cables o conectar un dispositivo.

Controlador de Semáforo: combinación de software y hardware que permite operar una intersección de semáforos con valores predeterminados, que pueden ser de: duración fija de ciclos, duración fija de intervalos y/o secuencias de intervalos .

D

Desfase: Relación en tiempo, expresada en segundos o en porcentaje de ciclo, determinada por la diferencia entre un punto del ciclo y un punto de referencia del mismo.

E

EAADT: Unidad de Transmisión Inalámbrica para Estaciones Automatizadas de Adquisición de Datos de Tráfico

G

Gabinete: Un cajetín exterior para contener la unidad de control y sus equipos asociados.

GSM: es un estándar para teléfonos móviles desarrollado por la ETS.

I

IATTC: Instituto Autónomo de Tránsito, Transporte y Circulación del Municipio Chacao

Intermitencia: Es el estado de un controlador de semáforos en el cual se irrumpe la secuencia normal de operación, desplegándose una cierta combinación de los semáforos con una frecuencia predeterminada.

Interrupciones: Eventos que se ejecutan en la programación, debido a un conjunto de condiciones predeterminadas.

M

Microntrolador: circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado. Sus prestaciones son limitadas. pero su característica principal es su alto nivel de especialización.

MPLAP: Ambiente de programación usado para la programación del PIC

P

PIC: Circuito integrado o chip que incluye de forma interna las tres unidades funcionales de una computadora: CPU, memoria y Unidades de E/S

Pin: Contacto que sobresale de todo conector macho, también llamado terminal, es cada uno de los contactos terminales de un conector o componente electrónico, fabricado de un material conductor de la electricidad.

PISACOTA: Plataforma Integrada para el Seguimiento Análisis y Control del Tráfico Automotor

R

Relé: dispositivo destinado a producir en un circuito una modificación dada, cuando se cumplen determinadas condiciones.

S

SEMAVENCA: Empresa Venezolana dedicada al desarrollo de tecnología electrónica, enfocada principalmente a los controladores de tráfico.

T

TDMA: *Time Division Multiple Access*, o acceso múltiple por división de tiempo, es una tecnología inalámbrica de segunda generación para teléfonos móviles celulares

U

UCI: Unidad de Control de Intersección

UCS: Unidad de Control de Semáforo

USC: Unidad de Simulación de Central

UTI: Unidad de Transmisión Inalámbrica

V

Voltaje: diferencia de potencial entre dos puntos de un campo eléctrico, es igual al trabajo que realiza dicha unidad de carga positiva para transportarla desde un punto al otro.

W

Wi-Fi: es un conjunto de estándares para redes inalámbricas basado en las especificaciones IEEE 802.11, y representa a su vez una marca de la *Wi-Fi Alliance* (anteriormente la *Wireless Ethernet Compatibility Alliance*), la

organización comercial que prueba y certifica que los equipos cumplen los estándares IEEE 802.11x.

Wi-Max: es una tecnología basada en el estándar 802.16, que ofrece conectividad para redes metropolitanas inalámbricas de banda ancha.

Anexo A
Extracto de la Norma Venezolana. COVENIN (2753:1999)

NORMA VENEZOLANA

COVENIN 2753:1999

EQUIPOS DE CONTROL DE SEMÁFOROS.

1era Revisión

CODELECTRA

COMITE DE ELECTRICIDAD DE VENEZUELA

Fondo para la Normalización y Certificación de la Calidad

FONDONORMA

PROLOGO

La presente norma sustituye totalmente a la norma venezolana COVENIN **2753: 1999 Equipos de control de semáforos**, fue revisada de acuerdo a las directrices del Comité Técnico de Normalización **CT-11 Electricidad Electrónica y Comunicaciones**, por el **SC-2 Electrónica**, a través del convenio para la elaboración de normas suscrito entre **CODELECTRA y FONDONORMA**, siendo aprobada por **FONDONORMA** en la reunión del Consejo Superior **N° 99-11** de fecha **20/10/1999**.

En la revisión de está Norma participaron las siguientes entidades:

ALCALDÍA DE CHACAO
CONSESE
DELCAN
FONTUR
GRUPO 1.994, C.A.
INVICTA ELECTRÓNICA
M.T.C.
M.T.C. DIR. DE VIGILANCIA
SEMAVENCA
U.S.B.
VELECTRA

NORMA VENEZOLANA
EQUIPOS DE CONTROL DE SEMÁFOROS

COVENIN 2753:1999 (1ª Revisión.)

1 OBJETO

1.1 Esta norma venezolana establece los requisitos mínimos y métodos de ensayo que deben cumplir los equipos controladores de semáforos a ser utilizados en intersecciones viales señalizadas por semáforos.

1.2 La presente norma Venezolana tiene aplicabilidad a equipos controladores de semáforos. Esto incluye el gabinete, la unidad de control, los circuitos de conmutación de potencia, los circuitos de supervisión contra conflictos, los supresores de transitorios. Y los elementos de comunicación.

2 REFERENCIAS NORMATIVAS

2.1 Las siguientes normas contienen disposiciones que al ser citadas en este texto, constituyen requisitos de esta Norma Venezolana. Las ediciones indicadas estaban en vigencia en el momento de esta publicación. Como toda norma está sujeta a revisión, se recomienda a aquellos que realicen acuerdos en base a ellas, que analicen la conveniencia de usar las ediciones más recientes de las normas citadas.

2.2 NORMAS COVENIN:

COVENIN 200:1999	Código Eléctrico Nacional.
COVENIN 395:-1973	Ensayos fundamentales climáticos y de robustez mecánica para los equipos y componentes electrónicos. Parte 2: Ensayos. Ensayo B: Ensayo de calor seco.
COVENIN 405:1980	Ensayos fundamentales climáticos y de robustez mecánica para los equipos y componentes electrónicos. Parte 2: Ensayos. Ensayo Ca: Ensayo de calor húmedo continuo.
COVENIN 540:1998	Grados de protección proporcionados por las envolventes (cajas y gabinetes) utilizados en media y baja tensión (Código IP).
COVENIN 717:-1974	Ensayos fundamentales climáticos y de robustez mecánica para los equipos y componentes electrónicos. Parte 2: Ensayos. Ensayo Fc: Vibración (sinosoidales).

3 DEFINICIONES

3.1 ACCIONAMIENTO (ACTUATION)

Acción de un vehículo o un peatón que causa que un detector produzca una llamada en una fase.

3.2 ANILLO (RING)

En un semáforo accionado por el tránsito, sistema de indicaciones que gobiernan movimientos conflictivos y que se exhiben consecutivamente.

3.3 AVANCE DE INTERVALO (INTERVAL ADVANCE)

Comando que ocasiona la terminación inmediata del intervalo que está operando.

3.4 BOTÓN DE PULSADO MANUAL (MANUAL PUSH-BOTTON)

Dispositivo auxiliar para la operación manual de un controlador.

3.5 BOTÓN PARA PEATONES (PEDESTRIAN PUSH-BOTTON)

Dispositivo conectado a un semáforo mediante el cual los peatones pueden solicitar la indicación para cruzar una vía.

3.6 BRECHA (GAP)

Tiempo que transcurre entre el paso, por un punto fijo de una vía, del extremo trasero de un vehículo y el delantero del vehículo que lo sigue.

3.7 BRECHA MÁXIMA ACEPTABLE (ALLOWABLE GAP)

Brecha entre dos vehículos consecutivos que determina, en un valor superior a ella, cuando debe terminar el verde de una fase para ceder el derecho de paso a otra fase.

b) Duración fija de intervalo(s)
c) Secuencia(s) de intervalos

3.23 CONTROLADOR DE UN ANILLO (SINGLE-RING CONTROLLER)

Unidad de control que contiene dos o más fases temporizadas y seleccionadas individualmente y arregladas para ocurrir en un orden preestablecido.

3.24 CONTROLADOR MULTI-ANILLO (MULTI-RING CONTROLLER)

Unidad de control que contiene dos o más anillos entrelazados que se acomodan para temporizar en una secuencia preferida y para permitir temporizar en forma concurrente todos los anillos, sujeto a las restricciones de la línea de compatibilidad (barrera).

3.25 CONTROLADOR ESCLAVO (SLAVE CONTROLLER)

Unidad de control que opera los semáforos bajo la supervisión de un controlador maestro.

3.26 CONTROLADOR MAESTRO (MASTER CONTROLLER)

Unidad de control que opera los semáforos y proporciona la supervisión de otros controladores secundarios.

3.27 CONTROLADOR SEMI-ACCIONADO (SEMI-ACTUATED CONTROLLER)

Unidad de control donde se proporciona el accionamiento de uno o varios, pero no de todos los accesos de la intersección.

3.28 CONTROLADOR TOTALMENTE ACCIONADO (FULLY-ACTUATED CONTROLLER)

Unidad de control donde se proporciona el accionamiento de todos los accesos de la intersección.

3.29 CONMUTADOR DE LUCES (LOAD SWITCH)

Dispositivo usado para impartir energía a las luces de los semáforos.

3.30 COORDINACIÓN (COORDINATION)

Control de dos o más controladores de manera de proporcionar una relación entre verdes específicos de intersecciones adyacentes, en un periodo programado, para permitir la operación continua de grupos de vehículos sobre una vía a una determinada velocidad.

3.31 COORDINACIÓN EN BASE A TIEMPO (TIME-BASED COORDINATION)

Técnica de coordinación de controladores de semáforos que cambia los planes de temporización según reloj interno.

3.32 COORDINADOR (COORDINATOR)

Dispositivo, programa o rutina que proporciona coordinación.

3.33 DENSIDAD (DENSITY)

Medida de la concentración de vehículos, expresada en número de vehículos por kilómetro.

3.34 DESCANSO (REST)

Estado en el cual permanece el controlador al terminar la temporización de una llamada o de una fase.

3.35 DESCANSO EN ROJO (RED REST)

Opción mediante la cual, en la ausencia de demanda, el controlador regresará a un todo rojo en vez de permanecer en el verde de la última fase servida.

3.36 DESFASE (OFFSET)

Relación en tiempo, expresada en segundos o en porcentaje del ciclo, determinada por la diferencia entre un punto definido en el intervalo verde coordinado y un punto de referencia del sistema (tiempo cero del sistema).

3.37 DETECCIÓN (DETECTION)

Proceso de identificar el paso o la presencia de un vehículo en un punto específico o la presencia de uno o más vehículos en un área específica.

3.38 DETECCIÓN MEMORIZADA (LOCKING DETECTION)

Opción del circuito de diseño de una fase del controlador donde se registra la llamada de un vehículo que llega en el rojo (o amarillo), siendo la misma mantenida por el controlador después de que el vehículo abandona el área de detección y hasta que la llamada se satisface con una indicación de verde para dicha fase.

3.160 TIEMPO DE SOSTENIMIENTO DE PRESENCIA (PRESENCE HOLDING TIME)

Tiempo que un sistema de detección continuará indicando la presencia de un vehículo sobre el lazo del detector sin ningún ajuste para considerar que el vehículo ha salido de la zona de detección. Una vez hecho el ajuste, se da por terminado el accionamiento.

3.161 TIEMPO PARA REDUCIR (TIME TO REDUCE – TTR)

Durante el proceso de Reducción de Brecha, período de tiempo total para que el proceso de reducción de brecha tenga lugar.

3.162 TRANSICIÓN (TRANSITION)

Proceso mediante el cual el computador y los controladores locales de las intersecciones cambian de un programa a otro.

3.163 UNIDAD DEL DETECTOR (DETECTOR UNIT)

Dispositivo electrónico que es capaz de energizar un detector de lazo, de supervisar la inductancia del detector, y de responder a una disminución predeterminada en inductancia con una salida que indica el paso o la presencia de vehículos en la zona de detección. Es la unidad electrónica excluyendo los lazos y el cable al detector o detector. Existen dos tipos de unidades dependiendo del tipo de montaje en el gabinete:

a) De tarjeta (card-mounted): no tiene envoltura y se coloca insertado la tarjeta en la ranura correspondiente.
b) Con envoltura (shelf-mounted): la unidad electrónica viene protegida por una envoltura y se coloca directamente sobre un estante del gabinete.

3.164 VERDE MÁXIMO (MAXIMUM GREEN)

En un semáforo accionado por el tránsito, máxima duración que se permite a un intervalo verde cuando se ha recibido un accionamiento solicitando derecho de paso para una fase conflictiva.

3.165 VERDE MÍNIMO (MINIMUM GREEN)

Tiempo mínimo de verde garantizado de una fase. Si se coloca un tiempo en el control que se establece como verde mínimo, el tiempo de verde no deberá ser menor que dicho valor.

3.166 VERIFICACIÓN DE LLAMADA (CHECK)

Salida del controlador que indica la existencia de una llamada no servida.

3.167 VOLUMEN (VOLUME)

Número de vehículos que pasan por un punto de una vía en un período de tiempo determinado.

3.168 VOLUMEN-DENSIDAD (VOLUME-DENSITY)

Proceso usado con detectores colocados a cierta distancia de la intersección, y que hace uso del número de accionamientos vehiculares y del tiempo en espera de los vehículos para variar las porciones de los intervalos de verde, con el fin de aumentar la capacidad y minimizar demoras.

4. MATERIALES, DISEÑO Y FABRICACIÓN

4.1 MATERIALES

4.1.1 UNIDAD DE CONTROL

4.1.1.1 Las tarjetas de circuito impreso deberán estar elaboradas con láminas de fibra de vidrio epóxica o su equivalente, retardante a la flama.

4.1.2 GABINETE

4.1.2.1 El gabinete debe estar elaborado en lámina de acero tratado o en aluminio moldeado.

4.2 DISEÑO Y FABRICACIÓN

4.2.1 UNIDAD DE CONTROL

4.2.1.1 La unidad de control debe ser de diseño modular, mediante tarjetas desenchufables y fácilmente accesible para mantenimiento.

4.2.1.2 Las tarjetas de circuito impreso deben tener un espesor nominal no inferior a 1.5 mm y laminado de cobre no inferior a 55 μm.

4.2.1.3 Todos los huecos pasantes destinados para conexión eléctrica en las tarjetas de más de una cara, deberán ser metalizados, con un espesor mínimo de 25 μm de cobre.

4.2.1.4 Todas las superficies de contacto eléctrico, así como todos los caminos de conducción deben ser estañados con una capa conductora resistente a la corrosión con un espesor mínimo de 8 μm. Así mismo las

5.4.3.8.2 **Intermitencia de inicio**

5.4.3.8.2.1 El controlador debe proporcionar un período inicial ajustable de intermitencia entre 0 y 255 segundos, con incrementos de un (1) segundo, que debe ocurrir antes de la rutina de inicialización.

5.4.3.8.2.2 El periodo de intermitencia debe iniciarse cuando se restablezca la energía al controlador después de una interrupción de la misma. Este periodo no puede reducirse por ningún comando externo salvo por otra interrupción de energía.

5.4.3.8.2.3 Si existe un comando de cambio por prioridad durante la temporización del periodo de intermitencia inicial, el controlador mantendrá el periodo de intermitencia inicial por toda la duración de la demanda de cambio pro prioridad.

5.4.3.8.3 **Intermitencia programada**

5.4.3.8.3.1 Antes de iniciar la operación de intermitencia programada en un controlador accionado, se debe completar el tiempo mínimo de verde (o caminar más el despeje peatonal) de la fase activa para luego, en caso de ser necesario, temporizar los intervalos de despeje vehicular. Al finalizar el intervalo de despeje rojo (o final del ciclo), el controlador podrá temporizar una fase de entrada programada para entrar en modo de intermitencia por el tiempo establecido.

5.4.3.8.3.2 Una vez terminada la intermitencia programada, el controlador temporizará inmediatamente una fase programada de salida con las indicaciones de verde y caminar del primer intervalo del ciclo haciendo llamadas a todas las fases vehiculares y peatonales.

5.4.3.8.3.3 Antes de iniciar la operación de intermitencia, en un controlador de tiempo fijo, el controlador temporizará el intervalo programado de entrada y una vez completada la temporización, iniciará la intermitencia por el tiempo establecido. Una vez terminada la intermitencia programada, el controlador temporizará inmediatamente un intervalo programado de salida y después continuará temporizando los demás intervalos.

5.4.3.8.4 **Intermitencia por falla o conflicto**

5.4.3.8.4.1 En caso de producirse una falla en el equipo de control o detectar un conflicto por el monitor de conflicto malfuncionamiento, el controlador deberá transferir inmediatamente la operación de los semáforos a intermitencia.

5.4.4 **OPERACIÓN BAJO UN SISTEMA DE CONTROL**

5.4.4.1 En operación bajo un sistema de control, el controlador funciona coordinadamente con otros controladores, pudiendo recibir comandos desde la computadora del sistema de control central.

5.4.4.2 Bajo este modo de funcionamiento, el controlador puede operar de la siguientes maneras:

- Selección manual del modo de operación;
- Operación en base a tiempo;
- Operación interconectado (con controlador maestro);
- Operación por computadora.

5.4.4.3 **Selección Manual Del Modo De Operación**

5.4.4.3.1 Normalmente, en operación bajo un sistema de control, el controlador funciona recibiendo los comandos automáticamente de la computadora central o del controlador maestro. Sin embargo, para el tratamiento de situaciones anormales, tales como accidentes u otras ocasiones especiales, el controlador debe incluir las facilidades para permitir una operación manual de la secuencia de las fases.

5.4.4.3.2 Los siguientes modos de operación del controlador pueden ser seleccionada manualmente:

- Operación en base a tiempo;
- Operación accionada por el tránsito;
- Operación en tiempo fijo;
- Selección manual de las fases.

5.4.4.3.3 El controlador debe poder funcionar bajo cualquier plan de temporización y desfase que se haya seleccionado manualmente. Una selección manual del plan de temporización supersede cualquier otro comando para la selección de planes.

5.4.4.3.4 Cuando se opere bajo control manual, las fases terminarán tomando en consideración de que se hayan satisfecho los periodos mínimos de las fases involucradas, teniendo cuidado que no se sobrepasen las restricciones de secuencia de fases definidas para este modo de operación.

5.4.4.4 **Operación en Base A Tiempo**

5.4.4.4.1 El controlador funcionará en este modo de operación en base a la hora de su reloj interno. El reloj debe mantener una precisión mínima de dos (2) minutos por mes, con respecto al Tiempo Coordinado Universal, para el rango de temperatura de funcionamiento del

controlador. El reloj deberá mantener esta precisión por un período de treinta días donde puedan existir cortes de energía.

5.4.4.4.2 El control en base a tiempo debe ser un programa especial operando en el controlador. Deben ser posible programar un mínimo de 48 eventos en un año calendario y un mínimo de 12 eventos diarios. El calendario del reloj debe compensar por años bisiestos.

5.4.4.4.3 Bajo este modo de operación, el controlador proporciona la temporización de las fases y los desfases para la operación coordinada de los semáforos. El tiempo de referencia (punto cero) debe ser programable para referenciarlo a cualquier hora/minuto o al tiempo de un evento.

5.4.4.4.4 La pantalla del controlador debe proporcionar indicaciones que permiten identificar adecuadamente este modo de operación. Como mínimo, el controlador deberá indicar su estado actual de operación, destacando:

- Fecha, hora y día de la semana;
- Evento seleccionado.

5.4.4.4.5 Al ocurrir un cambio de plan, en este modo de operación, el plan en vigencia se terminará en la fase activa, y el nuevo plan deberá iniciarse con su primera fase.

5.4.4.4.6 Debe ser posible para el controlador buscar informaciones en una tabla de eventos y esas informaciones deben ser programadas a través de un interfaz. Además, debe ser posible analizar cada registro antes de ingresarlo a la tabla de eventos o anularlo. También, debe ser posible visualizar cada plan, de tal forma que al regresar al plan original no se pierda la coordinación del controlador con el resto del sistema.

5.4.4.4.7 Las condiciones de cambio de plan, de buscada de información en la tabla de eventos y visualización de los planes también son aplicables a la operación de controladores interconectados con un controlador maestro.

5.4.4.5 **Operación Interconectado (Con Maestro)**

5.4.4.5.1 Este modo de operación permite coordinar semáforos a lo largo de una arteria o en un área, utilizando para ello la cronometría del controlador maestro mediante su interconexión con los demás controladores del sistema. Todos los controladores de un sistema coordinado son operados a un ciclo común (o submúltiplo de éste) para cada plan particular. Básicamente, el controlador maestro genera un pulso de sincronización como punto de referencia para los desfases en cada controlador, y para el cambio de un plan al siguiente si se requiere.

5.4.4.6 **Operación Por Computadora**

5.4.4.6.1 Bajo esta modalidad de operación, el controlador será comandado remotamente por una computadora central vía una red de transmisión de datos. La unidad de comunicación debe venir integrada en el controlador.

5.4.4.6.2 La unidad de comunicación debe ser alimentada de la misma fuente que la lógica del controlador y debe existir un interfaz de estado sólido al microprocesador, en contraposición al interfaz de tipo aislado que se emplea en el caso de la unidad de comunicación separada.

5.5 SISTEMAS AUXILIARES

5.5.1 Los sistemas auxiliares del controlador incluyen:

- Los detectores;
- Las facilidades para semáforos peatonales;
- El sistema de coordinación por reloj maestro;
- Las facilidades de emergencia;
- La alimentación eléctrica;
- El monitor de conflicto o de malfuncionamiento.

5.5.2 DETECTORES

5.5.2.1 El equipo de detección o detectores, que sea vehicular o peatonal, es una unidad lógica independiente, la cual puede ser integrada en el controlador o ser instalada como una unidad independiente del mismo. Los detectores tienen como finalidad detectar e indicar la controlador la presencia y paso de vehículos o indicar la presencia de peatones.

5.5.2.2 Las salidas de los detectores están destinadas a influenciar la operación del controlador y generando cambios de fases y/o extensiones del derecho de paso. Tanto las características técnicas de estos equipos como su instalación serán motivo de otras normas independientes de ésta, sin perjuicio de los detalles cubiertos en esta norma.

5.5.2.3 Las demandas peatonales son registradas en base a botones peatonales cuya actuación establece una demanda por la fase adecuada.

5.5.2.4 Los detectores de vehículos están asociados con movimientos vehiculares y permiten:

- Llamar una fase;
- Extender un intervalo;
- Generar una demanda de cambio por prioridad;

Anexo B
Extracto de la hoja de información del microcontrolador C8051F120

C8051F120/1/2/3/4/5/6/7
C8051F130/1/2/3

Mixed Signal ISP Flash MCU Family

Analog Peripherals
- **10 or 12-bit SAR ADC**
  - ± 1 LSB INL
  - Programmable throughput up to 100 ksps
  - Up to 8 external inputs; programmable as single-ended or differential
  - Programmable amplifier gain: 16, 8, 4, 2, 1, 0.5
  - Data-dependent windowed interrupt generator
  - Built-in temperature sensor
- **8-bit SAR ADC ('F12x Only)**
  - Programmable throughput up to 500 ksps
  - 8 external inputs (single-ended or differential)
  - Programmable amplifier gain: 4, 2, 1, 0.5
- **Two 12-bit DACs ('F12x Only)**
  - Can synchronize outputs to timers for jitter-free waveform generation
- **Two Analog Comparators**
- **Voltage Reference**
- **V_{DD} Monitor/Brown-Out Detector**

On-Chip JTAG Debug & Boundary Scan
- On-chip debug circuitry facilitates full-speed, non-intrusive in-circuit/in-system debugging
- Provides breakpoints, single-stepping, watchpoints, stack monitor; inspect/modify memory and registers
- Superior performance to emulation systems using ICE-chips, target pods, and sockets
- IEEE1149.1 compliant boundary scan
- Complete development kit

100-Pin TQFP or 64-Pin TQFP Packaging
- Temperature Range: –40 to +85 °C
- RoHS Available

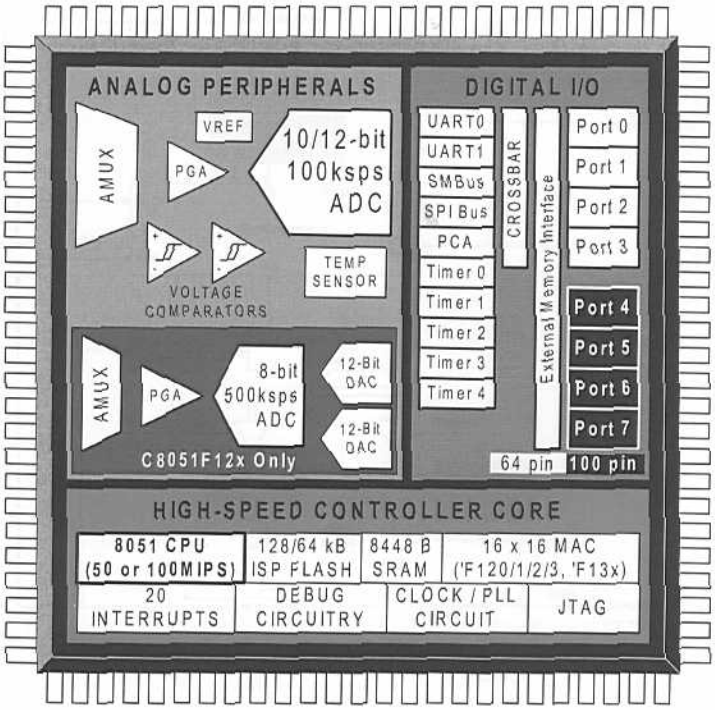
High Speed 8051 µC Core
- Pipelined instruction architecture; executes 70% of instruction set in 1 or 2 system clocks
- 100 MIPS or 50 MIPS throughput with on-chip PLL
- 2-cycle 16 x 16 MAC engine (C8051F120/1/2/3 and C8051F130/1/2/3 only)

Memory
- 8448 bytes internal data RAM (8 k + 256)
- 128 or 64 kB Banked Flash; in-system programmable in 1024-byte sectors
- External 64 kB data memory interface (programmable multiplexed or non-multiplexed modes)

Digital Peripherals
- 8 byte-wide port I/O (100TQFP); 5 V tolerant
- 4 Byte-wide port I/O (64TQFP); 5 V tolerant
- Hardware SMBus™ (I2C™ Compatible), SPI™, and two UART serial ports available concurrently
- Programmable 16-bit counter/timer array with 6 capture/compare modules
- 5 general purpose 16-bit counter/timers
- Dedicated watchdog timer; bi-directional reset pin

Clock Sources
- Internal precision oscillator: 24.5 MHz
- Flexible PLL technology
- External Oscillator: Crystal, RC, C, or clock

Voltage Supples
- Range: 2.7–3.6 V (50 MIPS) 3.0–3.6 V (100 MIPS)
- Power saving sleep and shutdown modes



21. UART0

UART0 is an enhanced serial port with frame error detection and address recognition hardware. UART0 may operate in full-duplex asynchronous or half-duplex synchronous modes, and mutiproccessor communication is fully supported. Receive data is buffered in a holding register, allowing UART0 to start reception of a second incoming data byte before software has finished reading the previous data byte. A Receive Overrun bit indicates when new received data is latched into the receive buffer before the previously received byte has been read.

UART0 is accessed via its associated SFR's, Serial Control (SCON0) and Serial Data Buffer (SBUF0). The single SBUF0 location provides access to both transmit and receive registers. Reading SCON0 accesses the Receive register and writing SCON0 accesses the Transmit register.

UART0 may be operated in polled or interrupt mode. UART0 has two sources of interrupts: a Transmit Interrupt flag, TI0 (SCON0.1) set when transmission of a data byte is complete, and a Receive Interrupt flag, RI0 (SCON0.0) set when reception of a data byte is complete. UART0 interrupt flags are not cleared by hardware when the CPU vectors to the interrupt service routine; they must be cleared manually by software. This allows software to determine the cause of the UART0 interrupt (transmit complete or receive complete).

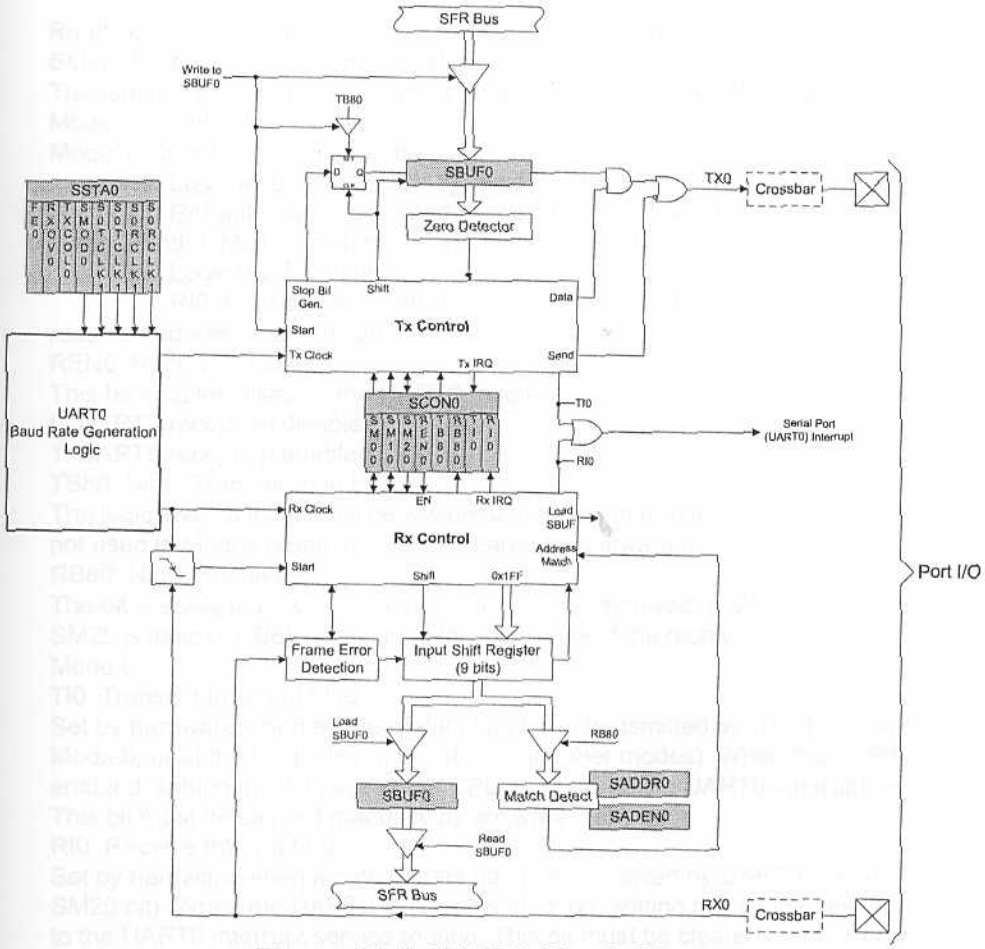


Figure 21.1. UART0 Block Diagram

SFR Definition 21.1. SCON0: UART0 Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SM00	SM10	SM20	REN0	TB80	RB80	TI0	RI0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable

SFR Address: 0x98
SFR Page: 0

Bits7–6: SM00–SM10: Serial Port Operation Mode:
Write:
When written, these bits select the Serial Port Operation Mode as follows:

SM00	SM10	Mode
0	0	Mode 0: Synchronous Mode
0	1	Mode 1: 8-Bit UART, Variable Baud Rate
1	0	Mode 2: 9-Bit UART, Fixed Baud Rate
1	1	Mode 3: 9-Bit UART, Variable Baud Rate

Reading these bits returns the current UART0 mode as defined above.

Bit5: SM20: Multiprocessor Communication Enable.
The function of this bit is dependent on the Serial Port Operation Mode.
Mode 0: No effect
Mode 1: Checks for valid stop bit.
0: Logic level of stop bit is ignored.
1: RI0 will only be activated if stop bit is logic level 1.
Mode 2 and 3: Multiprocessor Communications Enable.
0: Logic level of ninth bit is ignored.
1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1 and the received address matches the UART0 address or the broadcast address.

Bit4: REN0: Receive Enable.
This bit enables/disables the UART0 receiver.
0: UART0 reception disabled.
1: UART0 reception enabled.

Bit3: TB80: Ninth Transmission Bit.
The logic level of this bit will be assigned to the ninth transmission bit in Modes 2 and 3. It is not used in Modes 0 and 1. Set or cleared by software as required.

Bit2: RB80: Ninth Receive Bit.
The bit is assigned the logic level of the ninth bit received in Modes 2 and 3. In Mode 1, if SM20 is logic 0, RB80 is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.

Bit1: TI0: Transmit Interrupt Flag.
Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in Mode 0, or at the beginning of the stop bit in other modes). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software

Bit0: RI0: Receive Interrupt Flag.
Set by hardware when a byte of data has been received by UART0 (as selected by the SM20 bit). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.

SFR Definition 21.2. SSTA0: UART0 Status and Clock Selection

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
FE0	RXOV0	TXCOL0	SMOD0	S0TCLK1	S0TCLK0	S0RCLK1	S0RCLK0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x91
SFR Page: 0

Bit7: FE0: Frame Error Flag.*
This flag indicates if an invalid (low) STOP bit is detected.
0: Frame Error has not been detected
1: Frame Error has been detected.

Bit6: RXOV0: Receive Overrun Flag.*
This flag indicates new data has been latched into the receive buffer before software has read the previous byte.
0: Receive overrun has not been detected.
1: Receive Overrun has been detected.

Bit5: TXCOL0: Transmit Collision Flag.*
This flag indicates user software has written to the SBUF0 register while a transmission is in progress.
0: Transmission Collision has not been detected.
1: Transmission Collision has been detected.

Bit4: SMOD0: UART0 Baud Rate Doubler Enable.
This bit enables/disables the divide-by-two function of the UART0 baud rate logic for configurations described in the UART0 section.
0: UART0 baud rate divide-by-two enabled.
1: UART0 baud rate divide-by-two disabled.

Bits3–2: UART0 Transmit Baud Rate Clock Selection Bits

S0TCLK1	S0TCLK0	**Serial Transmit Baud Rate Clock Source**
0	0	Timer 1 generates UART0 TX Baud Rate
0	1	Timer 2 Overflow generates UART0 TX baud rate
1	0	Timer 3 Overflow generates UART0 TX baud rate
1	1	Timer 4 Overflow generates UART0 TX baud rate

Bits1–0: UART0 Receive Baud Rate Clock Selection Bits

S0RCLK1	S0RCLK0	**Serial Receive Baud Rate Clock Source**
0	0	Timer 1 generates UART0 RX Baud Rate
0	1	Timer 2 Overflow generates UART0 RX baud rate
1	0	Timer 3 Overflow generates UART0 RX baud rate
1	1	Timer 4 Overflow generates UART0 RX baud rate

*Note: FE0, RXOV0, and TXCOL0 are flags only, and no interrupt is generated by these conditions.

SFR Definition 21.3. SBUF0: UART0 Data Buffer

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x99
SFR Page: 0

Bits7–0: SBUF0.[7:0]: UART0 Buffer Bits 7–0 (MSB–LSB)
This is actually two registers; a transmit and a receive buffer register. When data is moved to SBUF0, it goes to the transmit buffer and is held for serial transmission. Moving a byte to SBUF0 is what initiates the transmission. When data is moved from SBUF0, it comes from the receive buffer.

SFR Definition 21.4. SADDR0: UART0 Slave Address

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xA9
SFR Page: 0

Bits7–0: SADDR0.[7:0]: UART0 Slave Address
The contents of this register are used to define the UART0 slave address. Register SADEN0 is a bit mask to determine which bits of SADDR0 are checked against a received address: corresponding bits set to logic 1 in SADEN0 are checked; corresponding bits set to logic 0 are "don't cares".

SFR Definition 21.5. SADEN0: UART0 Slave Address Enable

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xB9
SFR Page: 0

Bits7–0: SADEN0.[7:0]: UART0 Slave Address Enable
Bits in this register enable corresponding bits in register SADDR0 to determine the UART0 slave address.
0: Corresponding bit in SADDR0 is a "don't care".
1: Corresponding bit in SADDR0 is checked against a received address.

22. UART1

UART1 is an asynchronous, full duplex serial port offering modes 1 and 3 of the standard 8051 UART. Enhanced baud rate support allows a wide range of clock sources to generate standard baud rates (details in **Section "22.1. Enhanced Baud Rate Generation" on page 300**). Received data buffering allows UART1 to start reception of a second incoming data byte before software has finished reading the previous data byte.

UART1 has two associated SFRs: Serial Control Register 1 (SCON1) and Serial Data Buffer 1 (SBUF1). The single SBUF1 location provides access to both transmit and receive registers. Reading SBUF1 accesses the buffered Receive register; writing SBUF1 accesses the Transmit register.

With UART1 interrupts enabled, an interrupt is generated each time a transmit is completed (TI1 is set in SCON1), or a data byte has been received (RI1 is set in SCON1). The UART1 interrupt flags are not cleared by hardware when the CPU vectors to the interrupt service routine. They must be cleared manually by software, allowing software to determine the cause of the UART1 interrupt (transmit complete or receive complete).

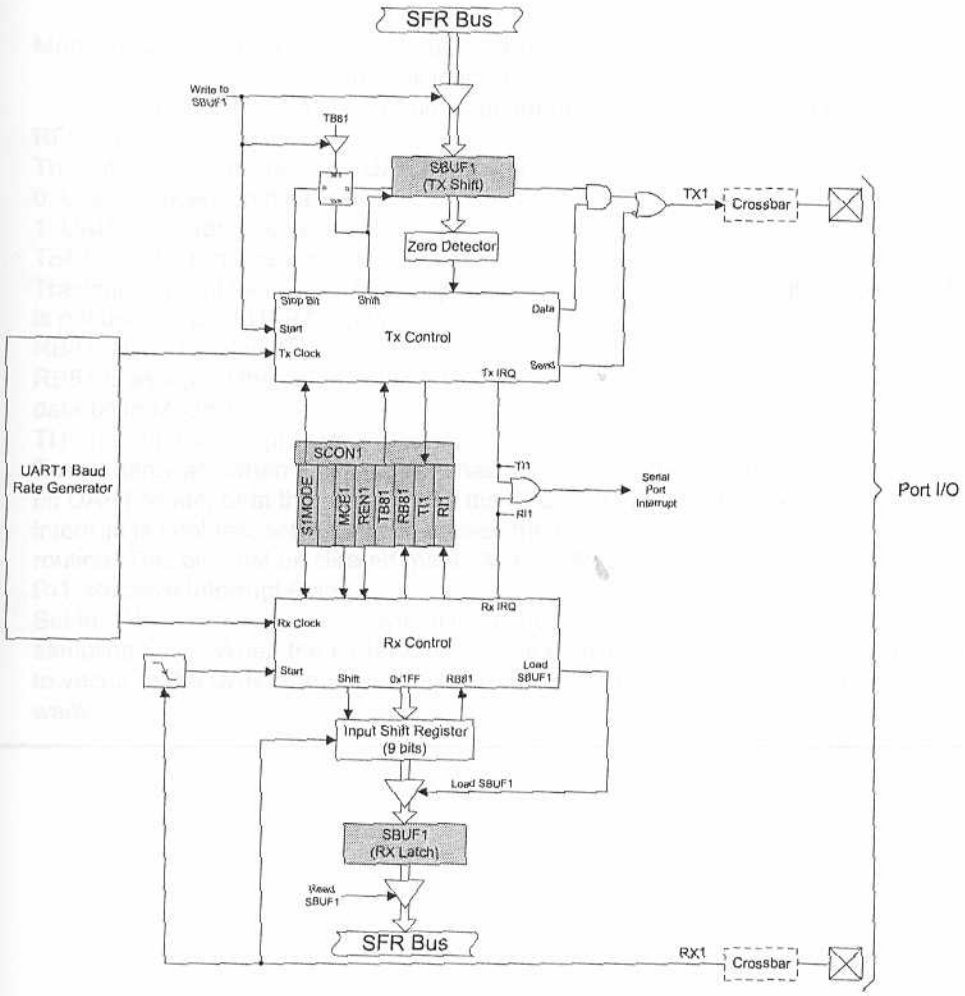


Figure 22.1. UART1 Block Diagram

SFR Definition 22.1. SCON1: Serial Port 1 Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
S1MODE	-	MCE1	REN1	TB81	RB81	TI1	RI1	01000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable

SFR Address: 0x98
SFR Page: 1

Bit7: S1MODE: Serial Port 1 Operation Mode.
This bit selects the UART1 Operation Mode.
0: Mode 0: 8-bit UART with Variable Baud Rate
1: Mode 1: 9-bit UART with Variable Baud Rate

Bit6: UNUSED. Read = 1b. Write = don't care.

Bit5: MCE1: Multiprocessor Communication Enable.
The function of this bit is dependent on the Serial Port 0 Operation Mode.
Mode 0: Checks for valid stop bit.
0: Logic level of stop bit is ignored.
1: RI1 will only be activated if stop bit is logic level 1.
Mode 1: Multiprocessor Communications Enable.
0: Logic level of ninth bit is ignored.
1: RI1 is set and an interrupt is generated only when the ninth bit is logic 1.

Bit4: REN1: Receive Enable.
This bit enables/disables the UART receiver.
0: UART1 reception disabled.
1: UART1 reception enabled.

Bit3: TB81: Ninth Transmission Bit.
The logic level of this bit will be assigned to the ninth transmission bit in 9-bit UART Mode. It is not used in 8-bit UART Mode. Set or cleared by software as required.

Bit2: RB81: Ninth Receive Bit.
RB81 is assigned the value of the STOP bit in Mode 0; it is assigned the value of the 9th data bit in Mode 1.

Bit1: TI1: Transmit Interrupt Flag.
Set by hardware when a byte of data has been transmitted by UART1 (after the 8th bit in 8-bit UART Mode, or at the beginning of the STOP bit in 9-bit UART Mode). When the UART1 interrupt is enabled, setting this bit causes the CPU to vector to the UART1 interrupt service routine. This bit must be cleared manually by software

Bit0: RI1: Receive Interrupt Flag.
Set to '1' by hardware when a byte of data has been received by UART1 (set at the STOP bit sampling time). When the UART1 interrupt is enabled, setting this bit to '1' causes the CPU to vector to the UART1 interrupt service routine. This bit must be cleared manually by software.

SFR Definition 22.2. SBUF1: Serial (UART1) Port Data Buffer

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x99
SFR Page: 1

Bits7–0: SBUF1[7:0]: Serial Data Buffer Bits 7-0 (MSB-LSB)
This SFR accesses two registers; a transmit shift register and a receive latch register. When data is written to SBUF1, it goes to the transmit shift register and is held for serial transmission. Writing a byte to SBUF1 is what initiates the transmission. A read of SBUF1 returns the contents of the receive latch.

Table 22.1. Timer Settings for Standard Baud Rates Using The Internal 24.5 MHz Oscillator

	Frequency: 24.5 MHz						
	Target Baud Rate (bps)	Baud Rate % Error	Oscillator Divide Factor	Timer Clock Source	SCA1-SCA0 (pre-scale select)*	T1M*	Timer 1 Reload Value (hex)
SYSCLK from Internal Osc.	230400	-0.32%	106	SYSCLK	XX	1	0xCB
	115200	-0.32%	212	SYSCLK	XX	1	0x96
	57600	0.15%	426	SYSCLK	XX	1	0x2B
	28800	-0.32%	848	SYSCLK / 4	01	0	0x96
	14400	0.15%	1704	SYSCLK / 12	00	0	0xB9
	9600	-0.32%	2544	SYSCLK / 12	00	0	0x96
	2400	-0.32%	10176	SYSCLK / 48	10	0	0x96
	1200	0.15%	20448	SYSCLK / 48	10	0	0x2B

X = Don't care
***Note:** SCA1-SCA0 and T1M bit definitions can be found in **Section 23.1**.

Table 22.2. Timer Settings for Standard Baud Rates Using an External 25.0 MHz Oscillator

	Frequency: 25.0 MHz						
	Target Baud Rate (bps)	Baud Rate % Error	Oscillator Divide Factor	Timer Clock Source	SCA1-SCA0 (pre-scale select)*	T1M*	Timer 1 Reload Value (hex)
SYSCLK from External Osc.	230400	-0.47%	108	SYSCLK	XX	1	0xCA
	115200	0.45%	218	SYSCLK	XX	1	0x93
	57600	-0.01%	434	SYSCLK	XX	1	0x27
	28800	0.45%	872	SYSCLK / 4	01	0	0x93
	14400	-0.01%	1736	SYSCLK / 4	01	0	0x27
	9600	0.15%	2608	EXTCLK / 8	11	0	0x5D
	2400	0.45%	10464	SYSCLK / 48	10	0	0x93
	1200	-0.01%	20832	SYSCLK / 48	10	0	0x27
SYSCLK from Internal Osc.	57600	-0.47%	432	EXTCLK / 8	11	0	0xE5
	28800	-0.47%	864	EXTCLK / 8	11	0	0xCA
	14400	0.45%	1744	EXTCLK / 8	11	0	0x93
	9600	0.15%	2608	EXTCLK / 8	11	0	0x5D

X = Don't care

***Note:** SCA1-SCA0 and T1M bit definitions can be found in **Section 23.1**.

Table 22.3. Timer Settings for Standard Baud Rates Using an External 22.1184 MHz Oscillator

	Frequency: 22.1184 MHz						
	Target Baud Rate (bps)	Baud Rate % Error	Oscillator Divide Factor	Timer Clock Source	SCA1-SCA0 (pre-scale select)*	T1M*	Timer 1 Reload Value (hex)
SYSCLK from External Osc.	230400	0.00%	96	SYSCLK	XX	1	0xD0
	115200	0.00%	192	SYSCLK	XX	1	0xA0
	57600	0.00%	384	SYSCLK	XX	1	0x40
	28800	0.00%	768	SYSCLK / 12	00	0	0xE0
	14400	0.00%	1536	SYSCLK / 12	00	0	0xC0
	9600	0.00%	2304	SYSCLK / 12	00	0	0xA0
	2400	0.00%	9216	SYSCLK / 48	10	0	0xA0
	1200	0.00%	18432	SYSCLK / 48	10	0	0x40
SYSCLK from Internal Osc.	230400	0.00%	96	EXTCLK / 8	11	0	0xFA
	115200	0.00%	192	EXTCLK / 8	11	0	0xF4
	57600	0.00%	384	EXTCLK / 8	11	0	0xE8
	28800	0.00%	768	EXTCLK / 8	11	0	0xD0
	14400	0.00%	1536	EXTCLK / 8	11	0	0xA0
	9600	0.00%	2304	EXTCLK / 8	11	0	0x70

X = Don't care

***Note:** SCA1-SCA0 and T1M bit definitions can be found in **Section 23.1**.

Table 22.4. Timer Settings for Standard Baud Rates Using the PLL

Frequency: 50.0 MHz						
Target Baud Rate (bps)	Baud Rate % Error	Oscillator Divide Factor	Timer Clock Source	SCA1-SCA0 (pre-scale select)*	T1M*	Timer 1 Reload Value (hex)
230400	0.45%	218	SYSCLK	XX	1	0x93
115200	-0.01%	434	SYSCLK	XX	1	0x27
57600	0.45%	872	SYSCLK / 4	01	0	0x93
28800	-0.01%	1736	SYSCLK / 4	01	0	0x27
14400	0.22%	3480	SYSCLK / 12	00	0	0x6F
9600	-0.01%	5208	SYSCLK / 12	00	0	0x27
2400	-0.01%	20832	SYSCLK / 48	10	0	0x27

X = Don't care

*Note: SCA1-SCA0 and T1M bit definitions can be found in Section 23.1.

Table 22.5. Timer Settings for Standard Baud Rates Using the PLL

Frequency: 100.0 MHz						
Target Baud Rate (bps)	Baud Rate % Error	Oscillator Divide Factor	Timer Clock Source	SCA1-SCA0 (pre-scale select)*	T1M*	Timer 1 Reload Value (hex)
230400	-0.01%	434	SYSCLK	XX	1	0x27
115200	0.45%	872	SYSCLK / 4	01	0	0x93
57600	-0.01%	1736	SYSCLK / 4	01	0	0x27
28800	0.22%	3480	SYSCLK / 12	00	0	0x6F
14400	-0.47%	6912	SYSCLK / 48	10	0	0xB8
9600	0.45%	10464	SYSCLK / 48	10	0	0x93

X = Don't care

*Note: SCA1-SCA0 and T1M bit definitions can be found in Section 23.1.

23.2. Timer 2, Timer 3, and Timer 4

Timers 2, 3, and 4 are 16-bit counter/timers, each formed by two 8-bit SFR's: TMRnL (low byte) and TMRnH (high byte) where n = 2, 3, and 4 for timers 2, 3, and 4 respectively. Timers 2 and 4 feature auto-reload, capture, and toggle output modes with the ability to count up or down. Timer 3 features auto-reload and capture modes, with the ability to count up or down. Capture Mode and Auto-reload mode are selected using bits in the Timer 2, 3, and 4 Control registers (TMRnCN). Toggle output mode is selected using the Timer 2 or 4 Configuration registers (TMRnCF). These timers may also be used to generate a square-wave at an external pin. As with Timers 0 and 1, Timers 2, 3, and 4 can use either the system clock (divided by one, two, or twelve), external clock (divided by eight) or transitions on an external input pin as its clock source. Timer 2 and 3 can be used to start an ADC Data Conversion and Timers 2, 3, and 4 can schedule DAC outputs. Timers 1, 2, 3, or 4 may be used to generate baud rates for UART 0. Only Timer 1 can be used to generate baud rates for UART 1.

The Counter/Timer Select bit C/Tn bit (TMRnCN.1) configures the peripheral as a counter or timer. Clearing C/Tn configures the Timer to be in a timer mode (i.e., the system clock or transitions on an external pin as the input for the timer). When C/Tn is set to 1, the timer is configured as a counter (i.e., high-to-low transitions at the Tn input pin increment (or decrement) the counter/timer register. Timer 3 and Timer 2 share the T2 input pin. Refer to **Section "18.1. Ports 0 through 3 and the Priority Crossbar Decoder" on page 238** for information on selecting and configuring external I/O pins for digital peripherals, such as the Tn pin.

Timer 2, 3, and 4 can use either SYSCLK, SYSCLK divided by 2, SYSCLK divided by 12, an external clock divided by 8, or high-to-low transitions on the Tn input pin as its clock source when operating in Counter/Timer with Capture mode. Clearing the C/Tn bit (TMRnCN.1) selects the system clock/external clock as the input for the timer. The Timer Clock Select bits TnM0 and TnM1 in TMRnCF can be used to select the system clock undivided, system clock divided by two, system clock divided by 12, or an external clock provided at the XTAL1/XTAL2 pins divided by 8 (see SFR Definition 23.13). When C/Tn is set to logic 1, a high-to-low transition at the Tn input pin increments the counter/timer register (i.e., configured as a counter).

23.2.1. Configuring Timer 2, 3, and 4 to Count Down

Timers 2, 3, and 4 have the ability to count down. When the timer's Decrement Enable Bit (DCENn) in the Timer Configuration Register (See SFR Definition 23.13) is set to '1', the timer can then count up or down. When DCENn = 1, the direction of the timer's count is controlled by the TnEX pin's logic level (Timer 3 shares the T2EX pin with Timer 2). When TnEX = 1, the counter/timer will count up; when TnEX = 0, the counter/timer will count down. To use this feature, TnEX must be enabled in the digital crossbar and configured as a digital input.

Note: When DCENn = 1, other functions of the TnEX input (i.e., capture and auto-reload) are not available. TnEX will only control the direction of the timer when DCENn = 1.

SFR Definition 23.8. TMRnCN: Timer 2, 3, and 4 Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TFn	EXFn	-	-	EXENn	TRn	C/Tn	CP/RLn	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable

SFR Address: TMR2CN:0xC8;TMR3CN:0xC8;TMR4CN:0xC8
SFR Page: TMR2CN: page 0;TMR3CN: page 1;TMR4CN: page 2

Bit7: TFn: Timer 2, 3, and 4 Overflow/Underflow Flag.
Set by hardware when either the Timer overflows from 0xFFFF to 0x0000, underflows from the value placed in RCAPnH:RCAPnL to 0xFFFF (in Auto-reload Mode), or underflows from 0x0000 to 0xFFFF (in Capture Mode). When the Timer interrupt is enabled, setting this bit causes the CPU to vector to the Timer interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit6: EXFn: Timer 2, 3, or 4 External Flag.
Set by hardware when either a capture or reload is caused by a high-to-low transition on the TnEX input pin and EXENn is logic 1. When the Timer interrupt is enabled, setting this bit causes the CPU to vector to the Timer Interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit5–4: Reserved.

Bit3: EXENn: Timer 2, 3, and 4 External Enable.
Enables high-to-low transitions on TnEX to trigger captures, reloads, and control the direction of the timer/counter (up or down count). If DCENn = 1, TnEX will determine if the timer counts up or down when in Auto-reload Mode. If EXENn = 1, TnEX should be configured as a digital input.
0: Transitions on the TnEX pin are ignored.
1: Transitions on the TnEX pin cause capture, reload, or control the direction of timer count (up or down) as follows:
Capture Mode: '1'-to-'0' Transition on TnEX pin causes RCAPnH:RCAPnL to capture timer value.
Auto-Reload Mode:
DCENn = 0: '1'-to-'0' transition causes reload of timer and sets the EXFn Flag.
DCENn = 1: TnEX logic level controls direction of timer (up or down).

Bit2: TRn: Timer 2, 3, and 4 Run Control.
This bit enables/disables the respective Timer.
0: Timer disabled.
1: Timer enabled and running/counting.

Bit1: C/Tn: Counter/Timer Select.
0: Timer Function: Timer incremented by clock defined by TnM1:TnM0 (TMRnCF.4:TMRnCF.3).
1: Counter Function: Timer incremented by high-to-low transitions on external input pin.

Bit0: CP/RLn: Capture/Reload Select.
This bit selects whether the Timer functions in capture or auto-reload mode.
0: Timer is in Auto-Reload Mode.
1: Timer is in Capture Mode.

Note: Timer 3 and Timer 2 share the T2 and T2EX pins.

SFR Definition 23.9. TMRnCF: Timer 2, 3, and 4 Configuration

			R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	-	TnM1	TnM0	TOGn	TnOE	DCENn	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: TMR2CF:0xC9;TMR3CF:0xC9;TMR4CF:0xC9
SFR Page TMR2CF: page 0;TMR3CF: page 1;TMR4CF: Page 2

Bit7–5: Reserved.

Bit4–3: TnM1 and TnM0: Timer Clock Mode Select Bits.
Bits used to select the Timer clock source. The sources can be the System Clock (SYSCLK), SYSCLK divided by 2 or 12, or the external clock divided by 8. Clock source is selected as follows:
00: SYSCLK/12
01: SYSCLK
10: EXTERNAL CLOCK/8 (Synchronized to the System Clock)
11: SYSCLK/2

Bit2: TOGn: Toggle output state bit.
When timer is used to toggle a port pin, this bit can be used to read the state of the output, or can be written to in order to force the state of the output (Timer 2 and Timer 4 Only).

Bit1: TnOE: Timer output enable bit.
This bit enables the timer to output a 50% duty cycle output to the timer's assigned external port pin.
NOTE: A timer is configured for Square Wave Output as follows:
CP/RLn = 0
C/Tn = 0
TnOE = 1
Load RCAPnH:RCAPnL (See "Square Wave Frequency (Timer 2 and Timer 4 Only)" on page 320.)
Configure Port Pin to output squarewave (See **Section "18. Port Input/Output" on page 235***)*
0: Output of toggle mode not available at Timers's assigned port pin.
1: Output of toggle mode available at Timers's assigned port pin.

Bit0: DCENn: Decrement Enable Bit.
This bit enables the timer to count up or down as determined by the state of TnEX.
0: Timer will count up, regardless of the state of TnEX.
1: Timer will count up or down depending on the state of TnEX as follows:
if TnEX = 0, the timer counts DOWN.
if TnEX = 1, the timer counts UP.

Note: Timer 3 and Timer 2 share the T2 and T2EX pins.

SFR Definition 23.10. RCAPnL: Timer 2, 3, and 4 Capture Register Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: RCAP2L: 0xCA; RCAP3L: 0xCA; RCAP4L: 0xCA
SFR Page: RCAP2L: page 0; RCAP3L: page 1; RCAP4L: page 2

Bits 7–0: RCAP2, 3, and 4L: Timer 2, 3, and 4 Capture Register Low Byte.
The RCAP2, 3, and 4L register captures the low byte of Timer 2, 3, and 4 when Timer 2, 3, and 4 is configured in capture mode. When Timer 2, 3, and 4 is configured in auto-reload mode, it holds the low byte of the reload value.

SFR Definition 23.11. RCAPnH: Timer 2, 3, and 4 Capture Register High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: RCAP2H: 0xCB; RCAP3H: 0xCB; RCAP4H: 0xCB
SFR Page: RCAP2H: page 0; RCAP3H: page 1; RCAP4H: page 2

Bits 7–0: RCAP2, 3, and 4H: Timer 2, 3, and 4 Capture Register High Byte.
The RCAP2, 3, and 4H register captures the high byte of Timer 2, 3, and 4 when Timer 2, 3, and 4 is configured in capture mode. When Timer 2, 3, and 4 is configured in auto-reload mode, it holds the high byte of the reload value.

SFR Definition 23.12. TMRnL: Timer 2, 3, and 4 Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: TMR2L: 0xCC; TMR3L: 0xCC; TMR4L: 0xCC
SFR Page: TMR2L: page 0; TMR3L: page 1; TMR4L: page 2

Bits 7–0: TL2, 3, and 4: Timer 2, 3, and 4 Low Byte.
The TL2, 3, and 4 register contains the low byte of the 16-bit Timer 2, 3, and 4

SFR Definition 23.13. TMRnH Timer 2, 3, and 4 High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: TMR2H: 0xCD; TMR3H: 0xCD; TMR4H: 0xCD
SFR Page: TMR2H: page 0; TMR3H: page 1; TMR4H: page 2

Bits 7–0: TH2, 3, and 4: Timer 2, 3, and 4 High Byte.
The TH2, 3, and 4 register contains the high byte of the 16-bit Timer 2, 3, and 4

Anexo C
Extracto de la hoja de información del microcontrolador 16F877

PIC16F87XA
Data Sheet

28/40/44-Pin Enhanced Flash Microcontrollers

Note the **following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

DNV Certification, Inc. USA
DNV MSC The Netherlands Accredited by the RvA
ANSI·RAB QMS ACCREDITED
DNV
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory, Up to 368 x 8 bytes of Data Memory (RAM), Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

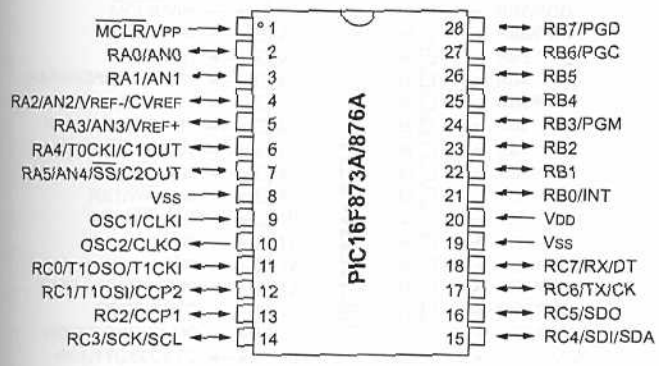
CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

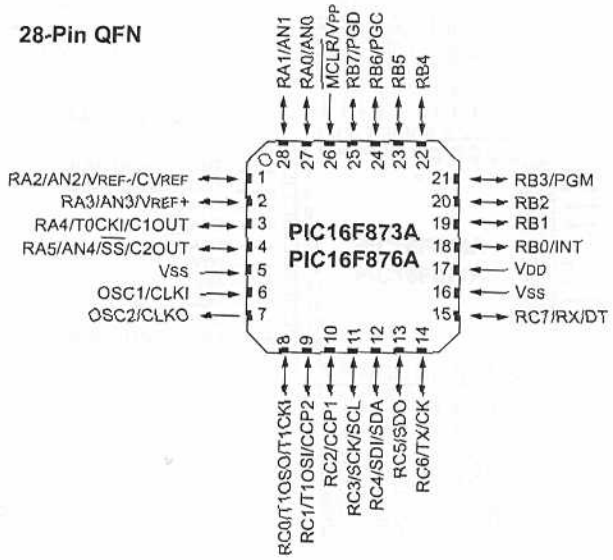
Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Pin Diagrams

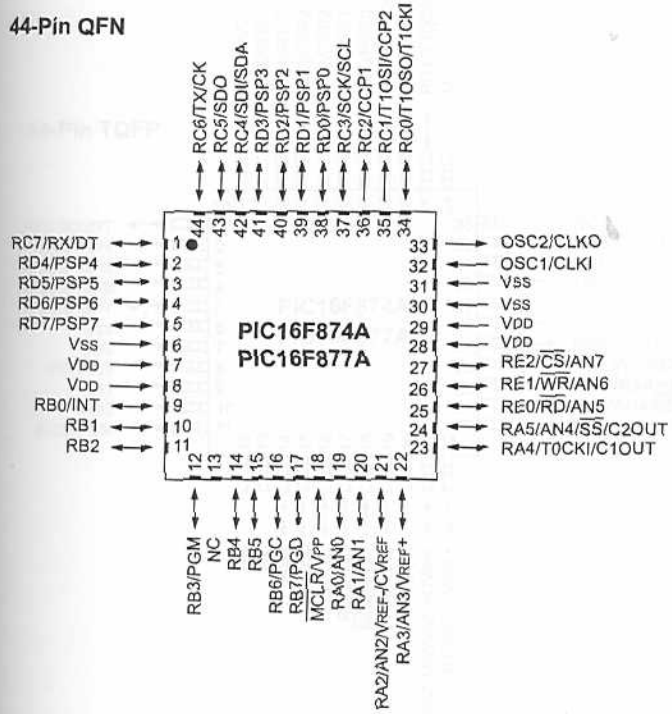
28-Pin PDIP, SOIC, SSOP



28-Pin QFN

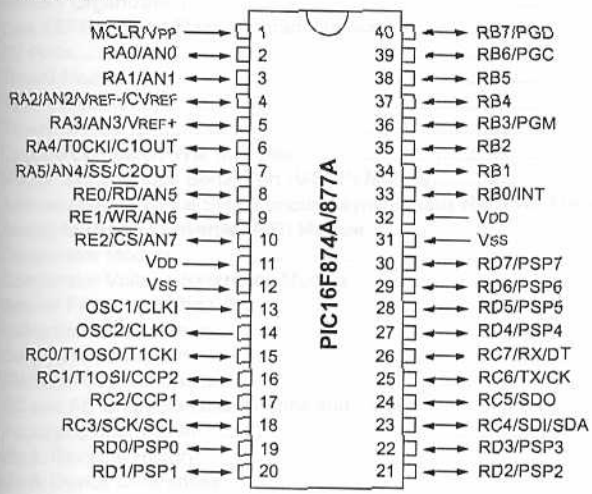


44-Pin QFN

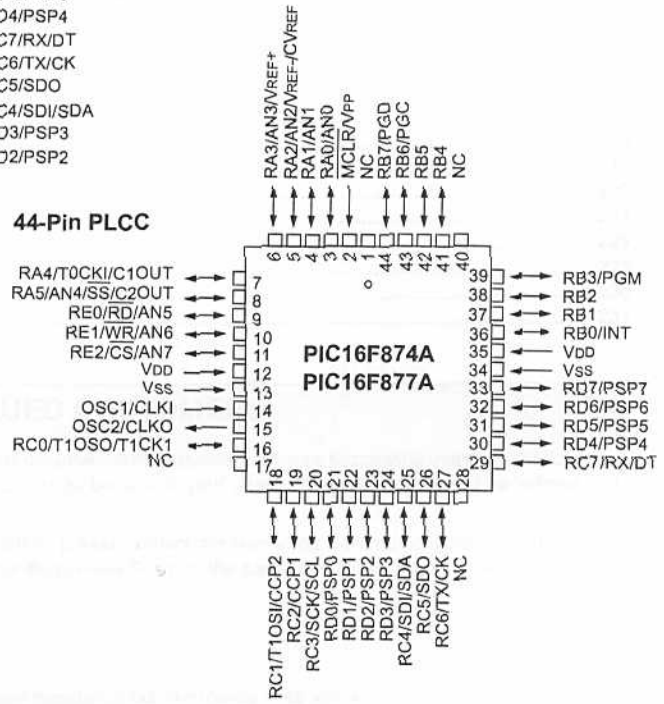


Pin Diagrams (Continued)

40-Pin PDIP



44-Pin PLCC



44-Pin TQFP

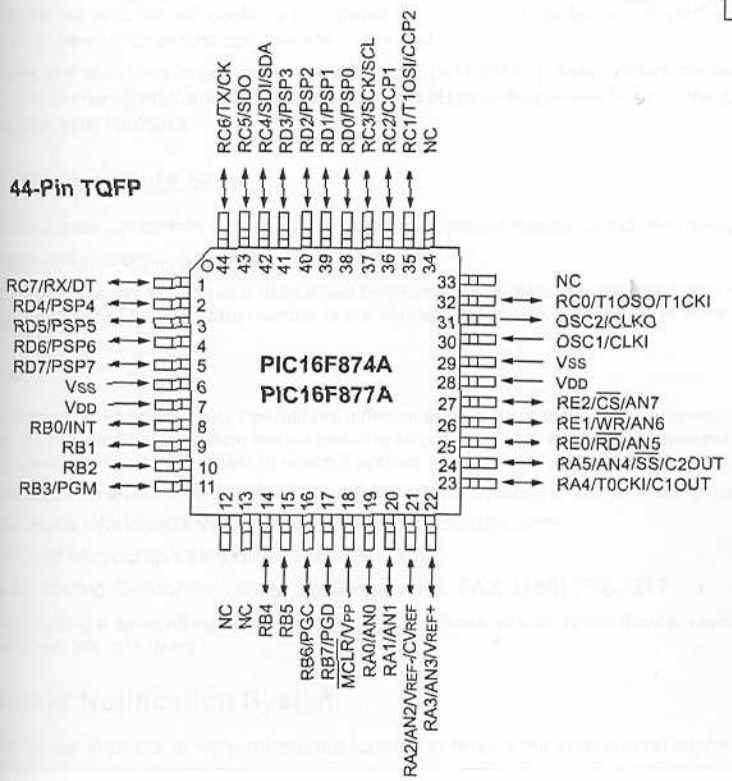


Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our Web site at **www.microchip.com/cn** to receive the most current information on all of our products.

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h	General Purpose Register 16 Bytes	110h	General Purpose Register 16 Bytes	190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
	7Fh		FFh		17Fh		1FFh

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.
2: These registers are reserved; maintain these registers clear.

FIGURE 2-4: PIC16F873A/874A REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h				
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
CCPR2L	1Bh		9Bh				
CCPR2H	1Ch	CMCON	9Ch				
CCP2CON	1Dh	CVRCON	9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh				
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 96 Bytes		accesses 20h-7Fh	16Fh	accesses A0h - FFh	1EFh
					170h		1F0h
	7Fh		FFh		17Fh		1FFh

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873A.
2: These registers are reserved; maintain these registers clear.

2.2.2.2 OPTION_REG Register

The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the external INT interrupt, TMR0 and the weak pull-ups on PORTB.

> **Note:** To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

> **Note:** When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

2.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB port change and external RB0/INT pin interrupts.

> **Note:** Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.4 PIE1 Register

The PIE1 register contains the individual enable bits for the peripheral interrupts.

> **Note:** Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.5 PIR1 Register

The PIR1 register contains the individual flag bits for the peripheral interrupts.

> **Note:** Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred
Note 1: PSPIF is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag bit
1 = The SSP interrupt condition has occurred and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI – A transmission/reception has taken place.
- I^2C Slave – A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated Start condition was completed by the SSP module.
  - The initiated Stop condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A Start condition occurred while the SSP module was Idle (multi-master system).
  - A Stop condition occurred while the SSP module was Idle (multi-master system).

0 = No SSP interrupt condition has occurred

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

5.2 Using Timer0 with an External Clock

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T0CKI to be high for at least 2 Tosc (and a small RC delay of 20 ns) and low for at least 2 Tosc (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

5.3 Prescaler

There is only one prescaler available which is mutually exclusively shared between the Timer0 module and the Watchdog Timer. A prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer and vice versa. This prescaler is not readable or writable (see Figure 5-1).

The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF 1`, `MOVWF 1`, `BSF 1,x`....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment.

REGISTER 5-1: OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note:	To avoid an unintended device Reset, the instruction sequence shown in the PICmicro® Mid-Range MCU Family Reference Manual (DS33023) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

14.11.1 INT INTERRUPT

External interrupt on the RB0/INT pin is edge triggered, either rising if bit INTEDG (OPTION_REG<6>) is set or falling if the INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, flag bit, INTF (INTCON<1>), is set. This interrupt can be disabled by clearing enable bit, INTE (INTCON<4>). Flag bit INTF must be cleared in software in the Interrupt Service Routine before re-enabling this interrupt. The INT interrupt can wake-up the processor from Sleep if bit INTE was set prior to going into Sleep. The status of global interrupt enable bit, GIE, decides whether or not the processor branches to the interrupt vector following wake-up. See **Section 14.14 "Power-down Mode (Sleep)"** for details on Sleep mode.

14.11.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in the TMR0 register will set flag bit, TMR0IF (INTCON<2>). The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). See **Section 5.0 "Timer0 Module"**.

14.11.3 PORTB INTCON CHANGE

An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<4>). See **Section 4.2 "PORTB and the TRISB Register"**.

14.12 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt (i.e., W register and Status register). This will have to be implemented in software.

For the PIC16F873A/874A devices, the register W_TEMP must be defined in both Banks 0 and 1 and must be defined at the same offset from the bank base address (i.e., If W_TEMP is defined at 0x20 in Bank 0, it must also be defined at 0xA0 in Bank 1). The registers, PCLATH_TEMP and STATUS_TEMP, are only defined in Bank 0.

Since the upper 16 bytes of each bank are common in the PIC16F876A/877A devices, temporary holding registers, W_TEMP, STATUS_TEMP and PCLATH_TEMP, should be placed in here. These 16 locations don't require banking and therefore, make it easier for context save and restore. The same code shown in Example 14-1 can be used.

EXAMPLE 14-1: SAVING STATUS, W AND PCLATH REGISTERS IN RAM

```
MOVWF    W_TEMP          ;Copy W to TEMP register
SWAPF    STATUS,W        ;Swap status to be saved into W
CLRF     STATUS          ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
MOVWF    STATUS_TEMP     ;Save status to bank zero STATUS_TEMP register
MOVF     PCLATH, W       ;Only required if using pages 1, 2 and/or 3
MOVWF    PCLATH_TEMP     ;Save PCLATH into W
CLRF     PCLATH          ;Page zero, regardless of current page
:
:(ISR)                   ;(Insert user code here)
:
MOVF     PCLATH_TEMP, W  ;Restore PCLATH
MOVWF    PCLATH          ;Move W into PCLATH
SWAPF    STATUS_TEMP,W   ;Swap STATUS_TEMP register into W
                         ;(sets bank to original state)
MOVWF    STATUS          ;Move W into STATUS register
SWAPF    W_TEMP,F        ;Swap W_TEMP
SWAPF    W_TEMP,W        ;Swap W_TEMP into W
```

Anexo D
EMII-800 User Manual Application Information

AnyDATA

AnyTime AnyPlace Any Wireless

Data Solutions

EMII-800 User Manual

Application Information

EMII-800-V2.0

December 5, 2002

AnyDATA.NET Inc.
Hanvit Bank B/D 6F
Byulyang-dong Kwachon
KOREA
Tel) 82-2-504-3360
Fax) 82-2-504-3362

AnyDATA.NET Inc.
Hanvit Bank B/D 6F
Byulyang-dong Kwachon
KOREA

Send Technical Questions to:
paiton@AnyDATA.NET

EMII-800 Reference Manual Application Information
EMII-800-V1.0
February 15, 2002

Introduction

This Manual provides hardware interface and programming information for EMII-800 CDMA Wireless Data Modem.

Users can connect the EMII-800 to their PC or Notebook and easily test the wireless communications.

Disclaimer and Limitation of Liability

AnyDATA.NET Inc. assumes no responsibility for any damage or loss resulting from the misuse of its products. AnyDATA.NET Inc. assumes no responsibility for any loss or claims by third parties, which may arise through the use of its products.

AnyDATA.NET Inc. assumes no responsibility for any damage or loss caused by the deletion or loss of data as a result of malfunctions or repairs.

FCC RF Exposure Information

Warning! Read this information before using this device.

In August 1996 the Federal Communications Commission (FCC) of the United States with its action in Report and Order FCC 96-326 adopted an updated safety standard for human exposure to radio frequency electromagnetic energy emitted by FCC regulated transmitters. Those guidelines are consistent with the safety standard previously set by both U.S. and international standards bodies. The design of this device complies with the FCC guidelines and these international standards.

CAUTION

- **Operating Requirements**

- The user cannot make any changes or modifications not expressly approved by the party responsible for compliance; otherwise it could void the user's authority to operate the equipment.
- To satisfy FCC RF exposure compliance requirements for a mobile transmitting device, this device and its antenna should generally maintain a separation distance of 20cm or more from a person's body.

Special accessories

In order to ensure that this device is in compliance with FCC regulation, special accessories are provided with this device and must be used with this device only. The user should not use accessories other than the special accessories given with this device

Contents

5 Electrical Specifications

5.1 Absolute Maximum Ratings

5.2 Recommended Operating Conditions

5.3 Power Consumption

5.4 Serial Interface Electrical Specifications

6 Mechanical Dimension

6.1 EMII-800 Outline

7 FCC Notice

1. Introduction

1.1 Purpose

This Manual provides hardware interface and programming information for the EMII-800 CDMA Wireless Data Module.

1.2 Organization

This manual discusses the interface and operation of the module and is divided into the following subsections:

- Section 2 – Introduces users to the EMII-800 CDMA Wireless Data Module's basic features and general specifications.
- Section 3 – Contains EMII-800 Pin descriptions - DC12V Input Port, 8-pin Serial Port, and Debugging Port.
- Section 4 – Describes the UART Interface.
- Section 5 – Specifies the recommended operating conditions, DC voltage characteristics, I/O timing, and power estimations for the module.
- Section 6 – Provides package dimensions and outlook features for the module.
- Section 7 – Describes the FCC Notice.

1.3 Revision History

The revision history for this document is shown in Table 1-1.

Table 1-1 Revision History

Version	Date	Description
V1.0	Feb. 2002	Initial Release
V2.0	Dec. 2002	Corrected document content

1.4 References

1. QUALCOMM Incorporated. MSM5100 Mobile Station Modem™: Component Supply Specification. 80-V2180-7-X1, July 13, 2001.
2. QUALCOMM Incorporated. MSM5100™ Mobile Station Modem: Device Specification (Preliminary Information). 93-V2180-1-X3, August 30, 2001.
3. QUALCOMM Incorporated. SURF5100 User Manual. 80-V2535-1-X1, March 28, 2001.

1.5 Acronym List

Term	Definition
CDMA	Code-Division Multiple Access
CODEC	Coder-Decoder
GPIO	General-purpose Input/Output
JTAG	Joint Test Action Group (ANSI/ICEEE Std. 1149.1-1990)
LCD	Liquid Crystal Display
LDO	Voltage Regulator
LED	Light Emitting Diode
PCB	Printed Circuit Board
PCM	Pulse Coded Modulation
PCS	Personal Communications Service
RF	Radio Frequency
Rx	Receive
TCXO	Temperature-Controlled Crystal Oscillator
Tx	Transmit
UART	Universal Asynchronous Receiver Transmitter

2. Overview

2.1 Application Descriptions

The CDMA Wireless Data Module is a complex consumer communications instrument that relies heavily on both digital signal and embedded processor technologies. The Wireless Data Modules manufactured by AnyDATA.NET support Code-Division-Multiple-Access (CDMA). This operates in the PCS spectrum.

In a continuing effort to simplify the design and to reduce the production cost of the Wireless Data Module, AnyDATA.NET has successfully developed the EMII-800. The EMII-800 is AnyDATA.NET's latest compact Wireless Data Module operating in the PCS spectrum. The EMII-800 contains a complete digital modulation and demodulation system for CDMA standards as specified in IS-95 A/B and IS-2000.

The subsystem within the EMII-800 includes a CDMA processor (MSM5100), an integrated CODEC with an ear piece and microphone amplifiers, and an RS-232 serial interface supporting forward link data communications at a rate of 153.6kbps.

The EMII-800 provides an external interface, that includes the standard RS-232 and Digital Audio.

The EMII-800 has the capability to power down unused circuits in order to dynamically minimize power consumption.

2.2 Technical Specifications

2.2.1 General Specifications

PARAMETERS	DESCRIPTIONS
External Access	Code-Division-Multiple-Access (CDMA)
CDMA Protocol	IS-95A/B/C, IS-98A, IS-126, IS-637A, IS-707A
Data Rate	153.6Kbps
Transmit/Receive Frequency Interval	45MHz
Band Width	1.23MHz
Operating Voltage	DC 6V ~ 12V
Current Consumption	Stand by mode: Idle (55mA), Busy mode: 280mA (Max) at 12V
Operating Temperature	-30°C ~ +60°C
Frequency Stability	±300Hz
Antenna	Whip Antenna, 50ohm
Size	57 X 121 X 24mm with case
Weight	About 110g
External Interface	RS-232, Digital/Analog Audio, LCD, Keypad, Ringer

2.2.2 Receive Specifications

PARAMETERS	DESCRIPTIONS
Frequency Range	869.04 ~ 893.97 MHz
Sensitivity	Below –104 dBm
Interference Rejection	Single tone (-30dBm @900KHz): Below –101dBm Two tone (-43 dBm @900KHz and 1700KHz): Below –101dBm Two tone (-32 dBm @900KHz and 1700KHz): Below –90dBm Two tone (-21 dBm @900KHz and 1700KHz): Below –79dBm
Spurious Wave Suppression	Below –80dBc
Input Dynamic Range	-25 dBm ~ -104dBm

2.2.3 Transmit Specifications

PARAMETERS	DESCRIPTIONS
Frequency Range	824.04 ~ 848.97 MHz
Nominal Power	0.32 W
Minimum Controlled Output Power	Below –50dBm
Max Power Spurious	900KHz: Below –42dBc/30KHz 1.98MHz: Below –54dBc/30KHz

2.2.4 Standards

IS-95A/B/C: Protocol Between MS & BTS
IS-96A: Voice Signal Coding
IS-98A: Base MS Function
IS-126: Voice Loop-Back
IS-637: Short Message Service
IS-707: Data Service

2.3 Interface Diagram

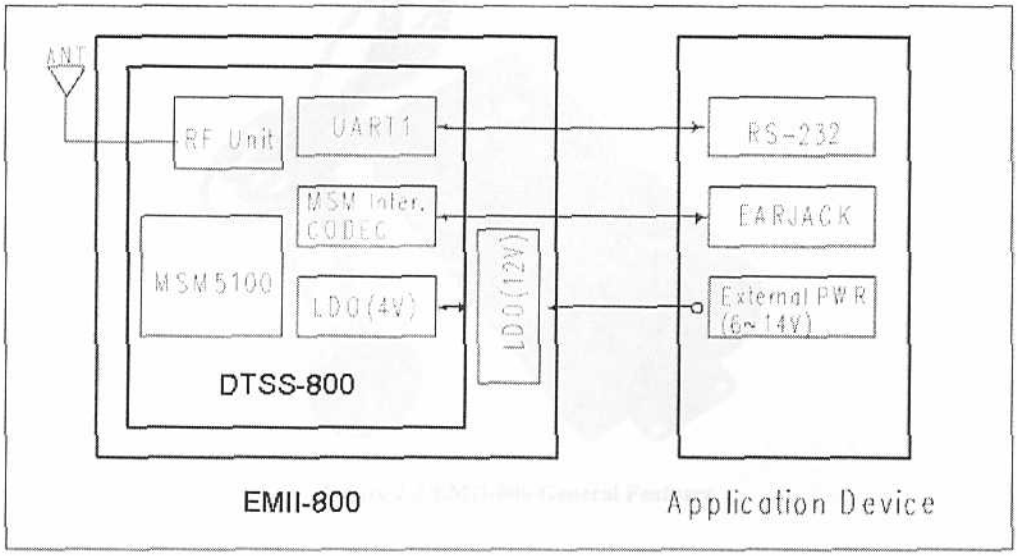


Figure 2-1 Interface Block Diagram

2.4 EMII-800 General Features

Figure 2-2 EMII-800 General Features

2.5 Internal Module Features

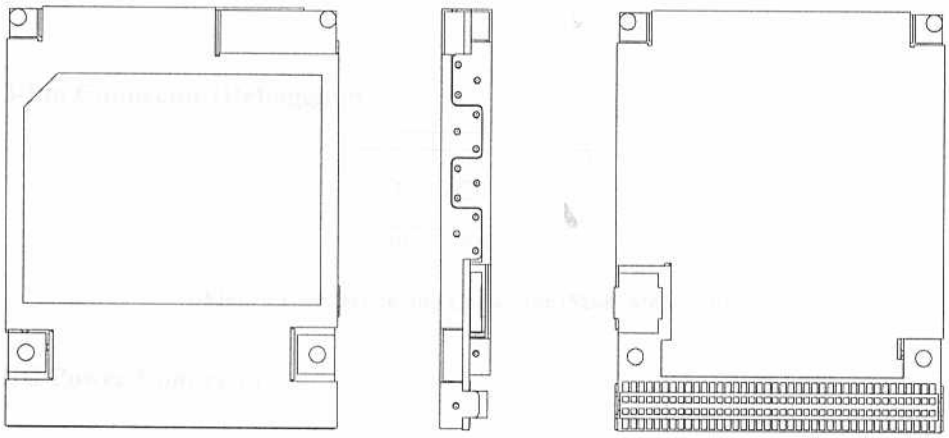
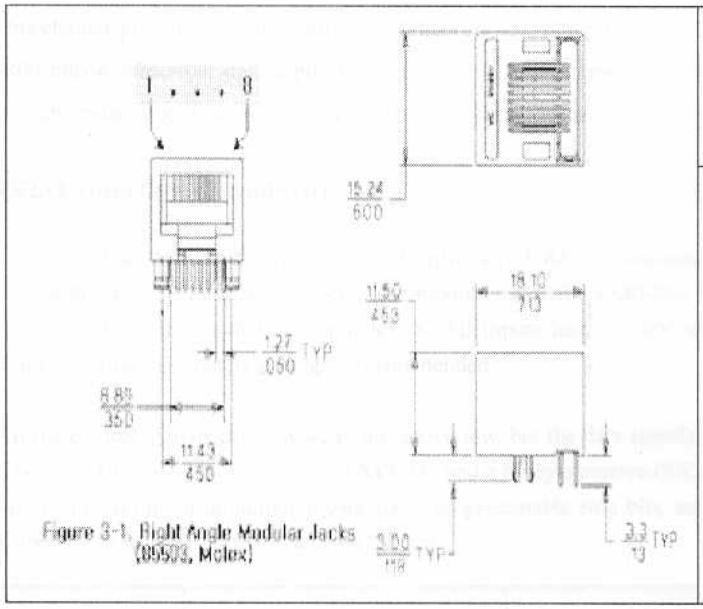


Figure 2-3 Internal Module General Features (DTSS-800)

3. PIN Description

3.1 8-Pin Male Modular Jacks (RS232 Standard)



Molex (85503)	RS-232
Pin 1	RI
Pin 2	GND
Pin 3	DTR
Pin 4	CTS
Pin 5	TX
Pin 6	RTS
Pin 7	RX
Pin 8	DCD

Reference Plane: User (not the EMII-800)

Figure 3-1 Right Angle Modular Jacks Pin Description (85503,Molex 8P)

3.2 3-Pin Connector (Debugging)

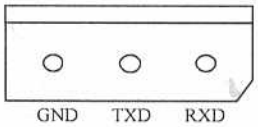


Figure 3-2 Debugging Connector (5268, Molex 3P)

3.3 DC Power Connector

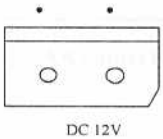


Figure 3-3 DC 12V Power Connector (5268, Molex 2P)

4. Interface Descriptions

4.1 Overview

This chapter provides essential information that the user needs to understand, in order to convert the EMII-800 into a subscriber unit application. In addition, the internal signals that are necessary for the complete understanding of the UART interfaces are described below.

4.2 RS232 Interface (Standard)

The Universal Asynchronous Receiver Transmitter (UART) communicates with serial data that conforms to the RS-232 Interface protocol. The module inside the EMII-800 provides 3.0V CMOS level outputs and 3.0V CMOS switching input levels. All inputs have a 5.0V tolerance, but 3.0V or 3.3V CMOS logic compatible signals are highly recommended.

All the control signals of the RS-232 are active low, but the data signals, RXD and TXD, are active high. The UART has a 64 byte transmit (TX) FIFO and a 64 byte receive (RX) FIFO. The UART features hardware handshaking, programmable data sizes, programmable stop bits, and odd, even, no parity. The UART operates at a maximum bit rate of 115.2kbps.

NAME	DESCRIPTION	CHARACTERISTIC
DP_DCD/	Data Carrier Detect	Network connected from the module
DP_RI/	Ring Indicator	Output to host indicating coming call
DP_RTS/	Request to Send	Ready for receive from host
DP_TXD	Transmit Data	Output data from the module
DP_DTR/	Data Terminal Ready	Host ready signal
DP_RXD	Receive Data	Input data to the module
DP_CTS/	Clear to Send	Module output signal
GND	Signal Ground	Signal ground

Figure 4-1 UART Interface Pinouts

4.3 LEDs

The EMII-800 has four LEDs that indicate the status and functionality of the module inside. The PWR LED turns on when adequate power is supplied to the module, and the module is able to turn on. The SMS LED will turn on, if there is a SMS message or a voicemail message. After the user has read the SMS message or listened to the voicemail message, the SMS LED will turn off. When the module is in the traffic or conversation stage, the BUSY LED will be on. Shortly after the module has been turned on, the IDLE LED should turn on indicating that the module is in-service. This means that the module is within the range of the base station and is able to receive a signal from the base station.

5. Electrical Specifications

5.1 Absolute Maximum Ratings

Operating the module under conditions that exceed those listed in the Absolute Maximum Ratings table may result in damage to the module.

Absolute Maximum Ratings should be considered as limiting value. The module may not function properly and should not be operated if any one of the parameters is not within its specified operating range.

Table 5-1 Absolute Maximum Ratings

PARAMETER	MIN	MAX	UNITS
Storage Temperature	-40	+80	°C
Voltage On Any Input Pin	-	+20.0	V
Voltage On Any Output Pin	-	+10.0	V
Supply Voltage	-	+15	V
Initializing Current	100		mA
Drop	No damages after 60-Inch drop over concrete floor		

5.2 Recommended Operating Conditions

PARAMETER	MIN	MAX	UNITS
Supply Voltage	+6	+12	V
Operating Temperature	-30	+60	°C
Operating Humidity	95%(50°C) Relative Humidity		

5.3 Power Consumption

CONVERSATION	STANDBY	
	Idle	Sleep
280 mA (MAX)	55 mA	18 mA

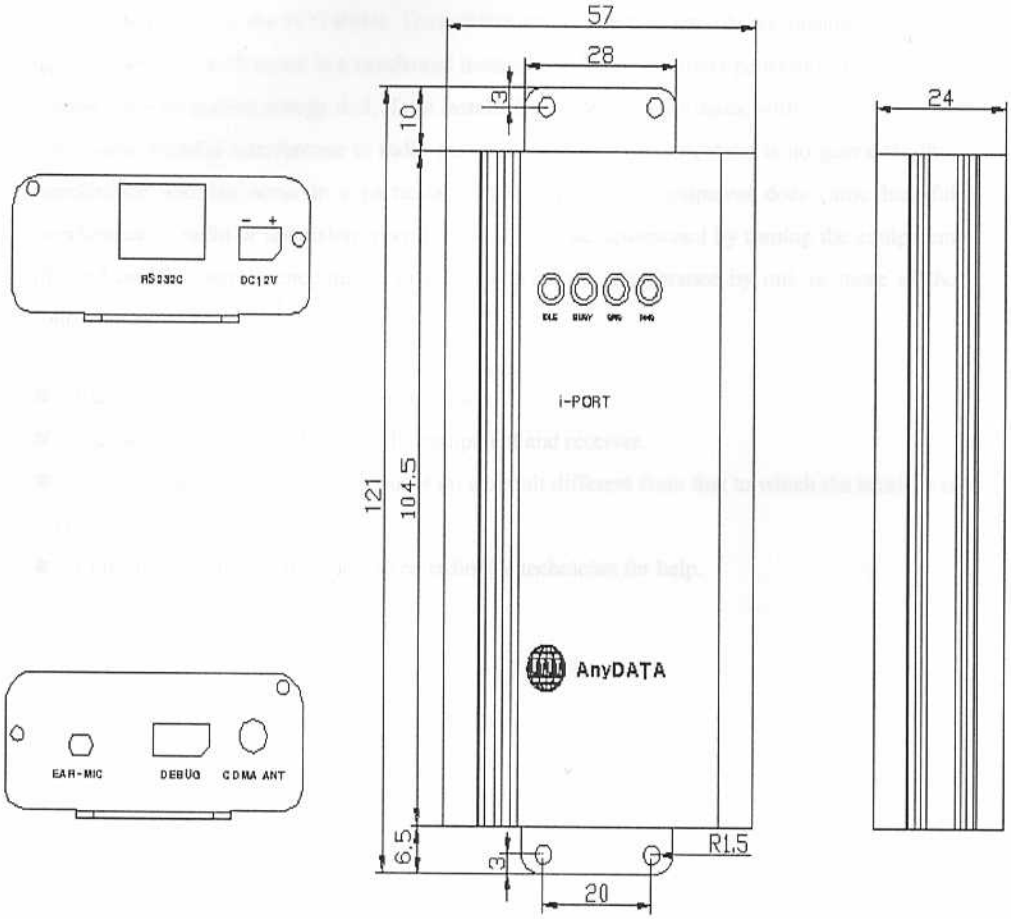
Note: Power consumption measured with a supply voltage of 12V

5.4 Serial Interface Electrical Specifications

PARAMETER	MIN	TYP	MAX	UNITS
Input High Voltage	+1.7		+15.0	V
Input Low Voltage	-15.0		+1.2	V
Output High Voltage	+5.0	+6.0	+7.0	V
Output Low Voltage	-7.0	-6.0	-5.0	V

6. Mechanical Dimensions

6.1 EMII-800 Outline



Units: mm

7. FCC Notice

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- *Connect the equipment into an outlet on a circuit different from that to which the receiver is* connected.
- Consult the dealer or an experienced radio/TV technician for help.