

Protocolos de Transferencia de Archivos

Xmodem Transferencia de Archivos PC – PIC

Autor: Carlos A. Narváez V. Ingeniero Electricista Universidad de Oriente Email: cnarvaez@bolivar.udo.edu.ve

Propósito

Xmodem es un protocolo usado para la transferencia de archivos entre PC. Este protocolo esta implementado en el programa Hyperterminal de Windows. En aplicaciones con microcontroladores que implementen un servidor web, se requiere normalmente, debido a lo limitado de la memoria de estos dispositivos, usar memorias externas para alojar las paginas web. Estas paginas web deben ser buscadas por un nombre y una extensión, por lo que es necesario la creación de un Sistema de Archivos para ROM, que implica un directorio y cierto ordenamiento, preferiblemente en un bloque al comienzo de la ROM, para de esta manera minimizar los ciclos de lectura secuencial necesarios para encontrar el archivo buscado. El trabajo normal es la creación de una imagen del Sistema de Archivos para ROM con un programa en el PC y luego transferir esta imagen a la memoria ROM utilizando Hyperterminal en el lado del PC y una implementación de Xmodem en el PIC el cual, finalmente, se encarga de transferir dicha imagen a la memoria ROM externa.

En el presente trabajo realizaremos un programa en Visual Basic a fin de estudiar el protocolo Xmodem y luego lo implementaremos en un PIC 16F877 para transferir una imagen del Sistema de Archivos para ROM a una memoria EEPROM 24LC256 de 256 Kbit (32Kbyte), programable vía la interfaz serial I2C.

Teoría de Operación

Xmodem es básicamente un método para la transferencia de archivos entre computadoras personales. Desde el punto de vista del hardware, Xmodem es implementado sobre líneas seriales asincrónicas, con configuración de 8 bits de datos, sin bit de paridad y un bit de parada. Típicamente los datos en un archivo son transmitidos sin cambios, excepto algunos casos como en el sistema operativo MS-DOS donde hay que agregar un ^Z (26 decimal) como caracter de fin de archivo.

Xmodem utiliza los siguientes caracteres ASCII especiales:

Nombre	Decimal	Hexadecimal	Descripción
SOH	01	H01	Comienzo Encabezado
EOT	04	H04	Fin de la transmisión
ACK	06	H06	Confirmación (positiva)
DEL	16	H10	Escape
X-On (DC1)	17	H11	Transmisión On
X-Off (DC3)	19	H13	Transmisión Off
NAK	21	H15	Confirmación(negativa)
SYN	22	H16	Synchronous idle
CAN	24	H18	Cancelar

Protocolos de Transferencia de Archivos

Para todos los efectos, se considera como transmisor la computadora/software que envía paquetes y recibe confirmaciones y como receptora la computadora/software que recibe paquetes y envía confirmaciones.

Xmodem es un protocolo que no incluye nombre de archivo en sus paquetes por lo que previamente, ambas computadoras, la transmisora y la receptora, deben conocer donde encontrar los datos (que archivo será transmitido) y donde poner los datos (archivo que almacenara los datos o buffer).

Una vez entrado en el protocolo, el computador transmisor, espera entre 10 segundos y un minuto, recibir un carácter NAK de la computadora receptora. Se dice entonces, que la computadora receptora controla el manejo del protocolo. La computadora transmisora puede repetir esto cualquier numero de veces. Si un carácter diferente a NAK o CAN es recibido por el transmisor, este lo ignora. El carácter CAN implica la cancelación de la transferencia de archivo y el transmisor debe salir del protocolo Xmodem. Una vez que el receptor ha enviado un carácter NAK, espera por 10 segundos la llegada del primer paquete Xmodem. Si no recibe nada en estos 10 segundos, envía otro NAK, repitiendo esto por 10 veces, luego de lo cual de no haber recibido respuestas, sale del protocolo Xmodem, indicando un error severo.

Transmisor		Receptor
[espera por un minuto]	<	[NAK]
[Comienzo transmisión de bloques	>	

El transmisor toma los datos y los divide en piezas de 128, 8 bits bytes y los coloca en un paquete Xmodem. Un paquete Xmodem consta de:

[SOH] [seq] [cpl seq] [128 bytes de datos] [csum]

SOH	Caracter de comienzo de cabecera (1 decimal).
seq	Un byte de número de secuencia el cual comienza en uno, incrementándose en uno hasta que alcance el valor de 255,
cpl seq	Un byte complemento a uno de seq. Este puede ser calculado como $cpl = 255 - (255 \text{ And } seq)$ o usando Xor como $cmp = (255 \text{ and } seq) \text{ xor } 255$.
datos	128, 8 bit byte de datos. Si un archivo MS-DOS es enviado, un carácter ^Z (26 decimal) debe agregarse al final del archivo. Si el último bloque de datos es menor que 128 byte, Xmodem rellena el paquete con caracteres ^Z
csum	Un byte que representa la suma de todos los datos, sin incluir los 3 primeros byte del encabezado. Cualquier overflow o carry es descartado.

Protocolos de Transferencia de Archivos

Una vez que la inicialización del protocolo ha sido completada, el transmisor envía el primer paquete Xmodem y entonces espera. Luego que el receptor tiene el paquete completo, este compara su propio checksum calculado con el checksum que fue enviado en el paquete. Si los checksum coinciden, el receptor envía una confirmación positiva (ACK). Si los checksum son diferentes, el receptor envía una confirmación negativa (NAK).

Después que el receptor envía un ACK, el transmisor envía el siguiente paquete Xmodem. Si el transmisor recibe un NAK, este reenvía nuevamente el mismo paquete Xmodem.

Una vez que el transmisor ha enviado el último paquete y ha recibido la confirmación positiva (ACK), respectiva, este envía un EOT al receptor y entonces espera por un ACK final antes de salir de Xmodem. Cuando el receptor ve un carácter EOT en lugar de un SOH, el receptor transmite un ACK y cierra sus archivos saliendo del protocolo Xmodem.

Todo lo anterior esta representado en la siguiente secuencia:

Transmisor		Receptor
	<<<<<<<<	[NAK]
[SOH] [001] [254] [.....] [csum]	>>>>>>>	
	<<<<<<<<	[ACK]
[SOH] [002] [253] [.....] [csum]	>>>>>>>	
	<<<<<<<<	[ACK]
[SOH] [003] [252] [.....] [csum]	>>>>>>>	
	<<<<<<<<	[ACK]
[SOH] [004] [251] [.....] [csum]	>>>>>>>	
	<<<<<<<<	[ACK]
[EOT]	>>>>>>>	
	<<<<<<<<	[ACK]

El receptor puede cancelar la transferencia enviando un carácter CAN y saliendo del protocolo. Si el transmisor recibe un carácter CAN cuando espera un carácter ACK o NAK, o cuando espera un carácter SOH (comienzo de paquete), este termina la transferencia y sale del protocolo. Muchas implementaciones del protocolo Xmodem, requieren de dos caracteres CAN ante de reconocer una condición de cancelación.

Protocolos de Transferencia de Archivos

Recuperación de Errores y Temporización Xmodem

La detección y recuperación de errores es el propósito principal de protocolo Xmodem. Algunas de las condiciones de errores comunes se muestran a continuación:

- **Error de complemento**

Si el número de secuencia, luego de ser complementado no coincide con el del campo complemento de secuencia, el paquete debe ser descartado y un carácter NAK enviado al transmisor.

- **Condición de paquete duplicado**

Si el número de secuencia es el mismo, que el numero de secuencia del ultimo paquete recibido, el paquete debe ser descartado y un ACK enviado al transmisor.

- **Error fuera de secuencia**

Si el número de secuencia, luego de ser complementado coincide con el numero en el campo complemento de secuencia, pero este no es el numero de secuencia esperado, el receptor debe enviar dos caracteres CAN al transmisor y salir del protocolo

- **Error tiempo de espera vencido en el Receptor**

Cuando el receptor espera datos y transcurren 10 seg, sin recibir un carácter, el receptor debe enviar otro NACK al transmisor. Esto se debe repetir unas 10 veces si la condición de error se mantiene. El tratamiento del error tiempo de espera vencido varia según la implementación del protocolo Xmodem.

- **Error tiempo de espera vencido en el transmisor**

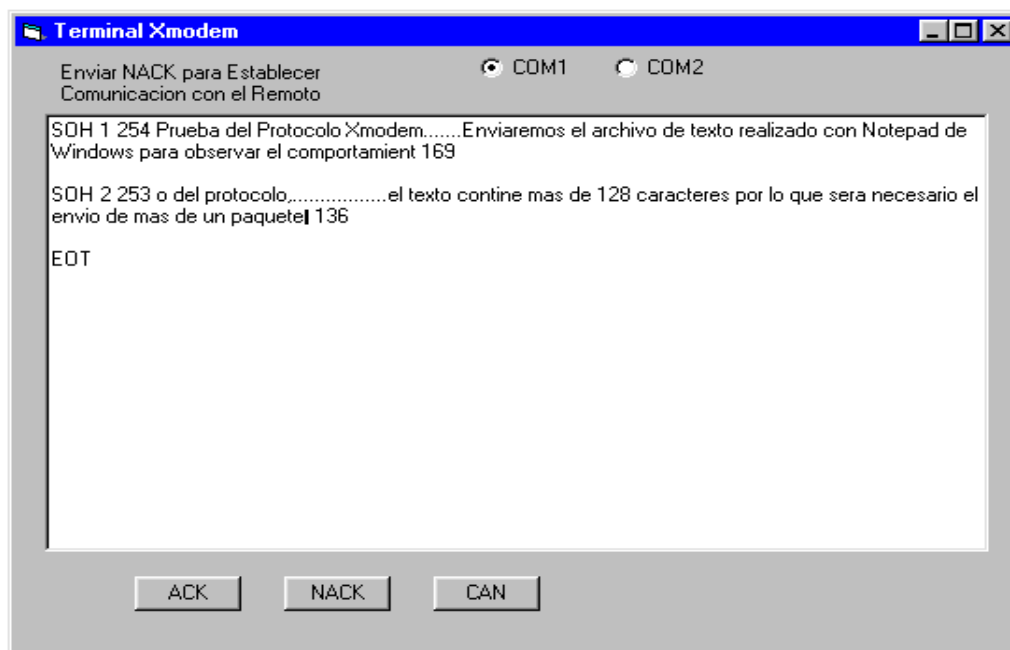
En la versión original del protocolo, el transmisor espera hasta 10 seg. por la llegada de un carácter ACK, NAK o CAN y luego retransmite el ultimo paquete como si hubiera recibido un NAK. En la mayoría de las implementaciones, el transmisor espera un tiempo mayor (30Seg. a 1Minuto) y luego termina la transferencia de archivos si no se ha recibido un ACK, NAK o CAN, o espera por 30 seg. y luego retransmite el ultimo paquete.

Además de todas las condiciones de errores anteriores, también debe considerarse las posibilidades de recibir un falso EOT o CAN debido a alteraciones en la comunicación, en el primer caso puede considerarse el envío de dos EOT en lugar de uno para asegurarse que realmente se llegó al final de la transmisión.

Protocolos de Transferencia de Archivos

Programa Visual Basic Terminal Xmodem

Hemos confeccionado el siguiente programa en Visual Basic para el estudio del protocolo Xmodem. La prueba del mismo fue realizada conectando dos PC a través de sus puertos seriales, una corriendo Hyperterminal (transmisor) y la otra corriendo nuestra aplicación (receptor). Se utilizó la opción transferencia de archivos de Hyperterminal escogiendo el protocolo Xmodem en una conexión configurada a 9600 bits/seg. 8 bit bytes, sin bit de paridad y un bit de parada. Se transfirieron pequeños archivos realizados en Notepad de Windows, comprobando su correcto funcionamiento.



Como podemos observar el primer paquete es enviado luego de presionar el botón NACK, observe que para la transferencia total del contenido del archivo, se requieren dos paquetes, cada paquete comienza con el carácter SOH, seguido del número de secuencia y luego el complemento a uno del número de secuencia, el número final en cada paquete es el checksum. Luego de recibir el primer paquete, presionamos el botón ACK, con lo que recibimos el segundo paquete. Luego presionamos nuevamente el botón de ACK y recibimos el carácter de fin de transmisión (EOT). Un ACK hace que el transmisor salga del protocolo Xmodem. La presión del botón NACK cuando se espera un ACK, produce el reenvío del último paquete y la presión dos veces del botón CAN, cuando se espera un ACK o un NACK produce la salida del transmisor del protocolo Xmodem, suministrando el mensaje "Cancelado por el usuario". El programa es simple y no contempla temporización ni chequeo de errores.

Protocolos de Transferencia de Archivos

Transferencia de Archivos PC – PIC

Como mencionamos en el propósito de este trabajo, la idea es grabar en una memoria 24LC256, una imagen de un Sistema de Archivos para ROM, creada en un PC. Nuestro trabajo se centra en escribir una pequeña versión de Xmodem para el PIC 16F877 al cual se conecta la memoria EEPROM a través del bus I2C. Un análisis de este bus, escapa del alcance del presente trabajo, basta indicar que el compilador C PCM de CCS, incorpora rutinas para escribir y leer en este bus.

Creando un Sistema de Archivos para ROM

Una Sitio Web normalmente implica varias paginas, cada una de ellas en un archivo. La pagina principal normalmente lleva por nombre Index.htm o default.htm. Si el sitio web debe ser alojado en memoria ROM, implica la creación de un directorio de archivos y un cierto ordenamiento, preferiblemente al comienzo de la ROM, a fin de minimizar el número de ciclos de lectura. Los elementos mínimos requeridos en el directorio son los siguientes:

- La longitud del archivo en bytes
- Un apuntador del comienzo del archivo en ROM
- Un checksum del archivo
- Nombre del Archivo (formato 8.3)

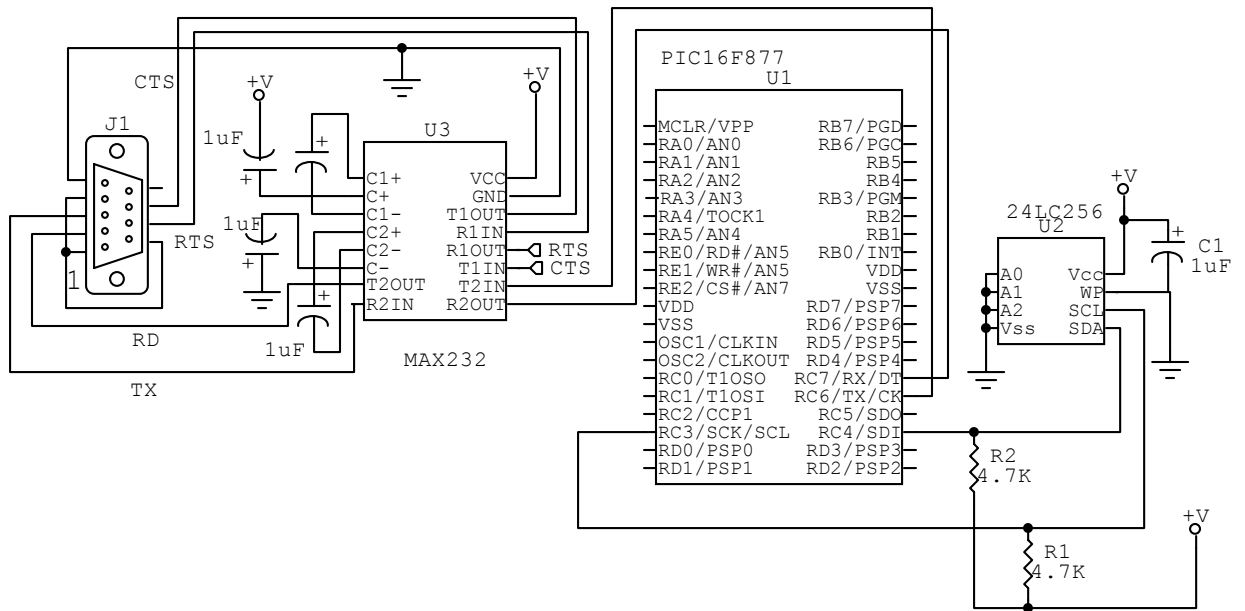
Es lógico pensar que el archivo Index.htm o default.htm, debe estar ubicado como primer archivo en el directorio, pues como pagina principal se debe facilitar encontrarla. Así mismo para simplificar el trabajo de un servidor Web basado en PIC, los encabezados HTTP deben ser incluido al comienzo del contenido de cada archivo en ROM. Esto es precisamente lo que hace programas como el WEBROM de Iosoft LTD. o el ROMMAKE, crean en el PC un archivo con extensión .ROM que no es más que una imagen de un Sistema de Archivos como el descrito anteriormente y realizado a partir de un subdirectorio del PC donde se encuentran los archivos de las páginas Web que conforman el sitio.

Los elementos del directorio descriptos, son suficiente para páginas Web estáticas, sin embargo, en Sistemas Empotrados, lo normal es que las paginas Web sean dinámica, pues estos sistemas deben suministrar información en tiempo real. En Servidores Web potentes el dinamismo puede lograrse usando la interfaz CGI. Debido a la limitación de los PIC esta interfaz resulta muy pesada para su procesamiento. Jeremy Benthon propone una interfaz sencilla denominada EGI y en MICRO-IO implementan algo similar denominado preprocesador de comandos. Estos tópicos escapan del alcance del presente trabajo, por lo que nos limitaremos exclusivamente a la transferencia de un archivo .ROM, a la memoria conectada al PIC, utilizando el protocolo Xmodem.

Otra manera de transferir la imagen del sistema de archivo ROM del PC a la memoria ROM externa es convirtiendo dicha imagen en un archivo en formato HEX, tal como lo realiza el programa bin2hex de MICRO-IO y luego transferir este usando hyperterminal e implementando en el PIC un programa que lea el formato HEX. También pudiera programarse la memoria externa usando la propia interfaz Ethernet.

Protocolos de Transferencia de Archivos

Hardware



Protocolos de Transferencia de Archivos

Software

```

/*****
/* Programa Xmodem */
/* Graba en una memoria 24LC256 externa un archivo enviado */
/* desde una PC utilizando Hyperterminal */
/* Compilado: CCS PCM C compiler */
/* Autor: Carlos A. Narvaez */
/* Fecha: 31/05/2003 */
*****/
#include <16F877.H> // PIC16F877

#FUSES HS,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP
#ID CHECKSUM
#ZERO_RAM
#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#use fast_io(D)
#use fast_io(E)

#define EEPROM_ADDR 0xa0 // Dirección I2C de la EEPROM
#define ROMPAGE_LEN 32 // 32 byte (1024 pages)

#use DELAY(CLOCK=19660800)

#use I2C(MASTER, SDA=PIN_C4, SCL=PIN_C3, RESTART_WDT, FAST)
#use RS232 (BAUD=9600, XMIT=PIN_C6, RCV=PIN_C7, ERRORS)

/* Caracteres especiales ASCII Xmodem */

#define SOH 0x01
#define STX 0x02
#define ETX 0x03
#define EOT 0x04
#define ENQ 0x05
#define EOF 0x1A
#define ACK 0x06
#define NAK 0x15
#define ETB 0x17
#define CAN 0x18
#define NUL 0x00

/* Maquina de Estado Xmodem */

#define GET_SOH 0
#define GET_NUMBLK 1
#define GET_NOTNUMBLK 2
#define GET_DATA 3
#define CHECK_ERROR 4
#define TRY_START 5
#define IDLE 6
#define ABORTED 7
#define TIMEOUT 8

#define OK 0
#define BAD_BLKNUM 1
#define BAD_CHECKSUM 2
```

Protocolos de Transferencia de Archivos

```
#define XBLOCK_LEN 128
#define TXBUFFLEN 64

BYTE txbuff[TXBUFFLEN];

#byte PORTA = 5
#byte PORTC = 7
#bit LED = PORTA.4

/* Variables Globales */
BYTE c, r, len=0, idx, blq, i, offset, sum=0, state, notblknum, blknum;
BYTE retries;
unsigned long ElapsedTime;

/* Prototipos */
void xmodem_recv(void);
BYTE xmodem_start(void);
BYTE CheckPacket(void);
void delay_seg(int);

/*****
/* Programa Principal */
*****/
void main() {

set_tris_a(0x03);
delay_seg(5);
printf("Xmodem\n\r");
Led = 0;
r = xmodem_start();

switch(r)
{

case IDLE:
printf("Transferencia OK\n\r");
break;

case TIMEOUT:

printf("Transferencia Timeout\n\r");
break;

case ABORTED:

printf("Transferencia Abortada\n\r");
break;

}
Led = 1;
}
```

Protocolos de Transferencia de Archivos

```

/*****
*/ xmodem_start()
/*****
BYTE xmodem_start(void)
{
    retrys = 6;
    state = TRY_START;

    while(retrys > 0)
    {
        ElapsedTime = 0;
        xmodem_rcv();
        while(!kbhit() && ++ElapsedTime <8000)
            delay_ms(1);
        if(kbhit())
        {
            state = GET_SOH;
            retrys = 10;
            c=getchar();
            xmodem_rcv();

            while ( !( (state==IDLE) || (state==ABORTED) || (state==TIMEOUT) ) )
            {
                if(kbhit())
                {
                    c=getchar();
                    xmodem_rcv();
                }

                if( retrys == 0 )
                {
                    state = TIMEOUT;
                }
            }
            return state;
        }
        retrys--;
    }
    state = TIMEOUT;
    return state;
}
/*****
*/ Maquina de estado Xmodem
/*****
void xmodem_rcv(void)
{
    if(state == GET_DATA) // Datos
    {
        idx = len & (ROMPAGE_LEN - 1); // idx = 0, 1,2,...31
        len++;
        txbuff[idx] = c;
        sum += c;
    }
}

```

Protocolos de Transferencia de Archivos

```
        if (idx == ROMPAGE_LEN-1)                // ..Escribir pagina ROM
        {
            i2c_start();                          // condicion de start
            i2c_write(EEPROM_ADDR);               // byte control + bit write(0)
            i2c_write(blq >> 1);                 // Direccion high byte
            offset = len - ROMPAGE_LEN;
            if (blq & 1)
                offset += 0x80;
            i2c_write(offset);                    // direccion low byte
            for(i=0; i<ROMPAGE_LEN; I++)
                I2C_WRITE(txbuff[I]);            // escritura byte a byte
            i2c_stop();                           // condicion de parada
        }
        if(len == XBLOCK_LEN)
        {
            state = CHECK_ERROR;
            return;
        }
        else
            return;
    }
    if (state == GET_SOH)                          // Verifica si 1er caracter
    {
        if (c == SOH)
        {
            // ..if SOH, move on
            state = GET_NUMBLK;
            return;
        }
        else if(c == CAN) state = ABORTED;

        else if (c == EOT)                         // ..Si EOT, Fin
        {
            putchar(ACK);
            state = IDLE;
            return;
        }
        // respuesta desconocida
        retries--;
        putchar(NAK);
        return;
    }

    if (state == GET_NUMBLK)                       // verifica si 2do caracter
    {
        blknum = c;
        blq =c;                                    // ..numero de bloque
        state = GET_NOTNUMBLK;
        return;
    }

    if (state == GET_NOTNUMBlk)                   // verifica si 3er caracter
    {
        state = GET_DATA;
        notblknum = c;                             // Complemento del numero de bloque
        blq--;
        return;
    }
}
```

Protocolos de Transferencia de Archivos

```
if ( state == CHECK_ERROR)                // Verifica errores
{
    switch( CheckPacket() )
    {
        case OK:                          // si no errores, enviar otro paquete
            state = GET_SOH;
            len = 0;
            sum = 0;
            retrys = 10;
            putchar(ACK);
            break;

        case BAD_BLKNUM:                   // Si numero de bloque y su complemento
            state = GET_SOH;               // no coinciden, retransmita el paquete
            len = 0;
            sum = 0;
            retrys--;
            putchar(NAK);
            break;

        case BAD_CHECKSUM:                 // si error de checksum,
            state = GET_SOH;               // retransmita el paquete
            len = 0;
            sum = 0;
            retrys--;
            putchar(NAK);
            break;
    }
    return;
}
if ( state == TRY_START )                  // Comienzo
{
    putchar(NAK);
    return;
}
}
```

Protocolos de Transferencia de Archivos

```

/*****
/* Chequea que el numero de bloque corresponda al */
/* complemento y verifica el checksum */
*****/
BYTE CheckPacket(void)

{

// Verifica el complemento del número de bloque

    if ( blknum != ~notblknum )
        return BAD_BLKNUM;

// Verifica Checksum

    else if( c != sum )
        return BAD_CHECKSUM;

    else
        return OK;
}

/*****
/* Delay Segundos */
*****/
void delay_seg( int n )
{
    for( ; n != 0; n-- )
        delay_ms(1000);
}

```

