Using ALFAT with SPI and I2C

GHI Electronics, LLC.

Updated April 2006

Introduction

ALFAT can run in different modes. The easiest one to use is UART. We highly recommend using UART. But, in some applications, UART is not an option so using SPI or I2C is required. In systems where is no available hardware peripherals, simulating SPI through software is much easier than UART and I2C. Standard SPI is constructed using 4 signals.

Those signals are:

1. SCK - This is the clock used to synchronize data
2. MISO - Data output pin from ALFAT to your system
3. MOSI - Data input signal from your system to ALFAT
4. SSEL - Salve select. When low, ALFAT will accept the data

On every SPI transaction, the master (your system) will generate 8 clocks on SCK. Through these clocks, MOSI and MISO swap their internal buffers; therefore, data will transfer in both directions on every SPI transfer.

DATARDY Signal

Slaves can't send data on SPI. Data must be requested by the master. Since ALFAT is a SPI slave, ALFAT needs a way to signal data availability. For this we added DATARDY pin. Whenever DATARDY goes high, a SPI transaction must be issued by the master to read the data. ALFAT will hang waiting for the master to read the data. It is recommended that DATARDY connects to a rising-edge-interrupt-pin to detect this pin easily. If edge-triggered-interrupt-pin is not available, you can pool the DATARDY pin and read the data whenever is necessary. After every data read, DATARDY goes low in about 8 microseconds, and will come back high again if another read is needed. ALFAT never sends data arbitrary. It responds to the issued commands only.

BUSY Signal

Also, ALFAT may not be ready for the incoming data. This is what BUSY line is used for. Before sending any data, make sure BUSY is low. If BUSY and DATARDY are both high, ALFAT is requesting read of its internal buffer but it is not ready for any new incoming data yet. In most applications, BUSY can completely ignored. ALFAT buffers the incoming data, up to 250 bytes. BUSY goes high only if the buffer is full. Since ALFAT continuously read the buffer, the buffer may never get filled.

SCK Mode

SPI SCK has 4 modes. To synchronize the data correctly, SCK must idle high and data is latched on the rising edge. Since SCK is idles high, it is required to go low and back high (rising) to latch the first bit. For example, on PIC microcontrollers, set CKP=1 and CKE=0. The internal LPC register inside ALFAT is set to 0x18.

Older Versions

Important: Older versions of ALFAT had SCK set to idle low and to latch data on rising edge. This didn't work correctly for some customers so we switched to the new SCK settings, idle high and latch on rising edge. This is no problem in the firmware as updating the firmware will update you SPI settings. This is not the case in the boot loader. If your boot loader is older than 3.3, you will have to run SPI in the old settings and after sending 'R', you have to switch SPI mode to the new settings. In the old settings, SCK is idle low and data is latched on rising edge.
In most cases, this is not an issue because both modes latch data on rising edge.

Writing Code

You can easily miss data if you try to send data when DATARDY is high. **In this case, ALFAT will ignore the incoming data and will assume you are trying to read. Many customers had problems in text mode because they forgot that text mode echoes back what you send till you disable the echo or enter framed mode.** You **MUST** check DATARDY before trying to send data.

When you send 'EE 0<CR>' or send FM<CR>', you have to read every character back as DATARDY will go high after every byte (echo.)
Her are the steps to disable the echo 'EE 0<CR>'

1. Send 'E'
2. Read the echo
3. Send the second 'E'
4. Read the echo
5. Send ' ' (space)
6. Read the echo
7. Send <CR>
8. Read the echo and the command prompt, ie. Z:>
9. Now ALFAT will not echo your commands

This is the right way to send data, DATARDY must be checked before every byte:

```
if(DATARDY==1)
{
    SELECT_ALFAT;
    DataInFromALFAT=ReadALFAT();// we have new data
    UNSELECT_ALFAT;
}else
{
    // if we have data
    if(DataIsAvailableToSendToALFAT())
    {
       // if ALFAT is ready for data
      if(BUSY==0)// this may not be necessary, see above!
       {
            SELECT_ALFAT;
            WriteALFAT(DataOutToALFAT);
            UNSELECT_ALFAT;
        }
    }
}
```

New SPI mode and full duplex

Starting in version 1.16, the SPI interface runs in full duplex. This means that you can send and receive data simultaneously. When you send data to ALFAT and ALFAT has no data to send back, it will send NDT (No Data Token.) The NDT is the value 0xFF in hexadecimal and 255 in decimal. And when you want to read data from ALFAT but you have nothing to send then you can use NDT to transmit. In most case you wouldn't need to send 0xFF to ALFAT but what if you need to send 0xFF? In this case you use HDT (Half Data Token.) HDT is 0xFE or 254 decimal.
To send 0xFF or FE to ALFAT (to go in file for example) you first send 0xFE and then you follow it by another 0xFE if you need to send 0xFE or you follow it by any number (except 0xFF or 0xFE) and this will mean 0xFF to ALFAT. When ALFAT sends data to you it will follow the same procedure.

Here is a simple code example on how to use HDT and NDT.

```
void SendToALFAT(int8 c)
{
        if(c==0xFF)
        {
                SendSPI(0xEF);
                SendSPI(0x00);// this can be any number but 0xFE or 0xFF
        }else
        if(c==0xFE)
        {
                SendSPI(0xFE);
                SendSPI(0xFE);
        }else
                SendSPI(c);
}
```

The previous example is a very simplified one and shouldn't be used. Consult uALFAT_lib library on uALFAT page to see how this should be done.
ALFAT library code uses the old half duplex mode that will not work if you need to send 0xff or 0xfe. You will need to modify ALFAT library to the new SPI driver.

The example below also shows how to use SPI in half duplex mode only. uALFAT library have a full SPI driver.

SPI to UART Data Converter Example (half duplex)

This is a full example code on how to use a PICmicro to convert ALFAT SPI bus to UART data.

```
#define LED                     LATBbits.LATB0
#define SELECT_ALFAT            LATCbits.LATC1=0
#define UNSELECT_ALFATLATCbits.LATC1=1

int8 fifo[256];
int8 put_pointer=0;
int8 get_pointer=0;

int8 GetFIFO(void)
{
    int8 c;
    c= fifo[get_pointer];
    get_pointer++;

    return c;
}
//------------------------
void PutFIFO(int8 c)
{
    fifo[put_pointer]=c;
    put_pointer++;
}
//--------------------------
int8 ReadWriteMasterSPI(int8 data_to_send)
{
   SSPBUF = data_to_send;          // send SSPBUF
   while ( !SSPSTATbits.BF );      // wait until cycle complete
   return ( SSPBUF );              // return with byte read
}
//---------------------------
void main(void)
{
   int8 i=0;

   int8 data[10];
   InitializeProcessor();// setup the registers
   TRISC=0b11010101;

    //    OpenSPI
    /*
    // the old settings
    SSPSTAT=0x40;
    SSPCON1=0x22;
    //*/
    // the new settings
    SSPSTAT=0x00;
    SSPCON1=0x32;
    OpenUSART();
```

```
          UNSELECT_ALFAT;
        PrintROM("SPI converter is ready....\n\r");
        while(1)
        {

          // if we have UART then add to FIFO
          if(DataRdyUSART())
          {
                    PutFIFO(getc());
             }
             // if ALFAT has data to send us
          if(PORTDbits.RD2)//DATARDY
          {
                    SELECT_ALFAT;
                    putc(ReadWriteMasterSPI(0));
                      UNSELECT_ALFAT;
                    LED=!LED;
             }else
             {
                    // if we have data
                       if(put_pointer!=get_pointer)
                       {
                                 // if ALFAT is ready for data
                                 if(!PORTDbits.RD1)// BUSY
                                 {
                                                SELECT_ALFAT;
                                                ReadWriteMasterSPI(GetFIFO());
                                                UNSELECT_ALFAT;
                                          }
                                 }
                    }
          }
     }
```

I2C on ALFAT

I2C on ALFAT is very similar to SPI, DATARDY and BUSY work exactly the same. ALFAT run as a I2C slave with address 0xA4

Here is a simple example on how to read a byte:

```
#define ADDRESS (0xA4)
    if(DATARDY)
    {

        StartI2C();
        while ( ReadyI2C());
        WriteI2C(ADDRESS+1); // the  '+1' is to indicate read
        IdleI2C();
        I=(ReadI2C());
        IdleI2C();
        StopI2C();
    IdleI2C();
}
```