Extract from Application Note 16

“Updating MC75/TC6x Firmware”

The purpose of this document is to provide a preliminary description of the Generic Firmware Update. Please note that the following chapter is only an extract from the complete Application Note which will soon be ready for publication.

3 Generic Firmware Update

This chapter provides information on how to download MC75/TC6x firmware into the module by starting the download from the host application or a dedicated download utility designed by the customer. The download can be done over the serial interfaces ASC0 or ASC1 or the USB interface.

3.1 Format of the Firmware File Supplied from Siemens

For this download solution, the MC75/TC6x firmware must be available in the format *.USF. Files for each type of module can be requested from Siemens. The firmware file consists of records to be transferred one by one from the host application to the module.

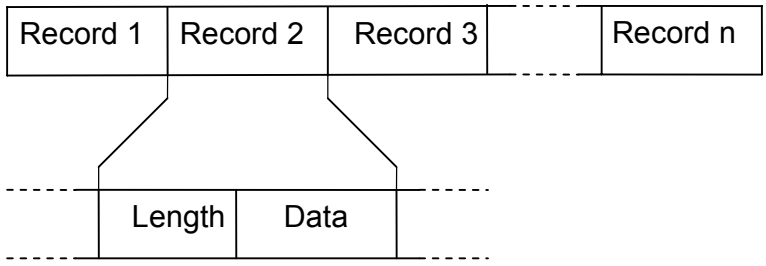


Figure 5: Format of *.USF firmware file

Length: The 1st byte of each record is the length byte which states the number of data bytes to follow.
Data: 2nd to nth byte

3.2 Requirements for the Firmware Download Utility

The host application or download utility must be designed to meet the requirements of the download process specified in the download protocol. See also Figure 6 and Figure 7 for detail.

First of all, to start the download process, the host application has to send the AT^SFDL command to the module. This causes the module to reset, enter the firmware download mode and send the start pattern to the host application to notify that it is ready to receive records. Upon receipt of the start pattern ("`ANSWER_OK`" in the example below) the host application must start to send the first record. Optionally, we recommend that the buffer of the serial interface be purged before the module starts to send the first record.

After each record the host application has to wait for an acknowledgment delivered by the module. If this is "OK", the next record can be transferred. "BUSY" means that the module is still processing the record last received – in this case, the transfer must be halted until the next "OK" acknowledgment is received. A checksum verifies the correct transmission of each

record. If a download error is detected, the module will send a “RETRY” message.
The host application must be capable to evaluate the answer patterns received from the module and to correctly transmit the data.

Once all records are successfully received, the module switches off automatically. To restart the module use the ignition line IGT.

Appropriate examples are provided in section 3.2.1 for ASC0 or ASC1 and in section 3.3.1 for USB.

Table 3: Answer patterns required for generic download utility

Answer pattern	Hex value	Function
`#define ANSWER_OK`	`0x01`	After the reset caused by AT^SFDL, “`ANSWER_OK`” notifies the host application that the first record can be sent. After each successfully received record, “`ANSWER_OK`” notifies the host application that the next data record can be transferred.
`#define ANSWER_RETRY`	`0x02`	Notifies that a failure was detected after verifying the checksum. The host application is requested to send the last record again and wait until this record is successfully received and acknowledged by “`ANSWER_OK`”.
`#define ANSWER_FATAL`	`0x03`	Download failed at all. Possible causes: The firmware you are trying to download is not intended for this module type, or you are trying to install an older version. The currently installed software is not affected.
`#define ANSWER_BUSY`	`0x04`	The module is still processing the record received. The transfer from the host application must be halted until the module sends the next “`ANSWER_OK`”. For example, “`ANSWER_BUSY`” may appear when the flash is erased on the module. This procedure usually takes a couple of seconds. In this case, “`ANSWER_BUSY`” is no error.

Known limitations:
The blocks sent to the module should not really differ in the size of the blocks. Do not use single byte and multiple byte transfer in combination.

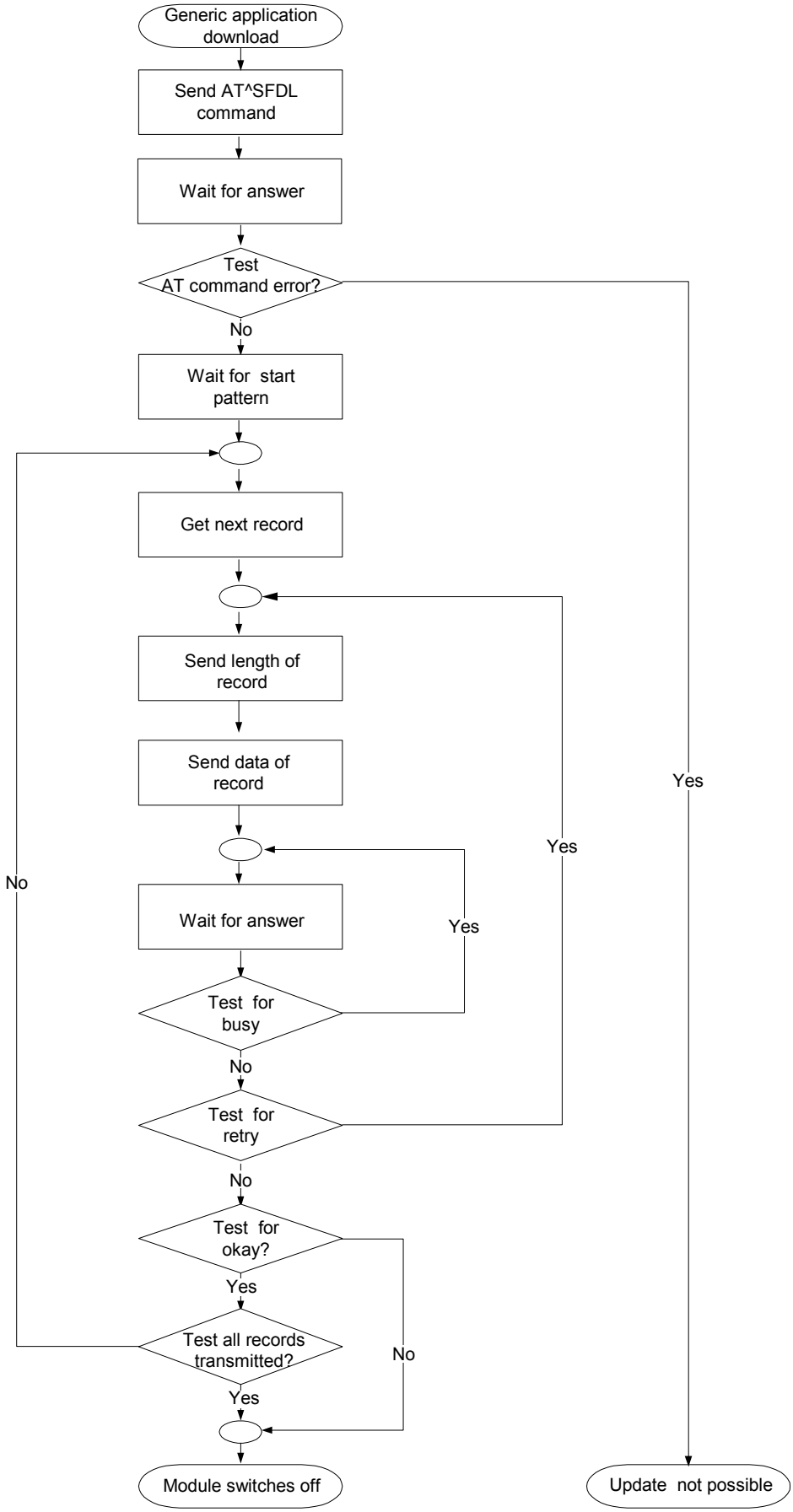


Figure 6: Download protocol for ASC0 / ASC1

3.2.1 Example of Firmware Download Program for ASC0/ASC1

This section presents an example of a firmware download utility based on the widely used programming language “C”.

```
void DoUpdate (void)
{
    BYTE bAnswer ;
    BYTE bRecordLen ;
    BYTE bRecordData[MAX_RECORD_SIZE] ;

    /* Send AT command */
    WriteToModule ((BYTE *) "AT^SFDL\r", 8) ;

    /* Wait for answer */
    if (!ReadResultLineFromModule ())
        /* Error */
        return ;

    /* Wait for start pattern */
    do
    {
        ReadFromModule (&bAnswer, 1) ;
    }
    while (ANSWER_OK != bAnswer) ;

    do
    {   /* Read record from file */
        bRecordLen = MAX_RECORD_SIZE ;
        FileReadRecord (&bRecordLen, bRecordData) ;

        do
        {   /* Send length of record */
            WriteToModule (&bRecordLen, 1) ;

            /* Send data of record */
            WriteToModule (bRecordData, bRecordLen) ;

            do
            {   /* Wait for answer */
                ReadFromModule (&bAnswer, 1) ;
            }
            while (ANSWER_BUSY == bAnswer) ;
        }
        while (ANSWER_RETRY == bAnswer) ;
    }
    while (ANSWER_OK == bAnswer && !FileEnd ()) ;
}
```

3.3 Special Considerations for USB

The firmware download via USB can take place when the module is connected via the USB interface to the USB host. There is no need for a firmware inside a module because the Siemens bootloader includes a fully functional USB stack with a CDC (Communication Device Class) ACM (Abstract Control Model) device implementation.

The module simulates a modem connected via a standard serial COM port. Through this virtual COM port the module can be forced to enter the firmware download mode.

At first the module must be started by an appropriate mechanism, such as asserting the IGT (Ignition) signal or plugging in the USB cable.

Code fragments shown below are in pseudo C code based on the operating system with enabled USB functionality and should be easily adaptable to other operating systems.

What is the major difference between a generic update through ASC0/ASC1 and a generic update through USB?

- The USB scenario requires that the virtual COM port be closed immediately after the AT^SFDL command was sent and OK received. Then, after 2 seconds the COM port shall be opened again.
- The length byte and the data bytes of each record must be transferred in one USB transaction, i.e. by calling the API only once per record.

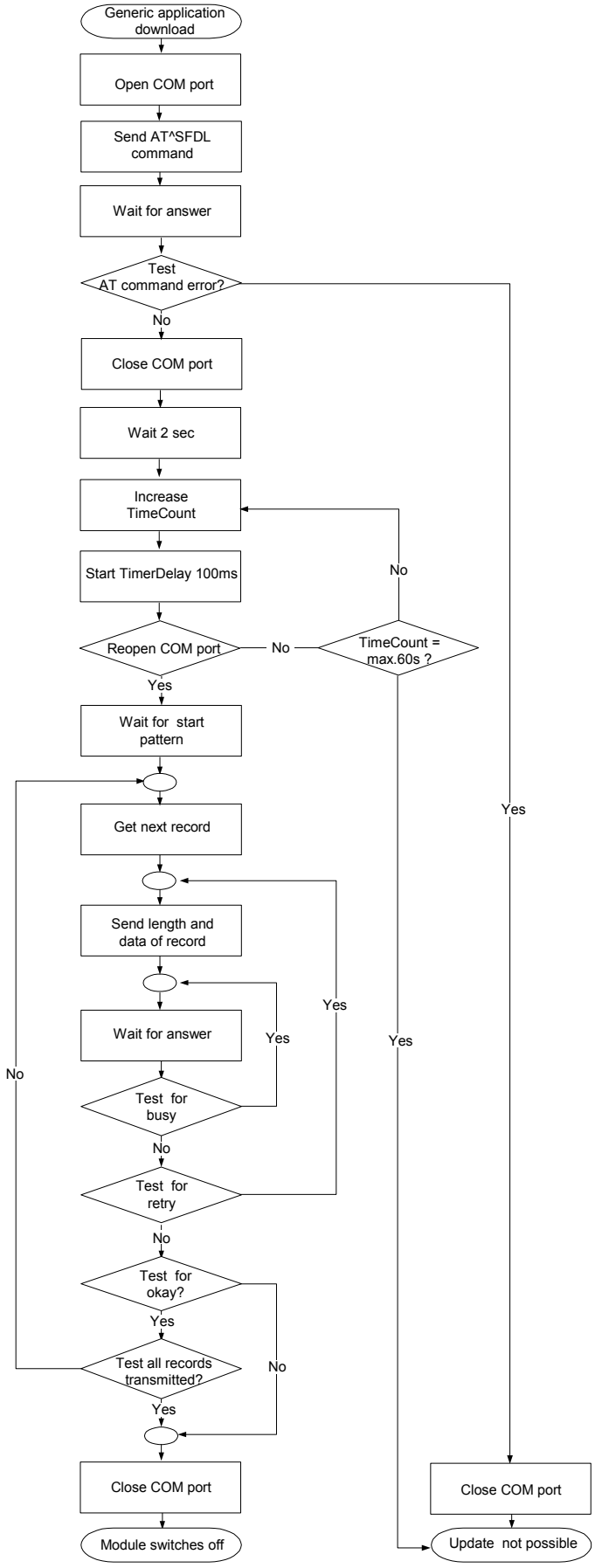


Figure 7: Download protocol for USB

3.3.1 Example of Firmware Download Program for USB

```
void DoUpdate (void)
{
    BYTE bAnswer ;
    BYTE bRecord[MAX_RECORD_SIZE] ;
    Int iTry ;          /* Number of tries of open virtual COM port*/

    /* Send AT command */
    WriteToModule ((BYTE *) "AT^SFDL\r", 8) ;

    /* Wait for answer */
    if (!ReadResultLineFromModule ())
        /* Error */
        return ;

    /* Close port because the virtual COM port of the module shall
       be disabled for a short time */
    CloseComPort() ;

    /* Allow USB host to disable the virtual COM port. */
    Wait (2 SECONDS) ;

    /* Try to reopen the virtual COM port again */
    iTry = 0 ;
    do
    {
        if (OpenComPort ())
            /* Virtual COM port is opened again */
            iTry = -1 ;
        else
        {   /* Virtual COM port is not opened, wait 100 ms */
            Wait (100 MILLISECONDS) ;
            iTry++ ;
        }
    }
    while (0 <= iTry && 80 > iTry) ;
    if (-1 != iTry)
        /* Error, virtual COM port is not opened again */
        return ;

    /* Wait for start pattern */
    do
    {
        ReadFromModule (&bAnswer, 1) ;
    }
    while (ANSWER_OK != bAnswer) ;

    do
    {   /* Read record from file */
        bRecord[0] = MAX_RECORD_SIZE ;
        FileReadRecord (bRecord, bRecord + 1) ;
```

```
        do
        {   /* Send length and data of record */
            WriteToModule (bRecord, bRecord[0] + 1) ;

            do
            {   /* Wait for answer */
                ReadFromModule (&bAnswer, 1) ;
            }
            while (ANSWER_BUSY == bAnswer) ;
        }
        while (ANSWER_RETRY == bAnswer) ;
    }
    while (!FileEnd () && ANSWER_OK == bAnswer) ;
}
```

3.4 Error Handling for Generic Firmware Update

1. If a valid firmware is installed, but the AT^SFDL command is issued without sending any firmware afterwards the module will stay in the download mode. In this case it must be switched off.
2. Disconnecting the power supply during a firmware download should be strictly avoided. If nevertheless the download is aborted due to power failure, the module will be left in an undefined state with no valid software installed. On the ASC0 / ASC1 interfaces, the module's bit rate will be set to 57600 bps in this case.
   After reconnecting power and restarting the module, you have only 1 minute to enter the AT^SFDL command and retry the download from one of the interfaces ASC0, ASC1 or USB. Also note that no other AT commands are accepted. If the download fails to start within this time the module switches off again.