Future Technology Devices International Ltd.

Application Note AN_130

FT2232H Used In An FT245 Style Synchronous FIFO Mode

Document Reference No.: FT_000186

Version 1.2

Issue Date: 2010-03-05

This application note describes how to use the FT2232H device in FT245 Style Synchronous FIFO mode

Table of Contents

1   Introduction

This application note illustrates how to set the FT2232H into an FT245 Style Synchronous FIFO mode. If the requirement is to transfer data at 60MHz, and the data rate must greater than 8MB per second, then the FT245 Style Synchronous FIFO mode is the best solution.

The FT2232H only channel A can be configured as a FT245 style synchronous FIFO interface. When configured in this mode, channel B is not available as all resources have been switched onto channel A.

Note : This mode is only available with port A of FT2232H device.

2 Hardware Configuration

EEPROM Configuration:

The FT2232H device requires an external 93C46 EEPROM organized in 16-bit words. It also supports the 93C56 and 93C66 EEPROMs organized in 16-bit words.

Device connection setting:

Use a USB cable to connect an FT2232H device to a PC - the pin assignment in **Table 2-1** give details of the required connection between the system and FT2232H.

2.1 Pin Assignment under Synchronous FIFO Interface

Only channel A of FT2232H device can be configured as a FT245 style synchronous FIFO interface. When it is configured in this mode, the pins used and the descriptions of the signals are shown as **Table 2-1**:

Channel A Pin No.	Name	Type	RS245 Configuration Description
24,23,22,21, 19,18,17,16	ADBUS[7:0]	I/O	D7 to D0 bidirectional FIFO data. This bus is normally input unless OE# is low.
26	RXF#	OUTPUT	When high, do not read data from the FIFO. When low, there is data available in the FIFO which can be read by driving RD# low. When in synchronous mode, data is transferred on every clock that RXF# and RD# are both low. Note that the OE# pin must be driven low at least 1 clock period before asserting RD# low.
27	TXE#	OUTPUT	When high, do not write data into the FIFO. When low, data can be written into the FIFO by driving WR# low. When in synchronous mode, data is transferred on every clock that TXE# and WR# are both low.
28	RD#	INPUT	Enables the current FIFO data byte to be driven onto D0...D7 when RD# goes low. The next FIFO data byte (if available) is fetched from the receive FIFO buffer each CLKOUT cycle until RD# goes high.
29	WR#	INPUT	Enables the data byte on the D0...D7 pins to be written into the transmit FIFO buffer when WR# is low. The next FIFO data byte is written to the transmit FIFO buffer each CLKOUT cycle until WR# goes high.
32	CLKOUT	OUTPUT	60 MHz Clock driven from the chip. All signals should be synchronized to this clock.
33	OE#	INPUT	Output enable when low to drive data onto D0-7. This should be driven low at least 1 clock period before driving RD# low to allow for data buffer turn-around.
30	SIWU	INPUT	The Send Immediate / WakeUp signal combines two functions on a single pin. If USB is in suspend mode (PWREN# = 1) and remote wakeup is enabled in the EEPROM , strobing this pin low will cause the device to request a resume on the USB Bus. Normally, this can be used to wake up the Host PC. During normal operation (PWREN# = 0), if this pin is strobed low any data in the device TX buffer will be sent out over USB on the next Bulk-IN request from the drivers regardless of the pending packet size. This can be used to optimize USB transfer speed for some applications. Tie this pin to VCCIO if not used.

Table 2-1 Channel A FT245 Style Synchronous FIFO Configured Pin Descriptions
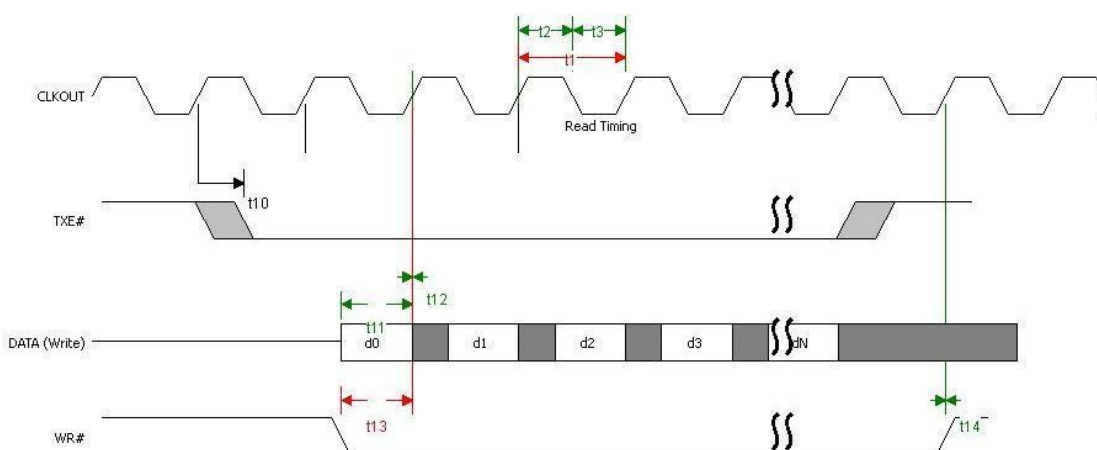
2.2 IO Timing

It's necessary to follow the IO timing as shown in Figure 2-1 and Figure 2-2 to access the data. Note that only a read or a write cycle can be performed at any one time. Data is read or written on the rising edge of the CLKOUT clock.

Table 2-2 shows the detail information for the timing.
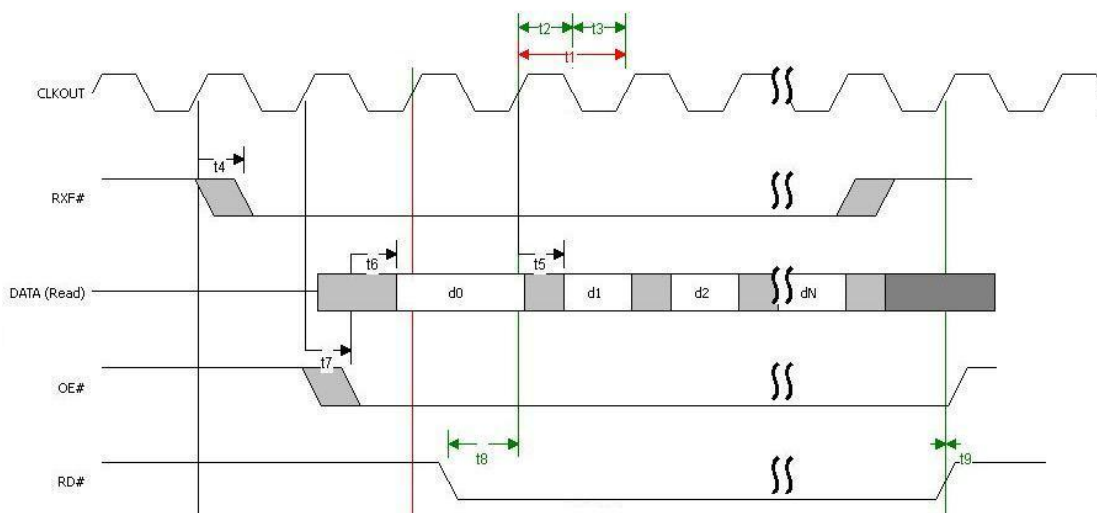
Read Timing (External system read data from FTDI chip)

Figure 2-1 Read Timing

Write Timing (External system write data into FTDI chip)

Figure 2-2 Write Timing

Name	MIN	NOM	MAX	Units	COMMENT
t1		16.67		ns	CLKOUT period
t2	7.5	8.33		ns	CLKOUT high period
t3	7.5	8.33		ns	CLKOUT low period
t4	1		7.15	ns	CLKOUT to RXF#
t5	1		7.15	ns	CLKOUT to read DATA valid
t6	1		7.15	ns	OE# to read DATA valid
t7	1		7.15	ns	CLKOUT to OE#
t8	11			ns	RD# setup time
t9	0			ns	RD# hold time
t10	1		7.15	ns	CLKOUT TO TXE#
t11	11			ns	Write DATA setup time
t12	0			ns	Write DATA hold time
t13	11			ns	WR# setup time
t14	0			ns	WR# hold time

Table 2-2 Timing table

3   Software Configuration

With the FT2232H device, it is necessary to install the FTDI D2XX driver on the PC. Please visit

http://www.ftdichip.com/Drivers/D2XX.htm and download and install the necessary driver which matches

the PC.

The table below is taken from the FT2232H datasheet. It indicates that it is necessary to set the FT245

mode by configuring the EEPROM to 245 FIFO mode before developing a software application to access

data under FT245 style Sync FIFO mode. Configuring the EEPROM is illustrated in chapter3.1. Developing

software application is illustrated in chapter4

	SYNC 245 FIFO
EEPROM configured	YES
Application Software configured	YES

Table 3-1 Configuration Using EEPROM and Application Software

3.1 EEPROM Setting

Connect the FT2232H to a PC via USB cable, if the driver is installed already, the EEPROM settings can be programmed using either FTDIs MPROG or FT_PROG utilities (downloadable from FTDI website. Any one of these utilities can be used to set port A to "245 FIFO" mode in EEPROM. Figure 3-1 for the setting illustrates this using MPROG.
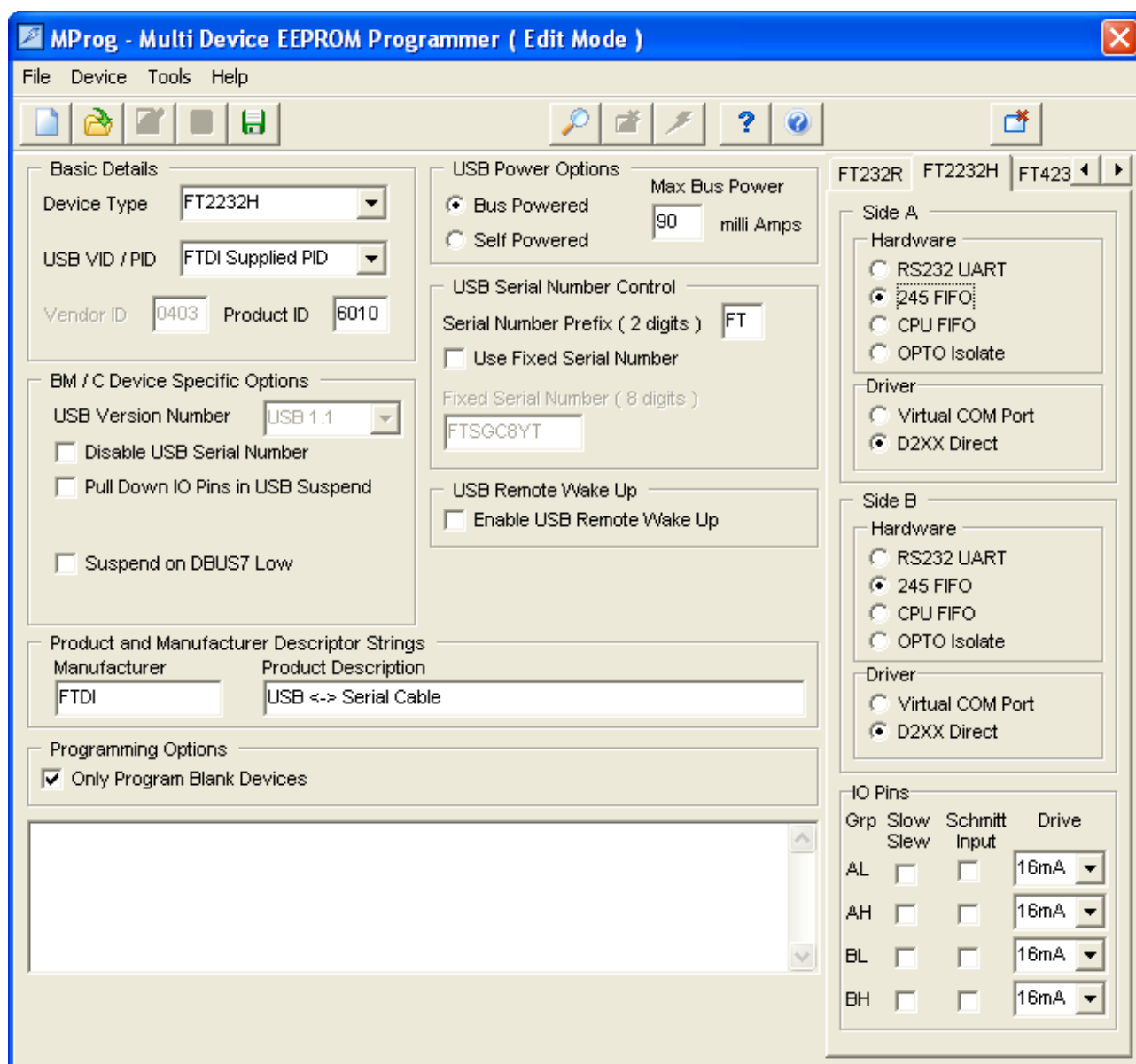

Figure 3-1 MProg Tool Setting

4   Application Development

As shown in Table3.1, it is necessary to develop an application to access data under FT245 style Sync FIFO mode. In following section describes how to do this.

In the application code, it is firstly necessary to openPort A of the FT2232H.

Next send command FT_SetBitMode(Mask, 0x40) to the FTDI driver to switch the FT2232H to FT245 Synchronous FIFO mode (this mode uses the RX & TX buffer of port B, so port B cannot be used for any other purpose in this mode).

Then we can send command FT_SetLatencyTimer(ftHandle, latencytime) to the FTDI driver to configure latency time, which is the receive buffer timeout that is used to flush remaining data from the receive buffer. If you don't call this command, the default setting is 16ms, it can be set at 1 ms intervals between 2ms and 255 ms.

The command FT_SetUSBParameters() can then be used to set the USB buffer size for the required data transfer. The buffer sizes must be set to a multiple of 64 bytes between 64 bytes and 64k bytes. The default size is 4KB. It is recommended changing this to 64KB.

Next the command, FT_SetFlowControl(), should to be called and parameter *usFlowControl* used to set FT_FLOW_RTS_CTS. This is configures the device driver to avoid data loss.

When the FT2232H has been set to FT245 Synchronous FIFO mode, the CLKOUT pin will output 60MHz a clock. Observing this with an oscilloscope is a good check to make sure the interface has entered FT245 Synchronous FIFO mode. If the waveform edges do not appear sharp enough, then the drive strength of the IO can be increased by altering the EEPROM values using MPROG or FT_PROG.

Although the RX and TX buffers are 4KB, they only use 2x 512 bytes for each buffer under FT245 Synchronous FIFO mode (maximum USB2.0 packet size under BULK mode is 512 bytes). The two buffers continually swap between each other to increase the performance.

To use the software commands, it is necessary to download the ftd2xx.dll, ftd2xx.lib, ftd2xx.h from the FTDI CDM driver. Details are available in the following programmers guide on how to use the software commands D2XX_Programmer's_Guide(FT_000071).pdf.

4.1 Code example

The following section gives an application example. This code is not guaranteed and is provided for illustration only. It is not supported by FTDI.

Example code

```
FT_HANDLE ftHandle;
FT_STATUS ftStatus;
UCHAR Mask = 0xff;
UCHAR Mode;
UCHAR LatencyTimer = 16; //our default setting is 16

ftStatus = FT_Open(0, &ftHandle);
if(ftStatus != FT_OK)
{
// FT_Open failed return;
}
Mode = 0x00; //reset mode
ftStatus = FT_SetBitMode(ftHandle, Mask, Mode);
delay_ms(10);
Mode = 0x40; //Sync FIFO mode
ftStatus = FT_SetBitMode(ftHandle, Mask, Mode);

if (ftStatus == FT_OK)
{
    ftStatus = FT_SetLatencyTimer(ftHandle, LatencyTimer);     ftStatus =
FT_SetUSBParameters(ftHandle,0x10000,0x10000);
    ftStatus = FT_SetFlowControl(ftHandle,FT_FLOW_RTS_CTS,0,0);
//access data from here
}
else
{
// FT_SetBitMode FAILED!
}
FT_Close(ftHandle);
```

4.2 Getting The Best Performance

Performance depends on the interaction between the FTDI device and external system. If the external system uses a polling method to check the detect pins (RXF#, TXE#) then this may delay the data transfer speed. To get the better performance connect the RXF# and TXE# pins to the external system interrupt pins.

When transferring large amounts of data, in order to get best performance, it is recommended to send the following commands to the FTDI driver at the initial setup stage.

```
ftStatus  = FT_SetLatencyTimer(ftHandle, 2);

ftStatus  = FT_SetUSBParameters(ftHandle,0x10000, 0x10000);
```

The following example code gives an example of how to calculate performance of "write data mode". This code is provided for illustration only and is not supported by FTDI.

```
#define OneSector 64*1024

#define SectorNum 2000

    FT_HANDLE ftHandle;

    FT_STATUS ftStatus;

    UCHAR Mask = 0xff;

    UCHAR Mode;


    ftStatus = FT_Open(0, &ftHandle);

    if(ftStatus != FT_OK)

    {

    // FT_Open failed return;

        printf("FT_Open FAILED! \r\n");

    }


//set interface into FT245 Synchronous FIFO mode

    Mode = 0x00; //reset mode

    ftStatus = FT_SetBitMode(ftHandle, Mask, Mode);
```

```
Sleep(1000);

Mode = 0x40; //Sync FIFO mode

ftStatus = FT_SetBitMode(ftHandle, Mask, Mode);


if (ftStatus != FT_OK)

{

// FT_SetBitMode FAILED!

    Printf("FT_SetBitMode FAILED! \r\n");

}

FT_SetLatencyTimer(ftHandle, 2);

FT_SetUSBParameters(ftHandle,0x10000, 0x10000);


FT_SetFlowControl(ftHandle, FT_FLOW_RTS_CTS, 0x0, 0x0);

FT_Purge(ftHandle, FT_PURGE_RX);


DWORD EventDWord;

DWORD RxBytes;

DWORD TxBytes;

DWORD BytesReceived;

char RxBuffer[OneSector];


LARGE_INTEGER lPreTime, lPostTime,lFrequency;

QueryPerformanceFrequency(&lFrequency);

QueryPerformanceCounter(&lPreTime);


DWORD dwSum = 0;


for(int i=0; i< SectorNum;i++)
```

```
{

  //ftStatus = FT_GetStatus(ftHandle,&RxBytes,&TxBytes,&EventDWord);

 // if((ftStatus == FT_OK) && (RxBytes >= OneSector))

  {

    ftStatus = FT_Read(ftHandle,RxBuffer,OneSector,&BytesReceived);

      if (ftStatus == FT_OK)

      {

      // FT_Read OK

         //printf("Read one sector bytes!\r\n");

         dwSum += BytesReceived;

      }

      else

      {

      // FT_Read Failed

      }

  }

          //check the end condition and quit

}


QueryPerformanceCounter(&lPostTime);

float lPassTick = lPostTime.QuadPart – lPreTime.QuadPart;

float lPassTime = lPassTick/(float)lFrequency.QuadPart;


printf("Received data bytes:%d \r\n", dwSum);

printf("Time passed:%f \r\n", lPassTime);
```

4.3 Write Data Mode

When an external system writes data to a PC via FT2232H, it is referred to as write data mode.

External system – When the external system is ready to transmit data, the external system should first check the TXE#. The external system must wait until TXE# goes low, it should then pull WR# signal low, before sending data bytes to D0~D7 at the CLKOUT frequency. If the external system detects TXE#=high, then it should stop data transferring and pull WR# signal high. The external system must then continue to poll the TXE# signal.

PC application – The application in PC side should poll the RX buffer to read data which has been transferred by the external system until the data transfer task is done.

Example code

```
#define OneSector 1024
FT_HANDLE ftHandle;
FT_STATUS ftStatus;
DWORD EventDWord;
DWORD RxBytes;
DWORD TxBytes;
DWORD BytesReceived;
char RxBuffer[OneSector];
  //set interface into FT245 Synchronous FIFO mode
  While(1)
  {
     ftStatus = FT_GetStatus(ftHandle,&RxBytes,&TxBytes,&EventDWord);
     if((ftStatus == FT_OK) && (RxBytes >= OneSector))
     {
       ftStatus = FT_Read(ftHandle,RxBuffer,RxBytes,&BytesReceived);
       if (ftStatus == FT_OK)
       {
          // FT_Read OK
       }
       else
       {
          // FT_Read Failed
       }
     }
             //check the end condition and quit
```

```
    }
```

4.4 Read Data Mode

When an external system wants to read data from PC via FT2232H, it is referred to as read data mode.

PC application – The application at the PC side sends data to FT2232H using the call FT_Write() command. This step is repeated until the data transfer is complete.

External system – The external system must poll until RXF# = low. It should then pull OE# = low then RD#=low after one clock delay, and read data bytes from D0~D7 at the CLKOUT frequency. When the external system detects RXF#=high, then it should stop reading data and pull OE# = high and RD#= high. The external system should then continue to poll the RXF # signal until data transfer task is complete.

Example code

```
#define OneSector 1024
FT_HANDLE ftHandle;
FT_STATUS ftStatus;
DWORD EventDWord;
DWORD RxBytes;
DWORD TxBytes;
DWORD BytesWritten;
char TxBuffer[OneSector];
  //set interface into FT245 Synchronous FIFO mode
  While(1)
  {
    ftStatus = FT_GetStatus(ftHandle,&RxBytes,&TxBytes,&EventDWord);
    if((ftStatus == FT_OK) && (TxBytes == 0))
    {
      ftStatus = FT_Write(ftHandle, TxBuffer, sizeof(TxBuffer), &BytesWritten);
      if (ftStatus == FT_OK)
      {
        // FT_Write OK

      }
      else
      {
```

```
                // FT_Write Failed

        }

    }

    //check the end condition and quit

}
```

5   Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1,2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales)   sales1@ftdichip.com
E-mail (Support)   support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com
Web Site URL   http://www.ftdichip.com
Web Shop URL http://www.ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales)   tw.sales1@ftdichip.com
E-mail (Support)   tw.support1@ftdichip.com
E-mail (General Enquiries)   tw.admin1@ftdichip.com
Web Site URL   http://www.ftdichip.com

Branch Office – Hillsboro, Oregon, USA

Future Technology Devices International Limited (USA)
7235 NW Evergreen Parkway, Suite 600
Hillsboro, OR 97123-5803
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales)   us.sales@ftdichip.com
E-Mail (Support)   us.admin@ftdichip.com
Web Site URL   http://www.ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 408, 317 Xianxia Road,
Shanghai, 200051
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales)   cn.sales@ftdichip.com
E-mail (Support) cn.support@ftdichip.com
E-mail (General Enquiries)   cn.admin@ftdichip.com
Web Site URL   http://www.ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – Revision History

Revision History

Version draft	Initial draft	02/10/2009
Version 1.0	Initial release	23/10/2009
Version 1.1	Added chapter 4.2 Getting the Best Performance	30/10/2009
Version 1.2	Edited figure 3-1 MProg tool setting	05/03/2010