Future Technology Devices International Ltd.

Application Note AN_100

Using the FT232R/FT245R with an External Crystal or Oscillator

Document Reference No.: FT_000067

Version 1.2

Issue Date: 2011-08-01

This document provides details of the steps required to replace the internal oscillator of the FT232/245R IC with an external clock source.

Future Technology Devices International Ltd (FTDI)

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 (Support): support1@ftdichip.com Web: http://www.ftdichip.com

Table of Contents

1 Introduction

The FTDI FT232R and FT245R devices have an integrated clock oscillator to simplify USB designs and reduce component count. In some applications it may be necessary to use an external clock source. For example when operating at a reduced VCC supply voltage or for improved EMC. This document describes how to disable the internal oscillator and enable the external clock source.

Users can use the example schematic and functional software code to begin their design. Note that software code is not guaranteed and not supported by FTDI. It is provided as an illustration only.

2 External Oscillator – Hardware Requirements

With the internal oscillator enabled both the OSCI pin and the OSCO pin are disconnected from the signals internally to the chip and should be left unconnected on the PCB.

Enabling and disabling the internal oscillator is done in the EEPROM as described in Section 3.0. By default a FT232R/245R device is delivered with the internal oscillator enabled.

When using an external crystal a crystal must be connected across the OSCI and OSCO pins as shown in Figure 2.1.

The crystal frequency must be 12.0MHz ± 0.5%. An AT-cut crystal is recommended. The FT232R and FT245R have internal load capacitors and so external capacitors will not be required in many applications. The values are 6.27pF for OSCI and 30pF for OSCO.

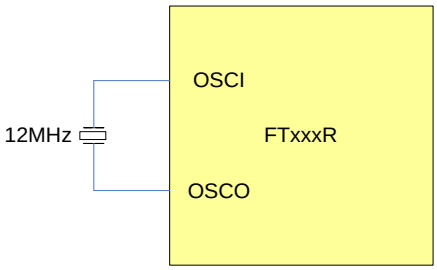

Figure 2.1 Crystal Connection

If using an external oscillator or other clock source then the frequency must also be 12.0MHz ± 0.5%. It must also drive at +5V CMOS levels or be AC coupled to centre around +VCC/2. In this case the source drives the OSCI pin. The OSCO pin is not connected.

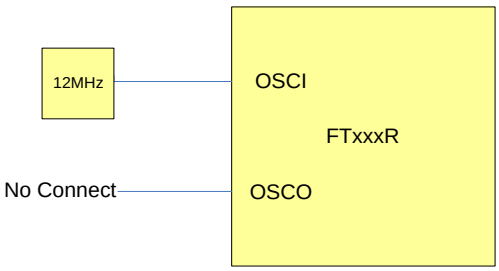

Figure 2.2 Oscillator Connection

Note: Some applications require using an external oscillator and operate at VCC= +3.3V. When powering up the device for the very first time, it must have VCC > +4.0V in order to start correctly and communicate on the USB bus (for example, the power supply could come from the USB VBUS supply +5.0V). The device can then be re-programmed using one of the utilities described below.

3 External Oscillator – EEPROM modification requirements

To disable the internal oscillator the EEPROM must be re-configured. This section shows three different ways to achieve this.

3.1 FT_Prog Utility

This EEPROM modification can be made using the FT_Prog utility which is available free from the FTDI website. The 'Use External Oscillator' box should be ticked as shown in Figure 3.1 below. Please refer to the user guide for full details of how to use FT_Prog. The FT_Prog utility and user guide are available from the link below:

http://www.ftdichip.com/Support/Utilities.htm#FT_Prog

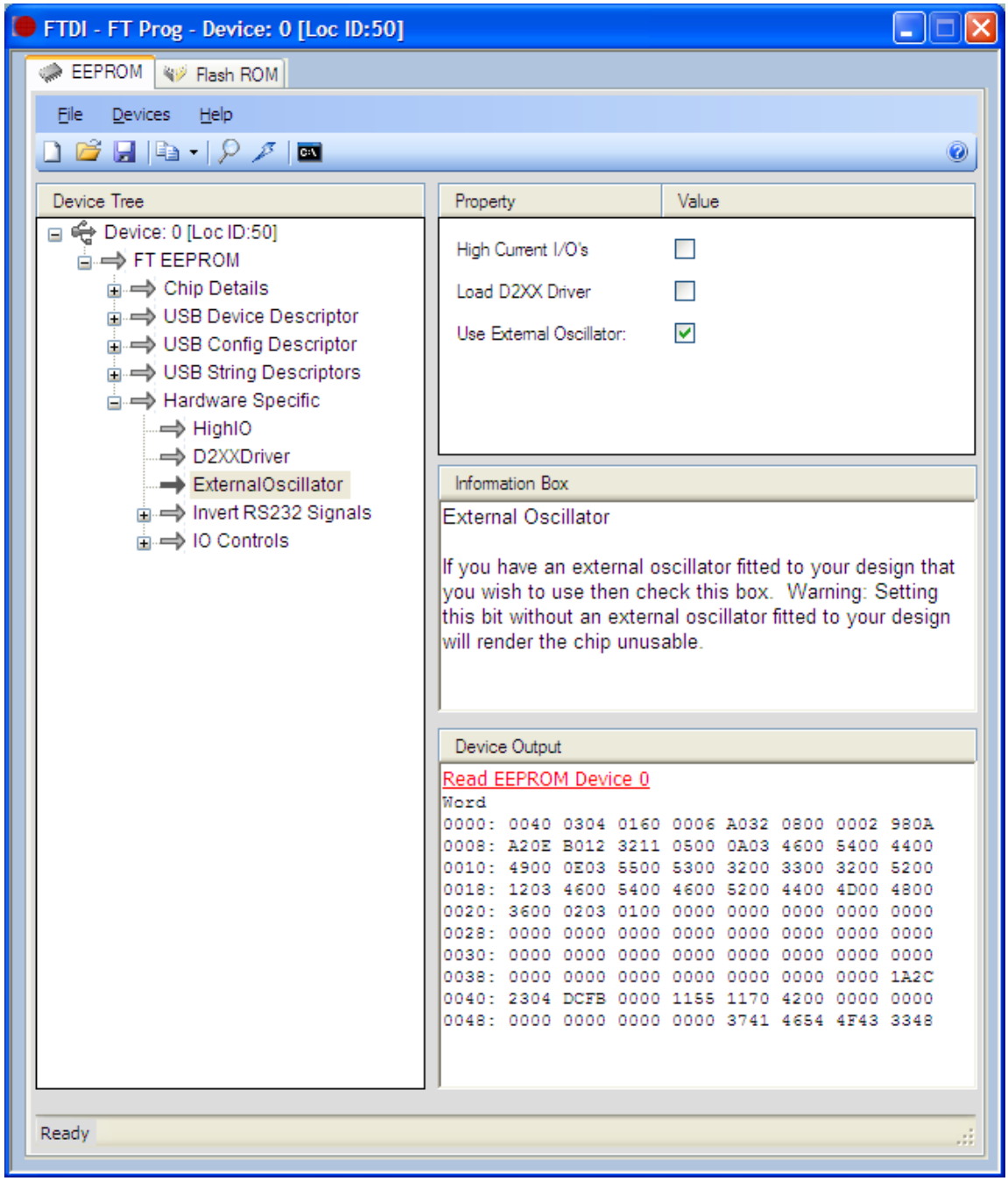

Figure 3.1 Enabling external oscillator in FT_Prog

3.2 Sample C++ Application

The sample code below shows how this can be done using a C++ application.

NOTE: This code is not supported or guaranteed by FTDI.

```
#include <windows.h>
#include <stdio.h>
#include "ftd2xx.h"

int main(int argc, char* argv[])
{
	FT_HANDLE fthandle;
	FT_STATUS status;

 status = FT_Open(0, &fthandle);

	    if(status != FT_OK)
	        printf("open status not ok %d\n", status);

	FT_PROGRAM_DATA ftData;

	WORD VendorIdBuf = 0x0403;
	WORD ProductIdBuf = 0x6001;
	char ManufacturerBuf[32];
	char ManufacturerIdBuf[16];
	char DescriptionBuf[64];
	char SerialNumberBuf[16];

	ftData.Signature1 = 0x00000000;
	ftData.Signature2 = 0xffffffff;
	ftData.Version = 2; //'0 = 'BM device, 1 = 'C Device, 2 = 'R device

	ftData.VendorId = VendorIdBuf;
	ftData.ProductId = ProductIdBuf;
	ftData.Manufacturer = ManufacturerBuf;
	ftData.ManufacturerId = ManufacturerIdBuf;
	ftData.Description = DescriptionBuf;
	ftData.SerialNumber = SerialNumberBuf;

	ftData.MaxPower = 90;
	ftData.PnP = 1;
	ftData.SelfPowered = 0;
	ftData.RemoteWakeup = 0;

	//'R features require section below
	ftData.UseExtOsc; // Use External Oscillator
	ftData.HighDriveIOs; // High Drive I/Os
	ftData.EndpointSize; // Endpoint size
	ftData.PullDownEnableR; // non-zero if pull down enabled
	ftData.SerNumEnableR; // non-zero if serial number to be used
	ftData.InvertTXD; // non-zero if invert TXD
	ftData.InvertRXD; // non-zero if invert RXD
	ftData.InvertRTS; // non-zero if invert RTS
	ftData.InvertCTS; // non-zero if invert CTS
	ftData.InvertDTR; // non-zero if invert DTR
	ftData.InvertDSR; // non-zero if invert DSR
	ftData.InvertDCD; // non-zero if invert DCD
	ftData.InvertRI; // non-zero if invert RI
	ftData.Cbus0; // Cbus Mux control
	ftData.Cbus1; // Cbus Mux control
	ftData.Cbus2; // Cbus Mux control
	ftData.Cbus3; // Cbus Mux control
	ftData.Cbus4; // Cbus Mux control
	ftData.RIsVCP; // non-zero if using VCP drivers

	status = FT_EE_Read(fthandle, &ftData);
```

```
		if (status != FT_OK)
			printf("EE_Read status not ok %d\n", status);
		else
		//It is optional to print out all the parameters.
		{
			printf("Signature1 =  0x%04x\n", ftData.Signature1);
			printf("Signature2 =  0x%04x\n", ftData.Signature2);
			printf("Version =  0x%04x\n", ftData.Version);

			printf("VendorID =  0x%04x\n", ftData.VendorId);
			printf("ProductID =  0x%04x\n", ftData.ProductId);
			printf("Manufacturer =  %s\n", ftData.Manufacturer);
			printf("ManufacturerID =  %s\n", ftData.ManufacturerId);
			printf("Description =  %s\n", ftData.Description);
			printf("SerialNumber =  %s\n", ftData.SerialNumber);
			printf("MaxPower =  %d\n", ftData.MaxPower);
			printf("PnP =  %x\n", ftData.PnP);
			printf("SelfPowered =  %x\n", ftData.SelfPowered);
			printf("RemoteWakeup =  %x\n", ftData.RemoteWakeup);

			printf("Use Ext Osc =  %x\n", ftData.UseExtOsc);
			printf("High Drives =  %x\n", ftData.HighDriveIOs);
			printf("Endpoint Size =  %x\n", ftData.EndpointSize);
			printf("Pull Down Enabled =  %x\n", ftData.PullDownEnableR);
			printf("Serial Number Enabled =  %x\n", ftData.SerNumEnableR);
			printf("Invert TXD =  %x\n", ftData.InvertTXD);
			printf("Invert RXD =  %x\n", ftData.InvertRXD);
			printf("Invert RTS =  %x\n", ftData.InvertRTS);
			printf("Invert CTS =  %x\n", ftData.InvertCTS);
			printf("Invert DTR =  %x\n", ftData.InvertDTR);
			printf("Invert DSR =  %x\n", ftData.InvertDSR);
			printf("Invert DCD =  %x\n", ftData.InvertDCD);
			printf("Invert RI =  %x\n", ftData.InvertRI);
			printf("CBUS0 =  %x\n", ftData.Cbus0);
			printf("CBUS1 =  %x\n", ftData.Cbus1);
			printf("CBUS2 =  %x\n", ftData.Cbus2);
			printf("CBUS3 =  %x\n", ftData.Cbus3);
			printf("CBUS4 =  %x\n", ftData.Cbus4);
			printf("RIsVCP =  %x\n", ftData.RIsVCP);
		}

	//use external clock.

	ftData.UseExtOsc = 1;	//Enabling use of external oscillator

	status = FT_EE_Program(fthandle, &ftData);
	status = FT_Close(fthandle);

		if (status != FT_OK)
		printf("EE_Program status not ok %d\n", status);
		else
			printf("EE_Program status ok %d\n", status);

	return 0;

}
```

CAUTION: IF THE INTERNAL OSCILLATOR IS DISABLED IN THE EEPROM, THE DEVICE WILL NOT FUNCTION UNTIL AN EXTERNAL CLOCK SOURCE IS APPLIED.

THIS EXTERNAL CLOCK SOURCE WILL BE NECESSARY IF THE APPLICATION NEEDS TO RE-ENABLE THE INTERNAL CLOCK SOURCE.

3.3 Sample Visual Basic.NET Application

The sample utility below accomplishes the same EEPROM modification through a Visual Basic program. To avoid accidentally programming other FT2xxR devices that may be plugged into the host PC, this program is configured to only work when a single FTDI device is installed.

NOTE: This utility is provided on an as-is basis and is not supported or guaranteed by FTDI.

This basic dialog shows the necessary information to interact with the user.

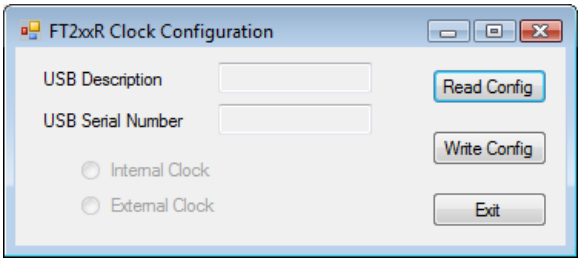

Figure 3.2 FT2xxR Clock Configuration – Initial Dialog

When the "Read Config" button is clicked, the USB description (Product Description in FT_Prog) and USB Serial Number are read from the EEPROM, as well as the current clock setting, and displayed.

The user simply needs to select the appropriate radio button and click "Write Config". If the external clock option is selected, a warning appears, asking for confirmation that the target circuit indeed has an external clock or crystal.

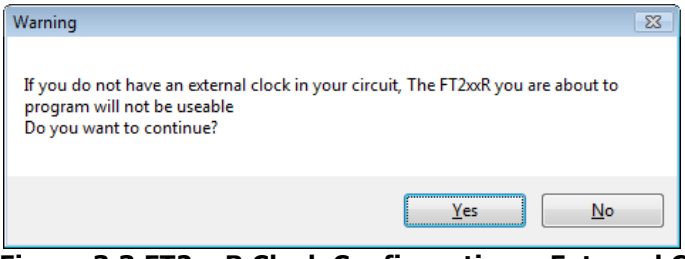

Figure 3.3 FT2xxR Clock Configuration – External Clock Warning

There are additional information dialog boxes that will appear to indicate when the EEPROM has been read or written, and to reset the target upon exiting the program.

Other EEPROM settings can be modified in the same manner and added to the dialog box.

The following D2XX function calls, as defined by the Managed .NET wrapper, are used to read and write the EEPROM:

- **GetNumberOfDevices** = Obtains the number of FTDI devices connected to the host PC. If the number <> 1, then the program will exit.
- **GetDeviceList** = Obtains the enumerated information including the USB Product String and USB Serial Number that were read by the operating system upon enumeration.
- **OpenByDescription** = Obtains the handle to the FT2xxR device.
- **ReadFT232REEPROM** = Reads the content of the FT2xxR EEPROM and populates the data structure.
- **WriteFT232REEPROM** = Writes the new settings as indicated in the dialog box.
- **CyclePort** = Forces the operating system to re-enumerate the FT2xxR

Note that the same .NET wrapper can be used in any of the Visual Studio.NET languages.

CAUTION: IF THE INTERNAL OSCILLATOR IS DISABLED IN THE EEPROM, THE DEVICE WILL NOT FUNCTION UNTIL AN EXTERNAL CLOCK SOURCE IS APPLIED.

THIS EXTERNAL CLOCK SOURCE WILL BE NECESSARY IF THE APPLICATION NEEDS TO RE-ENABLE THE INTERNAL CLOCK SOURCE.

4 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.support@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 - List of Figures and Tables

List of Figures

Appendix B – Revision History

Version 1.00	First Release	19/08/2008
Version 1.10	Added VB.NET Example.	19/03/2009
	Updated correct Company address information	06/05/2009
Version 1.20	Updated Figure 2.2 and page 3 text to show OSCO as no-connect instead of grounded	01/08/2011
	Added section showing how to select external oscillator with FT_Prog	