

Application Note

BRT_AN_063

FT9xx Programming, Debugging and Troubleshooting

Version 1.0

Issue Date: 29-08-2023

This document shows FT9xx programming and debugging methods and trouble shooting when developing with FT9xx MCU.

Use of Bridgetek devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold Bridgetek harmless from any and all damages, claims, suits or expense resulting from such use.

Bridgetek Pte Ltd (BRTChip) 1 Tai Seng Avenue, Tower A #03-05, Singapore 536464 Tel: +65 6547 4827 Web Site: <u>http://www.brtchip.com</u> Copyright © Bridgetek Pte Ltd



Table of Contents

1 Int	troduction
2 Pro	ogramming Methods5
2.1	Using the FT9xx Programming Utility5
2.1.	1 Programming via One-Wire
2.1.	2 Programming via DFU
2.1.	3 Generating a Binary Image File10
2.2	Using FT9xxProg.exe Command Line Utility11
2.2.	1 Command Line
2.2.	2 Eclipse
2.3	Using the Python Script Utility15
3 De	bugging Methods17
4 Pro	ogramming Errors19
4.1	The MCU is Not Powered/Connected
4.1.	1 Programming Utility
4.1.	2 Command Line Utility
4.2	The UMFTPD2A is Not Connected to the PC
4.2.	1 Programming Utility
4.2.	2 Command Line Utility
4.3	Active Debug Session Open20
4.3.	1 Programming Utility
4.3.	2 Command Line Utility
4.4	FT90x vs FT93x
4.5	Incorrect Board State21
5 De	bugging Errors
5.1	No MCU Connected
5.2	No UMFTPD2A Connected
53	Release Build 24
5.5	Project Build Error 24
J. 1 E E	
5.5	CIERCE C C C C C C C C C C C C C C C C C C
5.6	Build Configuration Name26



6 USB DFU Programming Errors 27
6.1 No DFU Device27
6.2 Binary File Preprocessing27
7 Other Tips and Tricks
7.1 MicroMatch Connection28
7.2 UMFTPD2A drivers28
7.3 Restore the Bootloader28
7.4 Reinstall the FT9xx Toolchain29
7.5 FT9xx Debugging29
8 Conclusion
9 Contact Information
Appendix A– References
Document References32
Acronyms and Abbreviations32
Appendix B – List of Tables & Figures
List of Tables
List of Figures
Appendix C– Revision History



1 Introduction

Bridgetek's MCU family consists of <u>FT90x</u> and <u>FT93x</u> MCUs.

Based upon Bridgetek's proprietary FT32 high-performance, 32-bit RISC core, the FT9xx series provides a plethora of connectivity options, making it the ideal choice for advanced technology bridging solutions. By executing instructions from program memory in RAM, rather than Flash Memory, the FT9xx can operate at true Zero Wait States (0WS) up to 100MHz with 310 DMIPS performance.

The <u>FT9xx Toolchain</u> provides developers with debug and programming abilities which are used in the development and production stages of a project.

This document shows programming and debugging methods and details common problems when developing with FT9xx MCUs and how to overcome them.

This version of the document specifically deals with programming the FT9xx on Microsoft Windows systems. Please contact Bridgetek for details of alternative provisions for Linux-based systems.



Document No.: BRT_000285 Clea

Clearance No.: BRT#200

2 Programming Methods

There are three ways to program the FT9xx MCU as described in this section.

A programming board such as the UMFTPD2A is required for Sections 2.1 and 2.2.

2.1 Using the FT9xx Programming Utility

Bridgetek has provided a GUI utility to program the FT9xx MCU. This is provided with the FT9xx Toolchain and can be found on the desktop after installation. Figure 1 shows the initial launch window.

🔹 FT9xx Programming Utility	-		×
Operation Help			
Operation:			
One-Wire Programming	BRI TEC	DGING CHNOLO	GIES
Scan for Devices			
Start			
Press "Scan for Devices" to find devices using One-Wire debugger.			

Figure 1 - FT9xx Programmer

The operation of the utility is selected by the "Operations" box. The operations available are shown in Figure 2.

🄹 FT9xx Programming Utility	-		×
Operation <u>H</u> elp			
Operation:			
One-Wire Programming 🛛 🗸	BRI	DGING	OGIES
One-Wire Programming	I LA	STINUE	OOILD
USB (DFU) Programming			
UART (Bootloader) Programming			
Binary File Generation			
Deta Lan Develand			
Data Log Download			
D2XX Configuration Setup			
DZAX Configuration File			

Figure 2 - FT9xx Programming Operation Selection



2.1.1 Programming via One-Wire

The first operation is "One-Wire Programming". This will program a binary image (produced by the compiler with the ".bin" extension) to a device.

First click on the "Scan for Devices" button. This will perform a scan for programming devices and any FT9xx device connected to them.

Note: Once a Scan for Devices operation has been performed the device will be reset by the programmer if another operation is selected.

After the scan is complete the first device found will be selected. An example is shown in Figure 3. Click on the button beside the "Binary File" box to browse for a file to program to the device. The type of the device is automatically filled in and cannot be changed.

The option for Flash Memory or Program Memory can be selected to change where the binary file is programmed. If it is in Program Memory then it will be run from there after programming is complete. For Flash Memory the device will be reset to allow the program to be run.

peration Help				
<u>H</u> elp				
Operation:				
One-Wire Program	nming ~		BF TE	RIDGING CHNOLOGIE
Scan for Devices				
Target:				
Programmer:	FT4UHELNA 🗸	Device:	UMFTPD2A A	
Туре:	FT900 🗸	Bootloader:	0x0108 (0x0000)	
Device Storage: ● Flash ○ Prog	gram Memory			Uerify
Bootloader:				
) Keep Existing	🔾 Default 🛛 🔾	Custom O Ex	clude OFactory	Restore
Binary File:				
C:\workspace\he	lloworld.bin			
Add Config File				
er au coning rite				

Figure 3 – First Device Selected After Scan for Device

A bootloader option is included. This can be used for advanced operations on the device.

The Keep Existing option will prevent the program from overwriting any bootloader currently on the device, this may be because it has been modified previously.

The Exclude option must only be used when the device does not need a bootloader and will not require debugging. It is not recommended to use this option.

The Custom option allows modification of the default bootloader before it is programmed to the device. When the bootloader presents a DFU interface it will use the VID, PID and BCD from these settings. The bootloader timeout is a time that it waits after reset for a UART connection from the PC system before running the program stored in Flash Memory. Options are shown in Figure 4.



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting

Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

🤹 FT9xx Programmir	ng Utility			-	- 🗆	×
Operation <u>H</u> elp						
Operation:						
One-Wire Programm	ming ~				BRIDGING) OGIES
Scan for Devices						
Target:						
Programmer:	FT4UHELNA $$	Device	:	UMFTPD2A A		
Туре:	FT900 🗸	Bootlo	ader:	0x0108 (0x0000)		
Device Storage: O Flash O Progr	ram Memory					erify
Bootloader:						
○ Keep Existing (🔾 Default 🛛 🔾	ustom	OExclude	e 🔿 Facto	ry Restore	
Custom:						
VID:			0403			
PID:			OFDE			
BCD:			2300			
Timeout:			10000			
Binary File:						
C:\workspace\hell	loworld.bin					
Add Config File						
		Sta	rt			

Figure 4 – Bootloader Custom Options

A Factory Restore will not program any image to the device, just erase the Flash Memory and restore the default bootloader to the device.

🤹 FT9xx Program	ming Utility		_	
<u>Operation</u> <u>H</u> elp				
Operation: One-Wire Progra Scan for Device:	mming V		BRID TECH	GING INOLOGIES
Target: Programmer: Type:	FT4UHELNA V	Device: Bootloader:	UMFTPD2A A	
Bootloader:	O Default O	Custom O Exc	clude OFactory Res	tore
		Start		

Figure 5 – Factory Restore

The Add Config File option is used when another binary file is to be placed in Program Memory or Flash Memory at a set location. For example, a program may use a config file for additional information which could be device specific. The D2XX library for FT9xx will use a config file for setup information for the D2XX ports created.



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting

Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

🍕 FT9xx Programming U	tility		_		×
Operation <u>H</u> elp					
Operation:					
One-Wire Programming	~		BI TE	RIDGING	ES
Scan for Devices					
Programmer: F	T4UHELNA 🗸	Device:	UMFTPD2A A		
Type: F	т900 🗸	Bootloader:	0x0108 (0x0000)		1
Device Storage: O Flash O Program	Memory			🗌 Verify	
Bootloader: O Keep Existing	efault 🔿 Cu	stom O Exclud	e 🔿 Factory	Restore	
Binary File:					
C:\workspace\hellowd	rld.bin				
Add Config File					
C:\workspace\myconf	ig.bin				
Config Address (hex): 1	e000				
		Start			

Figure 6 – Setting a Config File

2.1.2 Programming via DFU

A USB DFU interface is pre-programmed into production ICs and modules.

This allows users to program via the USB device interface rather than using the UMFTPD2A programming module. This is ideal for production or field updates. It's not suited for development purposes as the DFU interface can be easily removed by programming firmware devoid of the USB DFU capability.

The programmable binary file generated from the Eclipse IDE requires a DFU-suffix added and file padding before it can be programmed over the USB DFU interface. Figure 7 shows the programming utility with the operation set to "DFU Suffix Operations".

FT9xx Programming Utility	-		×
Operation Help			
Operation:			
DFU Suffix Operations	BRI TEC	DGING HNOLO	OGIES
Target: Type: FT900 ∨			
Binary File:			
C:\workspace\USBH Example HID to UART\FT90x_Debug\USBH Example HID to	o UART.bir		
Add Config File			
DFU Suffix: O Auto O Custom O Modify O Remove O Check			
Start			

Figure 7 – Add DFU-Suffix



The default suffix will be suitable for use with the built-in DFU firmware on production ICs. DFU Firmware can be modified to accept different settings to prevent incorrect images being loaded. This is achieved be adding a "custom" suffix as shown in Figure 8.

DFU Suffix Operations	BRIDGING TECHNOLOG
Target:	
Type: FT900 \checkmark	
Binary File:	
C:\workspace\USBH Example HID to UART\FT90x	_Debug\USBH Example HID to UART.bin
Add Config File	
DFU Suffix:	
○ Auto	⊖ Check
VID:	0403
PID:	6999
Version:	0123

Figure 8 - Add Custom DFU-Suffix Options

Once this one-time step is complete, the FT9xx Programming Utility can be used to program via the USB DFU interface as shown in Figure 9 and Figure 10.

🔹 FT9xx Programming Utility		-		×
Operation Help				
Operation: USB (DFU) Programming ~ Scan for Devices		BR	DGING CHNOLO	OGIES
	Start			
•				

Figure 9 – Program via USB DFU



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting

Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

			_		~
peration Help					
Operation:					
USB (DFU) Program	nming ~		BRI TEC	DGING HNOL(OGIE
Scan for Devices					
Target:					
Programmer:	00000000 ~	Device:	0403:0fde		
Туре:	FT900 🗸				
Add Config File					

Figure 10 – USB DFU Programming

2.1.3 Generating a Binary Image File

If an image file needs to be generated for use, then the "Binary File Generation" option is used. This works similarly to the other methods except that the output is an image file rather than programming the device.

The image is the size of the entire Flash Memory and includes the bootloader.

🤹 FT9xx Programming Utility			- 0	\times
<u>O</u> peration <u>H</u> elp				
Operation: Binary File Generation	~		BRIDGIN	G LOGIES
Target: Type: FT900 ~				
Bootloader:				
◯ Keep Existing O Default	◯ Custom	◯ Exclude	O Factory Restore	
Binary File: C:\workspace\helloworld.bin Add Config File				
Output File:				
C:\workspace\image.bin				
	Stat			
	Start			

Figure 11 – Image File Generation



Figure 11 shows a file called "image.bin" being created from one called "helloworld.bin".

2.2 Using FT9xxProg.exe Command Line Utility

Bridgetek has also provided a command line utility (included within the FT9xx Toolchain). This can be run from a command prompt, script or another utility. It can also be run from within Eclipse for FT9xx.

2.2.1 Command Line

From a command prompt or PowerShell on Windows, the program can be run.

C:\workspace> FT9xxProg.exe -loadflash '.\FT90x_Debug\USBH Example HID to UART.bin' onewire
Initializing...
Device = FT900
Erasing...
Still erasing...
Flash programming progress:
[===============] 100%

The --help parameter passed to the utility will print the latest command line arguments and their meaning to the screen.

C:\workspace> FT9xxProg.exe --help

The command line arguments are split into "Operations", "Interfaces" and "Options" sections.

Operations:	
No operation	List available devices for programming.
-h,	Show help message and exit.
help	
-f <file>,</file>	Load the binary <file> into the flash memory.</file>
loadflash <file></file>	
-p <file>,</file>	Load the binary <file> into the program memory.</file>
loadpm <file></file>	
<pre>-b [settings],</pre>	Program the bootloader to the top 4 kB of the flash memory.
loadbl [settings]	Various settings can be specified, see the Settings section below.
-e,	Display the chip ID and the CRC16 checksum of the flash.
info	When a serial number is not specified it will also list all available
	programming devices.
-C <file>,</file>	Check whether binary <file> has a DFU-suffix attached, if so, get</file>
<pre>checksuffix <file></file></pre>	the VID, PID and bcd from the dfu-suffix of the binary file.
-E,	Edit the bootLoader settings and write back the edited bootloader
editBL	back to predefined location.
<pre>-g <file> [settings],</file></pre>	Fix the binary file so that it can be programmed via the DFU.
dfufix <file> [settings]</file>	Various options can be specified in settings.
	An output file must be specified with theoutfile/-o flag.
-y,restore	Restore the chip to the factory settings.
-j <file>,</file>	Prepare the flash image from binary file provided.
<pre>prepareImage <file></file></pre>	
-r,	Reset the chip.
chipReset	
-V,	The version of this utility.
version	
Options:	



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting

Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

-s <serial number="">,</serial>	Serial number to select a specific programming device.
serialNo <serial< td=""><td>If no serial number is specified then the first available device is used.</td></serial<>	If no serial number is specified then the first available device is used.
number>	Available devices can be listed with theinfo/-e operation.
-v,	Verify the flash memory contents after flash programming. Only used for
verify	one-wire programming.
-n,	Prevent the device from resetting after the image is downloaded.
noReset	
-x,	Prevent the bootloader from being programmed to the last 4 kB of the
nobl	flash.
-k,	Keep the existing bootloader at the top 4 kB of the flash.
keepoldbl	
-c <file>,</file>	Embed a second binary file to the main binary file specified by the
conf <file></file>	loadflash/-f flag. Embed from the address specified by the
	confaddr/-a flag below.
-a <addr></addr>	Specify the address to embed a second binary file with theconf/-c
confaddr <addr></addr>	flag above. Hexadecimal.
-o <file>,</file>	Specify the output file for operations that require one.
outfile <file></file>	
-i <file>,</file>	Specify the input file for operations that require one.
infile <file></file>	
-L,Verbose	Verbose output for extra debugging info.
-D <0,1>,	Specify device type to be operated on.
device <0,1>	• FT900 0,
	• FT930 1.
	The default is FT900.
	This is optional when using operations that target the one-wire interface.
	It is mandatory when using the UART interface or file operations.

Settings:

occungs.		
-V	<value>,</value>	A 4-digit hex value which specifies the idVendor.
vid <value></value>		
-P	<value>,</value>	A 4-digit hex value which specifies the idProduct.
pid <value></value>		
- B	<value>,</value>	A 4-digit hex value which specifies the bcdDevice.
bcd <value></value>		
-T	<value>,</value>	When modifying the bootloader settings this is the timeout (in ms,
timeout <va< td=""><td>lue></td><td>decimal) before the bootloader starts verifying the firmware.</td></va<>	lue>	decimal) before the bootloader starts verifying the firmware.
		For the UART bootloader this is the timeout to detect the bootloader
		presence. This is signalled by a chip reset or at power on.

Attention:

Please make sure the programmer is the only device connected when programming or ensure the serial number flag is set correctly.

2.2.1.1 List All Available Devices

The command with no parameters will list all available devices using the one wire interface:

```
C:\workspace>FT9xxProg.exe
Number of devices: 4
Device 0:Serial Number: FT4UHELNA, Description: UMFTPD2A A.
Device 2:Serial Number: FT4UHELNB, Description: UMFTPD2A D.
Device 3:Serial Number: , Description: UMFTPD2A B.
Description: .
```

The information there can be used to select a device for programming.



An UMFTPD2A board consists of 4 interfaces. The first interface is the "A" interface and provides the One-Wire programming interface. The third interface is the "C" interface and that is taken to the UART pin headers on the UMFTPD2A.

To select the device required the with the "--serial/-s" argument only the serial number of the "A" interface is used. So, for example, the above programmer boards would be selected with "--serial FT4UHELNA".

2.2.1.2 Read the Chip ID

Read the chip ID and the flash checksum of the first device using the one wire interface:

```
C:\workspace>FT9xxProg.exe --info
Initializing...
Chip ID: 0x09000002
Flash CRC16: 0xF7D1
```

The first device is selected in this example. To choose a different device use the "--serial/-s" parameter to the program.

Reading the Chip ID in this way will halt execution of the program on the FT9xx and it will require to be reset afterwards if the program in flash is to be re-run. The program memory is altered when this command is run.

2.2.1.3 Program a Binary File

Program helloworld.bin into the flash via the one-wire interface and verify the image:

```
C:\workspace>FT9xxProg.exe -f "helloworld.bin" -O -V
Initializing...
Device = FT900
Erasing...
```

Still erasing...

```
Flash programming progress:
[======] 100%
```

The same operation using long options:

C:\workspace>FT9xxProg.exe --loadflash "helloworld.bin" -onewire --verify

2.2.1.4 Generate a Binary Image File with a Bootloader

The program can be used when there is no device or programmer connected. Operations such as the "--prepareImage/-j" operation which compiles a binary file and a bootloader into a whole flash image may need to have the device type specified explicitly. There is no default device type so it must be set with the "--device/-D" option.

This example will form a whole flash image for an FT93x using the file helloworld.bin and attaching a bootloader. The output will go to image.bin.

```
C:\workspace>FT9xxProg.exe -j "helloworld.bin" --outfile image.bin --device 1
Initializing...
FT930 Flash image written to file image.bin
```



2.2.1.5 Write a Bootloader to the Device

Rewrite the bootloader with the default settings to the top 4 kB of the flash memory via the one-wire interface:

C:\workspace> FT9xxProg.exe -b -0

2.2.1.6 Modify the Bootloader on the Device

Write a customized bootloader with a timeout of 3 seconds to the top 4 kB of the flash memory via the one-wire interface:

C:\workspace> FT9xxProg.exe -b -0 -T 3000

The timeout parameter alters the length of time that the bootloader will wait for a connection from a PC system at power up or reset. This allows a program running on a PC system connected to the UARTO of the FT9xx to program the device. See Section <u>2.3</u> for instructions for using this interface.

2.2.1.7 Restore the Device to Factory Settings

C:\workspace> FT9xxProg.exe --restore -0

This command will erase the Flash Memory and restore the default bootloader to the device.

2.2.1.8 Add a DFU Suffix to an Image

To use the DFU programming method in Section 2.1.2 an informational suffix has to be added to the image.

C:\workspace>FT9xxProg.exe -g "helloworld.bin" --outfile image.bin --device 1 -V 0403

The options for the DFU suffix are "--vid/-V", "--pid/-P" and "--bcd/-B" can be used in this command. These options tell the DFU utility what device type the image is targeted for.

2.2.2 Eclipse

The command line programming utility is used by a Debug Configuration within Eclipse when debugging, see Section $\underline{3}$. It can also be used as a Run Configuration or as an External Tool.

To use the command line utility as an External Tool is can be added as a new External Tool Configuration as shown in Figure 12. When it runs, it will output to a console window as in Figure 13.

The path to the utility is clearly shown in the figures using the environment variable FT9XX_TOOLCHAIN to locate the installation directory of the toolchain and programmer. The installation will add these directories to the PATH environment variable so this is not necessary.



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting

Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

				_ 0					
eate, manage, and run configuratio	ins			0					
un a program				1-					
' 🖻 🗫 🗎 🖻 🏹 🕶	Name: New_configuration (2)								
rpe filter text	× 🔲 Main 🤣 Refresh 🔒 Build 🚾 Environment 🔲 Common								
💁 Program	Location:								
Q New_configuration (2)	\${env_var:FT9XX_TOOLCHAIN}/programmer/FT9xxProg.exe	\${env_var:FT9XX_TOOLCHAIN}/programmer/FT9xxProg.exe							
		Browse Workspace	Browse File System	Variables					
	Working Directory:								
	C:\workspace\USBH Example HID to UART								
		Browse Workspace	Browse File System	Variables					
	Arguments:								
	-f "FT90x_Debug\USBH Example HID to UART.bin" -O -v			-					
				variables					
	Note: Enclose an argument containing spaces using double-guotes (").			variables					
	Note: Enclose an argument containing spaces using double-quotes (").			variables					
	Note: Enclose an argument containing spaces using double-quotes (").			v Variables					

Figure 12 - FT9xxProg in Eclipse

(2) 🗸 🔅 🔁 🖬 👘 🕅 🖬	!! ♥ ▶ □ ■ № 3. ◎ .2 ↦ ≂ ▓! ▓ ☆ • ◎ • � • ! ♂ • 입 • 집 • ♡ • ♥ ↔ + =
Console ×	= X 💥 🗟 💀 🥏 🕶 🗹 🗢 🗖
<terminated> New_configuration (2) [Program] C:\</terminated>	Users\gordon.mcnab\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain\programmer\FT9xxProg.exe [pid: 27620] (1 Jun 2023, 13
<pre>Initializing. Device = FT9(<terminated> New_configurat</terminated></pre>	ion (2) [Program] C:\Users\gordon.mcnab\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain\programmer\FT9xxProg.exe [pid: 2
Erasing	
Still erasing	
Flash programming progress:	
[=====] 100%	
Verifying	
Verification passed	

Figure 13 - FT9xxProg in Eclipse Console

2.3 Using the Python Script Utility

There is an alternative method of programming the FT9xx using only a UART on the PC system. This is in the form of a python script. It will wait until it recognises a reset or power-on event from the FT9xx bootloader and commence a programming sequence. The default time it will wait is 200 ms. This can be changed with the Timeout value for a custom bootloader. An example of changing it is in Section 2.2.1.5 for the command line programmer utility.

A specially created binary file, which covers the whole of the Flash Memory, must be used for this method. The file always includes a replacement bootloader. An example of how to generate this file is in Section 2.2.1.4.

The UARTO interface on the FT9xx must be connected to the PC system and configured as a VCP (Virtual COM Port). This does not need flow control and the utility will configure the VCP to the required settings.



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting

Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

C:\workspace> FT9xxProg.pyloadflash 'image.bin'device 0 -com COM5	
!!!Welcome to FT9xx Programming Utility!!!	
Copyright Bridgetek Pte Ltd	
	-
Device = FT900	
Initializing	
Reset the device to be programmed.	
Waiting for bootloader	

Figure 14 - FT9xxProg.py Windows Command Line Console

At this point the program will wait until the bootloader is activated on the FT9xx. This can be done by resetting the device or power-cycling the device.

Once connected the programming will proceed as follows.

[==========] 100% Figure 15 - FT9xxProg.py Windows Command Line Console



3 Debugging Methods

The FT9xx Toolchain includes a seamless debugging feature. When you click on FT9XX DEBUG within the IDE, it programs the device and launches the debugging environment automatically.

This can be accessed from the **Run** \rightarrow **Debug Configurations** menu as shown in Figure 16.

The <u>UMFTPD2A</u> is required for debugging as it is only supported over the one-wire interface.

Debug Configurations	×
Create, manage, and run configurations	To the second se
Image: Second Secon	Name: FT9XX DEBUG Main Image: Debugger Source Common Project: AN_414_FT90x_USBD_UVC_Webcam Browse C/C++ Application: C/C++ Application: FT900_Debug/AN_414_FT90x_USBD_UVC_Webcam.elf Image: Search Project Browse Build (if required) before launching Build Configuration: Use Active Image: Search Project Browse Obsende auto build Disable auto build Disable auto build Image: Search Project Image: Search Project Image: Search Project Image: Search Project Image: Search Project Image: Search Project Image: Search Project Image: Search Pr
Filter matched 8 of 10 items	Re <u>v</u> ert Appl <u>v</u>
0	Debug

Figure 16 - Debug Configurations

Note that when this has been run once, it can be found via the debug icon as shown in Figure 17.



Product Page Document Feedback



The Debug environment is shown in Figure 18 with key features highlighted.



Figure 18 – Debug Environment



Document No.: BRT_000285

Clearance No.: BRT#200

4 Programming Errors

This section describes the common programming errors.

4.1 The MCU is Not Powered/Connected

The FT9xx Development Module or custom hardware must be powered along with the UMFTPD2A Debug/Programming module as shown in Figure 19.



Figure 19 – Hardware Connection

If it is not powered, then errors will be seen as shown in Figure 20 or Figure 21 depending on which programming method is being used.

The same error is encountered when there is no physical connection between the two boards via the MicroMatch connector.

4.1.1 Programming Utility

The FT9xx Programming Utility will see the UMFTPD2A programmer module but the FT9xx device will not be found, so the user cannot continue to the next window as shown in Figure 20.



Figure 20 – Programming Utility No MCU

4.1.2 Command Line Utility

The command line utility will report an unknown device as shown in Figure 21.

```
C:\workspace\USBH Example HID to UART> FT9xxProg.exe -e
FT9xxProg: error: Failed to open device.
FT9xxProg: error: UMFTPD2A Module not found. Please check the connection and close all
active debug connections.
```

Figure 21 – Command Line Utility No MCU



To resolve this error, ensure that the FT9xx MCU is powered and connected to the UMFTPD2A via the MicroMatch connector. In some cases, the FT9xx needs to be power cycled to resolve this issue.

4.2 The UMFTPD2A is Not Connected to the PC

If the UMFTPD2A is not connected to the PC, the FT9xx tools cannot find the D2XX interface used to program them. The same error as in Section 4.1 will be shown.

This is also the case when the UMFTPD2A USB drivers have not been installed correctly. See Section $\frac{7.2}{7.2}$ for more information.

If the UMFTPD2A is not connected to the host then the green LED on the board will not be lit.

To resolve these errors, ensure that UMFTPD2A is connected to the PC and to the FT9xx MCU via the MicroMatch connector.

4.2.1 Programming Utility

The Programming Utility will fail to see the UMFTPD2A Programmer and report the same error as in Figure 20 – Programming Utility No MCU

4.2.2 Command Line Utility

The command line utility will report an unknown device as shown in Figure 21 – Command Line Utility No MCU

4.3 Active Debug Session Open

When there is an active debug session open, or there is another instance of the Programming Utility or command line Utility running, the D2XX interface which is used for programming the FT9xx is claimed and cannot be used. The same error as in Section 4.1 will be shown.

If Eclipse is debugging with the UMFTPD2A programming board, terminate the active debug session as shown in Figure 22.



Figure 22 – Terminate Active Debug Session

Otherwise, check that there are no more instances of the programmer running on the system. Open "Task Manager", press the Windows "Start" button then type "Task Manager", or press Win-X (Windows key + the "X" key together) then the "T" key for <u>T</u>ask Manager.



Once it is open then click on the "Details" tab, search for and select "FT9xxProg.exe" as shown in Figure 23. Click on the "End Task" button to terminate the task and allow other connections to the device.

<mark>الل</mark>	Task Manager	Q Type a name, pub	isher, o	r PID to search	n				- 0	1
\equiv		Details						n new task	⊘ End task	
₽	Processes	Name	PID	Status	U	CPU	Memory (a	Archite	Description	
		ExpressConnectServi	4036	Running	S	00	164 K	x64	Windows Service	Wra
-∿	Performance	FileCoAuth.exe	30380	Running	g	00	1,384 K	x64	Microsoft OneDriv	/eFil.
		FileSyncHelper.exe	31520	Running	S	00	1,680 K	x64	Microsoft OneDriv	/eFil.
D)	App history	fontdrvhost.exe	1900	Running	U	00	80 K	x64	Usermode Font Dr	river
		fontdrvhost.exe	1752	Running	U	00	2,740 K	x64	Usermode Font Dr	river
T/A	Startup apps	ft32-elf-gdb.exe	5020	Running	g	00	172 K	x86	ft32-elf-gdb.exe	
	Startup upps	FT9xxProg.exe	28464	Running	g	00	792 K	x64	FT9xxProg.exe	
20	lisors	FT9xxProgGUI.exe	6692	Running	g	00	11,328 K	x64	FT9xxProgGUI.exe	
0	0 Sel S	GitHubDesktop.exe	23076	Running	g	00	10,504 K	x64	GitHubDesktop.ex	e
_	Details	GitHubDesktop.exe	23644	Running	g	00	19,300 K	x64	GitHubDesktop.ex	e
_	Details	🛞 GitHubDesktop.exe	1852	Running	g	00	2,284 K	x64	GitHubDesktop.ex	e
		😨 GitHubDesktop.exe	8348	Running	g	00	73,968 K	x64	GitHubDesktop.ex	e
ŝ	Settings	igfxCUIServiceN.exe	4616	Running	S	00	16 K	x64	igfxCUIService Mo	odule

Figure 23 – Terminate Programmer in Task Manager

4.3.1 Programming Utility

The Programming Utility will fail to see the UMFTPD2A Programmer as shown in Figure 20.

4.3.2 Command Line Utility

The command line utility will report UMFTPD2A Module not found as shown in Figure 21.

4.4 FT90x vs FT93x

The FT9xx Toolchain allows the user to build a project for the FT90x or FT93x MCU as shown in Figure 24.

🗢 w	workspace - C/C++ - UART Example 1/uart_example_1.c - Ecl						
File	Edit	Source	Refactor	Navigate	Search	Proje	ct f
: 🖻	• 🖫	6	• 🔨 •	D 🔪		- 😂	- [
	Project	Explo	1 FT900	_Debug		} , ⊽	
	🐣 Fre	eRTOS	2 FT900	_Release			
	🐉 FT_	App_F	3 FT930	_Debug			
⊳	🔑 FT_	App_(4 FT930	_Release			
<u>ь</u> 1	🔑 FT	Ann Play	Video	-		_	

Figure 24 - FT9ox and FT93x Build

Care must be taken to build for the correct target MCU. The programmer will not show any errors in this case but if the wrong build is programmed, the MCU will not work.

Eclipse will not allow a program compiled for FT90x to be programmed to an FT93x and vice versa. If, however, the names of the Build Configuration within the Eclipse project are changed then it is possible to write or debug the wrong type of device.

4.5 Incorrect Board State

If the UMFTPD2A board or the FT9xx device gets into an incorrect state then the message in Figure 25.

C:\workspace\USBH Example HID to UART> FT9xxProg.exe --loadflash '.\FT90x_Debug\USBH Example HID to UART.bin' -onewire Can't get the lock from MCU! FT9xxProg: error: Chip ID incorrect or unknown.

Bridgetek

Document No.: BRT_000285 Clearance No.: BRT#200

Figure 25 – Incorrect Board State

To rectify this re-run the programmer and the board and programming board are in a consistent state.



5 Debugging Errors

This section describes some common debug errors and how to overcome them.

5.1 No MCU Connected

The error shown in Figure 26 can occur when the MCU is not powered or there is no physical connection between the FT9xx and UMFTPD2A via the MicroMatch connector.

```
**** Incremental Build of configuration FT900_Debug for project UART Example 1 ****
make all
'Invoking: FT90x Display Image Size'
ft32-elf-size --format=berkeley -x "UART Example 1.elf"
  text
          data
                  bss
                         dec
                                 hexfilename
0x10c4 0x1b4
                        4736
                                1280UART Example 1.elf
                  0x8
'Finished building: SIZE'
Build Finished (took 967ms)
Loading target device started...
Checking device...
C:/Program Files (x86)/Bridgetek/FT9xx Toolchain/Toolchain/programmer/FT9xxProg.exe --info
Error encountered while checking device. No MCU connected.
Loading Finished!
```

Figure 26 – Debug No MCU Connected

To resolve this error, ensure that the FT9xx MCU is powered and connected to the UMFTPD2A via the MicroMatch connector.

5.2 No UMFTPD2A Connected

If the UMFTPD2A is not connected or not enumerated with the PC, the error shown in Figure 27 is shown.

```
**** Incremental Build of configuration FT900 Debug for project UART Example 1 ****
make all
'Invoking: FT90x Display Image Size'
ft32-elf-size --format=berkeley -x "UART Example 1.elf"
  text
          data
                  bss
                          dec
                                 hexfilename
0x10c4 0x1b4
                  0x8
                         4736
                                1280UART Example 1.elf
'Finished building: SIZE'
. .
Build Finished (took 281ms)
Loading target device started...
Checking device...
C:/Program Files (x86)/Bridgetek/FT9xx Toolchain/Toolchain/programmer/FTxxProg.exe --info
```



Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

Error encountered while checking device. Could not detect UMFTPD2A or VII programmer module.

Loading Finished!

```
Figure 27 – Debut No UMFTPD2A Connected
```

To resolve this error, ensure that the FT9xx MCU is powered and connected to the UMFTPD2A via the MicroMatch connector.

5.3 Release Build

If the debug session is started when the release build is selected as shown in Figure 28, the debug session will launch but the release build contains no debug symbols.



The source code won't be able to be seen in the debug session as shown in Figure 29.

💽 main() at 0x528 🔅	
No source available f	or "main() at 0x528"
View Disassembly	

Figure 29 – Release Build No Source

To resolve this error, ensure that the Debug build is selected within Eclipse.

5.4 Project Build Error

If the project contains code errors, the debug cannot proceed because a .bin file is required to be programmed before the debug session can start. A typical build error can be seen in Figure 30.

```
**** Incremental Build of configuration FT930_Debug for project UART Example 1 ****
make all
'Building file: ../uart_example_1.c'
'Invoking: FT90x GCC Compiler'
ft32-elf-gcc -D__FT930__ -D__RAMSIZE=32K -D__PMSIZE=128K -I"C:/Program Files
(x86)/Bridgetek/FT9xx Toolchain/Toolchain/hardware/include" -00 -0g -g1 -fvar-tracking -
fvar-tracking-assignments -Wall -c -fmessage-length=0 -ffunction-sections -mft32b -
```



Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

5.5 FT90x vs FT93x Debug

The FT9xx Toolchain allows the user to build a project for the FT90x or FT93x MCU as shown in Figure 31.



Care must be taken to select the correct target MCU before launching the debug session or the error shown in Figure 32 will be seen.

```
**** Incremental Build of configuration FT930 Debug for project UART Example 1 ****
make all
'Invoking: FT90x Display Image Size'
ft32-elf-size --format=berkeley -x "UART Example 1.elf"
  text data
                  bss
                         dec
                                hexfilename
0x11dc 0x1ac
                              1394UART Example 1.elf
                  0xc
                        5012
'Finished building: SIZE'
. .
Build Finished (took 286ms)
Loading target device started...
Checking device...
C:/Program Files (x86)/Bridgetek/FT9xx Toolchain/Toolchain/programmer/FT9xxProg.exe --info
Error encountered while checking device. Device and binary does not match! Device is FT900.
Loading Finished!
```

Figure 32 - FT90x and FT93x Error

To resolve this error, ensure that the correct target build is selected within Eclipse.



Document No.: BRT_000285 Clearand

Clearance No.: BRT#200

5.6 Build Configuration Name

Eclipse gives 'device and binary mismatch error' if the build configuration name is other than 'FT9xx_Debug' or 'FT9xx_Release'.

Check the build configuration names by right-clicking on the project, selection **Properties** \rightarrow **C/C++ Build** \rightarrow **Settings**, then click on Manage Configurations.

PWM Example 1: Manage Configurations					
Configuration	Description	Status			
FT90x_Debug		Active			
FT90x_Release					
FT93x_Debug					
FT93x_Release					
Set Active	New Del	ete Rename			
	ОК	Cancel			

Figure 33 – Manage Configurations



6 USB DFU Programming Errors

Common USB DFU errors are detailed in the forthcoming sections.

6.1 No DFU Device

If there is no USB DFU interface available, the error is shown in Figure 34.

Figure 34 – No DFU Device	
	Close
No devices found during scan.	
USB (DFU) Programming	×

This can happen when the IC is not in factory programmed state, or there is no DFU interface available in the application firmware. It can also happen when the FT9xx device is not connected. To resolve this issue, there are a couple of options:

- Restore the bootloader. See Section <u>7.3</u> for more information.
- Include a USB DFU interface in your application firmware. More information can be found on the USB DFU interface in <u>AN 365 FT9xx API</u> <u>Programmers Manual</u>.

6.2 Binary File Preprocessing

In order to program via USB DFU, the .bin file has to be padded and a valid DFU-suffix added. If this is not done, the error shown in Figure 35 will be seen.



Figure 35 – DFU Caution

To resolve this error, ensure that pre-processing has been done on the .bin file. This can be done using the FT9xx Programming Utility as shown in Section 2.1.2. This is a one-time step on any binary file.



7 Other Tips and Tricks

This section details some other useful tips and tricks to help overcome any programming and debugging issues.

7.1 MicroMatch Connection

Ensure the MicroMatch connector between the UMFTPD2A and the FT9xx is the correct way round. There is an arrow which signifies pin 1 and the cable itself has a red side which helps with the connection as shown in Figure 36.



Figure 36 – MicroMatch Connection

7.2 UMFTPD2A drivers

Check that the UMFTPD2A drivers have been installed correctly as shown in Windows Device Manager. See Figure 37. COM10 to COM13 is the UMFTPD2A in this screenshot. The UMFTPD2A contains an FT4232H IC which is a four port USB device.

Ports (COM & LPT)
 Communications Port (COM1)
 Intel(R) Active Management Technology - SOL (COM3)
 USB Serial Port (COM10)
 USB Serial Port (COM11)
 USB Serial Port (COM12)
 USB Serial Port (COM13)

Figure 37 – Device Manager

The drivers can be easily uninstalled using <u>CDM Uninstaller</u>. When the device is plugged back into the PC, the drivers should install automatically via Windows Update otherwise see our <u>Installation</u> <u>Guides</u>.

7.3 Restore the Bootloader

Restoring the bootloader of the FT9xx MCU can be done using the FT9xx Programming Utility and UMFTPD2A only. See Figure 38.

This programs the MCU into a default factory state.



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

peration:					DOING	1
One-Wire Program	iming ~			TEC	CHNOLO	OGIE
Scan for Devices						
larget:						
Programmer:	FT4UHELNA 🗸	Device:	UMFTPD2A	Α		
Туре:	FT900 🗸	Bootloader:	0x0108 (0x0	000)		
Bootloader:						
Keep Existing	O Default)Custom OE	xclude OF	actory R	estore	

Figure 38 - Restore Bootloader

7.4 Reinstall the FT9xx Toolchain

Reinstalling the <u>FT9xx Toolchain</u> is always a last measure.

Ensure the default Python v2.7.10 is selected during the install. It comes packaged in the toolchain. Our python scripts have also been ported to work with version 3.x.

7.5 FT9xx Debugging

FT9xxProg.exe is run automatically when initiating the FT9XX DEBUG session via the Eclipse IDE. It provides a "gdbserver" interface for the "gdb" utility which Eclipse uses to perform the debugging.

Therefore, the debugging is affected by the same issues as the command line programmer utility.



8 Conclusion

This document shows programming and debugging methods, common debug and programming errors when developing with FT9xx MCU and how to overcome them. The tools provided by Bridgetek are comprehensive and are powerful tools to help developers create their own custom applications.



9 Contact Information

Refer to https://brtchip.com/contact-us/ for contact information.

Distributor and Sales Representatives

Please visit the <u>Sales Network</u> page for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Bridgetek Pte Ltd (BRTChip) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested Bridgetek devices and other materials) is provided for reference only. While Bridgetek has taken care to assure it is accurate, this assistance provided by Bridgetek. Use of Bridgetek devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless Bridgetek from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Bridgetek Pte Ltd, 1 Tai Seng Avenue, Tower A, #03-05, Singapore 536464. Singapore Registered Company Number: 201542387H.



Appendix A- References

Document References

FT90x and FT93x Product Page

FT9xx Toolchain

FT9xx Development Modules

CDM Uninstaller

Installation Guides

AN 365 FT9xx API Programmers Manual

Acronyms and Abbreviations

Terms	Description
DFU	Device Firmware Update
DMIPS	Dhrystone MIPS (Million Instructions per Second)
GDB	GNU Project debugger
GUI	Graphical User Interface
IC	Integrated Circuit
MCU	MicroController Unit
PC	Personal Computer
RAM	Random Access Memory
USB	Universal Serial Bus



Appendix B – List of Tables & Figures

List of Tables

NA

List of Figures

Figure 1 - FT9xx Programmer
Figure 2 - FT9xx Programming Operation Selection5
Figure 3 – First Device Selected After Scan for Device
Figure 4 – Bootloader Custom Options7
Figure 5 – Factory Restore7
Figure 6 – Setting a Config File
Figure 7 – Add DFU-Suffix8
Figure 8 – Add Custom DFU-Suffix Options9
Figure 9 – Program via USB DFU9
Figure 10 – USB DFU Programming10
Figure 11 – Image File Generation
Figure 12 - FT9xxProg in Eclipse15
Figure 13 - FT9xxProg in Eclipse Console15
Figure 14 - FT9xxProg.py Windows Command Line Console16
Figure 15 - FT9xxProg.py Windows Command Line Console16
Figure 16 - Debug Configurations17
Figure 17 – Debug Icon17
Figure 18 – Debug Environment
Figure 19 – Hardware Connection19
Figure 20 – Programming Utility No MCU19
Figure 21 – Command Line Utility No MCU19
Figure 22 – Terminate Active Debug Session
Figure 23 – Terminate Programmer in Task Manager21
Figure 24 - FT9ox and FT93x Build21
Figure 25 – Incorrect Board State
Figure 26 – Debug No MCU Connected
Figure 27 – Debut No UMFTPD2A Connected24
Figure 28 – Release Build24
Figure 29 – Release Build No Source
Figure 30 – Build Error25
Figure 31 - FT90x and FT93x Build25
Figure 32 - FT90x and FT93x Error25
Figure 33 – Manage Configurations



BRT_AN_063 FT9xx Programming, Debugging and Troubleshooting Version 1.0

Document No.: BRT_000285 Clearance No.: BRT#200

Figure 34 – No DFU Device	27
Figure 35 – DFU Caution	27
Figure 36 – MicroMatch Connection	
Figure 37 – Device Manager	28
Figure 38 – Restore Bootloader	29



Document No.: BRT_000285

Clearance No.: BRT#200

Appendix C- Revision History

Document Title:	FT9xx Programming, Debugging and Troubleshooting
Document Reference No.:	BRT_000285
Clearance No.:	BRT#200
Product Page:	https://brtchip.com/ft9xx-toolchain/
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial Release	29-08-2023