Application Note

AN_325

FT9XX Toolchain Installation Guide

Version 1.07

Issue Date: 2018-11-14

This guide documents the tools and methods required for building, programming and debugging the FT9XX series devices from BRTChip.

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.
Table of Contents

1 Introduction ................................................................. 4

1.1 Compiler: ft32-elf-gcc .................................................. 4
1.2 Assembler: ft32-elf-as .................................................. 4
1.3 Linker: ft32-elf-ld ...................................................... 4
1.4 Debugger: ft32-elf-gdb .................................................. 4
1.5 A useful utility: ft32-elf-objdump .................................... 5

2 Setting up the FT9XX Toolchain ......................................... 6

2.1 Installing the Toolchain .................................................. 6
2.1.1 Installing Java Runtime Environment Manually ................. 14
2.2 Verifying the installation ............................................... 15

3 Quick Start Guide: From creating to getting your application to run on the FT9XX MCUs .......................... 16

3.1 Creating a new project .................................................. 16
3.2 Building the project .................................................... 18
3.3 Programming the binary file into the chip ......................... 22
3.3.1 Eclipse Plugin ......................................................... 22
3.3.2 GUI Version .......................................................... 23
3.3.3 Command Line Version ............................................. 25
3.4 “Hello World” in action, and more .................................. 26

4 Setting up Eclipse for Debugging ........................................ 28

4.1 Build the application using the Debug configuration .............. 28
4.2 Starting a debug session ............................................... 28
4.3 Debugging the application in Eclipse ................................. 30
4.3.1 Watch variables in Eclipse Debug Perspective ................. 31
4.3.2 Og Compiler Option when debugging .......................... 31
4.4 Eclipse features supported by ft32-elf-gdb .......................... 32

5 Bridgetek Projects .......................................................... 33

6 Advanced Topics ............................................................ 36
List of Tables........................................................................................................54
List of Figures ........................................................................................................54
Appendix C – Revision History .................................................................57
1 Introduction

The free FT9XX toolchain is a port from the popular GNU toolchain which includes the following components:

- GCC based compiler
- ft32-elf-gcc (GCC) 8.0.0 20171111 (experimental) GNU Binary Utilities (binutils) based tools, most notably:
  - as - the assembler
  - ld - the linker
  - and some other useful tools such as objdump, ar, ranlib, addr2line, etc.
- GDB based debugger
- In addition, a plugin for the Eclipse IDE is also provided. This ‘Bridgetek FT9XX Eclipse plugin’ allows the FT9xx toolchain to integrate seamlessly into Eclipse and as a result, greatly simplify the development and debugging of applications for the FT9xx MCUs.

1.1 Compiler: ft32-elf-gcc

The FT9XX compiler is used similarly to standard GCC. It supports most GCC options such as -Wall, -O1, -O2...

Example: To compile a C file into an object file:

```bash
ft32-elf-gcc -c -o file.o file.c
```

1.2 Assembler: ft32-elf-as

The FT9XX assembler functions in the same way as the standard GNU assembler (GAS). The assembly files should be written using the GAS general syntax.

Example: To compile an assembly file into an object file:

```bash
ft32-elf-as -o file.o file.s
```

1.3 Linker: ft32-elf-ld

Typically running behind ft32-elf-gcc, the FT9XX linker performs two tasks. It first links all object files and libraries into a.out and then convert’s a.out into an executable file for FT9XX. Similar to the FT9XX compiler and assembler, the FT9xx linker supports most standard GNU linker options.

Example:

- To link various object files / libraries into an .elf file:
  ```bash
  ft32-elf-gcc -nostartfiles file1.o file2.o -L <libfolder> -l lib1 -l lib2 -o file.elf
  ```
- To convert file.elf into a FT9xx binary file, which can be programmed into the chips:
  ```bash
  ft32-elf-ld --oformat binary -o file.bin file.elf
  ```

1.4 Debugger: ft32-elf-gdb

The Bridgetek programmer/debugger module is needed for the communication between ft32-elf-gdb and the chip. The communication follows the GDB remote protocol. In addition to the debugger module, two software components are needed:

- GDB Bridge: for converting GDB commands into the debugger module commands
- Bootloader: for receiving & executing the debugger module commands
More information on how to use the FT9xx debugger can be found in section 6.1.3 of this document.

1.5 A useful utility: ft32-elf-objdump

ft32-elf-objdump displays various information about object files. Its usage is the same as standard GNU objdump.

Example: To disassemble file.elf into a text file
ft32-elf-objdump -d file.elf > disassembly.txt
2 Setting up the FT9XX Toolchain

2.1 Installing the Toolchain

The toolchain can be installed by running the setup wizard “FT9XX Toolchain Setup_version.exe”, which can be downloaded from the Bridgetek website. Please follow the steps in the wizard to complete the installation process. It is recommended to use the default settings for simplicity.

**Note:** all applications should be closed before the installation or a restart may be required.

![Figure 1 Toolchain Setup Wizard Dialog box](image1)

In the License Agreement dialog box, click **I Agree**.

![Figure 2 License Agreement Dialog box](image2)
Go through the Revision and Release information and click **Next**.

![Figure 3 Revision and Release Information Dialog box](image)

Select the Components and click **Next**. The toolchain has an option to install Python 2.7.10 (see Fig 4). You may opt out of the Python installation, if Python 2.7.x or later is present in your system, by unselecting the option for Python 2.7.10 in the Components Dialog box.

![Figure 4 Components Dialog box](image)

Click **Browse** and select a different file path for the FT9XX Toolchain installation. Alternatively, continue installing in the specified folder by clicking **Next**.
The toolchain components like Eclipse IDE, GNU toolchain, programmer utility, FT900 library headers and 3rd party library are installed in the selected path. If the specified path, which is a system path under ‘Program Files’ or ‘Program Files (x86)’ is chosen, one has to be aware of Virtual Store folders and files in the section 7.4.

![Figure 5 FT9XX Toolchain Install Location Dialog box](image)

Click **Browse** and select a different file path for installing FT9XX examples and documents. Alternatively, continue installing in the specified folder, by clicking **Install**.

![Figure 6 FT9XX Toolchain-Examples & Documents Install Location Dialog box](image)
The FT9XX Toolchain installation progress bar is displayed.

![Figure 7 FT9XX Toolchain - Installation Progress Window](image)

If Java is selected for installation, the following message is displayed. Please close any open Java Applications and click OK.

![Figure 8 Close any running Java Applications before launching JRE Installer](image)

Setup will then launch the JRE installer and the following window will be displayed.
Click **Install** and follow the instructions to install Java on the machine.

During installation, if a **Python 2.7.10 Setup** dialog box is displayed, select the appropriate option as required and click **Next**.
Select a different Destination Directory to setup Python. Alternatively, continue installing in the specified folder by clicking Next.

Select the Python features to be installed or continue with the default features and click Next.
Figure 13 Python Features Customization Dialog box

Python installation progress bar is displayed.

Figure 14 Python Installation Progress Dialog box
Click **Finish** to complete the Python installation.

![Python Installation Completion Dialog box](image)

**Figure 15 Python Installation Completion Dialog box**

The FT9XX Toolchain installation is continued.

![FT9XX Toolchain - Installation Progress Window](image)

**Figure 16 FT9XX Toolchain - Installation Progress Window**

Select the **Open AN_325** checkbox to start immediately after closing the Setup Wizard. Else leave it unchecked. Click **Finish** to complete the FT9XX Toolchain Setup.
After the installation, the toolchain can be found in the installation directory. The default location is "C:\Program Files\Bridgetek\FT9XX Toolchain" for 32-bit Windows and "C:\Program Files (x86)\Bridgetek\FT9XX Toolchain" for 64-bit Windows. This directory also contains the external utilities needed. The FT9XX drivers, sample applications and documents (if selected for installation) can be found in “My Documents\Bridgetek\FT9xx”.

2.1.1 Installing Java Runtime Environment Manually

The Toolchain requires the Windows MSI Installer (msiexec.exe) while installing the Java Runtime Environment (JRE). The MSI installer can only process one installation at a time. Under some conditions, msiexec.exe may have already been started by another Windows process during automatic Windows Update for example. If the installer detects another instance of msiexec.exe running in the background, the user will be prompted to either wait for the background MSI Installer to complete and retry after 5 seconds or to skip the JRE installation entirely. This is shown in Figure 18.
2.2 Verifying the installation

1. Open a Command Prompt window by typing "cmd" in "Windows Start button → Search box".
2. Type "ft32-elf-gcc --version" in the command prompt. It should give the following message:

   ft32-elf-gcc (GCC) 8.0.0 20171111 (experimental)
   Copyright (C) 2017 Free Software Foundation, Inc.
   This is free software; see the source for copying conditions. There is NO
   warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

   If this message appears, then the toolchain has been successfully setup.
3 Quick Start Guide: From creating to getting your application to run on the FT9XX MCUs

This Section guides you through the steps to create a new application, compile and program it into the chip. To debug your application, please refer to Section 4 - “Setting up Eclipse for Debugging”. For more information about the tools, as well as the advanced features, refer to Section 6 - “Advanced Topics”.

3.1 Creating a new project

Double click on the icon “Eclipse for FT9XX” to launch the Eclipse IDE.

![Eclipse for FT9XX Icon](image1)

Figure 19 Eclipse for FT9XX Icon

When you run Eclipse for the first time, it will ask you for the location of the workspace. Eclipse will create some files within this directory to manage the projects. Specify a folder of your choice and click OK.

![Eclipse Workspace Selection](image2)

Figure 20 Eclipse Workspace Selection

**Note:** The following message will be displayed if an existing workspace, which was created by an older version of Eclipse, is specified. As there may be some configurational changes in files related to workspace in the newer version of Eclipse which may cause issues, it is recommended to create a new workspace and import the existing projects to the new workspace.
To create a new C project in Eclipse, on the menu bar click “File → New → C Project”. The C Project wizard will open.

Give a name to the project, for example “Hello World”. By default, the new project will be created inside the workspace you have chosen. If you want to change it, uncheck the box “Use default location” and specify another location. Choose “Empty Project” for the project type and “Bridgetek FT9XX GCC” for the toolchain. This ensures all the relevant FT9XX include files are part of the project. Click Next.

In the next window, select FT900_* configurations if your project is to target FT900 series of MCU or FT930_* to target FT930 series, or both if you wish to target both series and click Next.
The last window is for the toolchain prefix and location. By default, the values will be prefilled as follows.

![Figure 23 Project Wizard - Build Configurations Selection](image)

Figure 23 Project Wizard - Build Configurations Selection

The last window is for the toolchain prefix and location. By default, the values will be prefilled as follows.

| FT9x compiler prefix | ft32-elf- |
| FT9x compiler path   | C:/Program Files (x86)/Bridgetek/FT9x Toolchain/Toolchain/tools/bin | Browse |

![Figure 24 C Project Wizard - Toolchain Details](image)

Figure 24 C Project Wizard - Toolchain Details

Click Finish to complete the New Project Wizard. A new FT9XX project will be created in Eclipse.

### 3.2 Building the project

After the wizard completes, some folders and an empty source file (main.c) will be created.

![Figure 25 New empty project structure](image)

Figure 25 New empty project structure
The main.c contains some default content:

```c
#include <stdint.h>
#include <ft900.h>

#if defined(__FT900__)
#define GPIO_UART0_TX 48
#define GPIO_UART0_RX 49
#elif defined(__FT930__)
#define GPIO_UART0_TX 23
#define GPIO_UART0_RX 22
#endif

int main(void){
    /* Enable the UART Device... */
    sys_enable(sys_device_uart0);
    /* Set UART0 GPIO functions to UART0_TXD and UART0_RXD... */
    gpio_function(GPIO_UART0_TX, pad_uart0_txd); /* UART0_TXD */
    gpio_function(GPIO_UART0_RX, pad_uart0_rxd); /* UART0_RXD */
    uart_open(UART0, /* Device */
              1, /* Prescaler = 1 */
              UART_DIVIDER_19200_BAUD, /* Divider = 1302 */
              uart_data_bits_8, /* No. Data Bits */
              uart_parity_none, /* Parity */
              uart_stop_bits_1); /* No. Stop Bits */

    /* Print out a welcome message... */
    uart_puts(UART0,
              "------------------------------------- \r\n" "Hello World! \r\n" "------------------------------------- \r\n");
    /* Now keep looping */
    while (1);

    return 0;
}
```

Now select the Build Configuration for either FT90X or FT93X via the project menu:
This can also be done via the Manage Configurations toolbar icon:

![Manage Configurations](image)

Figure 26 Build Configuration

Now the project can be built by clicking on the menu **Project → Build Project**, but note that there are a few options like right-click on the project → Build Project and the ![Build Project Icon](image) icon.

![Build Project](image)

Figure 27 Build Configuration

The console window at the bottom of the IDE shows the build status. If the build completes successfully, two files will be created - "Hello World.elf" and "Hello World.bin". The file to be programmed into the chip is "Hello World.bin". The .elf file is used for the debugger, as detailed in the next Section.
For syntax highlighting to work correctly as configurations are switched, the C/C++ Indexer must be configured to "work with the active build configuration". Refer to section 6.3.3 for more details on how to do this.
3.3 Programming the binary file into the chip

There are two ways to program a binary image into the chip.

The debugger may be used to program the binary image into flash or program memory prior to the start of a debug session. Alternatively, if debugging is not required, the FT9XX programmer utility may be used to program the binary image to flash. Unlike the debugger sessions, the programmer utility supports programming to flash memory only.

The programmer utility and Eclipse debugger both use the 1-wire interface to communicate with the FT9xx. Therefore, the utility and debugger shall not be used at the same time.

3.3.1 Eclipse Plugin

The FT9XX toolchain components, including the FT9XX Programmer, have been fully integrated to the Eclipse IDE to allow seamless debugging experience.

To program the compiled binary file into the chip, do the following:

1. On the Menu bar, select Run → Run Configurations...
2. In the Run Configurations window, expand the launch configuration type "FT9XX Application Run" and select the default instance "FT9XX RUN".
3. Click on the 'Run' button to start the download. This will internally launch FT9XX Programmer application. The Console window will show the progress of the download.
4. For succeeding run sessions, select Run → Run Last Launched.

Figure 31 Run menus - Run Configurations and Run Last Launched
Alternatively, the FT9XX Programmer can be accessed manually – either the GUI version or the command line version.

### 3.3.2 GUI Version

To run it, double click on the icon “FT9XX Programming Utility” created on your desktop, if selected during install, otherwise it can be found in:

C:\Program Files (x86)\Bridgetek\FT9xx Toolchain\Toolchain\programmer\dist
You can also open the programming utility from Eclipse by selecting it in the Bridgetek Utilities menu or the toolbar icon as highlighted in Figure 35:

![Bridgetek Utilities Menu](image)

**Figure 35 Bridgetek Utilities Menu**

After the splash message the following screen will appear.

![FT9XX Programmer - Work with One-Wire](image)

**Figure 36 FT9XX Programmer - Work with One-Wire**

Only the option ‘Program via One-wire interface’ in this screen is covered in this section. For information on the remaining options, please refer to the programmer utility help.

Select the “Program via One-wire interface” option and click Next. The next screen shows a list of supported devices that you might wish to program.

When a valid FT9xx and Programmer module are detected, the information will be displayed in the list. Select the device you wish to program and click Next to launch the programmer window.

![FT9XX Programmer - Device Selection](image)

**Figure 37 FT9XX Programmer - Device Selection**
In the programmer window, leave everything as default. Specify the location of the binary file and click Start. If the Verify check box is selected, an icon will show up next to the status bar to indicate whether the flash memory has been properly programmer.

![Figure 38 FT9XX Programmer – Flash and PM Screen](image)

More information on the utility can be found in the ‘About’ tab, then click on Help.

### 3.3.3 Command Line Version

FT900Prog.exe is available to run at a command prompt. Enter FT900Prog.exe to see the options available. See section 6.1.2 for more details.

This can also be run within Eclipse as an External Tool. See Figure 39 for settings found in Run → External Tools → External Tools Configurations. The argument string is:

```
-mf "${project_loc}/${config_name:${project_name}}/${project_name}.bin" -O -v
```
3.4 “Hello World” in action, and more...

The “Hello World” example above will send a message to a serial terminal via the FT9XX UART0 port. Open a terminal on your computer, for example Tera Term or HyperTerminal. Apply the following settings:

- Baud Rate: 19200
- Parity Bit: None
- Data Bit: 8
- Stop Bit: 1
- Flow Control: None

Now when you reset the MCU, the message will be printed to the terminal.
Congratulations! You have just completed your first project for FT9XX. The FT9xx toolchain comes with plenty of examples, which demonstrate a variety of features. If you have selected to install them in the Toolchain Installation Wizard, by default they can be found in:

"My Documents\Bridge tek\FT9xx\version\Examples"

The Eclipse project has already been setup for these examples, as suggested by the presence of two files - ".cproject" and ".project". Instead of creating a new project, you can simply import these projects into the workspace. To do this:

1. On the Menu bar, choose "File → Import"
2. In the Import window, choose "General → Existing Projects into Workspace" and click "Next".
3. In the next window, set the root directory to "My Documents\Bridge tek\FT9xx\version\Examples". The projects will be detected by Eclipse.
4. Select which projects you wish to import and click Finish to complete the importing process.

This is an example of how Eclipse would look like with the sample applications. Refer to AN_360 for more details about these applications.

Figure 41 FT9XX Examples
4 Setting up Eclipse for Debugging

Eclipse comes with an intuitive GUI for debugging applications. To enable this feature, Eclipse requires additional information about our debugger. The steps are presented below.

4.1 Build the application using the Debug configuration

The application should be built using the Debug configuration so that the debug information is available. It is the default build configuration but can be verified in the Project menu.

![Figure 42 Build Configuration](image)

Please note for FT93X: As of toolchain v2.3.0 it is recommended to disable the –mcompress option when using GDB as single stepping does not work properly with –mcompress. The option can be disabled in the project settings. See section 6.3.2.1 for more details.

4.2 Starting a debug session

The FT9XX toolchain components, including the FT9XX Programmer and FT9XX GDB Bridge, have been fully integrated to the Eclipse IDE to allow seamless debugging experience.

To program the compiled binary file into the chip and start a debugging session, do the following:

1. On the Menu bar, select Run → Debug Configurations...
2. In the Debug Configurations window, expand the launch configuration type “FT9XX Application Debug” and select the default instance “FT9XX DEBUG”. This default instance will setup the location of the ft32-elf-gdb.exe, connection type to “TCP”, host name to “localhost”, port number to “9998” as well as the project name and elf name.
3. Click on the 'Debug' button to start the downloading and debugging. This will internally launch the FT9XX Programmer application to program the binary and then launch the ft32-elf-gdb.exe and FT9XX GDB Bridge to start the debugging session. The Console window will show the progress of the download and debugging.

4. For succeeding debugging sessions, select Run ➔ **Debug Last Launched**.
4.3 Debugging the application in Eclipse

1. Start the debug session as stated in the previous section.
2. The Debug perspective will be opened. The execution will stop at the first line in main(), as shown below. Various debug commands (step into/over, resume, halt, stop, etc.) can now be accessed from the toolbar via buttons. Function variables, setting breakpoints and viewing physical memory in the memory tab, along with some other debug features are also available now.

![Console window after selecting launch configuration and clicking Debug](image)

![Eclipse Debug Environment](image)
**Note:** If there is an error message about missing source file as below, locate the source file that contains the main() function using the “Locate file...” button.

![Locate file or edit the source lookup path to include its location.](image)

**4.3.1 Watch variables in Eclipse Debug Perspective**

If the watch variables fail to update or display incorrect values, check that the following flags exist for the debug build (they are present by default in all projects created with Bridgetek Eclipse plugin)

-`-fvar-tracking -fvar-tracking-assignments`

![Debug flags](image)

**4.3.2 Og Compiler Option when debugging**

When compiling a project with no optimization (or `-O0`) some useful debugging information may not be generated at all, leading to possible unexpected results while debugging. To avoid this, it is recommended to turn on `-Og` option when no other optimization flags are used. The Bridgetek Eclipse plugin does this automatically.

Note that if multiple optimization options are used, only the last option will be effective.

4.4 Eclipse features supported by ft32-elf-gdb

- The features of Eclipse debug perspective, supported through ft32-elf-gdb, are:
  - Breakpoint creation.
  - Single stepping/stepping in/stepping out of functions
  - Watch variables
  - Assembly instruction stepping
  - Memory View
5 Bridgetek Projects

Besides the empty project used as the example in Section 3, there are several project types specific to Bridgetek. They can be found under "Others" in the project type selection window. Currently, there are three project types:

- D2XX Project
- Data Log (DLOG) Project
- FreeRTOS Project

For more details about these project types, refer to "AN_360 FT9XX Example Applications", which is included in the toolchain installation.

![Figure 50 Bridgetek Project Types](image)
The procedure to create a new project is similar to the empty project. When the wizard completes, a template source file will be generated. The template generated for the D2XX project is given below -

```
loop();

if (D2XXTEST_Detach)
{
    delayms(100);
    // Exit D2XX
    D2XX_exit();
}

// Take the system to GPU Mode. Provide timeout in ms to
// wait for re-connection with the host
STARTUP_GPU(8);
break;
}

if (D2XXTEST_Suspend)
{
    /* Suspend mode handling */

    suspendSource = 8;
    dbg("Enter Suspend Mode at Master \n\n");  // Output Suspend source info
    delayms(100);
}

// Set up remote wakeup
if (remoteWakeupEnabled)
{
    // Configuring GPIO pin to wakeup \n"
    
    #ifdef GPIO_REMOTE_WAKEUP
    
        if (1)
        {
            // Set up the pin
            gpio_dir(GPIO_PIN, GPIO_MODE_OUTPUT);
            gpio_out(GPIO_PIN, 1);  // Set up the pin
            // Attach an interrupt
            gpio_interrupt_enable(GPIO_PIN, GPIO_EDGE_FALLING);
            interrupt_attach(interrupt_gpio, &remote_gpio_int);  // Interrupt GPIO
        }
    
    #endif
}
```

**Figure 51 D2XX Project Template**

The template project can be compiled as it is but additional code is needed to customize it according to the user's need.

After selecting the project template, the wizard also allows selection of add-on third-party libraries that have been ported to work on FT9xx (see Figure 52). The source for the libraries are located in `<Installation directory>\Toolchain\hardware\3rdparty`. Currently there are three third-party libraries supported.

- **Tinyprint**: A tiny/lightweight general purpose printf function that could be used to replace the GNU library printf function.
- **FatFs** (based on version 0.12b): A FAT/exFAT filesystem module. FatFs accesses the storage devices via a simple media access interface which the FT9xx application should implement.
- **lwIP** (based on version 2.0.3): A lightweight TCP/IP stack for embedded device.
Figure 52 Add-on Libraries
6 Advanced Topics

6.1 Running the Toolchain from the command prompt

6.1.1 Compiling the sample applications using a Makefile

The FT9XX GNU toolchain can be used to compile source code from a command prompt in the same way the official GNU Toolchain is used, often with the help of a Makefile or a batch file.

The sample applications are available in "C:\Users\<User name>\Documents\Bridgetek\FT9xx\<Tool chain version>\Examples\", \ if you have installed them using the installation wizard.

NOTE: makefiles are not included with the toolchain installer.

6.1.2 Programming a binary image into the chip

The programmer can be found in the folder "programmer\dist" in the program installation directory (C:\Program Files (x86)\Bridgetek\FT9xx Toolchain). The command line programmer is FT900Prog.exe. The Toolchain is provided with a default bootloader. The bootloader is located at the top 4 KB of the flash memory (address 0x3F000 to 0x3FFFF). At boot, the FT9XX resets and executes instruction at 0x00000, jumping into the bootloader. The bootloader then performs the initializations needed and jumps to location 0x8c, which is the start of the user program. The bootloader is also needed to support debugging with the FT9XX port of GDB.

1. Run the tool FT900Prog.exe without any arguments, the options and usage will be printed. They will also be printed if the specified options are not valid. The most common usage is programming a binary file through the one-wire interface with the supplied bootloader. To do this, the command is:

   FT900Prog.exe -f <.bin file with path if needed> -O

   in which the options are:

   -f: programming the binary file into the flash. The path to the binary file must follow.
   -O: using the one-wire interface.

   If you want to verify the content of the flash memory after programming, specify "-v" in the command:

   FT900Prog.exe -f <.bin file with path if needed> -O -v

2. If the bootloader is not required, option "-x" can be specified, in which case the program will start executing from address zero and the command is:

   FT900Prog.exe -f <.bin file with path if needed> -O -x

   The supports for GDB debugging will not be available however.
6.1.3 Debugging the sample applications with ft32-elf-gdb

1. The applications must to be compiled with –g option (i.e. ft32-elf-gcc –g ...). An .elf file will be created which includes the debug information, for example GPIO/gpio_example1.elf. Note that this file is not used for programming the chip.

   **Note:** If the output file name for the linker is not specified in the Makefile (i.e. option –o is missing), a.out will be created instead of an .elf file. They are the same and these steps can be applied to a.out as well.

2. Flash the .bin file into the chip. Refer to section 5.1.2 above.

3. Open a command line window, run:

   "python <Installation directory>\Toolchain\utilities\gdb_bridge.py live"

   **Note:** An alternative is to double click on the shortcut “GDB Bridge” created after the installation.

   The correct response should be:

   ![Figure 53 FT9XX Debugging Status](image)

   **Note 1:** If there is an error message about permission being denied, the command line window may need to be opened with administrator rights by right-clicking and selecting ‘Run as administrator’.

   **Note 2:** It is also possible to run the GDB Bridge using the shortcut created after the installation.

   **Note 3:** If the path to gdb_bridge.py contains spaces, enclose it with double quotes (““”).

4. Open another command line window, go to the folder that includes the .elf file, run “ft32-elf-gdb <.elf file>”, for example “ft32-elf-gdb gpio_example1.elf”.

5. After ft32-elf-gdb starts, type in “target remote localhost:9998” to establish a connection to the MCU.

6. Use standard GDB commands to debug the program. Note that the command to start execution should be “continue”, not “run”.

6.2 Installing Eclipse and the FT9XX plugin manually

When running the installer, it is possible to choose not to install Eclipse as part of the installation. This might be useful if the user have already installed Eclipse for other purposes. This section details how to set it up for use with the FT9XX.

6.2.1 Eclipse Installation

1. Go to Eclipse website, download “Eclipse IDE for C/C++ Developers”. At the time of this writing, Eclipse Mars is the latest release and is the recommended version.
2. When Eclipse is run for the first time, it will ask for the workspace location.

**Figure 55 Eclipse Workspace Location**

A workspace is a directory on the hard drive where Eclipse stores the projects defined to it. More specifically, a workspace is a logical collection of projects. When you specify this directory name to Eclipse, Eclipse will create some files within this directory to manage the projects. The projects controlled by this workspace may or may not reside in this directory. Specify a directory name and click OK.

**Note:** To run Eclipse, it is required to download and install the Java Run Time Environment (JRE) or Java Developer Kit (JDK). Eclipse should display a warning if this is not installed. Oracle provides these tools for free.
6.2.2 FT9XX Eclipse Plugin Installation

To assist with completing the configuration of Eclipse for FT9XX coding an extra plug-in is provided as part of the download. To install the plug-in, the following steps are required:

1. From the Eclipse toolbar select Help -> Install New Software which will pop up the window as below.

![Eclipse Plugin Setup Wizard](image)

2. Select the ADD button, and browse to the LOCAL location of the folder ‘com.ftdichip.ft90x’ which can be found in “Toolchain\eclipse plugins” in the toolchain installation directory.

3. Press “SELECT ALL” followed by NEXT to install the plugin.

4. Close the window when complete.

6.3 Common project settings in Eclipse

6.3.1 Include paths

Eclipse uses its built-in indexer to resolve dependencies between files. In order for the indexer to work correctly, paths that contain the header files in the project need to be added as follows:

1. Right-click on the project and select Properties
2. In the Properties window, select C/C++ General > Paths and Symbols

3. Under the Includes tab, choose “GNU C” under Languages, then click “Add...” on the right side of the window

4. In the “Add directory path” window, specify the path to the folder that contains the header files. If the same path is used for some C++ files, check the box “Add to all languages”, then click OK.
6.3.2 Toolchain settings

The FT9XX toolchain supports most GNU toolchain options. To specify an option that is not included by default, for example to create a map file, do it as follows:

1. Right click on the project and select Properties
2. In the Properties window, select C/C++ Build > Settings. The toolchain settings can be adjusted in the Settings window.

![Figure 59 Eclipse Add Directory Path](image)

**Note:** The paths should be added one at a time. The use of semicolon is not supported.

![Figure 60 Eclipse Toolchain Settings](image)
6.3.2.1 FT32B options (only for FT93x)

The FT93X target has two special compiler options –mft32b and –mcompress. –mft32b enables the FT32B instruction set and –mcompress enables code compression, which can typically result in a code size reduction of about 20 – 30%. These options can be updated in the toolchain settings as shown in Figure 61. Note that this option is only available in the FT930_* configurations.

![Figure 61 FT32B options (for FT93X)](image)

6.3.3 C/C++ Indexer Settings

For Eclipse syntax highlighting to work correctly as you switch configurations, the Indexer has to be configured to work with the active build configuration as shown in Figure 64. This is a workspace specific setting and can be accessed in Eclipse via Window | Preferences | C/C++ | Indexer.

![Figure 62 Configure Eclipse indexer to use the active build configuration](image)
6.3.4 C++ Compilation

C++ compilation is supported; however there is no support for the standard C++ libraries or STL libraries. Some language features are also not supported, e.g. new and delete operators. Include files of FT9xx libraries are C++ safe when wrapped in extern "C" {} scope in order to be excluded from the C++ namespace. Constructors and destructors of singleton classes are not supported and will require the programmer to supply his own crt0.s file. Support for global and static objects shall be added in a future version.

6.4 FreeRTOS Kernel-Aware Debugging

FT9XX supports the popular real-time operating system, FreeRTOS. FreeRTOS allows creation of tasks, queues, events, semaphores, mutex and timers for inter-task communication and synchronization. To help debug these kernel objects, the FT9XX Eclipse plugin has been updated to provide custom views to list the FreeRTOS kernel objects.

These views can be opened via the menu Window → Show View → Others → FreeRTOS.

6.4.1 FreeRTOS views

The views will show the kernel objects that have been created at the point where the application is suspended, that is, when a breakpoint has been triggered.

These views allow the user to do the following:

- save the table of values to a CSV file
- clear the table
- refresh the table
- sort the table by the selected column

A tooltip description is also provided for each column of the views for a description of the column values.

For a demonstration of these views, refer to “FreeRTOS Example 4” application.
6.4.2 FreeRTOS Tasks view

This view displays the tasks that have been created. Each task has the following properties: task control block (TCB#), name, address, priority, state, start of stack, top of stack, total stack usage, event object and runtime percentage.

Note that deleted tasks will appear in this view with state 'DELETED'. These tasks will disappear in the view only when the idle task has freed up the corresponding memory allocation.

![FreeRTOS Tasks view](image1)

Some of these properties are dependent on a configuration in "FreeRTOSConfig.h". If these macros are set to 0, "Unknown" value will be displayed. Refer to the table below for these dependencies.

<table>
<thead>
<tr>
<th>FreeRTOS Task Property</th>
<th>FreeRTOS Configuration macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task TCB#</td>
<td>configUSE_TRACE_FACILITY</td>
</tr>
<tr>
<td>Task Runtime</td>
<td>configGENERATE_RUN_TIME_STATS</td>
</tr>
</tbody>
</table>

6.4.3 FreeRTOS Queues view

This view displays the queues that have been registered. Each registered queue has the following properties: name, address, type, maximum length, item size, current length, #tx waiting and #rx waiting.

![FreeRTOS Queues view](image2)
Some of these properties are dependent on a configuration in FreeRTOSConfig.h. If these macros are set to 0, "Unknown" value will be displayed. Refer to the table below for these dependencies.

**Table 2 FreeRTOS Queues macro dependencies**

<table>
<thead>
<tr>
<th>FreeRTOS Queue Property</th>
<th>FreeRTOS Configuration macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Type</td>
<td>configUSE_TRACE_FACILITY</td>
</tr>
</tbody>
</table>

**6.4.4 FreeRTOS Timers view**

This view displays the timers that have been created. Each timer has the following properties: id, name, address, period, reload and callback function.

![Figure 65 FreeRTOS Timers view](image)

**6.4.5 FreeRTOS Heap Usage view**

This view displays the current heap allocations or usage. Each heap block has the following properties: type, base address, end address, total size and available size.

![Figure 66 FreeRTOS Heap Usage view](image)

The heap type value can be changed via the preprocessor symbol FT32_PORT_HEAP.
6.4.6 FreeRTOS Configuration view

This view allows the user to configure the FreeRTOS configuration macros located at FreeRTOSConfig.h. The user can modify FreeRTOSConfig.h directly or via this view.

The values in the table appear when the application is not running. Once the application is running, it is not possible to update the values.

To change a value, just select the value and change. Any changes in it will be highlighted in yellow. To save the changes, click on the save button at the top right corner of this view. This will overwrite the existing configuration file FreeRTOSConfig.h. A backup of the original file will be saved in the same directory.

When overwriting the configuration file, this feature preserves the code comments and whitespaces and also supports macros, multi-line macro definitions and macros enclosed in #ifdef ifndef.

6.4.7 Built-in Debug view

The built-in Debug view has been customized to show the list of FreeRTOS tasks that have been created. Each task indicates its TCB number, name and current state.

Currently, only the running task displays a function call stack. The top frame of the running task call stack will always be selected automatically. In case no frame is selected, user must select a frame before doing a debug command (continue, step into/over/return, etc).

Note that deleted tasks will appear in this view with state ‘DELETED’. These tasks will disappear in the view only when the idle task has freed up the corresponding memory allocation.
The FreeRTOS configuration macro configUSE_TRACE_FACILITY must be set to 1 in FreeRTOSConfig.h for this view to display all the tasks created. Otherwise, only 1 task will be displayed.
7 Troubleshooting

This section documents the problems you may encounter when using the FT9XX Toolchain.

7.1 Makefile error

If using a makefile to build an application, some makefile errors may be reported, for example:

![Makefile Error](image)

This is usually because some existing toolchain on the system may be using its own “make” utility which is also referred to in the PATH variable. The FT9XX examples need to be built by the GnuWin32 “make” utility, which can be installed during the toolchain installation. To solve this problem, adjust the PATH variable so that the correct “make” utility is called by the toolchain. Note that it may be necessary to adjust PATH again for the other toolchain. Type “where make” in a command prompt to find out which “make” utilities are present on the system.

![“make” Locations](image)
7.2 Programming does not work for either RUN or DEBUG

1. Ensure that manually-executed FT900 Programmer and GDB Bridge are not running. The new Eclipse plugin automatically runs the FT900 Programmer and GDB Bridge so user does not need to manually run these applications anymore. These applications will fail to run if there are other instances running because they both share the one-wire programming interface.

2. Ensure that Eclipse-executed GDB Bridge is not running. This Eclipse-executed GDB Bridge is launched without a visible window. Check Task Manager and kill the process.

![Figure 71 Eclipse-executed invisible GDB Bridge](image)

7.3 Build errors due to anti-virus software protection

When building in Eclipse, the build process will sometimes stop or get delayed with the messages like:

```
make: *** Access is denied. Stop.
```

```
make: *** Waiting for unfinished jobs....
```

Or
This is a known issue with some antivirus software, such as AVG. Workarounds include (depending on the antivirus software):
- Temporarily disable the antivirus software.
- Add exceptions to every "exe/bat" under the FT9xx Toolchain folder.

In case of taking exceptions in AVG Business, refer to the following link –

7.4 Building of header files from Eclipse Virtual Store

If the FT9xx toolchain is installed in the system directory path 'C:\Program Files (x86)\Bridgetek\FT9xx Toolchain', when Eclipse IDE of FT9xx Toolchain is not run in administrator mode, and when the library header files also under system directory path 'C:\Program Files (x86)\Bridgetek\FT9xx Toolchain' are modified by the user in eclipse editor, the files get actually saved in C:\Users\%Username%\AppData\Local\VirtualStore\Program Files (x86)\Bridgetek\FT9xx Toolchain\. This is because Eclipse sees the files it changed in C:\Program Files (x86)\Bridgetek\FT9xx Toolchain, but other applications like the Windows Explorer will see only the unchanged files.

If the above behavior adds confusion to the building of eclipse projects, then,

The User can choose to install the toolchain in a non-system path so that the FT9xx library header files are also installed in non-system path

Or

Run the eclipse IDE with elevated privilege mode by selecting 'Run as administrator' while opening the eclipse.

Or

Clear the Virtual Store path (C:\Users\%Username%\AppData\Local\VirtualStore\Program Files (x86)\Bridgetek\FT9xx Toolchain\) for any stale headers.

More info about Virtual store can be found in the below link:

8 Contact Information

Head Quarters – Singapore
Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030
Tel: +65 6547 4827
Fax: +65 6841 6071
E-mail (Sales): sales.apac@brtchip.com
E-mail (Support): support.apac@brtchip.com

Branch Office – Taipei, Taiwan
Bridgetek Pte Ltd, Taiwan Branch
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District
Taipei 114
Taiwan, R.O.C.
Tel: +886 (2) 8797 1330
Fax: +886 (2) 8751 9737
E-mail (Sales): sales.apac@brtchip.com
E-mail (Support): support.apac@brtchip.com

Branch Office – Glasgow, United Kingdom
Bridgetek Pte. Ltd.
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): sales.emea@brtchip.com
E-mail (Support): support.emea@brtchip.com

Branch Office – Vietnam
Bridgetek VietNam Company Limited
Lutaco Tower Building, 5th Floor, 173A Nguyen Van Troi,
Ward 11, Phu Nhuan District,
Ho Chi Minh City, Vietnam
Tel : 08 38453222
Fax : 08 38455222
E-mail (Sales): sales.apac@brtchip.com
E-mail (Support): support.apac@brtchip.com

Web Site
http://brtchip.com/

Distributor and Sales Representatives
Please visit the Sales Network page of the Bridgetek Web site 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 (BRITChip) 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 information is subject to customer confirmation, and Bridgetek disclaims all liability for system designs and for any applications 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, 178 Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number: 201542387H.
Appendix A – References

Document References
http://brtchip.com/m-ft9/
AN_360 FT9XX Example Applications
TN_160 Eclipse Projects

Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD</td>
<td>Command-line interface</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic-link Library</td>
</tr>
<tr>
<td>DLOG</td>
<td>Data Log (Project)</td>
</tr>
<tr>
<td>GAS</td>
<td>GNU Assembler</td>
</tr>
<tr>
<td>GCC</td>
<td>GNU Compiler Collection</td>
</tr>
<tr>
<td>GDB</td>
<td>GNU Project Debugger</td>
</tr>
<tr>
<td>GNU</td>
<td>GNU (Gnu’s Not Unix) Operating System</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller Unit</td>
</tr>
<tr>
<td>PATH</td>
<td>PATH Environment Variable</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
</tbody>
</table>
Appendix B – List of Tables & Figures

List of Tables
NA

List of Figures
Figure 1 Toolchain Setup Wizard Dialog box ................................................................. 6
Figure 2 License Agreement Dialog box ........................................................................ 6
Figure 3 Revision and Release Information Dialog box .................................................. 7
Figure 4 Components Dialog box .................................................................................. 7
Figure 5 FT9XX Toolchain Install Location Dialog box ..................................................... 8
Figure 6 FT9XX Toolchain-Examples & Documents Install Location Dialog box ............... 8
Figure 7 FT9XX Toolchain - Installation Progress Window .............................................. 9
Figure 8 Close any running Java Applications before launching JRE Installer .................. 9
Figure 9 Java Setup Window ............................................................................................. 10
Figure 10 Java Setup Progress Window ........................................................................... 10
Figure 11 Python Setup Dialog box ................................................................................ 11
Figure 12 Destination Directory Selection Dialog box for Python Setup ......................... 11
Figure 13 Python Features Customization Dialog box ..................................................... 12
Figure 14 Python Installation Progress Dialog box .......................................................... 12
Figure 15 Python Installation Completion Dialog box ..................................................... 13
Figure 16 FT9XX Toolchain - Installation Progress Window .......................................... 13
Figure 17 FT9XX Toolchain Setup Completion Dialog box .............................................. 14
Figure 18 MSI Installer is busy ......................................................................................... 15
Figure 19 Eclipse for FT9XX Icon ................................................................................... 16
Figure 20 Eclipse Workspace Selection .......................................................................... 16
Figure 21 Eclipse Workspace Update ............................................................................. 17
Figure 22 C Project Wizard .............................................................................................. 17
Figure 23 Project Wizard - Build Configurations Selection .............................................. 18
Figure 24 C Project Wizard - Toolchain Details ............................................................... 18
Figure 25 New empty project structure ........................................................................... 18
Figure 26 Build Configuration ........................................................................................ 20
Figure 27 Build Configuration ......................................................................................... 20
Figure 28 Building the Project ........................................................................................ 20
Figure 29 Build Status ..................................................................................................... 21
Figure 30 List of Files after building (if all 4 configurations were selected) ....................... 21
Figure 31 Run menus - Run Configurations and Run Last Launched ............................... 22
Figure 32 Run Configuration window ............................................................................. 23
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Console window after selecting launch configuration and clicking Run</td>
</tr>
<tr>
<td>34</td>
<td>FT9XX Programming Utility Icon</td>
</tr>
<tr>
<td>35</td>
<td>Bridgetek Utilities Menu</td>
</tr>
<tr>
<td>36</td>
<td>FT9XX Programmer - Work with One-Wire</td>
</tr>
<tr>
<td>37</td>
<td>FT9XX Programmer - Device Selection</td>
</tr>
<tr>
<td>38</td>
<td>FT9XX Programmer – Flash and PM Screen</td>
</tr>
<tr>
<td>39</td>
<td>FT9XX Programmer in Eclipse</td>
</tr>
<tr>
<td>40</td>
<td>Hello World</td>
</tr>
<tr>
<td>41</td>
<td>FT9XX Examples</td>
</tr>
<tr>
<td>42</td>
<td>Build Configuration</td>
</tr>
<tr>
<td>43</td>
<td>Debug Configuration window</td>
</tr>
<tr>
<td>44</td>
<td>Debug Configurations and Debug Last Launched</td>
</tr>
<tr>
<td>45</td>
<td>Console window after selecting launch configuration and clicking Debug</td>
</tr>
<tr>
<td>46</td>
<td>Eclipse Debug Environment</td>
</tr>
<tr>
<td>47</td>
<td>Eclipse Missing Source File</td>
</tr>
<tr>
<td>48</td>
<td>Debug flags</td>
</tr>
<tr>
<td>49</td>
<td>Og compiler optimization option</td>
</tr>
<tr>
<td>50</td>
<td>Bridgetek Project Types</td>
</tr>
<tr>
<td>51</td>
<td>D2XX Project Template</td>
</tr>
<tr>
<td>52</td>
<td>Add-on Libraries</td>
</tr>
<tr>
<td>53</td>
<td>FT9XX Debugging Status</td>
</tr>
<tr>
<td>54</td>
<td>Eclipse Versions</td>
</tr>
<tr>
<td>55</td>
<td>Eclipse Workspace Location</td>
</tr>
<tr>
<td>56</td>
<td>Eclipse Plugin Setup Wizard</td>
</tr>
<tr>
<td>57</td>
<td>Eclipse Project Properties</td>
</tr>
<tr>
<td>58</td>
<td>Eclipse Paths and Symbols</td>
</tr>
<tr>
<td>59</td>
<td>Eclipse Add Directory Path</td>
</tr>
<tr>
<td>60</td>
<td>Eclipse Toolchain Settings</td>
</tr>
<tr>
<td>61</td>
<td>FT32B options (for FT93X)</td>
</tr>
<tr>
<td>62</td>
<td>Configure Eclipse indexer to use the active build configuration</td>
</tr>
<tr>
<td>63</td>
<td>FreeRTOS Tasks view</td>
</tr>
<tr>
<td>64</td>
<td>FreeRTOS Queues view</td>
</tr>
<tr>
<td>65</td>
<td>FreeRTOS Timers view</td>
</tr>
<tr>
<td>66</td>
<td>FreeRTOS Heap Usage view</td>
</tr>
<tr>
<td>67</td>
<td>FreeRTOS Configuration view</td>
</tr>
<tr>
<td>68</td>
<td>Built-in Debug view</td>
</tr>
<tr>
<td>69</td>
<td>Makefile Error</td>
</tr>
<tr>
<td>70</td>
<td>&quot;make&quot; Locations</td>
</tr>
<tr>
<td>71</td>
<td>Eclipse-executed invisible GDB Bridge</td>
</tr>
</tbody>
</table>
Figure 72 AVG warning during building in FT9xx toolchain

Figure 73 Adding exception in AVG for FT9xx toolchain executables
# Appendix C – Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Changes</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial release</td>
<td>2014-05-02</td>
</tr>
<tr>
<td>1.01</td>
<td>Expanded screenshots of Installation Wizard in Section 2</td>
<td>2015-08-21</td>
</tr>
<tr>
<td>1.02</td>
<td>Updated Version for Toolchain 2.1.0</td>
<td>2016-02-22</td>
</tr>
<tr>
<td>1.03</td>
<td>Added section 2.1.1 to document the handling of another instance of MSI installer running in the background while installing JRE (results in JRE install error 1618)</td>
<td>2016-09-19</td>
</tr>
<tr>
<td></td>
<td>Debugger related information moved to section 4.4. from Troubleshooting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updated screenshots for programmer</td>
<td></td>
</tr>
<tr>
<td>1.04</td>
<td>Updated release Migration of the product from FTDI to Bridgetek name – logo changed, copyright changed, contact information changed</td>
<td>2017-03-08</td>
</tr>
<tr>
<td>1.05</td>
<td>The toolchain references has been updated from FT90X to FT9XX</td>
<td>2017-03-23</td>
</tr>
<tr>
<td>1.06</td>
<td>Updated the document for ‘Seamless Debug’ eclipse plugin; Section1 and Section 3.3 for Seamless Debug; Added section 6.3.4 on C++ compilation</td>
<td>2018-01-22</td>
</tr>
<tr>
<td>1.07</td>
<td>Updated the document for Section 5: Bridgetek Projects, Section 6.4 FreeRTOS Kernel aware debugging, Section 7.3: Build errors due to anti-virus software protection.</td>
<td>2018-11-14</td>
</tr>
</tbody>
</table>