Application Note

AN_325

FT9XX Toolchain Installation and Start Guide

Version 2.0

Issue Date: 16-10-2023

This guide documents the installing the v2.7.6 tools required for building, programming and debugging the FT9XX series of devices from 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 Bridgetek harmless from any and all damages, claims, suits or expense resulting from such use.

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

Table of Contents

Copyright © Bridgetek Pte Ltd

Copyright © Bridgetek Pte Ltd

1 Introduction

The free FT9XX toolchain v2.7.6 contains a port of the popular GNU toolchain which includes the following components:

- GCC compiler based on gcc release 11.2.0.
- GNU Binary Utilities (binutils) based on release 2.37.50. This includes tools such as assembler, linker and object format tools.
- GDB-based debugger based on gdb 11.2.

Supporting tools for programming and debugging are also provided.

A customized version of the Eclipse IDE with CDT (for C/C++ support) is included. This is based on the 2023-09 release.

The Eclipse IDE provided has a preinstalled plugin for supporting the FT9XX cross-compiler, programming and debugging tools. 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.

Example code for facilitating development using the FT9XX devices is provided. The main feature is an open-source hardware library that demonstrates accessing the peripherals on the device. This also includes a variety of example applications which demonstrate using the on-chip peripherals of the FT9XX range.

1.1 Changes in this version

1.1.1 Administrator privileges

In version v2.6.0 and earlier, Administrator privileges were required to install the toolchain on Windows. For some users, this required additional permissions to be allocated to the user.

This Windows version has been modified to not require Administrator privileges and therefore some of the installation directories and file locations differ from previous releases.

1.1.2 New utilities

The FT9xxProg.exe and "FT9xx Programming Utility" have been updated. The programming speed has been improved and icons and logos updated.

1.1.3 Debugging support

Versions previous to v2.7.0 used a "GDB bridge" utility to enable debugging by transferring debug commands from the `ft32-elf-gdb` utility to the FT9XX device *via* the one-wire debugger interface.

Since v2.7.0, the utility has been removed and is incorporated in the programming utility. A device can be programmed and then debugged in one step.

In this release there is a huge improvement in debugging speed and responsiveness over the v2.7.0 release.

1.1.4 Removed components

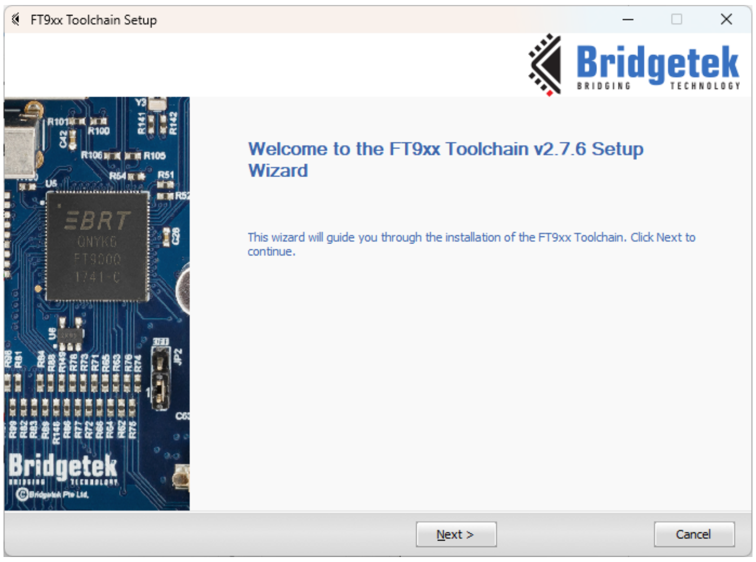
Since v2.7.0, none of the utilities provided require an installation of Java or Python. These components have been removed from installers.

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.



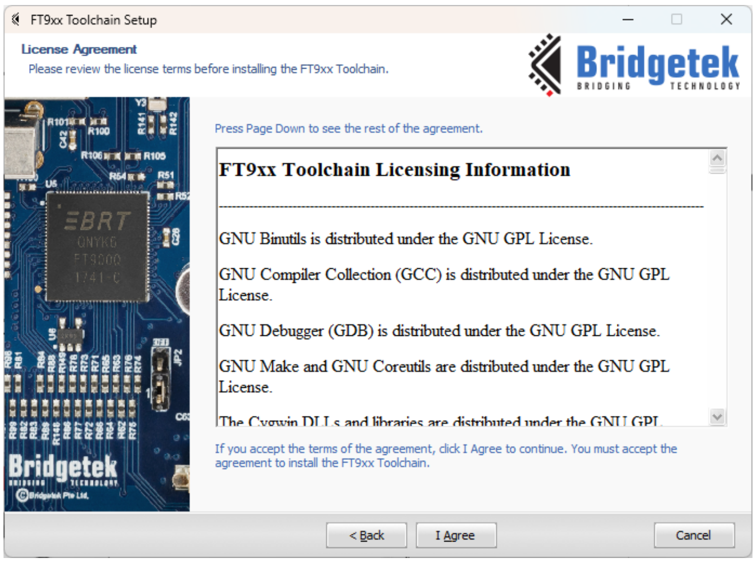
Welcome to the FT9xx Toolchain v2.7.6 Setup Wizard

This wizard will guide you through the installation of the FT9xx Toolchain. Click Next to continue.

Figure 1 Toolchain Setup Wizard Dialog box

2.1.1 Licensing Screen

The next screen displays the licensing information for all the Bridgetek components and, if the text of the agreement is scrolled down, all 3rd party components.



License Agreement
Please review the license terms before installing the FT9xx Toolchain.

Press Page Down to see the rest of the agreement.

FT9xx Toolchain Licensing Information

GNU Binutils is distributed under the GNU GPL License.

GNU Compiler Collection (GCC) is distributed under the GNU GPL License.

GNU Debugger (GDB) is distributed under the GNU GPL License.

GNU Make and GNU Coreutils are distributed under the GNU GPL License.

The Cygwin DLLs and libraries are distributed under the GNU GPL

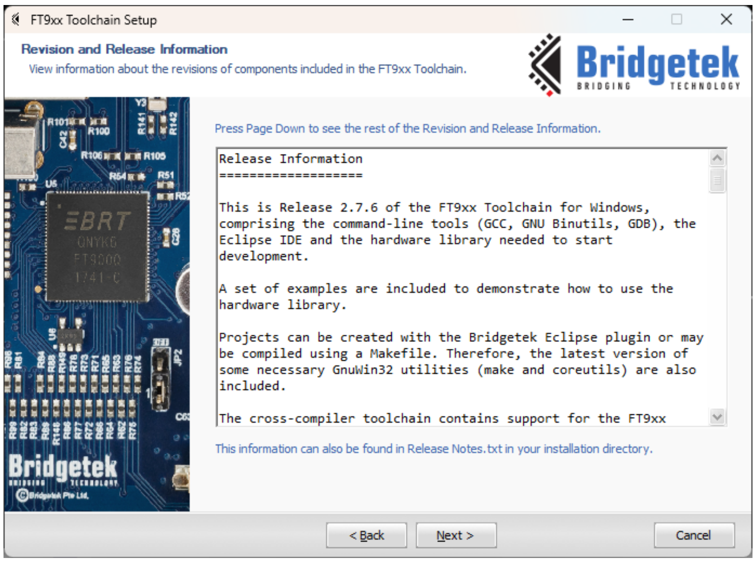
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install the FT9xx Toolchain.

Figure 2 License Agreement Dialog box

In the License Agreement dialog box, click "**I Agree"** button to proceed.

Copyright © Bridgetek Pte Ltd

2.1.2 Release Information


FT9xx Toolchain Setup
Revision and Release Information
View information about the revisions of components included in the FT9xx Toolchain.
Press Page Down to see the rest of the Revision and Release Information.

```
Release Information
===================

This is Release 2.7.6 of the FT9xx Toolchain for Windows,
comprising the command-line tools (GCC, GNU Binutils, GDB), the
Eclipse IDE and the hardware library needed to start
development.

A set of examples are included to demonstrate how to use the
hardware library.

Projects can be created with the Bridgetek Eclipse plugin or may
be compiled using a Makefile. Therefore, the latest version of
some necessary GnuWin32 utilities (make and coreutils) are also
included.

The cross-compiler toolchain contains support for the FT9xx
```

This information can also be found in Release Notes.txt in your installation directory.
< Back
Next >
Cancel

Figure 3 Revision and Release Information Dialog box

Go through the Revision and Release information and click **Next**.

2.1.3 Component Selection

The next screen allows you to select the components that are required.

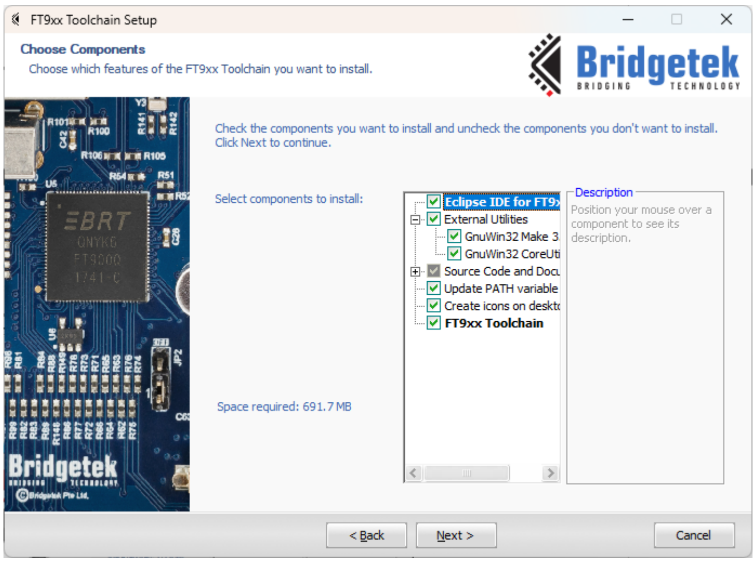

FT9xx Toolchain Setup
Choose Components
Choose which features of the FT9xx Toolchain you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Eclipse IDE for FT9
External Utilities
GnuWin32 Make 3
GnuWin32 CoreUti
Source Code and Docu
Update PATH variable
Create icons on deskt
FT9xx Toolchain
Description
Position your mouse over a component to see its description.
Space required: 691.7 MB
< Back
Next >
Cancel

Figure 4 Components Dialog box

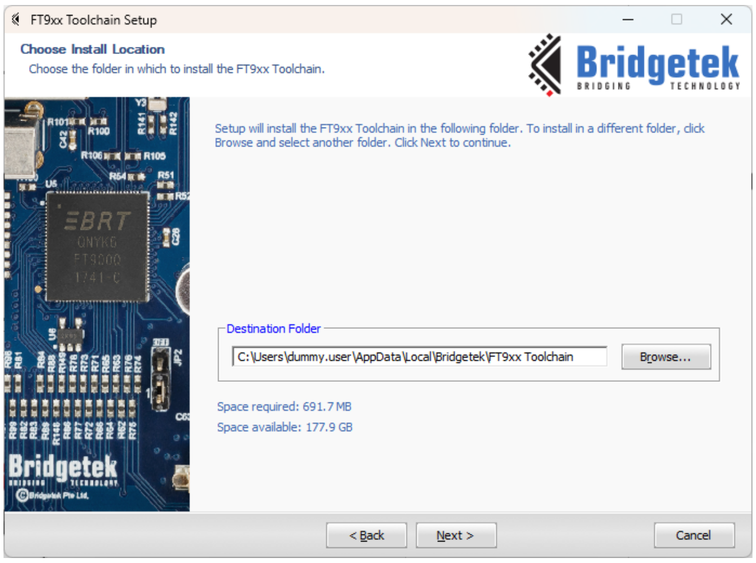
Select the Components and click **Next**. It is recommended to select **all** components.

The "Source Code and Documentation" section is intended to install files for direct access by users and places files in an easily accessible location for the user. The location for this is set later in Section 2.1.5. By default, only documentation is selected in this section. Other collateral can be installed by expanding the section and selecting the items on the page.

Other programs and code in the "Eclipse IDE for FT9xx", "External Utilities" and "FT9xx Toolchain" sections are generally not normally required to be accessible by users and are stored in an appropriate place set in Section 2.1.4.

Copyright © Bridgetek Pte Ltd

2.1.4 Toolchain Installation Location


FT9xx Toolchain Setup

Choose Install Location

Choose the folder in which to install the FT9xx Toolchain.

Setup will install the FT9xx Toolchain in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain

Space required: 691.7 MB

Space available: 177.9 GB

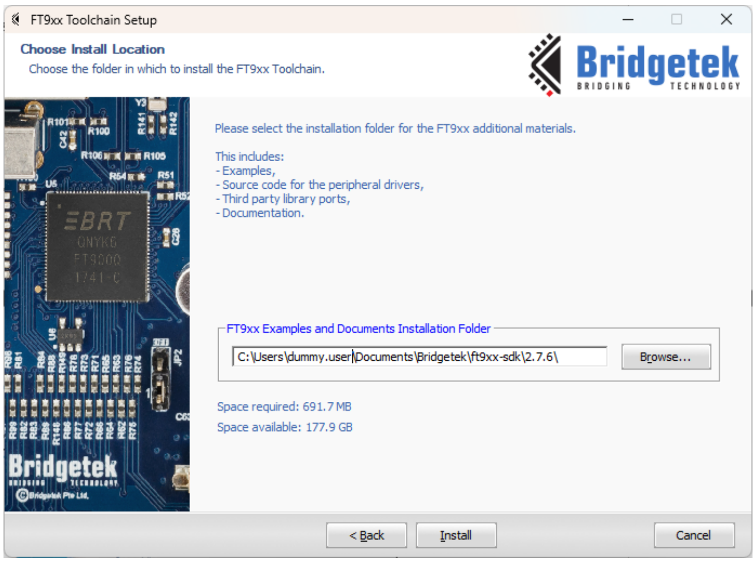
Figure 5 FT9XX Toolchain Install Location Dialog box

Click **Browse** and select a different file path for the FT9XX Toolchain installation. The "`AppData`" directory is normally used. This is a user-specific location and is located within the user's profile directory on Windows.

The "Eclipse IDE for FT9xx", "External Utilities" and "FT9xx Toolchain" components are installed in this location. Additionally, the hardware library and other library files are stored here to facilitate debugging, for importing libraries to existing projects, and importing example projects.

Continue installing in the specified folder by clicking **Next**.

2.1.5 Examples and Documents Location


FT9xx Toolchain Setup

Choose Install Location

Choose the folder in which to install the FT9xx Toolchain.

Please select the installation folder for the FT9xx additional materials.

This includes:
- Examples,
- Source code for the peripheral drivers,
- Third party library ports,
- Documentation.

FT9xx Examples and Documents Installation Folder: C:\Users\dummy.user\Documents\Bridgetek\ft9xx-sdk\2.7.6\

Space required: 691.7 MB

Space available: 177.9 GB

Figure 6 FT9XX Toolchain-Examples & Documents Install Location Dialog box

Depending on the selection in the Components page in Section 2.1.3, copies of the hardware library source code, third party libraries, examples and documentation are stored in the location set here. This normally appears in the "`Documents\Bridgetek\ft9xx-sdk\`*`version`*`\`" folder in Windows.

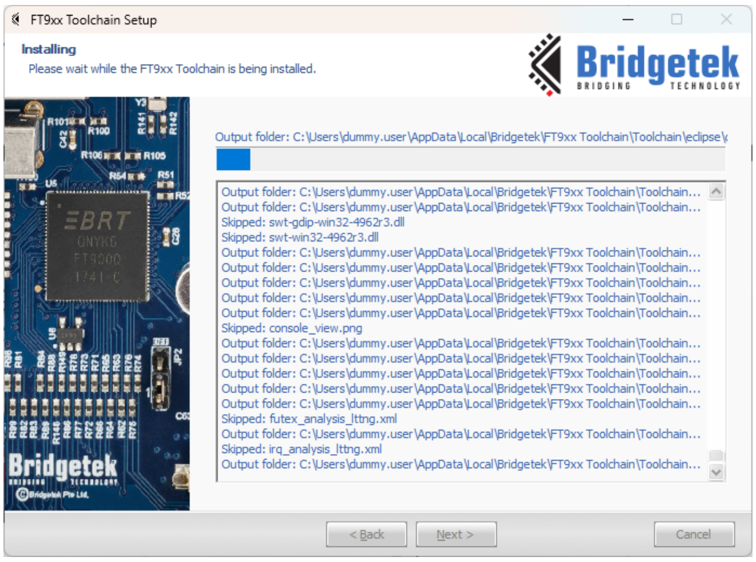
Copyright © Bridgetek Pte Ltd

Click **Browse** and select a different file path for installing FT9XX examples and documents.

Continue installing in the specified folder by clicking **Install**.

2.1.6 Installation Progress

After this, the FT9XX Toolchain installation progress bar is displayed.



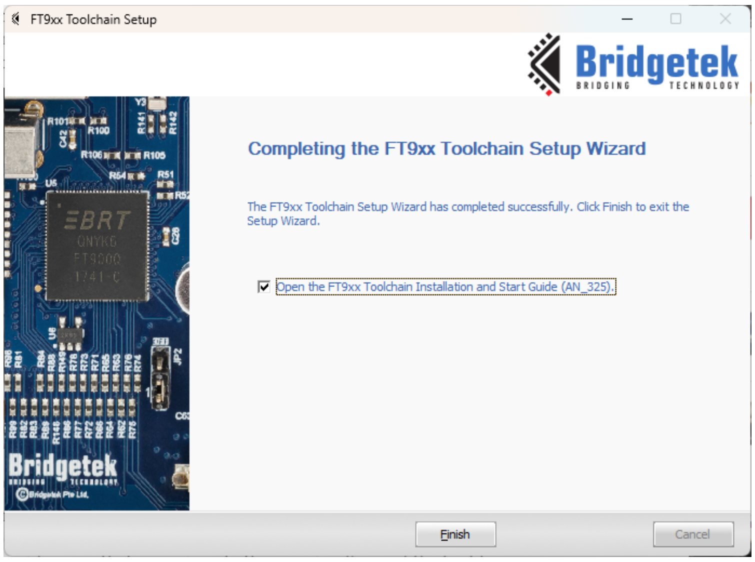
```
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain\eclipse\
```

```
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Skipped: swt-gdip-win32-4962r3.dll
Skipped: swt-win32-4962r3.dll
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Skipped: console_view.png
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Skipped: futex_analysis_lttng.xml
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
Skipped: irq_analysis_lttng.xml
Output folder: C:\Users\dummy.user\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain...
```

Figure 7 FT9XX Toolchain - Installation Progress Window

Select the **Open AN_325** checkbox to open the document in the system default PDF viewer after closing the Setup Wizard.

Click **Finish** to complete the FT9XX Toolchain Setup.



Completing the FT9xx Toolchain Setup Wizard

The FT9xx Toolchain Setup Wizard has completed successfully. Click Finish to exit the Setup Wizard.

[✓] Open the FT9xx Toolchain Installation and Start Guide (AN_325).

Figure 8 FT9XX Toolchain Setup Completion Dialog box

After the installation, the toolchain can be found in the installation directory from [Section 2.1.4](#). The default location is "`AppData\Bridgetek\FT9XX Toolchain`". This directory also contains the external utilities needed.

Copyright © Bridgetek Pte Ltd

The source code for previous versions of the toolchain is left untouched unless specifically removed by the uninstaller.

2.1.7 Installing a Java Runtime Environment Manually

The Toolchain installs Eclipse IDE. This includes a Java Runtime Environment (JRE) as part of the installation. Eclipse IDE will use this JRE rather than any pre-installed on the system.

2.2 Verifying the installation

1. Open a Command Prompt window by typing "`cmd`" in **Windows Start → Search** box.

2. Type "`ft32-elf-gcc --version`" in the command prompt. It should give the following message:

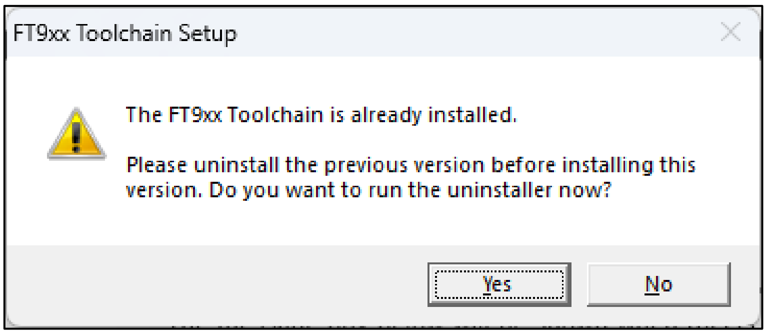
```
ft32-elf-gcc (GCC) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

If this message appears, then the toolchain has been successfully set up.

2.3 Updating the Installation

When updating from a previous version, the old version needs to be removed before proceeding with the new installation.

The message box below will be shown when this is required. Clicking **Yes** will launch the uninstaller program for the previous version. The installer for the new version will continue once the uninstaller has completed.


FT9xx Toolchain Setup

The FT9xx Toolchain is already installed.

Please uninstall the previous version before installing this version. Do you want to run the uninstaller now?

Figure 9 FT9XX Toolchain Update Dialog box

If the previous version requires Administrator privileges, then these will need to be provided in order to run the uninstaller.

3 Quick Start Guide

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.

Other methods of starting a project are discussed in Section 5 – Getting Started with Projects.

3.1 Creating a new project

Double-click **Eclipse for FT9XX** icon to launch the Eclipse IDE.

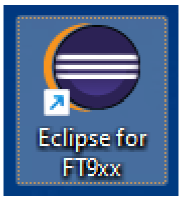
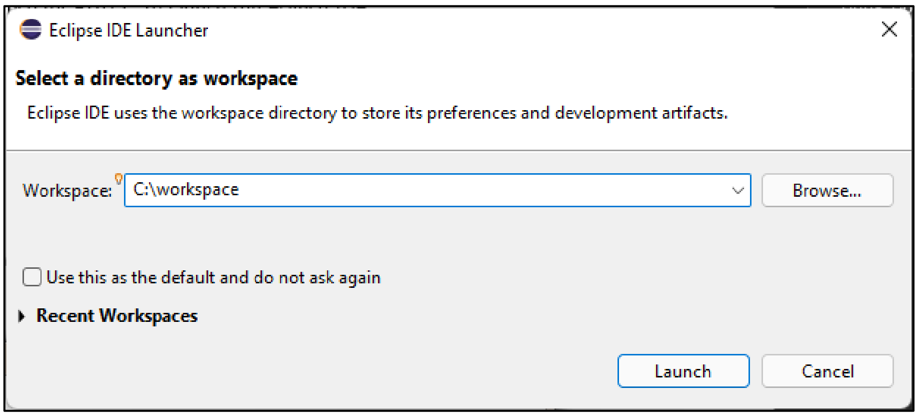


Figure 10 Eclipse for FT9XX Icon

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 IDE Launcher

Select a directory as workspace

Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\workspace

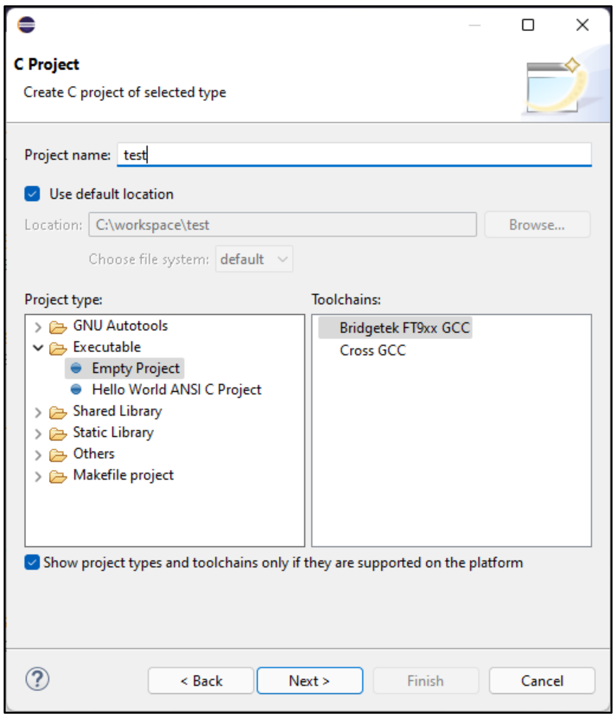
Figure 11 Eclipse Workspace Selection

Note: A 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 that you have chosen. If you want to change it, clear **Use default location** checkbox 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**.

Copyright © Bridgetek Pte Ltd



C Project

Create C project of selected type

Project name: test

☑ Use default location

Location: C:\workspace\test

Choose file system: default

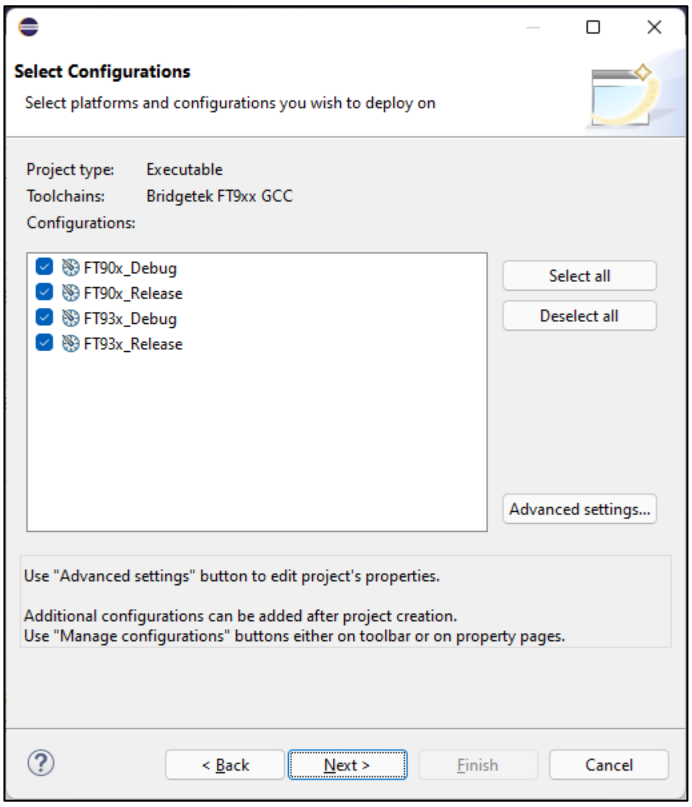
Project type:
- GNU Autotools
- Executable
  - Empty Project
  - Hello World ANSI C Project
- Shared Library
- Static Library
- Others
- Makefile project

Toolchains:
- Bridgetek FT9xx GCC
- Cross GCC

☑ Show project types and toolchains only if they are supported on the platform

Figure 12 C Project Wizard

In the next window, select `FT90x_*` configurations if your project is to target FT900 series of MCU or `FT93x_*` to target FT930 series, or both if you wish to target both series and click **Next**.



Select Configurations

Select platforms and configurations you wish to deploy on

Project type: Executable
Toolchains: Bridgetek FT9xx GCC
Configurations:

☑ FT90x_Debug
☑ FT90x_Release
☑ FT93x_Debug
☑ FT93x_Release

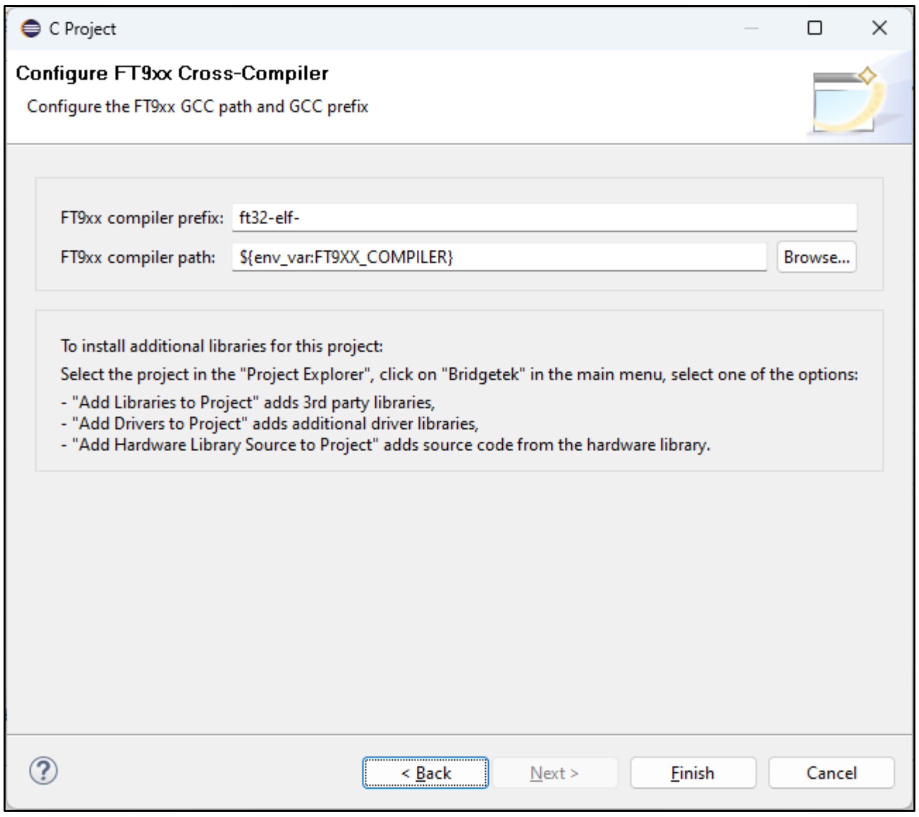
Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

Figure 13 Project Wizard – Build Configurations Selection

Copyright © Bridgetek Pte Ltd

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


C Project
Configure FT9xx Cross-Compiler
Configure the FT9xx GCC path and GCC prefix
< Back
Next >
Finish
Cancel
Browse...

FT9xx compiler prefix: ft32-elf-
FT9xx compiler path: ${env_var:FT9XX_COMPILER}

To install additional libraries for this project:
Select the project in the "Project Explorer", click on "Bridgetek" in the main menu, select one of the options:
- "Add Libraries to Project" adds 3rd party libraries,
- "Add Drivers to Project" adds additional driver libraries,
- "Add Hardware Library Source to Project" adds source code from the hardware library.

Figure 14 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.

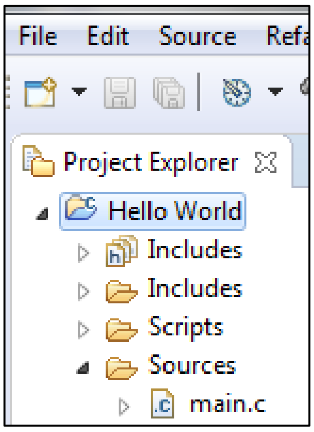


Figure 15 New empty project structure

The `main.c` contains some default content:

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

#if defined(__FT900__)
```

Product Page
Document Feedback
Copyright © Bridgetek Pte Ltd

```
#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,
     "  "--------------------------------------------------------------- \"\n"
     "  "Hello World! \"\n"
     "  "--------------------------------------------------------------- \"\n"
        );
    /* Now keep looping */
    while (1);

    return 0;
}
```

Now select the **Build Configuration** for either FT90X or FT93X via the **Project** menu.

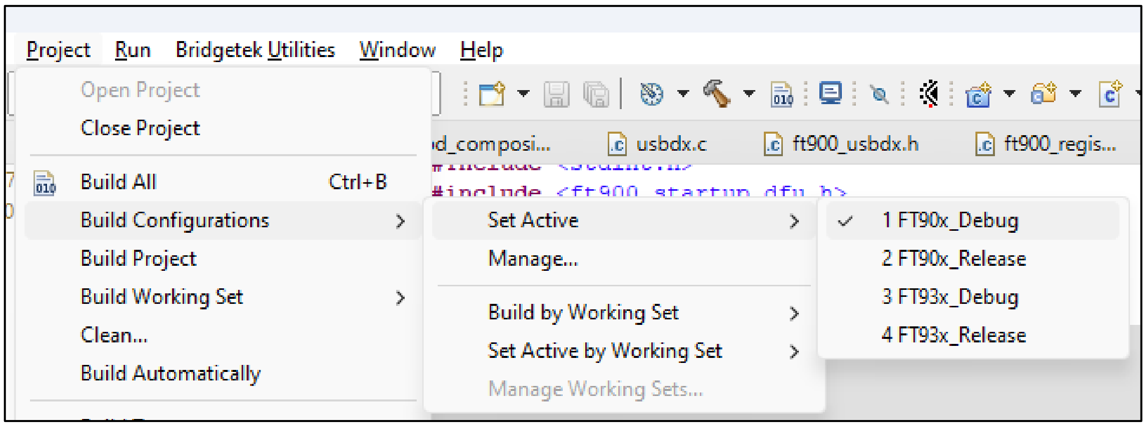


Figure 16 Build Configuration

This can also be done via the **Manage Configurations** icon on the toolbar.

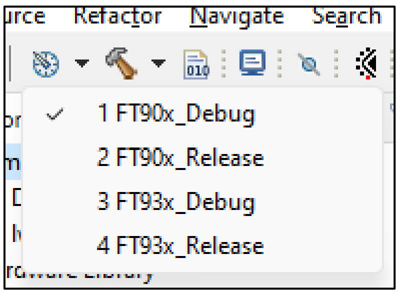


Figure 17 Build Configuration

Now the project can be built by clicking on the menu **Project → Build Project**. Note, alternative options are right-clicking the **Project → Build Project** and the icon.

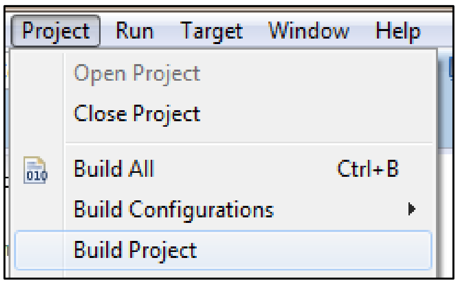
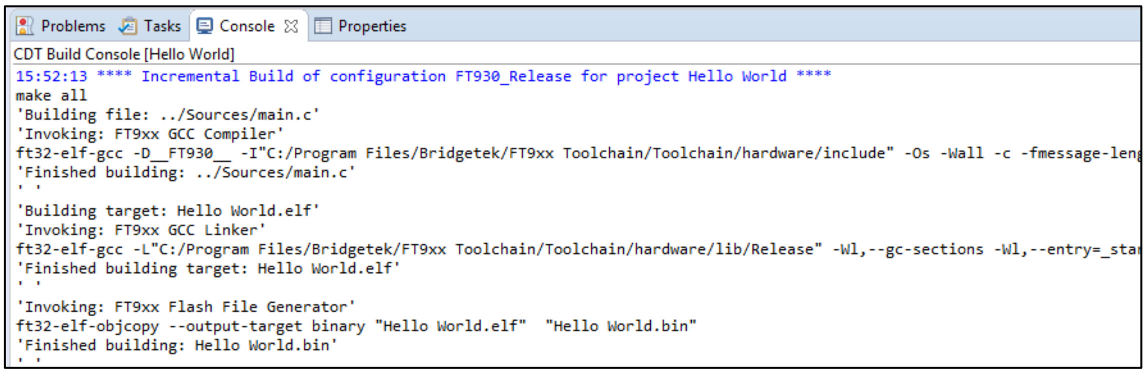


Figure 18 Building the Project

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.



```
CDT Build Console [Hello World]
15:52:13 **** Incremental Build of configuration FT930_Release for project Hello World ****
make all
'Building file: ../Sources/main.c'
'Invoking: FT9xx GCC Compiler'
ft32-elf-gcc -D__FT930__ -I"C:/Program Files/Bridgetek/FT9xx Toolchain/Toolchain/hardware/include" -Os -Wall -c -fmessage-leng
'Finished building: ../Sources/main.c'
' '
'Building target: Hello World.elf'
'Invoking: FT9xx GCC Linker'
ft32-elf-gcc -L"C:/Program Files/Bridgetek/FT9xx Toolchain/Toolchain/hardware/lib/Release" -Wl,--gc-sections -Wl,--entry=_sta
'Finished building target: Hello World.elf'
' '
'Invoking: FT9xx Flash File Generator'
ft32-elf-objcopy --output-target binary "Hello World.elf"  "Hello World.bin"
'Finished building: Hello World.bin'
' '
```

Figure 19 Build Status

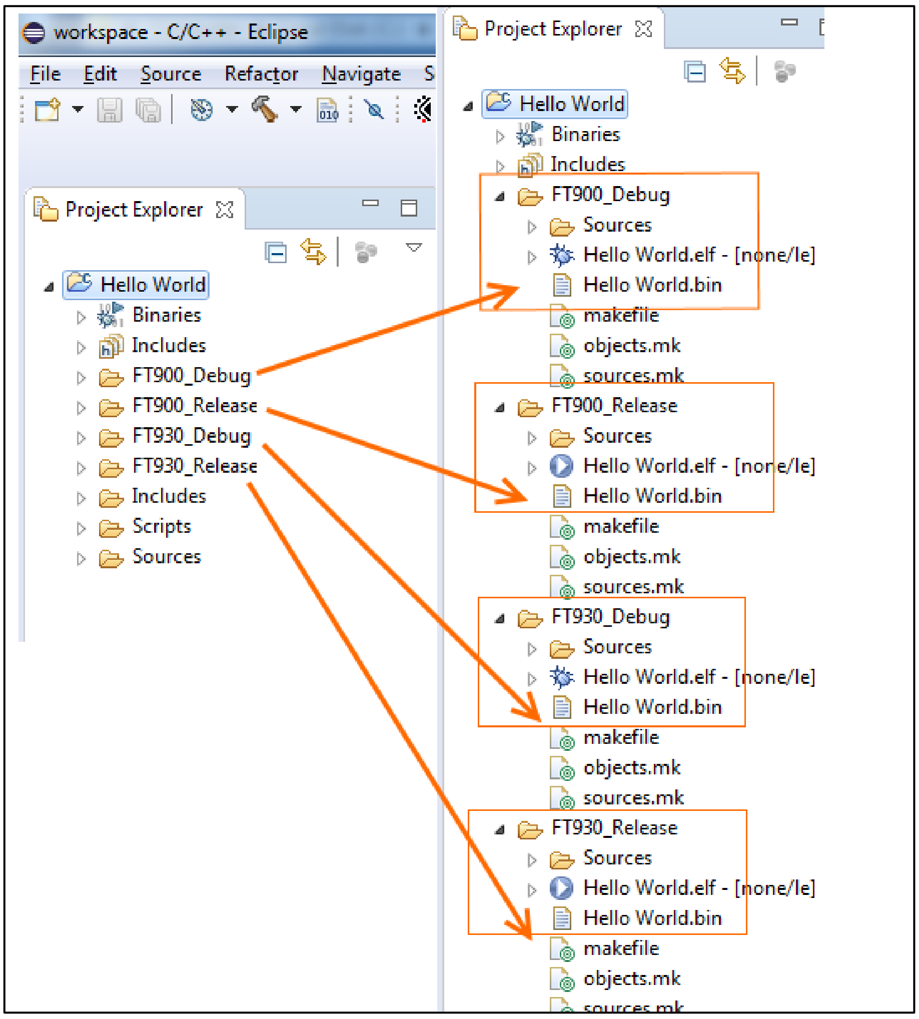


Figure 20 List of Files after building (if all 4 configurations were selected)

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 6.4.4 – C/C++ Indexer Settings 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 the 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 one-wire interface to communicate with the FT9XX device. 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 into the Eclipse IDE to allow a seamless debugging experience.

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

Copyright © Bridgetek Pte Ltd

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 a Run Configuration.
3. Click **Run** button to start the downloading. 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**.

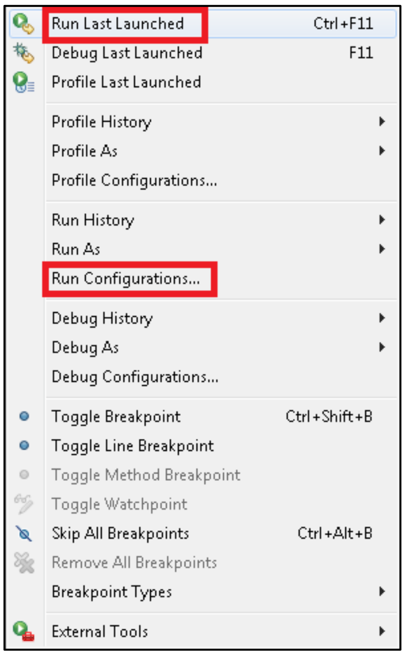
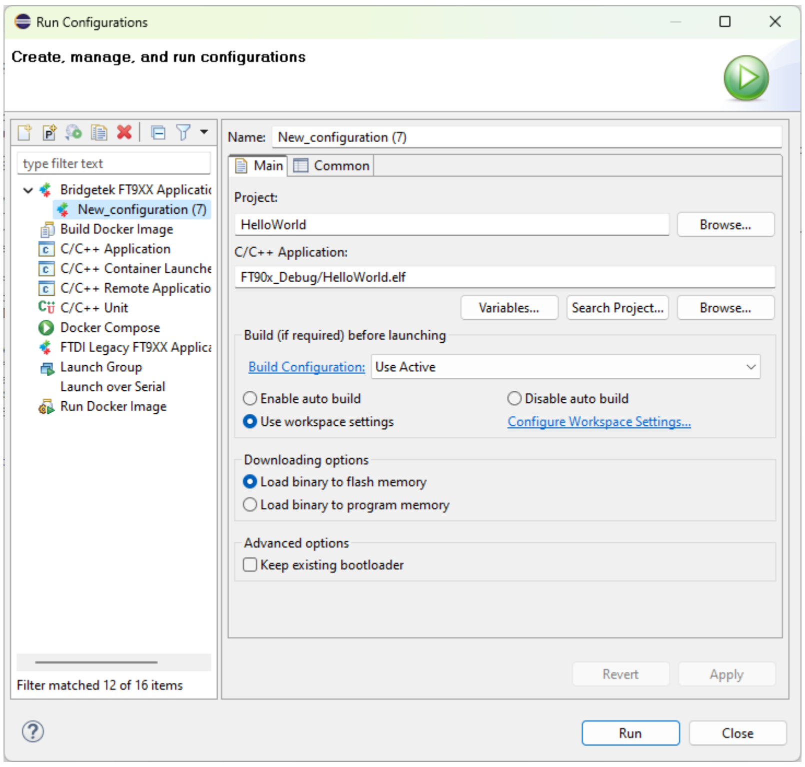
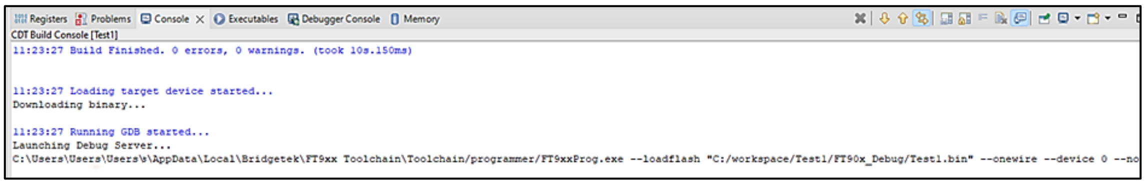


Figure 21 Run menus - Run Configurations and Run Last Launched


Run Configurations
Create, manage, and run configurations

Name: New_configuration (7)
Main | Common
Project: HelloWorld
C/C++ Application: FT90x_Debug/HelloWorld.elf
Build (if required) before launching
Build Configuration: Use Active
Enable auto build
[✓ Use workspace settings]
Disable auto build
Configure Workspace Settings...
Downloading options
[✓ Load binary to flash memory]
Load binary to program memory
Advanced options
☐ Keep existing bootloader

Figure 22 Run Configuration window



```
CDT Build Console [Test1]
11:23:27 Build Finished. 0 errors, 0 warnings. (took 10s.150ms)


11:23:27 Loading target device started...
Downloading binary...

11:23:27 Running GDB started...
Launching Debug Server...
C:\Users\Users\Users\s\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain/programmer/FT9xxProg.exe --loadflash "C:/workspace/Test1/FT90x_Debug/Test1.bin" --onewire --device 0 --no
```

Figure 23 Console window after selecting launch configuration and clicking Run

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 the **FT9XX Programming Utility** icon created on your desktop, if selected during the installation.

Figure 24 FT9XX Programming Utility Icon

Otherwise, it can be found in **Start** menu in **Bridgetek** section (under "**All Apps**" on Windows 11).

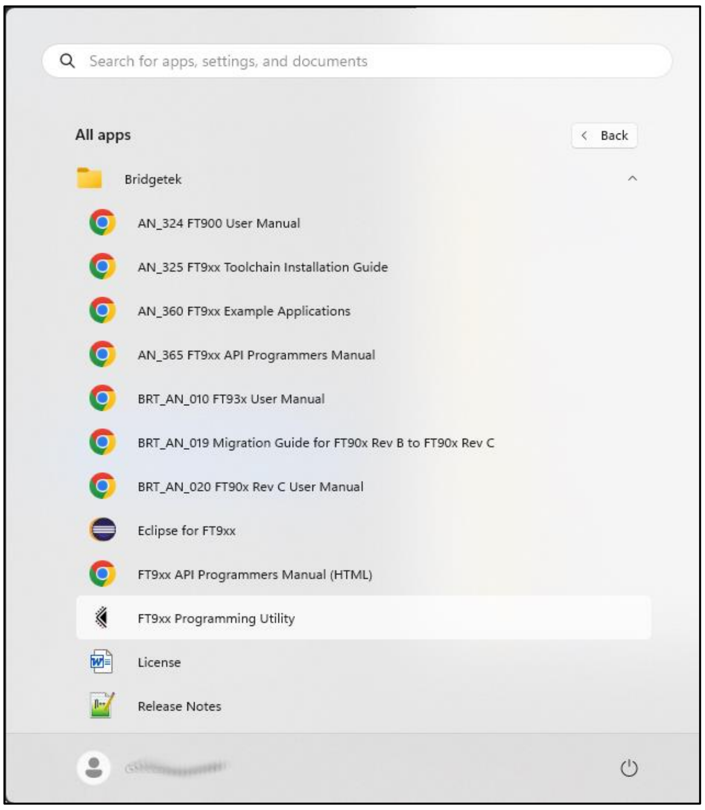
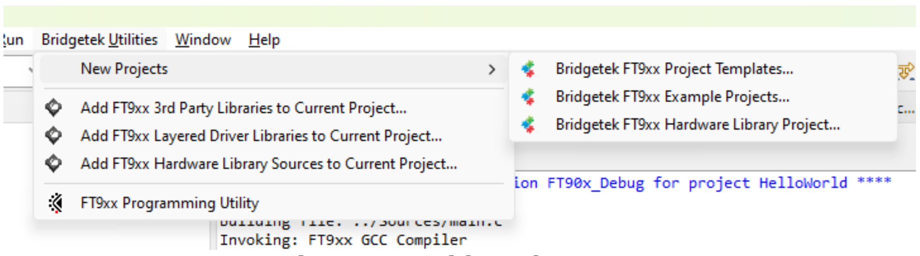

Search for apps, settings, and documents
All apps
Back
Bridgetek
AN_324 FT900 User Manual
AN_325 FT9xx Toolchain Installation Guide
AN_360 FT9xx Example Applications
AN_365 FT9xx API Programmers Manual
BRT_AN_010 FT93x User Manual
BRT_AN_019 Migration Guide for FT90x Rev B to FT90x Rev C
BRT_AN_020 FT90x Rev C User Manual
Eclipse for FT9xx
FT9xx API Programmers Manual (HTML)
FT9xx Programming Utility
License
Release Notes

Figure 25 Bridgetek Start Menu Section

You can also open the programming utility from Eclipse by selecting it in **Bridgetek** menu or the toolbar icon as highlighted in Figure 26.


Run
Bridgetek Utilities
Window
Help
New Projects
Add FT9xx 3rd Party Libraries to Current Project...
Add FT9xx Layered Driver Libraries to Current Project...
Add FT9xx Hardware Library Sources to Current Project...
FT9xx Programming Utility
Bridgetek FT9xx Project Templates...
Bridgetek FT9xx Example Projects...
Bridgetek FT9xx Hardware Library Project...

```
ion FT90x_Debug for project HelloWorld ****
Building file: ../Sources/main.c
Invoking: FT9xx GCC Compiler
```

Figure 26 Bridgetek Menu

The following screen will appear.

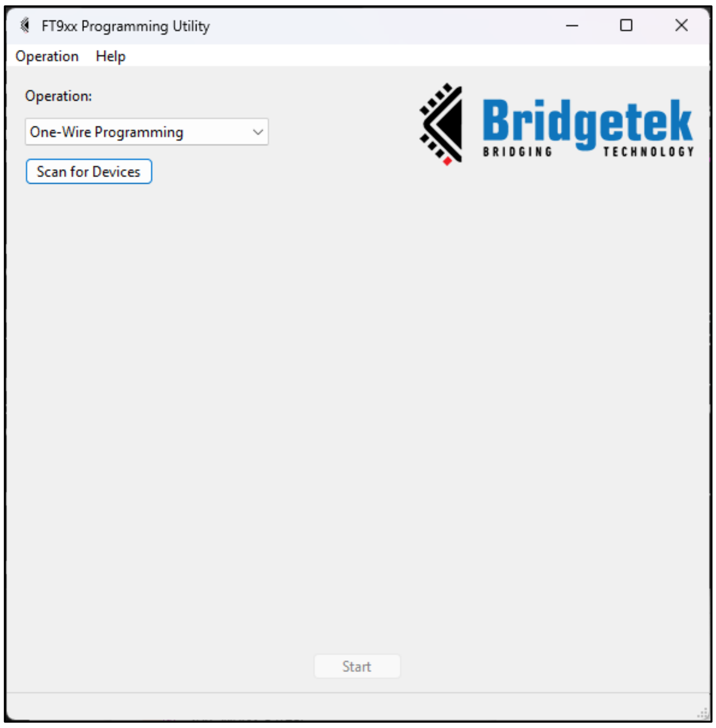
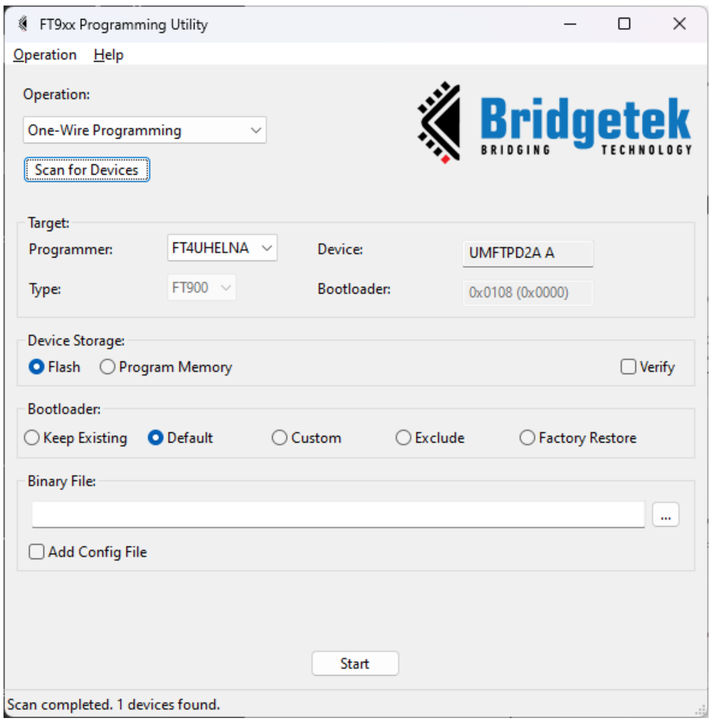


Figure 27 FT9XX Programmer - Work with One-Wire

Only the option "One-wire Programming" in the operation box is covered in this section. For information on the remaining options, please refer to the Programmer Utility help.

Select the "One-wire Programming" option and click **Scan for Devices**. The screen updates to show 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 in "**Target: Programmer:**". Select the device you wish to program.


FT9xx Programming Utility

Operation: One-Wire Programming

Target:
Programmer: FT4UHELNA
Type: FT900
Device: UMFTPD2A A
Bootloader: 0x0108 (0x0000)

Device Storage: [✓ Flash] Program Memory ☐ Verify

Bootloader: Keep Existing [✓ Default] Custom Exclude Factory Restore

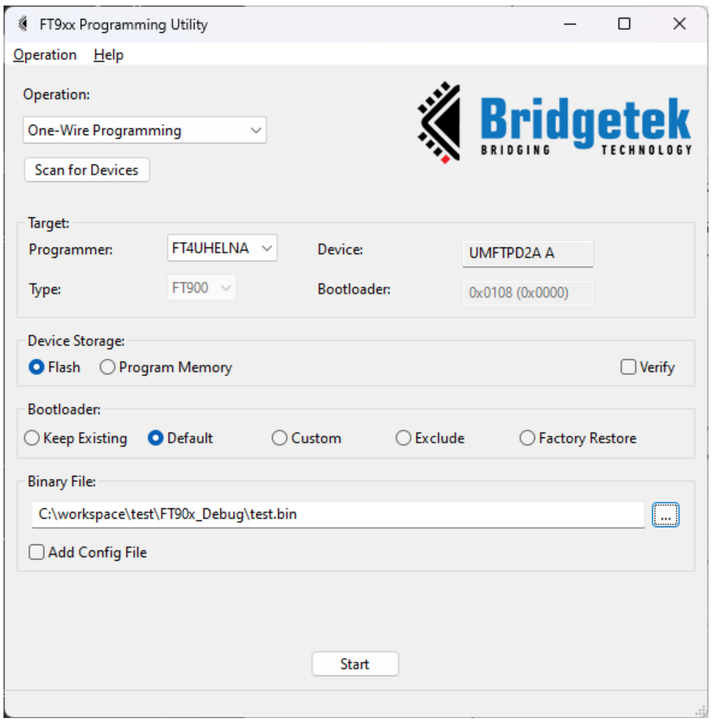
Binary File:

☐ Add Config File

Scan completed. 1 devices found.

Figure 28 FT9XX Programmer - Device Selection

Specify the location of the binary file and click **Start**.


FT9xx Programming Utility

Operation: One-Wire Programming

Target:
Programmer: FT4UHELNA
Type: FT900
Device: UMFTPD2A A
Bootloader: 0x0108 (0x0000)

Device Storage: [✓ Flash] Program Memory ☐ Verify

Bootloader: Keep Existing [✓ Default] Custom Exclude Factory Restore

Binary File: C:\workspace\test\FT90x_Debug\test.bin

☐ Add Config File

Figure 29 FT9XX Programmer – Binary File

More information on the utility can be found in the **About** tab → **Help**.

3.3.3 Command Line Version

Copyright © Bridgetek Pte Ltd

FT9xxProg.exe is available to run at a command prompt. Enter "`FT9xxProg.exe --help`" to see the options available. See Section 6.2.2 – "Programming a binary image into the chip" for more details.

This program can also be run within Eclipse as an External Tool. See Figure 30 for settings found in **Run → External Tools → External Tools Configurations**.

The external tool location:

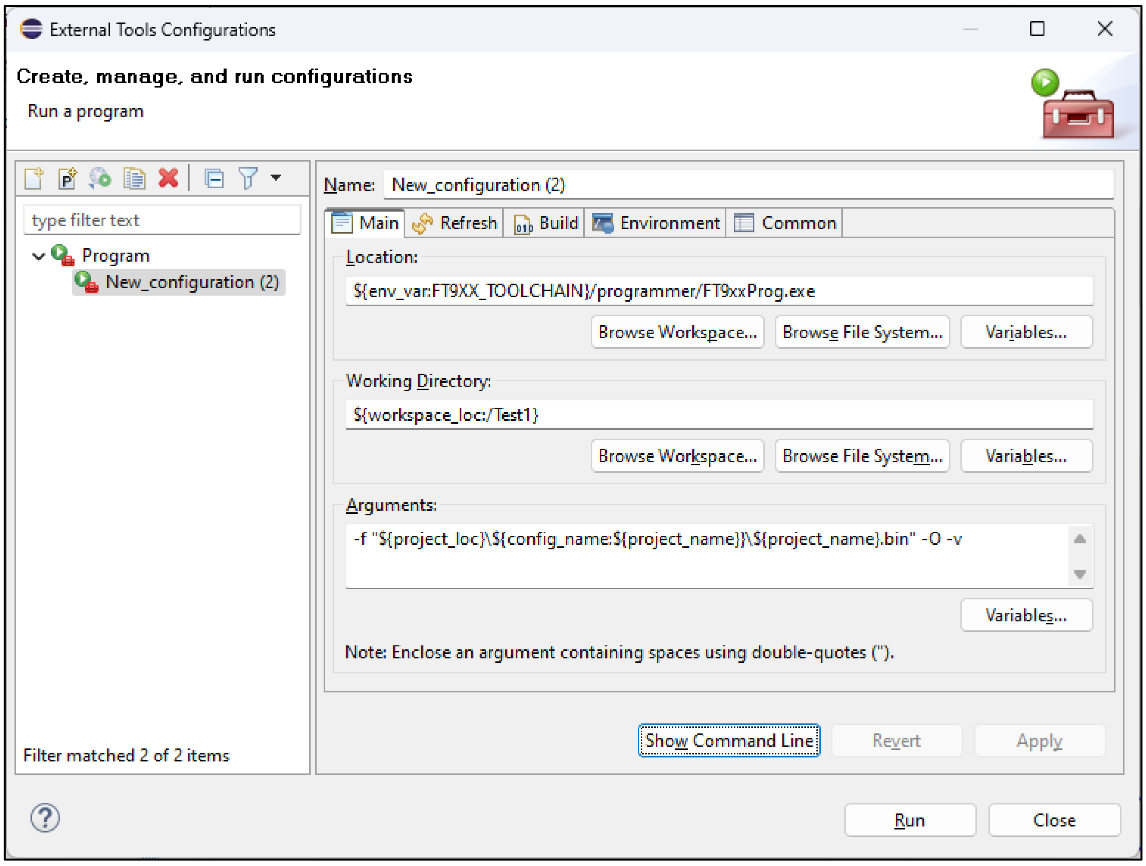
```
        ${env_var:FT9XX_TOOLCHAIN}/programmer/FT9xxProg.exe
```

The argument string is:

```
        -f "${project_loc}\${config_name:${project_name}}\${project_name}.bin" -O -v
```

This can use long options as:

```
        --loadfile "${project_loc}\${config_name:${project_name}}\${project_name}.bin" --
onewire --verify
```



Name: New_configuration (2)

Location:
${env_var:FT9XX_TOOLCHAIN}/programmer/FT9xxProg.exe

Working Directory:
${workspace_loc:/Test1}

Arguments:
-f "${project_loc}\${config_name:${project_name}}\${project_name}.bin" -O -v

Note: Enclose an argument containing spaces using double-quotes (").

Figure 30 FT9XX Programmer in Eclipse

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. The UART0 Tx and Rx lines can be connected to a host computer using a USB serial port adapter. There is an unused serial port on connector J2 of the UMFTPD2A FT9XX Programming Board. The 0.1" pin headers can be connected to the UART on the FT9XX device.

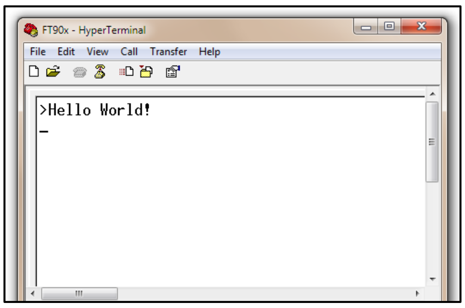
Open a terminal on your computer, for example Tera Term, HyperTerminal or PuTTY. Apply the following settings:

- Baud Rate: 19200

Copyright © Bridgetek Pte Ltd

- Parity Bit: None
- Data Bit: 8
- Stop Bit: 1
- Flow Control: None

Now when you reset the MCU, the message will be printed on the terminal.



```
>Hello World!
_
```

Figure 31 Hello World

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:

"`Documents\Bridgetek\ft9xx-sdk\`*`version`*`\Examples`"

The Eclipse project has already been set up 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 "`Documents\Bridgetek\ft9xx-sdk\`*`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 FT9XX Example Applications for more details about these applications.

Alternatively, the examples can be added directly to your workspace in Eclipse by the following method:

1. On the menu bar, choose **File → New.**

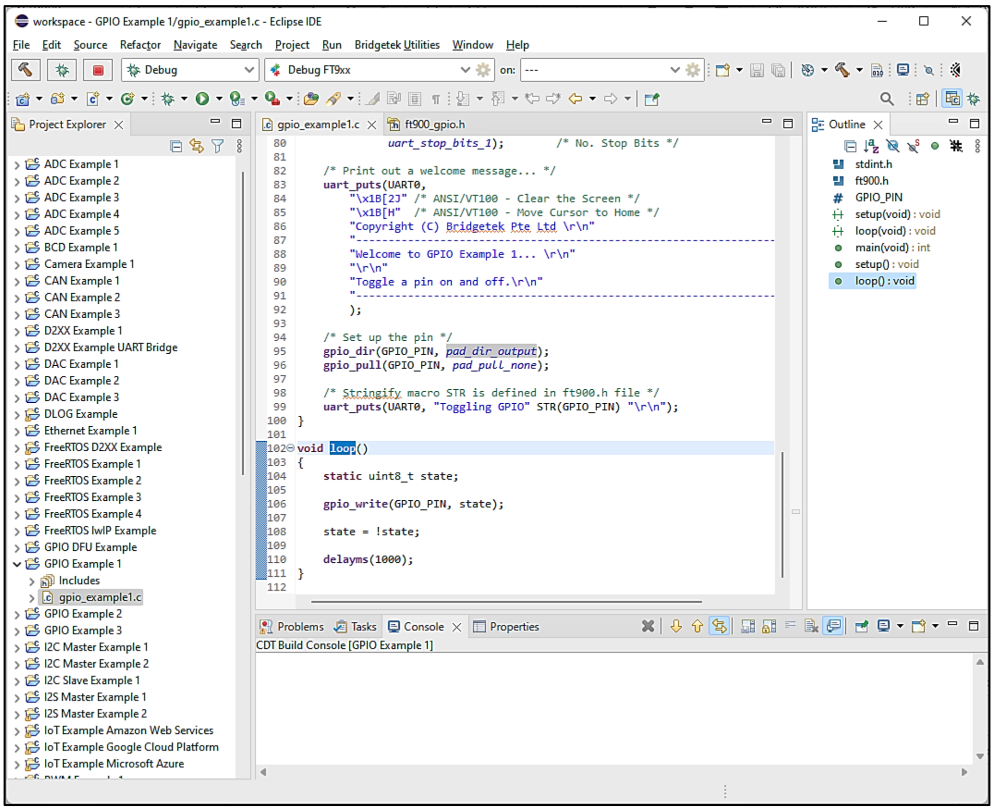
2. In the menu bar, look for the **Example** option.

3. Select the Wizard for "**FT9xx Examples**" and click **Next**.

4. In the next dialog window, choose the example projects you require and press **Finish**.

5. These will be copied to your workspace in the default directory for the workspace.

Copyright © Bridgetek Pte Ltd



```
                uart_stop_bits_1);        /* No. Stop Bits */

    /* Print out a welcome message... */
    uart_puts(UART0,
        "\x1B[2J" /* ANSI/VT100 - Clear the Screen */
        "\x1B[H"  /* ANSI/VT100 - Move Cursor to Home */
        "Copyright (C) Bridgetek Pte Ltd \r\n"
        "--------------------------------------------------------------
        "Welcome to GPIO Example 1... \r\n"
        "\r\n"
        "Toggle a pin on and off.\r\n"
        "--------------------------------------------------------------
        );

    /* Set up the pin */
    gpio_dir(GPIO_PIN, pad_dir_output);
    gpio_pull(GPIO_PIN, pad_pull_none);

    /* Stringify macro STR is defined in ft900.h file */
    uart_puts(UART0, "Toggling GPIO" STR(GPIO_PIN) "\r\n");
}

void loop()
{
    static uint8_t state;

    gpio_write(GPIO_PIN, state);

    state = !state;

    delayms(1000);
}
```

Figure 32 FT9XX Examples

Copyright © Bridgetek Pte Ltd

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.

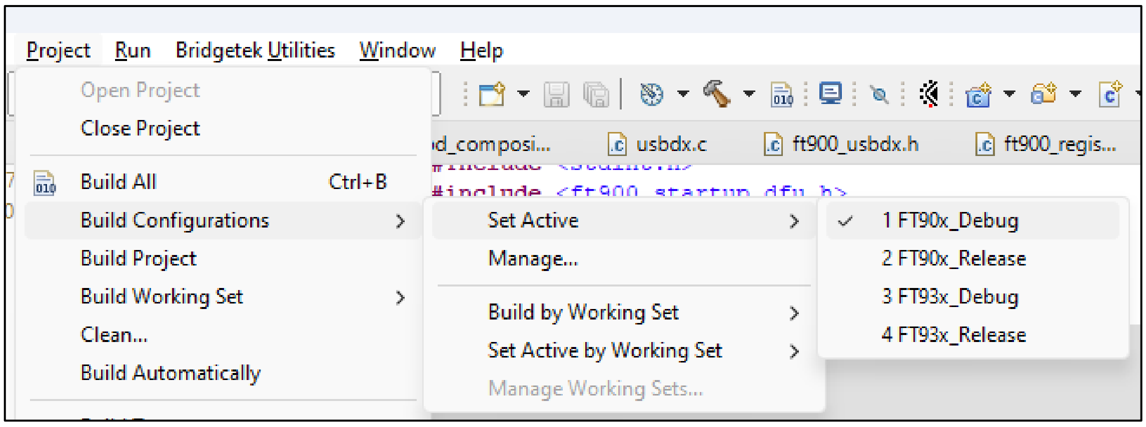


Figure 33 Build Configuration

Please note for FT93X: As for 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 6.4.3 – FT32B options (only for FT93x) for more details.

4.2 Starting a debug session

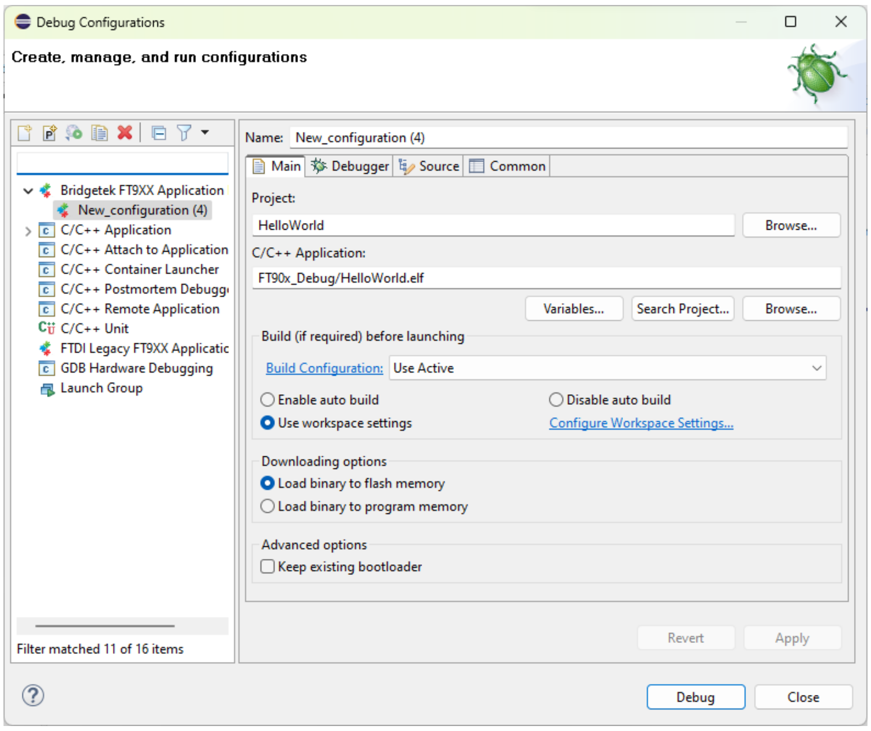
The FT9XX toolchain components, including the FT9XX Programmer, have been fully integrated to the Eclipse IDE to allow a seamless debugging experience.

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 create a new launch configuration.

   Click the button in the Debug Configurations toolbar.
3. This default instance will set up the location of the `ft32-elf-gdb.exe`, set the connection type to remote, set the transport to "TCP", set host name to "`localhost`", set port number to "`3333`" as well as set the project name and elf file name.

Copyright © Bridgetek Pte Ltd


Debug Configurations

Create, manage, and run configurations

- Bridgetek FT9XX Application
  - New_configuration (4)
- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugg
- C/C++ Remote Application
- C/C++ Unit
- FTDI Legacy FT9XX Applicatic
- GDB Hardware Debugging
- Launch Group

Filter matched 11 of 16 items

Name: New_configuration (4)

Main | Debugger | Source | Common

Project: HelloWorld

C/C++ Application: FT90x_Debug/HelloWorld.elf

Build (if required) before launching

Build Configuration: Use Active

Enable auto build | Disable auto build
[✓ Use workspace settings] | Configure Workspace Settings...

Downloading options

[✓ Load binary to flash memory]
Load binary to program memory

Advanced options

☐ Keep existing bootloader

Figure 34 Debug Configuration window

4. Click 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` program to start the debugging session. The Console window will show the progress of the downloading and debugging.

5. For succeeding debugging sessions, select **Run → Debug Last Launched**.

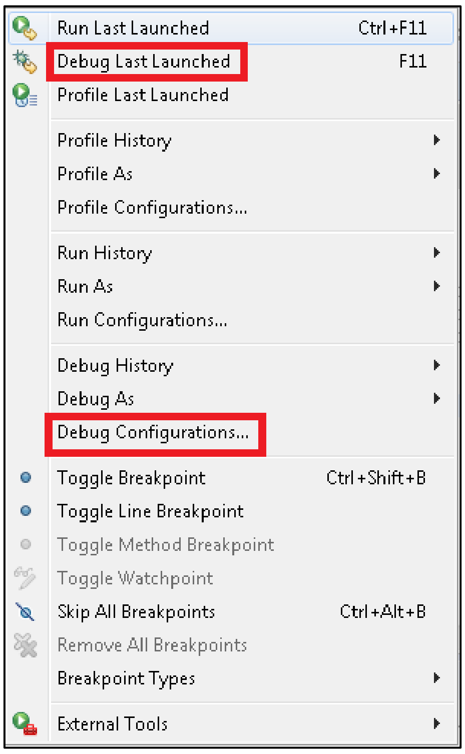
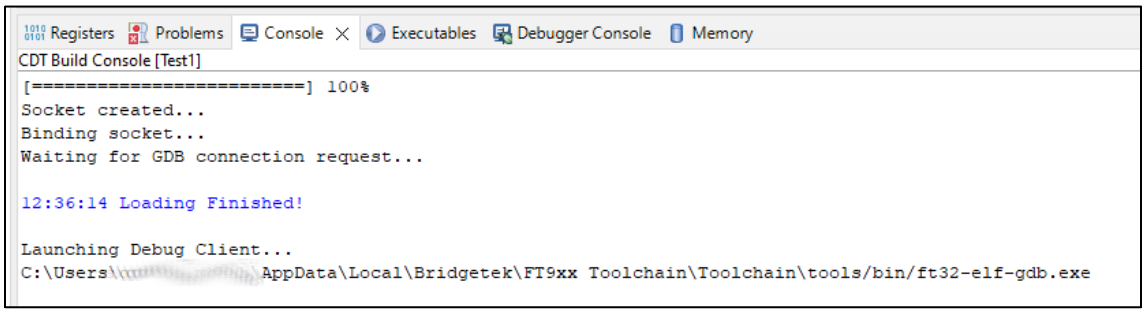


Figure 35 Debug Configurations and Debug Last Launched

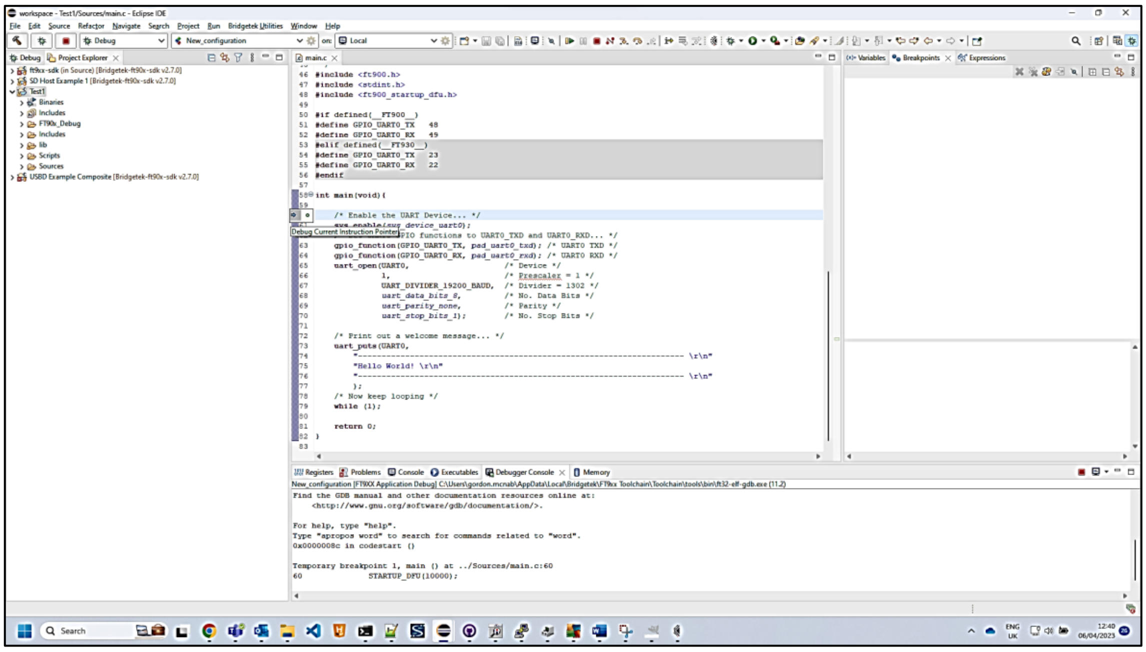


```
[=========================] 100%
Socket created...
Binding socket...
Waiting for GDB connection request...

12:36:14 Loading Finished!

Launching Debug Client...
C:\Users\[redacted]AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain\tools/bin/ft32-elf-gdb.exe
```

Figure 36 Console window after selecting launch configuration and clicking Debug

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.
3.


workspace - Test1/Sources/main.c - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Bridgetek Utilities Window Help
Debug
New_configuration
on: Local
Debug
Project Explorer
main.c
Variables
Breakpoints
Expressions
Debug Current Instruction Pointer
Registers
Problems
Console
Executables
Debugger Console
Memory

```
#include <ft900.h>
#include <stdint.h>
#include <ft900_startup_dfu.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;
}
```

New_configuration [FT9XX Application Debug] C:\Users\gordon.mcnab\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain\tools\bin\ft32-elf-gdb.exe (11.2)

```
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
0x0000008c in codestart ()

Temporary breakpoint 1, main () at ../Sources/main.c:60
60              STARTUP_DFU(10000);
```

Figure 37 Eclipse Debug Environment

Note: If there is an error message about missing source file as shown below, locate the source file that contains the main() function using the "Locate file…" button.

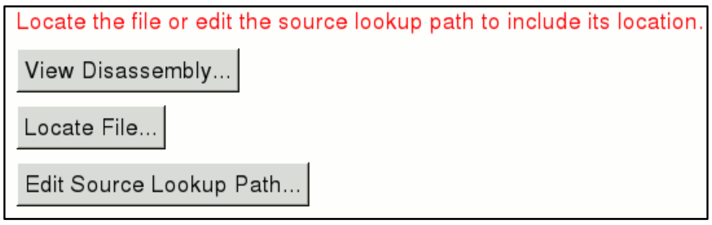

Locate the file or edit the source lookup path to include its location.

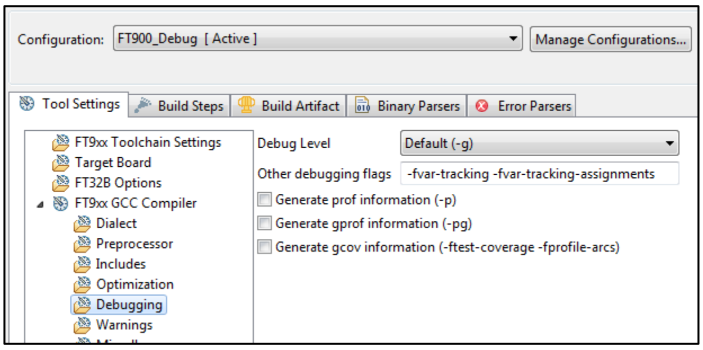
Figure 38 Eclipse Missing Source File

Copyright © Bridgetek Pte Ltd

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
```



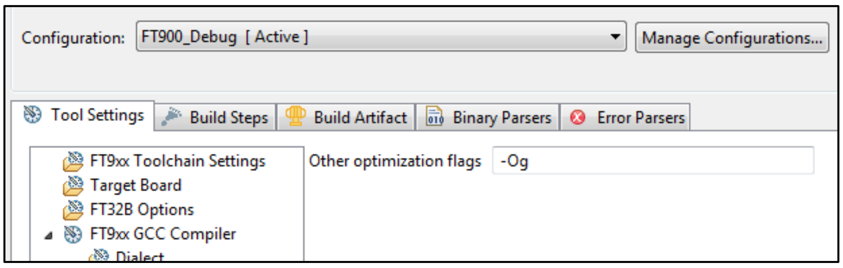
Debug Level: Default (-g)
Other debugging flags: -fvar-tracking -fvar-tracking-assignments
☐ Generate prof information (-p)
☐ Generate gprof information (-pg)
☐ Generate gcov information (-ftest-coverage -fprofile-arcs)

Figure 39 Debug flags

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.



Other optimization flags: -Og

Figure 40 Og compiler optimization option

More information can be found in the GCC documentation - https://gcc.gnu.org/onlinedocs/gcc/Debugging-Options.html and https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

4.4 Eclipse features supported by ft32-elf-gdb

The features of Eclipse debug perspective, supported through `ft32-elf-gdb`, are:

- Breakpoint creation.
- Register view.
- Single stepping/stepping in/stepping out of functions.
- Watch variables.
- Assembly instruction stepping.
- Memory View.

The following features are not supported:

- Watchpoints.

Copyright © Bridgetek Pte Ltd

5 Getting Started with Projects

This section shows how to create new projects from templates or base them on supplied example projects.

Section 3 - Quick Start Guide demonstrates how to make a project with minimum code to activate the UART line. This section uses templates and example files to quickly create more advanced projects.

5.1 Make a Template Project

Bridgetek provides 4 basic template projects to help start new applications with the FT9XX devices. Currently, these are "Base Project", "D2XX Template", "DLOG Template" and "FreeRTOS Template".

For more details about these project types, refer to "AN 360 FT9XX Example Applications" document which is included in the toolchain installation.

These can be selected under "Bridgetek FT9xx Project Templates" in the Bridgetek menu. See Figure 26. They are also available in the Eclipse "**New Project...**" menu.

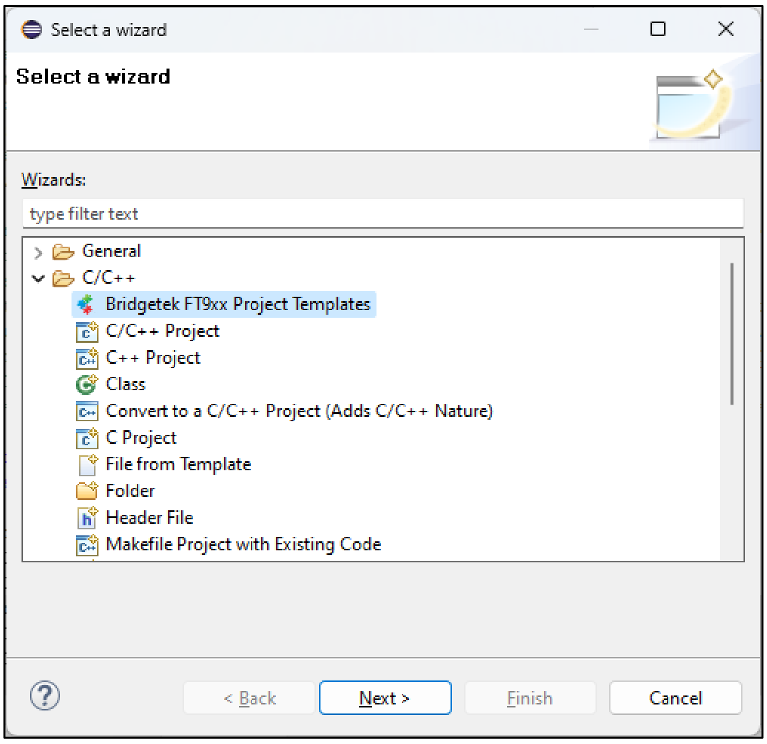
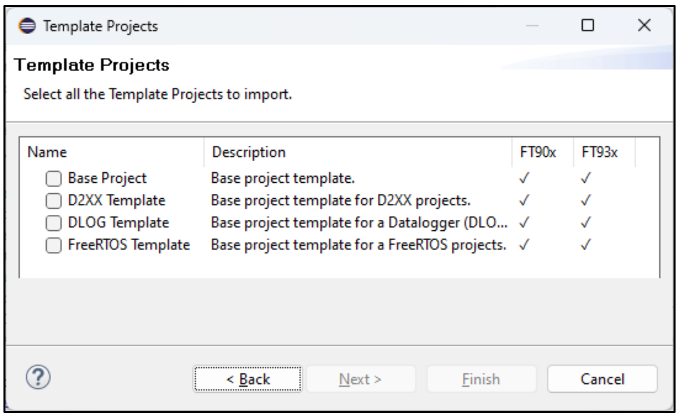


Figure 41 Bridgetek Templates in New Project Wizard

Copyright © Bridgetek Pte Ltd

Selecting this item will display a Wizard window to allow the type of the template to be selected.


Template Projects

Template Projects

Select all the Template Projects to import.

Name	Description	FT90x	FT93x
☐ Base Project	Base project template.	✓	✓
☐ D2XX Template	Base project template for D2XX projects.	✓	✓
☐ DLOG Template	Base project template for a Datalogger (DLO...	✓	✓
☐ FreeRTOS Template	Base project template for a FreeRTOS projects.	✓	✓

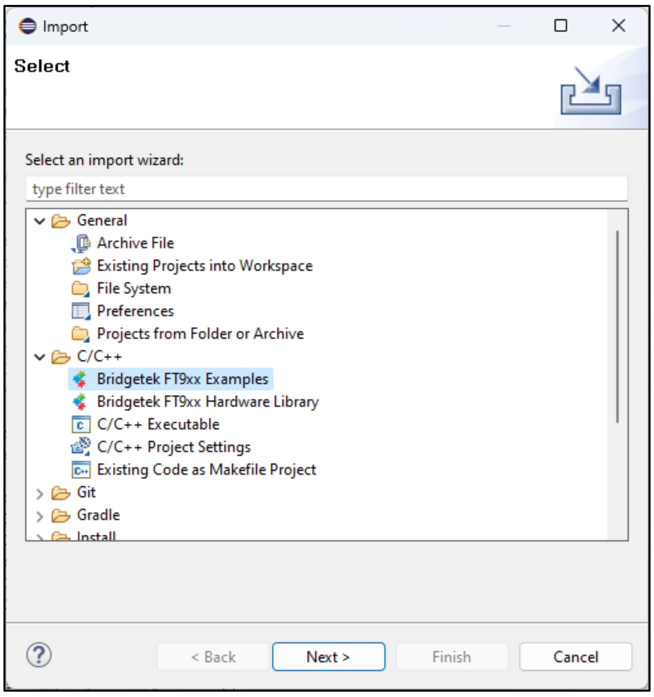
Figure 42 Bridgetek Template Types

Selecting one of the template types on the left will allow the Wizard to finish and the template project will appear in the Project Explorer.

The template project can be compiled as it is, but additional code is needed to customize it according to the user's need. The code is copied to the Eclipse workspace and can be modified as required.

5.2 Import an Example Project

The Eclipse "**Import…**" menu has an option to import "Bridgetek FT9xx Example". This works in the same manner as the "**New Project…**" method in the previous section.


Import

Select

Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
  - Projects from Folder or Archive
- C/C++
  - Bridgetek FT9xx Examples
  - Bridgetek FT9xx Hardware Library
  - C/C++ Executable
  - C/C++ Project Settings
  - Existing Code as Makefile Project
- Git
- Gradle
- Install

Figure 43 Bridgetek Template Types

Complete projects containing examples for accessing hardware peripherals on the FT9xx devices can be obtained quickly and easily with this method. The code is copied to the Eclipse workspace directory when they are imported.

Copyright © Bridgetek Pte Ltd

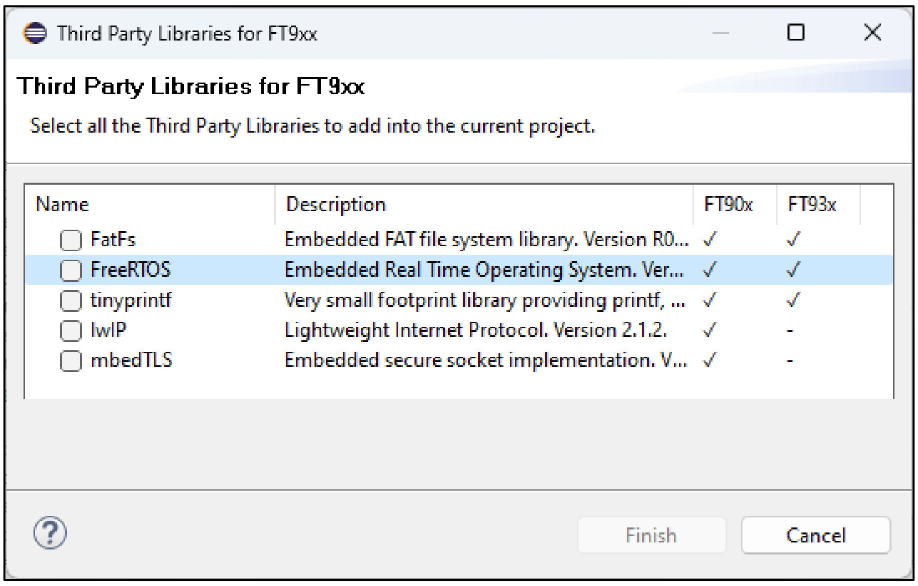
5.3 Adding to a Project

After selecting a project template or an example project, additional drivers or libraries can be added to it.

The wizard to add to a project is accessed through the Bridgetek menu when a project is selected in the Project Explorer.

The currently available libraries are:

- FreeRTOS: A small Real Time Operating System.
- Tinyprintf: A tiny/lightweight general purpose printf function that could be used to replace the GNU library printf function.
- FatFs (based on version 0.15r): 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 2.1.2): A lightweight TCP/IP stack for embedded devices.
- mbedTLS (based on version 2.11.0): A small secure socket implementation (SSL and TLS).



Third Party Libraries for FT9xx

Select all the Third Party Libraries to add into the current project.

Name	Description	FT90x	FT93x
☐ FatFs	Embedded FAT file system library. Version R0...	✓	✓
☐ FreeRTOS	Embedded Real Time Operating System. Ver...	✓	✓
☐ tinyprintf	Very small footprint library providing printf, ...	✓	✓
☐ lwIP	Lightweight Internet Protocol. Version 2.1.2.	✓	-
☐ mbedTLS	Embedded secure socket implementation. V...	✓	-

Figure 44 Add-on Third Party Libraries

The FT9XX Hardware Library files and Layered USB Drivers can also be added to a project by using the relevant Wizard.

6 Advanced Topics

6.1 Command prompt use of the GNU tools

6.1.1 Compiler: ft32-elf-gcc

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

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

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

6.1.2 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:

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

6.1.3 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 converts 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:

  ```
  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:

  ```
  ft32-elf-ld --oformat binary -o file.bin file.elf
  ```

6.1.4 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 4.3 – Debugging the application in Eclipse of this document.

6.1.5 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
```

Copyright © Bridgetek Pte Ltd

6.2 Running the Toolchain from the command prompt

6.2.1 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 that the official GNU Toolchain is used, often with the help of a Makefile or a batch file. The sample applications are available in "`Documents\Bridgetek\ft9xx-sdk\`*`version`*`\Examples\`", if you have installed them using the installation wizard.

<u>Note</u>: Makefiles are not included with the toolchain installer.

6.2.2 Programming a binary image into the chip

The programmer can be found in the folder "`programmer`" in the program installation directory ("`AppData\Local\Bridgetek\FT9xx Toolchain`"). The command line programmer is `FT9xxProg.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 the 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 `FT9xxProg.exe` with the `--help` argument. The options, and usage will be printed. The most common usage is programming a binary file through the one-wire interface with the supplied bootloader. To do this, the command is:

   ```
   FT9xxProg.exe --loadflash <.bin file with path if needed> --onewire
   ```

   In which the options are:
   --loadflash: programming the binary file into the flash. The path to the binary file must follow.
   --onewire: using the one-wire interface.

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

   ```
   FT9xxProg.exe --loadflash <.bin file with path if needed> --onewire -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:

   ```
   FT9xxProg.exe --loadflash <.bin file with path if needed> --onewire -x
   ```

   However, the support for GDB debugging will not be available.

6.2.3 Debugging the sample applications with ft32-elf-gdb

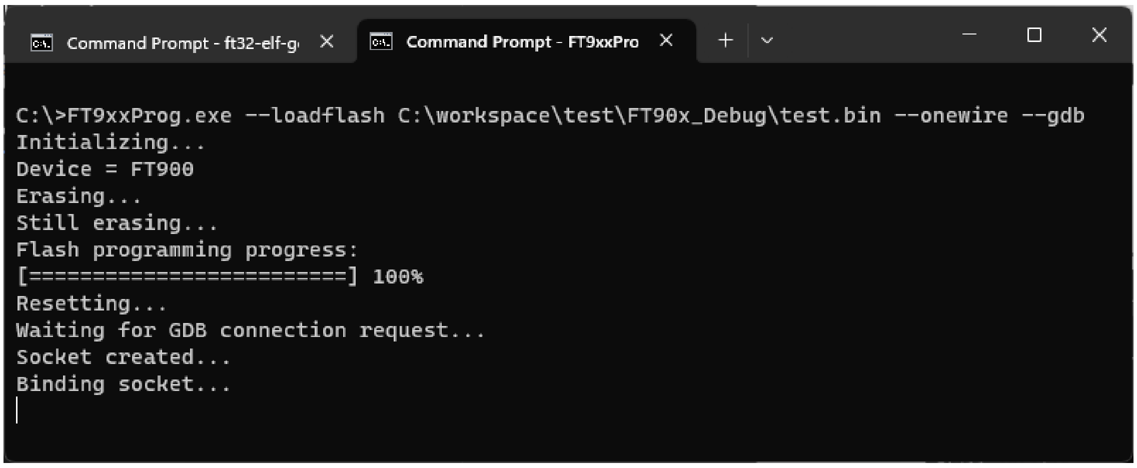
1. The applications must be compiled with –g option (i.e., `ft32-elf-gcc -g` …). An `.elf` file will be created which includes the debug information, for example `FT90x_Debug/Test1.elf`. Note that this file is not used for programming the chip.

   <u>Note</u>: 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 and start the GDB bridge function on the programming utility.

   ```
   FT9xxProg.exe --loadflash <.bin file with path if needed> --onewire -gdb
   ```

   The correct response should be:

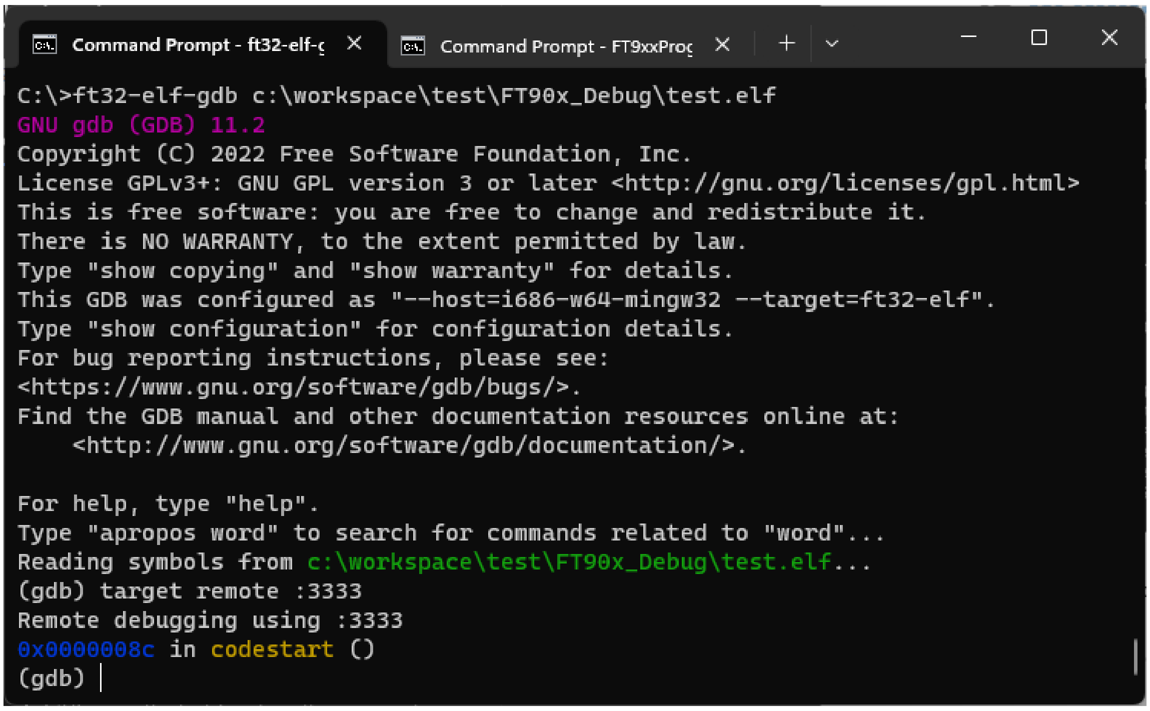


```
C:\>FT9xxProg.exe --loadflash C:\workspace\test\FT90x_Debug\test.bin --onewire --gdb
Initializing...
Device = FT900
Erasing...
Still erasing...
Flash programming progress:
[========================] 100%
Resetting...
Waiting for GDB connection request...
Socket created...
Binding socket...
```

Figure 45 FT9XX GDB bridge

3. 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 Test1.elf`".

4. After ft32-elf-gdb starts, type in "`target remote localhost:3333`" to establish a connection to the MCU.



```
C:\>ft32-elf-gdb c:\workspace\test\FT90x_Debug\test.elf
GNU gdb (GDB) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=ft32-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from c:\workspace\test\FT90x_Debug\test.elf...
(gdb) target remote :3333
Remote debugging using :3333
0x0000008c in codestart ()
(gdb)
```

Figure 46 FT9XX gdb to GDB bridge

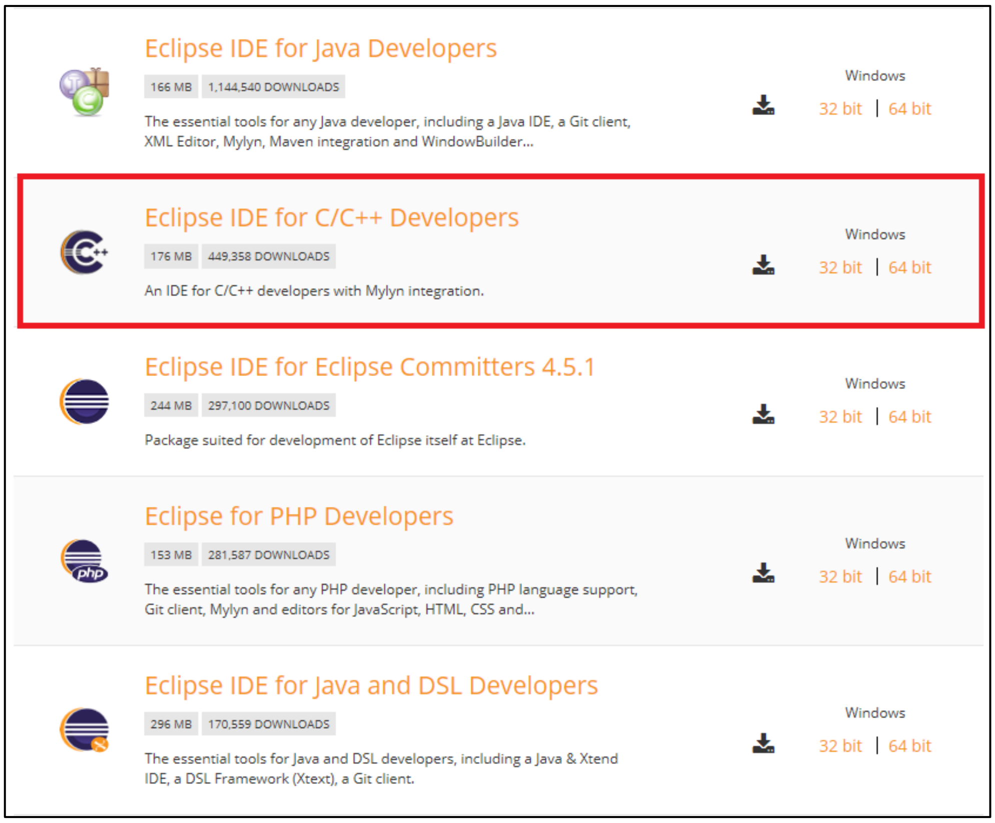
5. Use standard GDB commands to debug the program. Note that the command to start execution should be "`continue`", not "`run`".

6.3 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 has already installed Eclipse for other purposes. This section details how to set it up for use with the FT9XX.

6.3.1 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.



Eclipse IDE for Java Developers
166 MB | 1,144,540 DOWNLOADS
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven integration and WindowBuilder...
Windows 32 bit | 64 bit

Eclipse IDE for C/C++ Developers
176 MB | 449,358 DOWNLOADS
An IDE for C/C++ developers with Mylyn integration.
Windows 32 bit | 64 bit

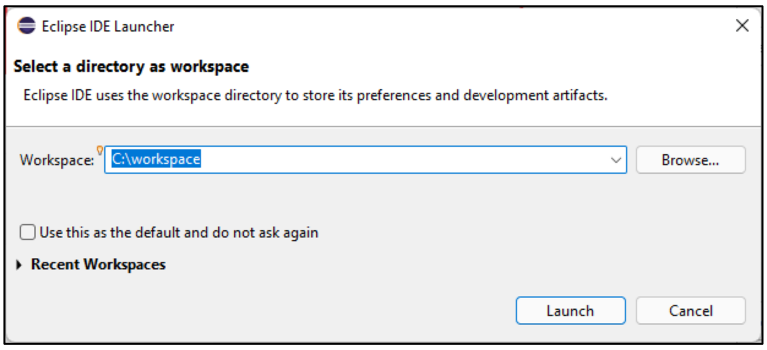
Eclipse IDE for Eclipse Committers 4.5.1
244 MB | 297,100 DOWNLOADS
Package suited for development of Eclipse itself at Eclipse.
Windows 32 bit | 64 bit

Eclipse for PHP Developers
153 MB | 281,587 DOWNLOADS
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and...
Windows 32 bit | 64 bit

Eclipse IDE for Java and DSL Developers
296 MB | 170,559 DOWNLOADS
The essential tools for Java and DSL developers, including a Java & Xtend IDE, a DSL Framework (Xtext), a Git client.
Windows 32 bit | 64 bit

Figure 47 Eclipse Versions

2. When Eclipse is run for the first time, it will ask for the workspace location.



Eclipse IDE Launcher
Select a directory as workspace
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.
Workspace: C:\workspace

Figure 48 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**.

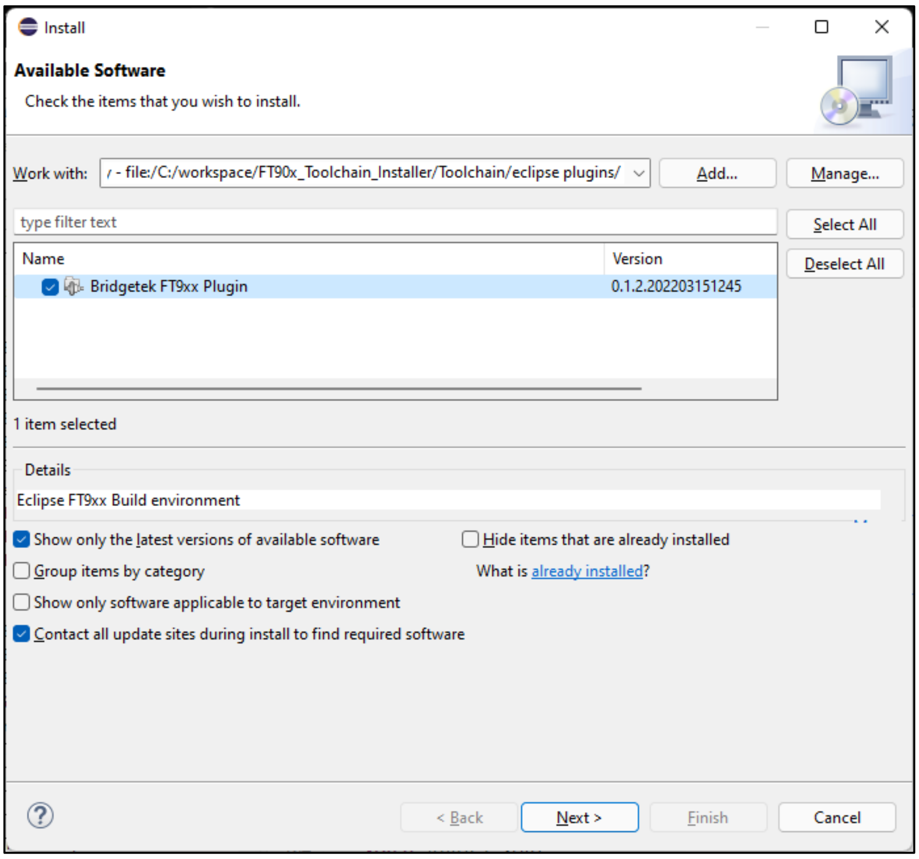
Note: To run Eclipse, it is required to download and install the Java Run Time Environment (JRE) or Java Developer Kit (JDK). Eclipse should display a warning if this is not installed. Oracle provides these tools for free.

Copyright © Bridgetek Pte Ltd

6.3.2 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 shown below.



Install

Available Software

Check the items that you wish to install.

Work with: / - file:/C:/workspace/FT90x_Toolchain_Installer/Toolchain/eclipse plugins/

type filter text

Name	Version
☑ Bridgetek FT9xx Plugin	0.1.2.202203151245

1 item selected

Details

Eclipse FT9xx Build environment

☑ Show only the latest versions of available software
☐ Hide items that are already installed
☐ Group items by category
What is already installed?
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software

Figure 49 Eclipse Plugin Setup Wizard

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

3. Select the **SELECT ALL** button followed by **NEXT** to install the plugin.

4. Close the window when complete.

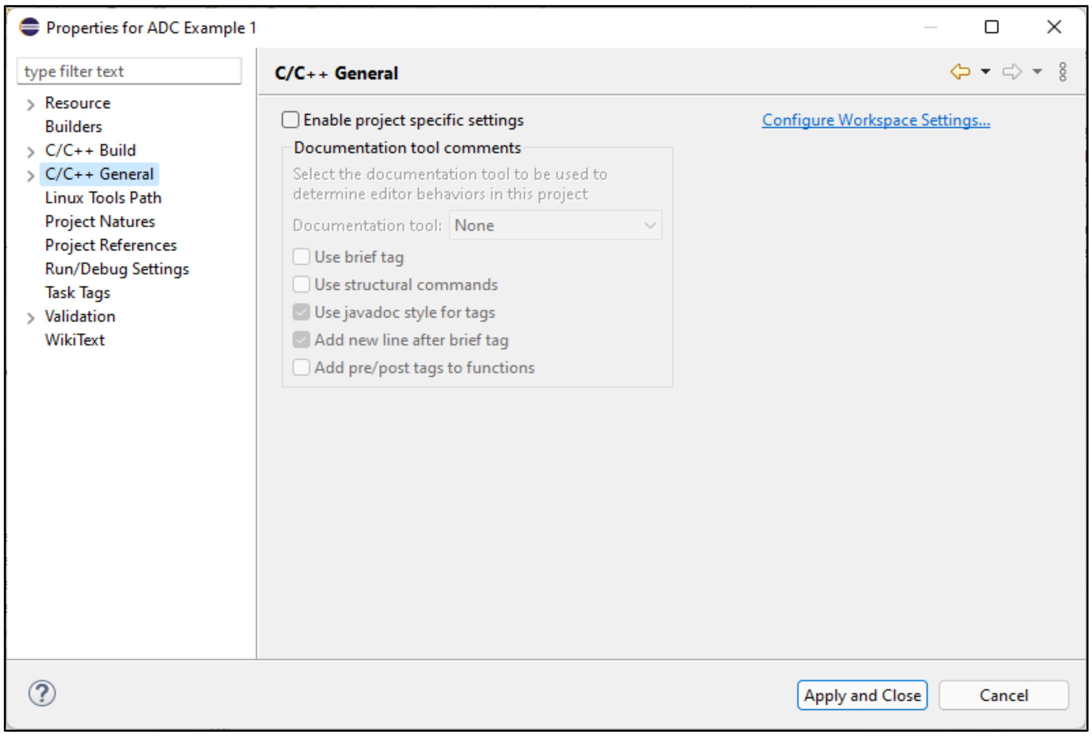
6.4 Common project settings in Eclipse

6.4.1 Include paths

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

1. Right-click the project and select **Properties**
or select the project in the "Project Explorer" and press **Alt+Enter**.

Copyright © Bridgetek Pte Ltd


Properties for ADC Example 1

C/C++ General

Enable project specific settings

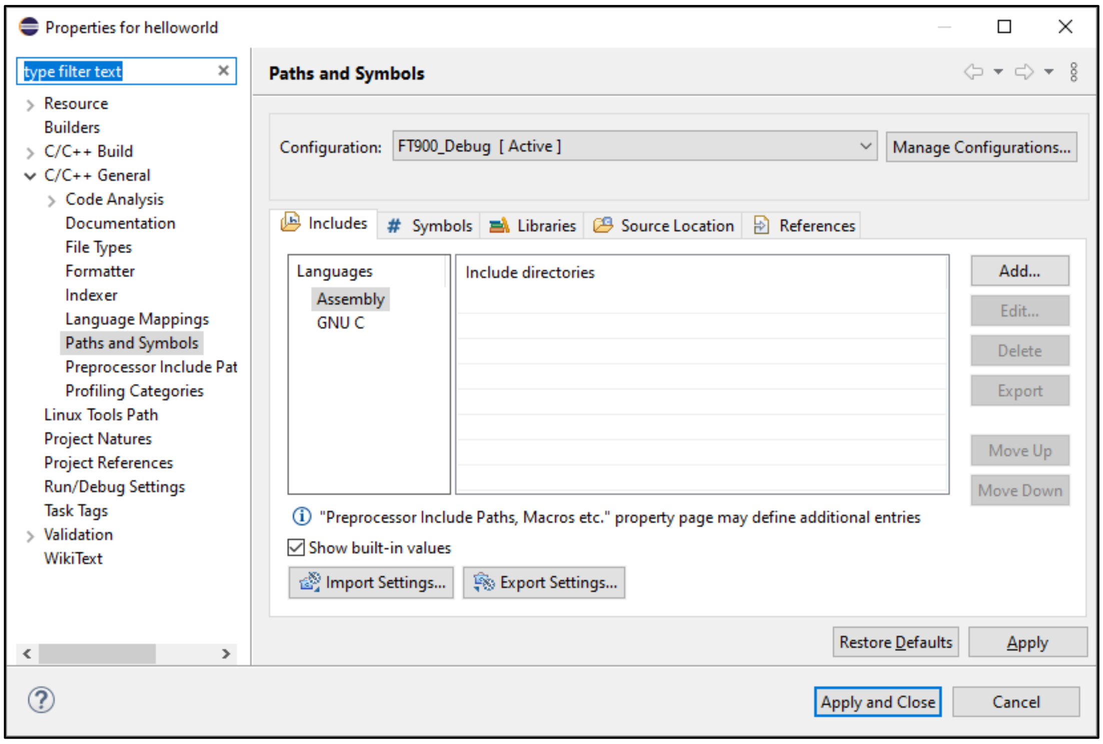
Documentation tool comments

Select the documentation tool to be used to determine editor behaviors in this project

Documentation tool: None

Figure 50 Eclipse Project Properties

2. On the left side of Properties window, select **C/C++ General → Paths and Symbols.**


Properties for helloworld

Paths and Symbols

Configuration: FT900_Debug [Active]

"Preprocessor Include Paths, Macros etc." property page may define additional entries

Show built-in values

Figure 51 Eclipse Paths and Symbols

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

Copyright © Bridgetek Pte Ltd

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, select ☑ **Add to all languages** checkbox, then click **OK**.

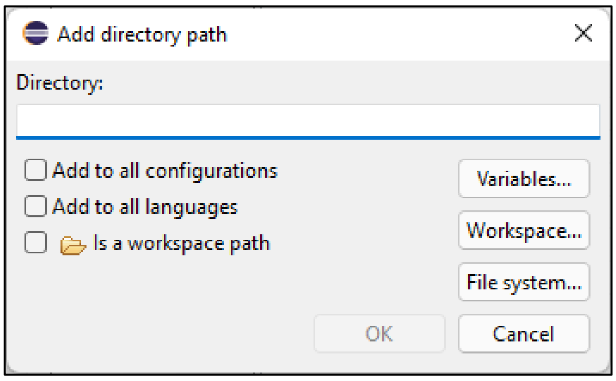

Add directory path
Directory:
☐ Add to all configurations
☐ Add to all languages
☐ Is a workspace path

Figure 52 Eclipse Add Directory Path

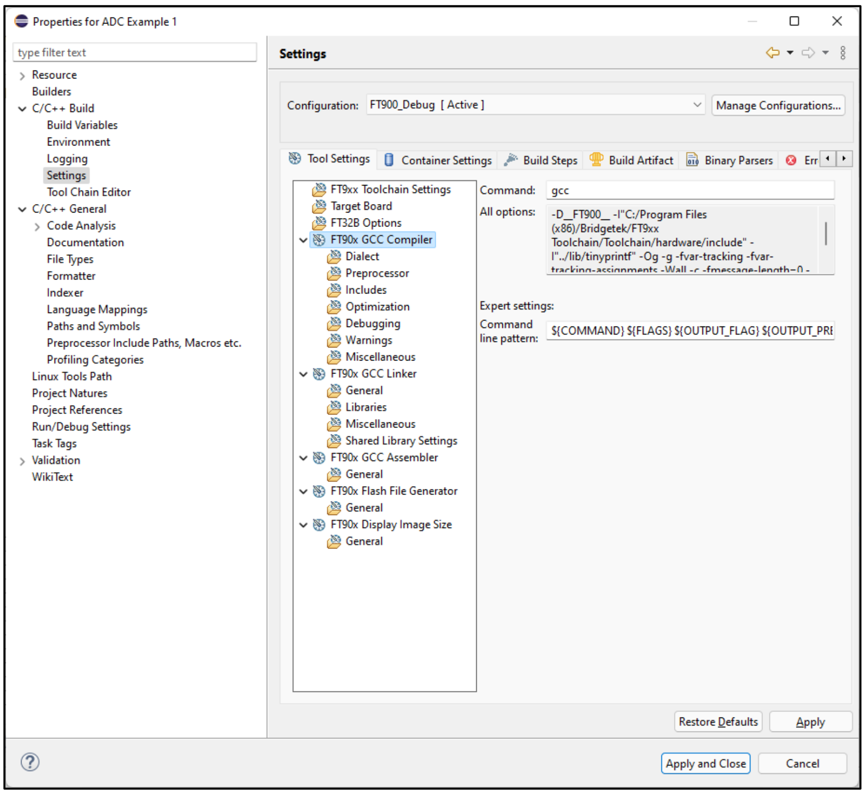
Note: The paths should be added one at a time. The use of semicolons is not supported.

6.4.2 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 as follows:

1. Right-click 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.


Properties for ADC Example 1

Settings

Configuration: FT900_Debug [Active]

Command: gcc

All options:
```
-D__FT900__ -I"C:/Program Files
(x86)/Bridgetek/FT9xx
Toolchain/Toolchain/hardware/include" -
I"../lib/tinyprintf" -Og -g -fvar-tracking -fvar-
tracking-assignments -Wall -c -fmessage-length=0 -
```

Expert settings:
Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PRE

Figure 53 Eclipse FT9XX Toolchain Settings

Copyright © Bridgetek Pte Ltd

6.4.3 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 54. Note that this option is only available in the FT930_* configurations.

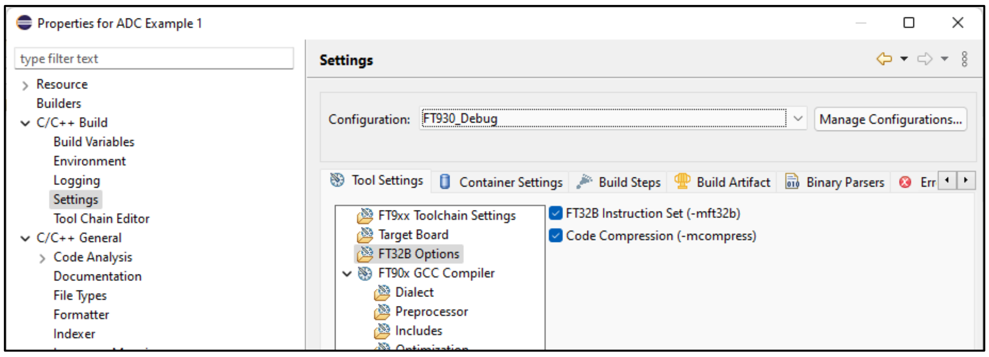
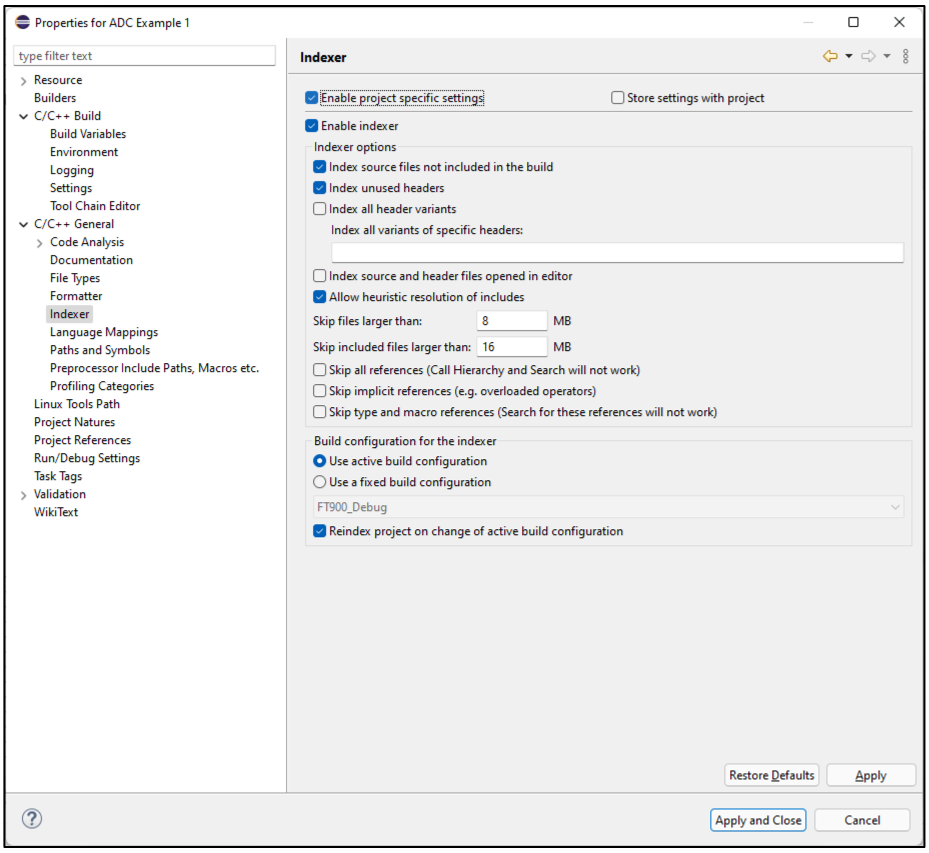

Configuration: FT930_Debug
FT32B Options:
- [✓] FT32B Instruction Set (-mft32b)
- [✓] Code Compression (-mcompress)

Figure 54 FT32B options (for FT93X)

6.4.4 C/C++ Indexer Settings

For Eclipse syntax highlighting to work correctly as you switch configurations, the Indexer must 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**.


Indexer
- [✓] Enable project specific settings
- [ ] Store settings with project
- [✓] Enable indexer

Indexer options
- [✓] Index source files not included in the build
- [✓] Index unused headers
- [ ] Index all header variants

Index all variants of specific headers:
- [ ] Index source and header files opened in editor
- [✓] Allow heuristic resolution of includes

Skip files larger than: 8 MB
Skip included files larger than: 16 MB
- [ ] Skip all references (Call Hierarchy and Search will not work)
- [ ] Skip implicit references (e.g. overloaded operators)
- [ ] Skip type and macro references (Search for these references will not work)

Build configuration for the indexer
- (●) Use active build configuration
- ( ) Use a fixed build configuration

FT900_Debug
- [✓] Reindex project on change of active build configuration

Figure 55 Configure Eclipse indexer to use the active build configuration

Copyright © Bridgetek Pte Ltd

6.4.5 C++ Compilation

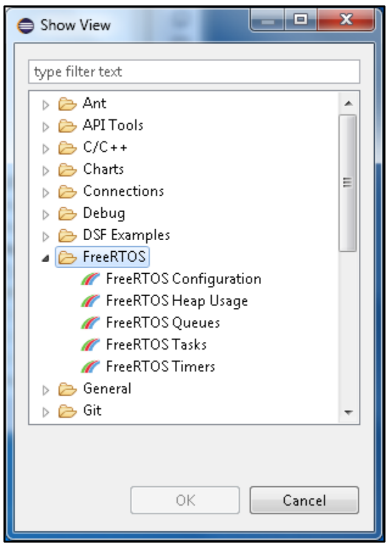
C++ compilation is supported; however, there is no support for the standard C++ libraries or STL libraries. Some language features are also not supported e.g., new and delete operators. Include files of FT9XX libraries are C++ safe when wrapped in extern "C" {} scope in order to be excluded from the 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.

6.5 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**.



6.5.1 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

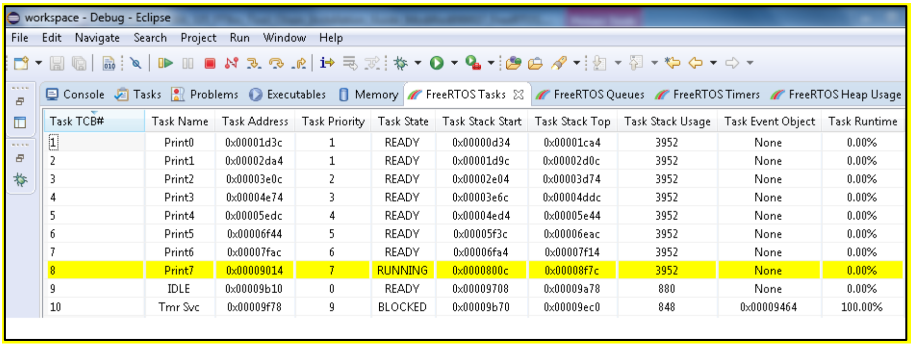
A tooltip description is also provided for each column of the views. The tooltip describes the column values.

For a demonstration of these views, refer to "FreeRTOS Example 4" application.

6.5.2 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 "DELETED" state. These tasks will disappear in the view only when the idle task has freed up the corresponding memory allocation.


workspace - Debug - Eclipse
File Edit Navigate Search Project Run Window Help
Console Tasks Problems Executables Memory FreeRTOS Tasks FreeRTOS Queues FreeRTOS Timers FreeRTOS Heap Usage

Task TCB#	Task Name	Task Address	Task Priority	Task State	Task Stack Start	Task Stack Top	Task Stack Usage	Task Event Object	Task Runtime
1	Print0	0x00001d3c	1	READY	0x00000d34	0x00001ca4	3952	None	0.00%
2	Print1	0x00002da4	1	READY	0x00001d9c	0x00002d0c	3952	None	0.00%
3	Print2	0x00003e0c	2	READY	0x00002e04	0x00003d74	3952	None	0.00%
4	Print3	0x00004e74	3	READY	0x00003e6c	0x00004ddc	3952	None	0.00%
5	Print4	0x00005edc	4	READY	0x00004ed4	0x00005e44	3952	None	0.00%
6	Print5	0x00006f44	5	READY	0x00005f3c	0x00006eac	3952	None	0.00%
7	Print6	0x00007fac	6	READY	0x00006fa4	0x00007f14	3952	None	0.00%
8	Print7	0x00009014	7	RUNNING	0x0000800c	0x00008f7c	3952	None	0.00%
9	IDLE	0x00009b10	0	READY	0x00009708	0x00009a78	880	None	0.00%
10	Tmr Svc	0x00009f78	9	BLOCKED	0x00009b70	0x00009ec0	848	0x00009464	100.00%

Figure 56 FreeRTOS Tasks view

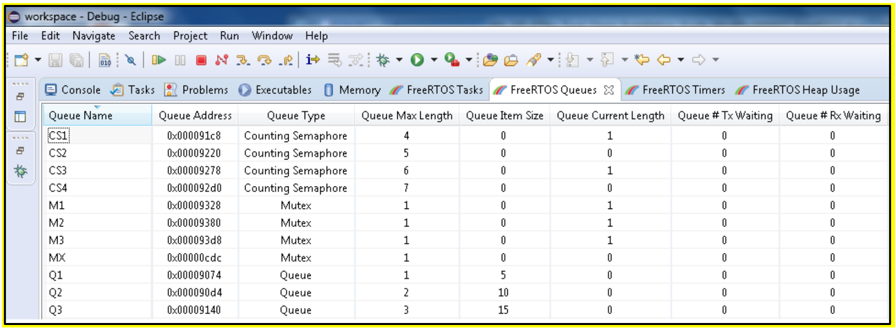
Some of these properties are dependent on a configuration in "FreeRTOSConfig.h". If these macros are set to 0, "Unknown" value will be displayed. Refer to the table below for these dependencies.

FreeRTOS Task Property	FreeRTOS Configuration macro
Task TCB#	configUSE_TRACE_FACILITY
Task Runtime	configGENERATE_RUN_TIME_STATS

Table 1 FreeRTOS Tasks macro dependencies

6.5.3 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.


workspace - Debug - Eclipse
File Edit Navigate Search Project Run Window Help
Console Tasks Problems Executables Memory FreeRTOS Tasks FreeRTOS Queues FreeRTOS Timers FreeRTOS Heap Usage

Queue Name	Queue Address	Queue Type	Queue Max Length	Queue Item Size	Queue Current Length	Queue # Tx Waiting	Queue # Rx Waiting
CS1	0x000091c8	Counting Semaphore	4	0	1	0	0
CS2	0x00009220	Counting Semaphore	5	0	0	0	0
CS3	0x00009278	Counting Semaphore	6	0	1	0	0
CS4	0x000092d0	Counting Semaphore	7	0	0	0	0
M1	0x00009328	Mutex	1	0	1	0	0
M2	0x00009380	Mutex	1	0	1	0	0
M3	0x000093d8	Mutex	1	0	1	0	0
MX	0x00000cdc	Mutex	1	0	0	0	0
Q1	0x00009074	Queue	1	5	0	0	0
Q2	0x000090d4	Queue	2	10	0	0	0
Q3	0x00009140	Queue	3	15	0	0	0

Figure 57 FreeRTOS Queues view

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	**FreeRTOS Configuration macro**

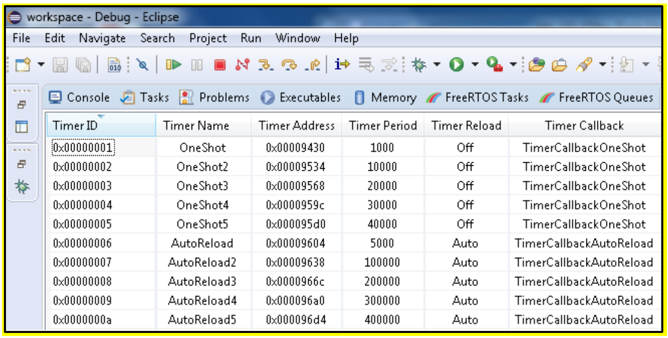
Copyright © Bridgetek Pte Ltd

below for these dependencies. **FreeRTOS Queue Property**	
Queue Type	configUSE_TRACE_FACILITY

Table 2 FreeRTOS Queues macro dependencies

6.5.4 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.

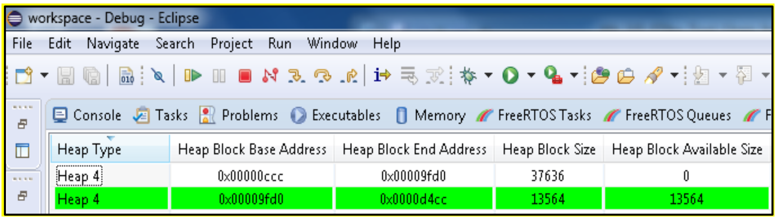

workspace - Debug - Eclipse
File Edit Navigate Search Project Run Window Help
Console Tasks Problems Executables Memory FreeRTOS Tasks FreeRTOS Queues

Timer ID	Timer Name	Timer Address	Timer Period	Timer Reload	Timer Callback
0x00000001	OneShot	0x00009430	1000	Off	TimerCallbackOneShot
0x00000002	OneShot2	0x00009534	10000	Off	TimerCallbackOneShot
0x00000003	OneShot3	0x00009568	20000	Off	TimerCallbackOneShot
0x00000004	OneShot4	0x0000959c	30000	Off	TimerCallbackOneShot
0x00000005	OneShot5	0x000095d0	40000	Off	TimerCallbackOneShot
0x00000006	AutoReload	0x00009604	5000	Auto	TimerCallbackAutoReload
0x00000007	AutoReload2	0x00009638	100000	Auto	TimerCallbackAutoReload
0x00000008	AutoReload3	0x0000966c	200000	Auto	TimerCallbackAutoReload
0x00000009	AutoReload4	0x000096a0	300000	Auto	TimerCallbackAutoReload
0x0000000a	AutoReload5	0x000096d4	400000	Auto	TimerCallbackAutoReload

Figure 58 FreeRTOS Timers view

6.5.5 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.


workspace - Debug - Eclipse
File Edit Navigate Search Project Run Window Help
Console Tasks Problems Executables Memory FreeRTOS Tasks FreeRTOS Queues

Heap Type	Heap Block Base Address	Heap Block End Address	Heap Block Size	Heap Block Available Size
Heap 4	0x00000ccc	0x00009fd0	37636	0
Heap 4	0x00009fd0	0x0000d4cc	13564	13564

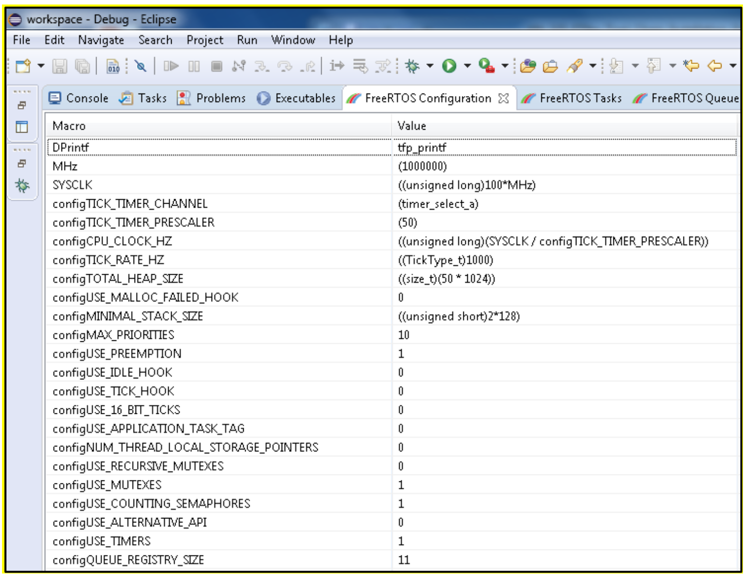
Figure 59 FreeRTOS Heap Usage view

The heap type value can be changed via the preprocessor symbol FT32_PORT_HEAP.

6.5.6 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.

Copyright © Bridgetek Pte Ltd


workspace - Debug - Eclipse
File Edit Navigate Search Project Run Window Help
Console Tasks Problems Executables FreeRTOS Configuration FreeRTOS Tasks FreeRTOS Queue

Macro	Value
DPrintf	tfp_printf
MHz	(1000000)
SYSCLK	((unsigned long)100*MHz)
configTICK_TIMER_CHANNEL	(timer_select_a)
configTICK_TIMER_PRESCALER	(50)
configCPU_CLOCK_HZ	((unsigned long)(SYSCLK / configTICK_TIMER_PRESCALER))
configTICK_RATE_HZ	((TickType_t)1000)
configTOTAL_HEAP_SIZE	((size_t)(50 * 1024))
configUSE_MALLOC_FAILED_HOOK	0
configMINIMAL_STACK_SIZE	((unsigned short)2*128)
configMAX_PRIORITIES	10
configUSE_PREEMPTION	1
configUSE_IDLE_HOOK	0
configUSE_TICK_HOOK	0
configUSE_16_BIT_TICKS	0
configUSE_APPLICATION_TASK_TAG	0
configNUM_THREAD_LOCAL_STORAGE_POINTERS	0
configUSE_RECURSIVE_MUTEXES	0
configUSE_MUTEXES	1
configUSE_COUNTING_SEMAPHORES	1
configUSE_ALTERNATIVE_API	0
configUSE_TIMERS	1
configQUEUE_REGISTRY_SIZE	11

Figure 60 FreeRTOS Configuration 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 to it will be highlighted in yellow. To save the changes, click **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 supports macros, multi-line macro definitions and macros enclosed in `#ifdef` or `#if defined`.

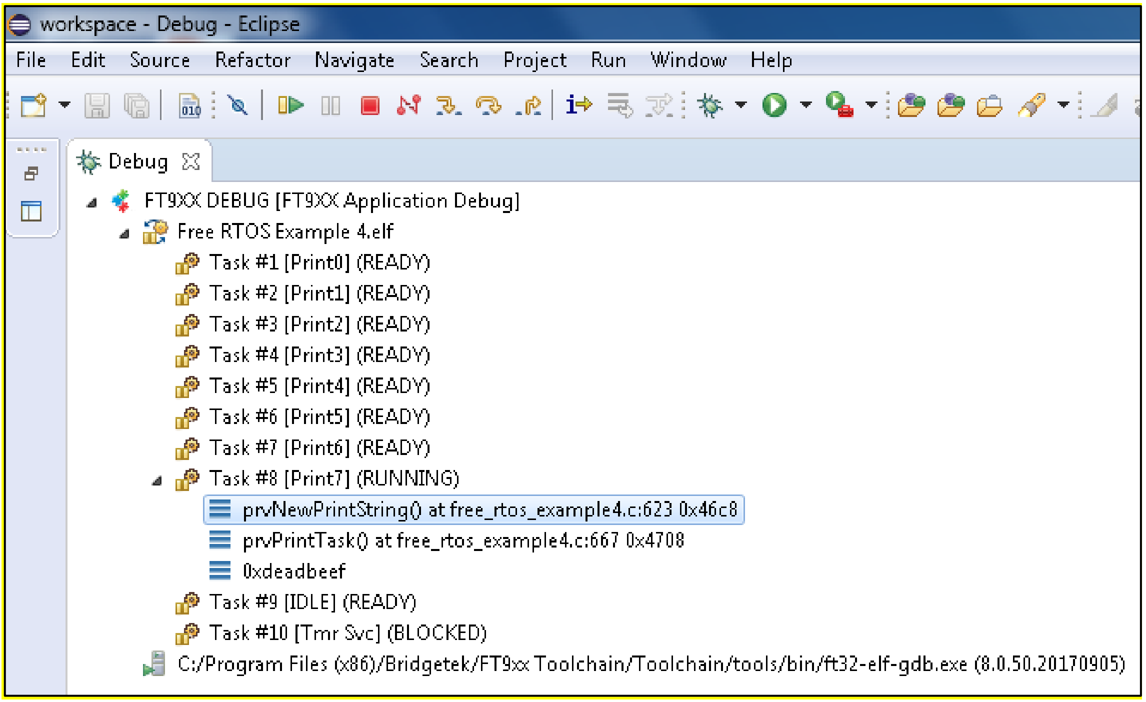
6.5.7 Built-in Debug view

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, the 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 "DELETED" state. These tasks will disappear in the view only when the idle task has freed up the corresponding memory allocation.

Copyright © Bridgetek Pte Ltd


workspace - Debug - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug

```
FT9XX DEBUG [FT9XX Application Debug]
  Free RTOS Example 4.elf
    Task #1 [Print0] (READY)
    Task #2 [Print1] (READY)
    Task #3 [Print2] (READY)
    Task #4 [Print3] (READY)
    Task #5 [Print4] (READY)
    Task #6 [Print5] (READY)
    Task #7 [Print6] (READY)
    Task #8 [Print7] (RUNNING)
      prvNewPrintString() at free_rtos_example4.c:623 0x46c8
      prvPrintTask() at free_rtos_example4.c:667 0x4708
      0xdeadbeef
    Task #9 [IDLE] (READY)
    Task #10 [Tmr Svc] (BLOCKED)
  C:/Program Files (x86)/Bridgetek/FT9xx Toolchain/Toolchain/tools/bin/ft32-elf-gdb.exe (8.0.50.20170905)
```

Figure 61 Built-in Debug view

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 Programming does not work for either RUN or DEBUG

1. Ensure that the manually executed FT900 Programmer and GDB Bridge are not running. The new Eclipse plugin automatically runs the FT900 Programmer and GDB Bridge, so the 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 Manage**r and kill the process.

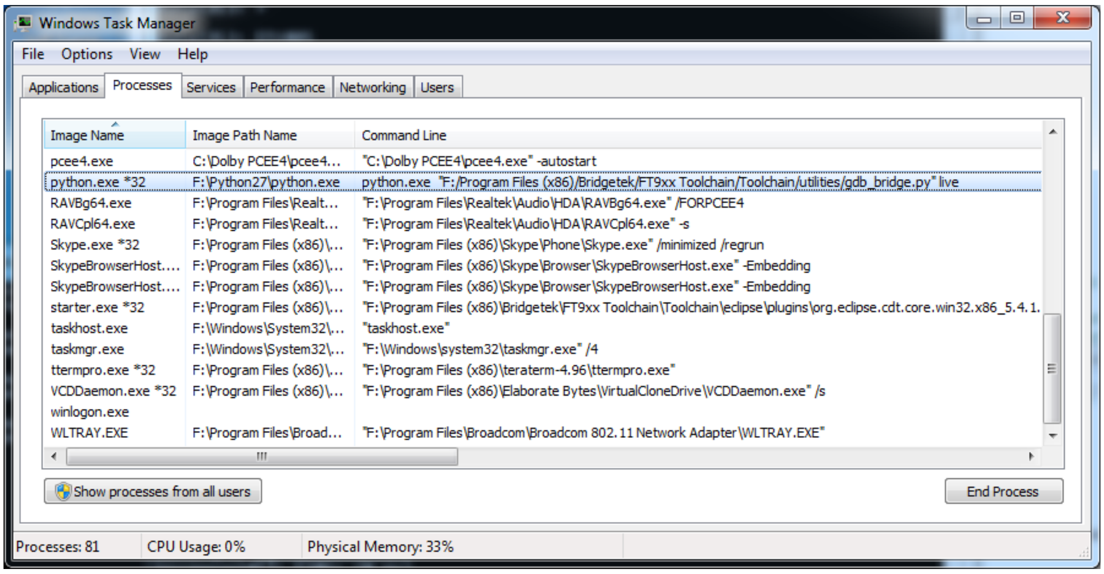

Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Show processes from all users
End Process

Image Name	Image Path Name	Command Line
pcee4.exe	C:\Dolby PCEE4\pcee4...	"C:\Dolby PCEE4\pcee4.exe" -autostart
python.exe *32	F:\Python27\python.exe	python.exe "F:/Program Files (x86)/Bridgetek/FT9xx Toolchain/Toolchain/utilities/gdb_bridge.py" live
RAVBg64.exe	F:\Program Files\Realt...	"F:\Program Files\Realtek\Audio\HDA\RAVBg64.exe" /FORPCEE4
RAVCpl64.exe	F:\Program Files\Realt...	"F:\Program Files\Realtek\Audio\HDA\RAVCpl64.exe" -s
Skype.exe *32	F:\Program Files (x86)\...	"F:\Program Files (x86)\Skype\Phone\Skype.exe" /minimized /regrun
SkypeBrowserHost....	F:\Program Files (x86)\...	"F:\Program Files (x86)\Skype\Browser\SkypeBrowserHost.exe" -Embedding
SkypeBrowserHost....	F:\Program Files (x86)\...	"F:\Program Files (x86)\Skype\Browser\SkypeBrowserHost.exe" -Embedding
starter.exe *32	F:\Program Files (x86)\...	"F:\Program Files (x86)\Bridgetek\FT9xx Toolchain\Toolchain\eclipse\plugins\org.eclipse.cdt.core.win32.x86_5.4.1.
taskhost.exe	F:\Windows\System32\...	"taskhost.exe"
taskmgr.exe	F:\Windows\System32\...	"F:\Windows\system32\taskmgr.exe" /4
ttermpro.exe *32	F:\Program Files (x86)\...	"F:\Program Files (x86)\teraterm-4.96\ttermpro.exe"
VCDDaemon.exe *32	F:\Program Files (x86)\...	"F:\Program Files (x86)\Elaborate Bytes\VirtualCloneDrive\VCDDaemon.exe" /s
winlogon.exe		
WLTRAY.EXE	F:\Program Files\Broad...	"F:\Program Files\Broadcom\Broadcom 802.11 Network Adapter\WLTRAY.EXE"

Processes: 81 CPU Usage: 0% Physical Memory: 33%

Figure 62 Eclipse-executed invisible GDB Bridge

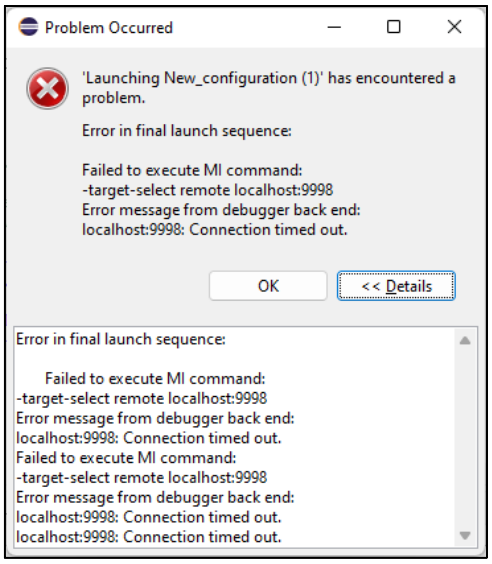
3. The program has overwritten the bootloader space (the top 4kB of the 256 kB of program memory). This will prevent the bootloader code from responding to requests from the GDB Bridge. The linker script should prevent this happening, but it can be done inadvertently. Check that the size of the *bin* file generated for the program is smaller than 252 kB in size.

7.2 Debugging does not work

The FT9XX Debugger relies on the GDB bridge within the FT9xxProg utility to perform the actions required by `ft32-elf-gdb`. The debugger and bridge communicate *via* sockets so they rely on an internal virtual network connection.

If the Eclipse IDE cannot call the utility to start the GDB Bridge, or it cannot access the sockets on the bridge, then it will show the following dialog box in Figure 63.

Copyright © Bridgetek Pte Ltd


Problem Occurred

'Launching New_configuration (1)' has encountered a problem.

Error in final launch sequence:

Failed to execute MI command:
-target-select remote localhost:9998
Error message from debugger back end:
localhost:9998: Connection timed out.

```
Error in final launch sequence:

    Failed to execute MI command:
-target-select remote localhost:9998
Error message from debugger back end:
localhost:9998: Connection timed out.
Failed to execute MI command:
-target-select remote localhost:9998
Error message from debugger back end:
localhost:9998: Connection timed out.
localhost:9998: Connection timed out.
```

Figure 63 GDB timeout

7.3 Build errors due to anti-virus software protection

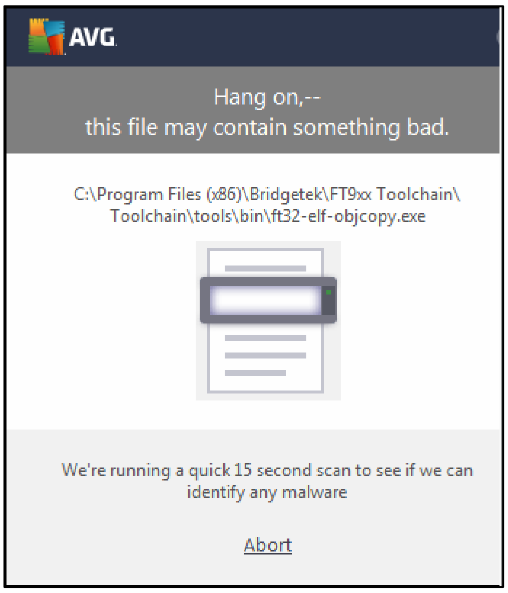
When building in Eclipse, the build process will sometimes stop or get delayed by messages similar to this.

*make: *** Access is denied.*

. Stop.

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

There may also be a graphical warning from a virus scanner as shown below.



Hang on,--
this file may contain something bad.

C:\Program Files (x86)\Bridgetek\FT9xx Toolchain\Toolchain\tools\bin\ft32-elf-objcopy.exe

We're running a quick 15 second scan to see if we can identify any malware

Figure 64 AVG warning during building in FT9XX Toolchain

This is a known issue with some antivirus software, such as AVG.

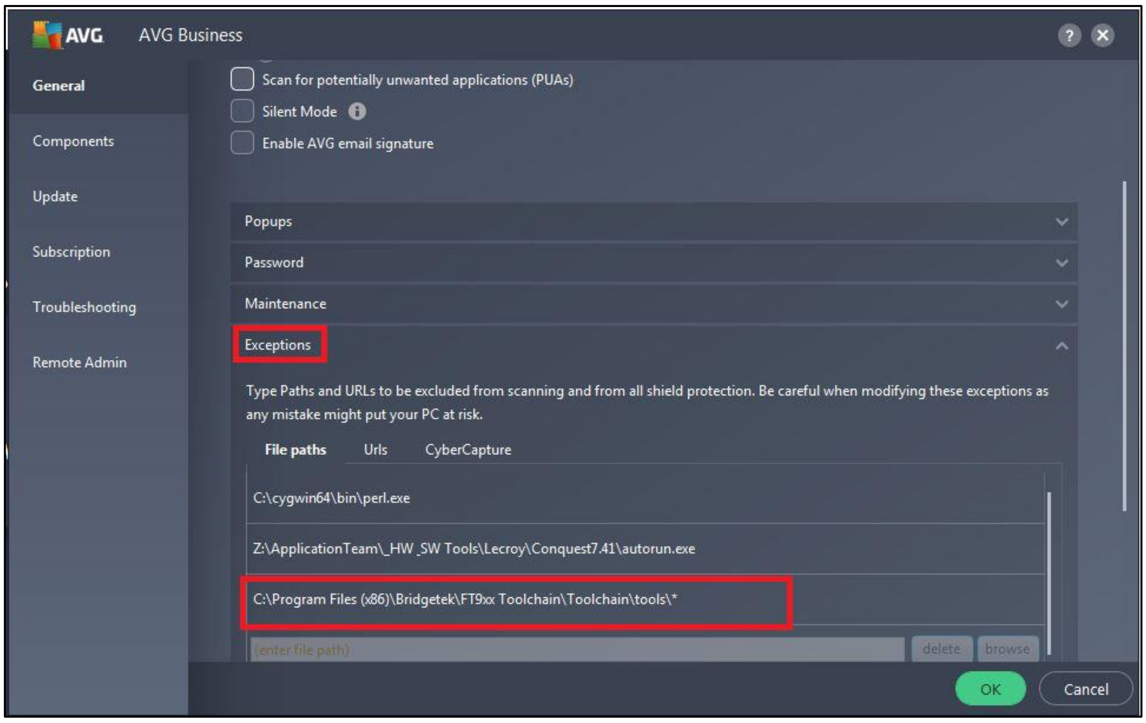
Workarounds include (depending on the antivirus software):

- Temporarily disable the antivirus software.
- Add exceptions to every "exe/bat" under the FT9XX Toolchain folder.

Copyright © Bridgetek Pte Ltd

Regarding adding exceptions in AVG Business, refer to the following link –

https://support.avg.com/SupportArticleView?l=en&urlname=Excluding-Files-from-Scanning-in-AVG-Business-Products



Scan for potentially unwanted applications (PUAs)
Silent Mode
Enable AVG email signature

Type Paths and URLs to be excluded from scanning and from all shield protection. Be careful when modifying these exceptions as any mistake might put your PC at risk.

C:\cygwin64\bin\perl.exe
Z:\ApplicationTeam_HW_SW Tools\Lecroy\Conquest7.41\autorun.exe
C:\Program Files (x86)\Bridgetek\FT9xx Toolchain\Toolchain\tools\*

Figure 65 Adding Exception in AVG for FT9XX Toolchain Executables

8 Contact Information

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

Distributor and Sales Representatives

Please visit the Sales Network 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 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, 1 Tai Seng Avenue, Tower A, #03-05, Singapore 536464. Singapore Registered Company Number: 201542387H.

Copyright © Bridgetek Pte Ltd

Appendix A – References

Document References

https://brtchip.com/ft9xx-toolchain/

AN_360 FT9XX Example Applications

Acronyms and Abbreviations

Terms	Description
CMD	Command-line interface
DLL	Dynamic-link Library
DLOG	Data Log (Project)
GAS	GNU Assembler
GCC	GNU Compiler Collection
GDB	GNU Project Debugger
GNU	GNU (Gnu's Not Unix) Operating System
GUI	Graphical User Interface
IDE	Integrated Development Environment
JDK	Java Development Kit
JRE	Java Runtime Environment
MCU	Microcontroller Unit
PATH	PATH Environment Variable
TCP	Transmission Control Protocol

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title:	AN_325 FT9XX Toolchain Installation Guide
Document Reference No.:	BRT_000116
Clearance No.:	BRT#074
Product Page:	FT9XX Toolchain
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial release	02-05-2014
1.01	Expanded screenshots of Installation Wizard in Section 2	21-08-2015
1.02	Updated Version for Toolchain 2.1.0	22-02-2016
1.03	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) Debugger related information moved to section 4.4. from Troubleshooting Updated screenshots for programmer	19-09-2016
1.04	Updated release Migration of the product from FTDI to Bridgetek name – logo changed, copyright changed, contact information changed	08-03-2017
1.05	The toolchain references have been updated from FT90X to FT9XX	23-03-2017
1.06	Updated the document for 'Seamless Debug' eclipse plugin; Section1 and Section 3.3 for Seamless Debug; Added section 6.3.4 on C++ compilation	22-01-2018
1.07	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.	14-11-2018
1.08	Updated for v2.6.0 Release.	03-05-2022
1.09	Updated for v2.7.0 Release. Updated Company address.	29-08-2023
2.0	Updated for v2.7.6 Release	16-10-2023

Copyright © Bridgetek Pte Ltd