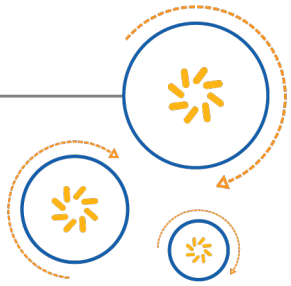
Qualcomm Technologies International, Ltd.



xIDE

User Guide

80-CT405-1 Rev. AM

October 27, 2017

Revision history

Revision	Date	Description
a	MAY 2005	Initial release. Alternative document number CS-00101500-UG.
b	JAN 2006	Minor updates to reflect minor software changes (blab-ug-002Pb).
3	DEC 2007	Prerequisites updated and formatted to new CSR™ style guidelines. (CSR reference: CS-101500-UGP3).
4	FEB 2008	Updated for Qualcomm® BlueLab™ v4.0 release and formatted to new CSR style guidelines.
5	OCT 2008	Updated for BlueLab v4.1. Graphics updated to Windows XP.
6	APR 2010	Correction to PS Key name, ONCHIP_HCI_CLIENT Chapter 5. Updated for 2010 SDK and remove BlueLab references.
7	MAR 2011	Updated to latest style guidelines
8	JAN 2012	Updated to the latest CSR style guidelines
9	JUN 2013	Screenshots on pages 10 and 21 updated
10	DEC 2013	Updated to latest CSR style
11	FEB 2016	Updated to conform to QTI standards; No technical content was changed in this document revision
AM	OCT 2017	Added to the Content Management System. DRN updated to use Agile number. No change to technical content.

Contents

1 xIDE overview

The QTIL Integrated Development Environment (xIDE) allows software engineers to build and configure the application projects provided with application specific SDKs or to independently develop applications to run on Qualcomm® BlueCore™ technology ICs.

It supports the development and debugging of both Virtual Machine (VM) and Native applications written in ANSI C for all BlueCore variants and Digital Signal Processor (DSP) code written in assembler code for BlueCore Multimedia ICs.

Code is written in the text editor and when complete, built and compiled along with the BlueCore firmware supplied as part of an SDK.

When compiled the resultant machine code can be downloaded to, and run on, a BlueCore hardware development platform such as CaSiRa or the multimedia development boards available separately from QTIL.

The code can then be debugged on-chip using the facilities in xIDE.

Applications can be developed from the reference application code provided, using the example code and library functions supplied, to adapt and add functionality.

The application source code provided implements various Bluetooth Profiles. These Profiles can be used as part of the user's own applications.

Using profile libraries and example applications as a starting point for development, greatly reduces the effort required to produce working Bluetooth applications that correctly implement one or more required Bluetooth Profiles.

NOTE The profile libraries supplied, support all the mandatory features and most but not all optional features of a particular profile. Refer to the individual SDK Release Notes for details.

2 How to install xIDE

xIDE should be installed on a PC with a Line Printer Terminal (LPT) port running Windows XP. A USB port is an alternative to the LPT port if a USB-SPI converter is available.

QTIL recommends that at least 600 Mbytes of free disk space is available.

NOTE Each application built needs extra space.

A minimum of Windows Power User privileges is required to install the software correctly. If you are unsure of your current level of privileges, please contact your system administrator.

New installations can coexist with previous releases.

NOTE Spaces in folder names of the directory path are not supported. Therefore, you should not try to install the software in a directory that has spaces in its name or is contained within a folder that has spaces in its name.

For example, xIDE cannot be successfully installed in the **Program Files** directory.

QTIL recommends that any applications running on the PC are closed before installing the software.

1. Download the `.zip` file containing the SDK installer executable and documentation from the QTIL support website.
2. Double-click on `<SDK-Name>-X.X.exe` file to launch the Setup wizard, which guides you through the rest of the installation process.
3. Follow the on-screen instructions, clicking **Next** to continue.
   For a first time installation, QTIL recommends that the default settings are accepted.
4. When the VM and Digital Signal Processor (DSP) libraries have been built, the PC the Setup Wizard displays the final setup screen.
5. Click **Finish** to complete the installation.
   If the default option to install the Serial Peripheral Interface (SPI) device driver was accepted, restart the PC to complete the installation.

2.1 MP3 support in xIDE

The DSP libraries required to build the MP3 decoder or encoder must be obtained and installed separately, contact the support channel for details.

They contain MP3 technology that incorporates intellectual property owned by Thomson and/or Fraunhofer Gesellschaft.

Supply of this product does not convey a license under the relevant intellectual property of Thomson and/or Fraunhofer Gesellschaft nor imply any right to use this product in any finished end user or

ready-to-use final product. An independent license for such use is required. For details, visit http://www.mp3licensing.com.

2.1.1 AAC+ support in xIDE

The AAC+ libraries required to build the AAC+ decoder must be obtained and installed separately, contact the support channel for details.

They contain AAC technology that incorporates intellectual property owned by numerous third parties. Supply of this product does not convey a license under the relevant intellectual property of those third parties nor imply any right to use this product in any finished end user or ready-to-use final product. An independent license for such use is required. For details, visit http://www.vialicensing.com.

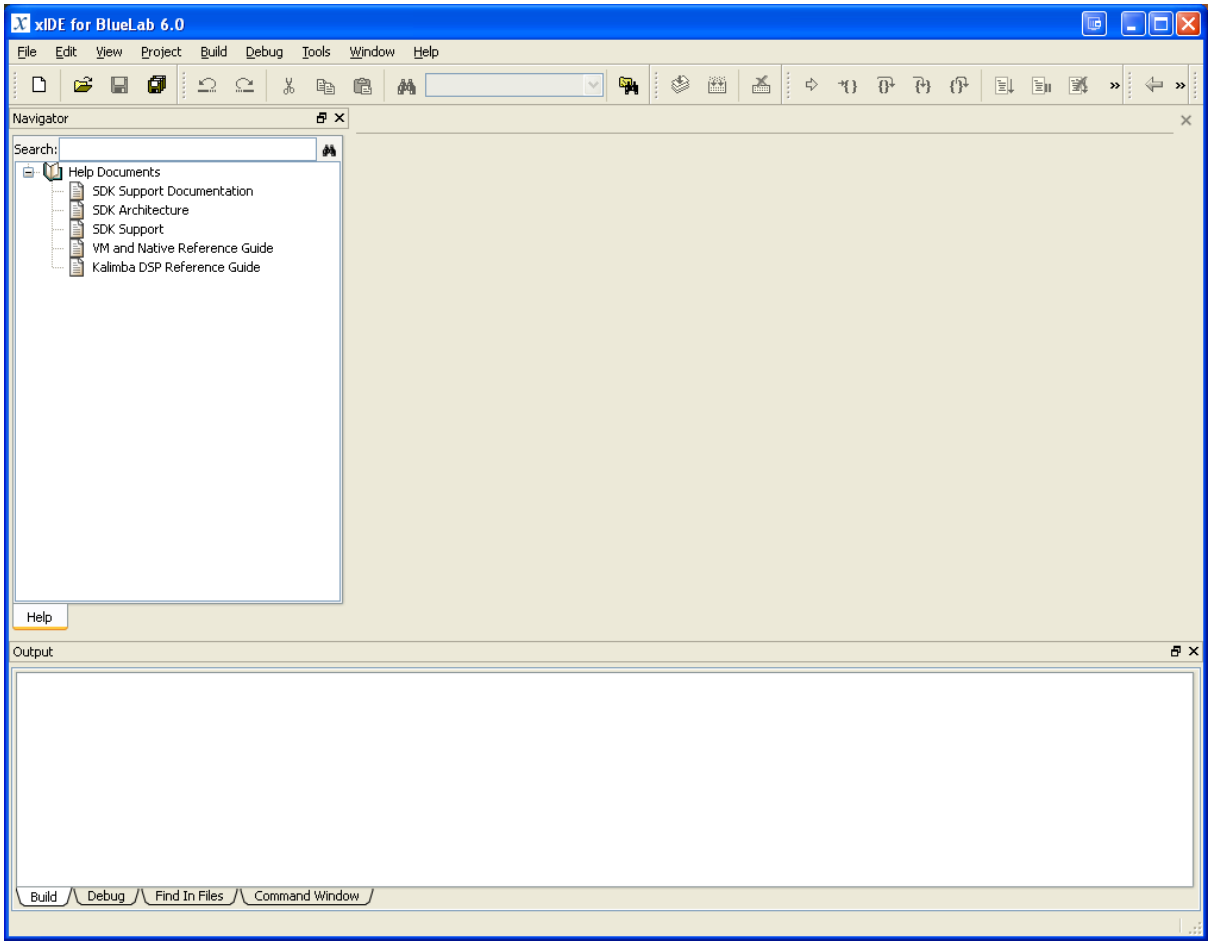
3 To test a completed xIDE installation

Connect a suitable hardware development platform (e.g. CaSiRa module) with an appropriate BlueCore IC to your PC using a SPI cable.

NOTE The documentation accompanying the Hardware development platform gives advice on connecting the unit to your PC.

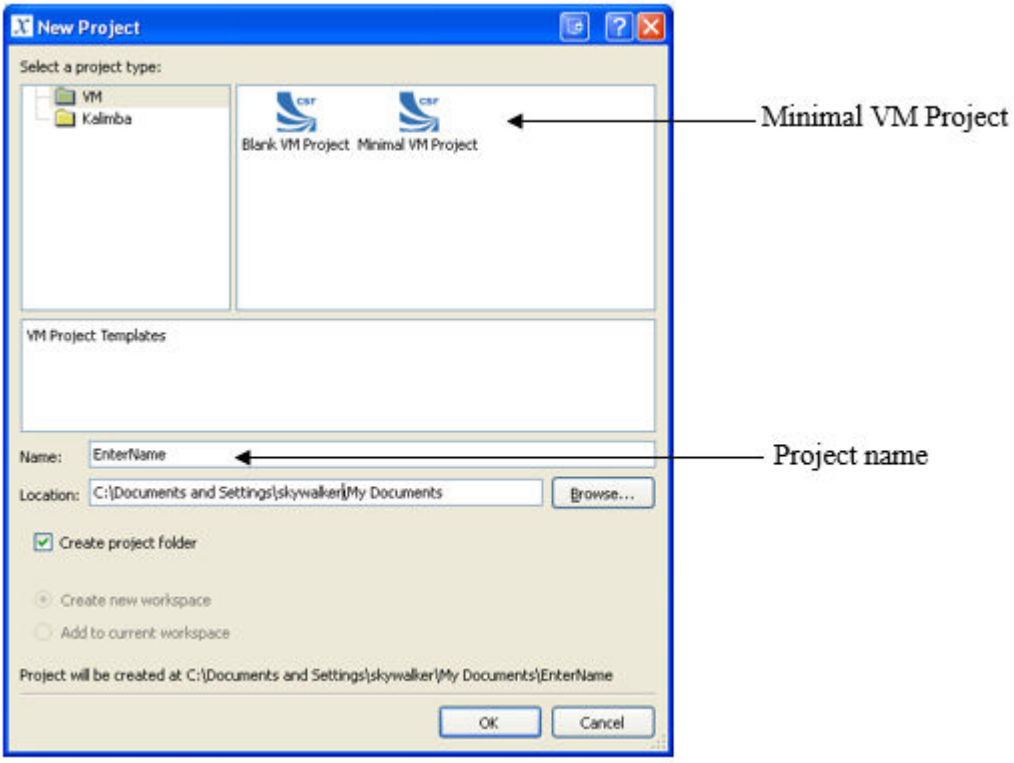
QTIL strongly recommends that a golden image is created to preserve the original chip settings. The required backup procedure is detailed in BlueCore firmware backup and restore procedure.

1. Launch xIDE by double-clicking on the shortcut icon on your desktop, the icon in the Quick Launch bar or from the Windows' **Start** Menu.
   The xIDE application window opens:



To confirm that the installation was successful and the application is working correctly create and run the simple Hello world program supplied. To do this:

2. Select **New** from the **Project** menu.
   The **New Project** window appears:



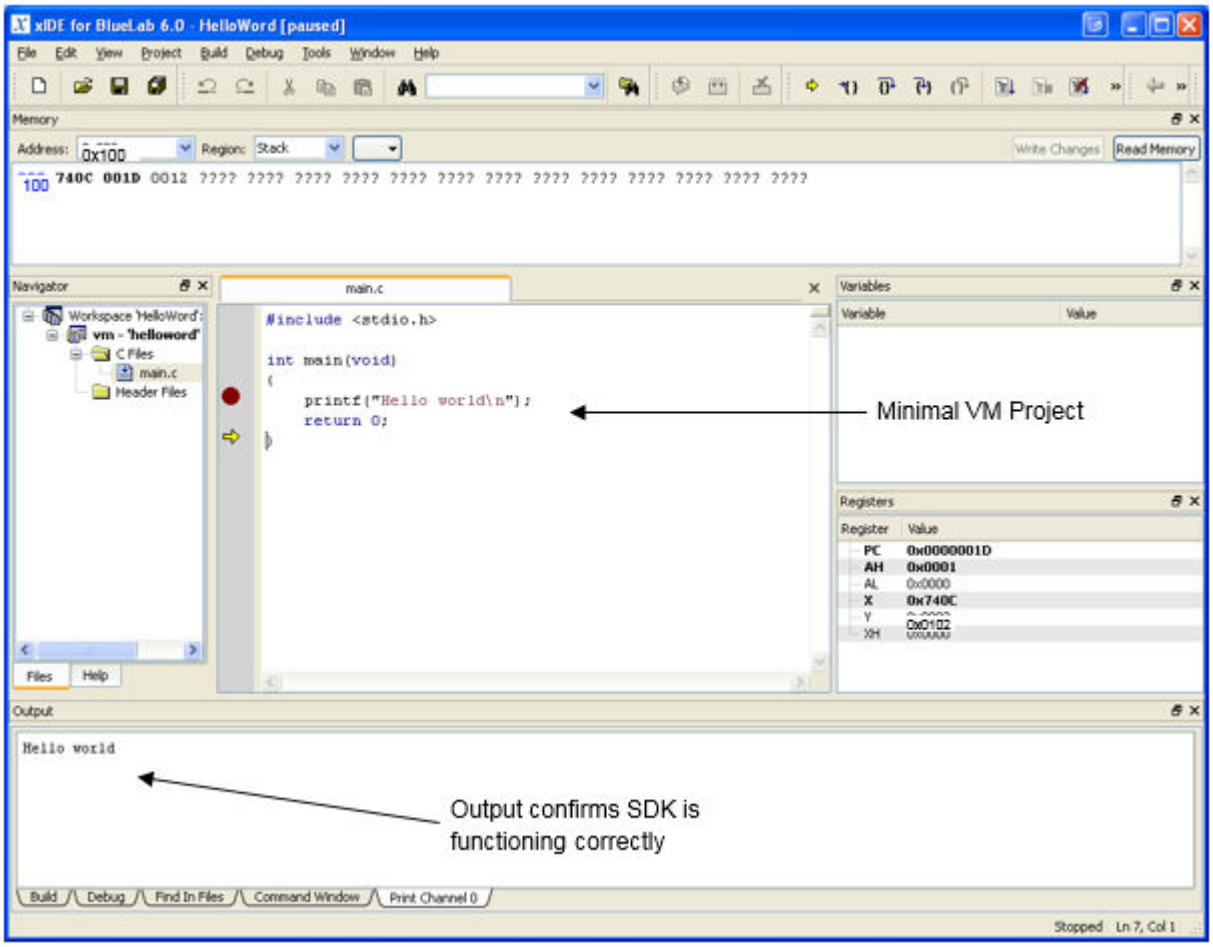
3. Select the **Minimal VM Project** and give the project a name, for example, hello.

   NOTE xIDE does not accept Project names if they contain a space.

4. Click **OK**.
   The project is loaded into xIDE.

5. Click on the **C Files** folder and select `main.c` to display the code in the Text Editor workspace.

6. Select **Run** from the **Debug** menu (or press the **F5** key).
   The program is downloaded to the BlueCore IC, when this process is complete.
   Click on the **Print Channel 0** tab, to view the output:



The output `Hello world` confirms that the software has been installed and is working correctly.

4 BlueCore firmware backup and restore procedure

An image of the firmware on a BlueCore IC can be backed up and/or restored at any time.

QTIL strongly recommend that a backup is performed to save the original firmware configuration prior to downloading any development code.

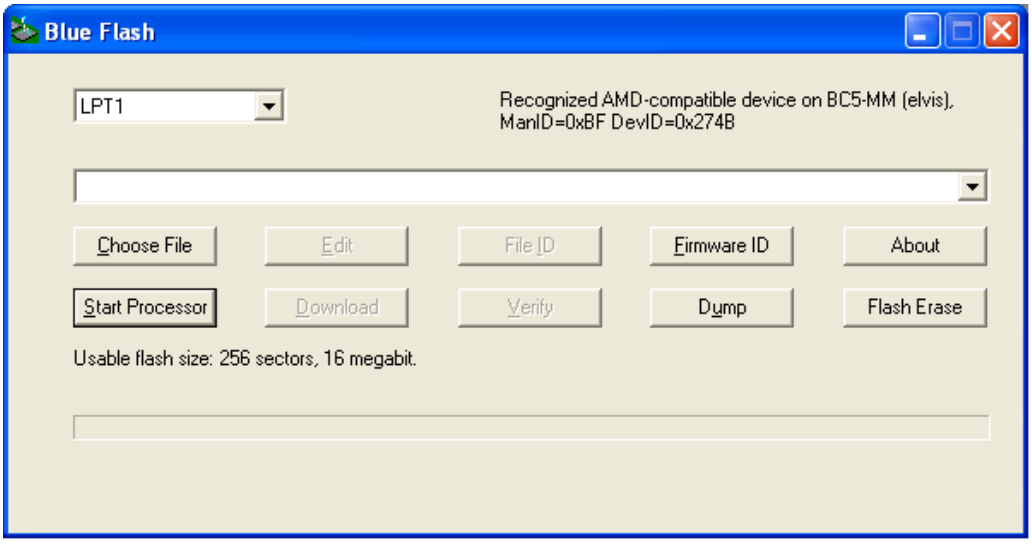
The backup and restore procedures are carried out using the **BlueFlash** application supplied as part of the toolset.

To Run the BlueFlash application:

1. Locate `BlueFlash.exe` in the SDK installation directory using Windows explorer:

   `C:\<SDK-Name>\tools\bin\BlueFlash.exe`

2. Double-click on `BlueFlash.exe` to run the application.
   The **BlueFlash** application window opens:

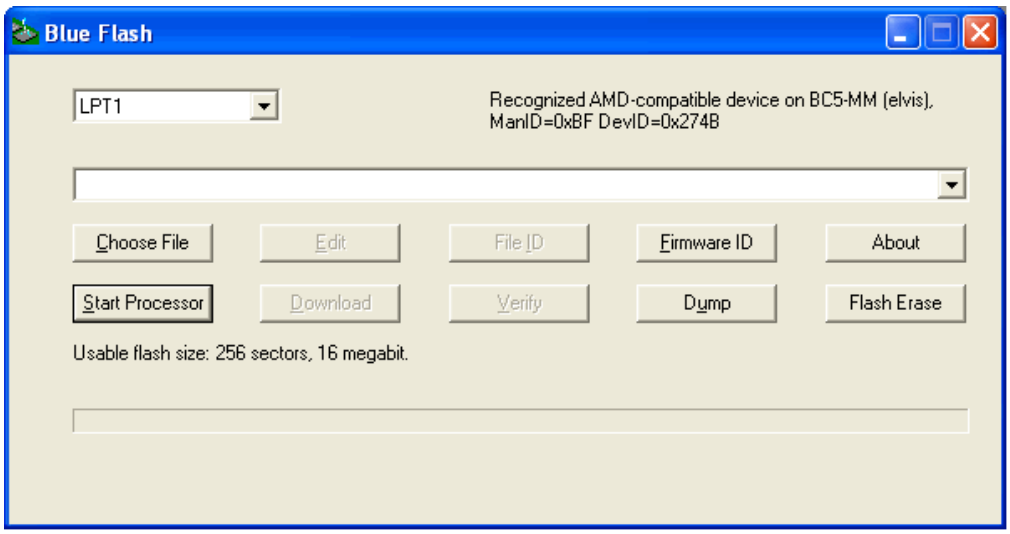


Backup and Restore procedures are described in BlueCore firmware backup procedure and BlueCore firmware restore procedure.

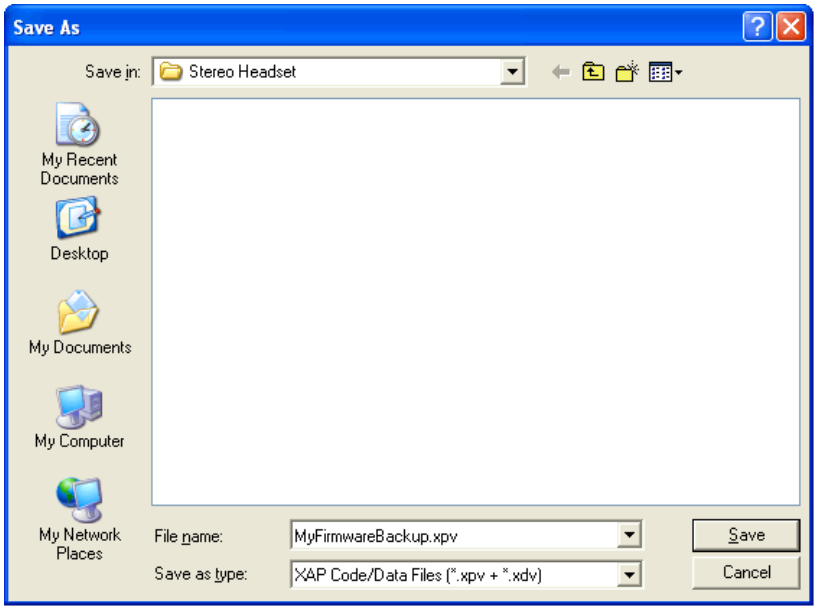
4.1 BlueCore firmware backup procedure

To backup an image of the current firmware and configuration settings using **BlueFlash**:

1. Click **Stop Processor**.
   The Processor is stopped and the **Dump** option becomes available:



2. Click **Dump**.
   A **Save As** window appears:



3. Browse to a location in which the backup is to be saved and enter a file name for the backup.
4. Click **Save**.
   BlueFlash creates and saves two files (a `.xpv` and a `.xdv` file) to the selected location.

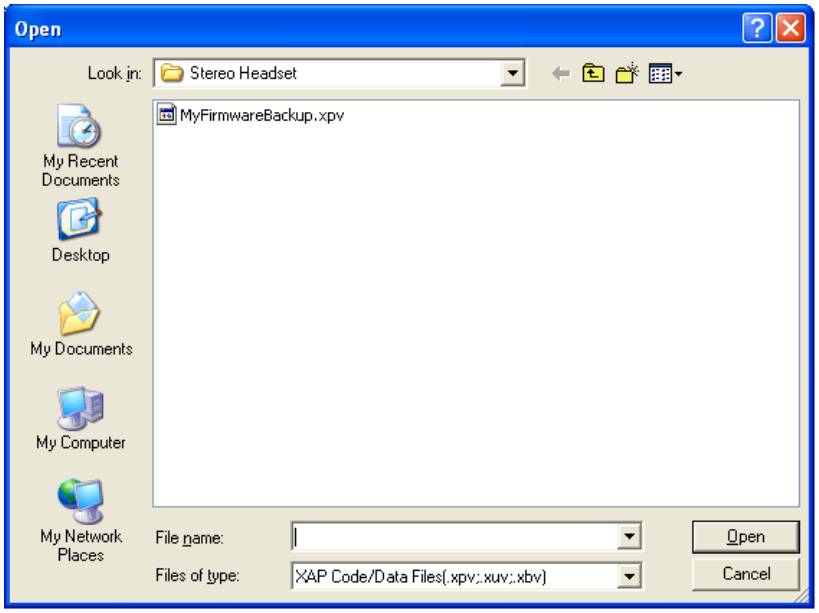
   NOTE This may take a few minutes, a progress bar displays the progress.

5. Click **Start Processor** to restart the chip.

4.2 BlueCore firmware restore procedure

To restore a backup image using **BlueFlash**:

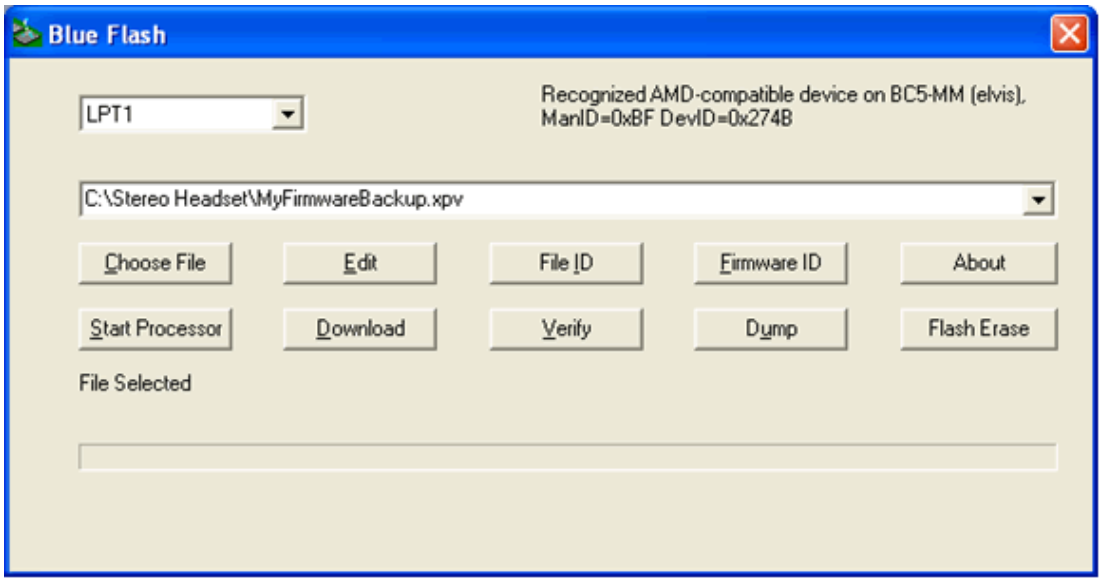
1. Click **Stop Processor**.
2. Click **Choose File**.
   An **Open** window appears:



3. Browse to the backup file location and select the required file.

NOTE The window only displays the `.xpv` file(s) saved at the selected location. **BlueFlash** automatically includes the `.xdv` file when downloading the image to the chip.

4. Click **Open**.
   The path to the file is added to the **BlueFlash** window and the **Download** option becomes available:



5. Click **Download**.
   BlueFlash downloads the backup image to the BlueCore IC. This may take a few minutes.
6. When the download has completed, click **Start Processor** to restart the IC.

5 Working with xIDE - overview

In most circumstances, it is envisioned that developers will make use of the reference applications provided as the basis for developing their own applications.

The reference applications provide basic functionality and conform to one or more relevant Bluetooth Profiles being implemented.

Adopting this approach greatly reduces the effort required to develop a final product application and allows software engineers to concentrate on developing the additional functionality and Man Machine Interface (MMI) features required for their particular product.

This chapter describes the procedure for loading a reference application as a project in xIDE and running the code on a hardware development platform. Specific details vary slightly depending on the application and hardware platform being used, further information is provided in the relevant product documentation.

NOTE Guidance on the use of Reference applications and examples is provided in readme files within the subfolders in the apps folder:

`C:\<SDK-Name>\apps\...`

where `C:\<SDK-Name>` is the install directory.

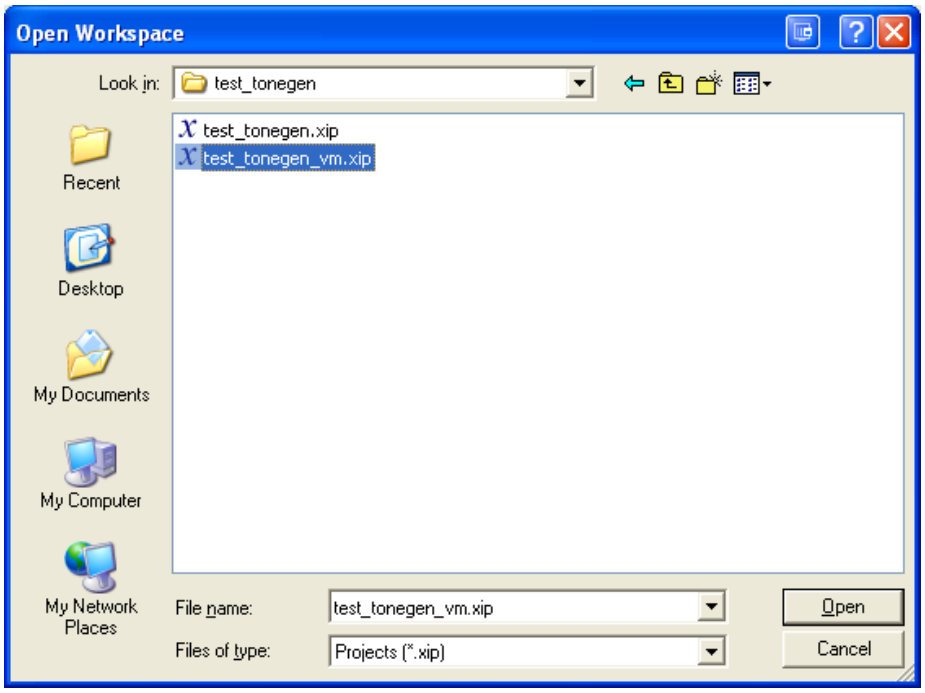
5.1 To build a supplied application project in xIDE

To open a project workspace for a supplied application:

1. Select **Open Workspace** in the xIDE **Project** menu.
   An Open workspace window appears.
2. Browse to the folder containing the example applications provided in the SDK.

   `C:\<SDK-Name>\apps\examples`
3. Open the required application folder.

4. Depending on the application chosen, one or more `.xip` project files are displayed, see the example below:



NOTE Where two or more project files are displayed, the additional projects are for DSP applications to be run on the DSP of BlueCore Multimedia ICs.

If DSP elements are required as part of the application these projects should be built in xIDE before attempting to build and finally run the application code.

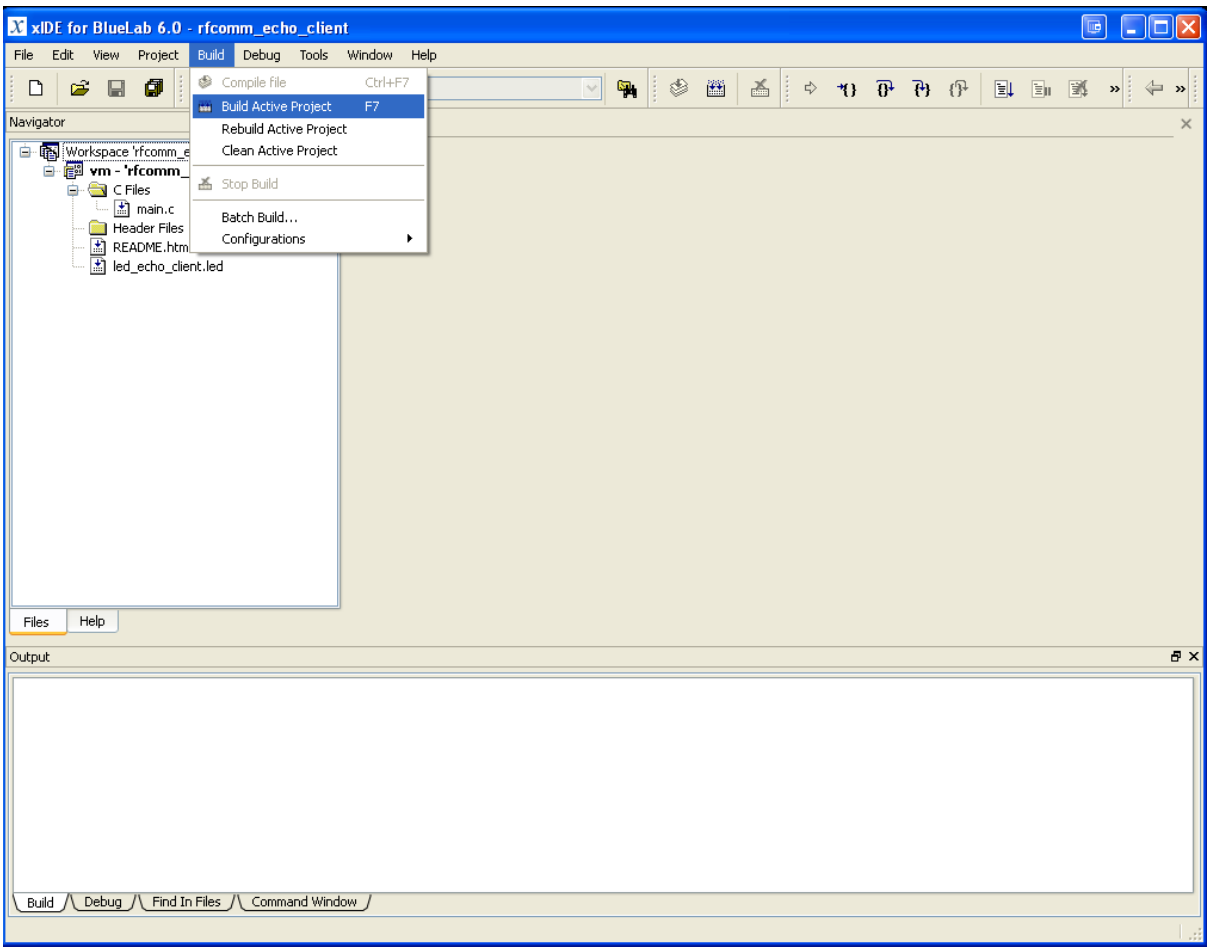
5.2 Build and run procedure in xIDE

Use the **Run** facility to build all the `projects` making up the application before downloading machine code to the BlueCore IC. Projects can be built individually:

Build individual project

1. Select the required project from the **File Navigation** window.
2. Right Click and **Set as Active Project.**
3. Select **Build** from the xIDE **Build** menu or press the **F7** key:

NOTE If the development hardware has been used for previous development work QTIL recommend that the factory settings are restored on the chip prior to downloading the new application code.



xIDE completes the build process for the loaded file.

4. If any other DSP `.xip` files that support the application code (e.g. for MP3 support) are present, build these before continuing.
5. Build the Virtual Machine application code in a similar manner.

   Run procedure

   To complete the compilation of the source code and to download the machine code produced to the BlueCore chip:
   With the main application file loaded in xIDE,

Postrequisites:

Run procedure

To complete the compilation of the source code and to download the machine code produced to the BlueCore chip:
With the main application file loaded in xIDE:

1. Select **Run** from the **Debug** menu or press the **F5** key.

The application should now be running on the BlueCore IC, see the relevant application documentation for further details.

5.3 To develope customized applications in xIDE

When the application has been downloaded and is working correctly, developers can begin to customize the source code and add features to meet the specific requirements of the final product.

NOTE If a backup image has not already been saved it may be advisable to create a backup image of the BlueCore IC before making further changes.

To work efficiently when developing an application it is important to become familiar with the library structure and functions provided. These are detailed in the library support documentation.

The example application code supplied is also a useful resource and can be found in:

`C:\<SDK-Name>\apps\examples`

5.3.1 To amend Project Properties in xIDE

When calling library functions the host Libraries must be listed in the **Project Properties**.

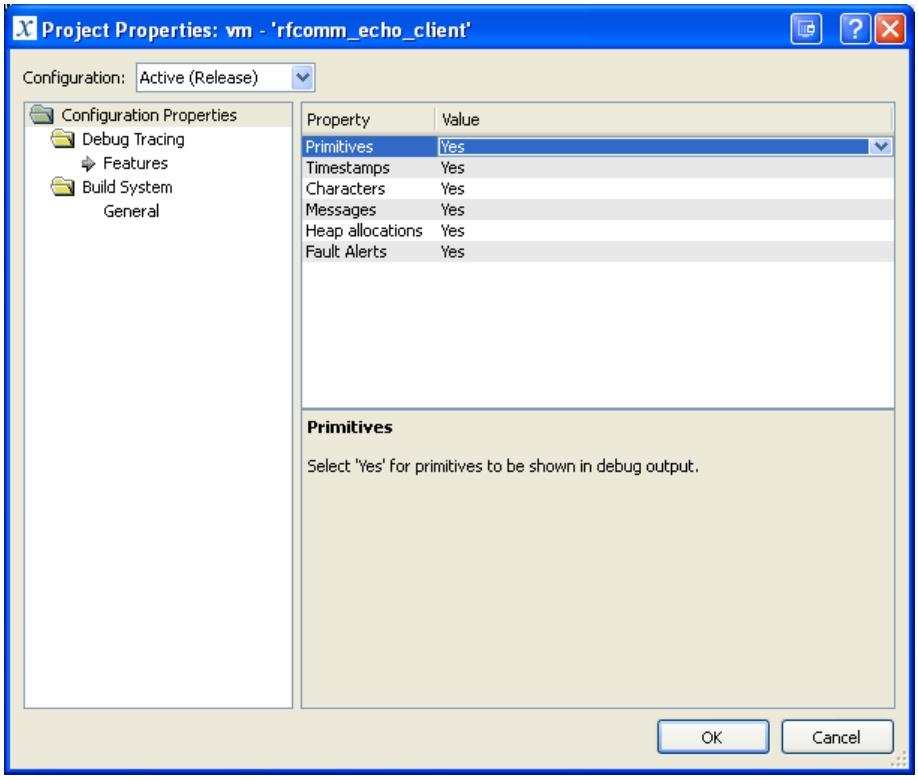
NOTE Libraries used by the reference application code along with the other project properties for the particular application are specified when the `.xip` file (i.e. the project file) is loaded to xIDE.

When function calls to other libraries are added to the application code the appropriate libraries must be added to the list of libraries specified in the **Project Properties**.

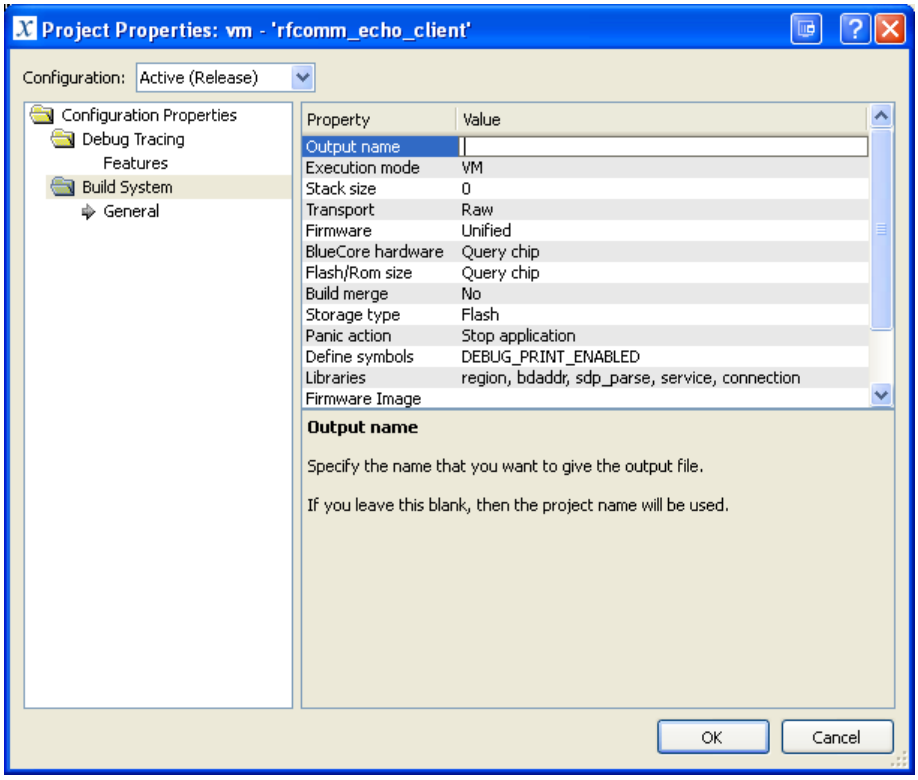
NOTE If more than one project exists in the workspace, changes to the **Project Properties** window only apply to the active project.

To amend xIDE Project Properties:

1. Select **Properties** from the **Project** menu.
   The **Project Properties** window appears:



2. Click on the **Build System** folder.
   The **Build System** properties are displayed:



3. Click on a row to activate the **Value** field for the **Property** you want to amend.
   The text below the list of properties provides tips relevant to the selected Property.
4. When the required properties have been amended, click **OK** to set the properties for the project.

6 Debugging applications in xIDE

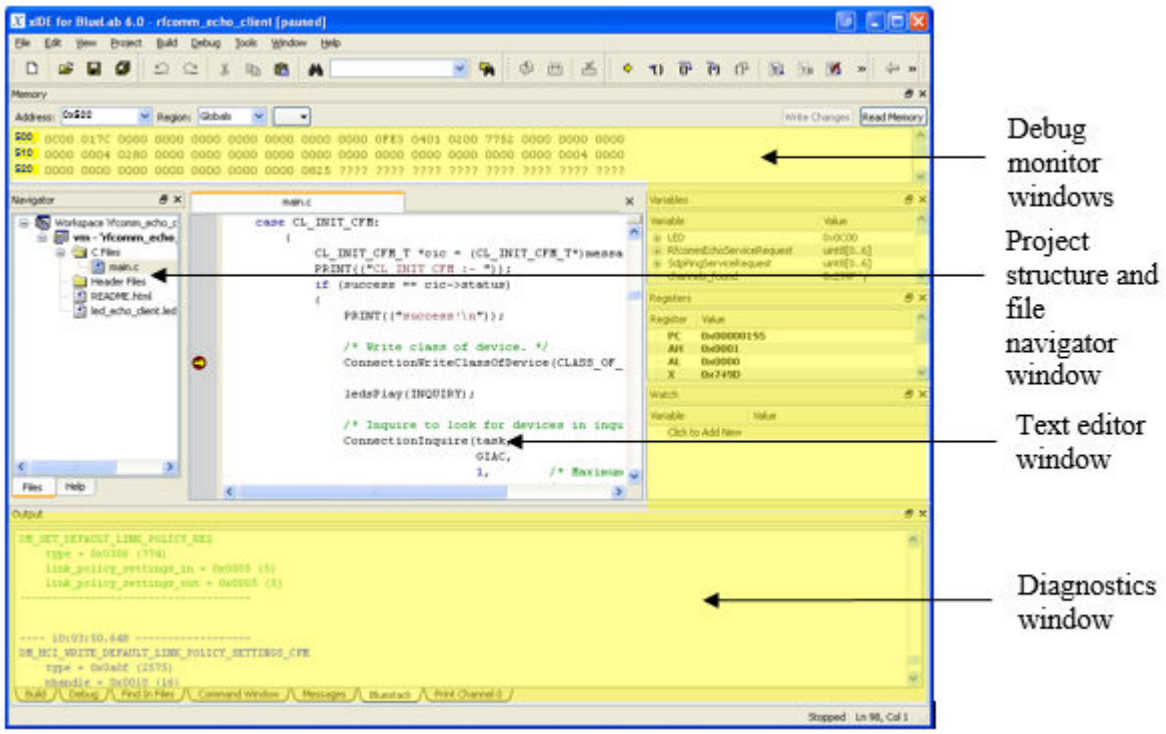
xIDE supports the debugging of the program code running on the VM of the BlueCore IC and the DSP code running on the on-chip DSP of BlueCore Multimedia chips.

The application is run on-chip, thus ensuring the debug environment matches the final execution environment of the product as closely as is possible.

xIDE provides a familiar debugging toolset that includes facilities required to efficiently debug programs running on a BlueCore IC.

While many of the facilities provided in xIDE are typical debugging tools, a few are more specific to an integrated implementation such as BlueCore, see Brief overview of debug facilities in xIDE.

6.1 Brief overview of debug facilities in xIDE



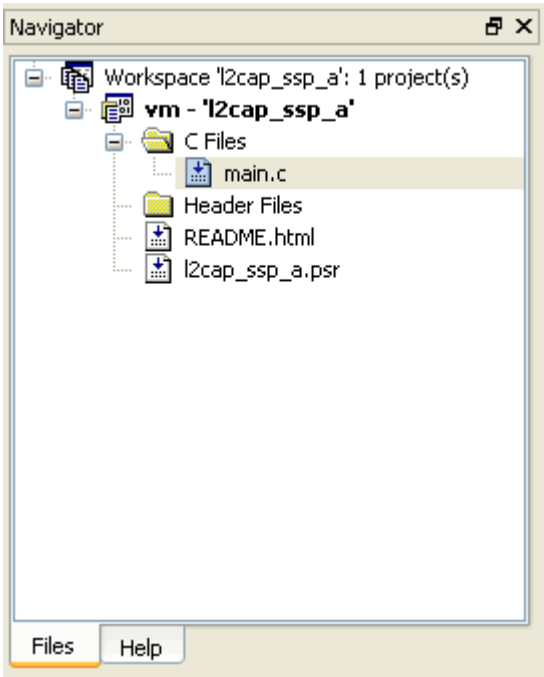
The xIDE debug view consists of four basic work areas:

- File navigation window
- Text editor window

- Debug monitor windows (highlighted in yellow)
- Diagnostics windows

6.1.1 xIDE file navigation window

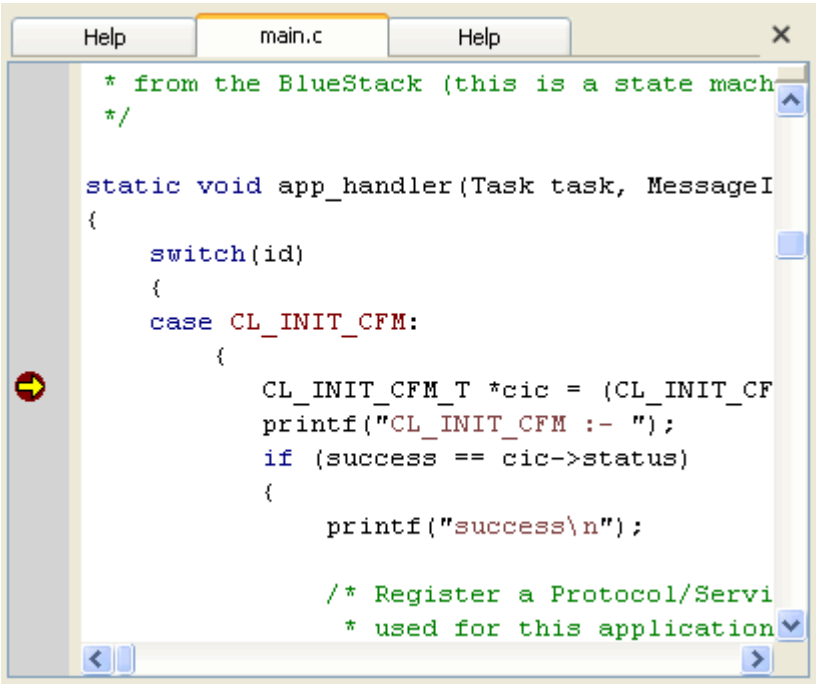
This area displays an explorer like view of the project workspace currently loaded in xIDE:



The file structure can be navigated and files opened in the text editor by double-clicking on a file.

A right-click shortcut menu can be opened for items listed in the **File Navigation** window, the menu options depend on the item selected.

6.1.2 xIDE text editor window



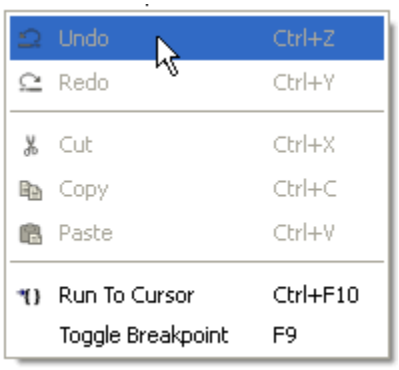
This area displays open project files and allows:

- Text editing
- Break points to be set

Tabs allow navigation between multiple files opened in the text editor.

NOTE An * displayed after the file name on a file tab (e.g. `main.c*`) indicates that the file has been amended locally and has not been saved.

A right-click in the text editor window displays a shortcut menu:



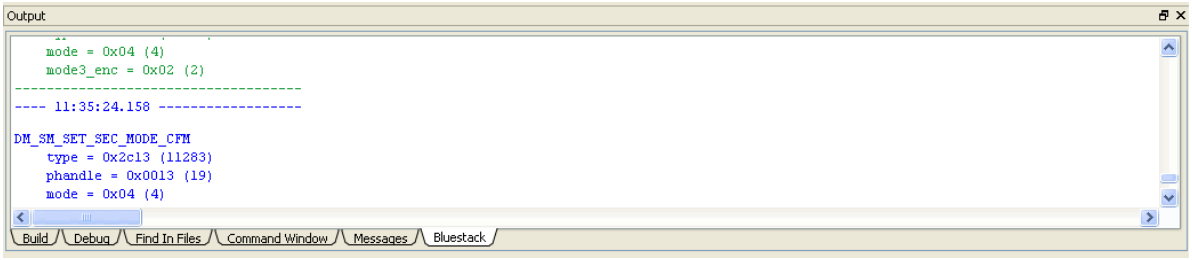
6.1.3 xIDE debug monitor windows

Developers can select to view various windows that monitor the BlueCore chip state:

- Memory: Displays current values of selected memory addresses
- Register: Displays current values of registers
- Variables: Displays current values of program variables
- Watch: Allows the user to view the current value of specific program variables

The views available can be toggled on and off from the **View /Debug Windows** menu list.

Diagnostics windows



This area displays various tabbed windows that display useful information when building and debugging code:

- Build: The **Build** tab displays progress information detailing the build process and status.
- Debug: Displays progress information showing the debug process and status.
- Find in Files: Displays the results of the **Find in Files** facility accessed from the **Edit** menu or toolbar.
- Command Window: This window can be used to evoke python scripts to extend xIDE.

NOTE Most developers need not concern themselves with this facility.

- Messages: Displays a time stamped list of messages passed between tasks in the VM.
- Bluestack: This tab displays the messages to and from the stack relating to the lower level radio interface.
  This can be useful in confirming that the code is resulting in the expected event messages being sent and received by the radio.
- Print Channel 0: This tab displays any print output generated by the program running on the chip.

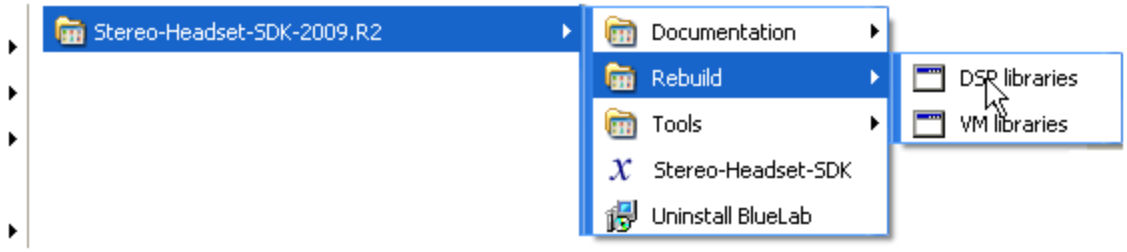
A right-click shortcut menu can be opened in each of the diagnostic windows, the menu options depend on the active tab.

NOTE The **Messages**, **Bluestack** and **Print Channel** tabs only appear when xIDE has information to display.

7 xIDE related frequently asked questions

1. *Can I customize the xIDE environment?*
   Yes. The **Options** menu item in the **Tools** menu allows you to select various options affecting the appearance and behavior of xIDE.
   The **View** menu offers a number of layout options allowing you to toggle the display to show your preferred debug windows and menu items.
   Windows can be reorganized by dragging and dropping within the main xIDE window or on the desktop to create separate displays.

2. *Can I rebuild the VM and DSP libraries?*
   Yes, you can rebuild the VM and DSP libraries from the Windows **Start** menu.
   Go to **Start\Programs\<SDK-Name>\Rebuild** and select the Libraries you wish to rebuild:



   NOTE You may need to build the libraries if the default option to build libraries was unchecked during the install process.

   Or
   If code has been added to or amended by the developer. However, QTIL do not recommend altering the supplied libraries.

3. *My application fails to initialize the Connection Library, why?*
   It may be that PSKEY_ONCHIP_HCI_CLIENT is incorrectly set. This key enables the upper Bluetooth layers on the device and should be set to `1`.
   Use the PSTool application described in question 5 to check that the PSKEY_ONCHIP_HCI_CLIENT is set to `0001`.

   NOTE The friendly name for this PS Key is **HCI traffic routed internally**.

   xIDE normally sets the value of this key automatically based on whether the application makes use of Bluestack.

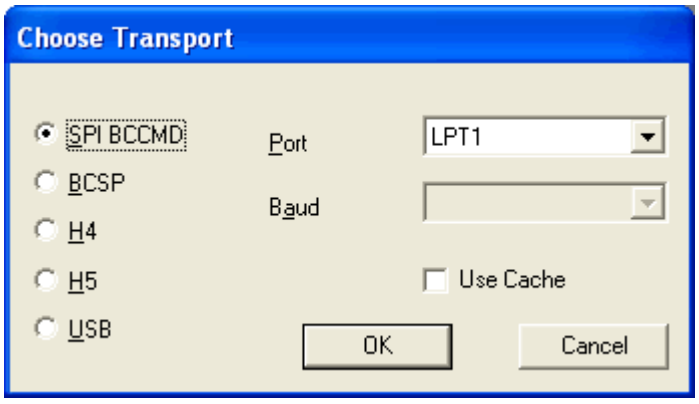
4. *I have debug print messages in my application. When I run my application within xIDE the application runs and I can view the debug messages. However, when I run the application on-chip, it fails to execute, why?*
   For the application to run with debug messages, the messages generated must be picked up either in xIDE or using the **vmSpy** application, otherwise the application will be stopped to avoid overflowing buffers.

Debug print messages must be disabled or removed before the application can be run exclusively on-chip.
If you have enabled debug printing using the print library and defining `DEBUG_PRINT_ENABLED` in the **Define symbols** field of the project properties remove it from the project properties and rebuild the application to disable the debug messages.

NOTE Use of VMSpy requires that the host transport is set to **BCSP**, **H4** or **USB** (as described in question 6) and that a suitable cable is used to provide the connection.

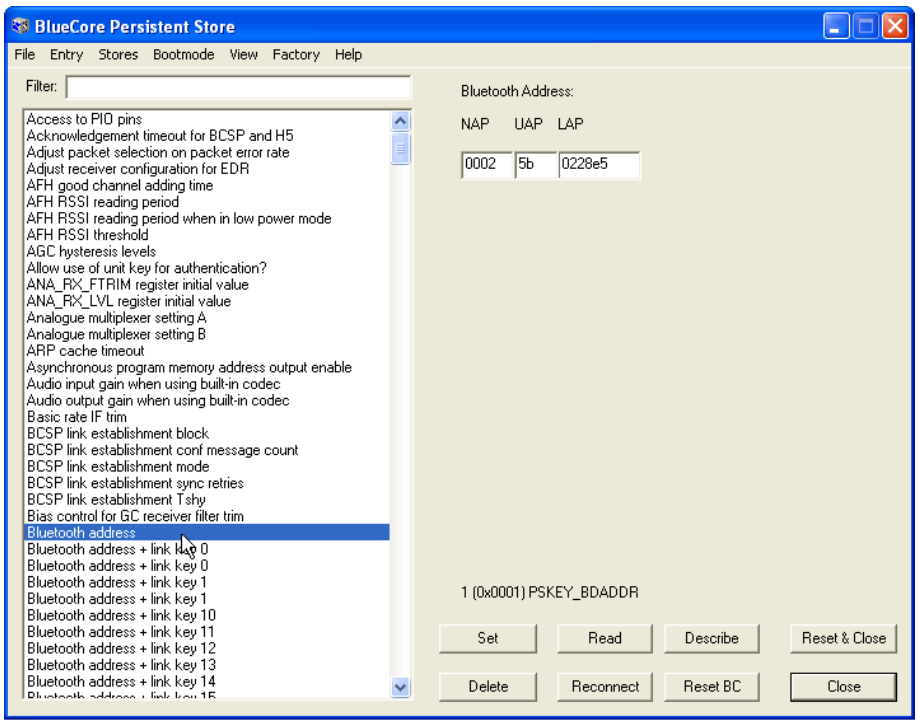
5. *How can I restore the chip's factory settings?*
If you created a backup of the chip in its original state, you can use the procedure described in Chapter 1 to restore the chip.
Alternatively, the factory settings can be restored using the PSTool application supplied with the SDK.
To restore factory default settings using PSTool:

   a. Open the **PSTool** application. The `PSTool.exe` can be found in the install directory. e.g. `C:\<SDK-Name>\tools\bin\PSTool.exe`.
   The **Choose Transport** window appears:



   b. Select **SPI BCCMD** and **LPT1** or **USB** as the transport settings and click **OK**.

   c. The **PSTool** application window opens:



d. Select **Merge** in the **File** menu.

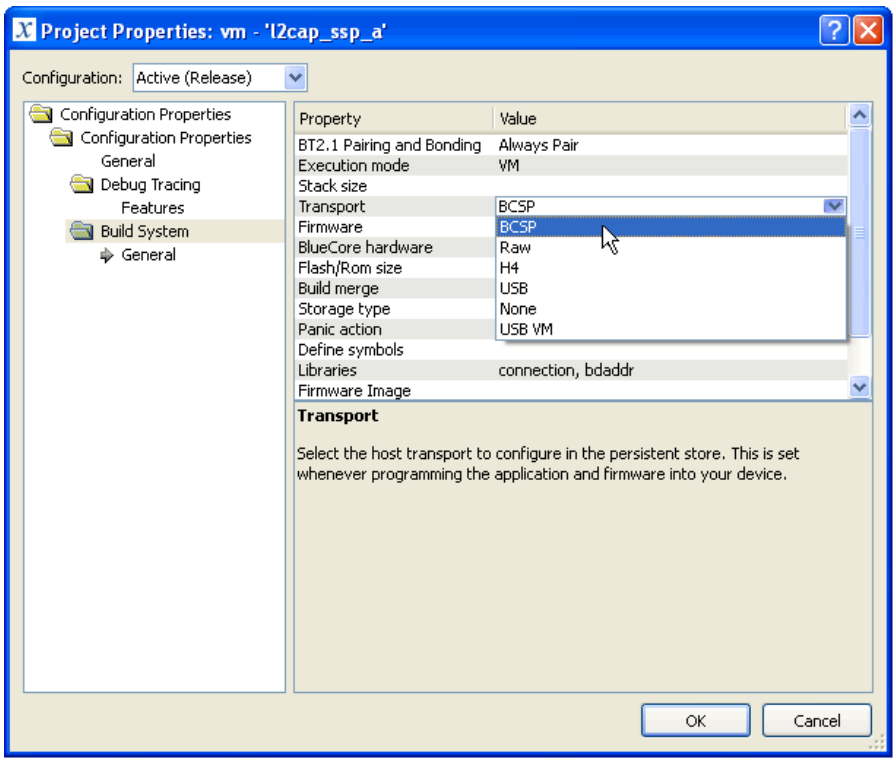
e. Browse to the location of the `force_defaults.psr` file. The file is located in the install directory:

`C:\<SDK-Name>\tools\force_defaults.psr.`

f. Select the file and click **Open**.
The file is run and the PS Keys, which if incorrectly set may result in unexpected chip behavior, are restored to their factory defaults. This may take a few minutes.

6. *I have modified the host transport using PSTool but every time I run my application in xIDE the host transport is modified, why?*
xIDE sets the host transport to the type specified in the **Project Properties** settings each time the application is executed.
To set the required **Transport** in the **Project Properties** window:

a. Select **Properties** in the **Project** menu. The **Project Properties** window opens:



b. Click on the **Build System** folder.

c. Click on **Transport** in the list of properties, to activate the **Value** field.

d. Select the required **Transport** from the drop-down list.

e. Click **OK** and then close the **Project Properties** window.

Terms and definitions

Term	Definition
BCSP	BlueCore Serial Protocol, a proprietary transport protocol
BlueCore	Group term for the QTIL range of Bluetooth wireless technology ICs
Bluetooth SIG	Bluetooth Special Interest Group
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
DSP	Digital Signal Processor: a microprocessor dedicated to real-time signal processing.
H4	UART-based HCI transport, described in section H4 of v1.0b of the Bluetooth Specification
IC	Integrated Circuit
LPT1	First Parallel Printer Port
MMI	Man Machine Interface
QTIL	Qualcomm Technologies International, Ltd.
raw	raw transport protocol where the VM has full control of the UART port
SDI	Software Development Interface
SDK	Software Development Kit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus protocol
usb_vm	Universal Serial Bus protocol where the VM has full control of the USB port
VM	Virtual Machine; environment in the BlueCore firmware for running application-specific code produced with CSR SDKs.