
TCAT Firmware Development Environment

Version 4.0.x

Installation Guide



CONTENTS

Overview	3
Features	3
Firmware Sources	4
New In This Release	5
Checklist	7
System Requirements	8
Required	8
Recommended	8
Planning your Installation	9
Existing Cygwin installation	9
Login User Name	9
Administrator credentials	9
Windows 7	Error! Bookmark not defined.
Installable Components	10
Cygwin Environment	10
Tools	11
Installation	13
Upgrading a 3.x Firmware SDK to 4.x	13
Network Environments and Domain Users	19
Uninstalling	20
Adding Cygwin Modules	20
What's happening behind the scenes?	24
Installer	24
Launch the Installer	24
Setup Script	24
Install Cygwin	25
Install GNUARM	25
Install GNU ARM-EABI	25
Updated Insight debug GUI	25
Install OpenOCD Server	25
Install Documents	25
Miscellaneous	25
Cleanup	26
Uninstaller	26
Launch Uninstall	26
Cygwin Root Directory	26
About Non-Native Serial ports	28
Component Origins	28

Overview

The TCAT Firmware Development Environment is a complete, version-stabilized development system for creating firmware applications for the DICEII STD, DICE-JR and DICE-MINI family.

This document describes the installation of the TCAT Firmware Development Tools, and Documentation. Firmware Source Code is not included with the Development tools. Sources are available on the TCAT Subversion repository.

Once you have installed the Environment, please see the *TCAT Firmware Development Environment User Guide* (online in your Subversion tag) for the next steps.

Features

- ✓ Version Stabilized distribution of Toolchain Components
- ✓ Open Source, Royalty-Free RTOS
- ✓ Windows Cross-development using Cygwin and GNUARM (ARM_ELF) and/or Gnu Tools (ARM-EABI)
- ✓ Standard ARM Compiler, Linker, Debugger
- ✓ Serial and JTAG debugging support
 - JTAG interface is not required for debugging, however it is required for bring-up of new boards with empty flash, and is recommended for devices which support MIDI.
- ✓ Support
 - Ongoing development of new features provided in future updates
 - Forums
 - Technical Support, Updates and Documentation

Firmware Sources

All of the sources needed for developing DICE firmware and Host interfacing software are available online, including:

- ✓ Standard RedBoot BSP
 - Supports DICE EVM's
 - Easily ported to custom hardware
 - FLASH file system provides easy image management
- ✓ Self-documenting Code using doxygen
- ✓ Full featured 1394 Software Stack
- ✓ Intuitive and flexible API's
 - Access to all DICEII/DICE-JR/DICE-MINI chip functions
 - RTOS abstraction layer
 - 1394
 - Asynchronous access
 - Isochronous stream configuration and control
 - Bus Management functions
 - Minimal code required for implementing new applications
- ✓ Supports the Classic DICEII EVM, EVM002 board and EVM003 board with all DICE variants
- ✓ Several selectable streaming configurations
 - Examples and starting points for custom implementations
- ✓ AV/C Support
- ✓ Comprehensive CLI
 - Command line versions of all major API's

Host interfacing code is also included in Subversion checkouts

- ✓ Platform Abstraction Layer (PAL)
 - Cross-platform GUI can be developed using API's such as Juce, WxWidgets, etc.
- ✓ Control Panel Examples
- ✓ Utilities
 - Dice test application, Device Inspector, etc.

New In This Release

Updated Cygwin

The included Cygwin installer and package download contains an updated setup installer and package set, with a few additional packages such as subversion, python, etc.

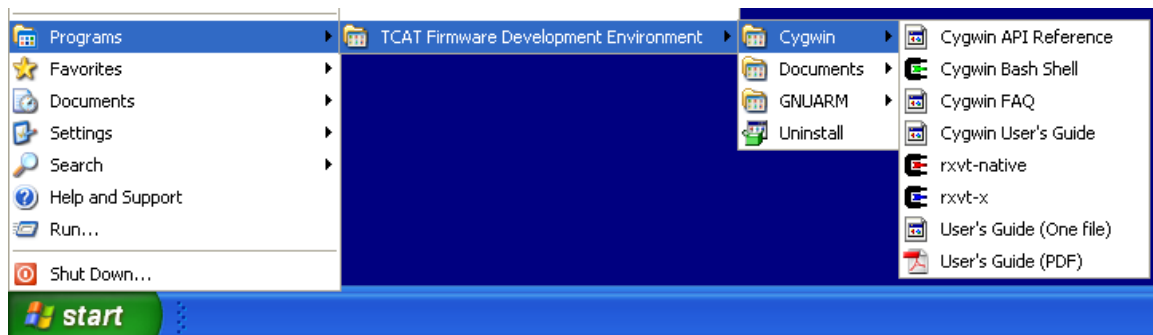
This Firmware SDK installer can be run over an existing installation, and your /home directories will be backed up before being overwritten.

The sources for all DICE product development are available online in a Subversion repository. These sources are updated regularly and are the most recent that TCAT makes available to developers.

Please contact TCAT for access to the server. In addition to firmware sources, other resources are also provided on this server, such as the latest documentation, driver binaries and sources for developing host applications.

rxvt

As before, an Xterm-like window can be used with bash using the included Cygwin rxvt package. This rxvt now creates separate blue and red icons which invoke bash in the xterm window. The green icon runs bash in a plain window.



The SDK setup program also creates a .inputrc and .Xdefaults file for you which allows ctrl+v pasting into bash windows, and extends the history buffer to 20000 lines.

EABI compiler

The Gnu Tools EABI compiler is now included which allows building with eCos 3.0 firmware sources. This is now the default compiler environment. The binaries are built from sources provided by eCosCentric. See the section *Component Origins* to find out where to look for source updates.

Developers can still use the GNUARM tools with older firmware trees (i.e. eCos 2.0-based) and new firmware trees (eCos 3.x-based). The tools are invoked depending on the GNUARM and GNUARMEABI environment variables in your **.bashrc** file.

Your **.bashrc** file should have the following definitions:

```
# for arm-elf toolchain
export GNUARM=/gnuarm

# for arm-eabi toolchain
export GNUARMEABI=/gnutools/arm-eabi

PATH=${PATH}:/gnuarm/bin:/gnutools/arm-eabi/bin
export PATH
```

Emacs

Emacs is no longer included; however the installer will still prepare your bash environment for use with emacs if you choose to install it later. Developers who wish to use emacs can download it separately from gnu.org (URL below) or one of its mirrors.

<http://ftp.gnu.org/gnu/emacs/windows/>

OpenOCD server

A recent version (0.4.0 at the time of this SDK release) of the OpenOCD JTAG server is included, which supports the new configuration file syntax which is used in the TCAT sources. *Developers will need to install the bus drivers that are compatible with their OpenOCD-compatible JTAG hardware interface.*

OpenOCD has frequent updates, so developers should be able to use more recent versions as long as they are 0.4.0 or newer. However, the syntax of the OpenOCD command language may change from time to time so you may need to adapt your configuration scripts accordingly. See the section *Component Origins* to find out where to look for source updates.

Checklist

- ☐ Obtain a Subversion repository password and Tag URL from TCAT
- ☐ Determine your system configuration
 - 1) RAM and Disk Drive requirements
 - 2) Administrator account
 - 3) Whether or not the computer is in a Windows Domain
 - 4) Whether or not an existing Cygwin non-text mode installation is installed
- ☐ Review the *Planning Your Installation* section below
- ☐ Follow the steps outlined in the *Installation* section below
- ☐ Configure your *bash* shell
- ☐ Checkout your Subversion Tag

See the *TCAT Firmware Development Environment User Guide* in your subversion tag for the following:

- ☐ Build the initial projects using the install shell script and evaluate as needed
- ☐ Review the *Development Steps Detailed Checklist* document

<https://dev.tctechnologies.tc/tcat/tags/release/public/latest/docs/dice/>

- ☐ Create a new project from an appropriate template and create your custom application

System Requirements

Required

Windows XP 32-bit, Windows Vista 32-bit or Windows 7 32-bit
64-bit Windows operating systems are not supported
1 GHz or greater Pentium Class CPU
750MB free hard drive space
(1) Serial Port
1 GB RAM
CD-ROM Drive

Recommended

Windows XP 32-bit, Windows Vista 32-bit or Windows 7 32-bit
(2) Serial ports, at least one native serial port
See section 11 below *About Non-Native Serial Ports* for more information.
(1) OHCI 1394 port (if using 1394 drivers on the same computer)

Planning your Installation

Existing Cygwin installation

Note If you are already using Cygwin on the target workstation for other purposes, note that the Installer will install the Cygwin environment in **text** (DOS) mode. If you are relying on an environment for other development work that assumes **binary** mode mounts and file access, then consider using a different computer for this install. Use the bash **mount** command to find out which mode is currently installed.

This distribution is a predefined Cygwin distribution that may differ from your existing Cygwin environment in terms of modules and versions. If you install this in an existing Cygwin directory that was not created with this Installer, you may have unexpected results.

When installing over an existing Cygwin, the /home directory and etc/passwd and /etc/group files will be backed up. All other files and directories may be overwritten.

If you are using a computer that is not in a Windows Domain and your login account has no spaces in the name, and has Administrator credentials, simply run the Setup program and follow the defaults and instructions all the way through.

Otherwise, there are a few things to do to prepare for a smooth installation.

Login User Name

Unix shells, including those used in Cygwin, use spaces as a delimiter. Therefore, for ease of use and to avoid any problems with existing shell scripts in the Cygwin distribution, make sure your login name doesn't have a space in it. If it does, you can rename the user in the *Windows User Manager* before you install, or if you realize the problem after you've installed Cygwin, you can rename the user in the Windows User Manager GUI and then run **mkpasswd** in a bash shell.

Non-domain users will enter: **\$ mkpasswd -l > /etc/passwd**

Domain users will enter: **\$ mkpasswd -l -d > /etc/passwd**

Note that the command for domain users may take a long time in some cases, so be patient if it seems unresponsive.

Administrator credentials

Since included component installers will make use of the Windows Registry, the Installer must run with Administrator privileges. Windows XP users should use "Run As..." and use an administrator credentials, and Windows Vista users should use "Run as administrator."

Installable Components

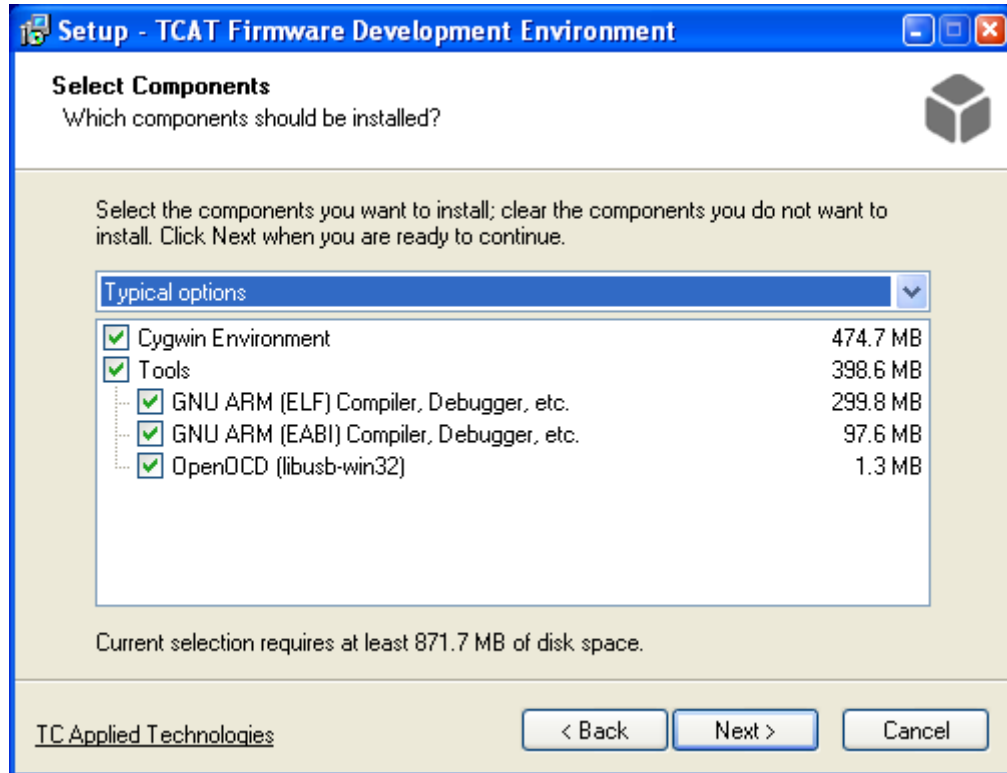


Figure 1: Installation components

Selecting the various components will cause them to be installed, or reinstalled if they already exist. Deselecting them will not uninstall them.

Cygwin Environment

Cygwin Base Installation

This includes a Unix emulation environment and various utilities that developers typically use. This is a required component for cross development of DICE firmware in Windows. The installation must be run for text (DOS) mode, since this is required by the other components. This distribution of the Cygwin environment does not include the sources.

If you wish to get the source files for the included modules, or add new ones, from the Cygwin mirrors, see section 9 below *Adding Cygwin Modules*. Note that the Cygwin site regularly updates the DLL's that support the environment. Take care to make sure that the versions on the web do not

diverge from this version such that they become incompatible with the rest of the tools in this version-stabilized collection. If you find yourself in that situation, you can always reinstall the Cygwin component by itself later from the Installer.

More information about Cygwin can be found at <http://www.cygwin.com>

Where is CygwinX?

CygwinX is not included with the preloaded package mirror in the Development Environment. The developer may add X or other modules not included in this distribution if they wish. See section 9 *Adding Cygwin Modules* below for details.

Tools

GNU ARM Compiler, Debugger, etc.

This is the GNUARM (arm-elf) toolchain for Cygwin. This includes bin utils, a compiler set, and gdb debugger and Insight debugger GUI for ARM embedded processors. This component is required for firmware based on the eCos 2.0 compatible firmware in the TCAT Subversion repository. Do not update this component with versions found on the GNUARM website. This is a version stabilized collection of tools, and you may have unexpected results by updating your compiler and tools. Also, while an updated compiler may still be compatible, the collection of utilities included in the new updates may be incomplete. This option can be installed along with the GNU ARM-EABI tools..

More information about this component can be found at <http://www.gnuarm.org>

GNU ARM-EABI Compiler, Debugger, etc.

This includes bin utils, a compiler set, and gdb debugger and a more recent version of the Insight debugger GUI for ARM embedded processors. This component is required for firmware based on the eCos 3.0 compatible firmware in the TCAT Subversion repository. These tools can be installed along with the GNU ARM tools.

OpenOCD

For those who wish to use OpenOCD compatible JTAG interfaces for board bring-up and debugging, this option will install the OpenOCD JTAG software.

TCAT firmware sources assume this version which has different configuration file syntax from previous versions.

Note that this version uses the libusb-win32 drivers. To use this version, update any existing drivers to the version provided by this installer. The drivers will be found in your Program Files directory in the OpenOCD\0.2.0\drivers directory. You must unzip the appropriate driver file corresponding to your JTAG dongle. For more information, see the *OpenOCD* section below.

Installation

Upgrading a 3.x Firmware SDK to 4.x

You can run the setup program with all options selected. The setup program will back up your /home directory tree with a new name that includes the date and time of installation. Also, your **/etc/group** and **/etc/passwd** files will be backed up for you as a convenience, but this is usually only useful to users who are in a Windows domain where the **mkgroup** and **mkpasswd** commands take a long time (i.e. in vpn situations with a large network). See below for details of these commands.

The Cygwin distribution in this installer will create separate icons for

Note This 4.x version of the SDK includes two versions of the GNU compiler, linker, etc. The tools that are invoked depend on environment variables in your .bashrc file in your Cygwin home directory. The firmware trees will invoke the tools it requires using these variables. The ELF tools are used for TCAT firmware that is based on eCos 2.0, and the EABI tools are used with firmware based on eCos 3.0 or later. The entries below should be present in your .bashrc file for the sources to build properly.

```
# for arm-elf toolchain
export GNUARM=/gnuarm

# for arm-eabi toolchain
export GNUARMEABI=/gnutools/arm-eabi

PATH=${PATH}:/gnuarm/bin:/gnutools/arm-eabi/bin
export PATH
```

The steps for installing and building your development environment are as follows:

1) *Run setup as Administrator*

- 1) Read the License Agreement and Installation Notes
- 2) Select the Install directory
- 3) Select Components to install (See the section *Installable Components* below).
- 4) Follow the prompts as required to install the selected components

Most developers will use the "Typical" selections. See the *Installable Components* section below for more information about which components to choose.

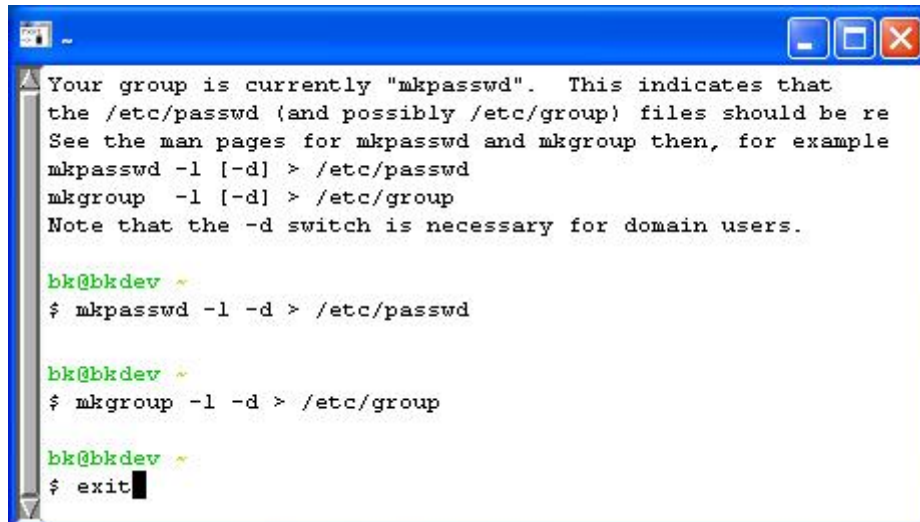
Note If you are reinstalling a component or updating from a previous version, there may be a pause while the setup program verifies the directory permissions for the existing directory tree. This is normal and can take a few minutes.

2) Run a bash shell using the icon created by the Installer



After the Installer completes, open a shell by double-clicking the Cygwin desktop icon.

Domain users will have to create **passwd** and **group** files as instructed by the bash initialization script (see the section Network Environments and Domain Users below). This extra step, shown in Figure 2, will be skipped for non-domain users.

A screenshot of a Cygwin terminal window. The window has a blue title bar with standard Windows window controls. The terminal text shows a message about the 'mkpasswd' group, followed by commands to create the /etc/passwd and /etc/group files, and finally an 'exit' command.

```
Your group is currently "mkpasswd". This indicates that
the /etc/passwd (and possibly /etc/group) files should be re
See the man pages for mkpasswd and mkgroup then, for example
mkpasswd -l [-d] > /etc/passwd
mkgroup -l [-d] > /etc/group
Note that the -d switch is necessary for domain users.

bk@bkdev ~
$ mkpasswd -l -d > /etc/passwd

bk@bkdev ~
$ mkgroup -l -d > /etc/group

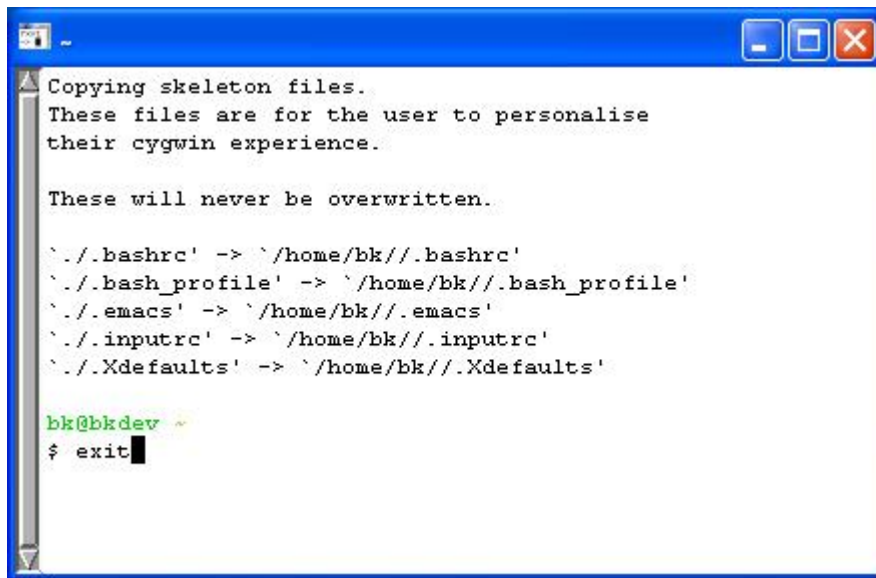
bk@bkdev ~
$ exit
```

Figure 2: Domain Users will see this message when using bash for the first time

Close this window before continuing.

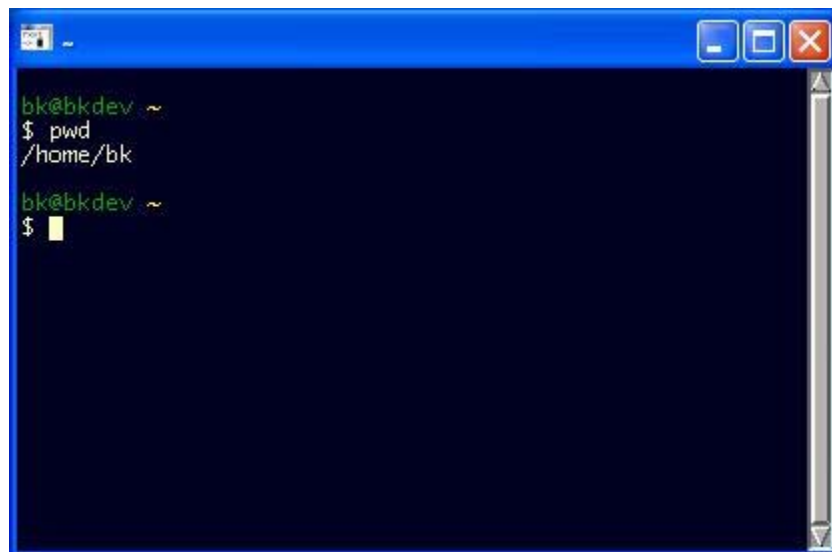
Your **bash** environment will be now configured for you when you next run a bash shell (see Figure 3). This step creates your **/home** directory and copies various default shell startup files.

If your home directory must reside somewhere other than the default, please see the section 6 below regarding *Network Environments*.



```
Copying skeleton files.  
These files are for the user to personalise  
their cygwin experience.  
  
These will never be overwritten.  
  
`./.bashrc' -> `/home/bk/./.bashrc'  
`./.bash_profile' -> `/home/bk/./.bash_profile'  
`./.emacs' -> `/home/bk/./.emacs'  
`./.inputrc' -> `/home/bk/./.inputrc'  
`./.Xdefaults' -> `/home/bk/./.Xdefaults'  
  
bk@bkdev ~  
$ exit
```

Figure 3: Default configuration files are copied for you
Note here that the default terminal is **rxvt** rather than the default 'DOS' bash window. If you prefer the default window, edit **cygwin.bat** in your installation root directory and comment out the relevant lines.

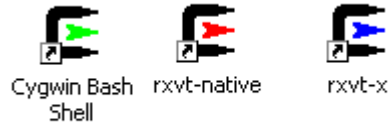


```
bk@bkdev ~  
$ pwd  
/home/bk  
  
bk@bkdev ~  
$
```

Figure 4: bash window after initial configuration.

The Cygwin installer will include three different icons for using the bash shell. The green icon will use a standard DOS window. The red and blue icons will open an xterm-like window which will use settings from your `~/.Xdefaults`

file. This allows changing of text colorization, window dimensions, fonts, and allows for a long history buffer.



The installer creates a default `~/.Xdefaults` file for you. Make changes in this file for your own preferences.

Note The root directory in the bash shell is not the root level of your Windows file system. Rather, it is set to the installation directory. For example, in the case where the environment is installed in **C:\cygwin**, this becomes the bash root directory. All of the paths below are equivalent:

Windows
C:\cygwin

Bash
/
/cygdrive/c/cygwin

3) Subversion repository sources

The source code for development is not included in the setup program. Please checkout the sources from the Subversion repository before continuing with the steps below.

In terms of firmware development, most if not all of the changes you'll be making will be to files contained in the project folder that you'll create from a template, such as:

/firmware/project/myProject

Updates to the sources from the TCAT server will never change the files in this folder of course, and you can commit them to your local version control system as you like.

If you are making changes to files outside of your project folders, then you can use a strategy such as keeping a checkout of the sources that are always synchronized to TCAT, and have a separate internal copy which you can use to diff/merge the TCAT changes into your development sources.

Information about accessing the sources from the online Subversion server is provided separately.

The *TCAT Firmware Development Environment User Guide* can be found in the Subversion repository. See your /docs/public/firmware directory in your svn tag, or view it directly from the repository here:

<https://dev.tctechnologies.tc/tcat/tags/release/public/latest/docs/firmware>

4) Sources on shared network drives

The firmware can be located anywhere on your file system, however, you must have both the **/firmware** and **/interface** directories together. Developers typically check out their entire subversion tag into a single directory, which keeps all of the dependent directories in the right place. Sources can also be located on network shares. See the section below *Network Environments and Domain Users*.

5) Build the development file structure and binaries

The development directory structure is completed by executing a shell script inside the **/firmware/project/** directory. Once you have configured your bash environment as above, open a bash shell and run the script.

```
user@machinename ~  
$ cd /firmware/project/
```

```
user@machinename /firmware/project/  
$ source install.sh
```

This script creates and builds a number of the development projects from templates. The required kernels are also built at this time, so it may take a while.

After the builds complete, the debug and binary images will be written to the **bin** directories within the newly created projects.

You are now ready to edit, compile, and debug the DICE Application code. See the *TCAT Firmware Development Environment User Guide* for more information about working with the DICE code. This guide will be in your subversion tag in the repository.

6) Build the source code documentation (optional)

The EVM project directory contains a **doxygen** configuration for making html and Windows Help versions of the source code developer reference. The Firmware includes a pre-built Help file based on the released source code. You can update these at any time using the doxygen command from your <projectName>/doxygen directory. The new documentation will then reflect your additions or changes to the source code.

In the case of the EVM002 project for example, the doxygen command is as follows:

```
user@machinename ~  
$ cd /firmware/project/evm002_tcd22x0/make  
  
user@machinename /firmware/project/evm002_tcd22x0/make  
$ make doxygen
```

The resulting html index file is written in the doxygen directory within the project directory:

```
... /doxygen/GeneratedDocumentation/html/index.html
```

The doxygen system will also automatically create a Windows Help file for you if you have the Windows Help Workshop installed. The resulting Windows Help file will be:

```
...  
/doxygen/GeneratedDocumentation/help/FirmwareUserCompiledHelp.chm
```

To remove all generated docs, delete the GeneratedDocumentation directory.

To install the Windows Help Workshop, go to

<http://go.microsoft.com/fwlink/?LinkId=14188> and install it with all defaults.

Network Environments and Domain Users

Domain Users

If you are using a workstation that is part of a domain, there are a number of considerations. You will almost certainly be using an account that does not have Admin credentials. In that case, use "Run As..." as described above using an administrator account provided by your system administrator.

Home directory

Collaborative environments commonly use mapped drives to network shares for file storage. In some cases, developers may be required to maintain their Cygwin **/home** directories on a shared drive. For example, if you want your home directory to be located on a network share that is mapped to drive **H:** in the **users** directory, run **mkpasswd** as follows:

```
user@computername ~  
$ mkpasswd -l -d -p /cygdrive/h/users > /etc/passwd
```

-p specifies the path, **-d** is only necessary if you are using a domain account

For a user account name of "joe" home directory will now be

/cygdrive/h/users/joe

which corresponds to the DOS path **h:\users\joe**, and all default configuration files will be copied there. You can verify your new home location with: **echo \${HOME}** at a bash prompt once you've opened a new bash shell.

Sources on shared drives

Another common situation is to store the development sources on a shared drive. In this case you can move the sources after installing the Environment, which includes both the **/firmware** and **/interface** directories.

Uninstalling

To uninstall the tools, use the **Uninstall** shortcut in the DICE Firmware Development Environment Start Menu Group.

This removes the Start Menu Groups, Registry entries and GNUARM tools from your computer. Note that the uninstaller does not remove the Cygwin "root" directory. This is for safety, so you will not unexpectedly lose your work.

In some circumstances, the directory may appear difficult to delete. The files and links created by the build process, etc., may have different Windows ACL's than the account that had been used to install them (i.e. Admin account). If this is the case, you will find that many of the files will be difficult to delete. To remove the files and links, login with an Administrator account, use the Windows ACL Editor (Security Tab in folder properties) and overwrite the Ownership and Permissions recursively for the Cygwin folder. Then delete the folder.

Your workstation will now be free of all files and Registry entries created by the Installer. Your system PATH variable will not be changed, however. You can manually remove the PATH entries after uninstalling the environment.

Adding Cygwin Modules

To add modules to the Cygwin installation, you can download the corresponding packages from the Cygwin mirror sites, copy them to your **<installroot>\cygwin_mirror\release** directory, and then install them with the Cygwin **setup.exe** program.

The details of this are as follows:

- 1) Assuming your installed root directory is **c:\cygwin**, run **setup.exe** in the **c:\cygwin\cygwin_mirror** directory.
- 2) Choose "Download without installing"
- 3) Use **c:\cygwin\cygwin_mirror** for the local package directory
- 4) Click through and select a download mirror.
- 5) When you see the Select Packages dialog, click on the "All" category until it says "Uninstall," as in Figure 5 below.

Note As noted earlier, the Cygwin site regularly has updated the DLL's and modules that support the environment.

Take care to make sure that the versions on the web do not diverge from this version such that they become incompatible with the rest of the tools in this version-stabilized collection.

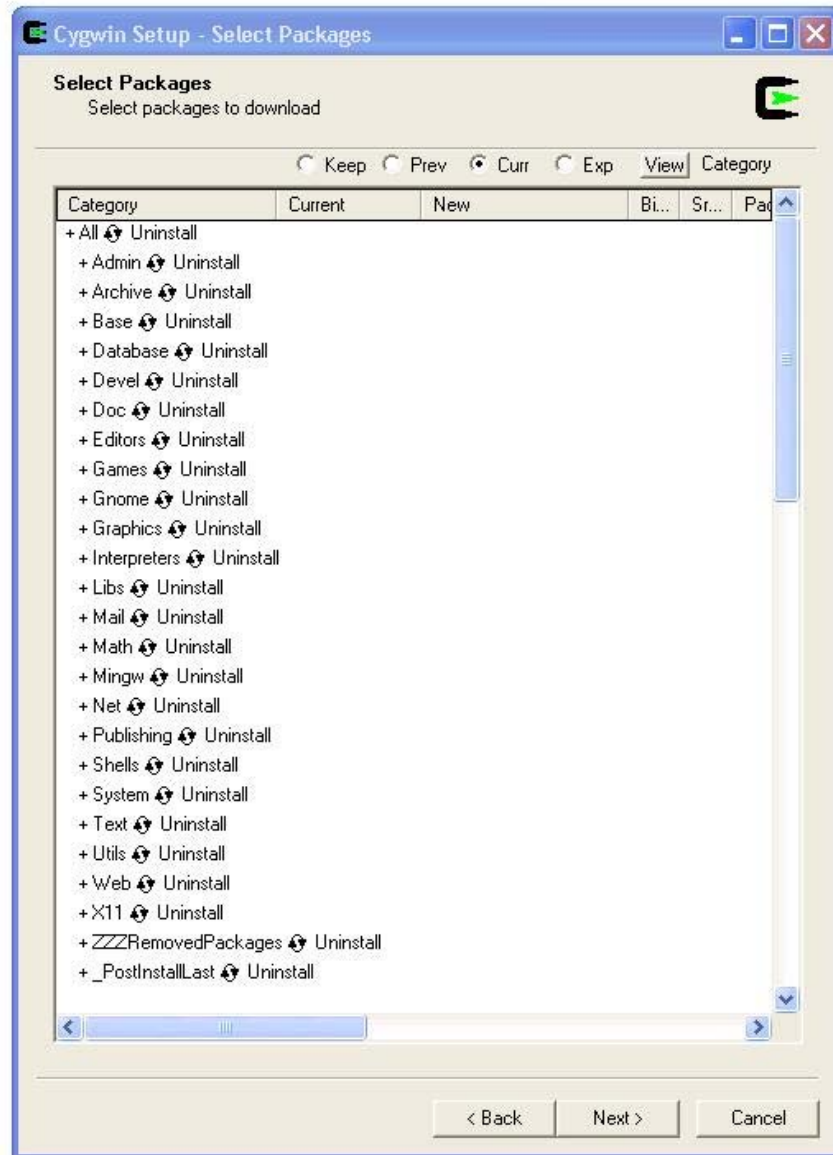


Figure 5: Deselect all of the packages

- 7) Click Next to download the module. This step often fails for one reason or another. If it does, you may have to go back to step 4 and start again...
- 8) Once the download completes and setup.exe exits, you will see a new directory in **cygwin_mirror** that is named with a decorated version of the mirror site's URL. Look for the directory name that contains the URL of the mirror that successfully downloaded your files.

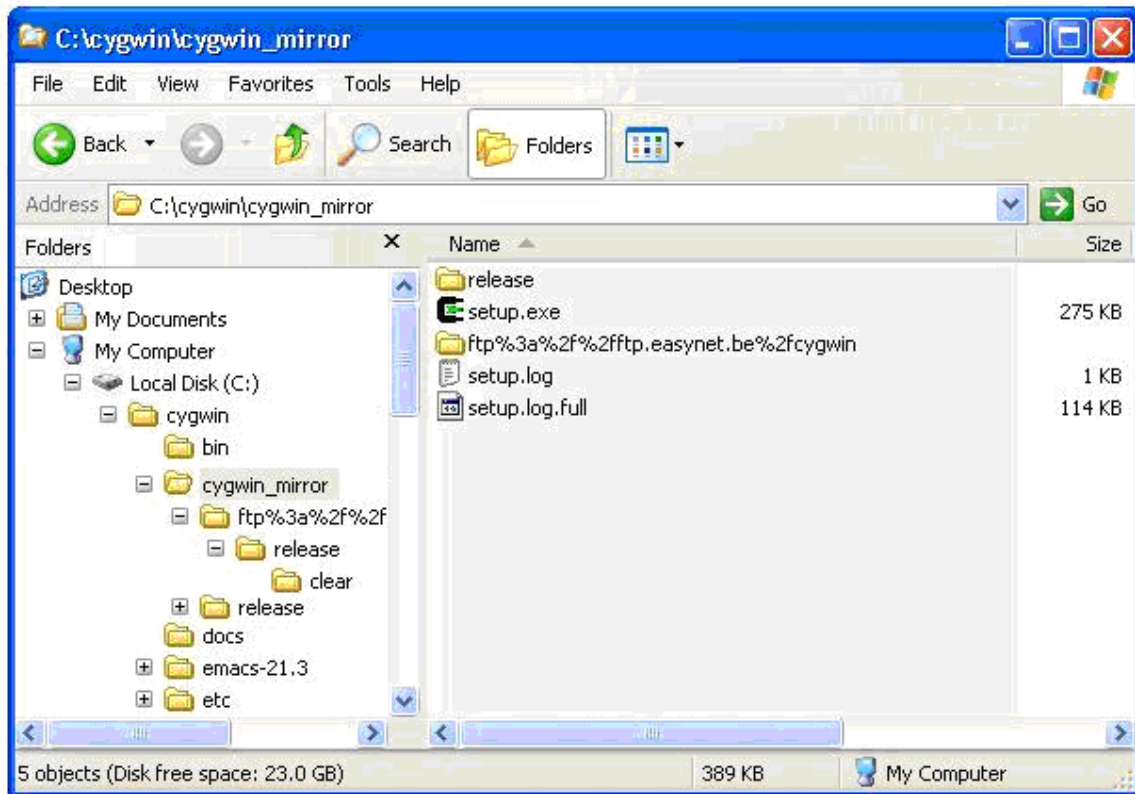


Figure 7: The download directory contains the new package(s)

- 9) Copy the modules from the release directory into your original **c:\cygwin\cygwin_mirror\release** directory. You can delete the download directory that was created by the download steps, and the setup log files.
- 10) Run **setup.exe** again.
- 11) Choose "Install From Local Directory"
- 12) Choose the same root directory as your original installation, **c:\cygwin** for this example, and select "All Users" and "DOS" Default Text File Type.

- 13) Use `c:\cygwin\cygwin_mirror` again for the local package directory
- 14) You'll see the "Select Packages" dialog again. Expand the relevant categories and make sure that your new module is selected for installation, then click Next.
- 15) After setup exits, open a bash shell and type `clear` to test the new command.

What's happening behind the scenes?

Installer

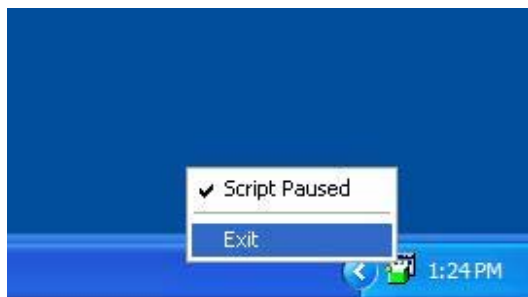
Launch the Installer

Checks for Administrator credentials, gets a password and an installation path from the User. Then the Installer gets the component selections, calculates available disk space for selected components.

If the path points to a new directory, the installation root directory is created with "Full" access Permissions for the "Everyone" Group. This allows developers who do not have Administrator privileges to create/delete/modify files in all directories below this root. If the directory already exists, the installer makes sure that the directory tree below the specified directory has the correct access permissions.

If you are reinstalling a component in this directory, this can take a few minutes and is normal.

Setup Script



The Installer then copies relevant files and an uninstaller launches a setup script, which performs the steps described below. Note that if you wish to cancel this part of the installation at any time, you can right-click on the tray icon and choose Exit. The script will then exit after it has finished its current operation.

Figure 8: Use the tray icon to exit the setup script if desired

Install Cygwin

The setup script adds Registry entries that cause Cygwin's setup.exe to install from the mirror with the appropriate settings, copies various files that enforce our particular installation type, and runs setup.exe as an unattended install. Setup.exe processes the package mirror copied earlier, and then executes configuration scripts for each package where found. The package mirror contains a previously downloaded set of Base packages. You may download and add your own packages later from the online Cygwin mirror sites, as described above in *Adding Cygwin Modules*.

Note This may overwrite existing customized configuration files.

Install GNUARM ELF tools

This writes a Registry value that makes GNUARM setup default to the installation directory specified above by the User. Then the setup script runs the standard GNUARM setup with strong defaults. Patches a tkl script on the Insight debugger target connection settings dialog.

Install GNU ARM-EABI tools

Copies the tools to /gnutools.

Updated Insight debug GUI

If either GNU tools option is selected, a version of Insight that will work with both arm-elf and arm-eabi images will be copied to the /gnuarm/bin directory. The existing Insight will be renamed with the data and time in the new file name.

Install OpenOCD Software

Runs the installer. This is a version that uses the libusb-win32 driver. Copies the installer to the /uninst directory so the uninstaller can use it.

Install Documents

If selected, copies the files to disk, and creates Start Menu items for them.

Miscellaneous

The setup program also creates a **.emacs** file in your home directory, for developers who want to use emacs as their edit/debug environment. It will back up any existing .emacs file with a name including the data and time of installation.

Cleanup

Removes temporary files, and corrects a PATH issue caused by the GNUARM installer.

Uninstaller

Launch Uninstall

- Removes Registry entries created by Installer and Components.
- Removes Start Menu Folders
- Removes some unneeded folders and files.
- Runs GNUARM Uninstaller if present
- Runs OpenOCD Uninstaller if present
- Removes most files from the Cygwin directory.

Cygwin Root Directory

The Uninstaller does not remove the Cygwin “root” directory.

The best way to deal with the Cygwin folder is to make sure your work within the Cygwin directory structure is saved and then delete the folder.

If you have installed the environment as a different User (i.e. Administrator) you will find that many of the files will be difficult to delete. This is due to files being created by the build process, etc., by Users that have different Windows ACL's than the User that installed the environment.

In that case, to remove the files and links, login with an Administrator account, use the Windows ACL Editor (Security Tab in folder properties) and overwrite the Ownership and Permissions recursively for the Cygwin folder. Then delete the folder.

OpenOCD

There are two different USB-layer drivers available for OpenOCD compatible JTAG dongles. One is the open source libusb-win32 and the other is the proprietary FTD2xx drivers from FTDI. The libusb-win32 drivers have slower throughput than the FTD2xx drivers, and may not work on Windows Vista and Windows.

FTDI

Since it is not legal to distribute an open source program that is linked to proprietary libraries, we cannot include an OpenOCD that will work with the proprietary FTD2xx.dll. However, you can use the proprietary drivers if you build the OpenOCD server yourself. The drivers are generally distributed by the manufacturer of the dongle.

libusb-win32

This development environment uses an OpenOCD server that is compiled for the libusb-win32 drivers, and includes several drivers for commonly used JTAG dongles. To install the drivers for your dongle, go into the Program Files directory and look for OpenOCD\0.2.0\drivers directory. There, unzip the appropriate zip file for your device.

If you had previously installed the FTDI drivers, see the info.txt file in the OpenOCD\0.4.0 directory for a full description for uninstalling them and reinstalling with the libusb driver.

Source code snapshots can be found here

http://developer.berlios.de/project/showfiles.php?group_id=4148

The SVN repository is here

<http://svn.berlios.de/svnroot/repos/openocd/trunk>

About Non-Native Serial ports

For debugging, a native serial port is recommended. A second serial port is also recommended for using the CLI during debug. A pair of native serial ports is not commonly included on recent workstations and laptops. If your computer doesn't have a native serial port, or only has one, there are still a number of good alternatives. You can add a PCI card (most any will work fine) or USB-Serial adapter.

Due to the way the Toolchain uses serial gdb target debugging, communications over certain brands of USB-serial adapters can be intolerably slow. Some adapters are not only slow, but the drivers can cause your machine to seize up even when used normally. We do not recommend Belkin adapters for this reason. We have found that adapters from Keyspan operate reliably at full speed using this Toolchain.

Keyspan models, such as the 19-HS and 49-WLC, perform at speed and have stable drivers.

Component Origins

The TCAT Firmware Development Environment is provided free of charge. For details on the origins of the included components, please refer to the following information for the components:

Cygwin

<http://cygwin.com/>

<http://cygwin.com/licensing.html>

GNUARM (ELF)

<http://www.gnuarm.com/>

<http://www.gnu.org/licenses/licenses.html>

GNU ARM (EABI), built from a gnutools distribution provided by eCosCentric at

<http://mirrors.kernel.org/sourceware/ecos/gnutools/cygwin/>

<http://www.gnu.org/licenses/licenses.html>

OpenOCD: Windows Installer

<http://www.freddiechopin.info/index.php/en/download/category/4-openocd>