

---

# DICE Firmware Development Environment Installation Guide



## CONTENTS

<b>Overview</b>	<b>3</b>
<b>Features</b>	<b>3</b>
<b>Firmware Sources</b>	<b>4</b>
<b>New In This Release</b>	<b>5</b>
<b>Checklist</b>	<b>6</b>
<b>System Requirements</b>	<b>6</b>
Required	6
Recommended	6
<b>Planning your Installation</b>	<b>7</b>
Existing Cygwin installation	7
Login User Name	7
Administrator credentials	7
<b>Installable Components</b>	<b>8</b>
Cygwin Environment	8
Tools	9
Documents	10
<b>Installation</b>	<b>10</b>
Upgrading a 2.x Firmware SDK to 3.x?	10
Fresh Install	10
<b>Network Environments and Domain Users</b>	<b>16</b>
<b>Uninstalling</b>	<b>17</b>
<b>Adding Cygwin Modules</b>	<b>17</b>
<b>What's happening behind the scenes?</b>	<b>21</b>
Installer	21
Launch the Installer	21
Setup Script	21
Install Cygwin	22
Install GNUARM	22
Install Documents	22
Install GNU Windows Emacs	22
Cleanup	22
<b>Uninstaller</b>	<b>23</b>
Launch Uninstall	23
Cygwin Root Directory	23
<b>About Non-Native Serial ports</b>	<b>24</b>

### Overview

The DICE 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 DICE Firmware Development Tools, and Documentation. Firmware Source Code is no longer included with the Development tools. Sources are now available on the TCAT Subversion repository.

Once you have installed the Environment, please see the *DICE Firmware Development Environment User Guide* for the next steps.

### Features

- ✓ Version Stabilized distribution of Toolchain Components
- ✓ Open Source, Royalty-Free RTOS
- ✓ Windows Cross-development using Cygwin and GNUARM
- ✓ Standard ARM Compiler, Linker, Debugger
- ✓ Serial debugging, with JTAG optional support  
JTAG pod is not required, however several devices are supported.
- ✓ Support  
Ongoing development of new features provided in future updates  
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
  - Searchable help file provided
  - Context-sensitive integration for several common IDE's
- ✓ 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, and EVM002 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.
- ✓ Utilities
  - Dice test application, Device Inspector, etc.
- ✓ Control Panel Examples

### New In This Release

#### Subversion Repository

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.

#### Common

- Versioning

The version in `/interface/tcat_dice_version.h` is used for host drivers and firmware. The build number referenced in the title page here is based on the version in this file.

- Simple Persistent Storage (SPS)

An API for storing runtime settings in flash memory that must persist across reset.

#### DICEII

- DICEII microboard support on the EVM002
- EVM002 project template for developing firmware with the DICEII microboard.

See the shell scripts in `/firmware/project` for examples of using the templates.

- SPI Support

Software SPI implementation for DICEII which mirrors the hardware SPI on DICE JR/MINI.

A number of improvements and other changes have been made.

See [/firmware/firmware\\_release\\_notes.html](/firmware/firmware_release_notes.html) for detailed changes.

### 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

See the DICE Firmware Development Environment User Guide for the following:

- Checkout your Tag
- Build the initial EVM projects
- Ready to make your application

### System Requirements

#### Required

Windows 2000, Windows XP  
1 GHz or greater Pentium Class CPU  
750MB free hard drive space  
(1) Serial Port  
640MB RAM  
CD-ROM Drive

#### Recommended

1GB or more RAM  
(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 DICE 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.

Please specify a new directory for this initial installation of the Environment.

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**

### Administrator credentials

Since included component installers will make use of the Windows Registry, the Installer must run with Administrator privileges.

The most straight-forward way to do this is to install the Environment from the account that you will use for development. If that account does not have Admin credentials, then use "Run As..." by right-clicking the setup program icon (Windows 2000 users hold the shift key while right-clicking). Then enter an account name and password that has either Local or Domain Administrator credentials.

## Installable Components

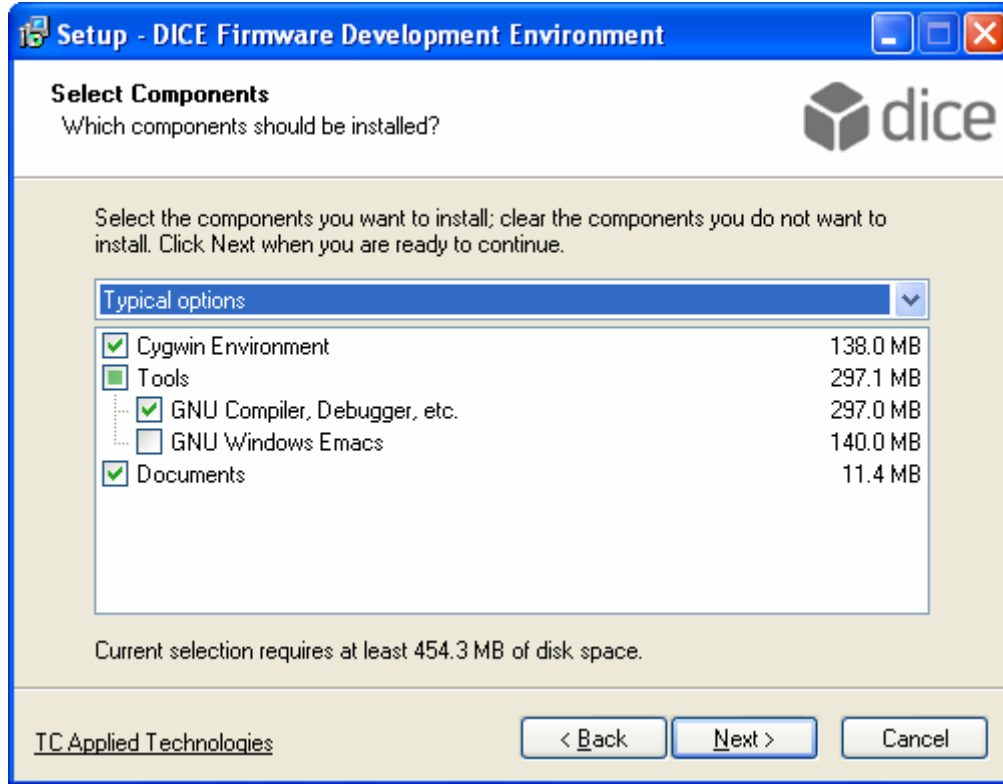


Figure 1: Installation components

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 Compiler, Debugger, etc.*

This is the GNUARM 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 DICE cross development in Windows. 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.

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

*GNU Windows Emacs*

This is the GNU Windows implementation of Emacs. This component is not required to develop applications for DICE. It's included as an optional editor for those who are used to Emacs but do not wish to have the overhead of the full X Window system distribution on their development machine to run the X version of Emacs. Once this is installed you can run GNU Windows Emacs using the Start Menu icon, or you can type "**runemacs**" in a bash shell.

### Documents

Documents are provided in PDF form. If this component is selected, the documents are copied and shortcuts are created in the Program Files menu for convenience.

Also, a searchable Windows Help File is provided, which documents the major API's in the DICE software stack, including a CLI reference. This file is generated using tags in the source header files themselves. See the instructions above for using doxygen. This can also be integrated with IDE's, such as Visual Studio 8.0 and others.

## Installation

### Upgrading a 2.x Firmware SDK to 3.x?

You can use your existing SDK installation. All you have to do is checkout your tag from the online sources into a new `/firmware` directory and unset your `TC_DIR` variable in your `~/bashrc` file.

### Fresh Install

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.

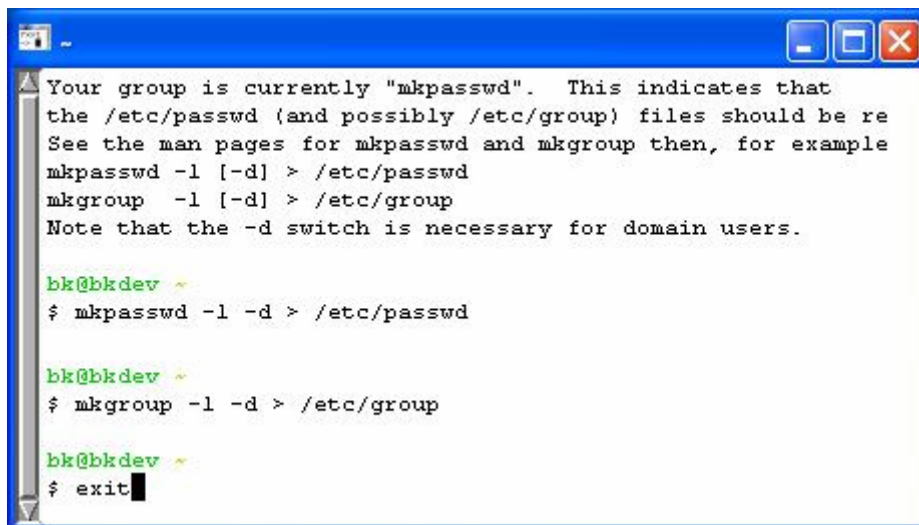
**Note** If you are reinstalling a component, there may be a pause while the setup program verifies the directory permissions for the existing directory tree. This is normal and can take a few minutes.

### 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 terminal window with a blue title bar. The window contains text instructions for creating passwd and group files, followed by command-line examples. The text reads: "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." Below this, there are three lines of command-line input: "bk@bkdev ~", "\$ mkpasswd -l -d > /etc/passwd", "bk@bkdev ~", "\$ mkgroup -l -d > /etc/group", and "bk@bkdev ~", "\$ exit".

```
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

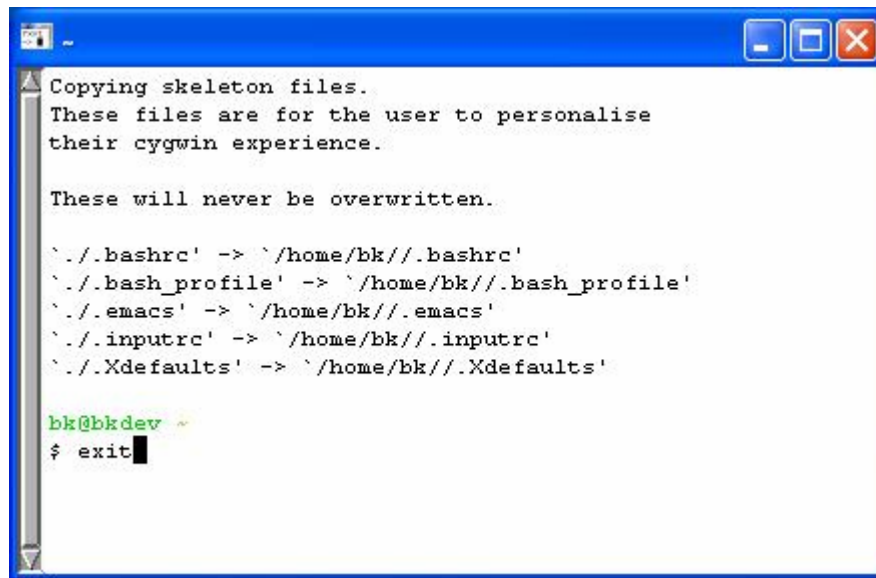
bk@bkdev ~
$ exit
```

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*.



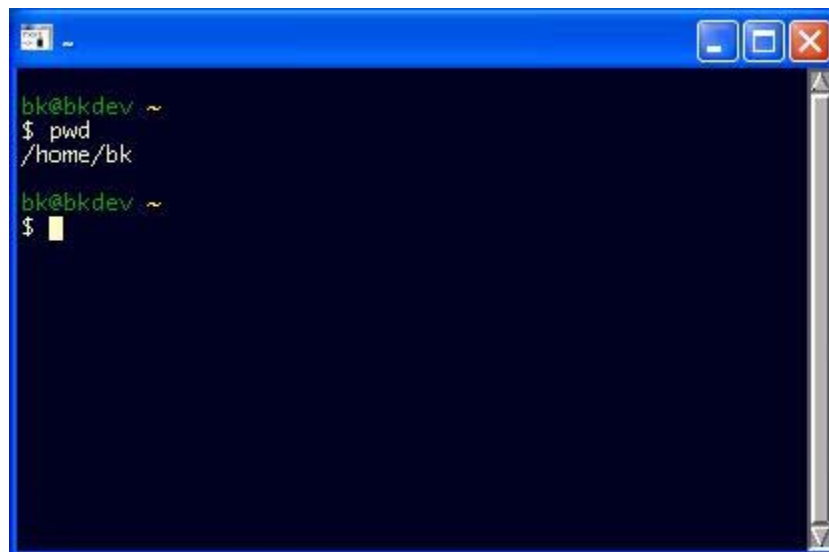
```
Copying skeleton files.
These files are for the user to personalise
their cygwin experience.

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: rxvt bash window now loads customizations from `~/.Xdefaults`

From now on, your bash window will load customizations from the `~/.Xdefaults` file that is placed there 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

If you wish to use the latest sources from the Subversion repository, then checkout the sources 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 (see the Firmware Development Environment User Guide for more info), 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.

### 4) Sources on shared network drives

The firmware can be located outside of the default location if necessary. If you wish to move the sources to another place on the workstation, or use them from a shared directory, you must move both the **/firmware** and **/interface** directories together. 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 the development projects from templates, one for the DICEII on the DICEII-EVM, one for the DICEII microboard in the

EVM002, and one for the DICE-JR/DICE-MINI microboards on the EVM002. The required kernels are also built at this time, so it will 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 *DICE Firmware Development Environment User Guide* for more information about working with the DICE code.

### 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.

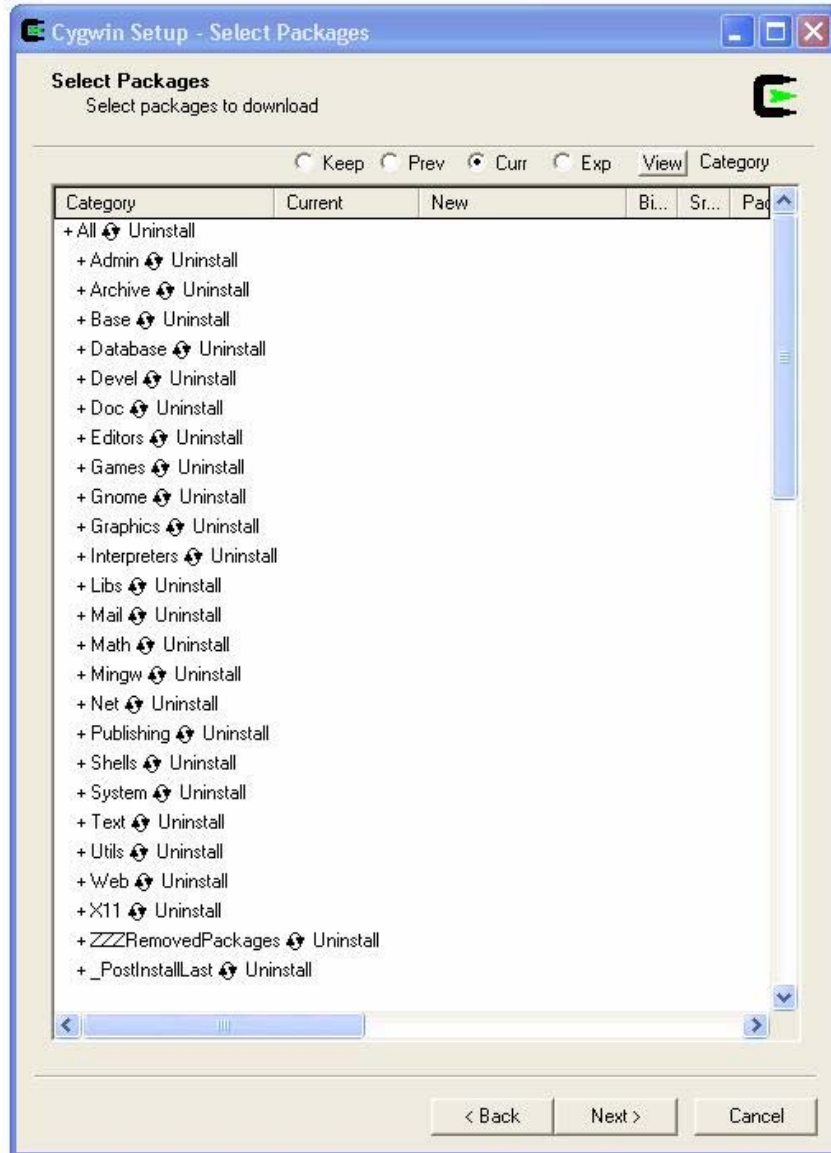


Figure 5: Deselect all of the packages

- 6) Now browse for the packages you want to add (any dependencies will be automatically added for you). For this example, the **clear** module is selected. **clear** is a shell command that clears the screen. [Of course, this is just an example, you can already clear the screen using Control+L]

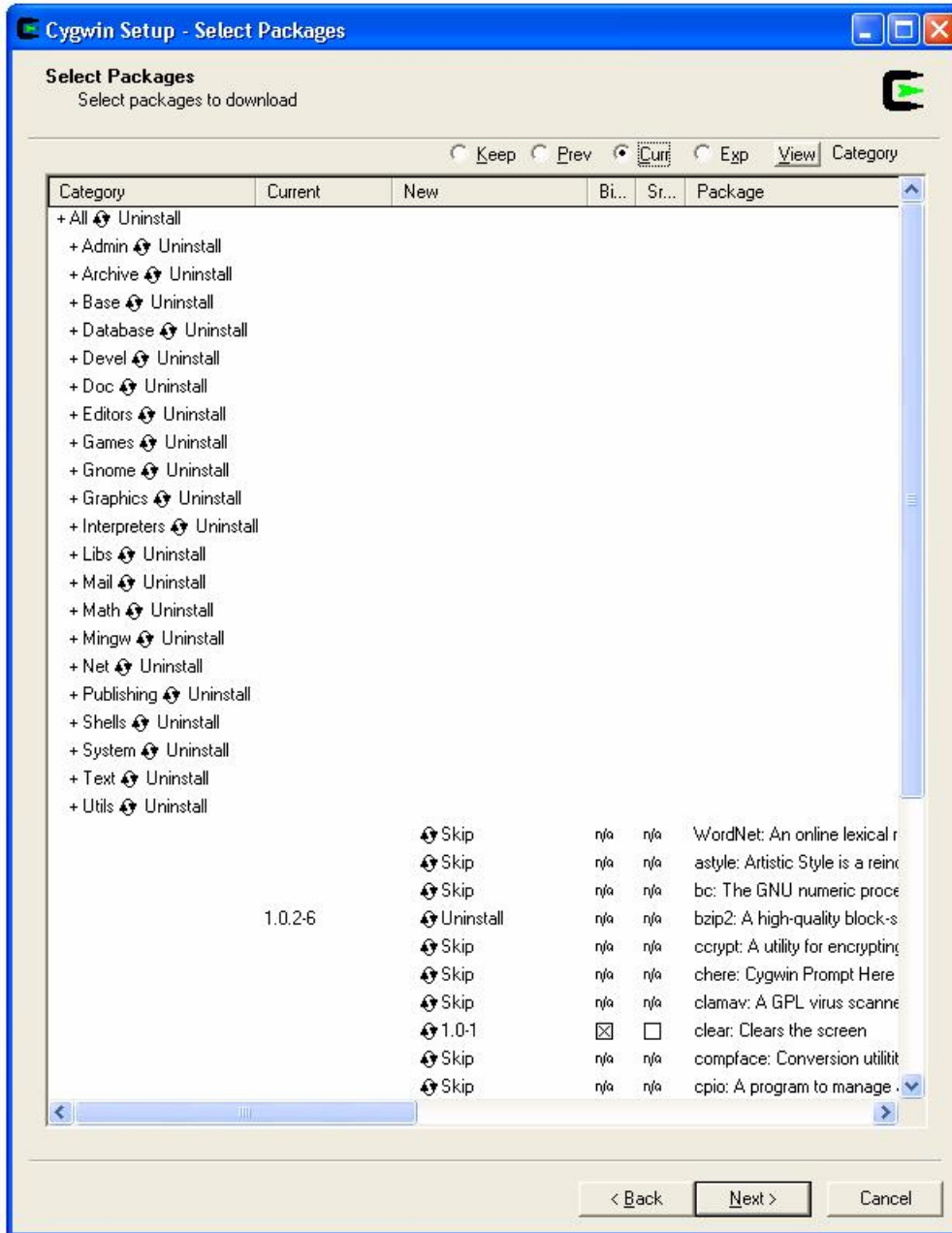


Figure 6: Choose the clear module.

- 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.

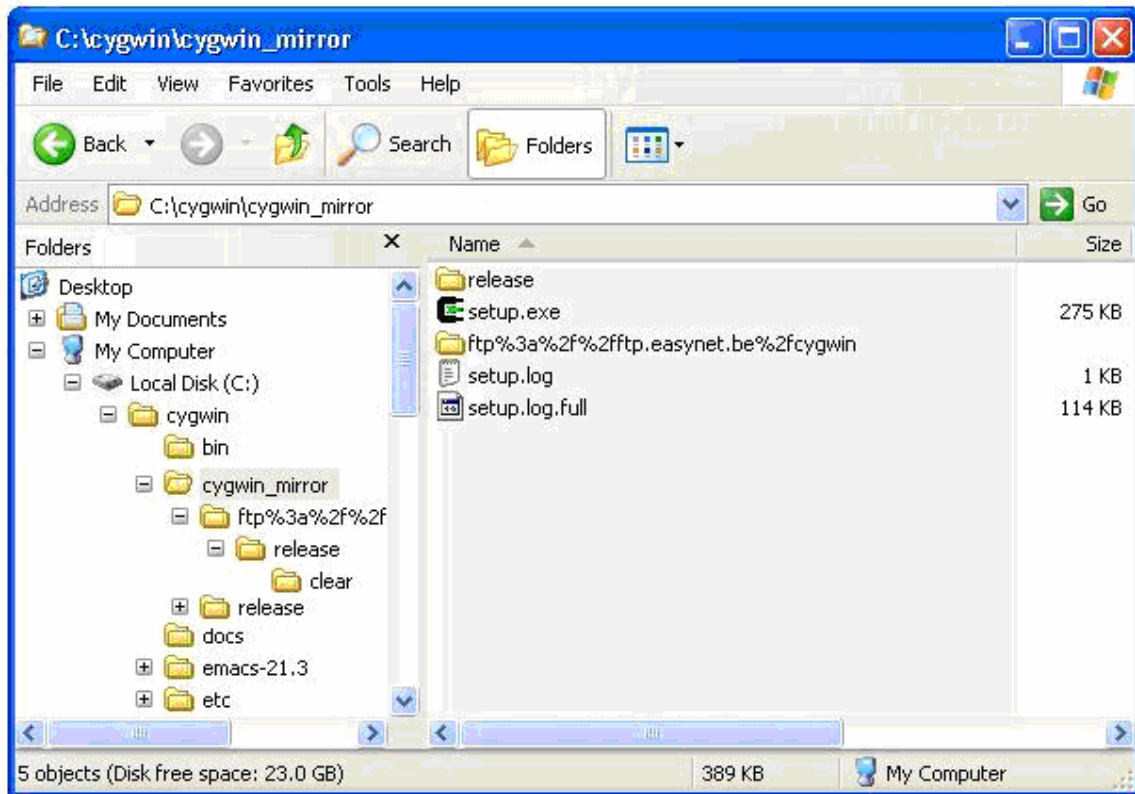


Figure 7: The download directory contains the new package(s)

- 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

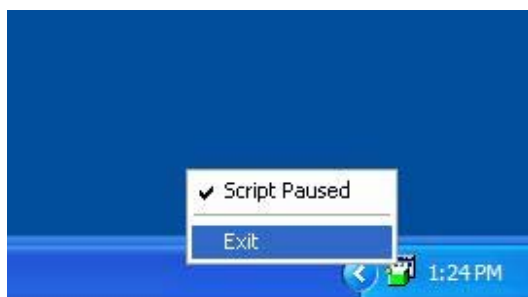
#### Launch the Installer

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

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 tk1 script on the Insight debugger target connection settings dialog.

### Install Documents

If selected, copies the files to disk, and creates Start Menu items for them. This also copies a Windows Help file that details the DICE firmware API's and CLI commands.

### Install GNU Windows Emacs

Copies and extracts the Emacs binaries. Installs a Start Menu icon, and copies a `.emacs` file customized for Cygwin integration in your chosen path.

**Note** This may overwrite existing customized emacs configuration files.

### 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
- 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.

### 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.