TC | APPLIED
TECHNOLOGIES

DICE Firmware SDK
Migration Guide

TC | APPLIED
TECHNOLOGIES

CONTENTS

TC | APPLIED
TECHNOLOGIES

Overview

Developers using version 2.x of the DICE Firmware SDK will find information here that describes the differences between versions, and for migrating their applications to the new 3.x source tree and development methods.

The newest sources can be used in your existing DICE Firmware SDK installation. The Development Environment (previously known as the Firmware SDK) no longer includes the firmware sources, Contact TCAT for access to the online source code repository. It is otherwise the same so there is no need to reinstall the environment to use the new source tree.

Use this document along with the *DICE Firmware Development Environment User Guide*

Checklist

☐ Install (from your Subversion checkout) the new source code directory from the online repository in to a new local location.

☐ Change your bash environment

☐ Review Differences and Additions

☐ Migrate your changes to 2.0.x sources into the 3.x sources

☐ Update Host drivers to those using the New Vendor ID (1394 OUI)

Using 3.x Source Code in an Existing 2.x SDK

Online sources

When upgrading to this new version, the developer will checkout the sources from the TCAT Subversion repository. This will create the new self-contained **/firmware** directory, which is used instead of the **/dice** directory that's installed with the 2.x SDK, and copy the Documentation. Contact TCAT for access and instructions for using the repository.

Environment

You will no longer have to manage your TC_DIR environment variable in your **/home/.bashrc** file. The variable is set dynamically by the build system, so you can remove it.

TC APPLIED TECHNOLOGIES

Main Differences

New Directory Structure

The new SDK makes it possible to easily develop multiple applications in the same source tree, and paves the way for drop-in support of new variants of the DICE chip family.

The sources have been separated into new groupings that separate chip-specific code, core modules, operating system sources, and project-specific files.

All firmware source code, including the eCos source files, is contained in the main source tree in the **/firmware** directory. Additionally, an **/interface** directory is also used at the same level as the /firmware directory for common header files used in firmware and host application development. Contact TCAT for more information about host application development software for DICE.

EVM Support

The 3.0.x Firmware and later supports the DICEII, DICE-JR and DICE-MINI chips, on the EVM002 development board, and continued support of the DICEII chip on the classic DICEII EVM.

Projects

Template Projects

Developers are provided with template projects for new development. These projects are fully working implementations that are based on the DICE II EVM and the EVM002, which supports the DICEII, DICE-JR and DICE-MINI microboards. Custom projects can be created from the templates using the included shell scripts.

Most source code that is changed by the developer for creating custom applications is collected in the resulting project directory.

EVM Projects

Each EVM is supported in its own project and has a number of improvements, including a **myMode** example that configures the EVM to boot into a number of different audio channel configurations for evaluation and example code.

After running the install.sh script in the **/firmware/project** directory, you will find projects created from templates for the DICE EVMs. For example, the DICE II EVM project is in

/firmare/project/evm_dice2

Developers who wish to create custom projects can use a template project to create new starting points from the EVM tamplate that most closely matches their custom hardware.

If you use the Visual Studio IDE for development, you can use the included VC++ project files in the project's **/make** directory. Build options will apply to the currently open project as before, and if you install a Tool that launches the Insight debugger, that tool will always debug the binary associated with the currently loaded project.

Otherwise, as before, you can build and launch the debugger from a bash command line, the included emacs environment, or within your favorite IDE.

eCos Sources

As mentioned above, all firmware sources are contained in the **/firmware** directory, including the eCos distribution.

Note that using a command-line search tool will traverse the eCos sources, which can make this an unnecessarily long process. You may wish to modify your search to exclude the **/firmware/os/ecos/src** directory. Otherwise, make sure your IDE is set up to search only files included in the Project.

Build System

The build system has been changed to accommodate the new structure, and multiple projects. Object, library and executable files are kept in each local project directory. Additionally the build system creates resulting binary files and directory names by concatenating the Project defines used in Make.params.

New or Changed Modules

Inter IC Communications

The **i2c** interface now has an interrupt-driven implementation, which improves performance over the polled method. Note that the interface operates in polled mode until the i2c interrupt-based service is running. This happens after the call to **TCTaskingStart()** is called.

Timer2

This module provides a fine timer in system clock ticks (1/49152000 sec).

AML

MIDI is now abstracted into a common interface, so MIDI can be more easily integrated and routed between different serial and 1394 interfaces.

See the new **aml** commands in the CLI, for looking at the configuration and for testing byte transfers between the interfaces.

myMode

The template projects implement various configurations which work on the EVM's, including different configurations of audio inputs and outputs.

TC APPLIED TECHNOLOGIES

SPI

A serial-parallel interface driver is supplied for the DICE-Jr and DICE-Mini chips.

Driver Modules

DiceDriver

The DiceDriver Host implementations have greatly improved in terms of performance and stability. Along the way some changes and additions to the interface have been implemented. Contact TCAT for the latest Host driver interface for use with your Control Panel or other GUI applications.

AV/C

This release contains an AV/C implementation that has been updated for class compliant AV/C Audio. When run on the EVM it works with Mac OSX without host driver installation. The majority of the files reside in the

/firmware/module/1394avc

directory with additional changes to the **targetplugs** interface among others. An AV/C Application is built by defining the **1394AVC** driver module in **Make.params**

The current version supports 8-in and 8-out (AES0-AES3) with MIDI, on the DICE II EVM. This is a class-compliant AV/C and streaming implementation. This implementation will undergo ongoing improvements as the Host drivers are improved. Contact TCAT for details on AV/C application development.

OGT

mLAN is now implemented as the Open Generic Transporter (**OGT**) and is provided separately. This module is a project-level drop-in. The 1394 stack supports the necessary CSR registers needed to support OGT requirements. Contact TCAT to obtain the firmware modules that support OGT.

New Default Vendor ID

Host drivers, based on the DiceDriver model, load the appropriate drivers based in part on the vendor ID (1394 OUI) that is contained in the device's Configuration ROM.

This is encoded in the firmware in the file:

/firmware/project/myProject/target/interface/targetVendorDefs.h

using the #define for `THIS_VENDOR_ID`

In the 3.x firmware distribution this ID has been changed to the TC vendor ID. Therefore, older drivers will not load for this version of the firmware. At this point there are two choices:

1) Update to the new development drivers from TCAT

> It is recommended that developers update to the latest drivers from TCAT during preproduction firmware development, i.e. until they are working with drivers that use their organization's OUI.

2) Change the #define back to the old vendor ID temporarily until you have new updated drivers from TCAT.

> If you wish to continue working with your existing drivers, this is a temporary solution. However, to benefit from the latest performance, functionality and stability improvements of the drivers, we recommend the first option above.

Note

In production devices, you must use your own Organizationally Unique Identifier (OUI), which is coded for in your driver distributions. Contact TCAT for details on using your OUI with the Host drivers. See the *DICE Firmware SDK User Guide* for information about where to obtain an OUI.

JTAG development and bring-up

JTAG debugging is covered in more detail in the User Guide. Also, new board bring-up with JTAG does not require an installed Firmware to work if you wish to use the prebuilt binaries available online. This can be the basis for product service, factory bring-up and test, etc.

Merging Your 2.x Changes into the New Structure

In most cases, the firmware files still reside in the same subdirectory names, although the directories themselves have been moved around. In most cases, the sources are now separated between C code and header files in ..**/src** and ..**/interface** directories respectively for each module.

Unless you have made changes to the core modules, there will be relatively few sources to merge into the new structure. Knowing this, Developers will generally know what to do, however we include an approach here for completeness.

Diff and Merge

The developer can compare an original 2.0.12 version of the sources with the current **/dice** working sources to see what files differ, then merge the changes into the relevant project in the new **/firmware** tree.

This can be done by exporting it from your local version control if you committed it all before making changes, by installing the source code from the original installation (renaming the **/dice** directory first), or by extracting the sources from the Installation distribution.

2.x sources

The files in the Installation distribution (CD or zip file) are:

..\distrib\dicecode\dice\dice_2_0_12.tar.gz

> This is copied into **/dice/diceApp** and extracted

..\distrib\dicecode\dice\misc_2_0_12.tar.gz

> This is copied into the Cygwin root **/** and extracted

If you have changed the kernel use:

..\distrib\eCos\ecos_2_0_12.tar.gz

This is copied into /dice/ecos-2.0 and extracted

Preparing for Diff

DOS vs. Unix newlines

> Note that if your favorite diff utility has setting to ignore end-of-line
> charaters, then use it here. Otherwise, when you diff the files, any end-of-line
> characters that are different between the two files will cause diff tools to view
> the entire file to be 'different.' This can happen if you have used various
> editors that save with different end-of-line formats (i.e. emacs, vi, vs. Visual
> C++), and will make it impossible to visually merge your files. To make sure
> this doesn't present a problem, you can make sure both sources have
> consistent end-of-lines as follows:

> user@computername ~
> **$ cd /dice/diceApp**
> user@computername /dice/diceApp
> **$ find . \(-name "*.h" –o –name "*.c" –o –name "Make*" \) \
> –exec /unix2dos '{}' **;

Merge into a Template-based project

> If you are going to make a lot of new applications based on your changes to
> the 2.0.12 sources, you may with to merge them into a copy of a DICEII EVM
> template project. Otherwise just create a new project based on the DICEII
> template and merge into it.

> user@computername ~
> **$ cd /firmware/project**
> user@computername /firmware/project
> **$ ./ new_diceII_proj.sh myProject**

> Or whatever name you wish to use for your project. This creates

/firmware/project/myProject and is ready for use.

3.x sources introduce the DICE-MINI and DICE-JR chips, and also include support for the DICEII on the new EVM002. If you wish to merge 2.0.12 DICEII sources into a 3.x project, you may have the best results if you familiarize yourself fully with the new features and other differences to determine which new project template will best suite your needs.