Freescale Semiconductor
Application Note

Document Number: AN4816
Rev 0, 09/2013

Introduction to DSC56800EX Quick Start Development Environment

1 Introduction

The DSC56800EX Quick Start is a software environment for the embedded applications development. It provides fully debugged peripheral drivers, examples and interfaces that allow programmers to create their own C application code.

The Quick Start helps users to accelerate the application development by the graphical configuration of used peripherals and the run-time peripheral configuration by peripheral drivers. An embedded developer becomes quickly familiar with the target device and how to create real-time applications rapidly and efficiently while retaining complete control over each portion of the underlying hardware.

The DSC Quick Start toolset is specially designed for the hard real-time applications written in C or mixed Assembler/C languages where deterministic behavior and transparent software structure are required. It provides a software infrastructure that allows development of efficient applications that are portable and reusable between devices within the DSC architecture family.

Contents

1 Introduction 1
2 Core-system infrastructure 2
3 Quick Start low-level drivers..................... 3
4 Project stationery 4
5 Graphical Configuration Tool (GCT)........ 5
6 FreeMASTER software drivers 6
7 Example applications and user documentation... 7
8 New features in Quick Start version 2.6.... 7
9 References 7
10 Revision history.. 7

© 2013 Freescale Semiconductor, Inc.

The Quick Start tool version 2.6 [2] is now available for the new 56800EX family of Digital Signal Controllers. It provides the out of box support for 56F82xxx and 56F84xxx devices and evaluation boards. The version 2.5 is still available for download and supports the 56800 and 56800E families of DSC (56F83xx, 56F82xx and 56F80xx [1]).

This figure shows the PINOUT configuration page of the Graphical Configuration Tool (GCT) as an example.

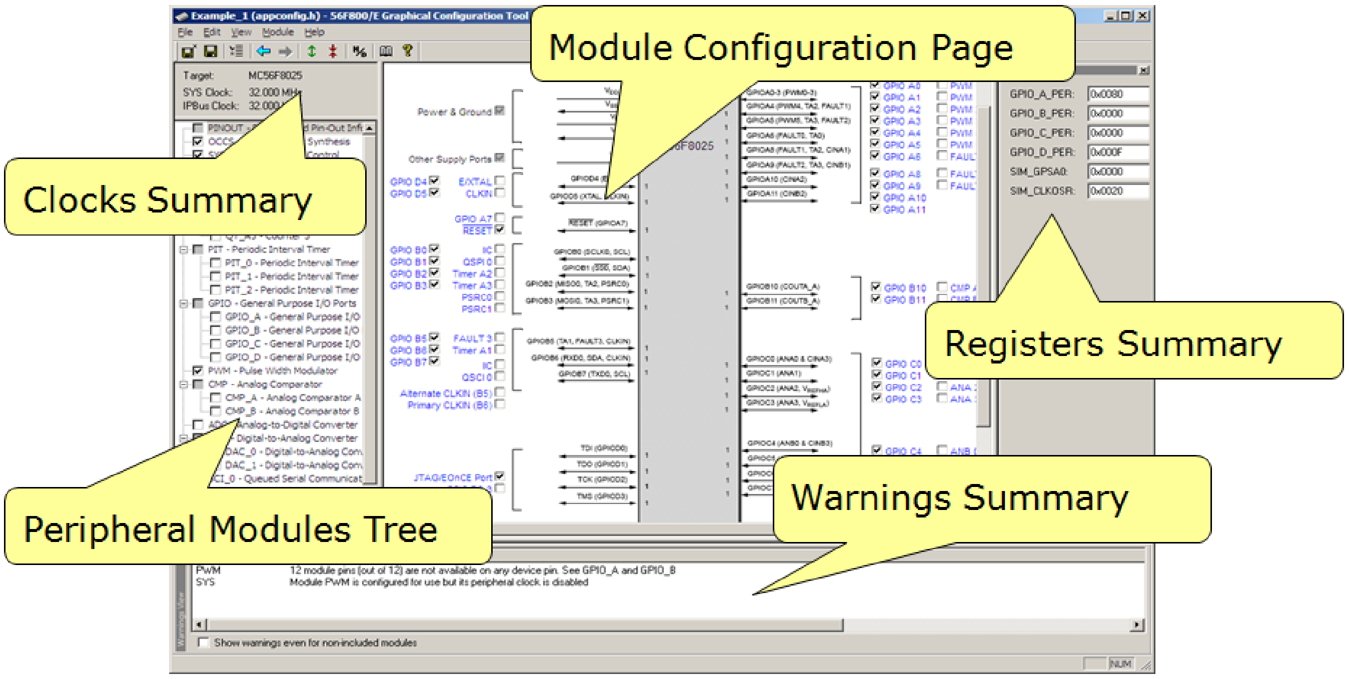


Figure 1. Graphical Configuration Tool

2 Core-system infrastructure

The core-system infrastructure creates the fundamental environment for the processor operation and enables further integration with other components, such as low-level drivers. For more information about the core-system infrastructure, see Chapter 2, "Core System Infrastructure" of [2]. The core components are as follows:

- Common C types and macros: All Quick Start declarations use the same set of types. It is up to the user either to adopt these variable types, use his/her own, or even use the native C types for an application development.

- ArchIO peripheral register structures: A single master structure describes all peripheral registers of the target device. The global symbol ArchIO provides a C interface (structure type) to all peripheral and core registers mapped in data memory. This mechanism increases code readability and portability and simplifies access to the registers. The ArchIO is declared in the C header file arch.h.

- Project templates for creating a new project in the CodeWarrior framework: Such templates are referred to as the "project stationery" and contain complete projects ready to be used on the selected devices and development boards.

- Debugger and linker configuration files for each device: As mentioned earlier, the debugger and linker files are specific for each device and each target in the Quick Start projects.

- The startup code and interrupt vector table are both graphically configurable from within the GCT. These two components are also part of the project stationery and are always copied into each new project created with Quick Start.

3 Quick Start low-level drivers

The peripheral drivers isolate the hardware-specific functionality into a set of driver commands with a well-defined API. The API implements a thin abstraction layer between the software and the hardware levels. Such an isolation enables a sufficient level of portability or architectural and hardware independence for the application code. It also allows separating the C application from hardware particularities that are often tough to code in C, such as access to registers with read or write side effects or interrupt flag clearing mechanisms.

The Quick Start low-level drivers give full control and access to all processor resources. Although possible, the registers are not accessed directly. The low-level drivers unify accesses to peripheral memory space using the ioctl macros or calls, which are, in most cases, compiled into an optimal assembly code. The general form of the ioctl driver command is:

```
ioctl(peripheral_module_id, command, command_parameter);
```

where

- The `peripheral_module_id` parameter represents the base address of the peripheral module. Predefined symbolic constants, such as SCI_0, PIT, EFPWM etc., are used as such module identifiers. Identifiers can be found in Chapter 5, “On-chip Drivers” of [2] before each driver command table.

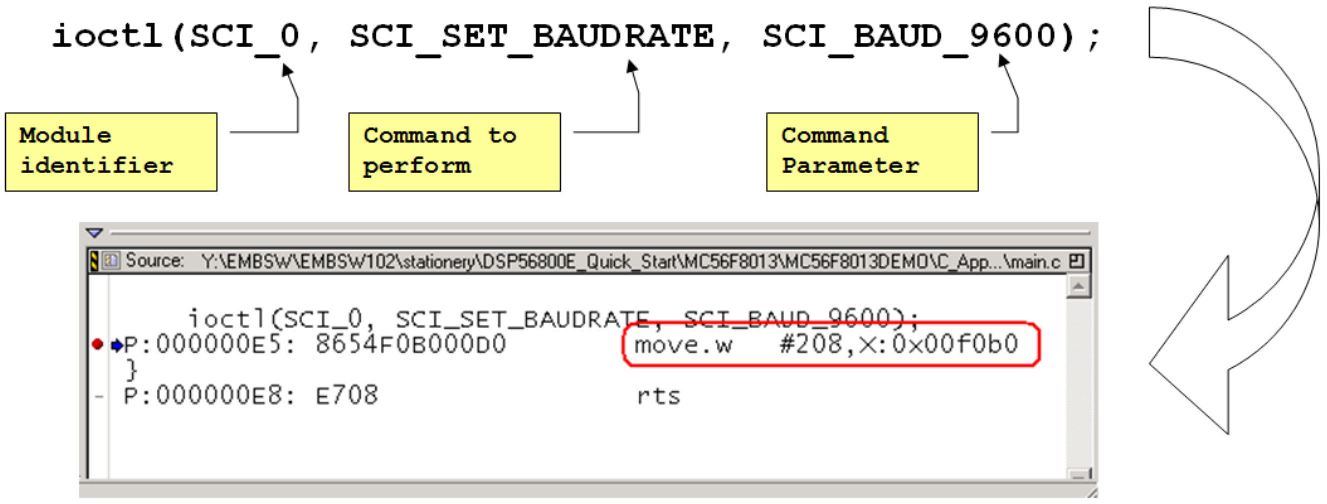
- The `command` specifies the action that will be performed on the peripheral module. The command name gives self-explaining meaning, instead of just assigning a value to a register. Following example commands are defined in pit.h:
  ```
  PIT_INIT
  PIT_ROLLOVER_INT
  PIT_SET_PRESCALER
  ```

- The `command_parameter` then specifies the data required to execute the command, or is NULL when no parameter is required. Following example commands are defined in pit.h:

  ```
  PIT_PRESCALER_1
  ```

```
PIT_PRESCALER_2
PIT_PRESCALER_4
PIT_PRESCALER_8
```

As an example of ioctl command, see Figure 2, which depicts the setting of the baud rate of SCI0 across DSC family.


ioctl(SCI_0, SCI_SET_BAUDRATE, SCI_BAUD_9600);
Module identifier
Command to perform
Command Parameter
Source: Y:\EMBSW\EMBSW102\stationery\DSP56800E_Quick_Start\MC56F8013\MC56F8013DEMO\C_App...\main.c

```
     ioctl(SCI_0, SCI_SET_BAUDRATE, SCI_BAUD_9600);
P:000000E5: 8654F0B000D0          move.w   #208,X:0x00f0b0
}
P:000000E8: E708                  rts
```

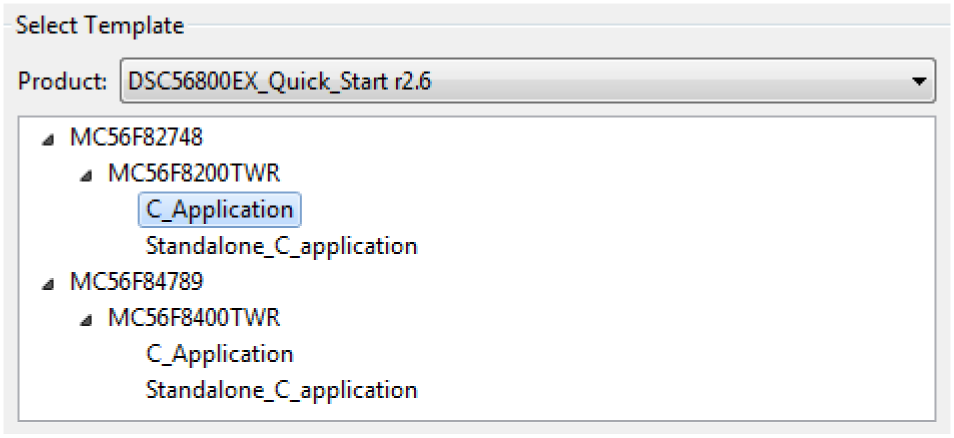
Figure 2. ioctl command which set baud rate on SCI0

A typical example of ioctl command call is shown in Figure 2. There are also more advanced commands in the Quick Start tool, which incorporate some higher functionality rather than a simple access to the peripheral registers. An example can be commands that perform mathematical calculations for data scaling to fit the results into the desired data range, such as recounting of the PWM duty cycle (pulse width) as a percentage of the actual value to be written to the dedicated PWM register.

As already mentioned above, all the peripheral drivers always contain one command which performs a static initialization of the module according to the constants defined in the GCT-edited header file (the file is named appconfig.h). Examples of such initialization commands may be SCI_INIT, ADC_INIT, and so forth.

4 Project stationery

CodeWarrior "clones" the project template and creates a ready-to-use skeleton of a new application. In Quick Start, a dedicated project stationery exists for 56F84xxx and 56F82xxx family processors as shown in this figure.


Select Template

Product: DSC56800EX_Quick_Start r2.6

- MC56F82748
  - MC56F8200TWR
    - C_Application
    - Standalone_C_application
- MC56F84789
  - MC56F8400TWR
    - C_Application
    - Standalone_C_application

Figure 3. New project stationery in CodeWarrior 10.x

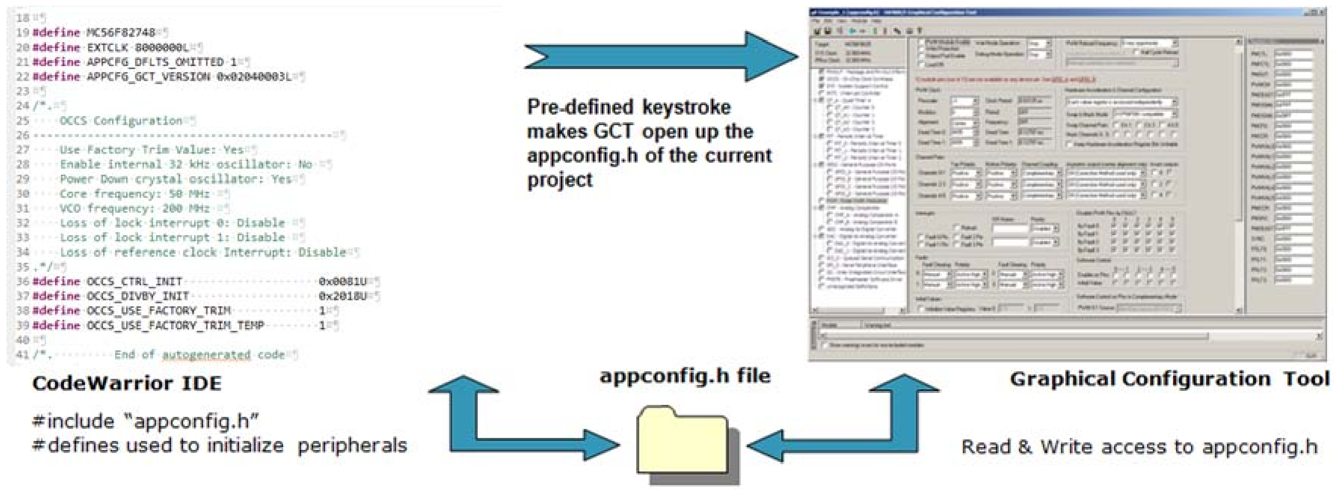
Processors differ in memory layout, peripheral modules, etc. To create a new project based on the DSC56800EX Quick Start project templates (stationery), two options are available:

- Standalone C-application: All the essential driver files are located in the project directory. This option is valuable in case the project source code will be compiled on a PC where QS tool is not installed.
- C-application: The driver files are linked to the project from DSC56800EX Quick Start repository. This option is common when the Quick Start tool is managed locally under a Code Version System.

5 Graphical Configuration Tool (GCT)

The GCT is an easy-to-use graphical interface in which the user defines an initial configuration for all peripheral modules, including the processor core and interrupt vector table. This tool simplifies the configuration of on-chip peripheral modules and the device itself. It also guides the user by supplying a lot of useful information and hints, such as providing a list of available settings and modes for register bits or bitfields. The GCT modifies the application configuration header file appconfig.h. This file is also parsed and used as a native input to the GCT, which enables manual editing of the file, as well as a cut-and-paste method of importing the configuration between different projects.

The interaction of GCT and appconfig.h is depicted on a block diagram in Figure 4.



```
#define MC56F82748
#define EXTCLK 8000000L
#define APPCFG_DFLTS_OMITTED 1
#define APPCFG_GCT_VERSION 0x02040003L

/*
    OCCS Configuration
--------------------------------------------
    Use Factory Trim Value: Yes
    Enable internal 32 kHz oscillator: No
    Power Down crystal oscillator: Yes
    Core frequency: 50 MHz
    VCO frequency: 200 MHz
    Loss of lock interrupt 0: Disable
    Loss of lock interrupt 1: Disable
    Loss of reference clock Interrupt: Disable
*/
#define OCCS_CTRL_INIT                 0x0081U
#define OCCS_DIVBY_INIT                0x2018U
#define OCCS_USE_FACTORY_TRIM          1
#define OCCS_USE_FACTORY_TRIM_TEMP     1

/*          End of autogenerated code
```

Pre-defined keystroke makes GCT open up the appconfig.h of the current project
CodeWarrior IDE
#include "appconfig.h"
#defines used to initialize peripherals
appconfig.h file
Graphical Configuration Tool
Read & Write access to appconfig.h

Figure 4. Interaction between GCT and appconfig.h

The constants generated in the header file may be used by the user in any way. The most convenient way is to use the Quick Start low-level driver initialization calls, which apply the module configuration by writing the values directly into the peripheral registers (for example, GPIO_INIT).

Each peripheral module has its own control page with detailed graphical settings for each bit or bitfield in peripheral registers. Potential configuration conflicts between individual modules are monitored and displayed in the warning window. To understand and resolve all possible conflicts, each configuration window is linked with the particular section in the device user manual. The tool optionally displays register values, making it more convenient for the user to look up an appropriate section in the documentation.

The GCT can work as a standalone tool; however its integration into the CodeWarrior environment increases the tool efficiency. When integrated, the GCT can be invoked simply by pressing a hot key in the CodeWarrior IDE. The GCT then opens and automatically loads the appconfig.h file of the active project.

6 FreeMASTER software drivers

The FreeMASTER is a graphical visualization and control tool which is freely available to be used and redistributed with applications based on the Freescale microprocessor units. The FreeMASTER communication drivers are available for a wide range of Freescale processors as a standalone software pack available for download. For user convenience, the Quick Start tool includes the latest released version of the FreeMASTER communication drivers; so there is no need to download and install them separately. The Quick Start contains two FreeMASTER example projects for both supported Tower boards. See more details about the FreeMASTER tool and the FreeMASTER serial communication driver on the FreeMASTER homepage www.freescale.com/FreeMASTER.

7 Example applications and user documentation

Another important part of the Quick Start tool is a set of sample applications for all supported development boards that are available from Freescale. Although very small and simple, the sample applications show the basic principles of the GCT and demonstrate correct usage of the low-level drivers. The user documentation is designed to be very comprehensive, providing all the information required for successful application development. It starts with a list of guidelines for installing the GCT and low-level drivers and a description of the core system infrastructure. This is followed by explanations for data types, system routines and special macros. The next sections are dedicated to the interrupt processing, on-chip drivers and FreeMASTER interface. All sections include software examples that illustrate discussed topics in C or assembler language.

8 New features in Quick Start version 2.6

- Supports MC56F82748 and MC56F84789 DSC platforms
- Out-of-box support for the new DSC Tower evaluation boards
- Project templates and example applications available for CodeWarrior 10.3
- Discontinued support for MC56F83xx, MC56F82xx and MC56F80xx (version 2.5 is still available for download)
- The latest FreeMASTER driver files included enabling to connect to target device over UART, JTAG or CAN interface.

9 References

The following reference sources are available on freescale.com.

1. DSP56800E Quick Start User Manual, included as a part of DSP56800E_QuickStart installation package
2. DSC56800EX Quick Start User Guide, as a part of DSP56800E_QuickStart installation package
3. DSC Quick Start Initialization and Development Tool

10 Revision history

Revision number	Date	Substantial changes
0	09/2013	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, and CodeWarrior are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc.

Document Number AN4816
Revision 0, September 2013