

AN-8207

Fairchild's Motor Control Development System (MCDS) Integrated Development Environment (IDE)

Summary

The Motor Control Development System (MCDS) is a collection of software and hardware components constructed and developed by Fairchild Semiconductor to facilitate the development of motor control applications. It includes the MCDS Integrated Development Environment (IDE), MCDS Programming Kit, Advanced Motor Control (AMC) libraries, evaluation board, and motor.

The purpose of this user guide is to describe how to use the MCDS IDE to develop the firmware of motor applications. Please visit Fairchild's web for more information:
<http://www.fairchildsemi.com/applications/motor-control/bldc-pmsm-controller.html>.

Features

The MCDS IDE has following features:

- Project Management
- Registers Setting
- Code Generator
- Editor
- Compiler Interface
- Programming
- Debug Mode
- Motor Tuning
- Configure AMC Library

System Requirements

The MCDS IDE requires the following:

- Microsoft® Windows® XP or Windows® 7
- 1 G RAM or greater
- 100 MB of hard disk space
- USB port
- Keil µVision® 3 or above
- MCDS Programming Kit

Getting Started

Software Installation

Please follow steps below to install the MCDS IDE on the computer where development will occur.

Download the MCDS IDE installation files from:
<http://www.fairchildsemi.com/applications/motor-control/blde-pmsm-controller.html>

The Welcome to the MCDS Setup Wizard (See Figure 1) appears on the screen.

1. Click **Next>** to continue (or **Cancel** to exit).

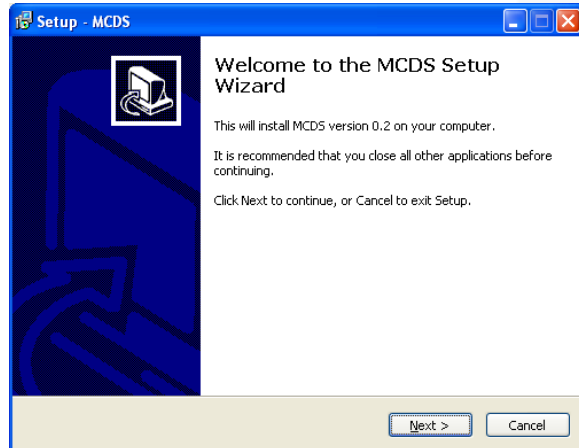


Figure 1. Welcome to the MCDS Setup Wizard

Accept the default location and click **Next>** (or click **Browse...** to change the location before proceeding).

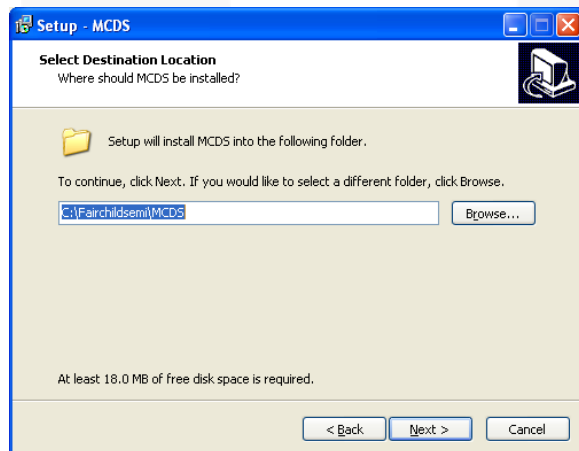


Figure 2. Select Destination Location

The default folder on the Windows® Start menu is "MCDS." To use the default path, click **Next>** to continue (or click **Browse...** to select a different folder).

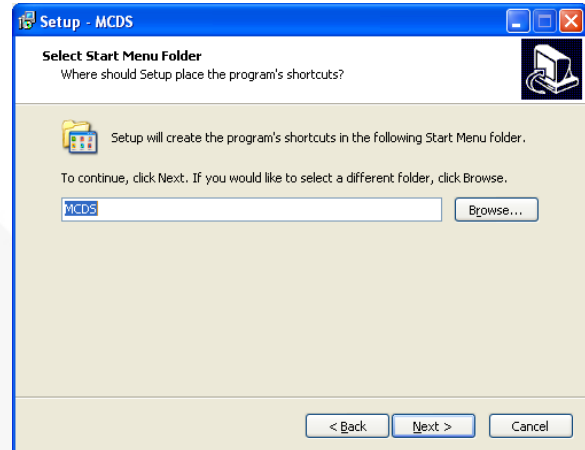


Figure 3. Select Start Menu Folder

When the Ready to Install (see Figure 4) dialog appears, the selections are shown. If correct, click **Install** to continue (or click **<Back** to make changes).

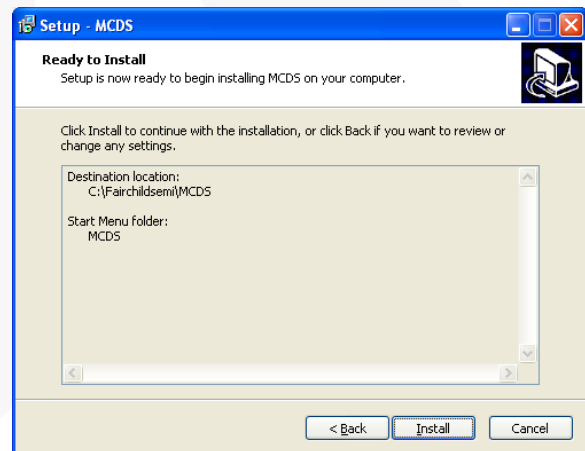


Figure 4. Ready to Install

As the MCDS IDE is installing, the progress bar shows the installation progress (see Figure 5). Please wait while the installation completes (or click **Cancel** to abort the installation).

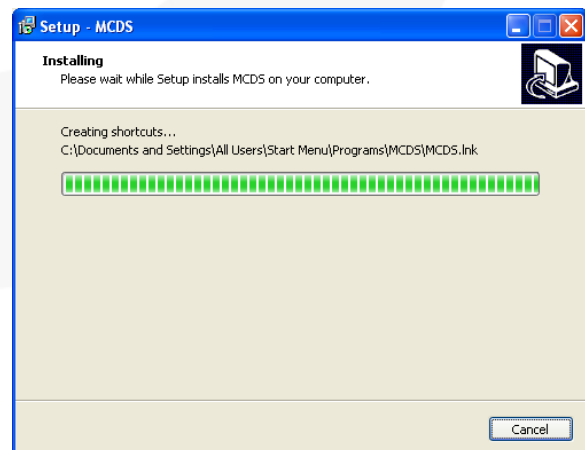


Figure 5. Installing

Once the MCDS IDE is installed, there are two options in the dialog box (see Figure 6): **Install the MCDS Programming Kit USB Device Driver** and **Setup Debug mode of Keil C**. Select both. Both are recommended.

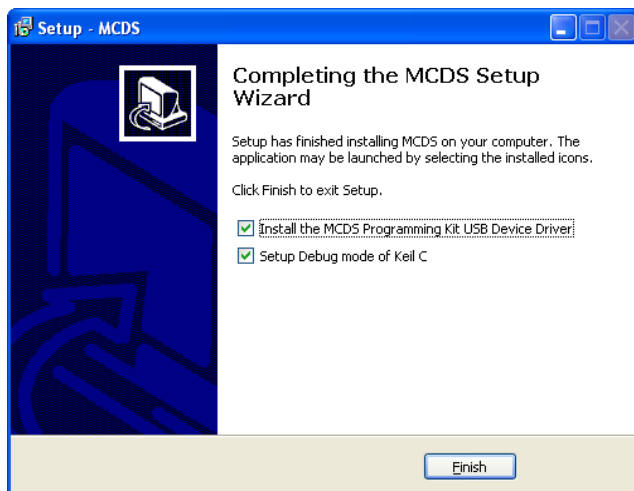


Figure 6. Completing the MCDS Setup Wizard

If the **Install the MCDS Programming Kit USB Device Driver** option is chosen, the Silicon Laboratories dialog appears (see Figure 7). Click **Install** to accept the default installation location (or **Change Install Location...** to select a different installation location).

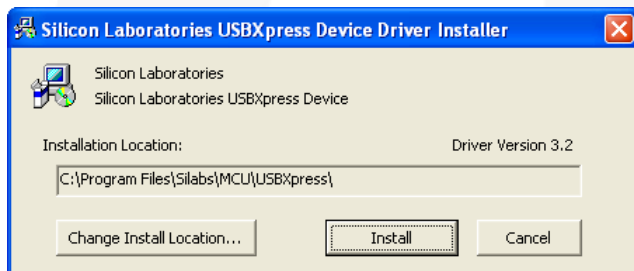


Figure 7. Install MCDS Programming Kit USB Device Driver

When the “Installation Complete” message appears, click **OK**.

If the **Setup Debug mode of Keil C** option is chosen, the installer continues to install the Keil C debug mode (as shown in Figure 8). Choose the correct version of Keil C uVision (3 or 4) and determine the installation directory. Then click **Start Setup**.

Note:

1. The Fairchild MCDS supports both uVision 3 and uVision 4.

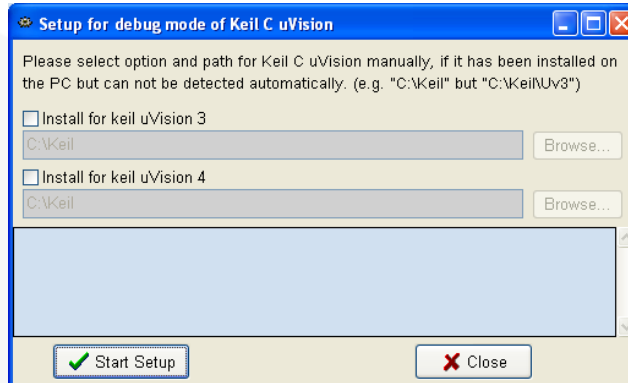


Figure 8. Install Keil C Debug Mode

If the “Setup completed” message appears, click **OK**.

Once the installation is completed, the program shortcuts are created in the **Start** menu. The default directory is **Start → Program Files → MCDS**. The default folders and contents of the MCDS IDE are:

MCDS Main Program

C:\Fairchildsemi\MCDS\MCDS.exe

MCDS Programming Kit USB Device Driver Installer

C:\Fairchildsemi\MCDS\MCDSKit\USBDriver\USBXpressInstaller.exe

Installer of AGDI of uVision of Keil

C:\Fairchildsemi\MCDS\MCDSKit\Setup.exe

Documentation

C:\Fairchildsemi\MCDS\DOC

Note:

2. The folder paths will be different if non-default directories are chosen during installation.

Development Flow

The recommended development flow is as following:

Create a Project

Menu: **Project → New Project**

In the Project Setting window; choose the target device, memory model, and AMC library for the application requirements.

Configure Registers

Menu: **Device → Registers Setting**

The registers are grouped into two sections, the MCS[®]-51 and MotorCtrl. Set those registers that include Timer, Interrupt, I/O Configuration, Watchdog, Interface, PWM, ADC/DAC, and etc.

Generate Code

Menu: **Device → Generate Code**

After the code is generated, the files are listed on the Project Workspace (*see Workspace in the next section*). The source code is stored in the directory where the project is located.

Use Motor Tuning Tool

Menu: **Tools → Tuning...**

The MCDS IDE provides a tool for motor tuning that must be used with the MCDS Programming Kit. To get better performance, the related parameters should be optimized by the motor tuning tool. Those tuned parameters are mapped into the source code after tuning.

Review/Edit Source Code

Double-click the filename on the Project Workspace to open the relevant file. Review the register settings and add codes as necessary.

Build Target

Menu: **Project → Build Target**

Compiles program and generates the programming file. The compiling messages are shown on the Output Window (*see Figure 10*).

Program

Menu: **ISP → Programming...**

Before programming, connect the MCDS Programming Kit to the computer and the target board. Use the Programming function to write the program to the chip on the target board.

Verify and Debug

The MCDS Programming Kit supports On-Chip Debug Support (OCDS), which verifies and debugs firmware.

Review

If the result is satisfactory, the development flow is complete. If not, use the Motor Tuning feature to improve the design.

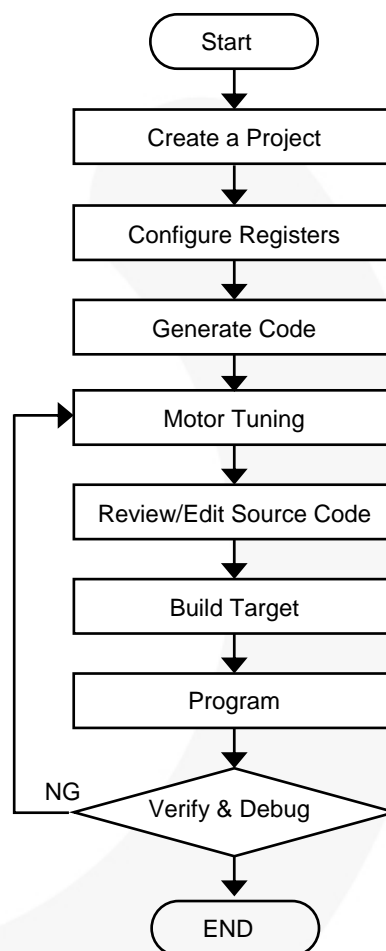


Figure 9. Software Development Flow

MCDS IDE

Overview

The MCDS IDE is a software development platform that runs on Microsoft Windows. It integrates several functions to help develop motor applications.

Supported functions:

- Project Management
- Register Setting
- Code Generator
- Editor
- Compiler Support
- Programming
- Debug Support
- Motor Tuning
- AMC Library Configuration

Workspace

The main window of the MCDS IDE is shown in Figure 10. It is separated to three areas: Project Workspace, View/Edit Workspace, and Output Window.

Project Workspace

- In this workspace, two kinds of tree structure can be viewed, the Registers Setting and the Files. The **Registers Setting** tab contains the register groups in tree structure. The user can click on any item and relevant registers settings open in a pop-up window where the user can set or change the registers. Click the **Files** tab to display the files of the project. In this view, the user can click on any one of files to open the file for editing.

View / Edit Workspace

- The main purpose of this workspace is to display the block diagram of the chosen chip and the opened files of the project. Users can click the tab on the top of this workspace to switch between them. When on the window of an opened file, the user can use a built-in editor to edit the file.

Output Window

Two windows are available: Message and Build. The first shows the message of the MCDS IDE operations. The latter shows the message from the compiling/building project. Click the **Message** tab or the **Build** tab to display the corresponding window.

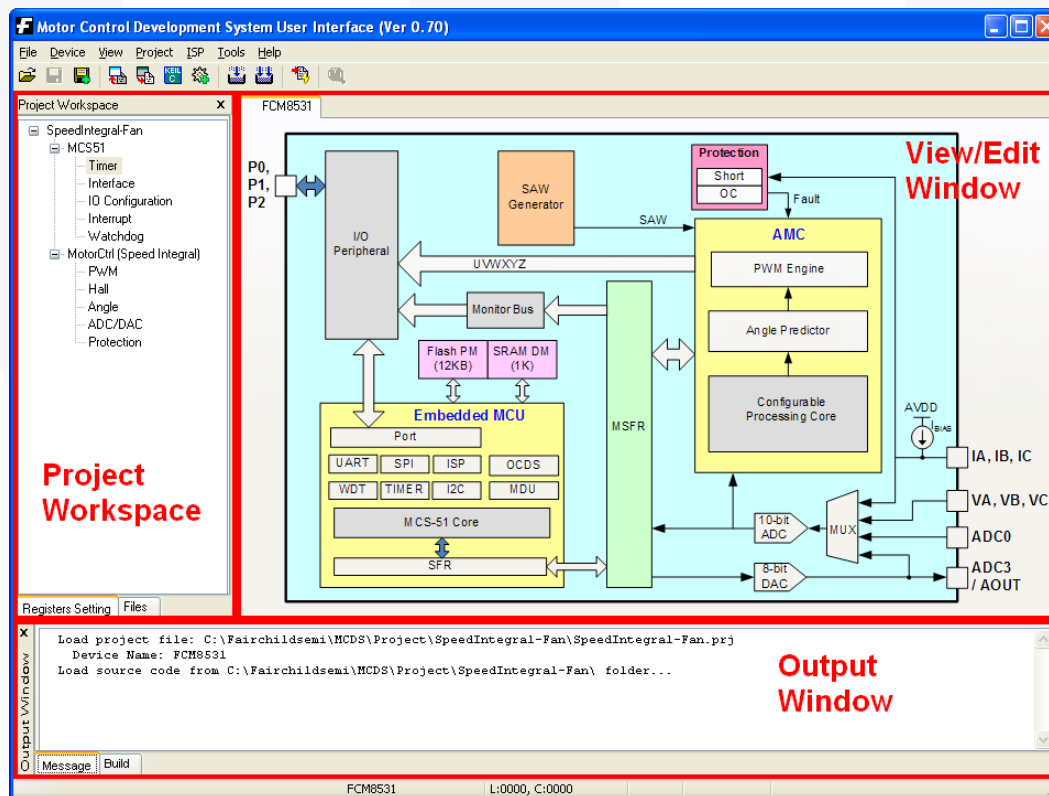


Figure 10. Main Window of MCDS IDE

Menus

The following tables define the commands available from each menu in the main interface.

Table 1. File Menu



File Menu	Toolbar Icon	Shortcut Key	Function
Open File		Ctrl + O	to open a file
Close File			to close the current file
Save File		Ctrl + S	to save the current file
Save As			to save the current file as
Add File to Project			to add file into the project
Exit		Ctrl + X	to exit the MCDS IDE

Table 2. Device Menu




Chip Menu	Toolbar Icon	Function
Generate Code		to generate files into the project, including a Keil C project file and source files (*.c and *.h). Please refer to <i>File Structure</i> . The source codes for registers initialization are stored into appropriate files.
Reload Setting from Code		to synchronize the register setting window with loaded source files
Transfer to Keil C		to start Keil C uVision and open the current project
Register Setting		to open the register setting windows (<i>see Table 3</i>)

Table 3. Resister Setting Sub-Menu

Register Setting Sub-Menu	Relevant Register
Timer	Timer 0, Timer 1 and Timer 2
Interface	SPI, I2C, and UART
IO Configuration	Port 0, Port 1, Port 2, INT12 and Pin configuration
Interrupt	Interrupt and Interrupt Priority
Watchdog	Watchdog Timer
PWM	PWM Clock, Sine-Wave and Square-Wave
Hall	Hall interface
Angle	Angle source, Angle range, and Angle Shift
ADC/DAC	ADC and DAC
Protection	OCH, OCL, Short and Hall

Table 4. View Menu

View Menu	Function
Status Bar	to display or hide the Status Bar
Project Window	to display or hide the Projects Workplace
Output Window	to display or hide the Output Workplace

Table 5. Project Menu






Project Menu	Toolbar Icon	Shortcut Key	Function
New Project			to create a new project
Load Project			to load a project
Close Project			to close the project
Save Project			to save the project
Save as			to save the project as
Project Setting			to set project's parameters
Build Target			to build a project
Rebuild all Target Files		F7	to rebuild a project
Packet the Programming Files			to generate a programming file composed of MCU hex code and AMC library
Toolchain Configuration			to open the configuration window of compiler and linker

Table 6. ISP Menu

ISP Menu	Toolbar Icon	Function
Programming		to program (download) the programming file into chip

Note:

3. The MCDS Programming Kit is needed for this function.

Table 7. Tool Menu

ISP Menu	Function
Tuning	to tune the motor parameters

Note:

4. The MCDS Programming Kit is needed for this function.

Project Management

The main function of project management is to coordinate chip selection, AMC library selection, registers setting, and project files.

- **Project Settings:** selections of chip, memory model, AMC Library, and etc.
- **Registers Setting:** the windows of registers setting.
- **File Management:** to manage all of files (*.c, *.h) of a project. User can manage files in the **Files** tab of Project Workspace.

Project Setting

Project Tab

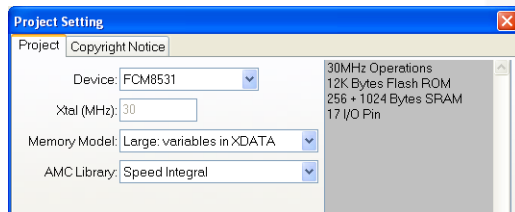


Figure 11. Project Setting Window - Project

Device: select a Fairchild chip for the project from the pull-down menu. If a device is not available, download the latest MCDS IDE from Fairchild.

Xtal (MHz): the operation frequency of the selected device in MHz. (invariable).

Memory Model: three models can be selected from the pull-down menu.

- **Small: variables in DATA:** Small, variables are stored in the DATA section.
- **Compact: variables in PDATA:** Compact, variables stored in the PDATA section.
- **Large: variables in XDATA:** Large, variables are stored in the XDATA section.

AMC Library: to select the AMC library, such as Hall Sensor, Speed Integral, and Sliding Mode.

Note:

5. When considering the selection of the AMC libraries, refer to the device datasheet and the AMC Libraries' user guide.

Copyright Notice Tab

When generating codes, all text in the window (see Figure 12) is inserted into the beginning of each files (*.c and *.h) of a project. For example, user can add the information of copyright notice or version number. Every file or the project has this information in the header.

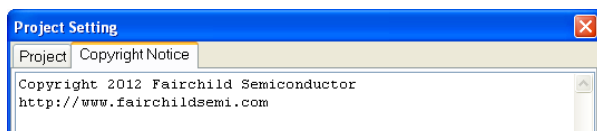


Figure 12. Project Setting Window - Copyright Notice

Registers Setting

Within the Project Workspace, the **Registers Setting** tab provides access to a tree structure of the editable registers. The **Registers Setting** function depends on the chosen device and AMC library. The Registers Setting windows are the interface for editing and configuring the registers. For more detail regarding to the registers of the chosen device, please refer the datasheet and AMC Libraries' user guide at: <http://www.fairchildsemi.com/applications/motor-control/bldc-pmsm-controller.html>.

For example, Figure 13 below is one of the Registers Setting windows. Set the registers via the pull-down menu, check-box, text-box, and etc. Those settings are stored into the files (*.c and *.h) once the code is generated.

When reloading a project, all of the settings in the Registers Setting windows are synchronized with the project.

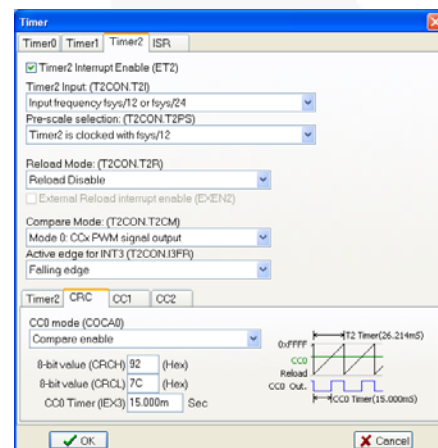


Figure 13. Registers Setting – Timer 2

File Management

Within the Project Workspace, the **Files** tab provides access to a tree structure of the project files and the file management functions of open, add, and remove. Double-click a filename to open the file. Right-click the filename to see available options for the file selected (such as add or remove). Right-clicking the C files or header files can add a file into the project.

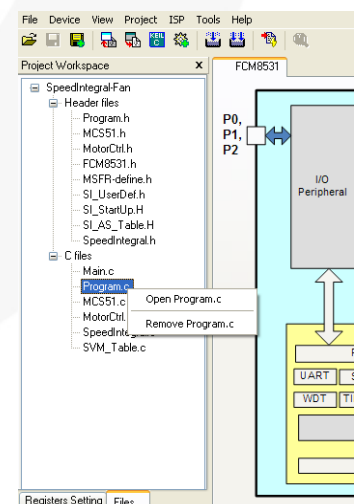


Figure 14. File Management

Code Generator

The purpose of the Code Generator is to convert the configuration of register setting to C code and save them into appropriate files (*.c, *.h) of the project with a pre-defined pattern. The C code is stored in place to initiate

registers following by the sequences (see Figure 15). With the pattern, user can focus on developing the functions to fit the application requirements but the register initiation.

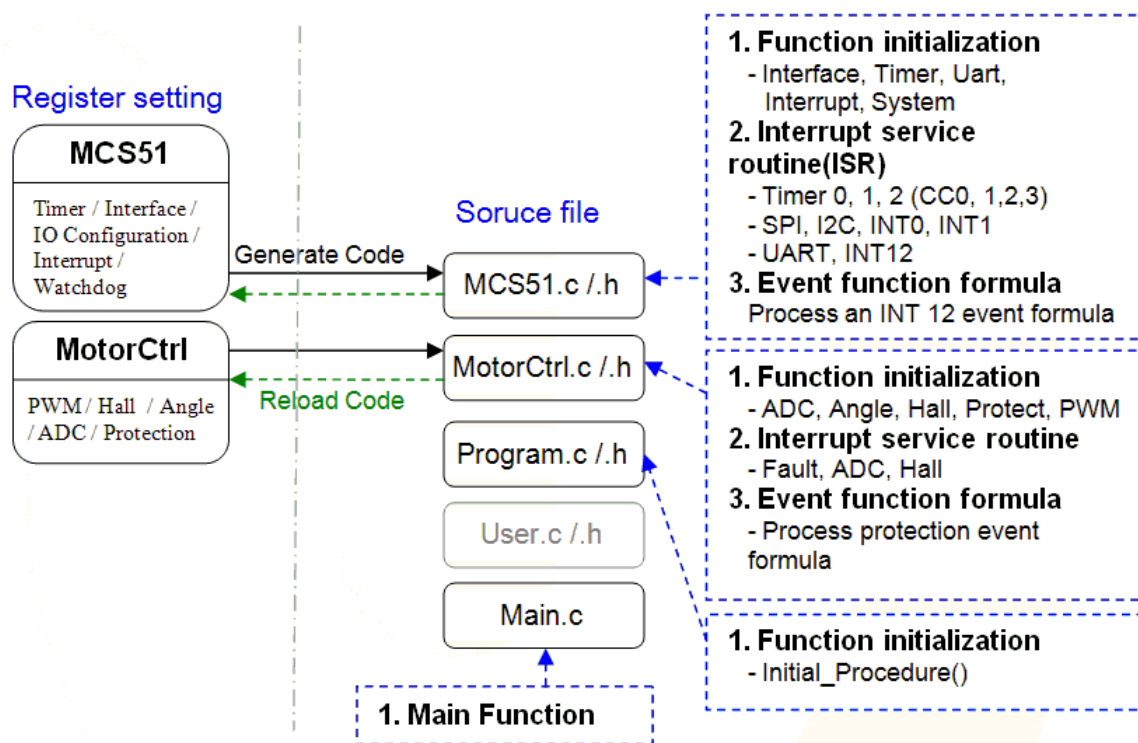


Figure 15. Initial Sequence

File Structure

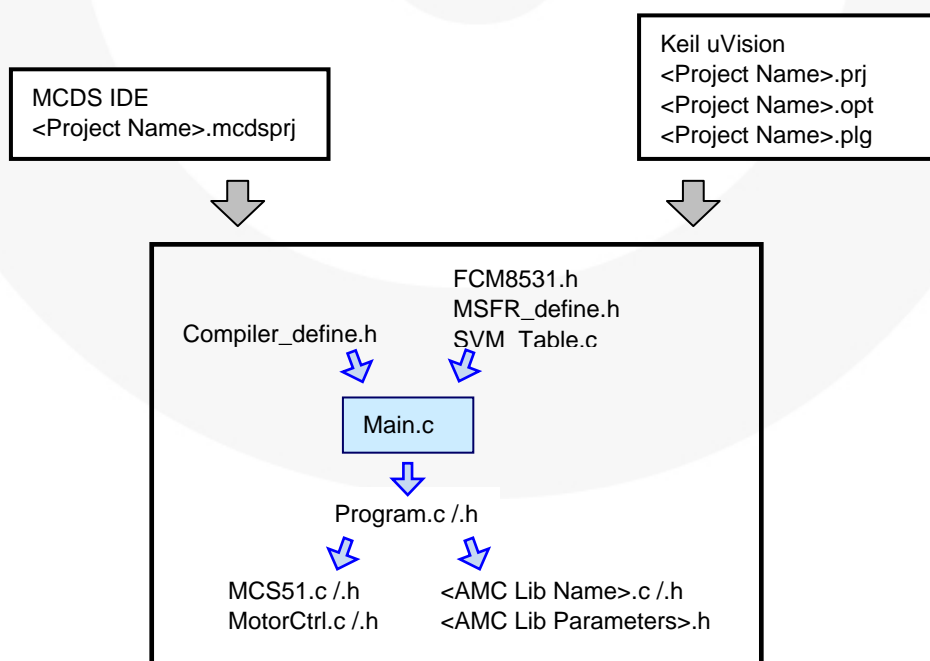


Figure 16. File Structure

File Structure (Continued)

Figure 16 is the file structure of pre-defined template. All of the registers related to MCS-51 are stored in *MCS51.c/h* and the motor control related registers are stored in *MotorCtrl.c/h* (details are below).

Compiler_define.h

Pre-defined common data structure, keywords, and declarations for support of various compilers. *Please refer to Coding Tips.*

FCM8531.h

Pre-defined FCM85xx SFR (Special Function Register) name and address mapping.

MSFR_define.h

Pre-defined FCM85xx MSFR (Motor Special Function Registers) name and address mapping.

SVM_Table.c

Pre-defined customize sine-wave PWM waveform.

Main.c:

Top hierarchy initialization and main loop.

Program.c/h:

Middle hierarchy file between *main.c* and *MCS51.c/MotorCtrl.c*. It links the Initial_Procedure() and initial subroutines in *MCS51.c* and *MotorCtrl.c*. It is also reserved for customized code.

MCS51.c/h:

Codes for register initiation of Timer, I²C, SPI, UART, I/O Configuration, Interrupt, Watchdog Timer, and Interrupt Service Routine (ISR) are in this file.

MotorCtrl.c/h:

Codes for register initiation of PWM, Angle, Hall Interface, ADC, DAC, Protection, Interrupt Service Routine (ISR), and AMC Library are in this file.

<AMC Lib Name>.c/h:

Communication between MCU and AMC and initial procedures for AMC.

<AMC Lib Parameters>.h

Define the AMC library motor parameters value. *Please refer to Motor Tuning.*

All codes are stored in a pre-defined pattern. Fairchild recommends not breaking any of the rules to avoid unexpected errors.

Compiler Setting

All codes generated by the MCDS IDE support various compilers.

Different compilers have various data structures, keywords, and declarations. Therefore, *Compiler-define.h* pre-defines common data structure, keywords, and declarations to unify the usage across compilers.

MCDS IDE currently supports Keil C and SDCC compilers.

```
#if defined SDCC
// SDCC - Small Device C Compiler

#elif defined __C51__
// Keil C51

#endif
```

Figure 17. Conditional Compilation Directives

Code Editor

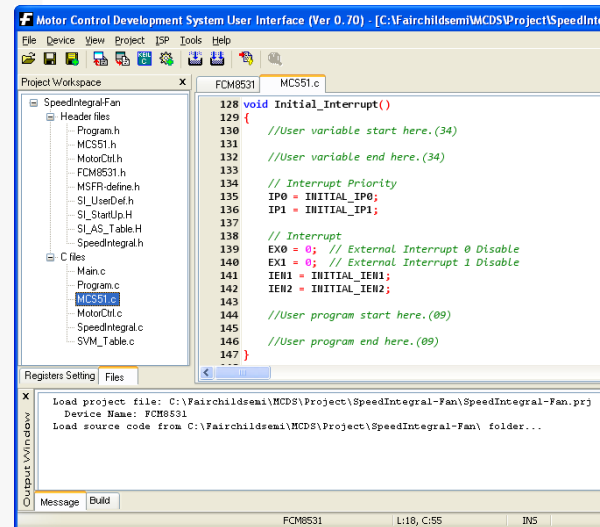


Figure 18. Code Editor

Code Editor allows user to edit the source file and supports the functions of save, close, copy, paste, cut, find, replace, etc. It is similar to any commercial editor. Additional functions are listed below:

- **Line number:** show line number in front of each line.
- **Highlight editing line:** highlight the current line with different background color.
- **Highlight keywords:** highlight keywords with a different font color.

Pre-Defined Macros

The MCDS IDE pre-defines two macros for accessing the MSFR. The purpose of the macros is to simplify the code and makes the code concise. The macros are stored in the *AMC.h* file.

```
#define READ_MSFR(addr, data) {MSFRADR = addr; data = MSFRDAT;}
```

```
#define WRITE_MSFR(addr, data) {MSFRADR = addr; MSFRDAT = data;}
```

For detailed information, please refer to [AN-8204 — FCM8531 AMC Library: Speed Integral](#), [AN-8205 — FCM8531 AMC Library: Hall-Interface](#), and [AN-8206 — FCM8531 AMC Library: Sliding Mode](#).

Coding Tips

The *Compiler-define.h* file pre-defines union: signed/unsigned 8/16/32 bits data structures (see Figure 19). Through usage of the pre-defined union, it can directly access the different significant byte instead of time-consuming shifting (see Figure 20).

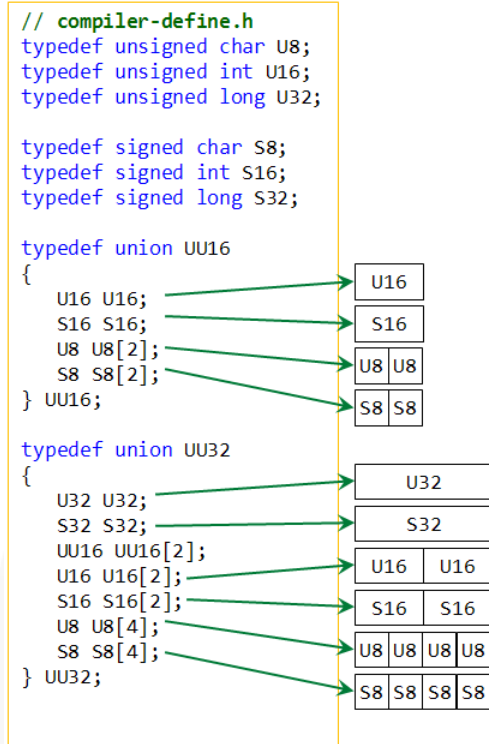


Figure 19. Pre-Defined Unions

```

TH0 = (INITIAL_T0_INTERVAL >> 8) & 0xFF;
TL0 = INITIAL_T0_INTERVAL & 0xFF;

```

↓(The code can be rewritten to)

```

UU16 V;

V.U16 = INITIAL_T0_INTERVAL;
TH0 = V.U8[b1];
TL0 = V.U8[b0];

```

Figure 20. Rewrite Example

Coding Rules

The codes in “Area A” of Figure 21 are generated by Code Generator. Following are the guidelines:

- The codes are synchronized with the Registers Setting windows when generating code.
- The registers’ values are loaded to the Registers Setting windows when a project reloads.
- Fairchild recommends updating registers’ values in the Registers Setting windows rather than manual changes.

The “Area B” of Figure 21 is reserved for customer’s codes. Following are the guidelines:

- Please do not change the comments of “//User program start...” and “//User program end.” These are used to identify the generated codes and customer’s codes. Otherwise, the code generator doesn’t work properly and may cause unexpected error.
- The customer’s codes are stored in between.
- All of codes in between remain the same; no changes are made when re-generating code.

```

void Initial_T0T1()
{
    UU16 V;

    TMOD = INITIAL_TM0D;

    //Set Timer0 Interval
    V.U16 = INITIAL_T0_INTERVAL;
    TH0 = V.U8[MSB];
    TL0 = V.U8[LSB];
    ET0 = 1; //Timer0 Interrupt Enable
    TR0 = 1; //Timer0 Start

    //Set Timer1 Interval
    V.U16 = INITIAL_T1_INTERVAL;
    TH1 = V.U8[MSB];
    TL1 = V.U8[LSB];
    ET1 = 1; //Timer1 Interrupt Enable
    TR1 = 1; //Timer1 Start

    //User program start here.(03)

    //User program end here.(03)
}

```

Figure 21. Coding Rules Example

Access MSFR

Two instructions are needed to access MSFR: the first one is to set the MSFRADR, followed by a read/write to the MSFRDAT. A conflict occurs once an interrupt appears between the two instructions while the ISR accessing the MSFR. To avoid the conflict, Fairchild recommends a backup of the MSFRADR before changing the MSFRADR, and restoring it back at the end of the ISR, as shown in Figure 22.

```

INTERRUPT(ISR_EX0, INTERRUPT_INT0)
{
    U8 T;
    T = MSFRADR;
    //User program start here.(1E)

    //User program end here.(1E)
    MSFRADR = T;
}

```

Figure 22. Access MSFR in the ISR

Building Project

The Toolchain Configuration sets compiler options and linker options. With the settings, the MCDS IDE calls the linked third-party compiler during project building.

The Toolchain Configuration is activated through the menu **Project → Toolchain Configuration...**

Four tabs are set in this window, Assembler, Compiler, Linker, and Convert to Hex. The contents of these tabs are dependent on the **Build Vendor** setting selected from the drop-down list and described below.

Keil C Build Vendor Settings

On the **Assembler** tab, two settings are specified here: the path of the executable file of assembler and its options.

The MCDS IDE calls the *A51.exe* (with full path specified) to assemble the project with the command line parameter “DB EP NOMOD51”. If more options are needed, click **Advanced...** to revise the options. Click **Default** to restore original options.

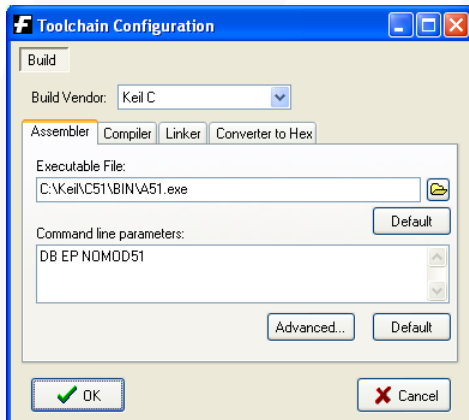


Figure 23. Assembler (Keil C)

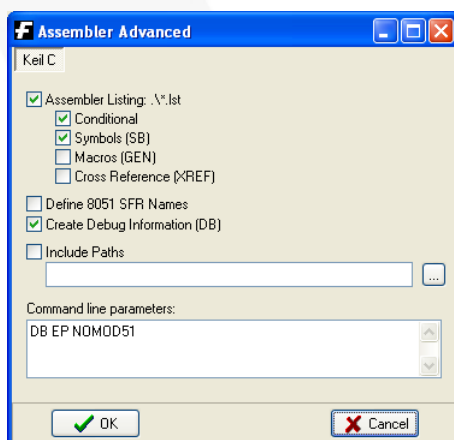


Figure 24. Advanced Assembler Settings (Keil C)

On the **Compiler** tab, the MCDS IDE calls the *C51.exe* (with full path specified) to compile the project with the command line parameter “DB OE BR”. If more options are needed, click **Advanced...** to revise the options. Click **Default** to restore original options.

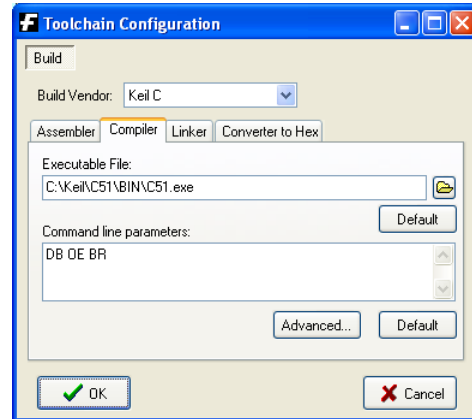


Figure 25. Compiler (Keil C)

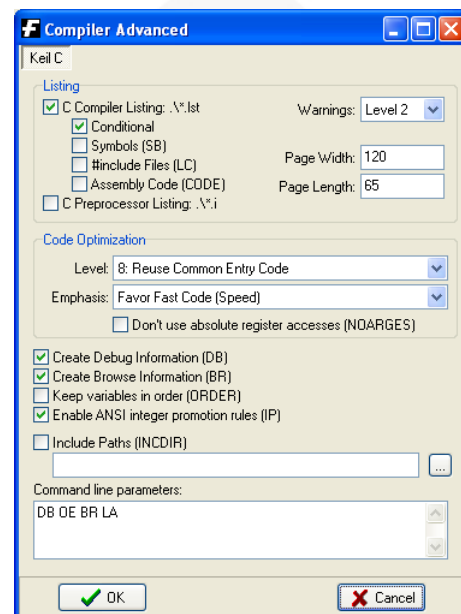


Figure 26. Advanced Compiler Settings (Keil C)

In the **Linker** tab, the MCDS IDE calls the *BL51.exe* (with full path specified) to link the project with the command line parameter “RAMSIZE[256].” If more options are needed, click **Advanced...** to revise the options. Click **Default** to restore original options.

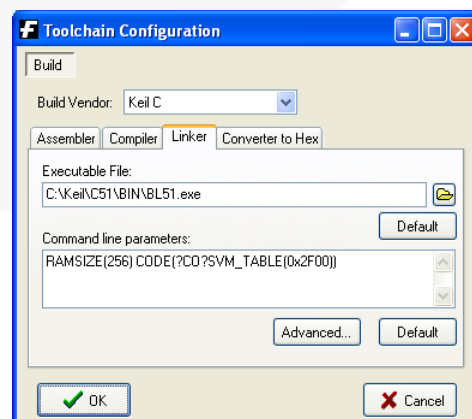


Figure 27. Linker (Keil C)

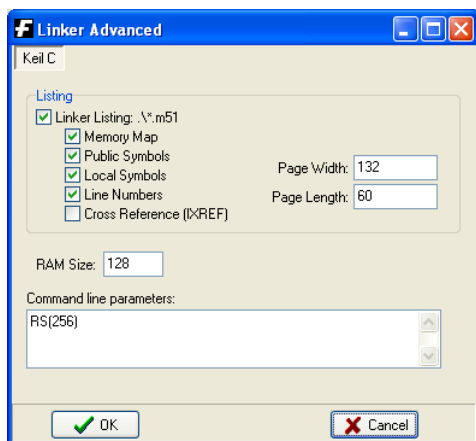


Figure 28. Advanced Linker Settings (Keil C)

The final tab is **Convert to Hex**. The MCDS IDE calls the *OH51.exe* (with full path specified) to convert the results above into a hex file.

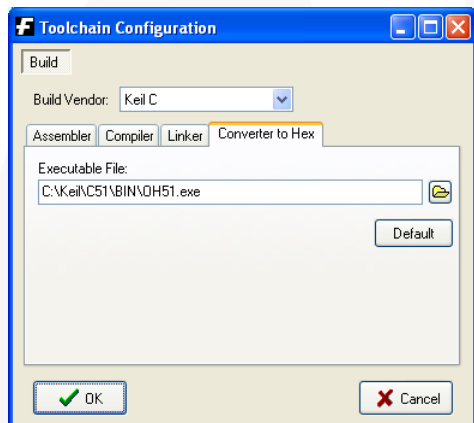


Figure 29. Converter to Hex (Keil C)

Note:

- The executable file in each tab should specify the path in full file-name and its directory.

SDCC Build Vendor Settings

SDCC, an open-source free compiler, can be downloaded from the free software resource website <http://sdcc.sourceforge.net/>. The MCDS IDE also supports the interface for building a project by SDCC. Following is an example for SDCC.

On the **Compiler** tab, the MCDS IDE calls the *sdcc.exe* (with full path specified) to compile the project with the command line parameter “-c -vc.” The user may change the options in the Command Line Parameters textbox.

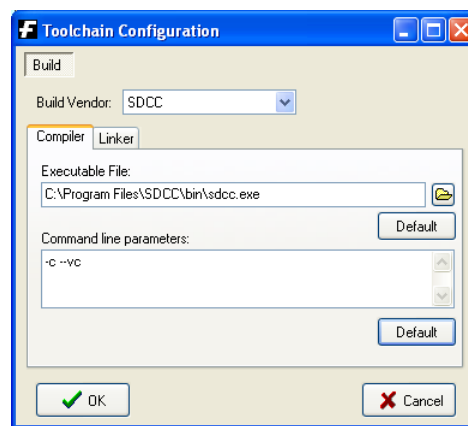


Figure 30. Compiler (SDCC)

On the **Linker** tab, the MCDS IDE calls the *sdcc.exe* (with full path specified) to link the project with the command line parameters “-vc --xram-size 1024 --code-size 0x3000.” The user may change the options in the Command Line Parameters textbox.

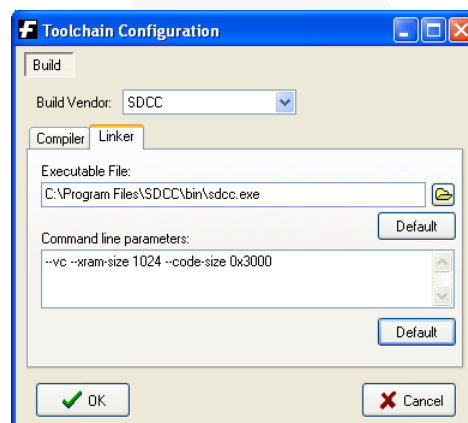


Figure 31. Linker (SDCC)

Note:

- The executable file in each tab should specify the path in full file-name and its directory.

Build Target

A project can be built in the following ways: from the main menu **Project** → **Build Target**, click the **Build Target** button, or press the **F7** short-cut key.



Figure 32. Build Target

Rebuild All Target Files

Two ways: from the main menu **Project** → **Rebuild all target files** or click **Rebuild all target files** button as the figure shown above.



Figure 33. Rebuild All Targets

Programming

The Programming function is one of the ways to write the code into the flash memory of device. Before programming, the MCDS Programming Kit has to connect to a PC and a target board.

Please refer to the MCDS Programming Kit for the guidelines of connecting and refer to Getting Started for installation of the USB device driver.

Device Programming

Select **ISP → Programming...** or click **Programming...** to start Device Programming. The Programming window (see Figure 34) appears. The filename of programming file is in the text-box of **Programming File**. Click the browse button “...” to select another file. Click “**✓Programming**” to start programming the file into the chip.

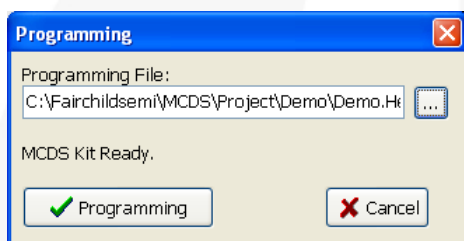


Figure 34. Programming Window

When the “**✓Programming**” button is clicked, a window (see Figure 35) indicates the progress. Please wait... do not close the window until programming is complete.

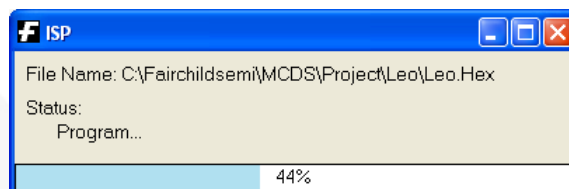


Figure 35. Programming Progress

Programming File

The programming function writes the customer's code and AMC library into the flash memory of the chip. User should packet (package into a single file for consumption by compilers) the programming file before programming when using a third-party IC programmer. **Project → Packet the Programming Files** is the way to packet both the customer's code and AMC library into a file with an extension file-name “burn.” The file format is Intel Hex.

Debugging

The MCDS IDE supports a debugging function with Keil C μ Vision®. Before using the debugging function, make sure Keil C μ Vision is installed on the computer, then follow sections below to set up the debugger's options.

The debugger has three parts: the Debug Mode in Keil C μ Vision, the MCDS Programming Kit, and the target board. The connection between the PC and the MCDS Programming Kit is through the USB port. The connection between the MCDS Programming Kit and the target board is through the flat cable via On-Chip Debug Support (OCDS) interface. The USB device driver should be installed on the computer and the chosen chip must support the OCDS feature.

Please refer to the MCDS Programming Kit for the guidelines of connecting and refer to Getting Started for installation of the USB device driver.

Debug Mode Setting

The options are set from **Project** → **Options for Target** '...' of the Keil C μ Vision main menu. The page of debugging options is shown when the **Debug** tab is active. In the right half, select "Use:" then select **Fairchild MCDS Programming Kit Driver** from the pull-down menu.

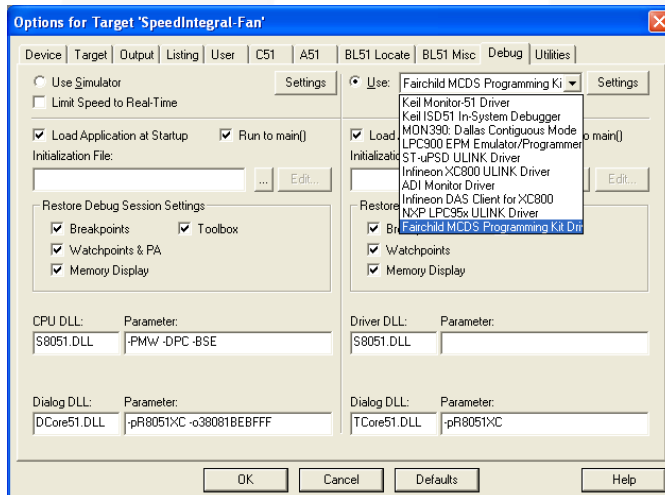


Figure 36. Setup Debug Tool

Click **Settings** to access the window for MCDS Programming Kit Setup. Select the **Cache Code** option, as shown in Figure 37.

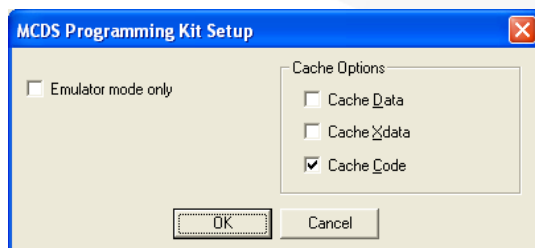


Figure 37. MCDS Programming Kit Setup

Using a Fairchild Chip in Keil C μ Vision

After choosing the **Fairchild MCDS Programming Kit Driver**, the Fairchild Chip Data Base is available and can be selected from the **Device** tab. Click the **Device** tab, then select the pull-down menu of **Databse** (see Figure 38). Select the target chip from the list and press **OK**.

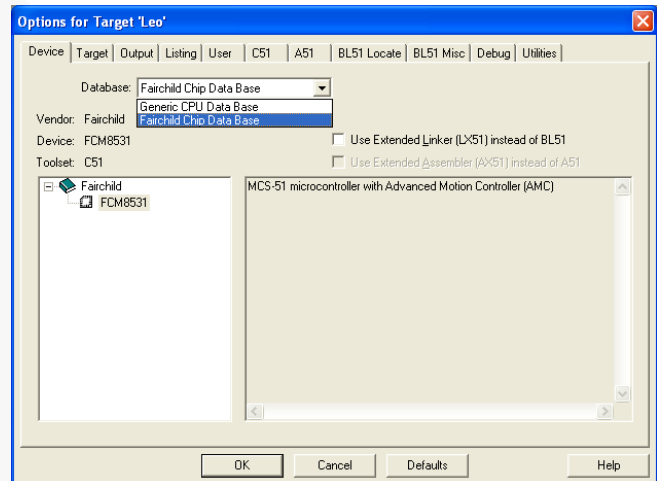


Figure 38. Select Fairchild Chip Data Base

Debugging in Keil μ Vision

Return to the main window, then click the **Debug** → **Start/Stop Debug Session** from the main menu to enter the Debug Mode, as shown in Figure 39.

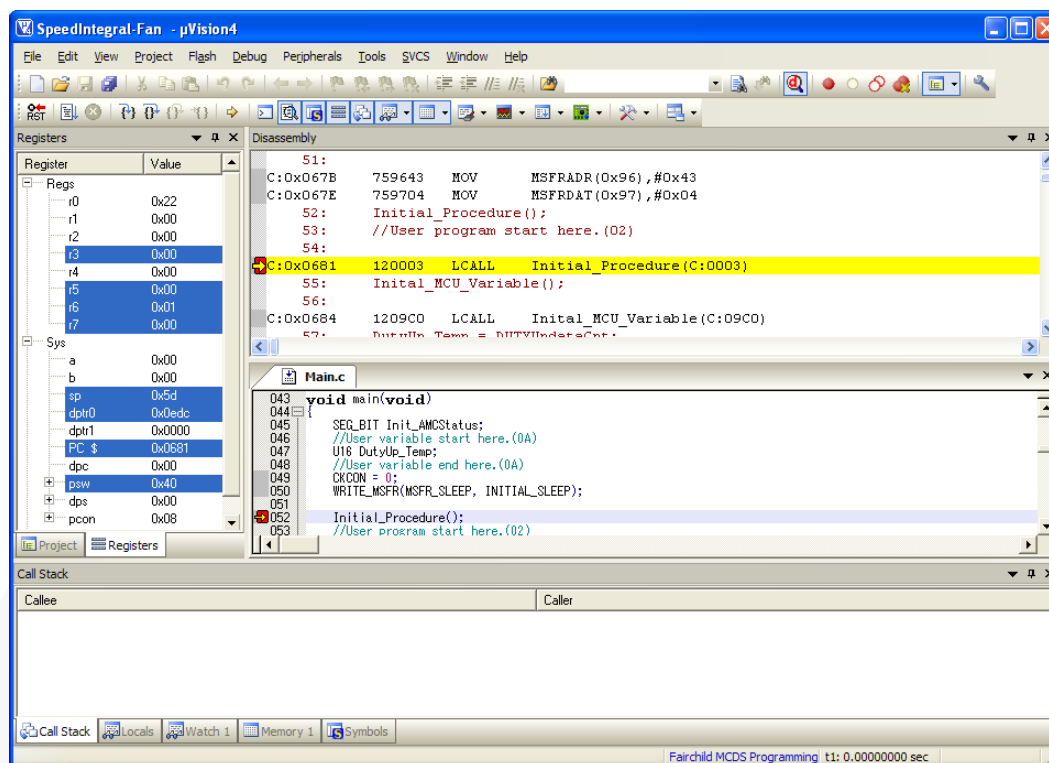


Figure 39. Debug Mode

Review Debugger Settings

The debugger needs the Keil C uVision, MCDS Programming Kit, and USB device driver to properly perform debugging functions. If any one of them is missing, an unexpected error may occur. If the debugger works improperly, the steps below might help to solve the issue.

- Verify the USB device driver is installed properly or re-install it. *Please refer to Software Installation to install the USB device driver.*
- Verify the Keil C Debug Mode is installed properly or re-install it. *Please refer to Software Installation to install the Keil C Debug Mode.*
- Verify the cables are connected properly. Turn off the power of the target board and connect the flat cable of the MCDS Programming Kit to the target board, then connect the USB cable with PC. *Please refer to MCDS Programming Kit to connect the cables.*

- Turn on the power of the target board.
- Verify the project is opened correctly or re-open the project in Keil C uVision by selecting **Project → Open Project**, then selecting the correct project. It is also possible to use the Transfer to Keil C function of the MCDS IDE.
- Verify the Debug Mode settings are correct. *Please refer to the Debug Mode Settings and Using a Fairchild Chip in Keil C uVision to set the Debug Mode.*
- *Please refer to the user's manual of Keil C uVision for the operation of the debugger.*

Note:

8. The MCDS Programming Kit is a non-isolated design. Please pay attention to the devices connected to the target board; including the PC, PC monitor, Oscillator Scope, DVM, etc. Fairchild recommends using an USB isolator and removing the earth ground wire of the power plugs connected to the target board to avoid the damage to the board and devices.

Appendix

MCDS Programming Kit

Overview

The MCDS Programming Kit supports two functions: the In-System Programming (ISP) and the On-Chip Debug Support (OCDS). The block diagram is shown in *Figure 40*.

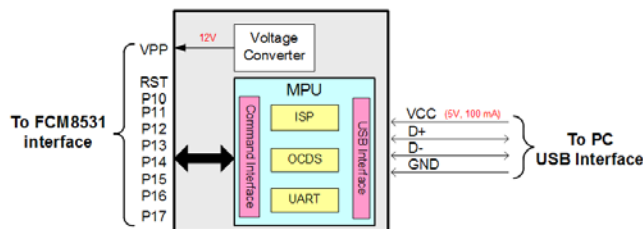


Figure 40. Block Diagram of MCDS Programming Kit

Connection

The MCDS Programming Kit includes a PCB of the MCDS Programming Kit and two cables: a USB cable and a flat cable. The USB cable is used to connect the PC with the MCDS Programming Kit and the flat cable is used to connect the MCDS Programming Kit with the target board. The connections are shown in *Figure 41*.

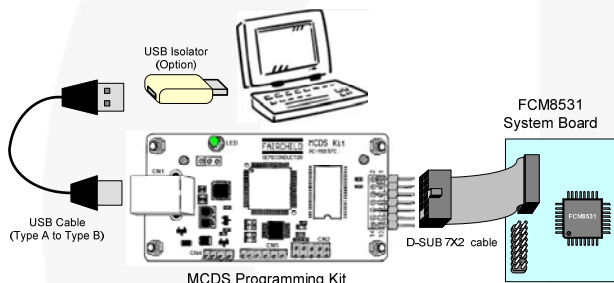


Figure 41. Connection of MCDS Programming Kit

Note:

- The MCDS Programming Kit is a non-isolated design. Please pay attention to the devices connected to the target board; including the PC, PC monitor, oscillator scope, DVM, etc. Fairchild recommends using an USB isolator and removing the earth ground wire of the power plugs that connected to the target board to avoid the damage to the board and devices.

ISP Pinout

To support the ISP, the pins highlighted in Table 8 are required, and must connect to the corresponding pins of the chip on the target board.

Table 8. ISP Pins

Pin Name	Function	Pin #	Function	Pin Name
UART (Rx)	P10	2	1	+5 V
UART (Tx)	P11	4	3	5 V Input
OCDS (TDO)	P14	6	5	VPP
OCDS (TDI)	P15	8	7	12 V Output
OCDS (TMC)	P16	10	9	P12
OCDS (TCK)	P17	12	11	P13
GND Pin	GND	14	13	RST
				ISP (RST)
				NC
				GND Pin

OCDS Pinout

To support the OCDS, the pins highlighted in Table 9 are required and must connect to the corresponding pins of the chip on the target board.

Table 9. OCDS Pins

Illustration	Function	Pin Position	Function	Illustration
UART (Rx)	P10	2	1	+5 V
UART (Tx)	P11	4	3	5 V Input
OCDS (TDO)	P14	6	5	VPP
OCDS (TDI)	P15	8	7	12 V Output
OCDS (TMC)	P16	10	9	P12
OCDS (TCK)	P17	12	11	P13
GND Pin	GND	14	13	RST
				ISP (RST)
				NC
				GND Pin

Firmware Examples

Fairchild provides several code examples for motor applications. They are the Examples sub-directory under the directory specified during installation.

The code examples were developed using the evaluation board for specific applications and includes the techniques available using the AMC library.

Please visit Fairchild's web site to download the MCDS IDE for latest firmware examples.

Motor Tuning

Fairchild provides motor tuning tools for AMC libraries. Parameters can be tuned through the motor tuning tools, which helps achieve better performance. Please refer to *AMC Libraries User Guides* (application notes AN-8204, AN-8205, and AN-8206).

Related Resources

[*FCM8531 — MCU Embedded and Configurable 3-Phase PMSM / BLDC Motor Controller*](#)

[*AN-8202 — FCM8531 User Manual Hardware Description*](#)

[*AN-8203 — FCM8531 User Manual Instruction Set*](#)

[*AN-8204 — FCM8531 AMC Library: Speed Integral*](#)

[*AN-8205 — FCM8531 AMC Library: Hall-Interface*](#)

[*AN-8206 — FCM8531 AMC Library: Sliding Mode*](#)

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION, OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION.

As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.