MiniDragon+ Trainer

With USB interface for Freescale HCS12 microcontroller family

User's Manual

Revision 1.03

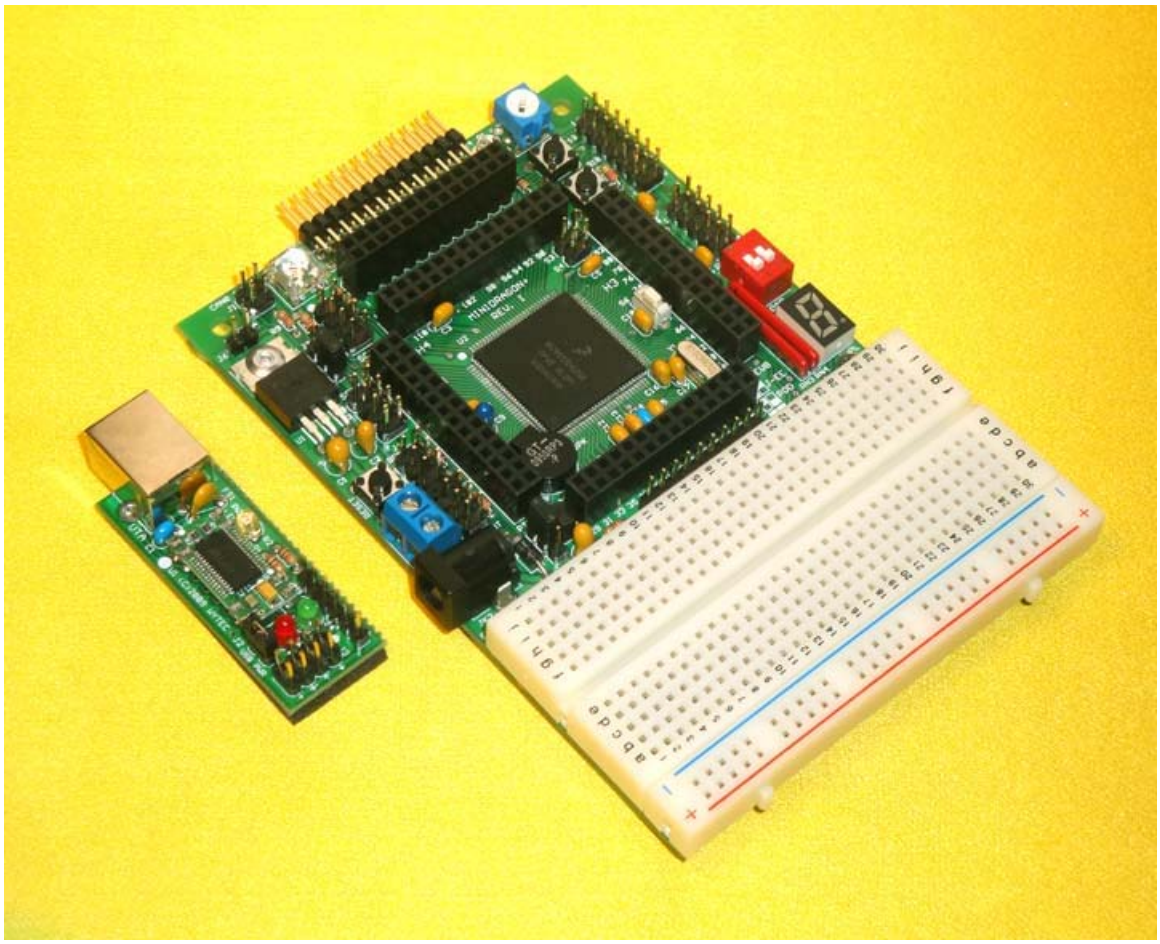


Table OF Contents

Note: For users who will use Code Warrior IDE with serial monitor:

1. This manual is written for the board that is pre-installed with bootloader and D-Bug12 monitor. For some university students, the board is pre-installed with serial monitor and a factory test program. The software installation on the page 12 is not needed. Once the serial monitor is installed the board will not work with AsmIDE or other terminal programs.

   The left DIP switch of S7 has been set in the "up" position as a factory default setting for running the test program. When the board is turned on, it will display a diagnostic code "U-4-3" on the 7-segment LED before running the test program. If it displays "5-4-3" then the left DIP switch of S7 is reset in the "low" position. You have to set it in the "up" position and then press the reset button. The functions of the test program are described in the test16,asm in the CDROM\Code_Warrior\examples. The "U" stands for running User program. After running through the test program you have to reset the left DIP switch of S7 in the "low" position for interfacing with Code Warrior IDE. Oakland university students should follow their professor's instructions to operate the board.

2. When the left DIP switch of S7 is reset in the "up" position, then if you press the reset button, the small 7-segment LED should display a diagnostic code "5-4-3" momentarily. The number 5 means letter S which stands for Serial monitor mode, the number 4 stands for single chip mode and the number 3 means that 2 switches (S3, S4) are open. Here is the code explanation:

   Second number: 4 = Single chip mode, 6 = Narrow expended mode, 7 = Wide expended mode

   Third number: 0 = S3 and S4 closed, 1 = S3 open and S4 closed,

   2 = S3 closed and S4 open, 3= S3 and S4 open

Chapter 1. Introduction

1.1 Welcome

Thank you very much for purchasing the MiniDragon+ trainer. The MiniDragon+ trainer is a low-cost, feature-packed training board for the Freescale HCS12 microcontroller family. It is compatible with the Freescale 9S12DP256EVB board and other similar development boards on the market today, but it also incorporates many on-board peripherals that make this board one of the best trainers in universities around the world.

For engineers, it is a convenient prototype system suitable for designers who want to rapidly develop and prototype new HCS12 applications. For students, it can not only to be used as a general trainer for freshman and sophomore students, but also as a powerful platform for senior projects as well. The compact size and new features of the MiniDragon+ board create a new potential for students at every level.

The MiniDragon+ trainer kit comes with the following items:

1. MiniDragon+ board
2. USB to TTL adapter
3. USB type B cable
4. 110V AC adapter for North America customers

If you miss any part of the kit, please contact sales@EVBplus.com or call 630 894-1440 for help.

Sometime the kit includes two 6-pin IDC connectors at no charge if we have them in stock. You may use them to make a BDM cable in the future.

Please carefully examine the default jumper settings before turning on the board:

1. The J7 should have a jumper installed in the “up” positions so the board will be powered by an external AC adapter.
2. The J15 should have a jumper in the “up” position, so the speaker will be driven by PT5. The speaker can be driven by timer (PT5) or PWM (PP5). It defaults for PT5. Without a jumper installed on J15 the speaker won’t sound.

The specification of the AC adapter is:

DC input:	110V
DC output:	7.5V-9V
Current rating:	300mA-1A
Type of plug:	2.1mm female barrier plug, center positive

The AC adapter is only available to North American customers.

1.2 MC9S12DG256 features and memory map:

The MiniDragon+ board may come with theMC9S12DP256CCPV or the MC9S12DG256CVPE installed. The MC9S12DG256 is the best replacement for the MC9S12DP256 since the latter has been discontinued by Freescale. The only difference between DG256 and DP256 is the number of CAN ports. The DG256 has 2 CAN ports, but the DP256 has 5 CAN ports. Other than the different number of CAN port these two microcontrollers have the same features. If you don't use more than 2 CAN ports these two chips are identical and **all datasheets and manuals** for the DP256 can be used for the DG256.

The MC9S12DG256 microcontroller consists of a powerful 16-bit CPU (central processing unit), 256K bytes of flash memory, 12K bytes of RAM, 4K bytes of EEPROM and many on-chip peripherals.

The main features of the MC9S12DG256 are listed below:

- Powerful 16-bit CPU
- 256K bytes of flash memory
- 12K bytes of RAM
- 4K bytes of EEPROM
- SCI ports
- SPI ports
- CAN 2.0 ports
- I^2C interface
- 8-ch 16-bit timers
- 8-ch 8-bit or 4-ch 16 bit PWM
- 16-channel 10-bit A/D converter
- Fast 25 MHz bus speed via on-chip Phase Lock Loop
- BDM for in-circuit programming and debugging
- 112-pin LQFP package offers up to 91 I/O in a small footprint

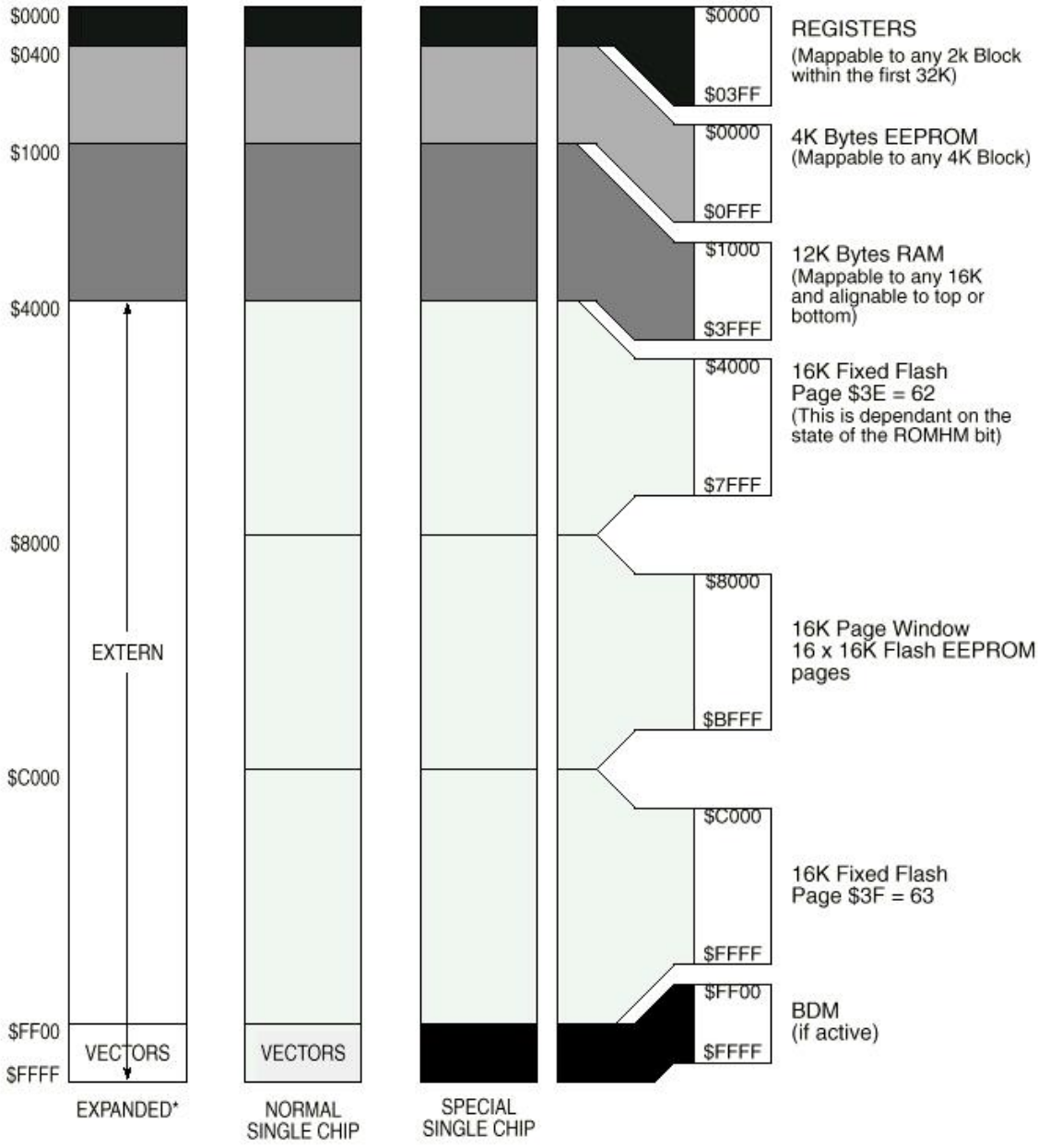


Fig 1-1: MC9S12DG256 Memory map

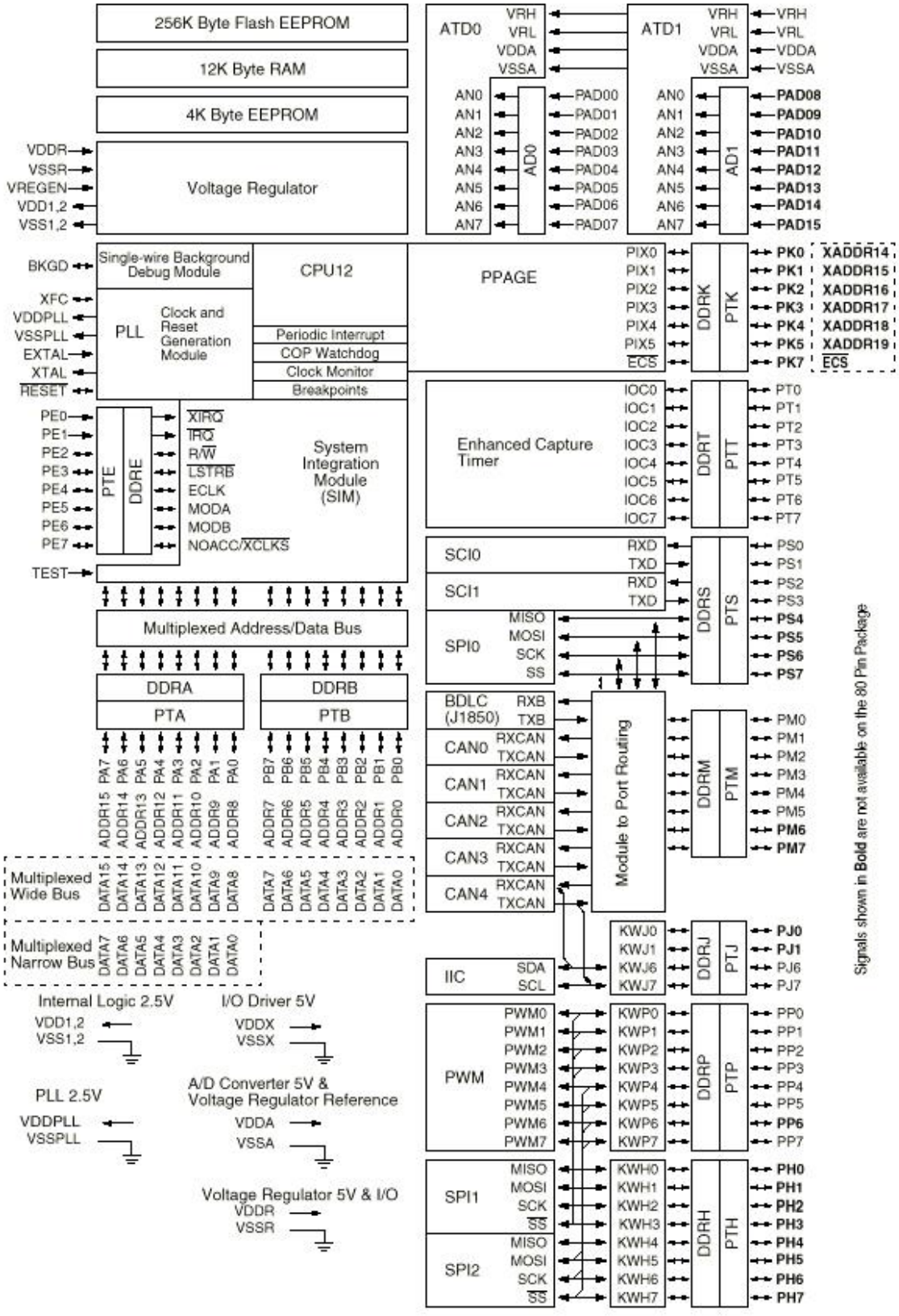


Fig 1-2: MC9S12DG256 MCU block diagram

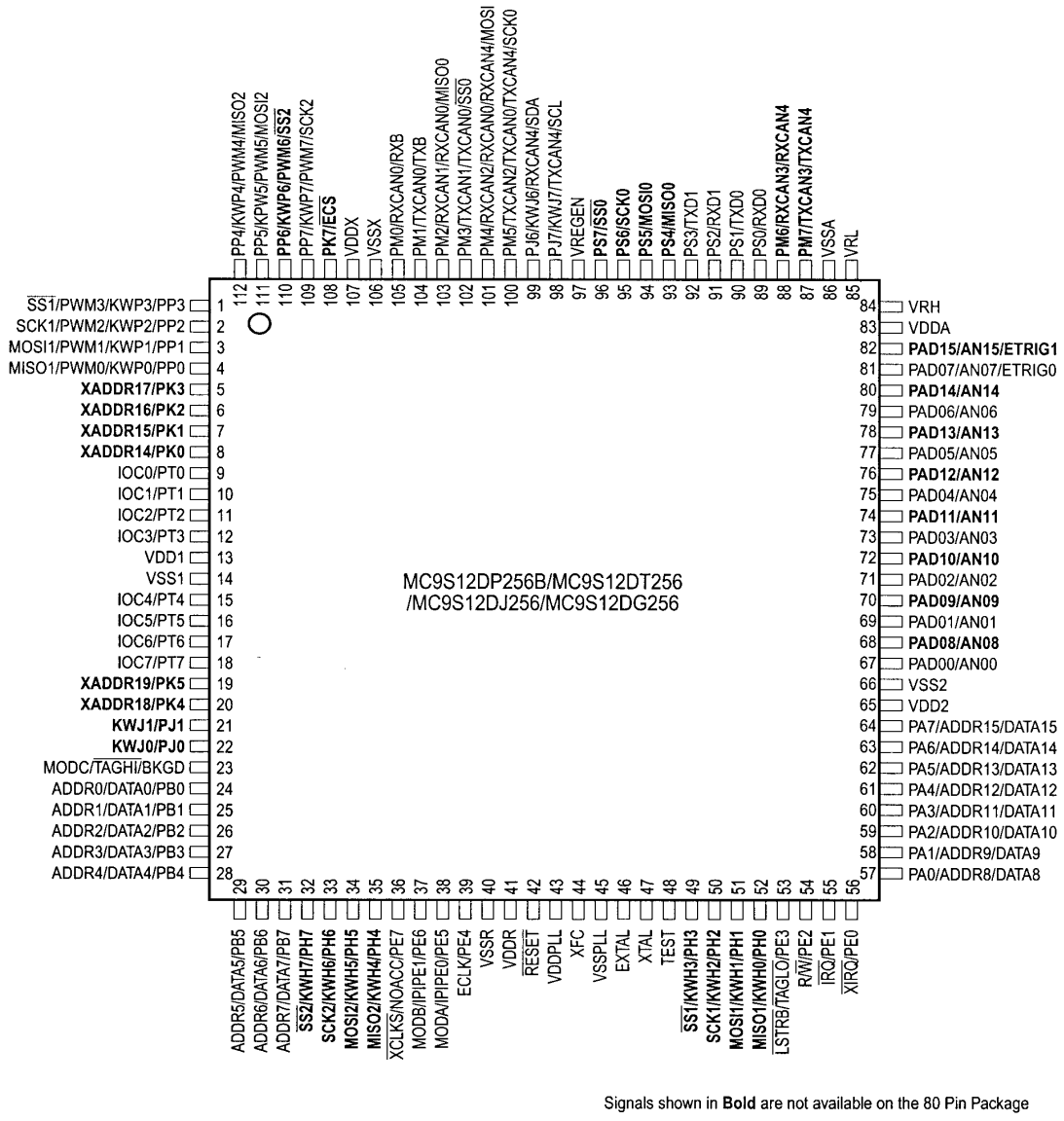


Fig 1-3: MC9S12DG256 MCU pin assignments

1.3 On-board hardware features:

The MiniDragon+ board includes the following features:

1. Dual SCI communication ports
2. CAN port (option)
3. SPI expansion port for interfacing external SPI devices
4. 7-segment LED display
5. RGB tri-color LED
6. Light sensor
7. Two jumper switches
8. Two push button switches
9. 5V regulator with DC jack
10. Speaker to be driver by timer, or PWM signal for alarm or music applications.
11. Power-On LED indicator
12. BDM-in connector to be connected with a BDM from multiple vendors for debugging
13. BDM POD mode for programming other HCS12 boards. No extra hardware needed
14. Abort switch for stopping program when program is hung in a dead loop
15. Mode switch for selecting 4 operating modes: EVB, Jump-to-EEPROM, BDM POD and Bootloader
16. 4 X 4 keypad header
17. Wytec's TinyBee 3-axis accelerometer interface or GP2-D12 distance measuring sensor interface for distance measurement
18. Potentiometer trimmer pot for analog input
19. Female or male headers provides all I/O pins of the MC9S12DG256
20. 400-tie solderless breadboard included
21. Small PC board size 5.25" X 3.40"

The MiniDragon+ board has the following features as options:

22. 4 X 4 keypad
23. 16X2 LCD

1.4 I/O Pin Usage

Many I/O pins of the MC9S12DG256 on the MiniDragon+ board are used by on-board peripherals, but thanks for the large 112-pin LQFP package, there are still many I/O pins available for your circuits on the breadboard. Also it's unlikely that all on-board peripherals will be used by one application program. So the I/O pins on unused peripheral devices can still be used by your circuits on the breadboard. For instance, if you don't connect a keypad, the entire port A will be available to your circuits. If you don't use LCD, the port K will be available as well. Port B drives H-Bridge, but if you don't connect a motor, the port B can drive any other I/O devices on the breadboard.

Pin Name	Pin #	I/O Usage
PA0 (output)	Pin 57	Col_0 of keypad
PA1 (output)	Pin 58	Col_1 of keypad (output)
PA2 (output)	Pin 59	Col_2 of keypad (output)
PA3 (output)	Pin 60	Col_3 of keypad (output)
PA4 (input)	Pin 61	Row_0 of keypad
PA5 (input)	Pin 62	Row_1 of keypad (input)
PA6 (input)	Pin 63	Row_2 of keypad (input)
PA7 (input)	Pin 64	Row_3 of keypad (input)
PB0	Pin 24	not used
PB1	Pin 25	not used
PB2	Pin 26	not used
PB3	Pin 27	not used
PB4	Pin 28	not used
PB5	Pin 29	not used
PB6	Pin 30	not used
PB7	Pin 31	not used
PE0 (input)	Pin 56	Abort switch SW8
PE1	Pin 55	not used
PE2	Pin 54	not used
PE3	Pin 53	not used
PE4	Pin 39	not used
PE5	Pin 38	not used
PE6	Pin 37	not used
PE7	Pin 36	not used
PH0	Pin 52	Segment A on display
PH1	Pin 51	Segment B on display
PH2	Pin 50	Segment C on display
PH3	Pin 49	Segment D on display
PH4	Pin 35	Segment E on display
PH5	Pin 34	Segment F on display
PH6	Pin 33	Segment G on display
PH7	Pin 32	not used
PJ0	Pin 22	not used
PJ1	Pin 21	not used
PJ6	Pin 99	not used
PJ7	Pin 98	not used
PK0	Pin 8	not used
PK1	Pin 7	not used
PK2	Pin 6	not used
PK3	Pin 5	not used
PK4	Pin 20	not used
PK5	Pin 19	not used
PK7	Pin 108	not used

Table 1-1: I/O pin usage list 1

Pin Name	Pin #	I/O Usage
PM0	Pin 105	CAN0
PM1	Pin 104	CAN0
PM2 (output)	Pin 103	EN of LCD module
PM3 (output)	Pin 102	RS of LCD module
PM4 (output)	Pin 101	DB4 of LCD module
PM5 (output)	Pin 100	DB5 of LCD module
PM6 (output)	Pin 88	DB6 of LCD module
PM7 (output)	Pin 87	DB7 of LCD module
PP0	Pin 4	not used
PP1	Pin 3	not used
PP2	Pin 2	not used
PP3	Pin 1	not used
PP4	Pin 112	RGB LED
PP5	Pin 111	RGB LED
PP6	Pin 110	RGB LED
PP7	Pin 109	not used
PS0	Pin 89	SCI0 for PC communication, RECV (RJ11 connector JK1)
PS1	Pin 90	SCI0 for PC communication, XMIT (RJ11 connector JK1)
PS2	Pin 91	SCI1 for user applications, RECV (J2)
PS3	Pin 92	SCI1 for user applications, XMIT (J2)
PS4	Pin 93	not used
PS5	Pin 94	not used
PS6	Pin 95	not used
PS7	Pin 96	not used
PT0	Pin 9	not used
PT1	Pin 10	not used
PT2	Pin 11	not used
PT3	Pin 12	not used
PT4	Pin 15	not used
PT5 (output)	Pin 16	Speaker
PT6 (output)	Pin 17	BDMout reset (used in POD mode only)
PT7	Pin 18	BDMout data line (bi-directional, used in POD mode only)
PAD0	Pin 67	D-bug12 mode select, S7
PAD1	Pin 69	D-bug12 mode select, S7
PAD2	Pin 71	Light sensor
PAD3	Pin 73	Pushbutton S2
PAD4	Pin 75	Pushbutton S1
PAD5	Pin 77	Jumper switch S3
PAD6	Pin 79	Jumper switch S4
PAD7	Pin 81	Trimmer pot VR1
PAD8	Pin 68	X axis input for Wytec accelerometer or ADC input for GP12D2
PAD9	Pin 70	Y axis input for Wytec accelerometer or ADC input for GP12D2
PAD10	Pin 72	Z axis input for Wytec accelerometer or ADC input for GP12D2
PAD11	Pin 74	not used
PAD12	Pin 76	not used
PAD13	Pin 78	not used
PAD14	Pin 80	not used
PAD15	Pin 82	not used

Table 1-2: I/O pin usage list 2

Chapter 2. Quick Start

By default the MiniDragon+ board is pre-installed with the bootloader (Freescale AN2153.pdf) and the D-Bug12 monitor (Freescale DB12RG4.pdf). In chapters 2 and 3 the AsmIDE is used as the main software tool to develop and debug assembly programs. If you prefer to use Code Warrior IDE for C program development and your board is pre-installed, per your request, with the serial monitor (Freescale AN2548.pdf), **skip the chapters 2 and 3 after installing software.**

People often use different terminologies. In our product manuals, **Download** means to transfer a file from PC to a development board, while **Upload** means to transfer a file from a development board to PC. Through out the manual, **left click** means that you click the left button of the mouse and **right click** means that you click the right button of the mouse.

2.1 Install software CD:

After download the CD from our web site the installation is automated by double clicking on the **SETUP.BAT.** It will create a folder c:\MiniDragon+\examples and copy all example program files from the CD to c:\MiniDragon+\examples

If the filename is only shown as **SETUP**, not **SETUP.BA**T, you should change a folder option of the Explorer to show file extension. When a file's extension is hiding, it is hard to know what it is. To have your files to be shown with extensions, click on the TOOL tab in Explorer menu, then click on folder options, then click on view tab, finally un-check the item named 'Hide extensions for knowing file types'.

After the software is successfully installed, you can make a shortcut to AsmIDE.exe on the desktop. It's important to make a shortcut so that its target location is C:\MiniDragon+, not c:\Windows\desktop or other locations. First, right click the Start button, then left click "Explorer", left click on C:\MiniDragon+, right click on AsmIDE.exe (an application program), left click "Send to" and finally left click "Desktop" (do not click "COPY"). It will create an icon named "shortcut to AsmIDE" on the desktop and you can rename it to MiniDragon+. You can double check the target location by right clicking on the icon, then left click on "properties". You should see that the target location is C:\MiniDragon+. If you want to make a shortcut for AsmIDE on the Desktop, this is the correct way to do it. If you don't follow this method, you may have a problem running your program. Never drag the AsmIDE.exe to the desktop folder.

The default setting of AsmIDE for the MiniDragon+ board is created in a text file named c:\MiniDragon+\AsmIDE.ini. In the future if you get lost with all the changes, you always can copy this file into the folder c:\MiniDragon+.

2.2 Getting Started

To operate the MiniDragon+ board, follow steps1 through 5 below:

1. Make sure that the two DIP switches of S7 must be set in the "low" positions for EVB mode, then plug the AC adapter into a wall outlet, and plug the female plug of the AC adapter into the DC jack on the left side of the MiniDragon+ board. After power up, the speaker should chirp once, the 7-segment LED should display the diagnostic code E-4-3 momentarily. The letter E stands for EVB mode, the number 4 stands for single chip mode and the number 3 means that 2 switches (S3, S4) are open. Here is the code explanation:

   First letter: E = EVB mode, J = Jump to EEPROM, P = Pod and b = Bootloader mode.

   Second number: 4 = Single chip mode, 6 = Narrow expended mode, 7 = Wide exp. mode

   Third number: 0 = S3 & S4 closed, 1 = S3 open & S4 closed, 2 = S3 closed & S4 open 3= S3 & S4 open

   If it does not occur check the Power-On LED indicator. The PWR LED is the decimal point of the 7-segment. It is lit when VCC (5V) is present. If the PWR LED is off check the jumper on J7. It should be set in the "up" position so the board is powered by the AC adapter. Also check the output of the AC adapter. It should be about 10V DC without a load (the output DC voltage of the AC adapter is rated for 7.5V at 300mA, but usually the voltage is much higher without a load).

2. Plug the USB cable into the UTA (USB to TTL adapter) and UTA to J3 via a 4x2 IDC cable., then plug the other end of the USB cable into a USB port (not a USB hub) on your PC. The other 4x2 header J2 is the MC9S12DG256's SCI1 port that can be used by a user's program.

   To invoke the AsmIDE, right click the Start button, then left click "Explorer", left click on C:\MiniDragon+ and finally, double left click on AsmIDE.exe. If you have created a shortcut icon on the desktop, just double click the AsmIDE icon on the desktop. **Note:** Always connect the UTA to your PC first before invoking the AsmIDE, otherwise the AsmIDE will not be able to find a COM port. When ending a debugging session, always close the AsmIDE first before disconnecting the UTA. Otherwise the AsmIDE may hang up and you need to re-establish the USB link again.

   This problem will happen to any IDE,

   In case the AsmIDE hangs up, you need to close the AsmIDE first, then pull the USB cable out the USB jack on the UTA, wait for a few seconds before re-plug the USB cable into the UTA. After cycling USB connection, you can invoke the AsmIDE again and it may restore the USB communication. If this does not work, you need to reset your PC, so in order to avoid it, always close the AsmIDE before unplugging the USB cable.

3. The AsmIDE is simple and very easy to use. You only need to use three commands from the AsmIDE for your HCS12 development work. Use the File command to edit your source code, the Build->Assemble command to assemble your source code, and the Build->Download command to download an s19 file to the MiniDragon+ board.

4. The COM port number that the AsmIDE uses must match the USB-to-Serial COM port number that is assigned by Windows O/S. Windows O/S assigns the USB-to-Serial COM port number randomly and it does not know which COM port number that AsmIDE is going to use. In order to find the USB-to-Serial COM port number, you can click through control panel -> systems -> hardware -> device manager -> ports, the USB-to-Serial COM port number will appear (In Windows Vista, you left click on Start, right click on Computer, left click on propriety, then Device Manager and then Continue).

   For setting the COM port of the AsmIDE to match that USB-to-Serial COM port number, you can click through View-> Option->Terminal Window Options menu, then select the correct COM port from COM1 to COM8.

5. Also, set the COM port options at 9600, N, 8,1, and check the “enable the terminal window” box.

6. After reset, the D-Bug12 monitor defaults baud rate at 9600 and Hyperbaud function is disabled. If Hyperbaud function is enabled, the Hyperbaud toolbar button sends the BAUD 57600 command to the D-Bug12 monitor, and then it also changes the serial port to the 57600 baud rate. **IMPORTANT:** When you reset your board it will go back to 9600 baud and you will see characters ‘aaaaaaaaaa’ on the screen. You will need to press the Hyperbaud button once to return AsmIDE to 9600 baud, and press it again to get 57600 baud. To stay at the 57600 baud all the time, you need to press the Hyperbaud button twice after every reset. The Hyperbaud function is disabled by default and it should only be used by an experienced user, not a beginner.

7. You can program text values for function keys to be sent from the terminal window. Some function keys are pre-programmed, but you can change it any time in configuration options (View->Options->Terminal Func Keys).

   In the View->Option->Assembler menu, make sure that the chip family is **68HC12**, not 68HC11. If you would like to use your own assembler, you can replace the as12.exe with the name of your own assembler.

8. The screen is divided into two windows. The top window is for editing your source code and the bottom window is shared by the **message window** and the **terminal window**.

   If the terminal options are set correctly, you should see the following prompt every time the reset button on the MiniDragon+ board is pressed. If you do not see this, the bottom window may be set for message window. Sometime it’s a little confusing when terminal window is disabled and the message window does not display what you have typed. In order to enable terminal window you have to click the terminal button in the bottom window to enable the terminal window display, then move the cursor to any location in the terminal window and click the left button on the mouse. After seeing a solid block cursor flashes, press the <Enter> key and it will enable the terminal window.

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
```

2.3 Test Hardware:

To help users get up and running, the MiniDragon+ board comes with many fully debugged, ready-to-run sample programs including source code. The hardware test program, test.asm, simultaneously scans the keypad, plays a song, changes 7-segment display brightness by adjusting the trimmer pot and vary music playing tempo according to temperature change.

All sample programs must be run from RAM in EVB mode. In order to run the test program in EVB mode, the two DIP switches of S7 must be set in the "low" positions to match the picture for EVB mode.

The steps to run your first sample program are as follows:

1. Click the File button to open the test.asm from c:\MiniDragon+\examples. After the test.asm is loaded into the top window, you can view instructions of how to test all hardware on the MiniDragon+ board.

2. Click the Build button to assemble code and generate the test.s19 file. This is how you normally generate an s19 file. You can omit this step, because the test.s19 is already on your hard disk.

3. Press the reset button on the board, you will see:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
```

4. Type "LOAD" <Enter>.

5. Click the Build button. Select Download option and locate the file 'test.s19' for downloading. If it prompts you with the "save changes?" message, you can ignore that message and click the "No" answer.

6. After download is done, type "G 2000" <Enter> to run the test program.

All sample programs on the CD are developed in RAM. You can try to run a different example program later after you have finished reading this manual. You should always press the reset button before downloading a new program, because the new program may not work if an interrupt was enabled by a previous program.

All example programs are fully debugged, so the assembler won't generate an error. If you have an error, even a warning error, in your program, you must correct it before it can generate an s19 file.

Chapter 3. Software descriptions

3.1 Bootloader and D-Bug12 Monitor

The MC9S12DG256 on the MiniDragon+ board is pre-loaded with bootloader and D-Bug12 monitor firmware and it will operate in 4 different modes depending on the setting of the 2-position DIP switch, S7. After power up or reset, the MC9S12DG256 will read the PAD0 and PAD1 to decide which mode to boot up.

The bootloader (**AN2153.PDF)**, the D-Bug12 reference guide (**DB12RG4.PDF**) and the MC9S12DG256 data book **(MC9SDG256.PDF**) are the most important documentation. They can be found on the folder named C:\MiniDragon+\document after software installation. The HCS12 instruction set, register map and memory map can be found on page 26, 65 and 120 of the data book, respectively.

The new D-Bug12 V4.x is much different and much larger (about 60K) than old D-Bug12 V2.x. The $C000-$EFFF are just a part of the monitor, In 16-bit S1 record they are $C000-$EFFF. In 24-bit S2 record, they are $FC000-FEFFF (ppage=$3F). Since the ppage register deals with the 16K window $8000-$BFFF the addresses $C000-$FFFF are not affected by the ppage. The other part of the monitor is at C0000-C87FF (16K window $8000-$BFFF when ppage=$30,$31 and $32). See details on page 20 of the app note AN2153 or page 71 of the D-Bug12 v4 reference guide on the CD.

3.1.1 EVB mode: PAD1=0, PAD0=0.

This is the standard debug environment running on the MC9S12DG256 for on-chip RAM or EEPROM based code development. Using an IDE program to view and modify registers and memory locations, you may set breakpoints, single step through programs, and assemble and disassemble code as you would in a BUFFALO monitor based Freescale 68HC11 EVB. It gives you 12K RAM and 3K EEPROM to develop and debug your code. You must place your interrupt vectors at $3E00-$3E7F, because real interrupt vector addresses are taken by bootloader, bootloader and D-Bug12 monitor will redirect interrupts to the RAM interrupt vector table at $3E00-$3E7F.

After booting up in this mode you should see the following message on PC screen:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
```

Typing “help” then <Enter> will display a list of available commands.

In this mode, you **cannot** erase or program on-chip flash memory.

If the D-Bug12 monitor is erased you can use bootloader to re-program D-Bug12 monitor into flash memory.

3.1.2 **Jump-to-EEPROM mode:** PAD1=0, PAD0=1

This mode enables the MC9S12DG256 to jump directly to the internal EEPROM at location $0400 upon reset.

This mode makes the MC9S12DG256 a replacement for the old 68HC811E2 microcontroller, but it also gives you 3K EEPROM instead of 2K EEPROM with the 68HC811E2. The bus speed is 8MHz, one half of the crystal frequency by default, the PLL function must be initialized by user's code for a higher bus speed, because the D-Bug12 monitor firmware that boosts bus speed to 24 MHz is bypassed. If you need to auto start your code upon reset, the procedure is available in the folder named eeprom_programming.

3.1.3 **BDM POD** mode: PAD1=1, PAD0=0

In this BDM POD mode, the D-Bug12 firmware acts as a master to access all target MCU resources on the target board (another MiniDragon+ board) via the BDM port in a non-intrusive manner. It becomes a BDM that will have all the features that a standard BDM has in debugging the target MCU. Also, it gains all the features a programmer has for programming the flash memory of the MCU on the target board (another MiniDragon+ board).

To use the master board as a programmer, you need a 6-pin ribbon cable to connect from the BDM OUT of the master board to the BDM IN of the target board (make sure that the orientation of the cable is correct). You don't have to provide the power to both boards, but only to one board. The master board communicates to a PC COM port while the target board does not need to be connected to a PC COM port.

After booting up in this mode you should see the following message on PC screen:

```
Can't Communicate With Target CPU

1.) Set Target Speed (48000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
?
```

You first must set the target speed with the choice 1). After entering the first choice, you will be prompted to enter the target speed. It's the crystal frequency, not the bus speed that is boosted up by the on-chip PLL. After a reset, before the PLL is enabled, the target MC9S12DG256 is running from the crystal frequency, not the PLL frequency. Enter 16000 for the target speed. After the correct speed is entered, the master will try to communicate with the target board. If it's not successful, enter choice 2) to reset the target board.

```
Can't Communicate With Target CPU

1.) Set Target Speed (16000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
? 1

Enter Target Crystal Frequency (kHz): 16000
```

```
Can't Communicate With Target CPU

1.) Set Target Speed (16000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
? 2
```

When the communication is established, you will see the following sign-on message:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"

S>
```

You will notice that the debug prompt is “S>” in the POD mode, not just a “>” in the EVB mode. The S> tells that this is the POD mode and the MC9S12DG256 on target (slave board) is stopped. Sometimes the prompt could be a “R>” that means the target MCU is running. If you see the “R>”, just type “reset” then <Enter> to reset the target and it will come back to the “S>” prompt.

```
R>Reset <Enter>
S>
```

Note: The initial communication in POD mode does not always work smoothly and sometimes the PC screen would only display an incomplete sign-on message. You need to re-start it all over again by pressing reset buttons on both master board and target board, then press the Enter key on PC keyboard. You cannot go to the next step until PC screen shows the prompt ‘s>’.

In order to program the flash memory, you have to erase it by using the FBULK command.

```
S>fbulk <Enter>
S>
```

When the prompt “s>” returns, the FBULK command has already erased all of the flash memory contents of the target MC9S12DG256 including the bootloader. If it returns with a message “Flash or EEPROM Failed To Erase” the MC9S12DG256 is defective.

Now we are going to program the bootloader and the D-Bug12 into the flash memory of the target MC9S12DG256.
Before we actually program the flash memory, we must understand there are two different types of s-record file that can be generated by compilers and assemblers.

An s1-record uses a 16-bit starting address field while an s2-record uses a 24-bit starting address field.
An s1-record file looks like this:

```
S123FFA0F64CF650F654F658F65CF660F664F668F66CF670F674F678F67CF680F684F6883D
S123FFC0F68CF690F694F698F69CF6A0F6A4F6A8F6ACF6B0F6B4F6B8F6BCF6C0F6C4F6C81D
S123FFE0F6CCF6D0F6D4F6D8F6DCF6E0F6E4F6E8F6ECF6F0F6F4F6F8F6FCF700F704F00009
S9030000FC
```

An s2-record file looks like this:

```
S2240FEFA0DB70DB66DB5CDB52DB48DB3EDB34DB2ADB20DB16DB0CDB02DAF8DAEEDAE4DADA41
S2240FEFC0DAD0DAC6DABCDAB2DAA8DA9EDA94DA8ADA80DA76DA6CDDD0DA62DA58DA4EDA4494
S2240FEFE0DA02DA0ADA12DA1ADA22DA2ADA32DA3AD9FAD9F2D9AFD98AD9D5EF00EF00EF0039
S9030000FC
```

We are not going to explain the s-record format here. If you would like to know more on the subject, you can review the D-Bug12 reference guide on the CDROM (**BD12RG4.PDF**). It explains the subject in great details. Right now, all you need to know is that an s1-record file must be converted to an s2-record file before using the FLOAD command. The "FLOAD" command in the D-Bug12 is for downloading an s2-record file.

Our MiniDragon+ bootloader is modified from the Motorola's BootDP256.asm. We added our modification to the original source code and the s record file is generated by the AsmIDE. It's an s1-record file and we converted it into an s2-record file by using the following commands:

Sreccvt –m c0000 fffff 32 –of f0000 -o Boot_MDP_16MHz.s29 Boot_MDP_16MHz.s19

Now we type "FLOAD" <Enter> at the prompt. Click the Build button, select the Download option, and select the file named **Boot_MDP_16MHz.s29** located in the folder named "D-Bug12_Monitor ". You should see the following on the terminal window when programming is done (when the prompt "s>" appears):

```
S>fload <Enter>

**************************************************************************
S>
```

Now we are going to program the D-Bug12 monitor into the flash memory. We need to type "FLOAD" <Enter> at the prompt. Click the Build button, select the Download option, and select the file named **DBug12v32_MDP_16MHz** located in the folder named "D-Bug12_Monitor". You should see the following on the terminal window when programming is done (when the prompt "s>" appears):

```
S>fload <Enter>

**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
**************************************************************************
************************************************************
S>
```

With the bootloader and the D-Bug12 programmed in the flash memory, the target board now becomes a true development board. That's how we program the board before we ship it. Your MiniDragon+ board actually becomes a programmer. You can then repeat above steps as many times as you want. Just unplug the 6-pin BDM cable from the target board, and then plug it into a new target board to program its flash memory with these two files. You even don't have to turn off the power while doing this.

For your convenience, we combined both the bootloader and D-Bug12 monitor into a single s2 file named **Boot_ DBug12v32_MDP_16MHz .s29**. In case you need to update both of them, you can download this combined file.

The D-Bug12 monitor is an application program runs from the bootloader. If you program the D-Bug12 portion of flash memory with your application program, your program will run automatically in EVB mode after power up or reset. When running your code instead of the D-Bug12 monitor, the bus speed is 8MHz, one half of the crystal frequency by default. The PLL function must be initialized by your code for a higher bus speed, because the D-

Bug12 monitor firmware was not in flash memory anymore. For your convenience, we include a PLL code template in chapter 7.

If you need to auto start your code upon reset, the procedure is available in the folder named flash_programming.

3.1.4 **BOOTLOADER** mode: PAD1=1, PAD0=1

This bootloader allows you to erase/program flash memory and erase EEPROM. It is mainly used to program the D-Bug12 monitor into flash memory or download a user's fully debugged code into the D-Bug12 portion of flash memory. The latter allows the board to be operated in EVB mode and start your code every time the board is turned on or reset.

When you program your code into the D-Bug12 portion of flash memory, it wipes out the D-Bug12 monitor. You can restore it any time, just as if you were downloading another application program since the bootloader is not erased. You can erase and program the D-Bug12 monitor portion of the flash memory of the MC9S12DG256 on its own board in bootloader mode, but you cannot erase and program bootloader by itself. **The bootloader can only be erased by an external BDM via BDMin port.**

After booting up in this mode you should see the bootloader menu on PC screen:

MC9S12DG256 bootloader menu:

a) Erase Flash
b) Program Flash
c) Set Baud Rate
d) Erase EEPROM
?

The option a) will erase the D-Bug12 portion of flash memory, not the bootloader itself.
The option b) will program the D-Bug12 portion of flash memory, not the bootloader itself.

The file to be programmed into flash memory must be an s2-record file. If your assembler and compiler generate s1-record files only, you must convert an s1-record file to an s-2 record file before programming flash memory with the bootloader.

The option c) will set a new baud rate.
The option d) will erase all on-chip EEPROM.

Note: Quite a few users would accidentally erase the D-Bug12 monitor when entering this mode, so it's important to know how to re-program the D-Bug12 monitor.

To program flash memory with the D-Bug12 monitor:

1. Enter the option a) to erase D-Bug12 portion of flash memory. Wait until the bootloader menu re-appears after flash memory is erased.
2. Enter the option b), the bootloader will wait for your file. **Do not type** any thing on keyboard.
3. Click the Build button, select the Download option, and select the file named **DBug12v32_MDP_16MHz .s29** located in the folder named "D-Bug12_Monitor" for downloading. You should see the following on the screen:

   ```
   ********************************************************************************
   ********************************************************************************
   ********************************************************************************
   ********************************************************************************
   ************************************
   ```

4. Bootloader menu appears again after the D-Bug12 monitor is programmed into flash memory. It would take a few minutes to program the D-Bug12 monitor so be patient.

3.2 Making a simple assembly program in RAM:

We are using AsmIDE as a terminal program and the following instructions to create your first assembly program. If you are using a different terminal program, the instructions may vary.

The steps to create your first program are as follows:

1. Click the **File** button to open a new file.

   In assembly language, you specify the starting address of your CODE by an ORG statement.

   You can start the data RAM at address $1000 with the statement org $1000 followed by RAM variables, as shown by:

```
            org     $1000

count:      rmb     1           ; reserve one byte of RAM for temp storage
temp:       rmb     2           ; reserve two bytes of RAM for temp storage
```

   If your program is small, say less than 4K, you can start your program at address $2000 with the statement org $2000 followed by your program, as shown by:

```
            org     $2000
```

   It will assemble your source program and generate hex code within 4K locations from $2000 to $2FFF.

   Here is a very simple program, but it's complete. It will light up all 7 segments one at a time when it's running. The RAM byte named 'counter' is added for demonstrating how a RAM data byte is used in a user program. In this simple program it's not really necessary, because the accumulator A can be used as the RAM byte 'counter'.

   For a good programming practice, you should always place the lds instruction in the first line of your code.

```
#include    reg9s12.h
REGBLK:     equ     $0000
STACK:      equ     $2000
;
            org     $1000
counter:    rmb     1

            org     $2000       ; program code
start:      lds     #STACK
            ldx     #REGBLK
            ldaa    #$ff
            staa    ddrh,x      ; make port H an output port
            clr     pth,x       ; turn off 7-segment LED display

begin:      ldaa    #1          ; start with segment A
back:       staa    pth,x       ; turn on display
            jsr     d250ms      ; delay 250ms
            rola                ; go to next segment
            tab
            andb    #$80        ; if it reaches decimal point
```

```
            bne     begin       ; do it all over again
            anda    #$fe        ; force bit0 to 0
            jmp     back
*
d250ms:     pshx
            psha
            ldaa    #250        ; delay 250 ms
            staa    counter
delay1:     ldx     #6000       ; 6000 x 4 = 24,000 cycles = 1ms
delay:      dex                 ; this instruction  takes 1 cycle
            bne     delay       ; this instruction  takes 3 cycles
            dec     counter
            bne     delay1      ; not 250ms yet, delay again
            pula
            pulx
            rts
            end
```

2. Click File button, select Save option to save your assembly source file. Save your file frequently while editing. If you are creating a new file and giving the file a name to save, enter the file name including file extension, such as “Flash_7seg.asm”, not just “Flash_7seg”.

3. Click Build button, select Assemble option, or click the assembler button on the toolbar to assemble your code and generate an s19 file. If the assembler detects an error, the error message will show the line numbers of your source code that caused the error. You have to correct all errors in your program.

4. Go to the line and correct the errors and go back to step 3 until there are no errors.

5. Press the reset button on the board, you will see:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
```

6. Type “LOAD” <Enter>

7. Click Build button, select Download option and locate the file named ‘Flash_7seg.s19”’ for downloading. After download is done, type “G 2000” and hit <Enter> key to run the program.

For your convenience, we have included this sample program, flash_7seg.asm, on the CD.

3.3 Software development

3.3.1 Use on-chip 12K RAM for software development in EVB mode.

You can download your s19 file into the RAM and debug it with the D-Bug12 monitor in this mode. You must place your interrupt vectors at $3E00-$3E7F, because real interrupt vector addresses are taken by the bootloader. The bootloader and the D-Bug12 monitor will redirect interrupts to the RAM interrupt vector addresses at $3E00-$3E7F

Because RAM will lose its contents after power off, you have to load your program every time after power-up. In the beginning of your program, you must initialize the interrupt vectors at $3E00-$3E7F.

In all sample programs, the user program code locations are at $2000-$3FFF. The user data RAM locations are at $1000-$1FFF. The 64 RAM interrupt vector addresses are at $3E00-$3E7F.

The 64 RAM interrupt vector addresses (128 bytes of RAM) are assigned by the D-Bug12 monitor to different interrupt sources. The listing of interrupt sources is show on chapter 8.

3.3.2 Use on-chip 3K EEPROM for testing your code in EVB mode.

If your program is small enough to fit into a 3K range, then you can download your code into the EEPROM. In this way, your program can be auto started from $0400 upon reset. You cannot set software breakpoints and single step in the EEPROM in EVB mode, so it makes sense to do development work in the RAM. When your code is completely debugged, then re-assemble or re-compile it at $0400 and download the final s19 file into the EEPROM for the auto start feature.

Like the RAM-based development, your interrupt vectors are at $3E00-$3E7F. In the beginning of your program, you must initialize the interrupt vectors at $3E00-$3E7F.

3.3.3 Use on-chip flash for testing your code in BOOTLOADER mode.

In this mode, you download your program directly into on-chip flash memory. You first erase the D-Bug12 monitor portion of flash memory, and then program that portion of the flash memory by downloading your application program code in an s29 file. Your program will replace the D-Bug12 monitor in the flash memory. The bootloader portion of the flash memory remains intact. To run your code, set the mode switch S7 to EVB mode, then press the reset button. It usually runs the D-Bug12 monitor, but now it runs your program. The flash memory is non-volatile like the EEPROM. Your code will run every time the board is turned on or reset.

The bootloader redirects interrupts to $EF80-$EFFF. The D-BUG12 is not present and the interrupt vectors of your program are at $EF80-$EFFF. The addresses $EFFE and $EFFF contains the starting address of your program.

In order to program the MC9S12DG256 flash memory, you must program an even number of bytes and begin on an even address boundary for each s-record. If any one s-record in the file contains an odd number of bytes or begins with an odd address, the flash memory cannot be programmed. If your assembler or compiler cannot generate the even format, you must use the Freescale s-record conversion utility **sreccvt.exe** to convert your odd format to the even format by using the following command line:

Sreccvt –m c0000 fffff 32 –of f0000 –o test.s29 test.s19

It will create a new file named test.s29 that has the even format and can be programmed into flash memory. For your convenience, the sreccvt.exe is included in the folder named CDROM\document\Sreccvt-GUI.

Chapter 4: Hardware Descriptions

The crystal frequency is 16 MHz and usually it will result in an 8 MHz bus speed, but on this board the MC9S12DG256's internal PLL boosts the bus speed up to 24 MHz.

The circuits are designed in such way that the value of all resistors and capacitors are not critical and they can be off -50% or +100%.

4.1 Power LED:

The power LED is made of the decimal point of the 7-segment display. It will always be lit when power is applied.

4.2 Jumper switches and pushbuttons:

The two pushbuttons (S1 and S2) are connected to AN04 and AN03. The two jumper switches (S3 and S4) are connected to AN05 and AN06.
S1-----→ AN04
S2-----→ AN03
S3-----→ AN05
S4-----→ AN06
That's the way the PC board was laid out, the AN03 and AN04 should be swapped.
Unfortunately we have to keep the way it is now.

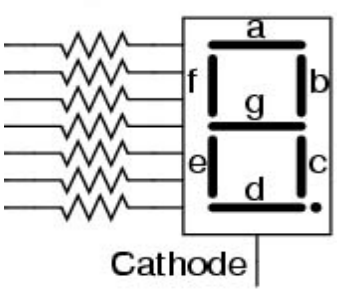
4.3 Seven-Segment LED

The type of the 7-segment LED on the MiniDragon+ board is called common cathode. All anodes are driven individually by an output port and all cathodes are internally connected together.

The MiniDragon+ board uses port H to drive 7-segment anodes. We will explain how to flash the number 1234 on the display

Before sending a number to a 7-segment LED, the number must be converted to its corresponding 7-segment code depending how the 7-segment display is connected to an output port.

By convention, the 7 segments are called segment A, B, C, D, E, F. and G. Their locations in the display are shown below:



The segment A, B, C, D, E, F and G are driven by PH0, PH1, PH2, PH3, PH4, PH5 and PH6, respectively. The hex value of the segment code for common cathode LED display is shown in the following table:

Number	DP	G	F	E	D	C	B	A	Hex Value
1	0	0	0	0	0	1	1	0	$06
2	0	1	0	1	1	0	1	1	$5B
3	0	1	0	0	1	1	1	1	$4F
4	0	1	1	0	0	1	1	0	$66

To flash the number 1234 on the 7-segment display, the following steps should be taken:

1. Output $06 to port H (ldaa #6, staa pth) and the number 1 will be shown on the 7-segment.
2. Delay 250ms.
3. Output $5B to port H (ldaa #$5B, staa pth) and the number 2 will be shown on the 7-segment.
4. Delay 250ms.
5. Output $4F to port H (ldaa #$4F, staa pth) and the number 3 will be shown on the 7-segment.
6. Delay 250ms.
7. Output $66 to port H (ldaa #$66, staa pth) and the number 4 will be shown on the 7-segment.
8. Delay 250ms.
9. Go back to step 1.

4.4 Trimmer pot

The VR1 is connected to the AN07 input of ADC port via J13, but the trace at J13 can be cut if AN07 must be used by target circuits.

4.5 Light sensor

The Light sensor Q1 is connected to the AN02 input of ADC port via J17, but the trace at J17 can be cut if AN07 must be used by target circuits.

4.6 RGB LED

The RGB LED consists of 3 individual common cathode RED, GREN and BLUE LEDs in one single package. They are connected to PP4, PP5 and PP6. The common cathode is connected to ground via J19.

4.7 Speaker

The speaker is a 5V audio transducer and it can be driven by PT5, Output Comparator 5, or PP5, PWM 5. The jumper on J15 is preset for the PT5 at factory and all sample programs on the CD will drive the speaker via PT5.

After reset, the bootloader or the serial monitor will generate a chirp via the speaker. If the jumper is not placed for the PT5, the chirp won't happen.

4.8 Dual RS232 communication ports

Both P1 and P2 RJ11 connectors are configured as **DCE** devices and they can be directly connected to the PC 's COM ports.

The jack JK1 is used by SCI0 of the DG256 while the header J2 is used by SC1 of the DG256. The D-Bug12 monitor or serial monitor works with SCI0, so the JK1 should be connected to a PC's COM port during debugging sessions. The SCI1 can be used by user's application programs. The receiver of the SCI1 can receive signals from many different devices, but only one device at a time, or it will cause a signal collision. The jumper J14 connects the receiver of the SCI1 to the J2. When a jumper is installed on J14, the receiver of the SCI1, (PS2, pin 91 of the MCU) cannot be used for any other device. If you need to use PS2 for your circuits on breadboard, remove the jumper on J14.

4.9 LCD display (optinal)

Port M is an 8-bit bi-directional port. It's used for the LCD display module. If the port is not used for the LCD display, it can be used as a general-purpose I/O port.

The pinouts of J21 are as follows:

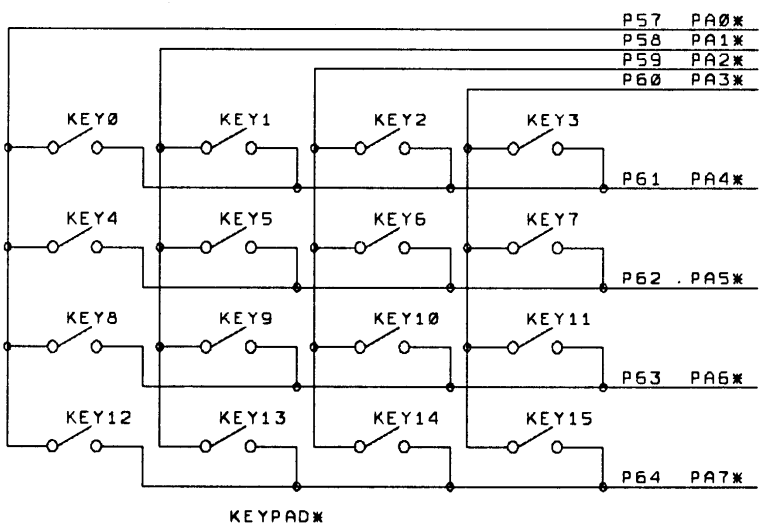
Pin 1	GND	
Pin 2	VCC (5V)	
Pin 3	Via a 220 Ohm resistor to GND	
Pin 4	PM3	RS pin for LCD module
Pin 5	GND	Write only for LCD module
Pin 6	PM2	EN pin for LCD module
Pin 7	Not used	
Pin 8	Not used	
Pin 9	Not used	
Pin 10	Mot used	
Pin 11	PM4	DB4 pin for LCD module
Pin 12	PM5	DB5 pin for LCD module
Pin 13	PM6	DB6 pin for LCD module
Pin 14	PM7	DB7 pin for LCD module

4.10 Keypad (optional)

Port A is an 8-bit bi-directional port. Its primary usage is for a 4X4 keypad. If the port is not used for the keypad, it can be used as a general-purpose I/O.

The schematic for the keypad connections is shown below:

PA0	PA1	PA2	PA3
Col_0	Col_1	Col_2	Col_3



PA4, Row_0

PA5, Row_1

PA6, Row_2

PA7, Row_3

Keypad connections:
- PA0 connects COL0 of the keypad
- PA1 connects COL1 of the keypad
- PA2 connects COL2 of the keypad
- PA3 connects COL3 of the keypad
- PA4 connects ROW0 of the keypad
- PA5 connects ROW1 of the keypad
- PA6 connects ROW2 of the keypad
- PA7 connects ROW3 of the keypad

Keypad scan routine sets PA3 low and PA0, PA1,PA2 high, then tests PA4-PA7.
- If no key is down, PA4-PA7 remain high.
- If PA7 = low, the key 15 is down.
- If PA6 = low, the key 14 is down.
- If PA5 = low, the key 13 is down.
- If PA4 = low, the key 12 is down.

Keypad scan routine sets PA2 low and PA0, PA1, PA3 high, then tests PA4-PA7.
- If no key is down, PA4-PA7 remain high.
- If PA7 = low, the key 11 is down.
- If PA6 = low, the key 10 is down.
- If PA5 = low, the key 9 is down.
- If PA4 = low, the key 8 is down.

Keypad scan routine sets PA1 low and PA0, PA2, PA3 high, then tests PA4-PA7.
- If no key is down, PA4-PA7 remain high.
- If PA7 = low, the key 7 is down.
- If PA6 = low, the key 6 is down.
- If PA5 = low, the key 5 is down.
- If PA4 = low, the key 4 is down.

Keypad scan routine sets PA0 low and PA1, PA2, PA3 high, then tests PA4-PA7.
If no key is down, PA4-PA7 remain high.
If PA7 = low, the key 3 is down.
If PA6 = low, the key 2 is down.
If PA5 = low, the key 1 is down.
If PA4 = low, the key 0 is down.

4.11 All jumpers

All on-board jumpers:

J1 CAN port
J2 SCI1, 2nd SCI interface, TTL logical level
J4 BDM input, pin 1 is the upper left pin
J5 BDM output, when the board is booted in POD mode, pin 1 is the upper left pin
J7 VCC select, from external DC input or USB port
J12 It connects RS of CAN0 (U5) to VSS via a SMD pad on the solder side.

J13 Connects trimmer pot VR1 to the AN07 of ADC and it's a SMD pad on solder side, but the trace between the pads can be cut if AN07 must be used by target circuits.

J14 Connects VR1 to the VRH and it's a SMD pad on solder side, but the trace between the pads can be cut if a different voltage is needed.

J15 Selects speaker driving source. The speaker can be driven by PT5 (OC3) or PP5 (PWM).

J16 CAN receiver connection to PM0 of MC9S12DG256. Remove the solder bridge if CAN0 is not enabled.

J18 Connects a terminating resistor for CAN0. Place a solder bridge on it at the last node in a network. If CAN0 is not used, it will save power consumption of the board by removing the solder bridge.
J19 Connects Common Cathode of the RGB LED to VSS

J21 LCD port
J22 Keypad header
J24 132 right angle male header
J25 13X2 female socket connector.

J26 X-Y-X Accelerometer module interface or IR distance sensor, GP2D12, interface.

Chapter 5: EmbeddedGNU

Eric Engler has published the EmbeddedGNU IDE that supports GNU C compiler and assembler for any 68HC11/HC12/HCS12 boards including our FOX11, EVBplus2, Dragon12 and MiniDragon+ boards. It's free software under Open Source, GNU GPL License. It's not freeware nor shareware (be aware that some freeware are not free). To download Eric's free tools including the GNU C compiler and assembler please visit his web site at: http://www.geocities.com/englere_geo/
For your convenience, we downloaded the egnu094.zip for you.

The following page shows the exact terms of the license (Mozilla Public License)
http://www.geocities.com/englere_geo/License.txt

The steps to set up the EmbeddedGNU are as follows:

1. Download the GNU GCC compiler from: http://m68hc11.serveftp.org/m68hc11_pkg_zip.php
   Select the release 3.1 to download. It has the following components in it:
   Gcc 3.3.6
   Gdb 6.4
   Binutils 2.15
   Newlib 1.12.0

2. Run the file that you downloaded to install GNU 68HC11/68HC12 tools into the default directory of C:\usr.

3. Install the EmbeddedGNU on your PC by double clicking on the egnu094.zip. If the egnu094.zip is not on the CD, you can download it from http://www.ericengler.com/EmbeddedGNU.aspx
   Extract all files into a new directory that you need to create on any hard drive. The name of the new directory can be like c:\egnu094 or d:\egnu094. The EmbeddedGNU.exe and example programs will be located at \egnu094, but your application programs can be located in any other directories.

4. Filename Association.
   When you first start EmbeddedGNU.exe it will ask if you want to associate the filename extensions used by EmbeddedGNU with itself. This lets you double-click on a filename and the EmbeddedGNU will be launched to let you edit the file. The default option is to associate ".prj" with EmbeddedGNU. This is the main project file type used by EmbeddedGNU.

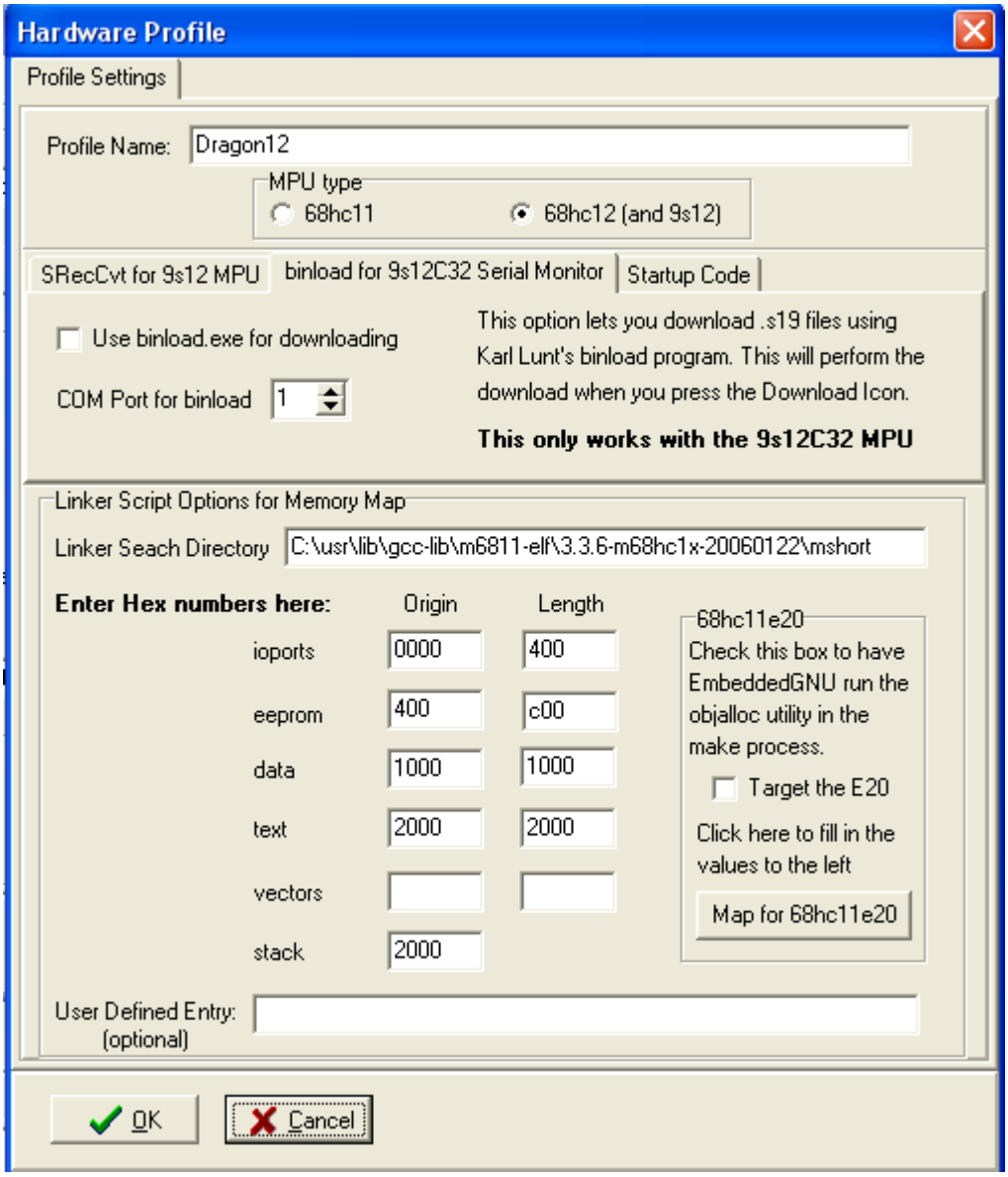
   You also should choose to associate .c, .h, and .s files with EmbeddedGNU.
   WARNING: if you are on WinNT/Win2K/WinXp, then you must be logged in as an administrator to use this option.

   Press OK to continue

5. COM Port Selection.
   It asks if you want to select your COM port. Say Yes. Select your port in the dropdown box. It defaults to 9600 baud, which is normally correct. Now press OK.

6. Select Option-> Environment Options->AutoDownload, then disable ALL automatic commands.

7. The current egnu094.zip is properly set up with the newest release version 3.1 (GCC 3.3.6). In the future when upgrading to a newer version you have to update the linker's search directory. See help file related version upgrade issues.



To change the linker search directory (search path) for GNU C compiler toolset you click on options->project options->edit profile. As it can be seen from above Linker Search Directory, the GCC 3.3.6 is installed on C drive.

Some university web sites offer educational resource for the EmbeddedGNU. The following web site provides A C sample program for the Dragon12 board using EmbeddedGNU and GCC

Chapter 6: Code Warrior and serial monitor

Code Warrior is a very powerful and professional IDE. The main feature of Code Warrior IDE is the source level debugger in assembler and C. Code Warrior Special Edition is a wonderful gift from Freescale to all of us and it's free for educational use. What's more, by Code Warrior supporting serial monitor, they have made it very affordable to support Code Warrior for the OEM.

Freescale has invested millions of dollar into Code Warrior and the current versions work very well. What's more, Freescale knows they will never sell enough copies of Code Warrior to make back what they have invested. They did it to drive chip sales.

As a software developer, the first thing you look at is available tools and what it will cost. There are many companies making MCU chips these days and for the most part they all have about the same features at a similar price. Special Edition Code Warrior sets Freescale apart from others.

Code Warrior IDE does not work with D-Bug12, but it works with serial monitor. Before Freescale created the serial monitor a BDM is needed as an interface between the PC and HCS12. Freescale created the serial monitor for working with Code Warrior to eliminate the cost of a BDM.

Now a student can use the serial monitor with Code Warrior to debug his program and in fact, many universities have been using the serial monitor with Code Warrior without a BDM in their classrooms.

Without spending money on a BDM, a student will be able to spend his savings on purchasing a more advanced trainer, like the MiniDragon+ board with many on-board peripherals. Purchasing an EVB board that comes with a BDM at a reasonable price, most likely leaves the student with an EVB of only limited functionality.

Some universities use D-Bug12 monitor first, then replace the D-Bug12 monitor with serial monitor to be used with Code Warrior IDE. In this case, a school laboratory only needs to have one BDM or use one Dragon12 board as a BDM POD, to program all students' boards with serial monitor.

To replace bootloader and D-Bug12 monitor with serial monitor, you need a BDM or a BDM POD to perform the task. The instructions to program the on-chip flash memory is shown on page 17. The latest D-Bug12 monitor and serial monitor can be downloaded from:

www.EVBplus.com\download_hcs12\download_hcs12.html

Some universities use Code Warrior IDE only. In this case, we pre-load the on-chip flash memory with serial monitor.

If your board is pre-installed with D-Bug12 monitor, it will display the diagnostic code "E-4-3" on the 7-segment LED and the speaker will chirp once when the board is turned on. The code character "E" stands for EVB mode.

If you ordered the board with serial monitor for Code Warrior, it would be pre-installed with serial monitor and a factory test program. The state of the left switch of the 2-position DIP switch (S7) is tested by the serial monitor for selecting RUN or LOAD mode during power up or reset.

If the left switch is placed in "LOAD" mode (in the "low" position) the monitor will display the diagnostic code "S-4-3" on the 7-segment LED and the speaker will chirp once and then wait for a command from PC.

If the left switch is placed in "RUN" mode (in the "up" position) the monitor will display the diagnostic code "U-4-3" on the 7-segment LED and the speaker will chirp once to indicate that the program execution is diverted to the user code.

The code character "**U**" stands for running a **U**ser program and the code character "**S**" is displayed as the number "**5**" and stands for **S**erial monitor.

Code Warrior communicates with serial monitor only in LOAD mode and so in order to interface with Code Warrior you have to reset the left switch in the “low “position. It will display the diagnostic code “**5**-4-3” on the 7-segment LED and the speaker will chirp once when the board is turned on

If the serial monitor is not installed or erased by a BDM, the LED indicators will not light up one at a time during power up or reset.

We will add setup procedures for Code Warrior in the future. For the time being you can visit some university web sites for more information.

Instructions of how to download Code Warrior from Freescale’s web site:
http://web.njit.edu/~paterno/ECET310/CodeWarrior.pdf

http://web.njit.edu/~paterno/ECET310/T3-CodeWarrior%20Simulator.pdf

CodeWarrior Familiarization & Project Setup:
A Guide to Beginning Embedded Assembly and C Programming on the S12
http://www.aet.cup.edu/~jsumey/cet360/cwintro/cwintro.html

The most resourceful “Code Warrior for the Dragon12 trainer” web site:
http://www.mecheng.adelaide.edu.au/robotics/wpage.php?wpage_id=56

Following is the web site for downloading the free Code Warrior special edition:
http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=01272600610BF1

Following is the web site for downloading the Code Warrior full edition for a 30-day free evaluation:
http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=01272600612247

Chapter 7: PLL code

```
;     The crystal frequency on the MiniDragon+ board is 16 MHz so the default bus speed is
;     8 MHz.  In order to set the bus speed high than 8 MHz the PLL must be initialized.
;
;     You can cut and paste the following code to the beginning of your program.
;
;           The math used to set the PLL frequency is:
;
;           PLLCLK = CrystalFreq * 2 * (initSYNR+1) / (initREFDV+1)
;
;           CrystalFreq =     16 MHz on MiniDragon+ board
;           initSYNR    =     5,  PLL multiplier will be 6
;           initREFDV   =     3,  PLL divisor will be 4
;           PLLCLK = 16*2*6/4 = 48MHz
;           The bus speed = PLLCLK / 2 = 24 MHz
;
;
start:

; PLL code for 24MHz bus speed from a 4/8/16 crystal
            sei
            ldx     #0
            bclr    clksel,x,%10000000      ; clear bit 7, clock derived from oscclk
            bset    pllctl,x,  %01000000    ; Turn PLL on, bit 6 =1 PLL on, bit 6=0 PLL off
            ldaa    #$05                    ; 5+1=6 multiplier
            staa    synr,x
            ldaa    #$03    ; divisor=3+1=4, 16*2*6 /4 =  48MHz PLL freq, for 16 MHz crystal
 ;          ldaa    #$01    ; divisor=1+1=2,   8*2*6 /2 =  48MHz PLL freq, for 8 MHz crystal
 ;          ldaa    #$00    ; divisor=0+1=1,   4*2*6 /1 =  48MHz PLL freq, for 4 MHz crystal

            staa    refdv,x
wait_b3:    brclr   crgflg,x, %00001000 wait_b3     ; Wait until bit 3 = 1
            bset    clksel,x, %10000000
```

8.1 D-Bug12 utility routines

The AN1280 was written for OLD 68HC12 family. If you happen to use printf routine with your old 68HC12 board you should be aware that I/O utility routines are moved to different addresses in D-Bug12 V4.x.

The address for the printf is $EE88 and addresses of other I/O routines are listed below:

Function	Description	Pointer Address
`far main()`	Start of D-Bug12	$EE80
`getchar()`	Get a character from SCI0 or SCI1	$EE84
`putchar()`	Send a character out SCI0 or SCI1	$EE86
`printf()`	Formatted Output - Translates binary values to characters	$EE88
`far GetCmdLine()`	Obtain a line of input from the user	$EE8A
`far sscanhex()`	Convert an ASCII hexadecimal string to a binary integer	$EE8E
`isxdigit()`	Checks for membership in the set [0..9, a..f, A..F]	$EE92
`toupper()`	Converts lower case characters to upper case	$EE94
`isalpha()`	Checks for membership in the set [a..z, A..Z]	$EE96
`strlen()`	Returns the length of a null terminated string	$EE98
`strcpy()`	Copies a null terminated string	$EE9A
`far out2hex()`	Displays 8-bit number as 2 ASCII hex characters	$EE9C
`far out4hex()`	Displays 16-bit number as 4 ASCII hex characters	$EEA0
`SetUserVector()`	Setup user interrupt service routine	$EEA4
`far WriteEEByte()`	Write a data byte to on-chip EEPROM	$EEA6
`far EraseEE()`	Bulk erase on-chip EEPROM	$EEAA
`far ReadMem()`	Read data from the M68HC12 memory map	$EEAE
`far WriteMem()`	Write data to the M68HC12 memory map	$EEB2

Fig 8-1: D-Bug12 utility routines

8.2 Interrupt vector table

Table 5-1 Interrupt Vector Locations

Vector Address	Interrupt Source	CCR Mask	Local Enable	HPRIO Value to Elevate
$FFFE, $FFFF	Reset	None	None	–
$FFFC, $FFFD	Clock Monitor fail reset	None	PLLCTL (CME, SCME)	–
$FFFA, $FFFB	COP failure reset	None	COP rate select	–
$FFF8, $FFF9	Unimplemented instruction trap	None	None	–
$FFF6, $FFF7	SWI	None	None	–
$FFF4, $FFF5	XIRQ	X-Bit	None	–
$FFF2, $FFF3	IRQ	I-Bit	IRQCR (IRQEN)	$F2
$FFF0, $FFF1	Real Time Interrupt	I-Bit	CRGINT (RTIE)	$F0
$FFEE, $FFEF	Enhanced Capture Timer channel 0	I-Bit	TIE (C0I)	$EE
$FFEC, $FFED	Enhanced Capture Timer channel 1	I-Bit	TIE (C1I)	$EC
$FFEA, $FFEB	Enhanced Capture Timer channel 2	I-Bit	TIE (C2I)	$EA
$FFE8, $FFE9	Enhanced Capture Timer channel 3	I-Bit	TIE (C3I)	$E8
$FFE6, $FFE7	Enhanced Capture Timer channel 4	I-Bit	TIE (C4I)	$E6
$FFE4, $FFE5	Enhanced Capture Timer channel 5	I-Bit	TIE (C5I)	$E4
$FFE2, $FFE3	Enhanced Capture Timer channel 6	I-Bit	TIE (C6I)	$E2
$FFE0, $FFE1	Enhanced Capture Timer channel 7	I-Bit	TIE (C7I)	$E0
$FFDE, $FFDF	Enhanced Capture Timer overflow	I-Bit	TSRC2 (TOF)	$DE
$FFDC, $FFDD	Pulse accumulator A overflow	I-Bit	PACTL (PAOVI)	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I-Bit	PACTL (PAI)	$DA
$FFD8, $FFD9	SPI0	I-Bit	SP0CR1 (SPIE, SPTIE)	$D8
$FFD6, $FFD7	SCI0	I-Bit	SC0CR2 (TIE, TCIE, RIE, ILIE)	$D6
$FFD4, $FFD5	SCI1	I-Bit	SC1CR2 (TIE, TCIE, RIE, ILIE)	$D4
$FFD2, $FFD3	ATD0	I-Bit	ATD0CTL2 (ASCIE)	$D2
$FFD0, $FFD1	ATD1	I-Bit	ATD1CTL2 (ASCIE)	$D0
$FFCE, $FFCF	Port J	I-Bit	PTJIF (PTJIE)	$CE
$FFCC, $FFCD	Port H	I-Bit	PTHIF(PTHIE)	$CC
$FFCA, $FFCB	Modulus Down Counter underflow	I-Bit	MCCTL(MCZI)	$CA

Fig 8-2: MC9S12DG256 Interrupt vector table 1

$FFC8, $FFC9	Pulse Accumulator B Overflow	I-Bit	PBCTL(PBOVI)	$C8
$FFC6, $FFC7	CRG PLL lock	I-Bit	CRGINT(LOCKIE)	$C6
$FFC4, $FFC5	CRG Self Clock Mode	I-Bit	CRGINT (SCMIE)	$C4
$FFC2, $FFC3	BDLC	I-Bit	DLCBCR1(IE)	$C2
$FFC0, $FFC1	IIC Bus	I-Bit	IBCR (IBIE)	$C0
$FFBE, $FFBF	SPI1	I-Bit	SP1CR1 (SPIE, SPTIE)	$BE
$FFBC, $FFBD	SPI2	I-Bit	SP2CR1 (SPIE, SPTIE)	$BC
$FFBA, $FFBB	EEPROM	I-Bit	EECTL(CCIE, CBEIE)	$BA
$FFB8, $FFB9	FLASH	I-Bit	FCTL(CCIE, CBEIE)	$B8
$FFB6, $FFB7	CAN0 wake-up	I-Bit	CAN0RIER (WUPIE)	$B6
$FFB4, $FFB5	CAN0 errors	I-Bit	CAN0RIER (CSCIE, OVRIE)	$B4
$FFB2, $FFB3	CAN0 receive	I-Bit	CAN0RIER (RXFIE)	$B2
$FFB0, $FFB1	CAN0 transmit	I-Bit	CAN0TIER (TXEIE2-TXEIE0)	$B0
$FFAE, $FFAF	CAN1 wake-up	I-Bit	CAN1RIER (WUPIE)	$AE
$FFAC, $FFAD	CAN1 errors	I-Bit	CAN1RIER (CSCIE, OVRIE)	$AC
$FFAA, $FFAB	CAN1 receive	I-Bit	CAN1RIER (RXFIE)	$AA
$FFA8, $FFA9	CAN1 transmit	I-Bit	CAN1TIER (TXEIE2-TXEIE0)	$A8
$FFA6, $FFA7	CAN2 wake-up	I-Bit	CAN2RIER (WUPIE)	$A6
$FFA4, $FFA5	CAN2 errors	I-Bit	CAN2RIER (CSCIE, OVRIE)	$A4
$FFA2, $FFA3	CAN2 receive	I-Bit	CAN2RIER (RXFIE)	$A2
$FFA0, $FFA1	CAN2 transmit	I-Bit	CAN2TIER (TXEIE2-TXEIE0)	$A0
$FF9E, $FF9F	CAN3 wake-up	I-Bit	CAN3RIER (WUPIE)	$9E
$FF9C, $FF9D	CAN3 errors	I-Bit	CAN3RIER (TXEIE2-TXEIE0)	$9C
$FF9A, $FF9B	CAN3 receive	I-Bit	CAN3RIER (RXFIE)	$9A
$FF98, $FF99	CAN3 transmit	I-Bit	CAN3TIER (TXEIE2-TXEIE0)	$98
$FF96, $FF97	CAN4 wake-up	I-Bit	CAN4RIER (WUPIE)	$96
$FF94, $FF95	CAN4 errors	I-Bit	CAN4RIER (CSCIE, OVRIE)	$94
$FF92, $FF93	CAN4 receive	I-Bit	CAN4RIER (RXFIE)	$92
$FF90, $FF91	CAN4 transmit	I-Bit	CAN4TIER (TXEIE2-TXEIE0)	$90
$FF8E, $FF8F	Port P Interrupt	I-Bit	PTPIF (PTPIE)	$8E
$FF8C, $FF8D	PWM Emergency Shutdown	I-Bit	PWMSDN (PWMIE)	$8C
$FF80 to $FF8B	Reserved			

Fig 8-3: MC9S12DG256 Interrupt vector table 2

Interrupt Source	Secondary Vector Address	Interrupt Source	Secondary Vector Address
Reserved $FF80	$EF80	I^2C bus	$EFC0
Reserved $FF82	$EF82	DLC	$EFC2
Reserved $FF84	$EF84	SCME	$EFC4
Reserved $FF86	$EF86	CRG lock	$EFC6
Reserved $FF88	$EF88	Pulse accumulator B overflow	$EFC8
Reserved $FF8A	$EF8A	Modulus down counter underflow	$EFCA
PWM emergency shutdown	$EF8C	Port H interrupt	$EFCC
Port P interrupt	$EF8E	Port J interrupt	$EFCE
MSCAN 4 transmit	$EF90	ATD1	$EFD0
MSCAN 4 receive	$EF92	ATD0	$EFD2
MSCAN 4 errors	$EF94	SCI1	$EFD4
MSCAN 4 wakeup	$EF96	SCI0	$EFD6
MSCAN 3 transmit	$EF98	SPI0	$EFD8
MSCAN 3 receive	$EF9A	Pulse accumulator A input edge	$EFDA
MSCAN 3 errors	$EF9C	Pulse accumulator A overflow	$EFDC
MSCAN 3 wakeup	$EF9E	Timer overflow	$EFDE
MSCAN 2 transmit	$EFA0	Timer channel 7	$EFE0
MSCAN 2 receive	$EFA2	Timer channel 6	$EFE2
MSCAN 2 errors	$EFA4	Timer channel 5	$EFE4
MSCAN 2 wakeup	$EFA6	Timer channel 4	$EFE6
MSCAN 1 transmit	$EFA8	Timer channel 3	$EFE8
MSCAN 1 receive	$EFAA	Timer channel 2	$EFEA
MSCAN 1 errors	$EFAC	Timer channel 1	$EFEC
MSCAN 1 wakeup	$EFAE	Timer channel 0	$EFEE
MSCAN 0 transmit	$EFB0	Real-time interrupt	$EFF0
MSCAN 0 receive	$EFB2	IRQ	$EFF2
MSCAN 0 errors	$EFB4	XIRQ	$EFF4
MSCAN 0 wakeup	$EFB6	SWI	$EFF6
FLASH	$EFB8	Unimplemented instruction trap	$EFF8
EEPROM	$EFBA	COP failure reset	$EFFA
SPI2	$EFBC	Clock monitor fail reset	$EFFC
SPI1	$EFBE	Reset	$EFFE

Fig 8-4: MC9S12DG256 secondary interrupt vector table

8.3 Useful web links

The web is the best source for getting more information about the HCS12. The Freescale web site has all documents and application notes that you need. The HC12 user group http://groups.yahoo.com/group/68HC12/ is a good place to ask a question and get a prompt answer from many other HC12 users.

You also can visit our web site at:
http://www.evbplus.com/hc11_68hc11_hc12_68hc12_9s12_hcs12_sites.html

to get links to many university web sites that offer course materials and lab assignments for the Dragon12 and MiniDragon+ boards.

All HCS12 boards that are pre-loaded with Freescale serial monitor, bootloader and D-Bug12 monitor on the market today are basically the same products as far as software development is concerned. If you are going to use a BDM to debug a HCS12 board, all HCS12 boards will respond to all BDM commands in the same manner because the BDM directly communicates with the MC9S12DG256 MCU. The information on our manual can apply to the boards from other manufacturers, and vice versa.

8.4 Troubleshooting notes

The following are some important notes that you should know and they may save you time:

1. Things to do if the board does not work.

Many little mistakes can cause a big problem, especially for beginners. Before troubleshooting the board, you must apply power to the board. When the board is powered, the decimal point of the 7-segment LED must be on. If it's off, the board does not have 5V DC. Sometimes it may be caused by a bad AC adapter or the AC adapter may not even be plugged in.

To determine if the board malfunctions, you can restore the following jumper settings to the original default settings when you receive the board. The default settings are as follows:

S3	AN5 input	No jumper
S4	AN6 input	No jumper

J15	Speaker driving source. Jumper is placed in the "top" position (driven by PT5)
S7	MODE select, both DIP switches of S7 are set in the "low" positions for EVB mode.

If all above settings are correct and when you press the reset button, the 7-segment LED should display the diagnostic code of E-4-3 momentarily. If this does not occur, the bootloader could have been erased by a BDM. You can use a BDM with instructions from the manufacturer or use another MiniDragon+ board as a BDM POD to re-program bootloader and D-Bug12 monitor into flash memory according to the instructions on page 17.

If the diagnostic code is displayed correctly, but the board does not communicate with the terminal, the EVB portion of flash memory could be erased or the com port number may not be set correctly by the AsmIDE.

You can re-program the D-Bug12 in bootloader mode according the instructions on page 20. The newest firmware can be downloaded at: www.evbplus.com/download_hcs12.html

If the screen displays some garbled characters, the baud rate may not be set correctly. The D-Bug12 resets the baud rate to 9600 at power up, if you changed the baud rate, you must change the AsmIDE's baud rate to the same value.

2. Always reset the board before downloading a new program.

If the previous application program that you ran was aborted, then you may need to reset the board before downloading a new application program. The reset action will disable the interrupt that was enabled by the previous application. If the interrupt was caused by a timer and is not disabled, the timer interrupt will continue even it's not called for in your new application program. The result will be unpredictable.

3. In EVB mode, reset clears your pseudo RAM interrupt vectors.

When you develop code with interrupts in RAM, you must initialize pseudo RAM interrupt vectors in the very beginning of your program, because if you press the reset button it will clear all pseudo RAM interrupt vectors. If you don't initialize pseudo RAM interrupt vectors in your program and your application program uses interrupts, your program may not run correctly since the interrupt vectors do not exist.

4. Operating mode changing is only effective after reset.

There are four operating modes that are selected by S7. The mode change won't be effective until you reset the board. So you must always press the reset button after a mode change.