ThunderBird12 Module

The Freescale MC9S12DG256 based microcontroller module

User's Manual

Revision: 1.00

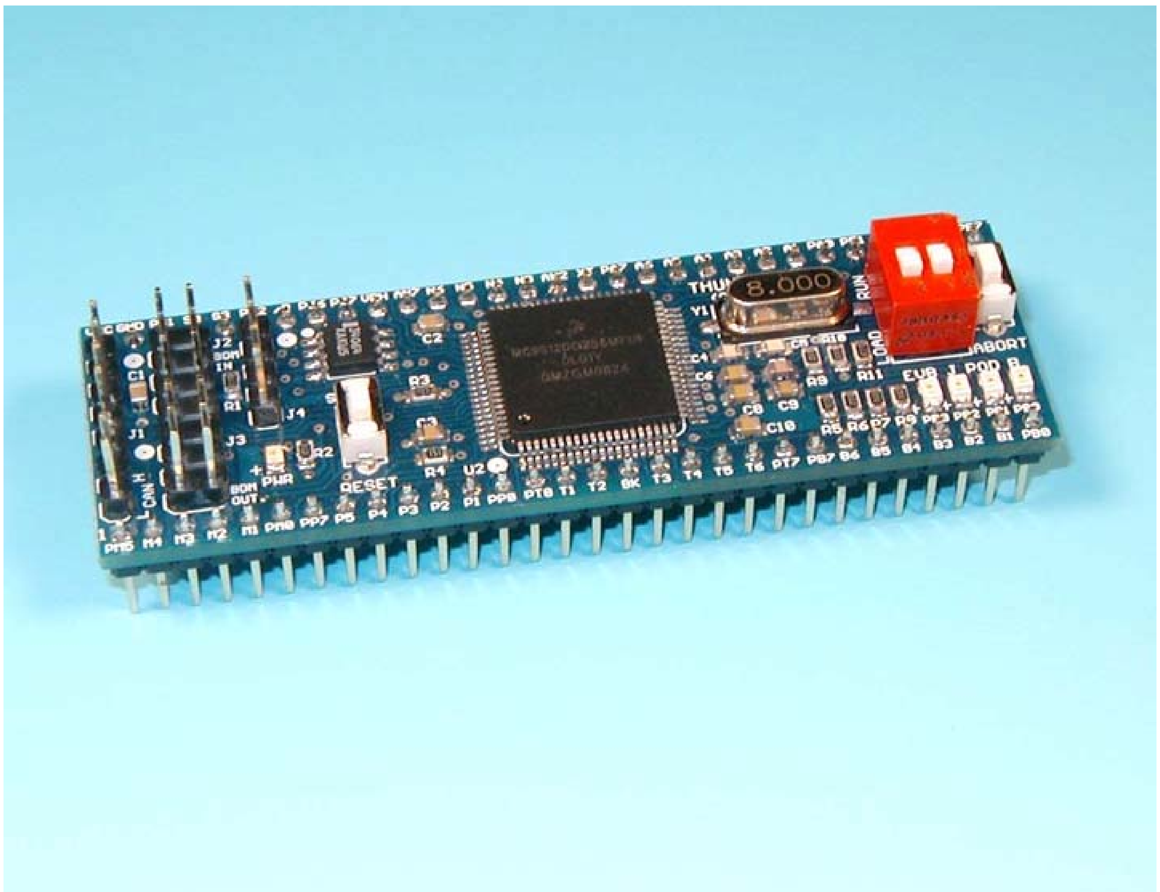

Table OF Contents

Chapter 1. Introduction

1.1 Welcome

Thank you very much for purchasing the ThunderBird12 module. The ThunderBird12 module is a low-cost, full-featured STAMP type microcontroller module based on the powerful 16-bit Freescale MC9S12DG256 microcontroller.

If you have any questions, please contact sales@EVBplus.com or call 630 894-1440 for help.

1.2 MC9S12DG256 features and memory map:

The ThunderBird12 board comes with the MC9S12DG256CFUE 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 DP256 has 5 CAN ports and one BDLC port but the DG256 has only 2 CAN ports and no BDLC port, otherwise these two microcontrollers have the same features. If you don't use more than 2 CAN ports and don't need the BDLC communication 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 MC9S12DG256CFUE 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
- 7-ch 8-bit or 3-ch 16 bit PWM
- 8-channel 10-bit A/D converter
- Fast 25 MHz bus speed via on-chip Phase Lock Loop
- BDM for in-circuit programming and debugging
- 80-pin LQFP package

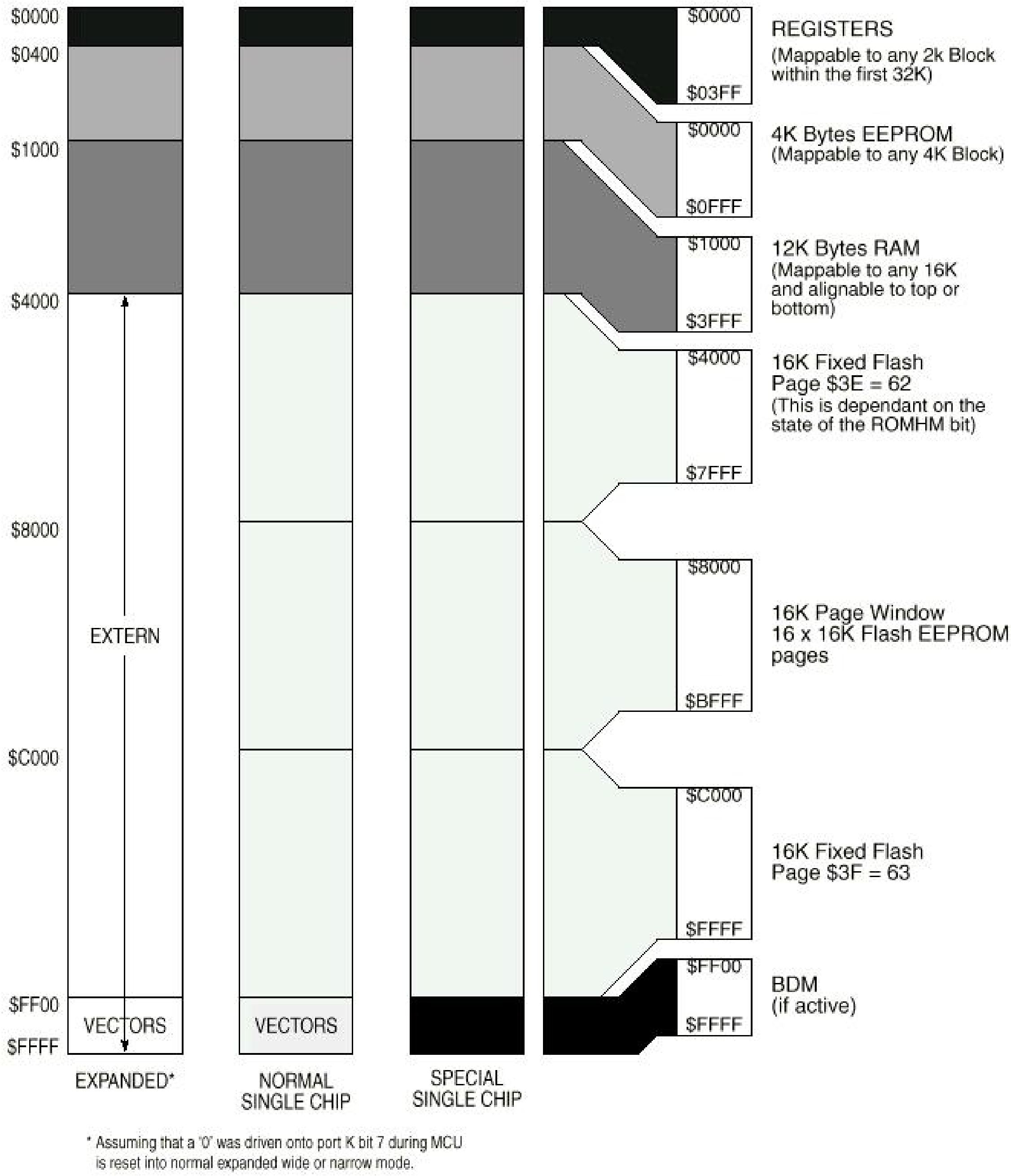

Fig 1-1: MC9S12DG256 Memory map

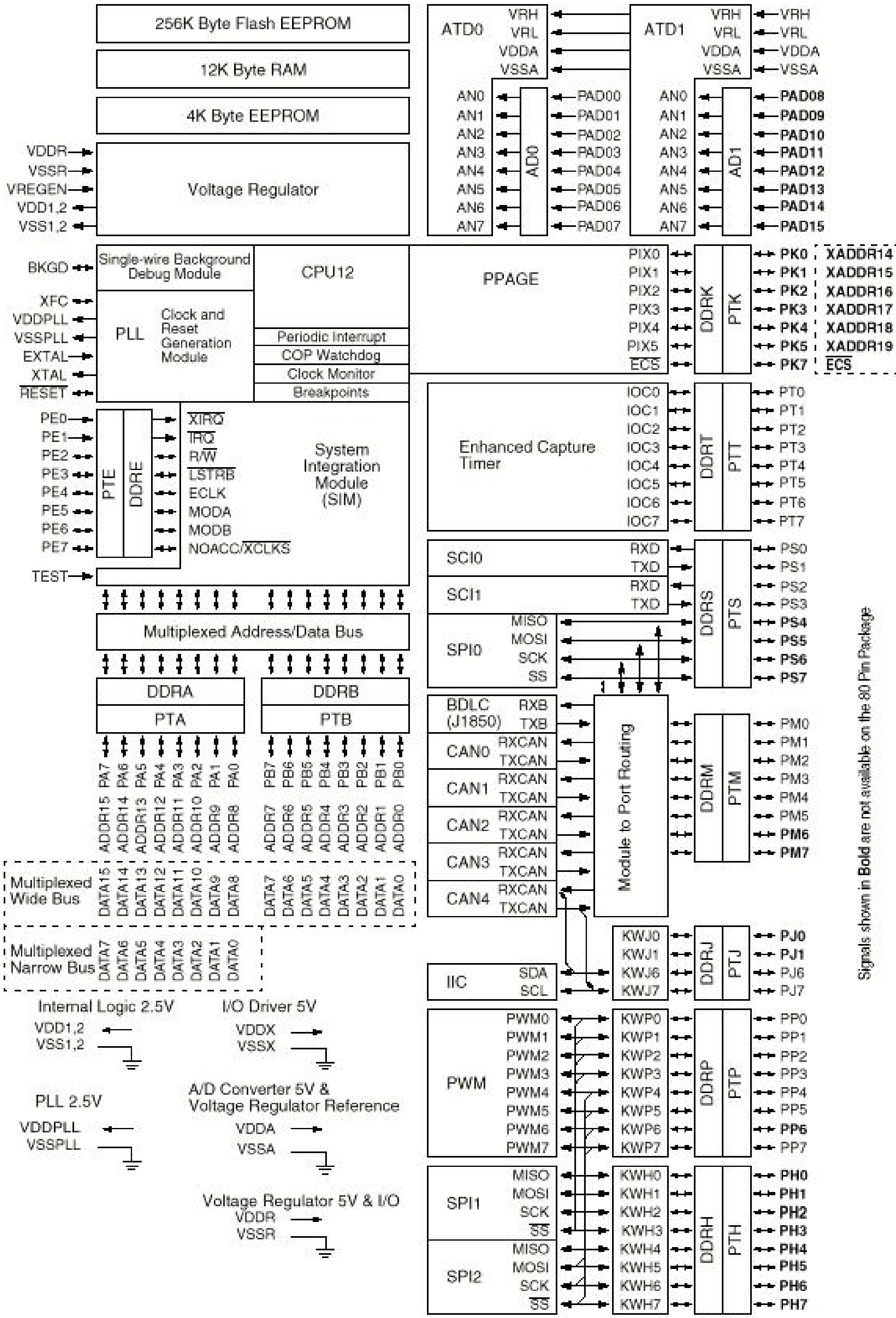

Fig 1-2: MC9S12DG256 MCU block diagram

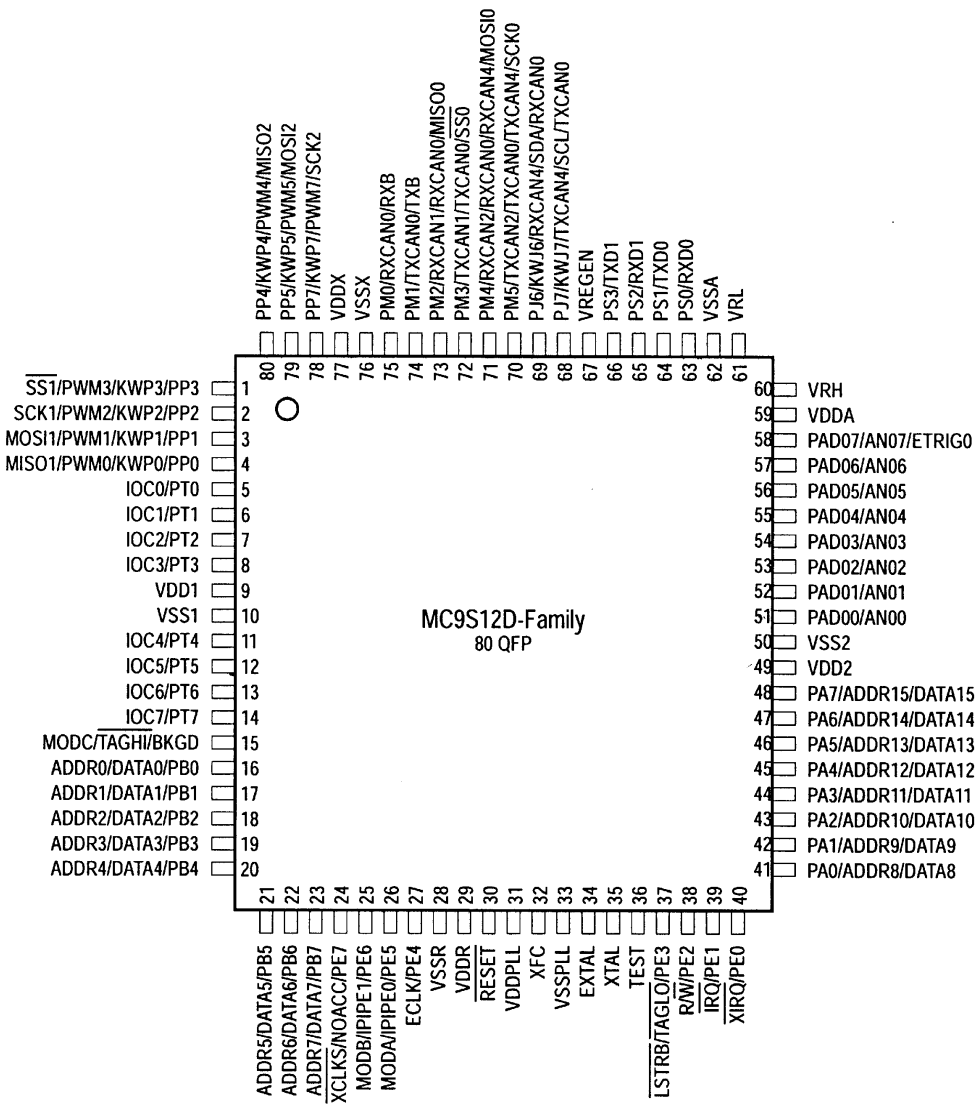

Fig 1-3: MC9S12DG256CFUE MCU pin assignments

1.3 On-board hardware features:

- Four LED indicators connected to PB0-PB3
- Dual SCI headers
- Power-On LED indicator
- Reset switch
- Abort switch for stopping program when program is hung in a dead loop
- Besides the on-chip 256K flash, 12K RAM and 4K EEPROM, the 9S12 MCU, MC9S12DG256CFUE, also includes the following on-chip peripherals:
    - 3 SPIs
    - 2 SCIs
    - 2 CANs
    - I^2C interface
    - 8 16-bit timers
    - 7 PWMs
    - 6-channel 10-bit A/D converter
- Super fast bus speed up to 25 MHz
- The 80-Pins 9S12 MCU (MC9S12DG256CFUE) is included
- 55 I/O pins are available to user applications
- BDM-in connector to be connected with a BDM from multiple vendors for debugging.
- BDM-out connector for making this board as a HCS12 / 9S12 BDM or programmer. No extra hardware needed.
- Comes with AsmIDE under GPL (general public license)
- Pre-loaded with D-Bug12 monitor for working with AsmIDE and EmbeddedGNU
- Or pre-loaded with serial monitor for working with Code Warrior
- Supports source level debugging in C and Assembler without a BDM
- Mode switch for selecting 4 operating modes with LED indicators in D-Bug12 monitor: EVB, Jump-to-EEPROM, BDM POD and Bootloader
- Auto start user programs with visual verification when the board is turned on
- Combined with a RoadRunner stick for fast prototyping on a user-provided solderless breadboard
- Hardware is compatible to the Dragon12 Plus board.
- Provided with a sample program including assembly source code
- PC board size 3.05" X 0.88"

1.4 I/O Pin Usage

Pinouts of the 60-pin DP module:

1	PM5/TXCAN0/TXCAN4/SCK0	60	VCC
2	PM4/RXCAN0/RTXCAN4/MOSI0	59	GND
3	PM3/TXCAN1/TXCAN0/SS0	58	PS1/TXD0
4	PM2/RXCAN1/RXCAN0/MISO0	57	PS0/RXD0
5	PM1/TXCAN0/TXB	56	PS3/TXD1
6	PM0/RXCAN0/RXB	55	PS2/RXD1
7	PP7/KWP7/PWM7/SCK2	54	/RESET
8	PP5/KWP5/PWM5/MOSI2	53	PJ6/KWJ6/RXCAN4/SDA
9	PP4/KWP4/PWM4/MISO2	52	PJ7/KWJ7/TXCAN4/SCL
10	PP3/KWP3/PWM3/SS1	51	VRH
11	PP2/KWP2/PWM2/SCK1	50	PAD07/AN7
12	PP1/KWP1/PWM1/MOSI1	49	PAD06/AN6
13	PP0/KWP0/PWM0/MISO1	48	PAD05/AN5
14	PT0/IOC0	47	PAD04/AN4
15	PT1/IOC1	46	PAD03/AN3
16	PT2/IOC2	45	PAD02/AN2
17	BKGD	44	XTAL or PE0/XIRQ
18	PT3/IOC3	43	PA7
19	PT4/IOC4	42	PA6
20	PT5/IOC5	41	PA5
21	PT6/IOC6	40	PA4
22	PT7/IOC7	39	PA3
23	PB7	38	PA2
24	PB6	37	PA1
25	PB5	36	PA0
26	PB4	35	PE1/IRQ
27	PB3	34	PE2
28	PB2	33	PE3
29	PB1	32	PE4
30	PB0	31	PE7

Pin Name	MCU Pin #	I/O Usage
PA0	Pin 41	not used
PA1	Pin 42	not used
PA2	Pin 43	not used
PA3	Pin 44	not used
PA4	Pin 45	not used
PA5	Pin 46	not used
PA6	Pin 47	not used
PA7	Pin 48	not used
PB0	Pin 16	LED indicator
PB1	Pin 17	LED indicator
PB2	Pin 18	LED indicator
PB3	Pin 19	LED indicator
PB4	Pin 20	not used
PB5	Pin 21	not used
PB6	Pin 22	not used
PB7	Pin 23	not used
PE0	Pin 40	Abort switch S3 (input)
PE1	Pin 39	not used
PE2	Pin 38	not used
PE3	Pin 37	not used
PE4	Pin 27	not used
PE5	Pin 26	not used
PE6	Pin 25	not used
PE7	Pin 24	not used
PJ6	Pin 69	not used
PJ7	Pin 68	not used

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

Pin Name	MCU Pin #	I/O Usage
PM0	Pin 75	not used
PM1	Pin 74	not used
PM2	Pin 73	not used
PM3	Pin 72	not used
PM4	Pin 71	not used
PM5	Pin 70	not used
PP0	Pin 4	not used
PP1	Pin 3	not used
PP2	Pin 2	not used
PP3	Pin 1	not used
PP4	Pin 80	not used
PP5	Pin 79	not used
PP7	Pin 78	not used
PS0	Pin 63	SCI0 for PC communication, RECV
PS1	Pin 64	SCI0 for PC communication, XMIT
PS2	Pin 65	SCI1 for user applications, RECV, not used
PS3	Pin 66	SCI1 for user applications, XMIT, not used
PT0	Pin 5	not used
PT1	Pin 6	not used
PT2	Pin 7	not used
PT3	Pin 8	not used
PT4	Pin 11	not used
PT5	Pin 12	not used
PT6	Pin 13	BDMout reset (output, used in POD mode only)
PT7	Pin 14	BDMout data line (bi-directional, used in POD mode only)
PAD0	Pin 51	D-bug12 mode select, DIP switch S2
PAD1	Pin 52	D-bug12 or Serial Monitor mode select, DIP switch S2
PAD2	Pin 53	not used
PAD3	Pin 54	not used
PAD4	Pin 55	not used
PAD5	Pin 56	not used
PAD6	Pin 57	not used
PAD7	Pin 58	not used

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

Chapter 2. Quick Start

By default the ThunderBird12 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 from CD.**

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 from CD:

The installation is automated by double clicking on the **SETUP.BAT** in the CD. It will create a folder c:\Wytec\ThunderBird12\examples and copy all example program files from the CD to c:\Wytec\ThunderBird12\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:\Wytec\ThunderBird12, not c:\Windows\desktop or other locations. First, right click the Start button, then left click “Explorer”, left click on C:\Wytec\ThunderBird12, 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 ThunderBird12. 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:\Wytec\ThunderBird12. 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 ThunderBird12 board is created in a text file named c:\Wytec\ThunderBird12\AsmIDE.ini. In the future if you get lost with all the changes, you always can copy this file into the folder c:\Wytec\ThunderBird12.

2.2 Getting Started

To operate the ThunderBird12 module, follow steps1 through 5 below:

1. The module does not have a built-in 5V regulator, so it must be supplied with a regulated 5V voltage source. If you bought it with a RoadRunner module or a USB to TTL adapter (UTA) the ThunderBird12 module will be supplied with 5V from a USB port in your PC through these two devices. If you supply your own 5V power supply for the module make sure that the Pin 60 is the VCC (5V) and the pin 59 is the ground.

2. After power up, the states of the 2-position DIP switch (S2) are tested by the bootloader for selecting one of 4 operating modes during power up or reset, and the four LED indicators will blink one at a time from **left to right** to indicate that the bootloader is functioning. Then one of them will be lit to indicate the selected operating mode and other three will be turned off.

   If you have a small speaker you can connect the positive terminal of the speaker to PT5 via a 2.2uf-10uf capacitor and the negative terminal to ground. The speaker would chirp once during power up.

   The 4 operating modes tested by the bootloader are EVB, Jump to EE, BDM POD and Bootloader.

   If it does not occur check the Power-On LED indicator. It is lit when VCC (5V) is present. If the POWER LED is off check VCC and Ground connection and USB cable connection.

3. To invoke the AsmIDE, you can right click the Start button, then left click “Explorer”, left click on C:\Wytec\ThunderBird12 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.

4. 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 ThunderBird12 board.

5. 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). See details at the following link: http://www.evbplus.com/usb232.html

6. 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.

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

8. 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.

9. 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.

10. 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 ThunderBird12 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 - 2007 Freescale Semiconductor
For Commands type "Help"
>
```

2.3 Test Hardware:

To help users get up and running, the ThunderBird12 board comes with a simple, ready-to-run test program including source code.

The test program must be run from RAM in EVB mode. In order to run the test program in EVB mode, both the DIP switches of S2 must be set in “low” positions.

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

1. Click the File button to open the Binary_counter.asm from c:\Wytec\ThunderBird12\examples.

2. After the Binary_counter.asm is loaded into the top window click the Build button to assemble code and generate the Binary_count.s19 file. This is how you normally generate an s19 file. You can omit this step, because the Binary_counter.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 ‘Binary_counter.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.

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.

The test program is 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.

Note: The starting address happens to be assigned at $2000 in the test program. It can be at any location in RAM.

Chapter 3. Software descriptions

3.1 Bootloader and D-Bug12 Monitor

The MC9S12DG256 on the ThunderBird12 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, S2. After power up or reset, the MC9S12DG256 will test the states of 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 in the folder named C:\Wytec\ThunderBird12\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 the original interrupt vector addresses are taken by bootloader, the bootloader and D-Bug12 monitor will map all 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 - 2007 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, the EVB mode LED will go off after power up.

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 from EEPROM, 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 host to access all target MCU resources on the target board (another HCS12 / 9S12) 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 HCS12 / 9S12 board).

To use the host board as a programmer, you need a 6-pin ribbon cable to connect from the BDM OUT of the host 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 host 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 will see the following sign-on message:

```
D-Bug12 v4.0.0b32
Copyright 1996 - 2007 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 host board and target board, and 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 ThunderBird12 bootloader is modified from the Freescale’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_ThunderBird.s29 Boot_ThunderBird.s19

Now we type “FLOAD” <Enter> at the prompt. Click the Build button, select the Download option, and select the file named **Boot_ThunderBird.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_DR12P.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>

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 ThunderBird12 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_ThunderBird .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_ThunderBird_8MHz.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 make port B as a binary counter 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    ddrb,x          ; make port B an output port
            clr     portb,x         ; turn off all LEDs

back:       inc     portb,x         ; turn on LEDs
            jsr     d250ms          ; delay 250ms
            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 “Binary_counter.asm”, not just “Binary_counter”.

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 - 2007 Freescale Semiconductor
For Commands type "Help"
>
```

6. Type “LOAD” <Enter>

7. Click Build button, select Download option and locate the file named ‘Binary_counter.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, Binary_counter.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 the original interrupt vector addresses are taken by the bootloader. The bootloader and D-Bug12 monitor will map 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 in the 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 S2 for EVB mode, and 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 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.

The crystal frequency is 8 MHz and usually it will result in a 4MHz 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 LED indicators:

The power-on indicator LED will always be lit when the 5V is present to the module.

4 other LED indicators are connected to PB0-PB3.

4.2 Dual SCI communication ports

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

The 4-pin male header J1 is connected to the SCI0 of the DG256 while the J4 is connected to the SC1 of the DG256. The D-Bug12 monitor or serial monitor works with SCI0, so the UTA board should connect the SCI0 to a PC's USB port during debugging sessions. The SCI1 can be used by user's application programs.

4.3 All jumpers

All on-board jumpers:

J1 TTL port for SCI0, may be connected to a Wytec's UTA board
J2 BDM in
J3 BDM output in POD mode
J4 TTL port for SCI1, may be connected to a Wytec's UTA board

J5 Selects Crystal Osc. frequency output or PE0 (/XIRQ) for pin 44, located at the solder side,
J6 Selects 5V or 2.5V for VRH (pin 51), defaults to 5V, located at the solder side
J7 RS of CAN0 (U3), is connected to VSS, located at the solder side
J8 Connects PM0 to RXD of CAN0, located at the solder side
J9 Connection of the terminating resistor for CAN0. Place a jumper on this header on the last note in a network only if you use CAN0. It will save power consumption of the module without the jumper if CAN0 is not used. Located at the solder side,CAN

Chapter 5: EmbeddedGNU

See installation instructions at:

http://hcs12text.com/links.html

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 Robot trainer board, like the ThunderBird12 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 ThunderBird12 MODULE 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 module is pre-loaded with SM (Serial Monitor) When the module is installed with serial monitor, the state of the left switch of the 2-position DIP switch (S2) is tested by the serial monitor for selecting RUN or LOAD mode during power up or reset, and the four LED indicators will blink one at a time from **right to left** to indicate that the serial monitor is functioning. If the left switch is placed in "LOAD" mode the monitor will wait for a command from PC. If the left switch is placed in "RUN" mode the

LEDs will sweep back from **left to right** to indicate that the program execution is diverted to the user code.

If the PT5 is connected to a speaker the speaker will chirp once when the board is turned on.

The setup procedures for Code Warrior are available from some university web sites and their links are provided at web site: http://www.evbplus.com/Code_Warrior_hcs12.html

Chapter 7: PLL code

```
;     The crystal frequency on the ThunderBird12 board is 8 MHz so the default bus speed is
;     4 MHz.  In order to set the bus speed high than 4 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 =    8 MHz on ThunderBird12 board
;          initSYNR    =    5,  PLL multiplier will be 6
;          initREFDV   =    1,  PLL divisor will be 2
;          PLLCLK      =    8*2*6/2 = 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	SCII	$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 Freescale forum http://forums.freescale.com/freescale/ and the HC12 user group http://groups.yahoo.com/group/68HC12/ are good places 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 ThunderBird12 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.