DRAGON12

MC9S12DP256 Development Board

Getting Started Manual

Version 1.31 For REV. D board

Table of Contents

INTRODUCTION

The DRAGON12 is a low-cost, feature-packed development board for the new Motorola MC9S12 microcontroller family. It is compatible with the Motorola 9S12DP256EVB board and other similar development boards on the market today, but it also incorporates many on-board peripherals that make this board a better value for users.

For engineers, it is a convenient prototype platform suitable for designers who want to rapidly develop and prototype new MC9S12 applications. For students, it is an advanced microcontroller trainer. Use it to build a solid foundation of microcontroller expertise and to create real world applications for a senior project.

To help users get up and running, the DRAGON12 board comes with many fully debugged, ready-to-run sample programs. The programs are ported from our EVBplus2 68HC11 development board and because the 68HC12 instructions are upward code compatible with the 68HC11 instructions, migrating from your 68HC11 projects into the new HCS12 world could not be any easier.

The state of the art MC9S12DP256 controller is the most powerful chip in the family and it is loaded with hardware features such as:

- 25 MHz bus speed
- 256K flash memory
- 12K SRAM
- 4K EEPROM

On-chip peripherals include:

- dual SCIs
- triple SPIs
- I2C
- five CAN modules
- two 10-bit 8 channel ADCs
- eight PWMs
- eight 16-bit timers

It can also be used in the evaluation and development of all other family members, such as the A and D series.

The on-board hardware includes:

- BDM in and out ports
- solderless breadboard
- logic probe
- four robot servo outputs
- a 16X2 LCD display module
- a 4x4 keypad connector
- fast SPI expansion port
- speaker
- 4-digit 7-segment LED display
- 8 data-LED indicators
- 4 mode-LED indicators
- 8-postion DIP switch
- 4 pushbuttons
- potentiometer
- IR transceiver
- RF transceiver (not installed)
- RS485 interface
- 3 CAN ports (one CAN installed)
- dual SCI ports

All I/O lines in the 112 pin male header and female receptacle provide an easy access for your experiments on the breadboard,

The package also includes a 7.5V 300mA wall plug-in power supply and a 6-foot DB9 cable.

The specification of the AC adapter is:

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

The AC adapter is only available to the countries that use 110V.

WARNING: If more power is needed in a robot or other applications, the user should upgrade the AC adapter. Otherwise, the board could keep resetting itself when the VCC drops below 4.6V.
If your board sometimes resets by itself you need to upgrade your AC adapter to 9V 500mA or 9V 1A. Do not apply a DC voltage higher than 9V to this board.

People often use different terminology. In our product menus, "Download" means to transfer a file from the PC to the development board, while "Upload" means to transfer a file from the development board to the 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.

Install DRAGON12 software from CD:

The installation is automated by running "**SETUP.BAT**" on the CD. It will create a folder c:\DRAGON12\examples and copy all example program files from the CD to c:\DRAGON12\examples

The AsmIDE is freeware, but if you would like to use it in the future, we encourage you donate $5.00 to Eric Engler at: http://www.geocities.com/englere_geo/ .

After the software is successfully installed, you can make a shortcut to AsmIDE.exe.

It's very important to make a shortcut so that its target location is C:\DRAGON12, not c:\Windows\desktop or other locations. First, right click the Start button. Then, left click "Explorer". Left click on C:\DRAGON12. 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. 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:\DRAGON12. If you want to make a shortcut for AsmIDE on the Desktop, this is the correct way to do so. If you don't follow this method, your program may have a problem to run. Never drag the AsmIDE.exe to the desktop folder.

The default setting of AsmIDE for the DRAGON12 board is created in a text file named c:\DRAGON12\AsmIDE.ini. In the future if you get lost with all the changes, you always can copy this file into the folder c:\DRAGON12.

GETTING STARTED

To operate the DRAGON12 board, follow steps1 through 5 below:

Step 1.

Plug the AC adapter into a wall outlet and plug the DC female plug at the other end into the DC jack on the lower left side of the DRAGON12 board. During power up, the PB7-PB0 LEDs should chase from left to right. If it does not occur, check the jumper setting on J30 (MODE B), J31 (MODE A) and J32 (MODE C). They must be set for single chip mode (0-0-1). The two DIP switches on the SW7 must be set at the low positions, so the **EVB** mode LED is lit.

Step 2.

Plug the DB9 male end of the cable into the DB9 connector P1 on the **upper** left side of the DRAGON12 board and plug the DB9 female end of the cable into the COM1 or COM2 port on your PC. The DB9 connector P2 on the lower left side of the board is the MC9S12DP256 SCI1 port that can be used by a user's program.

Step 3.

Press the reset button on the DRAGON12 board, and the PB7-PB0 LEDs should chase from left to right.

Step 4.

To invoke the AsmIDE, right click the Start button. Then, left click "Explorer". Left click on C:\DRAGON12 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.

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**.

Step 5:

The AsmIDE is simple and very easy to use. You only need to use three commands from the AsmIDE for your 68HC12 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 DRAGON12 board.

In the View->Option->Terminal Window Options menu, set the COM port as 1 or 2 to match your PC COM port. Also, set the COM port options at 9600, N,8,1, and check the "enable the terminal window" box.

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 new assembler.

If the assembler detects an error, it will show the error's line number in the file along with an error message.

If the terminal options are set correctly, you should see the following prompt every time the reset button on the DRAGON12 board is pressed. If you do not see this, the bottom window may be set for message window. Click the terminal button in the bottom window to enable the terminal window display.

```
D-Bug12 v4.0.0b14
Copyright 1996 - 2002 Motorola Semiconductor
For Commands type "Help"
>
```

SOFTWARE DEVELOPMENT

The MC9S12DP256 memory map in single chip mode is as follows:

$0000-$03FF Registers

$0400-$0FFF EEPROM

$1000-$3FFF RAM

$4000-$7FFF 16K fixed Flash

$8000-$BFFF 16K page window

$C000-$FFFF 16K fixed Flash

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

All sample programs must be run in EVB mode. In order to run the test program in EVB mode, the two DIP switches on the SW7 must be set at the low positions, so the **EVB mode LED is lit.**

1. Click the File button to open the test.asm from c:\DRAGON12\examples. After the test.asm is loaded into the top window, you can view instructions of how to test all hardware on DRAGON12 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 v4.0.0b14
Copyright 1996 - 2002 Motorola Semiconductor
For Commands type "Help"
>
```

4. Type “LOAD” <Enter>.

5. Click the Build button. Select Download option and select the file ‘test.s19’ for downloading.

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

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 a new program may not work if an interrupt was enabled by a previous program. In all example 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. You must place your interrupt vectors at $3E00-$3E7F, because real interrupt vector addresses are taken by the bootloader, the bootloader will redirect interrupts to D-Bug12 monitor which will redirect them to the RAM interrupt vector addresses 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 page 12 of the D-bug12 V4.x reference manual on the CD.

The AN1280 was written for OLD HC12 family. If you happen to use printf routine with your old HC12 board you should be aware that I/O utility routines are moved to different addresses in D-bug12 V4.x.

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. The address for the printf is $EE88, see

page 79 of the D-bug12 V4.x reference guide on the CD for addresses of other I/O routines.

The bootloader (**AN2153.PDF)**, the D-bug12 reference guide (**DB12RG4.PDF**) and the MC9S12DP256 data book **(MC9SDP256.PDF**) are the most important documentation. They can be found on the folder named C:\DRAGON12\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.

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

Four operating modes:

The MC9S12DP256 on the DRAGON12 board is pre-loaded with a bootloader and the D-Bug12 monitor firmware and it will operate in 4 different modes depending on the setting of the 2-position DIPswitch, SW7. The current mode is indicated by the LEDs above the SW7. During power up or reset, the MC9S12 will read the PAD0 and PAD1 to decide which mode to boot up and a corresponding LED will be lit to indicate that mode.

Mode0:
EVB mode: PAD1=0, PAD0=0.
This is the standard debug environment running on the MC9S12DP256 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 Motorola 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 the bootloader, the bootloader and the D-Bug12 monitor will redirect interrupts to the RAM interrupt vector addresses at $3E00-$3E7F.

After booting in this mode, you would see the following after reset:

```
D-Bug12 v4.0.0b14
Copyright 1996 - 2002 Motorola 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.

Mode1:
Jump-to-EEPROM mode: PAD1=0, PAD0=1.
This mode enables the MC9S12DP256 to jump directly to the internal EEPROM at location $0400 upon reset. This mode makes the MC9S12DP256 a replacement for the old 68HC811E2 microcontroller, but it also gives you 3K EEPROM instead of 2K EEPROM with the 68HC811E2.

Mode2:
POD mode: PAD1=1, PAD0=0.
In this POD mode, the D-Bug12 firmware acts as a master to access all target MCU resources on the target board (another DRAGON12 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 DRAGON12 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 polarity is correct). You don't have to provide the power to both boards, only to one board. Both boards should be set at single chip mode for the

best results (the MODEB, MODEA, and MODEC are set for 0-0-1). 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 after reset:

```
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 choice 1). After enter 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 MC9S12DP256 is running from the crystal frequency, not the PLL frequency. Enter 4000 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 (4000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
? 1

Enter Target Crystal Frequency (kHz): 4000



Can't Communicate With Target CPU

1.) Set Target Speed (4000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
? 2
```

When the communication is established, you will see the following:

```
D-Bug12 v4.0.0b14
Copyright 1996 - 2002 Motorola 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 MC9S12DP256 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>
```

In order to program the flash memory, you have to erase it through the use of 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 MC9S12DP256 including the bootloader. If it returns with a message “Flash or EEPROM Failed To Erase” the MC9S12DP256 is defective.

Now we are going to program the D-bug12 and bootloader into the flash memory of the target MC9SDP256. Before we actually program the flash memory, we must understand there are two different types of s-record files 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 detail. Right now, all you need to know is that the bootloader s-record file is an s1-record file, but the D-bug12 monitor s-record file is an s2-record file. They will be handled differently by the FLOAD command.

At first, we are going to program the bootloader by the “FLOAD ;B” command. The “FLOAD” command in the D-bug12 is for downloading an s2 record file. You need the “;B” to be added behind the “FLOAD” for downloading an s1 record file. Since the bootloader file Ep9S12_boot.s19 is an s1 record file, we need to type “fload ;b” <Enter> at the prompt. Click the Build button. Select the Download option and select the file **Ep9S12_boot.s19** located in the folder named “document”. You should see the following on the terminal window when programming is done:

```
S>fload ;b <Enter>

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

Now we are going to download the D-bug12 monitor into the flash memory. We need to type “fload” <Enter> at the prompt. The “;b” is not needed because the monitor file Ep9S12_mon.s19 is an s2-record. (Sometimes, it’s confusing because the file extension is .s19). Click the Build button, select the Download option, and select the file **Ep9S12_mon.s19** located in the folder named “document”. You should see the following on the terminal window when programming completes (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 DRAGON12 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 don't even have to turn off the power while doing this.

The D-bug12 monitor is an application program under the bootloader. If you program the D-bug12 portion of flash memory with your application program, your program will run automatically in the mode (EVB mode) after power up or reset.

As the same as in the mode1, the bus speed is 2MHz, 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.

Mode3:
BOOTLOADER mode: PAD1=1, PAD0=1.

This bootloader allows you to erase/program flash 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 or 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 MC9S12DP256 on its own board, but not the bootloader portion. **The bootloader can only be erased by a BDM via the BDMin port.**

```
HCS912DP256 Bootloader menu:

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

The a) option will erase the D-bug12 portion of the flash memory, not the bootloader itself.
The b) option will program the D-bug12 portion of the flash memory, not the bootloader itself.
The c) option will set a new baud rate.
The d) option will erase all on-chip EEPROM.

To program flash memory with the D-bug12 monitor:

1. Enter the b option
2. Type "fload" <Enter> at the prompt.
3. Click the Build button, select the Download option, and select the file Ep9S12_mon.s19 located in the folder named "document" for downloading. You should see the same thing on the terminal window shown on the previous page.

How to use the DRAGON12 board to develop or test your code? It depends if you use a BDM or not.

If you don't use a BDM, there are 3 approaches:

1. Use on-chip 12K RAM for code 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 the 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.

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

   In this mode, you download your program directly into the 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 s19 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 SW 7 to EVB mode, then press the reset button. It usually runs the D-bug12 monitor. Now, it will run 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 MC9S12DP256 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 Motorola s-record conversion utility **sreccvt.exe** to convert your odd format to the even format by using the following command line:

   Srccvt –m c0000 fffff 32 –of f0000 –o test_even.s19 test.s19

   It will create a new file named test_even.s19 that has the even format and can be programmed into the flash memory. For your convenience, the sreccvt.exe is included in CDROM\document.

If you ever use a BDM device with the DRAGON12 board, there is a different method for debugging. The BDM stands for background debug mode. With a BDM device, you don't need the D-bug12 monitor and the bootloader to physically reside in flash memory of the MC9S12DP256. So, a BDM device allows the use of all on-chip 256K flash memory of the MC9S12DP256 for your program code, including the portion of the

bootloader. You also can use the real interrupt vector addresses in your program. There aren't anymore pseudo RAM vector addresses.

When a BDM device is connected to the BDM IN of the DRAGON12 board, it can interrogate the state of the MC9S12DP256 microcontroller in real-time without stopping the program. The BDM device also permits single stepping and setting hardware breakpoints in flash memory, so you can debug your code in flash memory.

This is a new product and you will receive some software upgrade in the future. If you would like to get a free upgrade, you should send us an email once in a while to check out the status of our software.

The web is the best source for getting more information about the HCS12. The Motorola 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 HC12 gurus.

All HCS12 boards on the market today are commodity products and are basically the same, because they are all using the Motorola bootloader and a D-bug12 monitor. If you are going to use a BDM to debug a HCS12 board, all boards will be exactly the same because the BDM communicates with the MC9S12DP256 MCU. The information on our manual can apply to the boards from other manufacturers, and vice versa.

Here are some useful links of HC12 information:

http://www.almy.us/68hc12.html
http://www.almy.us/classes/index.html
http://www.almy.us/classes/assignments/EET338andDragon12.html

http://www.ece.utep.edu/courses/web3376/site_map.html University of Texas at El Paso web site
http://opentech.durhamc.on.ca/bertrandl/courses/mpro1131/index.php?dir=resources Durham College

http://www.ece.utexas.edu/~valvano/ University of Texas web site
http://www.ee.nmt.edu/~rison/ee308/labs/labs.html University of new Mexico web site
http://www.ee.nmt.edu/~rison/ee308_spr01/lectures.html DAC source code for MAX522
http://www.ee.nmt.edu/~rison/ee308_spr02/lectures.html
http://www.coe.montana.edu/ee/courses/ee/ee371/ee371hpg.htm University of Montana web site

http://www.geocities.com/englere_geo/ Free GNU development tools

http://sapphire.indstate.edu/~lzhao/application_notes_for_motorola.htm Free app note for C demo programs
http://www.ezl.com/~rsch/DS1302.htm Free software for DS1302 real time clock
http://www.ezl.com/~rsch/USB.htm Free software for USB applications

http://eet.etec.wwu.edu/CPU12/CPU12.html Western Washington University web site
http://www.technologicalarts.com/myfiles/WMU1.html Students at Western Michigan university have to pay US$135.15 + US$12 s&h at Special Student Price for a small MC9S12DP256 board. They could have bought our MiniDRAGON+ board ($95.00 with shipping) with more on-board I/O features for less money.

http://www.feaser.com/pdf/AN1002_Getting_Started_with_HCS12.pdf Free MiniDRAGON app note
http://www.eikimartinson.com/engineering/pipe/pipecrawler.pdf MiniDRAGON in FAU's Robot

ON-BOARD HARDWARE

Each port B line is monitored by an LED. It works OK in single chip mode. If the board is used in expanded mode, the port B becomes the address/data bus, AD0-AD7, and the LEDs will add too much load on the bus. In order to make it work in expanded mode, J24A and J24B must be removed to disable the 7-segment LED display and the PB0-PB7 LEDs.

Port A is used as the 4X4 keypad interface in single chip mode, but in expanded mode, port A becomes the address/data bus AD8-AD15 and it cannot be connected with a keypad.

Port H is connected to an 8-position DIPswitch. The DIPswitch is connected to GND via the RN9 (eight 4.7K resistors), so it's not dead short to GND. When port H is programmed as an output port, the DIPswitch setting is ignored.

If the user wants to replace the on-board LCD display module with a new one, the trimmer VR1 can be used for LCD contrast adjustment. If a new LCD is installed, the user can install a 5-10K trimmer pot to adjust the new LCD display module's contrast. The VR2 is connected to the AN07 input of the ADC port via J17, but the trace under J17 can be cut if AN07 must be used by target circuits.

An on-board logic probe LED is connected to pin 47 of header H4 and can be used to monitor a high or low status at any point of the circuit as a logic probe. Pin 47 of U10 (MC9S12DP256) is not connected to header H4.

U7, 4093, generates a 38KHz square wave for the IR transmitter. One of 4093's gate is used as a 38 KHz oscillator. The value of resistor R10 may vary if the 4093 is manufactured by a different company.

U5, SN75176, converts the TTL signal from SCI1 to RS485 differential signals and vice versa.
The two RJ12 jacks, JK1 and JK2, can be used to daisy chain many DRAGON12 boards together for a network application. The connections on JK1 and JK2 are identical, so either one can be input or output.

PJ0 (pin 22) from the MC9S12 is used to control the direction of the RS485 communication. If PJ0=0, the RS485 port, U9 DS75176, is set as the receiver port. If PJ0=1, the RS485 port, U9 DS75176, is set as the transmitter port.

PJ0=0 RS485 receiver port
PJ0=1 RS485 transmitter port

The crystal frequency is 4 MHz and usually it will result in a 2 MHz bus speed, but on this board the MC9S12DP256's internal PLL brings the bus speed to 24 MHz. It's just incredibly fast!

The circuit is designed in such way that the value of all resistors and capacitors are not critical, with the exception of R10 and C36 which determine the 38KHz for IR transmitter. They can be off -50% or +100%.

How to use port K:

Port K 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 J11 and J12 are as follows:

Pin 1 GND
Pin 2 VCC (5V)
Pin 3 Via a 100 Ohm resistor to GND
Pin 4 PK0 RS pin for LCD module

Pin 5	GND	
Pin 6	PK1	EN pin for LCD module
Pin 7	Not used	
Pin 8	Not used	
Pin 9	Not used	
Pin 10	Mot used	
Pin 11	PK2	DB4 pin for LCD module
Pin 12	PK3	DB5 pin for LCD module
Pin 13	PK4	DB6 pin for LCD module
Pin 14	PK5	DB7 pin for LCD module
Pin 15	Via a 10 Ohm resistor to VCC	LED backlight for LCD module
Pin 16	GND	

Please notice that PK2-PK5 (not PK4-PK7) are used to drive DB4-DB7 of the LCD module.

How to use port A:

Port A is an 8-bit bi-directional port. Its primary usage is for a 4X4 keypad interface. If the port is not used for the keypad, it can be used as a general-purpose I/O.
The pinouts of J29 is as follows:

Pin 1	GND	
Pin 2	PA0	connects ROW0 of the keypad
Pin 3	PA1	connects ROW1 of the keypad
Pin 4	PA2	connects ROW2 of the keypad
Pin 5	PA3	connects ROW3 of the keypad
Pin 6	PA4	connects COL0 of the keypad
Pin 7	PA5	connects COL1 of the keypad
Pin 8	PA6	connects COL2 of the keypad
Pin 9	PA7	connects COL3 of the keypad
Pin10	VCC	

An 8-pin male header is installed on pin 2 through pin 8. Pins 1 and 10 are not installed.
Keypad interface

PA0 connects ROW0 of the keypad via pin 1 of the 8-pin keypad header
PA1 connects ROW1 of the keypad via pin 2 of the 8-pin keypad header
PA2 connects ROW2 of the keypad via pin 3 of the 8-pin keypad header
PA3 connects ROW3 of the keypad via pin 4 of the 8-pin keypad header

PA4 connects COL0 of the keypad via pin 5 of the 8-pin keypad header
PA5 connects COL1 of the keypad via pin 6 of the 8-pin keypad header
PA6 connects COL2 of the keypad via pin 7 of the 8-pin keypad header
PA7 connects COL3 of the keypad via pin 8 of the 8-pin keypad header

The PA0-PA7 has a 100K pull-up resistor in each line.
The keypad scan routine sets PA3 low and PA0, PA1, and PA2 high. The routine 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.

The keypad scan routine then sets PA2 low and PA0, PA1, and PA3 high. The routine 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.

The keypad scan routine then sets PA1 low and PA0, PA2, and PA3 high. The routine 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.

The keypad scan routine then sets PA0 low and PA1, PA2, and PA3 high. The routine 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.

SPI port (J10) pinouts are as follows:

Pin 1	VCC (5V)	Pin 2	VCC (5V)
Pin 3	PM7 (LOAD)	Pin 4	PS4 (SPI DATA IN)
Pin 5	PS7 (STROBE)	Pin 6	PS5 (SPI DATA OUT)
Pin 7	not used	Pin 8	PS6 (CLOCK)
Pin 9	GND	Pin 10	GND

All on-board jumpers:

J1 Enables the LCD backlight. If the LCD display is not needed in an application, remove this jumper to turn off the backlight and save the power. You may leave it if the AC adapter can sufficiently provide the power to all circuits.
J2 CAN0 interface
J3 CAN1 interface
J4 CAN2 interface
J5 PK7
J6 PP4 PWM output for Robot servo
J7 PP5 PWM output for Robot servo
J8 PP6 PWM output for Robot servo
J9 PP7 PWM output for Robot servo
J10 SPI connector
J11 LCD port
J12 LCD port
J13 RS of CAN0 (U2), is connected to VSS
J14 RS of CAN1 (U3), is connected to VSS
J15 RS of CAN2 (U4), is connected to VSS
J16 Analog voltage reference VRH. It's connected to the on-board 5V DC reference voltage source, but trace under VRH can be cut if a more accurate analog reference voltage is needed.
J17 Connects the VR2 trimmer pot to AN07 of the ADC.
J18 CAN0 or RS485 selector

When it's in the up position (labeled with 'CAN'), JK1 and JK2 are used for the CAN0 interface.
When it's in the low position (labeled with 'RS485'), JK1 and JK2 are used for the RS485 interface.

J19 Connects the MC9S12 SCI's PS3 (TXD1) to all communication hardware (RS232, RS485, RF and IR transceiver) on this development board.

J20 BDM input
J21 BDM output, when the board is booted in POD mode
J22 RS485 direction control by PJ0 (pin22) through this jumper

J23 MC9S12 SCI1receiver source selector (numbering from top to bottom)
1= SCI1 PS2 receives signal from your target system via pin 91 of header H7
. The pin 91 of the U10 (MC9S12DP256) is not connected to header H7
2= SCI1 PS2 receives signal from P2. DB9 connector for RS232 input
3= SCI1 PS2 receives signal from JK1 or JK2 (RJ12 jacks) for RS485 input
4= SCI1 PS2 receives signal from the on-board RF receiver
5= SCI1 PS2 receives signal from the on-board IR receiver. This is the default setting.

J24A Enables the 7 segment LED display driver U6, 74HC367. If the 7 segment LED display is not needed in an application, remove this jumper to turn off the display and save power.

J24B Enables the PB0-PB7 LEDs. If the PB0-PB7 LEDs are not needed in an application, remove this jumper to turn off the LEDs and save power.

J25 SDI connector

J26 Enables speaker. The speaker is driven by PT5 (pin 16), Output Comparator 3, through this jumper.

J27 IR transceiver control source selector

When it's in the left position (this is the default position, labeled with 'SCI1'), the MC9S12's PS3 (TXD1) drives the IR transmitter and PS2 (RXD1) receives the data from the IR receiver. PS3 and PS2 can be programmed as general I/O lines or SCI UART.
When it's in the right position (labeled with 'PT4 and PT3'), the MC9S12's PT4 drives the IR transmitter and PT3 receives data from the IR receiver.

J28 Test mode and it's installed via a cut-off trace for defeating the test mode.

J29 4 X 4 keypad interface via PA0-PA7

J30 Mode B selector, it defaults at 0

J31 Mode A selector, it defaults at 0

J32 Mode C selector, it defaults at 1

The P2 DB9 female connector is configured as a **DCE** device and it can be directly connected to the PC 's COM port.

Application Circuit Corner (ACC) and RF transceiver:

For the user's convenience, the upper right corner of the PC board has footprints of four popular 68HC11/HC12 applications. The **ACC** consists of a DS1302 Real Time Clock with battery backup, a DS1620 Digital Thermometer/Thermostat, a 12V DPDT relay, and a L293D Motor driver. In the future, we will offer each circuit in a separate kit. With these application circuits, this board can easily be transformed into a complete platform for use as a robot, home automation or other useful applications.

The source code for DS1302 real time clock can be found at Roger Shaefer's web site:
http://www.ezl.com/~rsch/DS1302.htm

The RF transceiver is not available now, but we will offer a kit in the future.

IMPORTANT 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 new beginners. For instance, if you want to run the board in single chip mode, but MODEB, A, and C are set for expanded mode, you know it won't work. If the jumper on J1 is missing, the LCD backlight won't work and if the jumper on J24 is missing, the 7-segment display won't be lit.

Before troubleshooting the board, you must apply power to the board. When the board is powered, one of the 4 LED mode indicators (at the lower right side of the PC board) must be on. If none of them are on, the board does not have 5V DC. Use a DMM to check voltages at different points in the circuit to find a defective component. 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 board jumper settings to the original default settings when you receive the board. The default settings are as follows:

J1	LCD_EN	jumper is on
J5	PK7	jumper is on
J18	CAN/RS485	both jumpers are set for RS485 (at the lower positions)
J23	SCI 1 select	jumper is set for IR (at the bottom position)
J24	LED_EN	jumper is on
J27	IR select	both jumpers are set for the SCI 1 (at the left positions)
J30	MODEB	jumper is set for '0' (at the right position)
J31	MODEA	jumper is set for '0' (at the right position)
J32	MODEC	jumper is set for '1' (at the left position)
SW7	OP MODE select	both DIP switches are set for EVB mode (at the lower positions)
RN5	current limit for DSP1	RN5 is installed
RN6	current limit for DSP2	RN6 is installed
RN8	current limit for PB0-PB7 LEDs	RN8 is installed
Y1	4 MHz crystal	Y1 is installed

If all above settings are correct and you press the reset button, the PB0-PB7 LEDs should chase from left to right. If this does not occur, the bootloader could have been erased by a BDM. If the LEDs chased, but the board does not communicate with the PC, the EVB portion of flash memory could be erased or the com port number may not be set correctly by AsmIDE. If the screen displays some garbled characters, the baud rate may not be set correctly. The D-bug12 resets the baud rate to 9600 during power up, if you changed the baud, you must change the AsmIDE's baud rate to the same value.

The crystal is not soldered to the board and it's held by two machine pins. If you want to change the crystal, just unplug it and replace it with a new one. Don't cut a crystal's leads too short. If it's shorter than ¼", its metal case could short the two machine pins.

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 in RAM, it's a good idea to add instructions to initialize the RAM interrupt vectors in the very beginning of the program. If you don't do it and your application program uses interrupts, after you press the reset button which will clear the interrupt vectors, your program may not run correctly since the interrupt vectors do not exist.

4. Disconnect the LCD module from the bottom of the main PCB first.

The LCD display module has 2 supporting nylon spacers. They fit tighter with the LCD module PCB than with the main PCB. So if you need to remove the LCD module from the board, the spacers should stay with the LCD module. That means you should use a pliers to push up the spacers from the bottom of the main PCB to separate the spacers from the main PCB. Once the spacers are loose from the main PCB, you can easily unplug the LCD module.

5. Operating mode changing is only effective after reset.

There are four operating modes that are selected by SW7. The LED mode indictor changes immediately, but the mode change won't be effective until you reset the board. So you must always press the reset button after a mode change.

6. Things to do if use the board in expanded mode.

The 7-segment LED display module and 8 LEDs add a heavy load onto port B. It works in single chip mode OK, but it will have a problem when port B is an address/data port in expanded mode. If you have to use the board in expanded mode, you have to remove J24A and J24B. By removing J24A, you tri-state the 7-segment LED display module and basically disconnect the module from port B. By removing J24B, you disconnect the PB0-PB7 LEDs from port B.

7. Learn HC12 programming by modifying HC11 code.

When you learn programming on a new microcontroller family, you usually would create more bugs in your code. Sometimes you really cannot tell if your code is bad or the logic is bad. Here you have many example programs to hack. These fully debugged example programs are written in 68HC11 code. You can gradually replace the HC11 code with its equivalent HC12 code. You can run the program to verify your HC12 code immediately after each small replacement. Step by step, after you complete your code migration from HC11 to HC12 for the test.asm, you will become a pretty good HC12 programmer.