

Software & Hardware Engineering: Motorola 68HC12, by Fred Cady and James Sibigroth, published by Oxford University Press, ISBN 0-19-512469-3

Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, Brooks/Cole Publishing Co

New resources for using 68HC12 are becoming available all the time. Listed below are a few examples:

- MiniIDE - W9x/NT Integrated Development Environment with editor, assembler, terminal window
- HC12 Simulator - Windows-based 68HC12 Simulator
- Textbooks - there are at least four textbooks now available
- Software - many example subroutines in C and assembler are available for download

Check both the RESOURCES and APPLICATIONS pages on our website for links to the above and more.

1 INTRODUCTION

Congratulations!

You are now ready to explore the power and flexibility of Motorola's 68HC912B32 microcontroller with a unique and useful tool-- Adapt912. Your questions and comments are always welcome. We provide friendly, knowledgeable technical support by telephone, fax, and e-mail to all our customers. As well, we have a comprehensive website with a resource page featuring new information, software, and links to other useful sites on the Internet. See back cover for how to contact Technological Arts.

Purpose of Adapt912...

Adapt912 was designed as an evaluation and application tool for the Motorola MC68HC912B32 microcontroller. It is unique among evaluation boards in that it will easily plug vertically into any standard solderless breadboard, with the supplied adapter. It is a fully functional, standalone implementation of a 68HC912B32 configuration, and can be treated just like a chip in your breadboard. Then it is simply a matter of wiring up the desired application circuits and downloading the appropriate code into the micro to evaluate and develop your application ideas.

Product Configuration

The Adapt912 Starter Package features the 68HC912B32, with 1K SRAM, 768 bytes EEPROM, and 32K on-chip Flash. Motorola's DBug12 monitor/debugger is resident in the Flash memory, facilitating easy loading and debugging of your code via your PC serial port, using any terminal program. Logic levels at MCU pins PAD0 & PAD1 (JB2 "MODE SELECT") are sensed by DBug12 directly after reset, to decide whether to jump to internal EEPROM, run the program in on-chip Flash (Motorola's DBug12 monitor/debugger or a user-installed program), or operate in BDM pod mode. A second jumper block (JB1) is provided for setting MODA and MODB logic levels. These levels are sensed by the MCU after reset to determine the chip's operating mode (single-chip, narrow expanded, or wide expanded memory modes). In

expanded modes, the multiplexed address and data bus lines are brought out to a second 50-pin connector, H2. Operating in single-chip mode, all I/O ports are available for user applications. When operated in expanded-chip mode, additional memory can be added externally; however, two of the I/O ports form the address and data buses, and are thus unavailable for user applications.

Communications

An RS-232-compatible serial interface port (RX & TX only) is built into Adapt912, allowing communication with a PC, or any other device which has an RS-232 serial port. The logic-level RXD and TXD signals from the micro are also brought out to the 50-pin header, for applications such as MIDI. An RS-485 serial port is implemented as well, and can be selected via slide switch SW2. RS-485 is commonly used in networking industrial control applications, where long cable lengths and good noise immunity are required.

How is Adapt912 different from other evaluation boards ?

Most evaluation and development systems available tend to be too expensive and bulky for embedding into a real application. Also, the prototyping area provided is often limited, and does not lend itself to re-usability. By contrast, we took a modular approach. With the ADAPT12 system, all I/O lines and control signals are brought out to two standard 50-pin interface connectors. With several different connector options available, you can use the module in whatever way best suits your needs. With the solderless breadboard adapter supplied with your Starter Package, you can treat the module like a big chip, and plug it right into your breadboard. Forget soldering or wire-wrapping-- get started developing your application right away. Your prototyping space is virtually unlimited, using solderless breadboards! When you've got a design working and you're ready to make it permanent, modular prototyping accessories are available, which give you the ability to easily build fully customized, compact applications at low cost. A full range of accessories including backplanes, prototyping cards, memory expansion, and application-specific cards is available, and

5.0 SOURCES

Internet Resources

- Technological Arts: (new product info, files, tips)
www.technologicalarts.com
support@technologicalarts.com
sales@technologicalarts.com
- Motorola Freeware: <http://freeware.aus.sps.mot.com/freeweb/>
- Motorola Literature:
www.mcu.motsps.com/documentation/index.html
- Karl Lunt (SBASIC): <http://www.seanet.com/~karllunt>
- ImageCraft (ICC12): www.imagecraft.com/software

Publications

Data Books:

Motorola Semiconductor Literature Distribution Center
P.O. Box 20912, Phoenix, AZ 85036 1-800-441-2447

- CPU12 Reference Manual (CPU12RM/AD)
- CPU12 Reference Guide (CPU12RG/D)
- MC68HC912B32 Advance Information (MC68HC912B32/D)

Textbooks:

Programming the Motorola M68HC12 Family, by Gordon Doughman, published by Annabooks (California), ISBN 0-929392-67-1 (Order Code PMM929, available from Technological Arts)

Single- and Multi-Chip Microcontroller Interfacing for the Motorola 68HC12, by G. Jack Lipovski, published by Academic Press, ISBN 0-12-451830-3

click OK. When downloading has finished, re-configure JB2 for JMP-EE mode (for EEPROM) or EVB mode (for Flash), and press RESET. The program you just loaded into EEPROM (or Flash) will execute.

There are a couple of important things to be aware of. When you are using the JMP-EE function, the code you loaded into EEPROM starting at 0xd00 is executed immediately (ie. no reset vector required). However, if you are planning to put your program in Flash, you need to include the pseudo-vector table (or, at a minimum, the pseudo-reset vector). Refer to Table 4-1 in Motorola's 68HC912B32 data book for the Interrupt Vector Map. ImageCraft has implemented this vector table in a file called **vectors.c** in the **Examples** folder. To create a pseudo-vector table, modify this file by changing the **pragma abs_address** to the pseudo-vector table's base address, as discussed above, and save it as **pvector.c**. Then include **pvector.c** in your project or add **#include "pvector.c"** at the end of your program.

4.3 Using SBASIC

Use the appropriate **/c**, **/v** and **/s** compiler options to specify the starting addresses for code (EEPROM or Flash, usually), variables (RAM), and stack (end of RAM), respectively, where **myprog.bas** is your SBASIC program filename, and **myprog.asc** is the name you want the target assembly language file to be called. For example, to specify Flash as the target memory area for your code, enter:

```
sbasic myprog.bas /c8000 /v0800 /s0c00 >myprog.asc
```

After successful compilation, edit the ORG statement at the end of **myprog.asc** to correspond to the appropriate pseudo vector address. Then run **asmhc12** to create an s-record file, as follows:

```
asmhc12 myprog.asc
```

Once you have a .s19 file, you can use any of the downloading methods outlined previously to load it into Flash.

more are being developed all the time. Visit our website frequently to see what's new.

2 USING ADAPT912 WITH SOLDERLESS BREADBOARDS

Your Adapt912 Starter Package comes with a 50-pin adapter (ADHDR50-F) to allow you to plug the module into a solderless breadboard ("protoboard"). This adapter may be used on either connector (H1 or H2).

CAUTION!

Never insert or remove your module from a "live" breadboard. Make sure the power is OFF !

- 1) Any breadboard will do; however, you will find that the kind made with a softer, more pliable plastic (such as nylon) will be easier to use and more durable.
- 2) When plugging the adapter into your breadboard, press gently but firmly, rocking it back and forth slightly, until the pins are seated in the sockets. Then plug Adapt912 into the adapter. Note that either H1 or H2 can be used with the adapter. Use the same side-to-side (not end-to-end) gentle rocking motion, while pulling gently upward, to remove the adapter.
- 3) Use the middle area of your breadboard strip to allow maximum access on each end to all the signals. If possible, place an additional breadboard section in parallel on each side for easier wiring of your circuits. (HELPFUL HINT: If you are using the Analog inputs, make sure to wire your analog circuits as close to these pins as possible, to keep noise levels down.)
- 4) Choose a convention for wiring your power distribution buses. A logical approach is to make the inside bus logic 5V, and the outside buses GROUND. Never supply external power via J1 if you are supplying 5VDC via the breadboard connector pins. However, always connect the breadboard GROUND to the module GROUND.
- 5) If you are using voltages other than 5V, make sure to keep these well away from Adapt912 pins and tie-strips, to avoid accidental shorts which may damage the module.

3 TUTORIAL

Note that this manual is not meant to provide an exhaustive study of the 68HC12 family of microcontrollers, but rather to help you get started using the Adapt912 microcontroller board as a learning and application development tool for 68HC912B32, whether you're a beginner or an expert. If you are a beginner, you will benefit from additional material listed in the Reference section of this manual, and links provided on the Resource page of our website (see back cover for URL).

CAUTION!

Never apply power to your module with the Vfp switch on! This may damage the MCU. Make sure Vfp is OFF at all times except when erasing and programming Flash!

3.1 Getting Started

Adapt912 comes to you with Motorola's DDebug12 monitor/debugger in Flash. This is a useful program for testing your communications setup, developing programs, or using Adapt912 as a BDM pod for the purposes of developing/debugging another target HC12 board.

You can power the module in one of two ways:

1) supply power via the external power connector; just connect a DC voltage of 9 Volts or more (maximum 12V) to the external power connector J1. A DC power supply capable of supplying at least 200 mA is adequate. **Note: erasing and programming on-chip Flash requires regulated 12VDC. If you have the Adapt912B version of the board, the on-board Vfp generator circuit provides the required 12V. If not, you will need to provide regulated 12VDC via J1.** Red is positive, and black is negative (ground). **CAUTION! Make sure you have the polarity correct!**

2) or, supply regulated 5VDC via the appropriate pins on the 50-pin connector (H1). With this method, you will not be able to program/erase Flash (unless you have the Adapt912B version of the board with on-board Vfp generator circuit). See module pinout

wait for another * character before sending the next s-record. The process repeats until the bootloader receives an S9 record, indicating end-of-file, or until the board is reset. The s-record data must be within the Flash address range \$8000 - \$f7ff, otherwise you will see error messages on the terminal screen. **LoadEE** is for loading an s-record into the 768-byte EEPROM, which does not require any special programming voltage. It automatically performs a bulk erase before loading the s-record.

Flash. The on-chip Bootloader approach provides easy loading without the use of a BDM pod; however, it necessitates a few compromises. First, the top 2K of Flash is reserved, and is not available for your program. Secondly, since the MCU's vector space is in the protected block, it requires the use of pseudo-vectors, which DDebug12 implements in a block of the user-programmable area of Flash. To use any vector, simply initialize the corresponding pseudo-vector with the address of your interrupt service routine. The bootloader will intercept the interrupt and jump to your routine. This extra level of indirection adds a small amount of latency to interrupt servicing. The pseudo-vector block is from \$f7c0 - \$f7fe, and follows the same sequence as the real vectors. **EEPROM.** While the 768 bytes of on-chip EEPROM may not seem like much space for your code, it is possible to call DDebug12 routines from within your program. Motorola has written an Application Note on this subject that you may wish to read. Visit the RESOURCES page of our website for a link to Motorola's Literature site.

4.2 Using ICC12 for Windows

Before compiling, set up the linker sections with 0x0800 for *data* (RAM), 0x8000 (Flash) or 0x0d00 (EEPROM) for *text*, and the *stack* at 0x0c00. After compiling to executable, download the resulting s-record file using ICC12's terminal window. Configure the MODE SELECT jumpers (JB2) for BOOTLOAD Mode, and set the terminal communication options to **9600** baud, **wait for * prompt**, and **no character delay**. Reset the board, and select the operation you wish to perform (Erase, Program Flash, or LoadEE). For both Program and LoadEE, click on the **ASCII Download** button, and select the **.s19** file you wish to download. Reset Adapt912 and

serial EEPROM, 8-bit SPI-based output port, high-speed UART with RS232 or RS485 interface. The expansion RAM is configured for Expanded Wide operation, and has both Bank Select and Write Protect features. It is ideal for educational and code-development applications, where heavy usage may make Flash durability a concern. Details on Adapt912MX1 can be found on the website, along with an Application Note describing in detail suggestions for its use.

4 REFERENCE

NOTE: *Adapt912 contains DBug12 in Flash, just like the Motorola 912 EVB. Motorola's complete EVB912 manual is available in Acrobat format from Motorola's website. Visit the TECH SUPPORT page of our website for a link to download this highly-recommended document.*

4.1 BOOTLOAD Mode

Adapt912 uses on-chip 768-byte EEPROM or 32K Flash for program storage. A small bootloader is part of the DBug12 program in Flash, and resides in a protected boot-block. Even if you decide to erase Flash and install your own program via BOOTLOAD mode, the boot-block will not be erased. At some later time, you can re-load DBug12 using BOOTLOAD mode.

Using a terminal program, reset the board with JB2 set for BOOTLOAD mode. A prompt will appear on your terminal screen, as follows:

(E)rase, (P)rogram or (L)oadEE:

Select the desired function by typing the letter (upper or lower case) E, P, or L. *Note: Erase and Program relate to Flash, so a regulated 12V source is needed for Vfp. Unless you have the Adapt912B version of the board, with on-board Vfp generator, you will need to apply regulated 12VDC as your power source on J1. Leave the Vfp switch (SW3) off except when erasing or programming Flash. Erase simply erases the unprotected part of Flash (ie. the lower 30K). Program sends an ASCII * (asterisk character) to the host (the terminal program running on your PC), indicating it is ready to receive the first s-record. Your terminal program needs to*

diagram included with your manual for the 5V and GROUND pins to connect to on H1. **CAUTION! Double-check your connections before applying power!**

To run the program in Flash (DBug12, if you haven't replaced it yet) select **EVb** mode by setting both MODE SELECT jumpers (on JB2) to the 0 position. Connect the supplied serial cable between Adapt912 and a serial port on your computer. (With some PCs, you will need a 9-pin to 25-pin adapter.) Run any terminal program on your PC. Make sure your terminal program is set to 9600, parity to NONE, # DATA BITS = 8, and #STOP BITS = 1. Power up Adapt912, or press RESET button (SW1). The DBug12 prompt will appear in your terminal window. Type **HELP** to get a list of DBug12 commands.

Some examples of suitable terminal programs are: ProCommPlus (DOS), Windows Terminal program (included with W3.1), HyperTerminal (included with W95/98), miniIDE (W95/98/NT freeware integrated development environment for developing HC12 applications in assembler), or the Terminal function in ImageCraft's Windows-based Integrated C Development Environment for 68HC12.

3.2 Modes

Adapt912 (as shipped, with DBug12 in Flash) can be started up in any one of four modes, selected via jumper block JB2 (MODE SELECT), as follows:

PAD0	PAD1	MODE	DESCRIPTION
0	0	EVb	DBug12 is executed from Flash
0	1	POD	use as BDM pod via BDM OUT
1	0	JMP-EE	run user program in EEPROM
1	1	BOOTLOAD	load user code to Flash/EEPROM

3.3 Writing Your First Program

If you are already experienced with the 68HC11 family of microcontrollers, writing 68HC12 programs will not present a big challenge. In fact, you can use your existing 68HC11 assembly code and re-assemble it for the 68HC12. There are a couple of

things to keep in mind when doing this. The first is assembler syntax. You may need to edit your source file to conform to the syntax and directives requirements of the HC12 assembler you are using. There are several assemblers available, and each has its own syntax to be aware of. Another departure from the 68HC11 is that the register block default location is \$0000 and the RAM is at \$0800. This means you would initialize the Stack Pointer to \$0c00 (on the HC12, it should point to the address following the last RAM location). Also, the HC12 bus speed is a lot higher than the HC11. This will mean changing some initialization values for control registers and revising delay constants if you are doing software timing loops.

There are several documents that detail the new instructions and addressing modes of the 68HC12, explain differences from the 68HC11, and give examples of working with the 68HC12. Whether you're programming in assembler or not, you should definitely get the Motorola CPU12 Reference Manual and the applicable Application Notes, available from the Motorola Literature Center or in Acrobat format from their website. See our RESOURCES webpage for a link to Motorola and for links to dozens of other useful sites. A highly recommended book is *Programming the Motorola M68HC12 Family*, written by Motorola engineer Gordon Doughman (who is also the creator of DBug12). This book is available from Technological Arts (order code PMM929).

3.4 Downloading Your Code to Adapt912

Once you have assembled your code with no errors, you can download the resulting s-record file (*filename.s19*) to Adapt912. The way in which you accomplish this will depend on which of the on-chip memory types your program will reside in (RAM, EEPROM, or Flash). Connect the supplied serial cable between connector J4 on your module and COM1 or COM2 of your PC. (You can use a different COM port, but you will need to select the port in the communication settings menu of your terminal program).

EEPROM. If you wish to preserve DBug12 in Flash, you can put your program in the 768-byte EEPROM, residing at \$0d00; just

make sure to ORG your program there. You can use BOOTLOAD mode or EVB mode to load your program into EEPROM. If you use EVB mode, you can use the DBug12 **bulk** and **load** commands, to erase EEPROM and load your .s19 file into EEPROM, respectively. There are a few things you should keep in mind when using this method. Programming EEPROM bytes requires approximately 10mS each. One way of allowing for this is to slow down the baud rate in DBug12. In EVB mode, enter **baud 300** to change to 300 baud. Then change your terminal program settings to 300 baud and reset Adapt912. Before loading your s-record file into EEPROM in EVB mode, make sure you clear the EE block protect register (EEPROT) first by entering **mm f1 0**, and use the **BULK** command to erase EEPROM if it has been previously programmed. In BOOTLOAD mode, these things are taken care of automatically. NOTE: leave the jumpers of JB1 set to single-chip mode (or remove them altogether, since single-chip is the default)

3.5 The Demo Program

The Adapt912 demo program has both "true vectors" and "pseudo-vectors" included in the source code. To load the demo program into a blank target chip, just assemble it "as-is" and use a BDM pod to load the s-record. To load the demo program into Adapt912 using BOOTLOAD mode (ie. Load it into the lower 30K of Flash, delete the "true vectors" (leave pseudo-vectors intact) and assemble it. Using BOOTLOAD mode, erase the Flash, and Program the s-record as described in section 4.1.

3.6 Adapt912 Expanded Mode

When in expanded mode, the 68HC912B32 uses ports A and B as a multiplexed address/data bus, along with some control lines from PORTE, to access up to 64K of external memory. Chip modes are selected by jumpers on MODA and MODB (Jumper Block JB1). In addition, there are several internal registers which control and define the various possible configurations. For further information on using expanded modes, refer to Motorola's M68HC912B32/D data book.

A memory expansion card (Adapt912MX1) is available from Technological Arts, offering 64Kx16 fast SRAM, and many other optional functions, including battery backup, clock calendar,