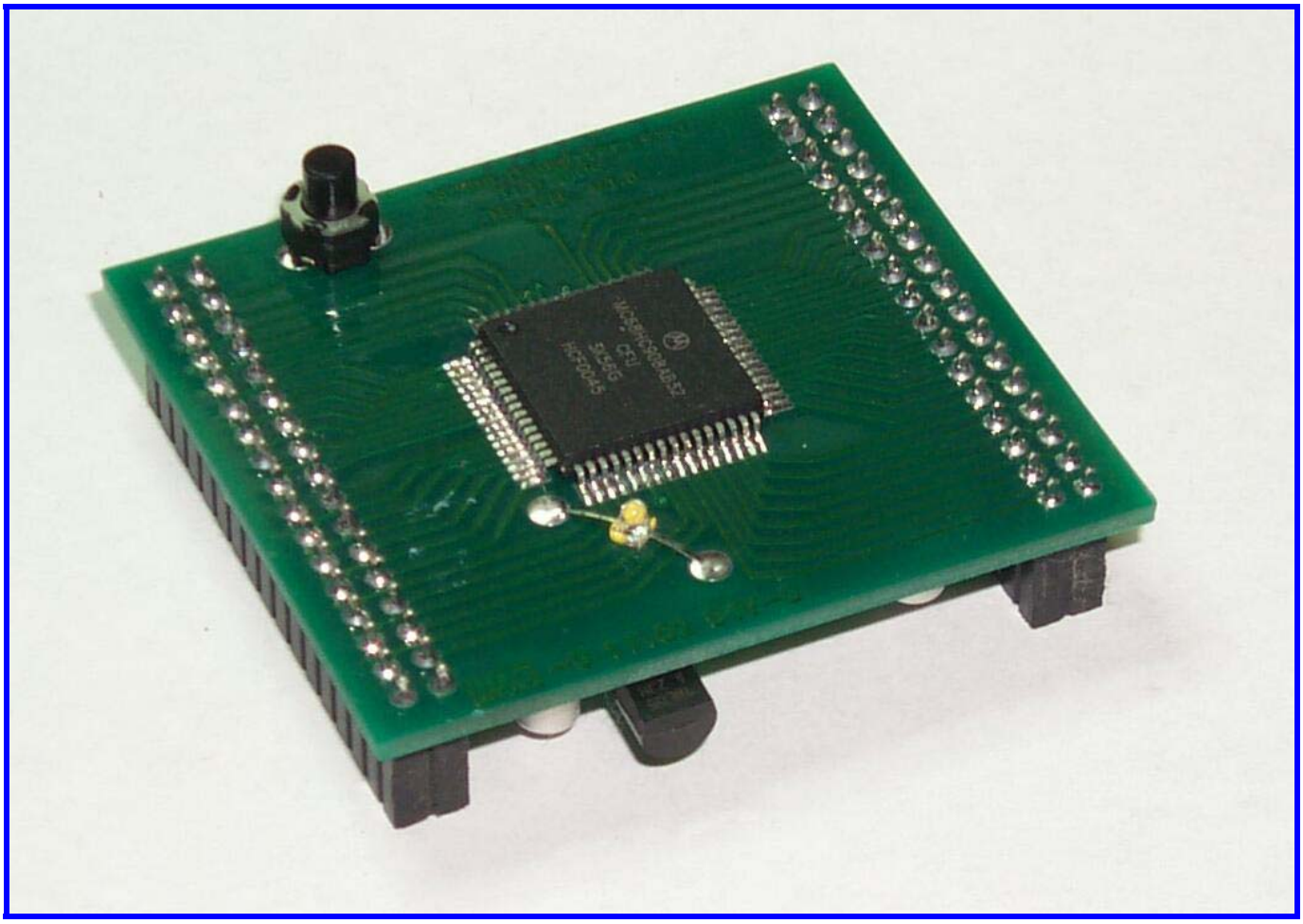
Reference Manual

HC908 Daughtercard



An 8 MHz module of 8-bit computing power with 32 KB of flash memory, 1 KB of RAM, 512 bytes of EEPROM, eight 8-bit A/D converters, a built-in RS-232 serial port for loading new programs, an onboard 5V regulator and lots of I/O to control just about any QRP project you might need on your bench!

CONTENTS

This package contains your fully assembled and tested HC908 Daughtercard. The Reference Manual contains a detailed guide for the pre-loaded software: 'HCmon' (debug monitor) and 'Exerciser' (a sample application). Be sure to check out the Quick Start section to understand the bare-minimum steps necessary to get your daughtercard up and running.

For the latest-available software, documentation and project ideas, be sure to visit the HC908 Daughtercard Resource Page at **www.njqrp.org/hc908/resource.html** .

Various important tools (editor, communications program, 'C' compiler), are available for you to download and use on your computer. The Resource Page contains the links for you to download these tools directly from the vendor sites. It's simple to do and it's all still free.

System Requirements

Almost any computer platform, with most any operating system, will be able to communicate with, and burn new programs into the HC908 Daughtercard. However when it comes to modifying existing programs or creating new ones, the tools available for the HC908 project only operate on Microsoft Windows (95, 98, ME, NT, 2000 and XP.)

Copyrights

Online Website

Be sure to often visit the HC908 Daughtercard Internet website (www.njqrp.org/hc908), as it contains the most current information and documents concerning this project. Documentation corrections and product software updates are also made available in this online manner.

Section 1
Manual Overview

Congratulations on your purchase of the HC908 Daughtercard! Get ready for an exciting and educational homebrewing journey, as there are many projects already available using his board as the computing controller; and there are even more planned.

The biggest and most aggressive 'HC908'-based projects is the Micro908, as chronicled in the pages of QRP Homebrewer and QRP Quarterly magazines during 2001-2002. The Micro908 serves as an ultra-configurable piece of test and measurement equipment that can be extensively used in the ham shack. Antenna Analyzer, Portable PSK, Audio Filter, Memory Keyer are all freely available software programs that run on the Micro908 platform. The Micro908 and all its different application programs leverage the inherent strengths and capabilities offered by its computing engine: the HC908 Daughtercard.

Other smaller-scale and derivative projects are available with the HC908 as the heart. With a simple 're-flashing' of the memory in the HC908 controller from software freely available on the NJQRP website, you can load up your project to serve as a programmable iambic keyer, a digital VFO, a signal analyzer, a 'commander' for remote control of an HF rig, and more.

Although there are many canned programs like these available for the download, you are not limited to using only them. If you have a bit of adventurous software development blood in you, the completely 'open' nature of the software will allow you to easily write your own software programs to instruct the HC908 Daughtercard to perform a custom job in your own project. Using the published library routines, example code and simple building blocks, you can piece together your own software program or modify an existing one. Several powerful (and free) tools are available to enable you to write your custom programs in assembly language or in 'C', and then interactively debug them in an Integrated Development Environment (IDE). Motorola provides tons of application notes and sample projects for the HC908 family of microcontrollers and all of it is available free for the download from their Internet web pages.

So you see, we've created the basic building block (the HC908 Daughtercard), the project environment (the editor, assembler, and debugger), and the resources (the monitor, routine library, sample code) such that there has never been a better time to dive into the blend of 'digital and ham radio'.

SECTION 1 of this manual provides a general overview of the HC908-based projects and a section-by-section description of the overall contents.

SECTION 2 is the Quick Start guidance for those who don't have time to read the full manual before applying power. You'll find this section describing the minimal connections and steps needed to see your HC908 Daughtercard actually do something straight out of the box.

SECTION 3 is the detailed review of the Features and Capabilities of the HC908 Daughtercard. Here you'll read of the exhaustive power available on this small card.

SECTION 4 describes the 'Exerciser' program that comes pre-loaded on the HC908 Daughtercard. This program exercises typical components of an HC908-based project and demonstrates the functional building block routines available to the custom software programmer. 'Exerciser' also serves to functionally test each board before it is shipped.

SECTION 5 describes the runtime monitor program 'HCmon' that is also pre-loaded in each HC908 Daughtercard. It provides low-level command processing for the user to inspect and edit memory, set breakpoints and burn (flash) new applications into the HC908 microcontroller.

SECTION 6 describes how to write your first software application by using the 'Template' program supplied on the website.

SECTION 7 overviews the Test Fixture hardware used to test each daughtercard before it was shipped. It may give you some ideas for your own project.

SECTION 8 contains the schematics and other diagrams.

IMPORTANT … I cannot stress enough the importance of regularly visiting the online HC908 Daughtercard pages at the NJQRP website. This Reference Manual will continue to evolve with additional sections, project overviews and corrections. You should bookmark this location (www.njqrp.org/hc908) and check it often to stay in touch with the availability of related hardware/software components. Many will want to share personal projects and experiences in the hopes of inspiring others, and we'll post these testimonials as well.

SUPPORT … You will undoubtedly have questions along the way. Feel free to contact me by email. Between work, family and hobby, I'll do my best to help out.

Lastly, I hope you have fun building up one of the pre-defined HC908 projects or perhaps even your own custom creation using the HC908 Daughtercard. As I hope you'll soon see, the sky is the limit for uses of this powerful and inexpensive controller module.

73, George Heron, N2APB
n2apb@amsat.org

Section 2
Quick Start

Okay, you've torn open the shipping box, see some goodies inside, and just feel *compelled* to immediately do something – anything – with your new HC908 Daughtercard. Well, this section was made especially for you!

BOX CONTENTS

First of all, this is what is included in the shipping box:

- HC908 Daughtercard – fully assembled and tested.
- Two pinheaders (34 pins: 2 rows x 17 position) – for mating with the J1 and J2 connectors on the daughtercard. A removable shunt (jumper) is supplied on one of the pinheaders.
- Some plastic bubble packing material – give it to your kids or significant other to pop and drive you crazy.

As you see, there's not much to this project in the form of "things to assemble", so let's get into *doing something* with it!

PREPARING FOR OPERATION

The first thing you need to do is supply power and serial port connections to the daughtercard.

Step 1 – Insert Pinheaders to J1 and J2

Insert the two pinheaders into the J1 and J2 connectors of the daughtercard, as shown in Figure 1. These pinheaders will normally be mounted on your project board and used to connect various signals to the respective power and I/O pins on the daughtercard.

Step 2 – Wire Power & Serial Port to Pinheaders

Wire a power connector and a DB9F serial port connector to J1 and J2, as illustrated in Figure 2. (Both the power and serial connectors are user-supplied – it's ultimately your decision how you wish to connect to your daughtercard. Chose connectors that are most convenient on your workbench. I've shown a simple method of using a 9V battery for power, and a standard 'DB9F' connector for interface to a PC over a standard serial patch cable.

Step 3 – Apply the "Monitor Jumper" to J1

As you'll see later in this manual, when you jumper together J1 pins 19 and 20, the 'HCmon' monitor program will be automatically entered upon application of power. Otherwise, with those pins open, the 'user application' will be entered. For now, jumper those pins together. Some homebrewers use a simple 0.1" molded jumper across the pinheader pins, while others actually use a small toggle or slide switch for this, thus making it quicker and easier to switch between startup modes.

Step 4 – Connecting to a Terminal

In this Quick Start exercise you'll be using a terminal program on your PC to talk with the board. Connect the daughtercard's DB9F connector to the serial port of your PC by means of a DB9M-to-DB9F patch cable. (**Note:** This cable is typically available from Radio Shack, Staples, Best Buy, etc., and is a straight-through cable, **not** a 'null modem' cable.)

Step 5 – Setting up a Terminal Program on PC

Launch a terminal program on your PC. You can use the program called 'HyperTerminal' provided in all Windows versions, typically located in Start > Programs > Accessories > Communications. Alternatively you could load the TeraTerm program (available on the Daughtercard 'Tools' web page), which will likely be the terminal program you will use later on with the Daughtercard.

Set up the terminal program for serial port communication at 9600 baud and '8N1'. These configurable settings are specified when you launch HyperTerminal and initially set it up, or located under the Settings > Serial Port menu for TeraTerm.

BLASTOFF!

Alright, you're ready to make some smoke! (Just kidding.)

Connect up your 9V battery or 12V supply and you should see the LED on the board illuminate and the HCmon banner message will be displayed on the PC screen, as shown below:

```
HC908 Monitor, rev 2.0a  Dec 2002,  G.Heron N2APB
HCmon>
```

If you DO NOT see this message … something was improperly wired or configured wrong along the way. Every daughtercard was tested prior to shipping to verify correct programming using this very same setup (and more), so ensure correct power polarity, correct interface pins were used on P1 and P2, the Monitor Jumper is in place and proper settings are being used in your PC terminal program.

If you DO see the HCmon sign-on banner … Congratulations! You're all set to continue happily on in using the daughtercard. Using the HCmon Command Reference Guide (in Section 5), try out some of the commands to inspect/edit memory and registers, set breakpoints, etc. (Be careful not to use the Clear and Load commands until you read about them!) You can also try re-applying power without the Monitor Jumper. This will start up the 'Exerciser' program, which is also pre-loaded into the daughtercard, as explained in Section 4.

HAVE FUN!

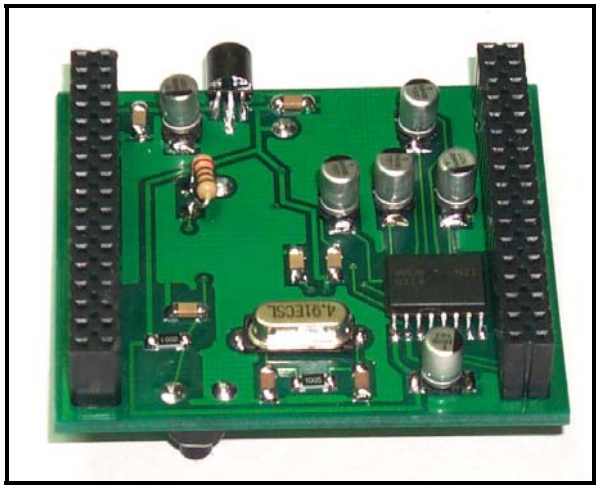


Fig 1: Underside of daughtercard before pinheaders inserted.

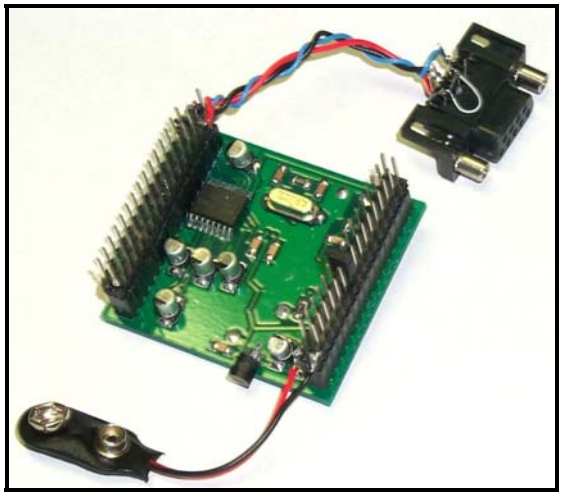


Fig 2: Pinheaders inserted. Power and Serial port wired in.

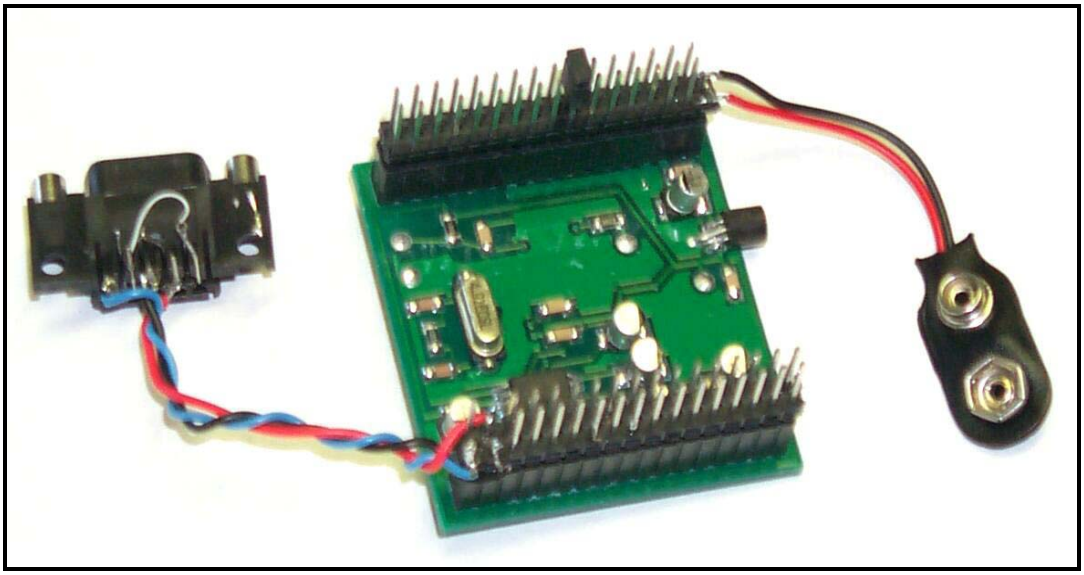


Fig 3: Note the Monitor Jumper located on J1 pins 19 and 20.

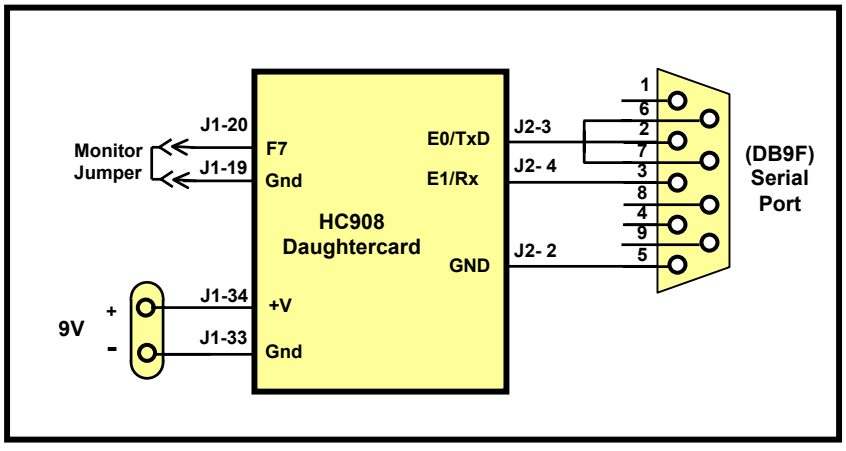


Fig 4: Wiring diagram for the minimal configuration.

Section 3
Features & Capabilities

Introduction

The HC908 Daughtercard is a modular microcontroller board that holds the powerful-yet-inexpensive 68HC908AB32 microcontroller made by Motorola. The daughtercard has lots of memory and I/O, and peripherals like counter/timers, asynchronous serial ports, and A/D converters to make the project useful for many standalone applications around the radio shack. It also contains the clock, reset pushbutton, voltage regulator and RS-232 drivers. The software supplied with the project allows for easy self-programming of the chip – just download new software programs to the chip and it burns the code into its flash memory. No need for special, expensive or complicated programming hardware with this project!

It is highly recommended that you install the software contained on this CD-ROM provided with this HC908 Daughtercard and refer often to the various Motorola documents concerning the 68HC908AB32 MCU. The Technical Data manual provides an absolute wealth of information concerning the capabilities of the microcontroller, its control register make-up and memory space assignment.

The following descriptions in this provide a cursory overview of the official and definitive documentation contained in the Motorola documents.

Architectural Overview

As shown in the HC908 Daughtercard schematic this product is a self-contained microcontroller board with onboard voltage regulation, clock/oscillator components, RS-232C serial port and Reset components. Board connectors are 0.1"-spaced dual-row pin header sockets, allowing the board to be plugged into mating pin headers on the Digital QRP Breadboard base board, or on any other project base board.

Also contained on this daughtercard are the components required for clock generation, a MAX-232 chip for serial communications, a voltage regulator and a RESET pushbutton. All I/O pins of the HC908 are brought out to edge connectors that permit the daughtercard to be plugged into mating pinheaders on a prototype base board containing all other components.

The photo on the cover shows a close-up of the daughthercard. The 64-pin PQFP microcontroller device is soldered in the center of the card's top side and its many I/O pins are connected by top-side traces to two 34-position sockets located on either side of the card. These long sockets will plug into mating pin headers located on the base board, as illustrated.

A small pushbutton is also provided on the top side of the daughtercard. This normally-open momentary contact switch serves as a manual RESET for the system.

An LED is also mounted on the top of the and serves as a "heartbeat" indicator under program control during normal operation of the board. Typically, the user application blinks this light at a regular rate to indicate that the board and software are operating properly. When the board is in "monitor mode" (i.e., when running the HCmon program), the LED is turned on all the time.

The bottom side of the Daughtercard contains the surface mount components required for clock generation – a crystal, two capacitors and a resistor. Also located on the bottom side are the LM78L05 3-terminal voltage regulator and associated filter caps, and the components used for serial interface – the MAX-232 level translator for the RS-232 communications port and the five electrolytic capacitors used for charge pump operation.

Creating a Daughtercard in this manner proved to be a very enabling design path. Besides being able to transfer the card from the prototype base board of a project to its final pcb form, one could easily use a single HC908 Daughtercard for multiple projects. There is lots of capability in this little 2"-square standalone card – just supply 9-12Vdc and a serial comm line from your PC and you can readily download and burn new programs into your HC908 microcontroller.

The Microcontroller

The popular Motorola 68HC908AB32 microcontroller was selected to serve as the heart of this building block module. It was helpful to select such a CISC (complex instruction set controller) instead of a low-end RISC (reduced instruction set controller) like those in the Microchip PIC family. The software designed to control the many peripheral chips and I/O functions would be present great programming challenges in a RISC device because of inherent program memory and register memory addressing restrictions. The 68HC908 is the Motorola equivalent of the popular 8051-class of processors from Intel, SST, and others, offering plentiful I/O capabilities, unrestricted addressing space and high clock speeds.

Another deciding factor in the selection process was the massive amount of I/O pins available for controlling various hardware peripherals typically used in instruments built around the module – LCD displays, DDS chips, pushbuttons, LEDs, 7-segment displays, keyboard, keypad, serial port, et al. Eight separate I/O ports provide up to 51 general purpose input and output pins. Many of these pins are software configurable to serve as analog interfaces,

contain integrated pull-up resistors, and couplings to the interrupt structure of the processor. You'll truly be able to work wonders in interfacing the HC908 Daughtercard to all the devices you will ultimately want to control.

Working in conjunction with the physical I/O pins, the HC908 has some internal macro functions that greatly ease the programmer's job. The microcontroller has built-in modules for asynchronous communications providing an RS-232 serial port, timer modules for frequency counting and timing, programmable interrupt timing for precise interval control, an 8-bit/8-channel A-to-D converter, a keyboard interrupt module, and a watchdog timer. This microcontroller is really quite amazing and is perfect for use on homebrew projects and on the Antenna Analyst AA-908.

Lots of Memory

Plentiful memory is a must for a CISC microcontroller being used in a large application such as typically needed on our ham benches. The 68HC908 has 32 kilobytes of flash memory that will hold the software program itself. There is 1 kilobyte of RAM space available for data variables and other time-changing data. The controller also has 512 bytes of EEPROM (electrically erasable programmable read only memory) that is used to store user-set configuration, calibration and custom string data that is used every time your project is turned on.

From a software perspective, the HC908 Daughtercard supports an enhanced version of the Motorola HC05 programming model. It has 16 addressing modes (direct, indirect, indexed, etc.), a 16-bit index register and stack pointer, and extensive loop controls (e.g., BRCLR n). It supports memory-to-memory data transfers and can perform fast 8x8 bit multiplication and 16/8 division. These last two capabilities will prove quite valuable when it comes time to scale input values and calculate SWR, power, and filter coefficients. Finally, the microcontroller's hardware and software architecture is optimized for controller applications and for C-language support as will be seen with MetroWerks' "Code Warrior" development environment.

HC Monitor Program

The HC Monitor program, or HCmon for short, is a simple, low-level debug monitor developed to support the projects based on the HC908 Daughtercard project. It is pre-programmed in every HC908 Daughtercard provided and resides in the high end of permanent flash memory (address DF00 –through-FFFF).

The operator interfaces to the HCmon by means of a dumb terminal connected to the RS-232 serial port of the daughtercard. Through the monitor's command/response structure, the operator may edit memory and MPU registers, set and reset breakpoints, "go" or single step from any executable location in the loaded program, load S-record files sent by the terminal, program Flash memory from the downloaded S-records, and read input ports and set output ports.

Refer to Section 5 "HCmon Command Reference Guide" for a thorough description of available commands and usage guidance.

Programming / Downloading

One of the prime goals for this project was to be able to easily and inexpensively reprogram the MCU, even after a project built with it was built and deployed to the filed.. The daughtercard has ample onboard flash memory, simplifying the board design and making for a non-volatile project. (That is, the microcontroller retains its program memory even when power is removed.)

But getting the software program into the device is sometimes a concern for a microcontroller homebrew enthusiast due to the expense of the "programmer". Oftentimes it's necessary to purchase a $100-or-more programming board from the manufacturer that will allow you to burn your custom software into the controller's flash memory. In many cases one is able to build his own programming tool (as in the case of the PIC devices), however this is yet another project that must be done before getting to the fun part of experimenting with your intended project.

The good news is that our 68HC908AB32 device has the ability to be in-circuit programmed, which means that a conventional +5V power supply and proper timing is all that's required in order to burn a new program into its flash memory ... even while the daughtercard is in your project! A special boot loader program was developed that allows you to download the binary image of your program over the built-in RS232 serial data port connected to your PC. All you need to do is develop a program with the (free) software development tools on your PC, download it to the HC908 Daughtercard and ***bingo***, you'll be running your new and improved program. In this way you'll be able to take advantage of newer software programs that are provided for download on the HC908 website, or you can develop your own customized versions of the programs and flash the daughtercard directly with your own code. Pretty cool, eh?

The HC908 Daughtercard debug monitor provides the user with an ability to program the permanent, nonvolatile "flash" memory directly from downloaded "S records". An S-record file is a common format for the assembled or compiled 68HC908 MPU binary code, as used in most Motorola-based processors and tools. Thus, when a user wishes to load a program other than what arrived on his HC908 Daughtercard, he may download any of the available S-record files from the project's Internet website and "burn" the program into the MPU's flash memory, thus making the program permanent and available for his use. In this way, the user is able to change the HC908 Daughtercard's "personality" without any other tools than a

terminal program connected to the Daughtercard on its serial port.

Most "dumb terminals" may be used to communicate with the HC908 Debug Monitor. Examples of such programs include HyperTerminal, Red Ryder, ProComm and PCPlus. However, a useful, public domain (freeware) terminal program called "TeraTerm" is also available to run on Microsoft Windows platforms. TeraTerm has a convenient scripting ability that can be invoked to send an S-record file (like a new software program) to your HC908 Daughtercard for flash programming by the Monitor. The TeraTerm terminal program and its S-record transfer script are provided on the HC908 Daughtercard distribution CD-ROM (as well as on the project website) for users to install and use on their systems. TeraTerm also provides a useful "pacing" inter-line delay that is used to give the HCmon enough time to completely program any given flash memory location on the MPU.

***IMPORTANT*:**
This inter-line delay should be set to 40 ms in TeraTerm's serial settings menu.

If you do not see an orderly, line-by-line display of the S-records on the terminal when sending the .s19 file to the HC908 card, the card will not be programmed and a series of "Huh?" messages will likely be seen on the PC screen.

So when it is time for you to get a new software program into your HC908, connect your PC's serial port to the daughtercard, install the Monitor Jumper on the specified pins of the daughrtecard J1 connector and press the RESET pushbutton to reset the MCU and start running the built-in HCmon program. The daughtercard's LED will illuminate, indicating that the monitor software is running.

First type a 'C' command to clear out the user application area of memory. You'll next type the 'L' command to initiate loading of an S-Record file (.S19 extension file) from the PC to the daughtercard. This ASCII-coded object file representing the new software program is transferred from your PC to the daughtercard. using a communications program like HyperTerminal (standard in the Windows Accessory folder), or preferably the TeraTerm program downloadable from the HC908 resource web page.

You will specify the file to be sent to the daughtercard in the terminal program's "Send File" menu, and after selecting the .S19 file you wish to send to the daughtercard, the file is sent line-by-line and echo'd to the PC screen while simultaneously being burned into the proper areas of flash memory.. When complete, control is returned to HCmon and the standard prompt is again displayed.

Once the software is downloaded to the user space in the MCU's memory, you can type the 'X' command (or 'G8040') in HCmon to go start executing at the start of your program, which usually starts at hex address 8040, or set breakpoints with which you can debug the program, etc.

For a thorough description of using these programming commands, please refer to the various portions of the HCmon Command Reference Guide in Section 5 of this manual.

Section 4
'Exerciser' – A Pre-Loaded Application

Introduction

The 'Exerciser' software program is located at the "user application" address of 8000 and comes pre-loaded on all initially-supplied HC908 Daughtercards. (The other pre-loaded software program is the 'HCmon' debug monitor loaded from location E000 to FFFF.)

The Exerciser user application is controlled by an external terminal program connected to the daughtercard's serial port and serves at least two main roles. First, it provides the user with a ready-made program that exercises (or demonstrates) some standard peripheral functions that can be used in a typical project employing the HC908 Daughtercard. Of course the various peripheral hardware components must be present and connected to the daughtercard for the program to work as described, and these peripherals include: an LCD display, a shaft encoder, a piezo-electric sounding device, a number of discrete LEDs and a serial port connector.

The second role performed by the Exerciser program is that of a daughtercard tester. Each HC908 Daughtercard made is placed into the "test fixture" consisting of the hardware peripherals mentioned above. (Refer to the Test Fixture schematic). A special connector is wired to specific I/O pins on the J1 and J2 connectors of the daughtercard, and the HCmon and Exerciser programs are burned into flash memory on the 68HC908AB32 microcontroller. This initial "bootstrap programming" of the debug monitor program (at least) is necessary to make the daughtercard intelligent enough to handle subsequent repeated loadings/burnings of custom programs into its user application memory space.

The Test Fixture schematic and photographs are shown in this section to illustrate ways that an HC908 Daughtercard may be configured/built into a system to serve as the heart of a custom project. For example, the Test Fixture as shown may be directly used for project that serves as a "commander" remote controller to an HF rig, delivering ASCII commands across its serial port to the radio. Another possible use of the Test Fixture hardware would be as digital VFO, just by simply wiring up a 3-wire connection of a DDS Daughtercard (see www.njqrp.org/dds).

So you see, this "typical hardware configuration" can serve as a launch pad for many different projects, all distinguished by different software programs being burned into the MCU's flash memory on the daughtercard.

Exerciser Commands

The Exerciser software program is located in the User Application memory space from 8000 to DFFF. There are two ways to run any user application program (i.e., Exerciser, or any program you Load into the MCU yourself). The first way is to merely reset the Daughtercard with the Monitor Jumper removed, which forces the HC908 MCU to start executing at location 8040. The second way is to Go start executing at address 8040 from the HCmon debug monitor by typing 'G 8040'. The third way is type type the command 'X'.

No matter how you start running the Exerciser program, the initial sign-on message shown below is sent to the terminal over the daughtercard's serial port:

```
~~ Digital Breadboard Exerciser v3.0b G. Heron N2APB ~~

Excerciser (D,S,L,P,V,Y,W,R,T,A,M,H) >
```

The second line prompts the user to enter one of 12 commands that may be executed by the Exerciser software to demonstrate proper connectrions and board operation.

The 'H' command is locally the best one to first describe, as it lists the menu of command represented by the single-letter commands.

```
Help Menu
=========
D = DDS Set
S = Shaft Enc
L = LED Ports
P = Piezo Tones
Y = Display Test
W = Write EEPROM
R = Read EEPROM
T = Timing Test
A = ADC Test
M = PWM Test
 H = Help
 X = eXecute user program
```

Each of the commands will be described in the following sections.

D = DDS Set

The 'D' command instructs the MCU to programs an AD9850 DDS integrated circuit to generate a specific frequency of 10.000000 MHz. The DDS serial control lines are expected to be wired to port F bits 0, 1 and 2. An appropriate string is sent to the serial terminal stating that this action has happened, and the Exerciser prompt is again displayed.

S = Shaft Enc

The 'S' command instructs the MCU to read the shaft encoder bits and increment or decrement a 2-digit counter displayed to the LCD. Whether the count increases or decreases is determined by the direction in which the user turns the encoder. The Piezo is also sounded with a frequency in relation to the numeric value of the shaft encoder display. This command is terminated by pressing any key on the terminal keyboard, or by pressing the pushbutton of the shaft encorder, whereupon the Exerciser prompt is again displayed on the serial terminal.

L = LED Ports

The 'L' command instructs the MCU to cycle through a sequencial ON-OFF illumination of each of the LEDs connected to available I/O pins of the MCU. This action make it clear to the user that all lines from the MCU are properly connected and continuous through the J1+J2 connectors. At the end of the sequence, the Exerciser prompt is again displayed on the serial terminal.

P = Piezo Tones

The 'P' command instructs the MCU to generate a square wave signal of approximately 4 KHz which causes the piezo device to emit a tone. The tone is beeped (turned on and off) to further demonstrate its functional state. The beeping is discontinued when the user presses any key on the terminal keyboard or when the shaft encoder pushbutton is pressed, and the Exerciser prompt is then displayed again.

Y = Display Test

The 'Y' command instructs the MCU to send a known set of characters to both lines of the LCD.

Line 1: `ABCDefghIJKLmnop`

Line 2: `QRSTuv0123456789`

After the user has verified the displayed data, pressing any key on the terminal keyboard will again display the Exerciser prompt.

W = Write EEPROM

The 'W' command instructs the MCU to place some user-specified data into EEPROM (nonvolatile) memory. The user is prompted for a short text string, and when the <ENTER> key is pressed the entered data is written to EEPROM memory and the Exerciser prompt is again displayed on the serial terminal.

R = Read EEPROM

The 'R' command instructs the MCU to read a portion of the EEPROM (nonvolatile) memory and display it to the serial terminal. The Exerciser prompt is then displayed again on the terminal.

T = Timing Test

The 'T' command instructs the MCU to enter into an infinite loop of precisely-known cycle instructions that continuously toggle the heartbeat LED control bit (port F bit 6). The purpose of this command is to allow the user to measure the toggling frequency of that bit and thus compute the MCU bus speed. The only way to exit this function is to reset the HC908 Daughtercard.

A = ADC Test

The 'A' command instructs the MCU to read the analog value presented on the analog-to-digital converter input port B bit 0. This 0-5V analog signal is presented to the ADC input bit by means of a potentiometer. Thus, the raw 8-bit value of that reading, corresponding to the percentage of rotation of the potentiometer, is repeatedly displayed to the serial terminal. A piezo tone is also delivered, with its frequency determined by the position of the potentiometer (i.e., analog voltage being send to the MCU chip) until a key is pressed by the user. The test also stops when the shaft encoder pushbutton is pressed. At that time the Exerciser prompt is again displayed on the serial terminal.

H = Help

The 'H' command instructs the MCU to display the list of available command definitions to the serial terminal, after which the Exerciser prompt is display again.

Summary of Exerciser Uses

With this small set of commands, corresponding to a similar group of software subroutines performing the various functions, the Exerciser program is able to:

1) Test the HC908 Daughtercards before shipping;
2) Demonstrate operation of basic peripheral "primitives" for the user to better understand software programming and control techniques; and
3) Provide a standard "framework" of program construct (interrupt vectors, RAM memory definition and locations, etc.) such that the user can easily modify the Exerciser program to create a customer application.

As with each of the files provided on the distribution CD-ROM, the user is encouraged to study the source code files (*.asm) to better understand the programming environment for the HC908 Daughtercard.

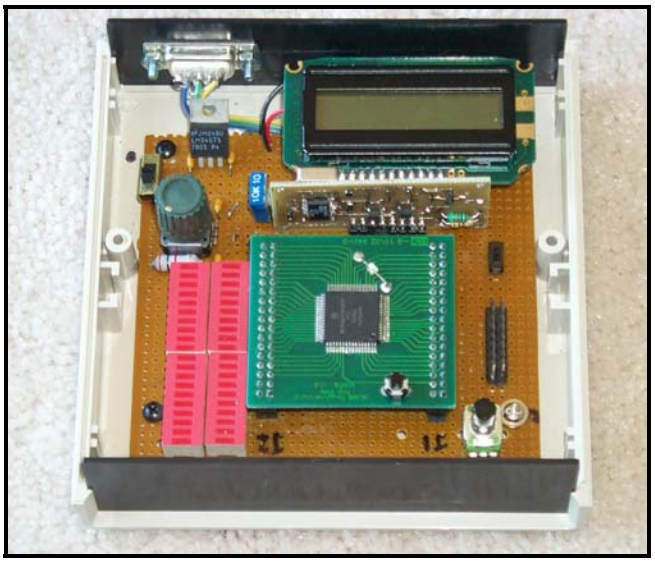
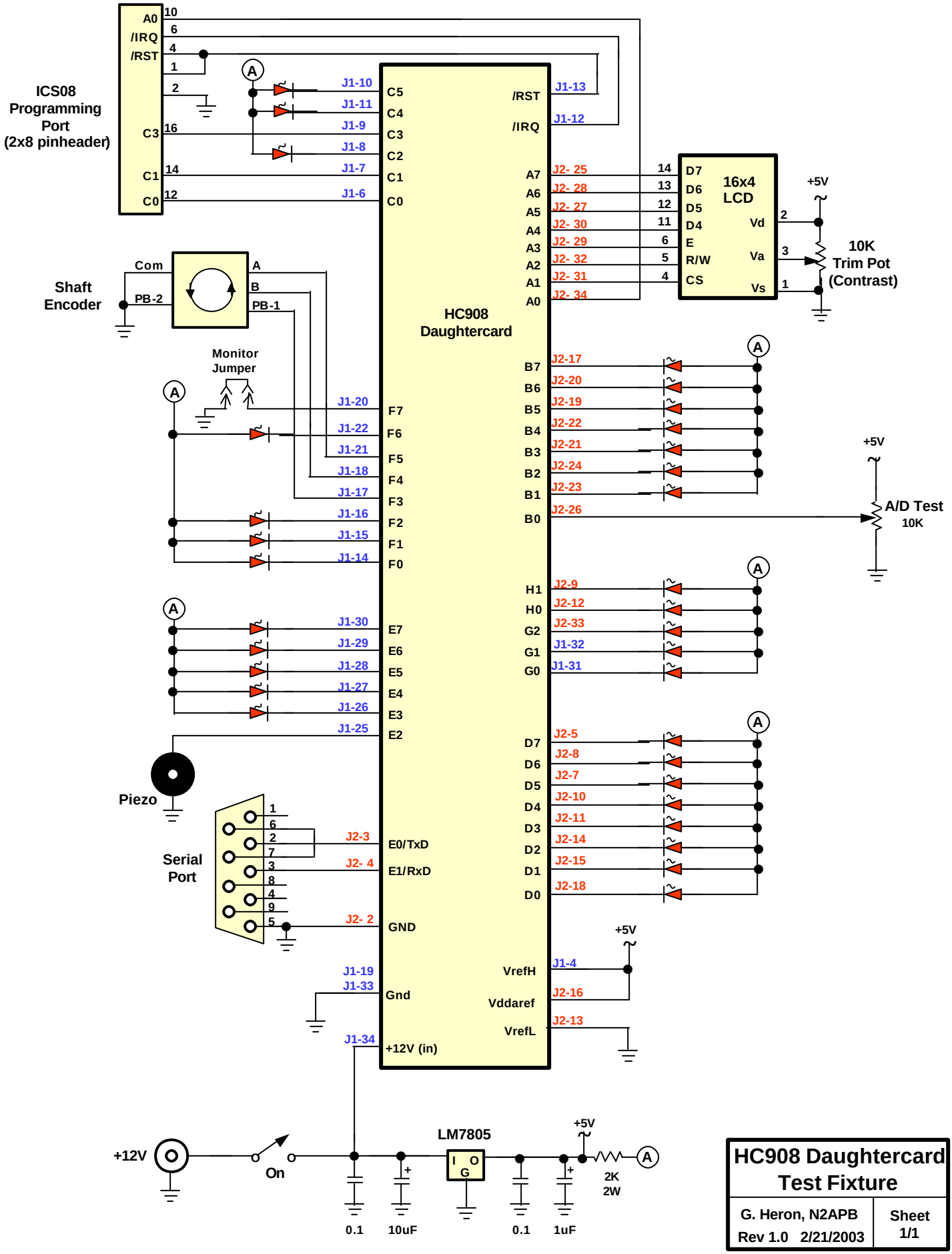


Photo of the HC908 Daughtercard Test Fixture. Housed in a standard Radio Shack plastic enclosure, the daughtercard is mounted roughly in the middle, with the DDS Daughtercard directly above it, and the LCD is above the DDS card. The initial boot programming connector is located to the right of the HC908 card, and the Monitor Jumper (actually a small slide switch) is just above that connector. The potentiometer is mounted below the boot programming connector. Forty LEDs are provided in the form of 4 bargraph LEDs mounted to the left of the HC908 card. The shaft encoder is mnounted above the LEDs and the power switch is located abovethe encoder. This package is sized to be quite convenient and I've actually taken it on many business trips throughout 2002, developing various software applications while on airplanes and in hotel rooms. (Airport security personnel certainly did not have a sense of humor with respect to my choice of carry-on apparatus.)



HC908 Daughtercard Test Fixture	
G. Heron, N2APB Rev 1.0 2/21/2003	Sheet 1/1

Section 5
HCmon Command Reference Guide

Introduction
The HCmon debug monitor program is pre-loaded in every HC908 Daughtercard provided. Access to HCmon is made by installing the Monitor Jumper across J1 pins 19 and 20 and resetting the board. The HCmon sign-on message will then appear on a terminal connected to the (9600 baud N81) serial port of the daughtercard.

Features
Small size -- Full blown debugger fits in under 3.5K, which allow to reside at F000. It uses 64 bytes of RAM, from 0x400 to 0x43f.

Disassembler -- Full disassembler, complete with address operand decoding

Breakpoint - Uses on chip break register to insert a breakpoint anywhere in flash or RAM, without having to write over flash. So it will work if your flash is protected. Special feature allow you to easily resume from breakpoint without removing the breakpoint.

Single Step -- Allows single step execution of a program. Every step shows the disassembled instruction and shows all the registers.

Downloader – S-record downloader, communicates at 9600 baud for fast programming.

Register View -- Allows viewing all user program registers.

Dump Memory -- Hex dump command allows easy viewing of memory

Memory Edit -- Easily change memory. Sequences forward and backward.

Memory Fill -- Allows filling memory a particular value. Begin and end addresses are specified.

Flash Clear -- One command will clear all flash, allowing for a new download.

User Interface
All commands are one character. Case is ignored. All commands end with and <ENTER> (carriage return, hex 0d).

Fields are separated by a space. The first field is shown separated from the command by a space, although it's not needed. (i.e., "g8000<ENTER>" works, as well as "g 8000<ENTER>"). Fields are variable length, meaning a 16 bit address can be 1,2,3 or 4 digits. For example "D 0", "D 10", "D 100" and "D 8000" are all valid commands. All fields are hexadecimal, both the fields displayed and the fields entered. The Fill command is the only command with more than one field, and it requires that the fields be separated by spaces. For example, "F 6B 1D0 FC" fills from memory location 0x6B to location 0x1D0 with 0xFC.

Commands that can provide another page or line of information allow for the SPACE character to go to the next display. Pressing <ENTER> will end the command. So when you do a dump command (i.e., D 8000<ENTER>), a SPACE will show the next page and <ENTER> will go back to the prompt. When doing single stepping (N, Next command), a space will show the next page and <ENTER> will go back to the prompt.

Terminal Interface requirements
To talk to HCmon on the HC908 Daughtercard you need a terminal emulator and the ability send an S-record file one line at a time, with a 5 millisecond pause between each line. Many terminal emulators have the ability to send a file with line delays. The terminal emulator I have been using is TeraTerm, which is freeware and can be downloaded from many sources. HCmon is default setup is for 9600 baud, 8 data bits, one stop bit and no parity.

Interrupt Vectors
The HC Monitor vectors all interrupts, except for SWI, to a jump table at the start of user Flash ($8000). The user application must provide a table of jump statements for all interrupts to its own interrupt processing routines, except for SWI, which is used by the monitor for breakpoints.

Example:

```
   ;
   ;  Interrupt Vectors
   ;
   org $8000
   ;
jmp _user_adc      ; ADC vector code
jmp _user_keyboard ; Keyboard vector code
jmp _user_scitx    ; SCI transmit vector code
jmp _user_scirx    ; SCI receive vector code
jmp _user_scierr   ; SCI error vector code
jmp _user_timbch3  ; TIMB Channel 3 Vector code
jmp _user_timbch2  ; TIMB Channel 2 Vector code
jmp _user_spitx    ; SPI transmit vector code
jmp _user_spirx    ; SPI receive vector code
jmp _user_timb_of  ; TIMB Overflow Vector code
jmp _user_timb_ch1 ; TIMB Channel 1 Vector code
jmp _user_timb_ch0 ; TIMB Channel 0 Vector code
jmp _user_tima_of  ; TIMA Overflow Vector code
jmp _user_tima_ch3 ; TIMA Channel 3 Vector code
jmp _user_tima_ch2 ; TIMA Channel 2 Vector code
jmp _user_tima_ch1 ; TIMA Channel 1 Vector code
jmp _user_tima_ch0 ; TIMA Channel 0 Vector code
jmp _user_tim_of   ; TIM Overflow Vector code
jmp _user_pll      ; PLL Vector code
jmp _user_irq      ; IRQ1 Vector code
```


Reset Vector

The HC Monitor vectors the RESET vector to are area of code that reads the Monitor Jumper. If this jumper is in place (i.., if port F7 is grounded, HCmon is entered directly. If, however, the jumper is not in place, the program jumps to the start of User Application space (at hex address 8040) and begins executing the user program. The monitor can thereafter be entered from the user application with a SWI instruction.

The Files Included

```
/code/hcmon/license.txt - License for the software
/code/hcmon/hcmon.asm   - Main source file
/code/utils.s           - Utilities source file
/code/hcmon/disasm08.s  - Source for disassembler
/code/local.inc         - Defines for constants options
/code/abregs.inc        - Processor specific defines
/code/macros.inc        - Global macros
/code/hcmon/pony.lst    - List files
/code/hcmon/hcmon.s19   - S record download file
/code/hcmon/dnld_test.s19  - S record test file
```

Commands

H -- Help

This command displays the total list of commands available:

```
HCmon Commands:

A ADDR Disassemble
B ADDR Set Break
U Undo Break

D ADDR Dump
E ADDR Edit
F BEGIN END VALUE Fill
R Print Regs

G ADDR Go
N Next
L Load Srec
C Clear Flash
X Execute User App at $803C
```

A ADDR -- Disassemble

This command disassembles memory starting at specified address. Display will show 20 lines. Pressing SPACE will display another 20 lines, or pressing <ENTER> will return to the prompt. The display shows, from left to right, the address, the instruction bytes, the operator, and operators (if any). Branch operators show the calculated addresses.

```
HCmon> a e000<ENTER>
E000 6E 03 1E       mov   #03,1E
E003 6E 01 1F       mov   #01,1F
E006 9B             sei
E007 45 02 40       ldhx  #0240
E00A 94             txs
E00B 45 E3 D8       ldhx  #E3D8
E00E CD E2 F3       jsr   E2F3
E011 6E 00 19       mov   #00,19
E014 6E 40 13       mov   #40,13
E017 6E 0C 14       mov   #0C,14
E01A 45 00 40       ldhx  #0040
E01D 7F             clr   ,x
E01E AF 01          aix   #01
E020 65 02 40       cphx  #0240
E023 26 F8          bne   E01D
E025 CD E0 5B       jsr   E05B
E028 C7 FF FF       sta   FFFF
E02B A6 FF          lda   #FF
E02D CD E3 91       jsr   E391
E030 45 E4 A2       ldhx  #E4A2 <ENTER>
HCmon>
```

B ADDR -- Set breakpoint

The break command sets the address where execution will stop and return back to the monitor, if that address is reached. The break works for any access, read or write.

```
HCmon> b 8027<ENTER>
Break: 8027
HCmon>
```

U -- Undo Break

This command erases the break setting.

```
HCmon> u
HCmon>
```

D ADDR – Dump Memory

The hex dump command shows memory starting at the address entered. Two notes:

- The command display will always start with the least significant hex digit of zero
- The display will pause at Hex --00 boundaries. Just press SPACE for the next page, or <ENTER> to return to the prompt.

```
HCmon> d e200

ADDR  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
---- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
E200 A1 FC 26 07 AD 24 F6 EE 01 87 8A CC E1 8A C6 02
E210 04 27 06 A1 01 26 09 20 05 C6 02 05 27 02 98 81
E220 99 81 AD 0F E6 03 EE 04 20 0F AD 07 EE 02 C6 02
E230 08 20 06 C6 02 04 CE 02 05 87 8A 81 8B 86 C7 02
E240 04 CF 02 05 81 CD E7 4F 25 10 8B 86 CF 02 02 45
E250 02 00 94 87 8A CE 02 02 9A FC AD B2 25 C3 C6 FE
E260 0B 2A 1F CD E2 22 C3 FE 0A 26 17 8B 86 C1 FE 09
E270 26 10 A6 01 C7 02 09 8B 86 C7 02 06 CF 02 07 CD
E280 E1 F8 AD AF 94 C6 02 08 87 8A 80 45 E4 A2 CD E6
E290 E2 45 E4 D8 CD E6 E2 81 45 00 3B D6 E3 60 D7 01
E2A0 99 5B F8 CD 01 9A CD E0 5B 81 45 00 47 D6 E3 90
E2B0 D7 01 99 5B F8 45 E4 7C CD E6 E2 C7 FF FF 95 35
E2C0 41 A7 DC CD E3 09 26 20 86 A1 39 27 0B A1 31 26
E2D0 17 CD 01 A5 A7 23 20 E3 A7 23 0B 16 C9 CD E6 B8
E2E0 0B 16 C3 CD E6 B8 20 BE A7 24 45 E4 8E CD E6 E2
E2F0 CC E0 28 19 36 1B 36 7E 36 7E 3A 7E 38 7E 39 1E
```

E ADDR – Edit RAM and Registers

The edit command allows you to change RAM and Registers (addresses 0-$23F, and $FDFF-$FE0C). THIS WILL NOT WORK ON FLASH!!!

There are four things you can enter once the memory location is displayed:

- nn<ENTER>, where nn is a Hex digit, will write nn to the memory location.
- <SPACE>, Will skip to the next location, without altering the present location.
- <BACKSPACE>, Will skip to the previous location, without altering the present location.
- <ENTER>, Will exit the edit function, with altering the present location.

```
HCmon> e 67<ENTER>
 0067 00 12<ENTER>
 0068 00 34<ENTER>
 0069 00 56<ENTER>
 006A 00 <BACKSPACE>
 0069 56 <BACKSPACE>
 0068 34 <BACKSPACE>
```

```
 0067 12 <SPACE>
 0068 34 <SPACE>
 0069 56 <ENTER>
HCmon>
```

F BEGIN END VALUE -- Fill memory

This command will write the entered Hex value to the specified range of addresses. Below shows writing the value $77 from address $00C8 to $00E4. The dump command is used to verify the operation.

```
HCmon> f c8 e4 77
HCmon> d b0

ADDR  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
---- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 00 00 00 00 00 77 77 77 77 77 77 77 77
00D0 77 77 77 77 77 77 77 77 77 77 77 77 77 77 77 77
00E0 77 77 77 77 77 00 00 00 00 00 00 00 00 00 00 00
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
HCmon>
```

R -- Print Regs

This command displays the registers on the user stack. The stack is stored in the variable _stack_save.

```
HCmon> r
SP=FFFF CC=01 A=00 H:X=00:03 -- PC=0000
HCmon>
```

G ADDR -- Go

The Go command tells the processor to start executing at the specified address. Or is no address is entered, execution begins at the address currently pushed on the user stack. If a download is done, and a reset jump table instruction is burned into the proper address, then the stack will be initialized to execute the user application after a download or power-up without specifying a starting address.

If a breakpoint exists at the go address, which will be the case if you have just reached a breakpoint, the go instruction will automatically do the following:

1. Set a breakpoint to the next instruction,
2. Execute at the current or specified instruction,
3. When break occurs at next instruction, it will re-install the breakpoint at the previous spot,
4. Then continue execution.

The above steps will all be transparent, looking just like it executed from the breakpoint spot.

```
HCmon> g 8000<ENTER>
```

N -- Next

The Next instruction single steps from the current program counter. Pressing <SPACE> executes the next instruction, and pressing <ENTER> returns to the prompt. Note that while the Next command uses the break register, the break register will be restored to the last value of the break command after command is done.

```
HCmon> n
Break: SP=01FA CC=63 A=00 H:X=E1:AF -- PC=800D 45 80 A0
                                    ldhx  #80A0
<SPACE>
Break: SP=01FA CC=65 A=00 H:X=E1:A0 -- PC=8010 CD 80 7D
                                    jsr   807D
<SPACE>
Break: SP=01F8 CC=65 A=00 H:X=E1:A0 -- PC=807D F6
                                    lda   ,x
<SPACE>
Break: SP=01F8 CC=65 A=9B H:X=E1:A0 -- PC=807E 27 06
                                    beq   8086
<SPACE>
Break: SP=01F8 CC=65 A=9B H:X=E1:A0 -- PC=8080 AD F3
                                    bsr   8075
<SPACE>
Break: SP=01F6 CC=65 A=9B H:X=E1:A0 -- PC=8075 AD E8
                                    bsr   805F
<SPACE>
Break: SP=01F4 CC=65 A=9B H:X=E1:A0 -- PC=805F 89
                                    pshx
<SPACE>
Break: SP=01F3 CC=65 A=9B H:X=E1:A0 -- PC=8060 8B
                                    pshh
<ENTER>
HCmon>
```

L -- Load S Record

The L command accepts S records from the host. S records should be sent with 5 millisecond delays between each line to allow time for the data to burn into flash. Also, the S record file should be produced to have a maximum of 16 data bytes (represented as 32 ASCII characters) per line. Example of an S record with maximum length below:

```
          S113F19007CFFE0A8B86C7FE09A680C7FE0BC7FEF3
                |------------------------------|
                          16 data bytes
```

A popular way to download files from a Windows computer is to use freeware TeraTerm application. You can set the line delay in the "Setup-->Serial" menu as "5" msec per line. Then use the "File-->Send File" menu to send the file. You can even program a macro to do this.

HCmon will return to prompt after receiving the last record, which is a line starting with the letters "S9".

```
HCmon> l
  - waiting ...
S1138000450200949A1C06458087CD807D4580A05A
S1138010CD807DCD8046C6004427F2A4DFA1412651
S1138020084580A4CD807D20E4A14226084580B087
S1138030CD807D20D820D64F0B1602B61881AD1FF7
S11380400B16FBB61881450044CD803EF7A10D26E2
S1138050037F200ACD8075AF0165006225EB81891D
S11380608B5542AFFF354226090802041802200 24C
S113807019028A8881ADE80F16FBB71881F6270626
S1138080ADF3AF0120F7810D0A4C3320537973749B
S1138090656D732048433038205465737 40D0A00AD
S11380A00D0A3A000D0A50756C73696E672E2E0026
S10C80B00D0A44756D6D79008020
S113DFCDCC80B8CC80B8CC80B8CC80B8CC80B8CC60
S113DFDD80B8CC80B8CC80B8CC80B8CC80B8CC809C
S113DFEDB8CC80B8CC80B8CC80B8CC80B8CC80B854
S106DFFDCC8000D1
S9030000FC
HCmon>
```

C -- Clear Flash

This command erases the user flash RAM space from $8000 to $DFFF.

```
HCmon> c
HCmon>
```

Tera Term Download Macro

If you use TeraTerm, here is a macro to clear flash and download a file:

```
;
;HC908_macro.ttl
;
; Macro for Tera Term
;  to download B52 Keyboard Controller S records
Fname = 'c:\dnload\srecords.s19'

:clrmem

; sendln 'c'
; timeout = 5
; waitrecv HCmon>' 6 1
; if result=1 goto sndfile

sendln 'c'
timeout = 5
waitrecv 'HCmon>' 6 1
if result=0 goto exiterr

:sndfile
sendln 'l'
sendfile Fname 0
end

:exiterr
TxtMsg = 'Download Failed'
statusbox TxtMsg 'Download Macro'
pause 3
end
```

I map an F8 key in the TeraTerm config file, editing keyboard.cnf and creating a new config file hcmon.cnf in the Tera Term directory adding the lines below at the end. You can load this special configuration when you run TeraTerm by adding the command line option "/K=hcmon.cnf".

Comment out the normal F8 Key:

Change:

```
;F8 key
F8=66
```

to:

```
;F8 key  Used for macro
; F8=66
```

Put the following 2 lines after [User keys]

```
[User keys]
;F8 key: Executing a macro file to download S records
User1=66,2,lh485.ttl
```

Test Application

The S records shown below are for a test application that runs at address 8000 hex, which just outputs a string, prompts for a key and returns to the monitor. You can use this to quickly and easily test the download capabilities of your HC908 Daughtercard. (This file is provided in the HC908 Program Library as **dnld_test.s19**.)

```
S113800045020094458025CD801BAD0227FC4F0B13
S11380101602B618810F16FDB71881F62706ADF5BE
S1138020AF0120F7810D0A506F6E79205465737487
S1138030204170706C69636174696F6E2C2070727A
S11380406573732061 6E79206B657920746F206588
S109805078697 40D0A00BA
S9030000FC
```

Credits for HCmon

Major portions of the HC Monitor program were adapted from the Pony 68HC908 Debug Monitor V1.4 by Larry Bateman. Copyright (c) 2000, L3 Systems.

Portions were also borrowed from Motorola's "MC68HC908GP32 User Bootloader", Copyright (c) Motorola 2000. Written by DGJ Klotz Rev ES 1.0 26-Feb-00

Pony HC08 Monitor 3/27/02, L3 Systems, Inc.
Copyright L3 Systems, Inc. 2000-2002 PO Box 2954
Author: Larry M. Bateman. Redmond, WA 98073
email: info@L3sys.com

Section 6
'Hello DDS' – Writing Your First Software Application

Introduction

The first program one usually creates on a new platform is 'Hello World!' This simple program merely displays a message to the console indicating that it is alive and that the assembly and download process works. As readers know, the HC908 Daughtercard already comes with a pre-loaded 'monitor' program (HCmon) that is controlled by a serial terminal, as well as a pre-loaded 'exerciser' program for the various I/O devices you might have hooked up. What we'll do next is create a new program called 'Hello DDS' to show how easy and straightforward it is to use the built-in subroutines provided in the Exerciser to enable you to produce your own first program.

Hello DDS is a simple terminal-driven program that prompts the user for the frequencies desired to be generated by the DDS Daughtercard connected to the HC908. As can be seen in Figure 1, the system is very easy to assemble. The "minimal breadboard" consists of the HC908 Daughtercard connected to an RS-232 serial terminal. I've used my PDA (a Palm Tungsten W) as the terminal, although you could use any terminal that has a serial port – e.g., a PC running HyperTerm, a Poqet Plus notebook computer running ProComm, or anything in between. Using a PDA is a convenience that allows me to more easily use projects when hamming out in the field or when on business trips.

The DDS Daughtercard is connected to the HC908 card using only three wires for the serial loading protocol and two wires for power and ground. (See the notes section at the end for details on this card.) A 9V battery powers both cards and the PDA terminal plugs into the HC908's built-in serial port. The schematic of this configuration is merely a simple subset of those published in previous issues of the column, and on the project's website, so it will not be repeated here.

When powered on or when reset, the HC908 card sends menu text to the PDA prompting the user for a command. As shown in Figure 2, he has a choice to:

1) Enter a freq – sends the specified frequency to the DDS chip
2) Set start freq – specifies the starting frequency for a sweep
3) Set end freq – specifies the ending frequency for a sweep
4) Set step size – specifies the step size to be used during the sweep
5) Sweep DDS – commands the sweep to begin

Creating 'Hello DDS'

Now that you see what the end goal is, let's go through the process of actually creating the HC908 program that does this.

Experienced software developer knows that the best way to create a new program is to start by modifying an existing one. We'll use the Exerciser program as the starting point for nearly all our programs on the Digital Breadboard project, as it provides a convenient template for program format, interrupt processing, timing and access to commonly-used subroutines. To emphasize the value of this approach, I've provided a 'stripped-down' version of the Exerciser program with the HC908 Daughtercard product itself, as well as on the Digital Breadboard website. (Recall that absolutely all software and hardware used in the Breadboard project is freely available for anyone's non-commercial use.) This Template.asm file is where all our edits will be done.

With no modifications, the Template program merely provides the interrupt hooks and common subroutines, as well as the interrupt structure for the heartbeat LED. This blinking LED on the HC908 card is a good indicator that your program is running satisfactorily (e.g., that it isn't hung up in an endless loop someplace.) If the Template program is downloaded to the HC908 card and run as-is, all you'd see is the blinking heartbeat LED with no other activity happening on the console, LCD, shaft encoder, DDS or anything. Try it and see!

This is the main area of interest for us in the code:

```
User_Main:
        ldhx    #TemplateBanner_msg
        jsr     _puts                   ;print the Banner message
Main_Loop:
        sta     copctl                  ;clear the COP counter
        bra     Main_Loop

TemplateBanner_msg:  fcb 'Template Program',CR,LF.0
```

What we'll do is place the new/modified program below in its entirety with line numbers that we can reference to help explain the program operation.

```
1   User_Main:
2           ldhx    #DDSmenu_msg
3           jsr     _puts               ;print the Banner message
4   Main_Loop:
5           sta     copctl              ;clear the COP counter
6           ldhx    #Command_msg        ;point to the 'Command' string
7           jsr     _puts               ;and print it to the terminal
8           jsr     _gets               ;get an input command
9           lda     _inbuf              ;get the entered character
10          cmp     #'1'                ;compare the input to '1'
11          beq     Set_Freq            ;if it was a '1', branch to
12                                      ; the routine to set freq
13          bra     Main_Loop           ;go get another command
14
15  Set_Freq:
16          ldhx    #EnterFreq_msg      ;point to the 'Enter Freq' string
17          jsr     _puts               ;and print it to the terminal
18          jsr     _gets               ;get a sting of numbers
19          jsr     Set_DDS             ;set the DDS output to that freq
20          bra     Main_Loop           ;and go get next command
21
22  DDSmenu_msg:
23          fcb     'HELLO DDS v1.01',CR,LF
24          fcb     '1) Enter freq',CR,LF
25          fcb     '2) Set start freq',CR.LF
26          fcb     '3) Set end freq',CR,LF
27          fcb     '4) Set step size',CR,LF
28          fcb     '5) Sweep DDS',CR,LF,0
29
30  Command_msg:
31          fcb     CR,LF,'Command: ',0
32
33  EnterFreq_msg:
34          fcb     CR,LF,'Enter Freq: ',0
```

The first thing we'll do is change the pointer to display a new set of text that represents the menu. Line 2 points to the new string to be output DDSmenu_msg at line 22. The call to the _puts (put string to console) routine then sends all the characters of those six 'fcb' lines to the terminal and only stops when it sees the '0' terminating character.

The program continues running at label Main_Loop (line 4), and accesses the copctl (Computer Operating Properly) control address, which resets the watchdog timer and prevents the program from automatically locking up.

Next, we added a couple of lines (6-7) to print the prompt 'Command: ', and added line 8 (jsr _gets) to get a user input string. (We won't do advanced things like range checking for valid entry.) The user input string is terminated when the <ENTER> is pressed.

Next we check the command character entered (located in '_inbuf') against the valid commands and branch to the appropriate desired routine when a match is found. (We'll only show one such command check. The full program listing at the end will show all command checks.) Lines 10-11 compare the input command to a '1', and if it was found to be a match, the Set_Freq routine is called. Otherwise, if no match was found, we instruct the computer to go back to the Main_Loop label and prompt for a command again.

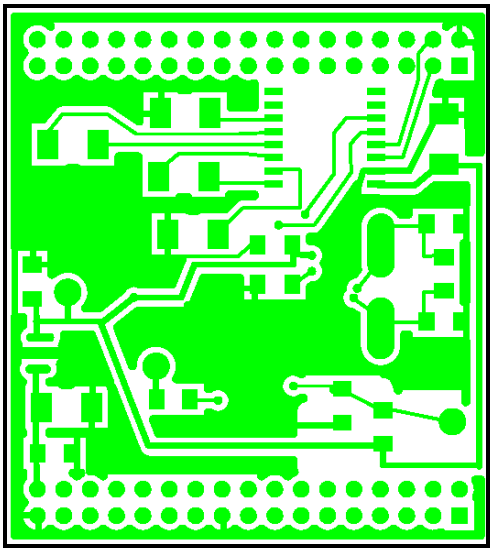
Assuming a valid command was found ('1' in this limited case), the program execution jumps down to continue running at the label Set_Freq on line 15. The user is prompted for entry of a frequency string at lines 16-17 which displays a line of text at label Enter_Freq_msg, and the program waits at '_gets' (line 16) for a full numeric string to be entered. Once the user hits <ENTER> to end

the frequency string entry, the program calls a library routine 'Set_DDS" on line 19. This routine converts to binary the frequency string residing in '_inbuf' and sends the appropriate control word to the DDS, thus setting the DDS output to that specified frequency. Once that has been accomplished, line 20 instructs the computer to prompt for another command again.

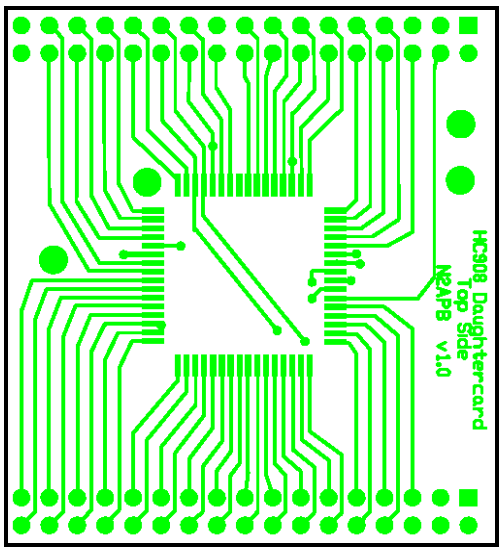
That Wasn't Hard …Was it?

For many of you, the joy is not in creating new capabilities, but in using the existing ones. Whatever your preference, this 'Hello DDS' program is still for you because you can have fun creating and modifying the Template as described, or you can just download this program from the Digital Breadboard website, load it into your HC908 card and use it directly. It's that simple. But no matter which route you follow, you'll end up having a new program for your HC908 Daughtercard that can serve an important role on your bench generating useful frequencies for test, measurement and VFO control.

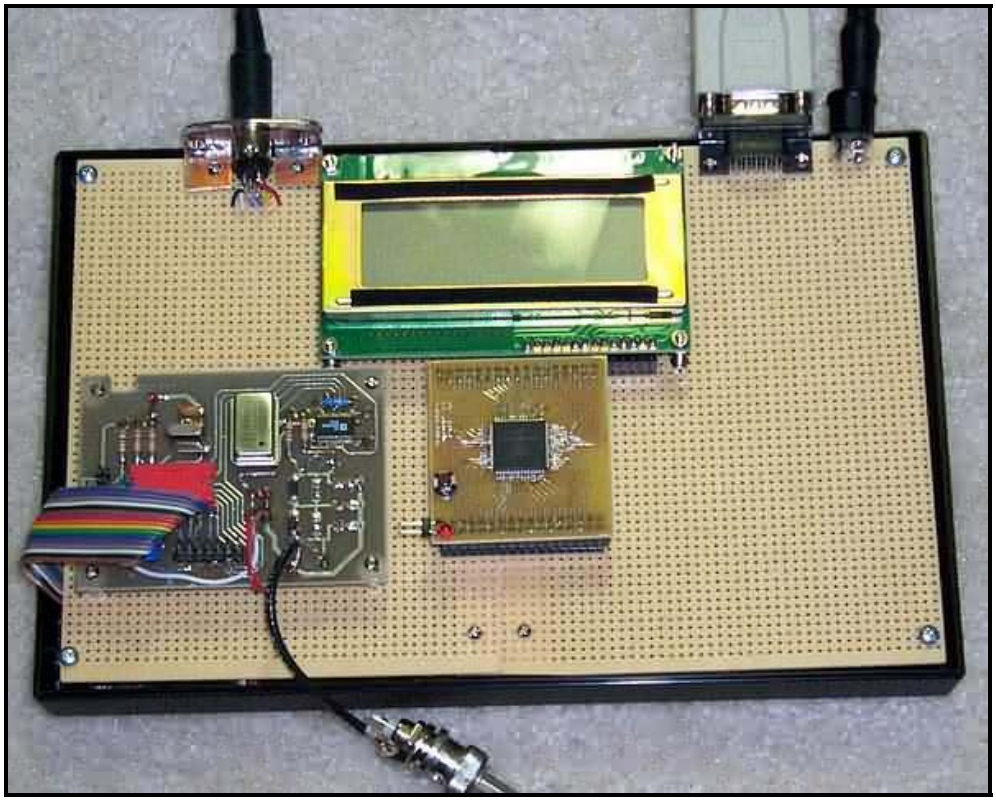
Section 7
Schematics, Photos & Diagrams



Bottom side of Daughtercard.

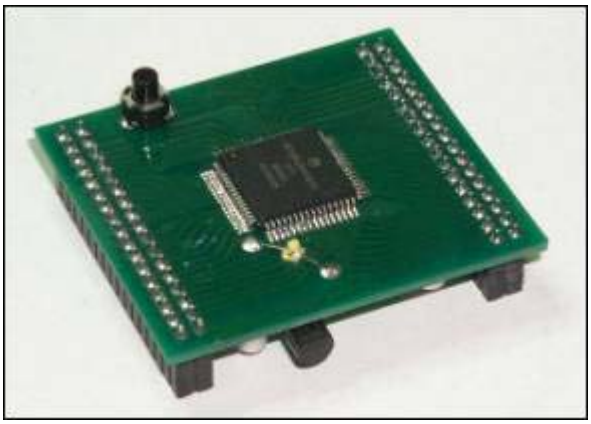


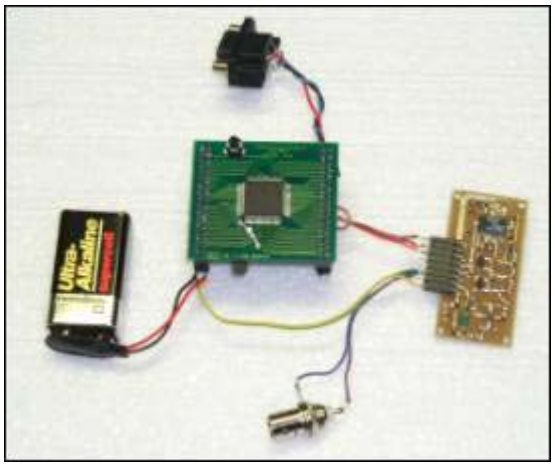
Top side of Daughtercard.



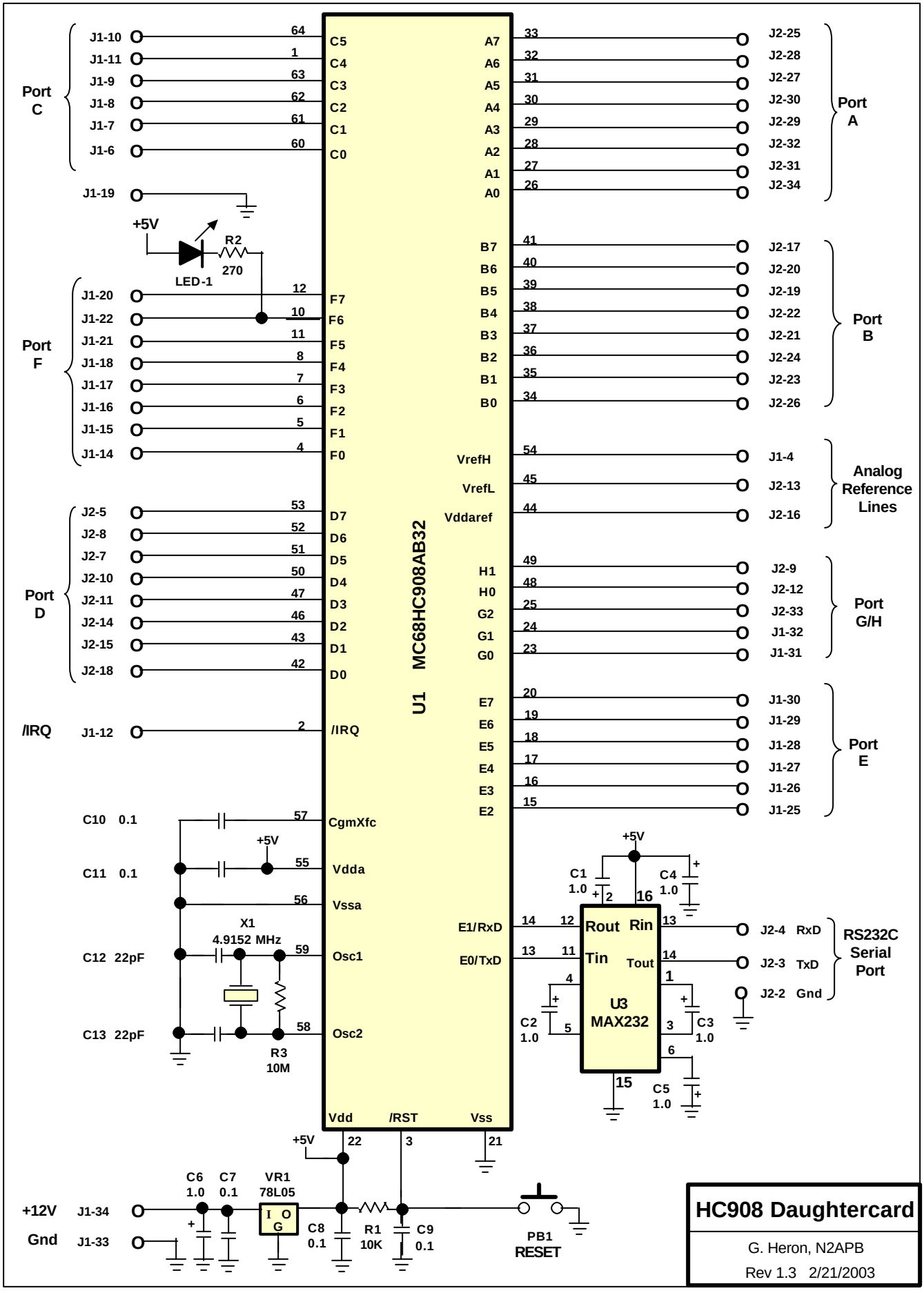
An early prototype of the Micro908, with the HC908 Daughtercard at the core of the project.

This photo shows the DDS Daughtercard included in the Exerciser … which is a test fixture that all HC908 Daughtercards get plugged into in order to get tested and qualified before being programmed with the appropriate software before shipping.





The HC908 Daughtercard controller is a popular and recurring component with the DDS-30 Daughtercard. Together they make a flexible combination of modules that can be used in many different applications, using a common set of software libraries (subroutines). The configuration of the two modules shown in the right photo is the minimal hardware arrangement needed for a "dumb terminal" (like HyperTerm on a Windows PC or ProComm on a handheld computer) to command the HC908 to generate frequencies … in effect, the HC908 and DDS daughtercards become a remotely-controlled signal generator!


Port C
J1-10
J1-11
J1-9
J1-8
J1-7
J1-6
64
1
63
62
61
60
C5
C4
C3
C2
C1
C0
J1-19
+5V
R2
270
LED-1
Port F
J1-20
J1-22
J1-21
J1-18
J1-17
J1-16
J1-15
J1-14
12
10
11
8
7
6
5
4
F7
F6
F5
F4
F3
F2
F1
F0
Port D
J2-5
J2-8
J2-7
J2-10
J2-11
J2-14
J2-15
J2-18
53
52
51
50
47
46
43
42
D7
D6
D5
D4
D3
D2
D1
D0
/IRQ
J1-12
2
/IRQ
U1 MC68HC908AB32
A7
A6
A5
A4
A3
A2
A1
A0
33
32
31
30
29
28
27
26
J2-25
J2-28
J2-27
J2-30
J2-29
J2-32
J2-31
J2-34
Port A
B7
B6
B5
B4
B3
B2
B1
B0
41
40
39
38
37
36
35
34
J2-17
J2-20
J2-19
J2-22
J2-21
J2-24
J2-23
J2-26
Port B
VrefH
VrefL
Vddaref
54
45
44
J1-4
J2-13
J2-16
Analog Reference Lines
H1
H0
G2
G1
G0
49
48
25
24
23
J2-9
J2-12
J2-33
J1-32
J1-31
Port G/H
E7
E6
E5
E4
E3
E2
20
19
18
17
16
15
J1-30
J1-29
J1-28
J1-27
J1-26
J1-25
Port E
C10 0.1
C11 0.1
C12 22pF
C13 22pF
+5V
X1
4.9152 MHz
R3
10M
57
55
56
59
58
CgmXfc
Vdda
Vssa
Osc1
Osc2
Vdd
/RST
Vss
22
3
21
+5V
C1
1.0
C4
1.0
C2
1.0
C3
1.0
C5
1.0
16
15
U3
MAX232
Rout
Rin
Tin
Tout
14
12
13
11
4
5
1
3
6
2
E1/RxD
E0/TxD
J2-4 RxD
J2-3 TxD
J2-2 Gnd
RS232C Serial Port
+12V
Gnd
J1-34
J1-33
C6
1.0
C7
0.1
VR1
78L05
I
O
G
C8
0.1
R1
10K
C9
0.1
PB1
RESET
HC908 Daughtercard
G. Heron, N2APB
Rev 1.3 2/21/2003