Technological Arts

APPLICATION NOTE AN106

Using Adapt912DG128

DOCUMENTATION

In addtion to the schematic diagram and pinout supplied with your board, you should get a copy of the 68HC912DG128 Advance Information (#MC68HC912DG128/D) from Motorola. This 400-page document is available in Adobe Acrobat format from the Motorola Literature Center s website, or in printed form, free of charge, from the Motorola Literature Center (1-800-441-2447). You should also get the applicable Errata sheet so you ll be aware of any bugs that may exist. Look for the mask number printed on the chip, and retrieve the corresponding Errata sheet from Motorola s website. Visit our RESOURCES webpage for links to Motorola literature.

RESIDENT FLASH UTILITY

The 68HC912DG128 MCU has eight blocks of Flash, consisting of 32K each. They are accessed through a 16K window from 0x8000 to 0xbfff. Each block is independently erasable and programmable, and has an upper 8K protected boot block . The last 16K page of the uppermost block is also accessible through the unpaged address space from 0xc000 to 0xffff. We have installed a Flash utility in the upper 8K protected area of the topmost block (ie. from 0xe000 to 0xffff) to facilitate erasing and programming individual blocks of Flash.

Following an MCU reset, this loader executes, examining the state of input port line PORTA6 to decide whether to display an erase/program prompt (via the RS232 port at 9600 baud), or to run a program that you have previously loaded. Since the loader includes the vector space of the MCU, pseudo-vectors are implemented for your program s use (refer to the **DG128demo** folder on the Adapt912 Starter Package disk for an example of their use). When setting up your compiler/linker (or assembler directives), you simply re-define the vectors to 0xdfxx instead of 0xffxx. The only penalty for this pseudo-vector implementation is a couple of extra instructions executing every time your interrupt service routine is called (due to the vector redirection by the loader). To load your file, you simply set switch SW2 to LOAD, press Reset, and switch on Vfp. Choose the block to erase, and then choose P. With your terminal program set up to wait for * at the end of each line of the .s19 file (or end-of-line delay set to 100 ms), use the terminal program s ASCII transfer function (eg. SEND TEXT FILE) to send your .s19 file. When the transfer is complete, switch off Vfp. Move SW2 to RUN and press Reset to run the program you just loaded. (Note that this utility does not currently support S2 records.)

USING A BDM POD

If you have a BDM pod, you can erase the Flash loader described above, and have access to the entire 128K of Flash for your program-- no pseudo-vectors needed.

Suitable low-cost BDM pods include Technological Arts products Adapt912 or Adapt912B32 (jumpered for pod mode) and the low-cost MicroBDM12 pod. Other alternatives include the BDM12 from Kevin Ross, Motorola s 912EVB jumpered for pod mode , and P&E Micro s BDM Cable. More expensive pods with additional capabilities are available from P&E Microsystems, Noral, Cosmic, and various other vendors. You should check with the vendors to ensure that they include support for erasing and loading your code directly into the Flash of a 68HC912DG128 chip.

To use a DBug12-based pod (such as the Technological Arts products mentioned above and Motorola s 912EVB), connect the pod s ribbon cable to the BDM IN connector on Adapt912DG128 (the target).

Open a terminal window set for 9600 baud, and connect the pod s serial cable to the appropriate COM port on your computer.

Make sure the VFp switch is off, and apply power to the target board only. Supply 9 to 12VDC via J1, or regulated 5VDC via the designated pins on one of the 50-pin headers, H1 or H2 (see pinout diagram or schematic to determine which pins to use). The BDM pod will receive the 5V power it requires from the target board, via the ribbon cable. When using a BDM pod, always set the target board s MODA and MODB jumpers to 0 (ie. SIngle-chip Mode). To support erasing and programming Flash in the DG128, your pod will need to have D-Bug12 v2.1.0 or higher. If you re using MicroBDM12DX (a 68HC912D60-based BDM pod), it will be running D-Bug12 v3 or higher.

After applying power, you should see a 3-item menu (unless the parameters are already valid). The first selection allows you to specify the target board s crystal frequency (if different from the displayed value). Enter a value of 8000 (for 8 MHz) and press ENTER. If DBug12 displays an R> prompt, enter **RESET** to stop the target board before proceeding.

Enter **DEVICE DA128** so DBug12 will know the attributes of the target MCU (if you always use the pod with a DG128 target, you won t need to do this in subsequent sessions). To erase Flash, turn on Vfp and type **FBULK**. To load your .s19 file, enter **FLOAD** (with an offset, if appropriate). Use the ASCII file transfer method described above to program your .s19 (or .s28) file into the chip, and then turn off Vfp. Reset your target board, and your code should be running. For details, see the relevant D-Bug12 Reference Guide.

DON'T FORGET THE COP!

The COP watchdog is enabled out of reset in all 68HC12 MCUs, so don t forget to clear the COPCTL register at the beginning of your code if you re not planning on servicing the COP. Forgetting to disable the COP is the most common reason user code doesn t work properly after the BDM pod is detached (the BDM pod s presence puts the target chip in a special mode which disables the COP automatically).

PHASE-LOCKED LOOP

While the DG128 is capable of running at 16 MHz, the on-chip oscillator configuration is highly susceptible to external effects such as humidity and fingerprints, so Motorola recommends a maximum crystal frequency of 8MHz. To run the chip at 16 MHz, the internal phase-locked loop feature can be enabled. The correct loop filter components for this feature have been installed on the board, so it is simply a matter of setting up the appropriate internal registers to implement this feature. The on-chip Flash utility implements this, and you can refer to its source code (in **128load8** folder) on the Adapt912 Starter Package Disk to see how to do this.

GOING FURTHER

The 68HC12 is a powerful and complex microcontroller family. If you re not already an expert on this MCU, you are urged to join the Motorola 68HC12 email discussion group to network with other HC12 users and Motorola engineers. Visit our Tech Support webpage for more information, and follow the links in our RESOURCES page for more tools, example code, and instructional materials.

Phone: (416) 963-8996 Fax: (416) 963-9179
support@technologicalarts.com
sales@technologicalarts.com
www.technologicalarts.com