Last update: 29 March 2000

Application Note 105: Using Adapt912 MX1 Wide RAM+ Expansion

Accessing External RAM

Probably the first thing you will discover after you have added an MX1 card to your Adapt912 is that you can't access external memory from DBug12. This is because DBug12 has already written to the WRITE-ONCE registers controlling operation modes at startup, so you've lost your chance to switch to expanded mode. If you try resetting with Adapt912's jumpers set for expanded mode (instead of single-chip mode), you'll get nowhere, since the Flash (containing DBug12) goes away at startup, and the MCU tries to fetch it's reset vector from RAM. Since you haven't had a chance to load a program into RAM yet, the reset vector is undefined.

So, you need DBug12. But you'll need to modify it to enable Expanded Wide mode via a register write during startup. This requires changing one byte in the s-record. You could use DBug12 to disassemble itself, line by line from startup, until you find the instruction which initializes the MODE register. Then take note of the address and value of the byte that needs to be changed and then use any text editor to edit the corresponding location in the s-record. Of course, you will have to be familiar with s-records in order to be successful at this. It turns out that the byte you need to change (from $08 to $68) is located at $F716. A version of the DBug12 s-record (v202d.s19) that has already been modified for you is available in http://www.technologicalarts.com/myfiles/files/v202d.zip. To use it, you need to have the normal version of DBug12 installed in Adapt912 (ie. as it came from Technological Arts). Then you set the MODE SEL jumpers on Adapt912 for LOAD mode, reset the board with a terminal connected at 9600 baud. When you get the (E)rase, (P)rogram, or (L)oadEE prompt, switch on the VFp switch (make sure you supply regulated 12VDC via J1), and choose E. After Flash erases, choose (P) and send the modified file via ASCII download with "wait for *" or "delay 100 ms after each line" set in your terminal program. See the Adapt912 manual for more details on loading files this way.

What other tricks has the MX1 got up its sleeve?

If you have port lines PS2 and PS3 available, you can take advantage of some logic included on the MX1 card to make working with DBug12 a lot more productive. PS3 can be used as a software-controlled WRITE PROTECT line, so that you can copy DBug12 into the upper half of RAM automatically following a reset, then write protect it, and disable the Flash. This has the advantage of allowing you to use the actual HC12 vectors instead of the pseudo-vectors DBug12 forces you to use. Again, a modified version of DBug12 must be loaded into Flash, following the same method outlined above. As you may have guessed, the file referenced above (v202d.s19) already has been modified so that it copies itself to RAM, activates Write Protection, and then turns off the Flash.

How do I set the jumpers?

JB2 can be used to select between the two banks of 32K words of RAM. It can be done through software, via PS3 (with the jumper in place), or manually, by leaving the jumper open (for upper RAM bank) or tying pin 2 of the JB2 to Ground (for lower RAM bank).

JB1 is for software or manual Write Protect. If a jumper is placed between pins 1 and 2 (ie. the position labelled R/W*), RAM will operate normally-- both reads and writes will work. If it is placed between pins 2 and 3 (ie. the position labelled W*), the upper half of the selected RAM bank can be protected via PS2. As mentioned above, the idea is that you can load DBug12 into this protectable bank, and then protect it from getting corrupted by runaway code during development.

Note: If you don't have PS2 and PS3 available (ie. you're using them for something else), you won't be able to take advantage of these features. Of course, you could jumper them to other available port lines, and modify DBug12 as needed. But don't worry-- the modified DBug12 will work anyway, as long as you set JB1 to the R/W* position. (Of course, the RAM will not be protected.)

Two Code Development Approaches:

1) Use DBug12 as a resident Debug/Monitor program, load your subroutines into RAM below $8000, and use the G command to jump to the subroutines. You need an RTS at the end of your routine if you want it to return control back to DBug12 after it's executed. Otherwise, you should put it in an infinite loop. Then just press the reset button to return to DBug12. If you're testing your interrupt routines, you can use the actual vector addresses defined for them in the HC912 data, since the memory in that space is now RAM. Don't forget to enable writes to the upper block of RAM, via JB1.

2) Forget DBug12, and load your complete program into RAM (a contiguous 60K block from $1000 to $F7FF is available). You can do this by using the EEPROM-resident quick RAMloader. After loading it into EEPROM via

DBug12 in BOOTLOAD mode, move the MODE SELECT jumpers to JMP-EE mode, and reset. The resident quick RAMloader will now present you with two choices. Load an s-record into RAM, or Run a program you loaded previously. Since you don't need DBug12 now, your s-record can occupy as much of the 60K external RAM as it needs. Your program can use the actual 912B32 vectors. There are a couple of caveats, however. If you reset the board, your program in RAM will be overwritten with DBug12. So you'll have to load it back in again. However, if you switch the jumpers to expanded wide mode before resetting, the MCU should execute your code from RAM. This is because Flash is disabled when the MCU is reset in expanded modes.

Gotchas...

- make sure your code clears the COPCTL register or services the COP regularly
- clock stretch defaults to maximum, so if you want to optimize the performance of your code, disable clock stretch (the RAM on the MX1 has 15ns access time, so clock stretch is definitely not required)
- if you have the battery-backed RAM option (using DS1210), the power-up delay for RAM Chip Enable can be as much as 250 ms. That means your code will not start executing properly from powerup if the board is jumpered for Expanded Mode. Of course the COP will kick in and reset the MCU a few times before the RAM is fully awake anyway, so your code will start running correctly eventually.
- the RAM chip draws a lot of current; if your application is battery-powered this could be a concern, since the battery may drain rapidly

More than memory...

Other functional blocks are included on the Adapt912 MX1 card. Below are descriptions and some tips on using them. If you purchased a minimally-populated card, and plan to add these functions yourself, you should read the following information carefully.

Battery-backup for RAM:
- important! the DS1210 has a 200ms start-up delay, so your program cannot execute from RAM directly from a power-on reset
- it is recommended that you start up in single-chip mode, then switch to expanded mode via 912 registers, and then jump to your program in RAM (if you want a standalone application without having to manually reset it following power-on)
- if your board doesn't have this chip, there should be two jumpers: CE to CEO, and Vcc to VCCO, for the RAM to work properly

Battery-backed clock/calendar:
- this chip uses PJ7, so you need to remove R11 and/or D1 (red LED) on Adapt912 for proper operation
- data sheets on these parts, as well as example assembly routines, are available-- check the Applications page, or contact us

SPI-based 8-bit output port, and 32K serial EEPROM (SPI):
- data sheets on these parts, as well as example assembly routines, are available-- check the Applications page, or contact us

SPI-based UART:
- example assembly routines are available-- check the Applications page, or contact us
- RS232 interface is functional
- the RS485 interface portion of this circuit on is non-functional (on Rev.0 boards)-- don't stuff it!

Adapt12 Adapt11 Family MicroCore-11 MicroStamp11 Emplant11 Solderless Breadboard Adapters
Drawings Software Price List Order Form What's New Tech Support Resources Applications Home

Technological Arts, Toronto, Ontario, Canada
Toll-free: 1-877-963-8996 (USA and Canada) Phone: +1(416) 963-8996 Fax: +1(416) 963-9179
sales@technologicalarts.com www.technologicalarts.com