Programming the 68HC912B32 Internal EEPROM using D-Bug12

The 68HC912B32 has 768 bytes of internal EEPROM originally mapped from $0D00 to $0FFF. The internal EEPROM can be used to run user code. At run-time, it is considered ROM (read-only memory). Therefore, all program variables (areas of memory declared with DS.B, DS.W, or DC.B/DC.W that are later modified at run-time) must be placed in RAM and not EEPROM.

Programming the internal EEPROM using D-bug12 requires that the code first be loaded into RAM using the LOAD command. Then, it can be moved to internal EEPROM using the MOVE command. The D-bug12 LOAD command cannot load programs straight into EEPROM because it does not incorporate the delay required to operate the internal charge pump. On the other hand, the MOVE command can recognize when an EEPROM delay is required. Programming internal EEPROM can be done as follows:

1. Organize your code so that all DS.B, DS.W, and DC.B/DC.W statements that are later modified at run-time are placed in RAM rather than EEPROM.

2. Use an ORG statement such as “ORG $0D00” to place your code at the beginning of EEPROM.

3. Assemble the code using MiniIDE.

4. Use Hyperterminal or the MiniIDE terminal window to communicate with your UF 68HC12 development board.

5. The .S19 must be loaded into SRAM first. This example assumes that there is 8K of external SRAM starting at address $4000. We want to place the contents of the .S19 that are originally assembled at $0D00 at address $4000. Therefore, the LOAD command is used with an address offset of $3300 (because we want $0D00 + offset = $4000; offset = $3300).

   > load 3300

6. Once the program is loaded into address $4000, it can be moved into EEPROM using the MOVE command as follows:

   > move 4000 42ff 0d00

   The move command parameters are:
   > move <source start address> <source end address> <destination address>

   The code is now properly loaded in EEPROM starting at $0D00.

7. The program can then be run using the usual G command.

   > g 0d00