Trilateration Module Overview

University of Wyoming Distributed Robotics Laboratory
Version 20090209

Introduction

The "trilat module" was developed through the efforts of graduate students and faculty at the University of Wyoming Distributed Robotics Laboratory (UWDRL), a collaboration of the Computer Science and Electrical and Computer Engineering Departments in the College of Engineering and Applied Science. This brief document is intended to provide the new investigator with the bare essentials for beginning work with the module. Exhaustive detail of the module construction and application can be found on the web at the following link, along with the details of "what is trilateration": http://www.cs.uwyo.edu/~hamann/TrilatModule

What Should it Look Like?

An assembled view of the trilat module is shown below in Figure 1. Due to storage and shipment considerations, you may receive a trilat module that "requires some assembly." Most typically, the rods holding the ultrasound transducers will be disconnected both from the mechanical base platform as well as from the electronic ports of the input conditioning boards (the XSRF boards). All that's needed for assembly is a pair of small adjustable wrenches and a careful plugging of the two-pin electronic connectors (these are not polarized, so you can't plug them in backwards).

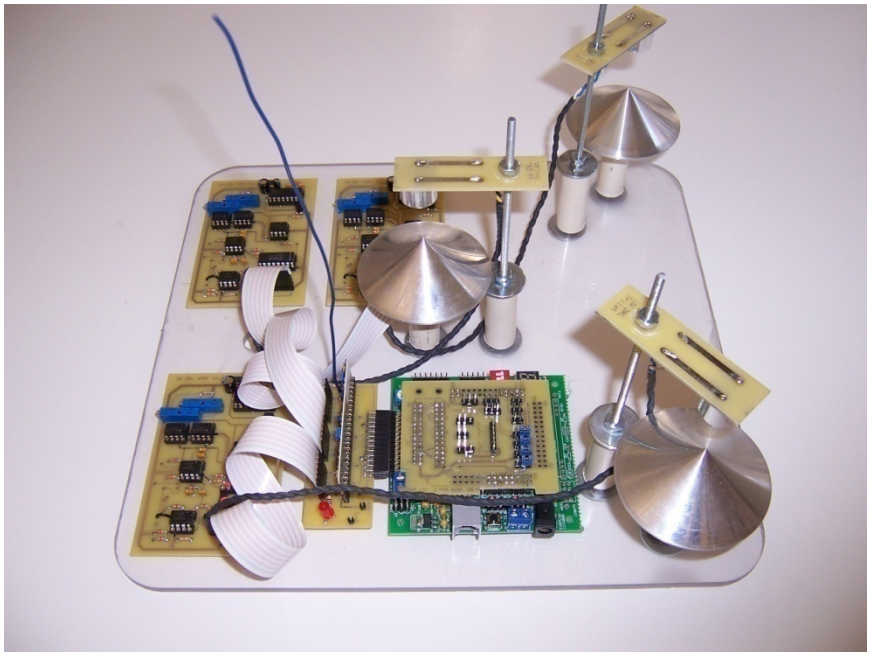

Figure 1. A Side View of the UW DRL Trilateration Module Fully Assembled.

The annotated photo in Figure 2 provides a quick reference for identification of the sub-modules or boards of the trilat module. As indicated by the annotations, the perpendicular line segments formed by the mounting points for the parabolic aluminum cones describe Cartesian coordinate axes, from which the location computations of the trilat module are based.

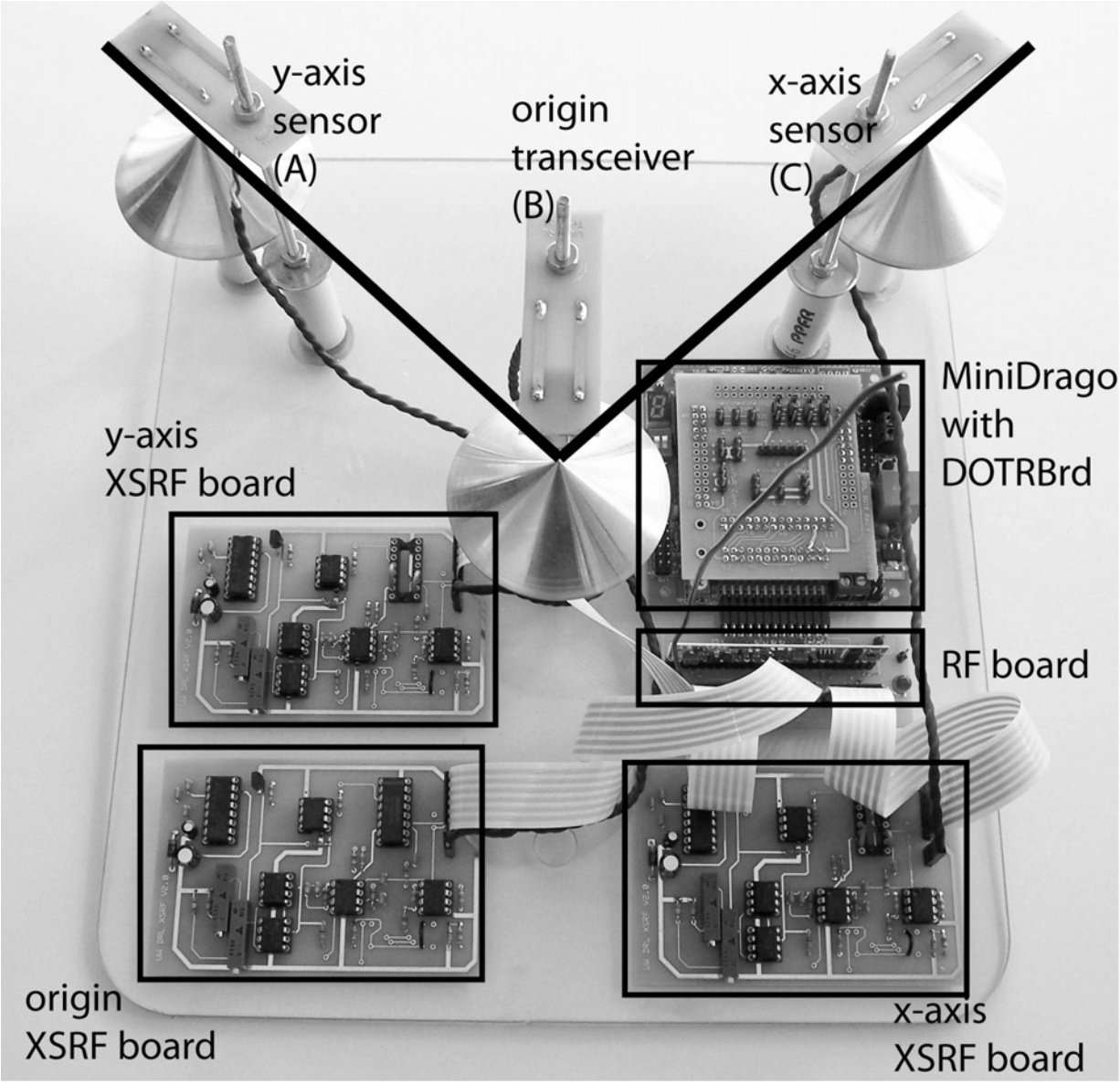

Figure 2. An Annotated Top View of the Trilat Module.

CAUTION: The ribbon cables which connect the XSRF boards to the RF board ARE POLARIZED. The outside conductors provide +5 VDC and GND respectively, as viewed left-to-right at the bottom edge of the RF board, as shown in Figure 2. The connection at the XSRF boards should be made with +5V on top, GND on bottom, as viewed in Figure 2. You'll note that the XSRF boards are essentially identical, with the exception of one integrated circuit which must be included on the "origin" board, but which is replaced by two wire jumpers on both the "y-axis" and "x-axis" boards. Figure 3 provides some details.

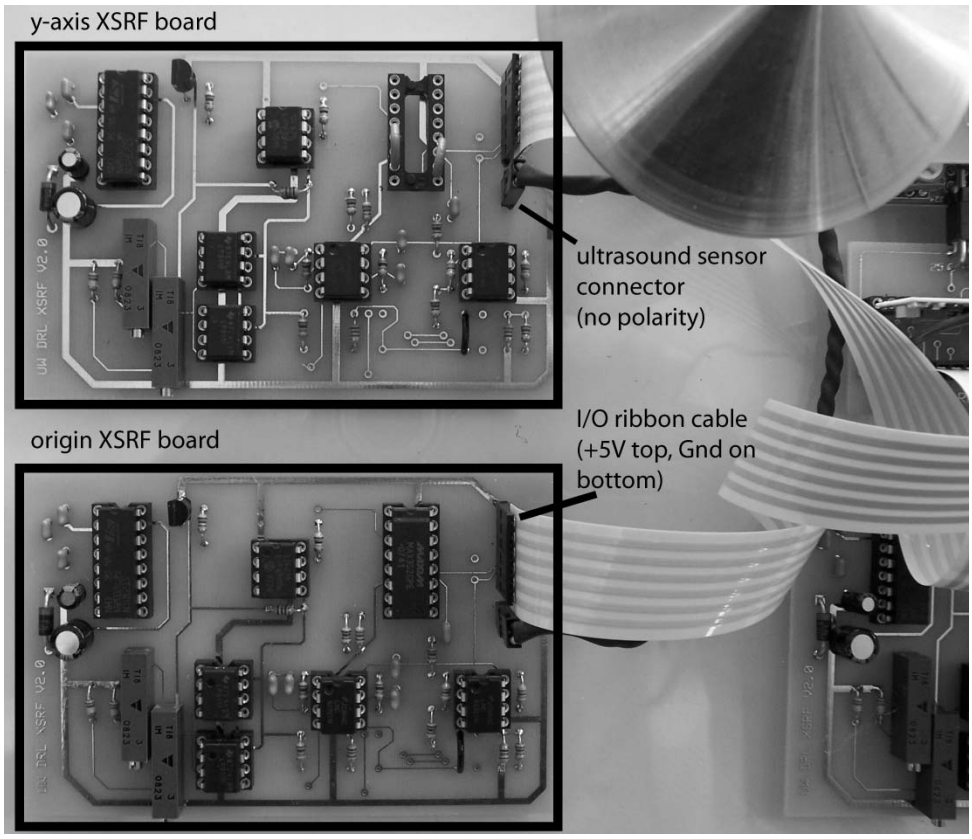

Figure 3. Annotated Details of the XSRF I/O Connections.

The circuit boards are mounted on the prototype acrylic platform with simple hot-melt glue. Should you need to remove, rearrange or otherwise modify the layout, the glue bond is readily broken with a knife or screwdriver carefully wielded.

Dimension Considerations

A quick summary of some dimensions which enter into the default configuration of the trilat module firmware is provided in Table 1.

Table 1. Measured and Empirical Dimensions Assumed in Trilat Module Computations

Dimension	Value and Units
Length from Origin to X and Y Sensor Cones	6 inches
Length from Tip of Parabolic Cone to Ultrasound Sensor Face	1 inch
Speed of Sound in Air	343.6 meters/second
Serial Port I/O Communication Rate with Module	9600 baud

How To Get Started with this Beast

The computational heart of the trilat module is a Freescale (Motorola) HCS12 16-bit microcontroller, located on a MiniDragon+ Trainer evaluation board. The MiniDragon+ provides a very stable platform for firmware design and evaluation. A custom printed circuit board (the DOTRBrd) is attached to the principal headers of the MiniDragon+ to provide signal routing for the trilat module functionality. The annotated photo in Figure 4 outlines the critical points of connection and configuration.

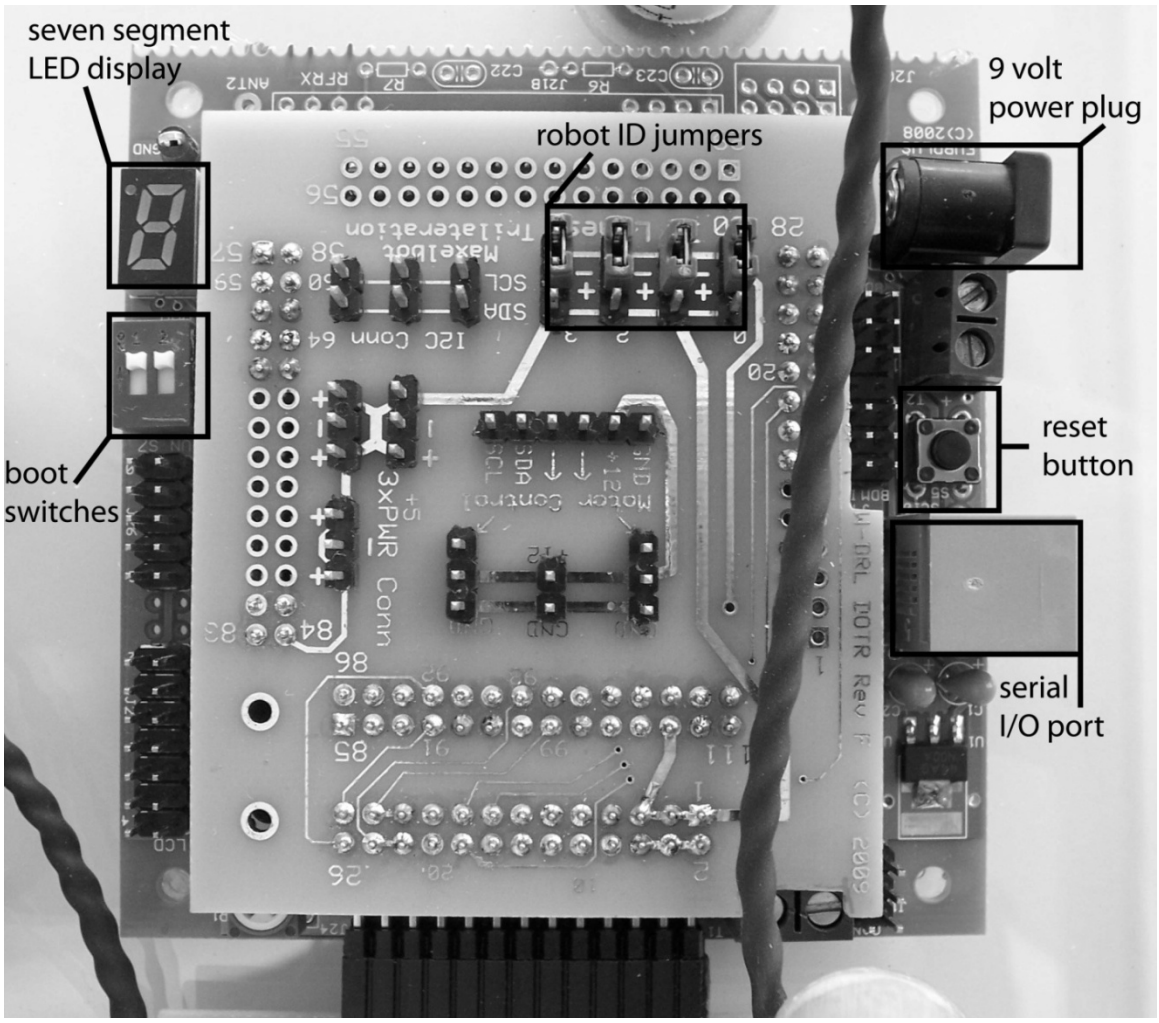

Figure 4. MiniDragon+ and DOTR (Daughter) Board Connection and Configuration Reference

Provided with your trilat module should be the following items, available from Wytec, the MiniDragon+ manufacturer (http://www.EVBPlus.com):

- A 110VAC to 7.5-9VDC (300mA) power adapter,
- A 9-pin RS-232 Serial to 6P4C modular jack cable,
- A CD containing documentation, application notes, schematics, examples for MiniDragon + use, as well as binary images for the Bootloader and D-Bug12 monitor as customized by Wytec.

Trilateration Module Configuration at the MiniDragon+

The trilat module is designed for distributed use across a collection or swarm of small robots. To uniquely identify each robot, four jumpers are provided on the DOTRBrd to allow for configuration of a unique 4-bit binary address for each robot in your swarm. As shown in Figure 4, the robot ID would be 0000 (the jumpers are positioned on the upper or "minus" side, thus a 0), with the least significant bit located on the right side in the photo.

Applications may be developed and loaded onto the MiniDragon+ in three essentially unique ways:

- RAM-only code and data (for early design and testing of algorithms, code is deleted from HCS12 memory at next reset, typically depends upon a firmware resident monitor program such as D-Bug12). A maximum of 12k bytes of static RAM available for use.
- EEPROM (for non-volatile variable storage OR program storage for small application programs). A maximum of 4k bytes of EEPROM available for use.
- FLASH code with RAM and EEPROM data (for long-term firmware configurations). A maximum of 256k bytes of paged FLASH available for use.

The "boot switches" indicated in Figure 4 provide a Wytec and D-Bug12 customized method for accessing each of these three memory configurations. As you develop code for trilat, you'll use them frequently to reconfigure.

Starting Up the Default Firmware

To get started with trilat functionality, a small trilat firmware (FLASHOne) has been installed in the MiniDragon+ by the UWDRL. All that you need to do to fire it up is…

- Ensure that the two boot switches are both positioned toward the seven segment display, as shown in Figure 4.
- Configure each of your trilat modules with a unique robot ID, ensuring that one of the modules is robot 0 (bits 0000).
- Provide DC power to each MiniDragon+ (this power is distributed to all sub-modules through the DOTRBrd and RF board connectors).

In this default configuration, robot 0 will act as the only "transmitter." To confirm that it is operating correctly, you should see a clockwise light rotation of the outer segments of the LEDs on robot "zero's" MiniDragon+. At each step, it is initiating a trilat "lightning and thunder" event.

All other "robot" trilat modules should be in "receiver" mode. To confirm this, observe the outer segments of the corresponding seven segment LED display. As you move these receiver trilat modules around the vicinity of robot 0, the LED segments should be directed toward robot 0. In addition, each listener should be communicating, via serial I/O, the calculated Cartesian position of robot 0, within their own "egocentric coordinate system." To view these computations, all you need to do is connect to the receiver MiniDragon+ serial I/O port with a standard RS-232 terminal (for example, a windows PC running Hyperterminal and connected by the Wytec communication cable to the MiniDragon+, 9600

baud). Press the MiniDragon+ reset button, see Figure 4, at any point to restart any "robot." Figures 5 and 6 demonstrate this configuration with one transmitting module and one receiving module.

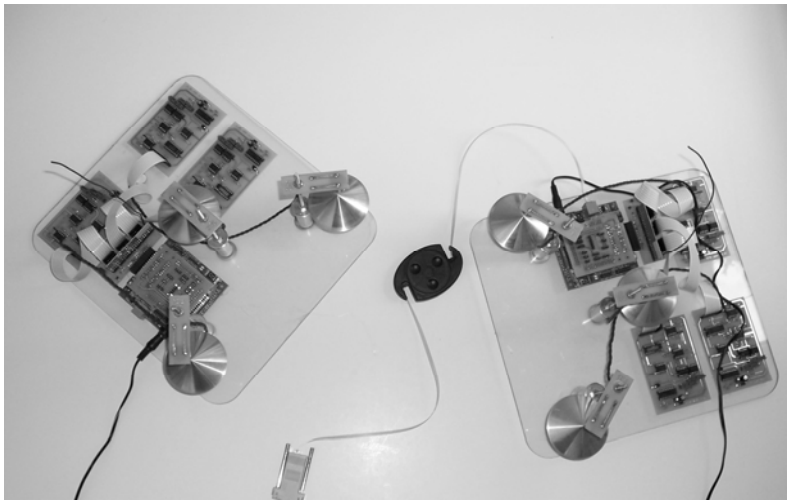

Figure 5. A Transmitting (left) and Receiving (right) Trilat Module Pair in Action.

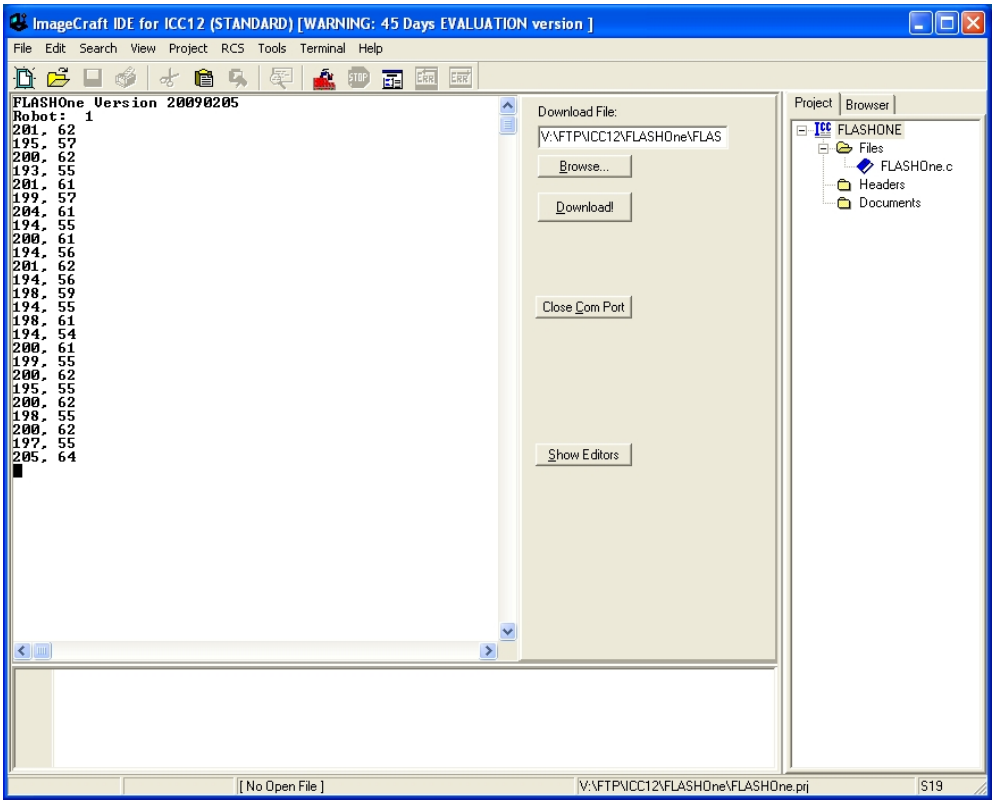

Figure 6. Example Serial I/O Stream Captured from a Receiving Module with the FLASHOne Firmware.

In Figure 6, the communication window is provided by the ICC12 Integrated Development Environment. The readings shown indicate that Robot 1 senses a transmitter at x=20 inches, y=6 inches (displayed values are given in units of tenths of an inch). The update rate is "free-running" in this code.

Modifying the Trilat Code

The FLASHOne code is provided as a single C language source file, written for the ICC12 Integrated Development Environment provide by ImageCraft Development Tools (http://www.imagecraft.com). A fully functional 45-day free trial of this tool is available for download from the ImageCraft website. The FLASHOne.c file can be downloaded from http://www.cs.uwyo.edu/~hamann/TrilatModule . The code provides in-line comments which should enable the investigator to change configurations between fully FLASH-able and fully RAM-able implementations. Details of these two development modes are described in the remainder of this document.

Working with RAM-only Code Development on the MiniDragon+

The RAM-only mode of code development is strongly recommended when a significant number of code modifications will be made. Iteration of write-test-refine cycles are simplified by the higher speed of download and debugging on the MiniDragon.+ To enable this mode, however, please consider the following:

The MiniDragon+ FLASH must contain the D-Bug12 Monitor

When an application program is written to the MiniDragon+ FLASH, the default configuration of the HCS12 will be modified by first erasing the resident D-Bug12 Monitor firmware. This is precisely the state of the UWDRL-delivered trilat modules with FLASHOne present. To re-write D-Bug12, the following steps must be completed:

1. Connect power and "terminal" serial I/O to the MiniDragon+ in question. NOTE: you'll need a development environment such as ICC12 or AsmIDE which can complete an s-record download in order to complete the next five steps.
2. Place the boot jumpers of the MiniDragon+ in "BOOT" mode (this places both switches "away" from the seven segment LED display).
3. Press the MiniDragon+ reset button, and observe the messages and prompt shown in Figure 7. The Bootloader is a tiny portion of D-Bug12 which is retained in FLASH until last resort (that is, you'll need to implement an external Background Debug Monitor or BDM in order to fully remove it, should you wish).
4. Press 'a' to erase the current content of FLASH (will remove all but the Bootloader). NOTE: the action will be initiated as soon as you press the 'a' key (no "Enter" required). Be patient, you should receive a new input prompt within a few seconds.
5. Press 'b' to initiate programming of FLASH, then select and download the Wytec-provided s-record object code for the D-Bug12 Monitor (on the Wytec CD, as the following file: Document\D-Bug12_Monitor\DBug12v32_MDP_16MHz.s29
6. When the FLASH programming completes and an input prompt is returned, reset the boot jumpers to the "EVB" mode (both switches "toward" the seven segment LED display). When you press the reset button on the MiniDragon+, you should next receive the D-Bug12 dialog as shown in Figure 8. You're now ready to read, write, execute and debug from RAM.

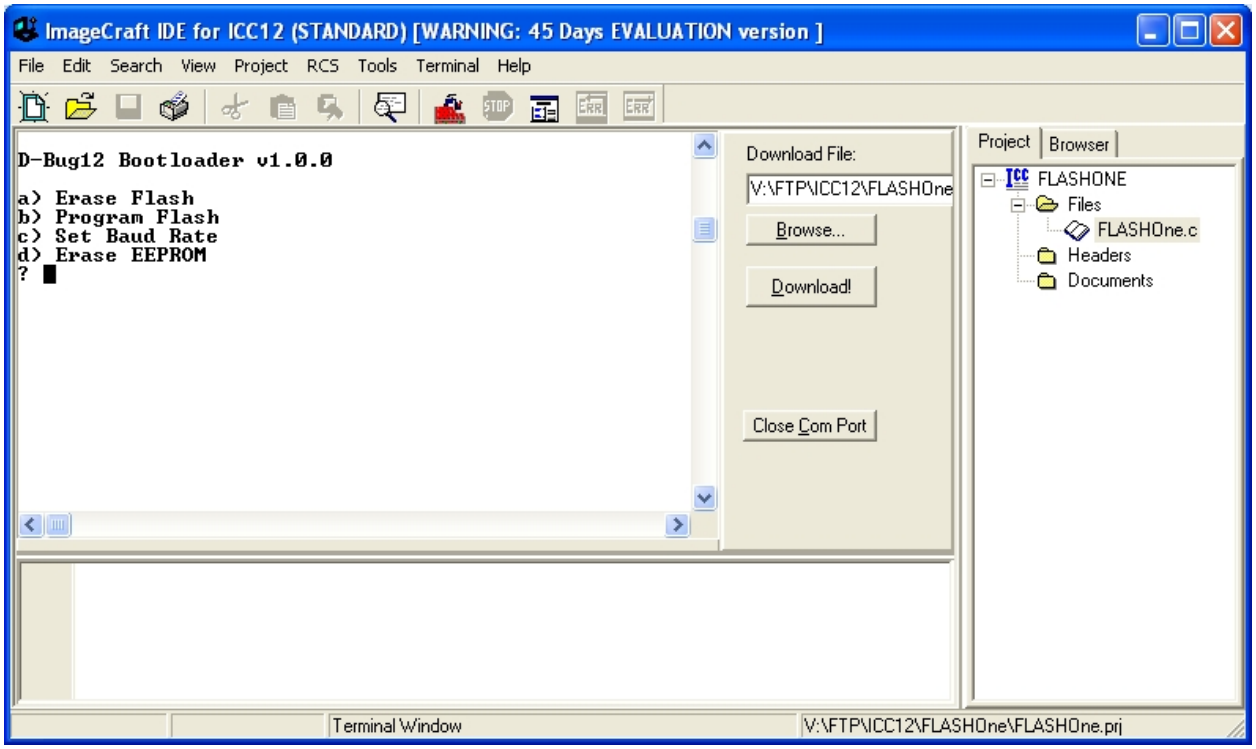

Figure 7. MiniDragon+ Bootloader Prompt.

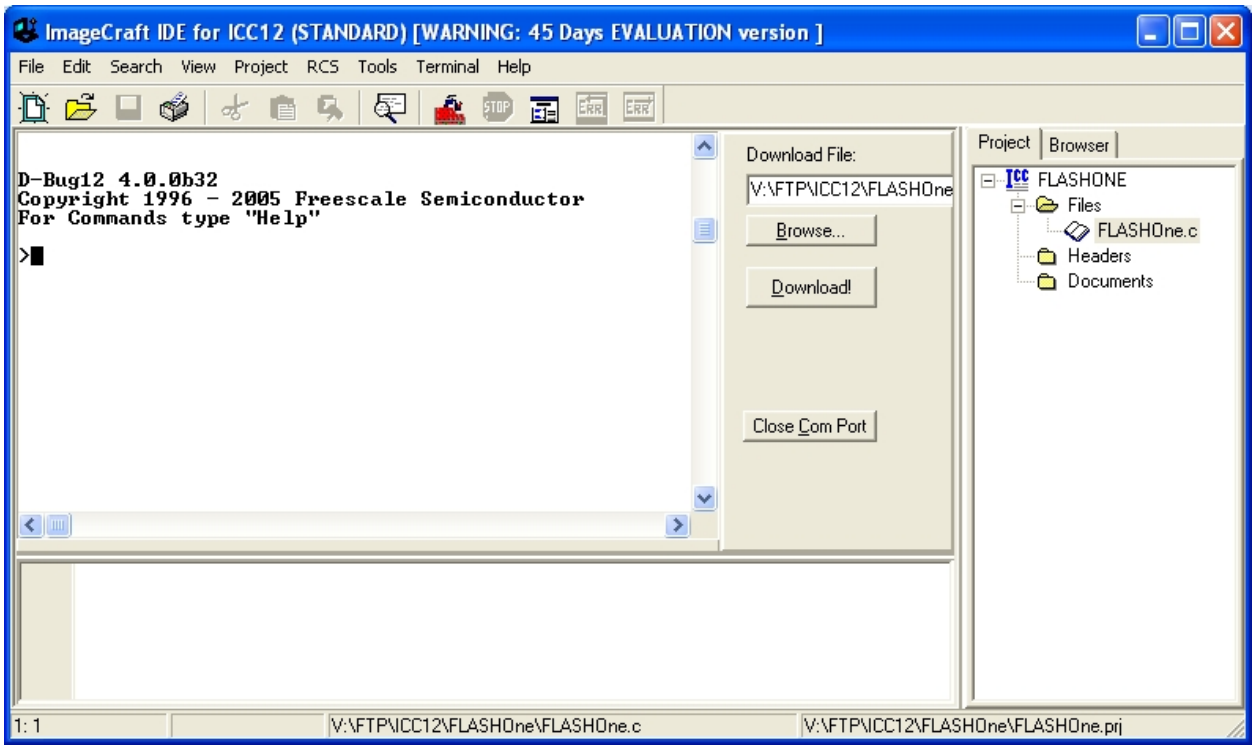

Figure 8. MiniDragon+ D-Bug12 Monitor Prompt.

Language Environment Configuration for RAM-only Code

The MiniDragon+ contains an HC9S12DG256 microcontroller with 12k of RAM occupying a portion of the lowest addresses of memory space (0x1000 thru 0x3FFF). The ICC12 IDE provides the following default configuration for program development in this RAM-only mode:

- Program Memory starts at 0x1000 .
- Data Memory interspersed with Program Memory as required.
- Stack Pointer begins at 0x4000 (stack grows toward start of memory).

Note that in the RAM-mode under D-Bug12, user serviced interrupts are handled via a secondary RAM interrupt vector table that is held in locations 0x3E00 thru 0x3E7F (refer to the "Reference Guide for D-Bug12" available on the Wytec CD as file Document\DB12RG4.pdf, pages 11-13). This allows for reasonable growth of the stack AND for a reasonably sized application program, but you may encounter limits. Also, the content of this RAM vector table is re-written at each press of the MiniDragon+ reset button, so your application program may need to re-write user vectors each time it runs.

Working with FLASH Code Development on the MiniDragon+

The FLASH of the MiniDragon+ microcontroller can be utilized at two levels:

1. Replace the D-Bug12 Monitor but retain the serial Bootloader.
2. Replace all of FLASH with application-specific firmware.

CAUTION: should you select the second option, you will need a Background Debug Monitor (a hardware accessory available from Wytec as well as many other vendors) in order to re-program and real-time debug your application code. In this introduction, I assume that you will retain the serial Bootloader, as described in the first option above. This allows you to selectively re-program only the Monitor portion of the FLASH while retaining the Bootloader (which it provides the FLASH erase and program capability for you).

Re-Programming the MiniDragon+ FLASH

To erase the D-Bug12 monitor and program the FLASH with your own application firmware, follow steps 1 through 6 of the RAM-only set-up process described above, HOWEVER YOU MUST SUBSTITUTE YOUR OWN APPLICATION OBJECT CODE IN STEP 5. Herein lies a bit of the rub: the s-record object code must follow a particular format which is not provided by default by most assembly or C IDEs for the HCS12. Luckily, FreeScale (Motorola) provides the tool required to accomplish the task.

Re-Formatting S19 Object Records to the S2/S29 Form

Provided on the Wytec disk is a utility program, SRecCvt.exe, which will convert tradition S19 object files to the S2 or S29 format required by paged memory devices such as the HC9S12DG256. In summary, here's the essential command-line example to convert an s-record file:

```
SRecCvt –m c0000 fffff 32 –of f0000 –o YourProject.s29 YourProject.s19
```

The S29 format is required for any object code written to FLASH as it recognizes the correct block size and addressing required. For more details of the s-record conversion process, refer to the "Reference Guide for SRecCvt" provided as the file Document\SRecCvtRG.pdf on the Wytec CD.

Language Environment Configuration for FLASH-able Code

The MiniDragon+ HC9S12DG256 microcontroller, in addition to having 12k of RAM in the range 0x1000 thru 0x3FFF, has 256k bytes of FLASH (electrically re-writable nonvolatile memory). The ICC12 IDE provides the following default configuration for program development in this couple RAM/FLASH mode:

- Program Memory at 0xC000, with linear expanded memory addressed at 0xC0000 thru 0xFFFFF (notice the additional address nibble in expanded form).
- Data Memory starts at 0x1000 .
- Stack Pointer begins at 0x4000 (stack grows toward start of memory).

Note that in the FLASH-mode under the Bootloader, user serviced interrupts are handled via a secondary FLASH-based interrupt vector table that is held in locations 0xEF80 thru 0xEFFF (refer to "Application Note AN2153: A Serial Bootloader for Reprogramming the MC9S12DP256" available on the Wytec CD as file Document\AN2153.pdf, pages 37-38). The programmer must specify these vector values at object code generation time in a manner dictated by the development environment. At a minimum, the start address of the code must be written to the secondary reset vector location, 0xEFFE – 0xEFFF.

Additional Resources

Full documentation, including hardware schematics and additional software designs for multi-transmitter applications, are provided online at http://www.cs.uwyo.edu/~hamann/TrilatModule .

To get you started immediately, accompanying this overview document is a CD providing the following content:

- An electronic version of this document: TrilaterationModuleOverview.pdf
- The ICC12 C sourcecode for the FLASHOne firmware: FLASHOne.c
- The self-installing archive of the 45-day free demo version of ICC12, for Windows XP: icc12dem.exe

Finally, the UWDRL group invites your comments and questions. We can be reached as follows:

- Dr. Jerry Hamann, hamann@uwyo.edu
- Dr. William (Bill) Spears, wspears@cs.uwyo.edu
- Dr. Paul Maxim, paulmax@cs.uwyo.edu