A Brainstem Tutorial using Palm Vx and Garcia

Bradley J. Barnes
July 2005

Introduction

This tutorial is designed to assist the person who wants to control Acroname's Brainstem modules (www.acroname.com) using the Palm Vx as the host controller. It will give explanations and examples of how to correctly connect and configure your Brainstem robot to work with the Palm Vx. The robot used to demonstrate these concepts is the Garcia robot from Acroname. Garcia comes assembled and includes the Brainstem controller, two Maxon A-max motors, and 6 Sharp IR sensors. Even though you may not be using the same motors and/or sensors in your robot, the methods for connecting your robot to the Palm Vx and configuring these devices to communicate with one another will be very similar. The concepts demonstrated in this tutorial can be applied to any Brainstem-based robot that uses the Palm Vx as the host controller. For Palm programming we used the CodeWarrior for Palm OS software package. The specific development tool is, of course, left up to the individual.

In this tutorial, we will discuss the Brainstem and the Garcia robot, how to download the Brainstem software, how to connect the Garcia to the Palm Vx, setting up and using motors and sensors with the Brainstem, and even give a few example programs written in C using Metroworks CodeWarrior for Palm OS.

General Brainstem Information

The Brainstem technology provides the hardware and software that allow users to efficiently build customized robots. The Brainstem suite consists of the controllers (GP and Moto modules), the hardware accessories, and the software.

The Brainstem controller modules include the GP and the Moto (seen in Figure 1). Both of the Brainstem modules have a common set of features including:

- Software library support for Java, C, and C++ on many platforms, including Windows, WinCE, PalmOS, MacOS X, and Linux
- Built-in command set that provides instant access to all I/O features
- Ability to execute 9000 instructions per second, comparable to the fastest EEPROM-based controllers
- A multitasking kernel that supports concurrent processes, interactive (slave) commands issued by the host platform or other modules, and configurable reflexive behaviors based on system inputs
- Heartbeat and power indicators showing system status
- Multiple modules may be networked via the IIC bus

In addition to these features, the GP and the Moto each offer their own unique set of features. The GP module provides all of the capabilities common to the BrainStem architecture, as well as the following:

- Onboard 1 Amp, 5V power regulation (low dropout)
- 40 MHz RISC processor
- 4 servo outputs with 8-bit resolution, variable range, and speed control
- 5 10-bit resolution A/D inputs
- 5 digital I/O pins with polling and timing functions
- 4 pins can be used for logic event timing
- 1MB IIC port
- RS-232 TTL serial port
- Sharp GP2D02 driver
- 544 bytes of RAM available to user
- Small size (2.5" square, 0.5" high) and easily stacked

The BrainStem Motion Control (Moto) module provides all of the capabilities common to the BrainStem architecture, as well as the following:

- Onboard 1 Amp, 5V power regulation (low dropout)
- 40 MHz RISC processor
- 2 PID motion control channels with variable PWM frequency from 2.5kHz - 5Mhz
- 1 dedicated 10-bit A/D input
- 1 dedicated digital I/O line (11 possible)
- 1MB IIC port
- RS-232 TTL serial port
- 368 bytes of RAM available to user
- Small size (2.5" square, 0.5" high) and easily stacked

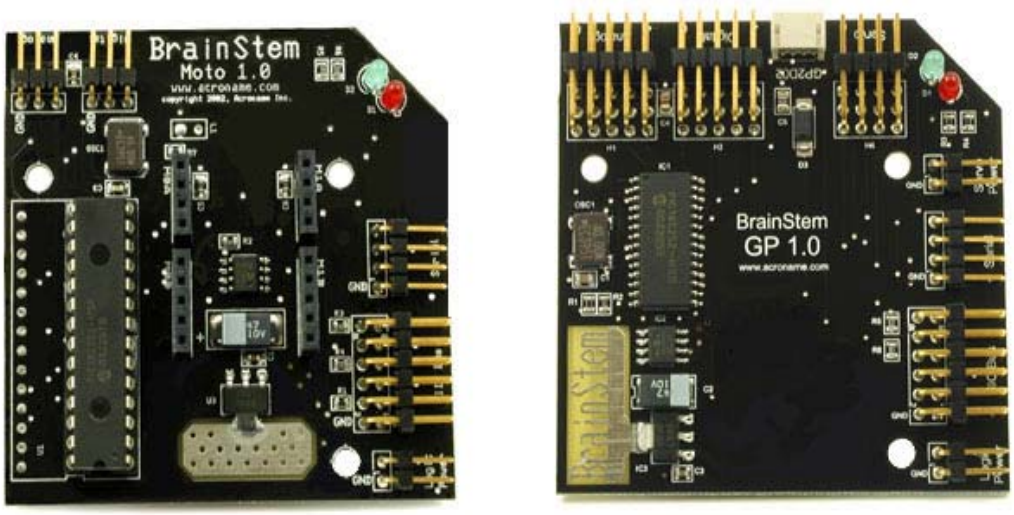

Figure 1: The Brainstem Moto (left) and GP (right) modules

General Garcia Information

In the Garcia robot, there is a Moto module networked with a GP module. The Moto is used to connect the H-bridges that control the two Maxon A-max motors and its analog ports are used to connect the front and side sensors. In Garcia, the GP module serves as the router, which receives commands from the wireless RF serial port, and it's analog ports are used to connect the rear sensors. The additional ports on these boards can be used to attach other devices such as a text-to-speech device, a compass, GPS, and a wide variety of others. Figure 2 shows all of the parts in the Garcia robot and their connections.

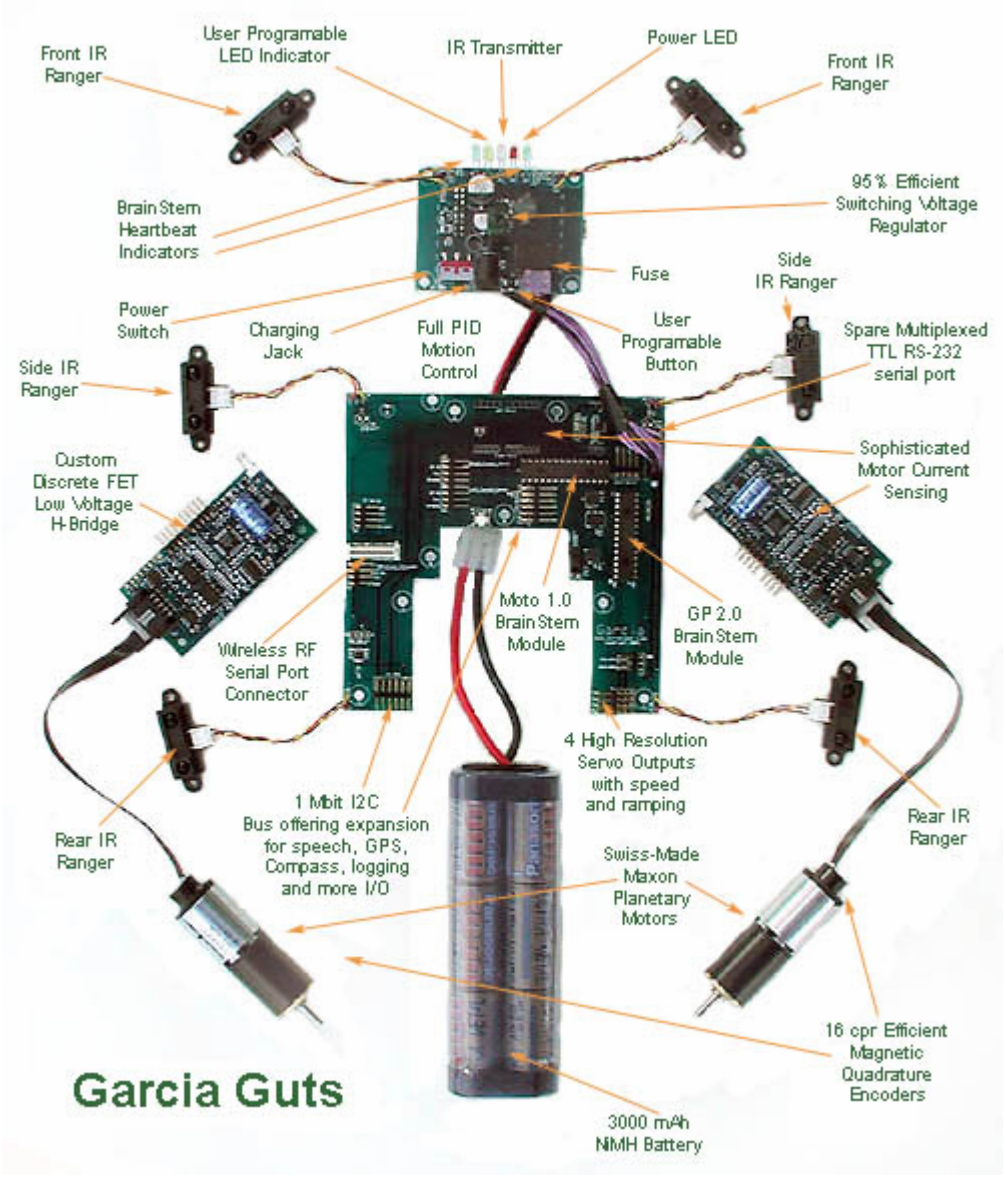

Figure 2: The insides of the Garcia robot (photo taken from Acroname.com)

Downloading Brainstem Programs

The first step toward interfacing the Palm Vx with the Brainstem is to download the necessary programs from Acroname's website and load them onto the Palm Pilot. In order to do this, log on to the website at www.acroname.com and click on the download link. You will be asked to give your name and e-mail address in order to log on. After doing this, you will be taken to the download page.

For programming the Palm Vx, we will need to click the Brainstem C Development download. This download gives us the necessary include files and source files that will be needed for our programs. Also, it contains the Palm database (pdb) files that must be installed onto the Palm Vx. These files are located in the {..}\brainstem\aBinary directory of our download where {..} refers to the directory where the download was unzipped. After locating these files, they should immediately be uploaded to the Palm Vx (as none of our programs will work without them).

Additional downloads for Palm OS on the Acroname site include the GP and console programs. The GP program is used to configure the Brainstem GP module (a handy tool if you are using a GP). The console program is used to interact with the Brainstem at the hardware level. This program will be used for configuring the Brainstem modules and the motors. These programs should also be downloaded and installed on the Palm Vx. Both programs are prc files and can be found in their respective {..}\brainstem\aBinary directories.

Connecting Garcia and the Palm

This section explains the physical connections that must be made between the Garcia and the Palm Vx. Remember, the connections explained in this section can be applied to any Brainstem-based robot. This section gives a list of necessary parts and an explanation of how to connect everything together.

Necessary Parts

To connect the Garcia to the Palm, we will need the following list of parts:

- Brainstem Palm V cable
- Serial interface connector
- Serial interface connector extender
- USB to serial adapter (if your computer does not have a serial port)
- Garcia robot

For more information about these parts including pictures, ordering information, and descriptive websites see part numbers 4-8 on the parts list located at the end of the tutorial.

Making the Connections

The first thing that we want to connect is the serial interface connector extender. This should be connected to the Brainstem in one of two places: the Moto serial port or the GP serial port. If you are comfortable removing the wireless RF serial board or if

your Garcia robot does not have this board installed (Figure 3), you should connect through the GP serial port. Otherwise, you should connect to the already exposed Moto serial port (Figure 4). Depending on which port you have chosen to connect to, there will be a unique set of steps to configure the Garcia robot to receive commands from the Palm (these are explained in more detail in the next section).

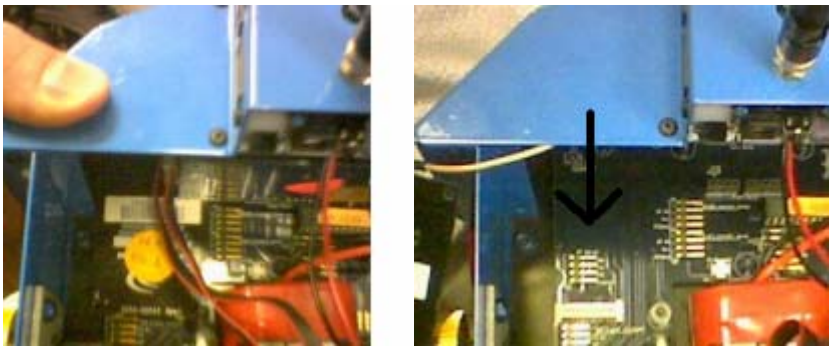

Figure 3: The Garcia robot is seen with the Wireless RF serial board connected (left-hand picture) and with the Wireless serial board disconnected (right-hand picture). If you are comfortable removing this board or if your robot does not have this board installed, you should connect the serial interface connector extender to the GP serial port (pointed to by the arrow in the right-hand picture).

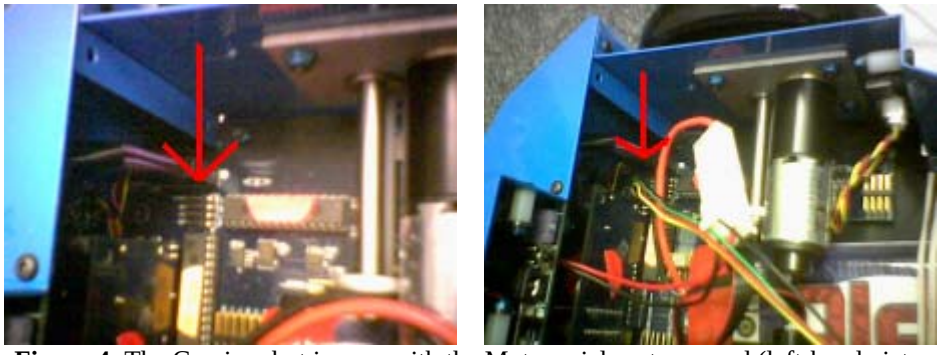

Figure 4: The Garcia robot is seen with the Moto serial port exposed (left-hand picture) and with the serial interface connector extender plugged into the Moto serial port (right-hand picture).

Once you have decided which port to use, you should go ahead and connect the cable to the serial port, ensuring that the arrow on the serial connector extender matches up with the ground pin on the serial port.

After connecting the serial interface connector extender to the Brainstem, we need to connect the serial interface connector. This plugs directly into the male end of the extender cable. Now, the Brainstem Palm V cable should be plugged into the serial interface connector and the Palm Vx should be connected into the Palm V cable. Figure 5 shows how the Garcia should look after all of the correct connections have been made.

Figure 5: Garcia and Palm connected

Now that all of the components have been connected, we need to configure the Garcia robot to receive commands from the Palm.

Configuring Garcia

We will now be making some configuration changes to the Garcia robot. In order to do this, we will need to make sure that Garcia is plugged into the Palm and that both are turned on. To make changes to the Garcia robot, we will be using the console program that we downloaded onto the Palm Vx earlier in the tutorial.

The two sections below explain how to configure your Garcia robot depending on which serial port (GP or Moto) you decided to plug the serial interface connector extender into. So, if you connected through the GP serial port, feel free to skip ahead to that section.

Connecting to the Moto serial port

The first thing that we want to do is to make sure that Garcia and the Palm Vx are both turned on and that the console program is running on the Palm. The heartbeat indicator on the console program (indicates that the connection is working) will not be blinking at this point, but we will fix that. The problem is that Garcia is initially set up to receive commands through the GP serial port but we have connected our Palm Vx to the Moto serial port. Since the GP module is initially set up to receive the commands, it is considered the router. This means that it receives all commands and sends out the ones that are intended for other hardware. Since we have connected our serial interface connector cable to the Moto module, we want to make it the new router and have it distribute commands to the GP module. The way we do this is to enter the following commands into the console program:

- 4 18 1 4 //sets the Moto module's router address to itself (4)
- 4 19 //saves the new setting to the Moto

- 2 18 1 4 //sets the GP (2) module's router address to the Moto Board (4)
- 2 19 //saves the new setting to the GP

By entering these four commands, we have turned the Moto board into the router and given it the address of the GP board so that it knows where to send GP commands. After doing this, the heartbeat indicator on the console program should be blinking. If not, we will need to change the baud rate of the controller modules with the following set of commands:

- 2 18 4 2 //sets the GP (2) module's baud rate to 9600 (designated by the last 2)
- 2 19 //saves the new setting to the GP
- 4 18 4 2 //sets the Moto (4) module's baud rate to 9600 as well
- 4 19 //saves the new setting to the Moto

Garcia is now configured to receive commands from the Palm! We can test the sensors by opening the GP program on the Palm. You will see the indicators move up and down as objects move toward and away from the sensors. The only thing left to do now is to configure the motors (skip to the section entitled: setting up the motors).

Connecting to the GP serial port

The first thing that we want to do is to make sure that Garcia and the Palm Vx are both turned on and that the console program is running on the Palm. The heartbeat indicator on the console program (indicates that the connection is working) will not be blinking at this point, but we will fix that. The problem is the GP module is, by default, set up to receive commands through the wireless RF serial port, which has a baud rate of 38400. Since we are connecting to the serial port through the interface cable (instead of wirelessly), we need to change the baud rate of the GP board to 9600. In order to do this, enter the following commands into the console program:

- 2 18 4 2 //sets the GP (2) module's baud rate to 9600 (designated by the last 2)
- 2 19 //saves the new setting to the GP
- 4 18 4 2 //sets the Moto (4) module's baud rate to 9600 as well
- 4 19 //saves the new setting to the Moto

After typing these commands, you will see the blinking heartbeat on Garcia as well in the console program. Garcia is now configured to receive commands from the Palm! We can test the sensors by opening the GP program on the Palm. You will see the indicators move up and down as objects move toward and away from the sensors. The only thing left to do now is to configure the motors.

Setting up the motors

Garcia comes equipped with two Maxon A-max motors. In order to use these motors in our programs, we will need to configure them with the correct settings. For the purposes of this tutorial, the motors will be configured to use PID encoder mode. In this

mode, motors can be set to turn a specified number of ticks before stopping. PID encoder mode is the best mode for us to use because Garcia's movements are defined using ticks. For example, to move Garcia one foot, each motor must move 3504 ticks. To turn Garcia a single degree, a motor must move 18 ticks and a 360-degree turn takes 6501 ticks. Since Garcia's movements are defined in this way, putting the motors in this mode will greatly simplify our programs. To put the motors in PID encoder mode, the following commands will need to be run from the console program:

(All settings were found to work well through trial and error testing on the motors)

- 4 63 0 0 5 0 //puts the motor on channel 0 into PID encoder mode
- 4 63 0 1 0 64 //sets the P parameter (1) to 2.0
- 4 63 0 2 0 0 //sets the I parameter (2) to 0.0
- 4 63 0 3 0 64 //sets the D parameter (3) to 2.0
- 4 63 0 4 0 0 //sets the control offset parameter (4) to 0
- 4 63 0 5 13 3500 //sets the PWM output limit parameter (5)
- 4 63 0 6 0 30 //sets the period (6) to 30
- 4 64 //saves the new settings

The next set of commands to the same configuration for the motor on channel 1:

- 4 63 1 0 5 0 //puts the motor on channel 1 into PID encoder mode
- 4 63 1 1 0 64
- 4 63 1 2 0 0
- 4 63 1 3 0 64
- 4 63 1 4 0 0
- 4 63 1 5 13 3500
- 4 63 1 6 0 30
- 4 64 //saves the new settings

These commands set both motors (channel 0 and channel 1) to be in PID encoder mode and they set all of the necessary parameters that will make the motors run smoothly.

Sensor Information

Garcia is equipped with 6 Sharp GP2D12 IR sensors. When trying to read information from these sensors (or any sensors connected to the analog pins), the most important thing to know is which analog pins correspond to which sensors. In Garcia, the sensors are connected as follows:

- The two rear sensors are on the GP board on Analog pins (numbered 0,1)
  - pin 0 is the rear left
  - pin 1 is the rear right
- Side and front sensors are on the Moto board on Analog pins (numbered 0-3)

- pin 0 is the front left sensor
- pin 1 is the front right sensor
- pin 2 is the side left sensor
- pin 3 is the side right sensor

Be sure and take note of this information, as it will be highly useful when we start writing programs to access the sensors. With our robots correctly connected and configured, we are ready to begin programming the Palm Vx.

Setup A CodeWarrior Project

Before setting up your own project, it is recommended that you download the example programs given at the end of the tutorial and familiarize yourself with some of the code and the project environment. These example programs are fully working and should be used as a guide when setting up your own projects. Or, you could simply add to the existing code to create your own robot's behaviors. If you have decided to create your own project from scratch, then there are a few specific files that you will have to include.

The main files that we will need to include in the project are the palmos_aIOGlue.c and the palmos_aStemGlue.c files. These are located in the {..}\brainstem\aBinary directory of our software download where, once again, {..} refers to the directory where the download was unzipped. These files serve as the "glue" between the Palm OS and the aIO and aStem libraries, which contain all of the necessary functions for moving the motors and reading from the sensors.

Don't forget to include all C files that contain the specific functions being used (aMotion.c, aAnalog.c, etc.). Figure 6 shows the project window for the SquareEight sample program (given at the end of the tutorial) with all necessary C files included.

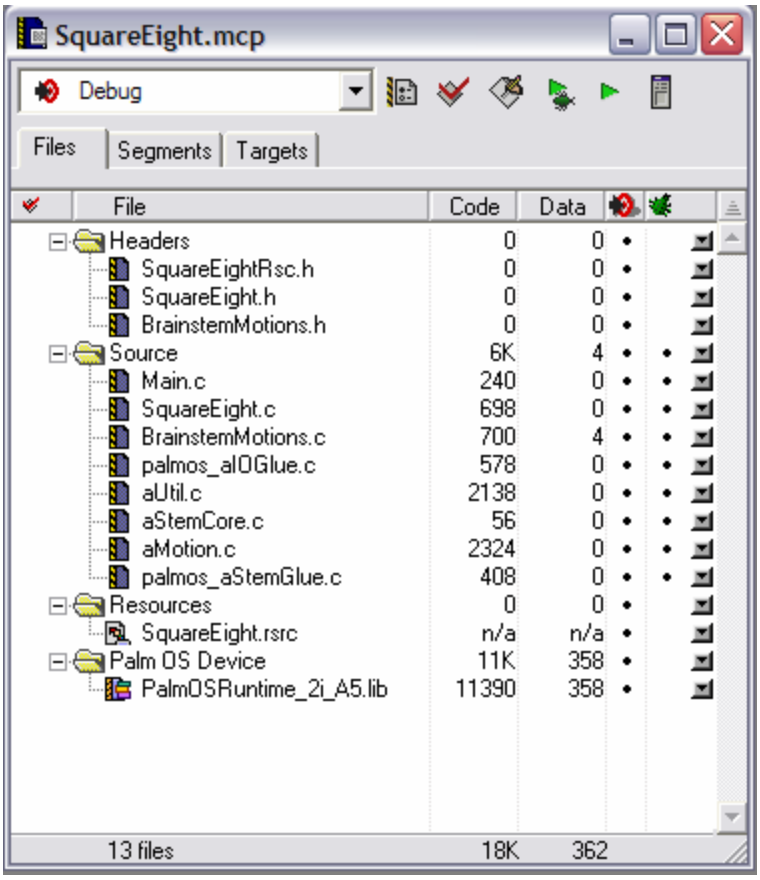

Figure 6: Complete project window for the SquareEight sample program

Also, be sure to list the access paths to all include (.h) files. To do this, click on the debug settings button of the project window (seen above) and then click on access paths in the target settings panel (seen in figure 7). You will probably only need to list the aInclude folder, but it doesn't hurt to add the other ones to the list.

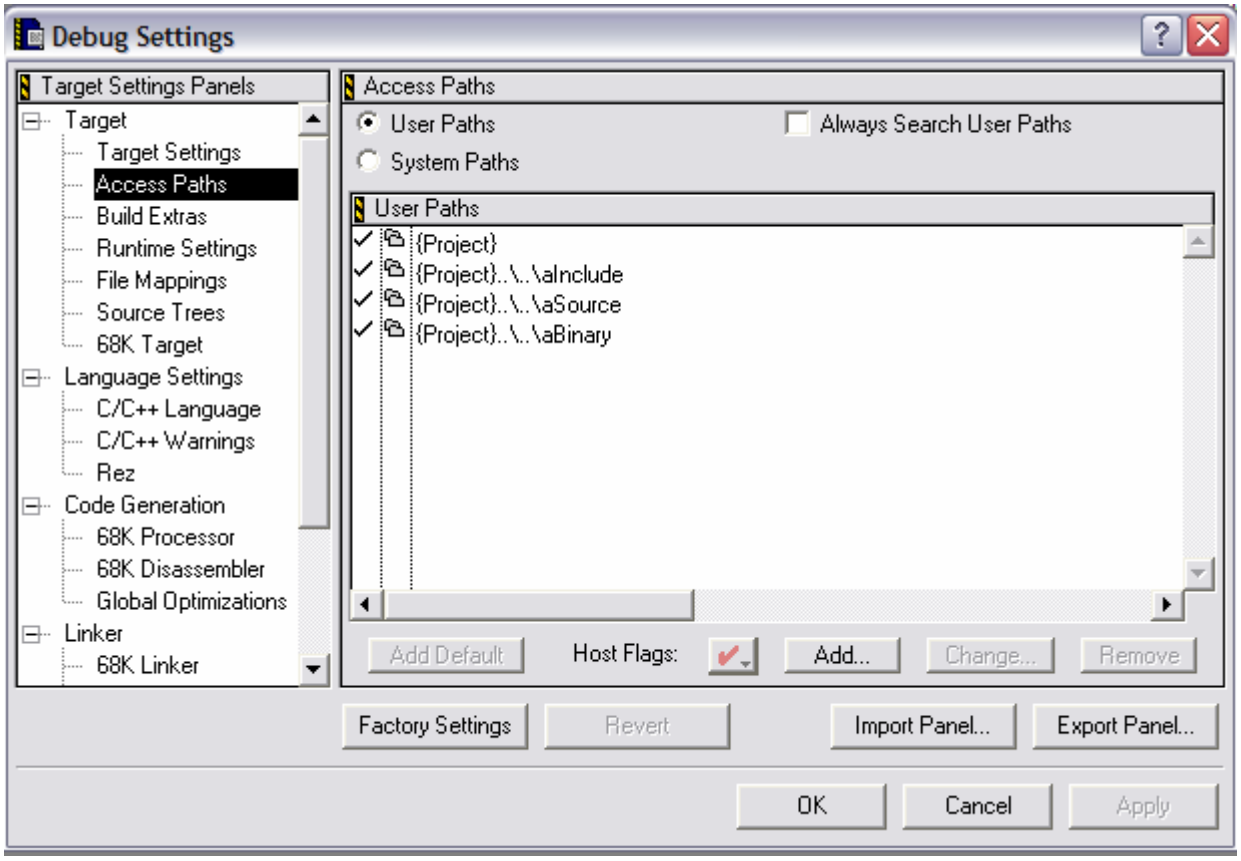

Figure 7: The Debug Settings with the necessary access paths

The final file that will need to be added to the project is the prefix file. This is a very important file that must be added in the src directory of your project and also must be included in the debug settings window (see Figure 8). It includes the PalmTypes.h file and necessary #define statements. The contents of this file should be copied exactly from the file given in the example programs; the only thing that you should change is the name of the file.

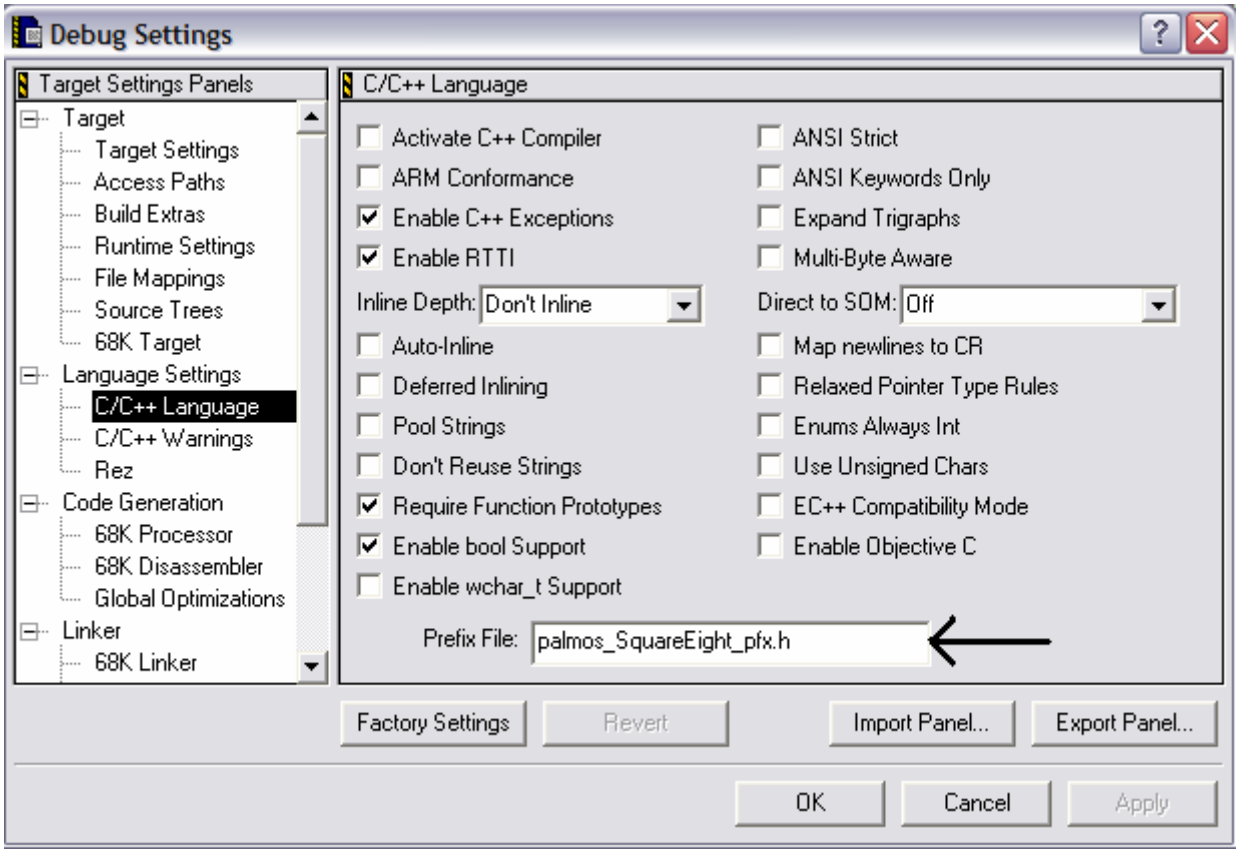

Figure 8: Adding the prefix file to the Debug Settings

Once all of these files are in place, you should have no trouble compiling and running your Palm programs.

Programming the Brainstem

Once we have the project set up, we need to know how to program the Brainstem. This will involve the use of the Brainstem C functions. A list of these functions and their descriptions is located at www.acroname.com/brainstem/ref/ref.html under the C link. Since there are too many functions to cover, we will focus on the functions to create our library references, control the motors, and retrieve sensor readings. However, you should browse through the available functions to get an idea of the full capabilities of the Brainstem.

We mentioned earlier that we have to include the Brainstem library files palmos_aIOGlue.c and palmos_aStemGlue.c in our projects. However, including them is not enough. In order to use these library files, we will have to create references to them using the aIO_GetLibRef and aStem_GetLibRef functions. These functions return a reference to the particular library file that can then be used to call the functions defined in that library file. For a specific example on the usage of both of these commands, see the init method in the BrainstemMotions.c, which is included in the example programs.

To control the motors (in PID encoder mode), we will simply need to use the aMotion_SetValue command in our programs to set the number of ticks for the motor to move. For example, the command aMotion_SetValue(stemRef, 4, 0, -1625) will tell the

motor on channel 0 to move backwards 1625 ticks (a 45 degree turn). Changing the 0 to a 1 in the above command will move the motor on channel 1, and changing the -1625 to a different number will change the number of ticks (forward or backward) moved. A good example of how to use the aMotion_SetValue command to make Garcia perform specific movements can be found in the example program called SquareEight. This program can be found on the website given in the next section.

When we want to take a reading from one of the sensors we will use the command aAnalog_ReadInt, which reads the voltage level of the specified analog pin. For example, if we want to access the information from the front left sensor, we would use the command aAnalog_ReadInt(stemRef, MOTO_MODULE, 0, &frontLeftSensor) where stemRef is our aStem library reference (created earlier), MOTO_MODULE is the module address of our Moto board (in our case this will be 4), 0 is the number of the analog pin (given earlier as front left), and frontLeftSensor is an integer that will hold the value read from the analog sensor. For more examples of how to use the aAnalog_ReadInt command to read the sensor input and make Garcia avoid an obstacle or follow an object, see the example program called AvoidObstacle. This program is located on the website given in the next section.

Now that we have a feel for the types of commands that are going to be used to control Garcia, we can take a look at some of the example programs that are provided.

Example Programs

The example programs mentioned in this tutorial can be found at: http://webster.cs.uga.edu/~barnes/GarciaPrograms.html. These programs include the square figure eight program and the avoid obstacle program. To install these programs onto your Palm Vx, simply download and unzip the programs and load the prc files.

The square figure eight program has four arrows for moving Garcia straight, left, right, and reverse. Also, there is a middle button, which makes Garcia do a square figure eight. This program uses the aMotion_SetValue command to control the motors (as described in the previous section). To view the CodeWarrior project and all of the source code, click on the SquareEight.mcp file. This shows all of the files that have been included in the project. To see how the motors are controlled, open the BrainstemMotions.c file and browse through the source code. There are many examples of the different uses of the aMotion_SetValue command.

The avoid obstacle program contains four different buttons, which correspond to different procedures that use the sensors to perform tasks. The top button is the follow button, which makes Garcia move until he “sees” something in the front sensors. After seeing the object he stops and then he will move forward if the object moves farther away or reverse if the object moves closer. The right-hand button makes Garcia move forward until he finds an object (needs to be square – preferably a box) and then move around the object in a counterclockwise fashion until he reaches the initial starting point. The neat thing about this procedure is that the square object can be of any size; Garcia uses the sensors to move around it. The left-hand button makes Garcia avoid an obstacle (should also be square). The robot will approach the obstacle and then move around it and continue in the same forward direction. The last button (at the bottom of the screen) was intended to make Garcia travel around an object and then send the size of that object back

to the program. However, this has not yet been implemented. The method signature is given in the BrainstemMotions.c file, and will need to be implemented in the future.

Conclusion

This tutorial has supplied all of the necessary information to begin programming the Palm Vx to control your Brainstem robot. However, the ideas and concepts presented in this tutorial are just an introduction to what can be done with the Brainstem modules. You should now be able to use these concepts to create your own unique behaviors for your Brainstem robot!

Parts List

1) Moto Module (http://www.acroname.com/robotics/parts/S10-MOTO-BRD.html) part number: S10-MOTO-BRD

2) GP Module (http://www.acroname.com/robotics/parts/S1-GP-BRD.html) part number: S1-GP-BRD

3) Brainstem IIC 6 Inch Cable (http://www.acroname.com/robotics/parts/C7-BS-IIC-6.html) part number: C7-BS-IIC-6

4) Brainstem Palm V cable (http://www.acroname.com/robotics/parts/S8-BS-PALMV-CBL.html) part number: S8-BS-PALMV-CBL

5) Serial interface connector (http://www.acroname.com/robotics/parts/S13-SERIAL-INT-CONN.html) part number: S13-SERIAL-INT-CONN

6) Serial interface connector extender (http://www.acroname.com/robotics/parts/C10-SER-INT-CONN-EXT.html) part number: C10-SER-INT-CONN-EXT

7) USB to serial adapter needed if you don't have a serial port (http://www.acroname.com/robotics/parts/R233-USB-ADPT.html) part number: R233-USB-ADPT

8) Garcia robot (http://www.acroname.com/garcia/garcia.html)

References

-Acroname Robotics Website (www.acroname.com)