Create Open Interface Scripts:
Using the Create Without an XBC
Jeremy Rand
Norman High School
jeremy@asofok.org

Create Open Interface Scripts: Using the Create Without an XBC

1 Introduction

When the iRobot Create [1] was introduced to Botball [2] in 2008, it added a new dimension to the game. Suddenly, teams had access to a ready-made chassis, with bump, cliff, drop, and wall sensors, powerful motors, and reasonably accurate encoders, all without needing to build anything themselves. But few teams took advantage of one feature of the Create, Open Interface Scripting, which allows a Create to work without any external hardware attached. This means that teams will be able to use their Create as a third robot, while the two XBC's [3] command the other two robots. This paper will discuss Open Interface Scripting, its use in Botball, and the limitations teams should be aware of when taking advantage of this very cool feature.

2 Prerequisites

Before proceeding with this paper, programmers should be familiar with the Interactive C (IC) programming language [4] as well as the XBC-Create libraries for IC.

3 Entering Full Mode

One of the selling points of the Create is its numerous safety features. By default, the XBC libraries put the Create in Safe Mode. This mean that whenever the cliff or drop sensors are tripped, the Create will shut off its motors and ignore any actuator commands until the XBC resends the create_safe() command. This is a nice feature if you have an XBC which can send the create_safe() command. But when your Create doesn't have an XBC attached, this can cripple your bot, since if it passes too close to an edge, it will stop with no way to recover until the round is over. If you are certain that your Create is not in any danger of dropping off the table and smashing into pieces, you can disable this safety feature by using the create_full() function. Note that if your Create gets damaged due to the safety being disabled, the damage is not covered by the iRobot warranty.

4 The Script Commands

The main feature used to make the create work without an XBC is Open Interface Scripting. Essentially, you can send up to 100 bytes of Open Interface commands (called a script) to the Create, and it will store them. You can then instruct it to play the script, and once the script starts, you won't need to have an XBC attached.

To tell the Create that you would like to store a script, send the following bytes (all numbers are in decimal): 152 *length*, where *length* is the total length in bytes of the commands you would like to comprise the script. Note that the official iRobot documentation says that *length* is the

number of commands. This is incorrect! It is the number of bytes, not commands, so don't let that confuse you. To determine the number of bytes in each command, just look up the command in createlib.ic in the IC library directory, and count the calls to serial_write_byte(). Once you have sent those two bytes, send the commands themselves, in the order in which you would like them to execute. For example, if you wanted a script to move forward at 500mm/s, and set the Power LED to red, you would use this:

```
serial_write_byte(152); // we want to store a script
serial_write_byte(9); // 9 total bytes
create_drive_straight(500); // 5 bytes
create_power_led(255, 255); // 4 bytes
```

Once you have stored the script, to play it, just send the byte 153. The script will remain in the Create's memory until the Create is power-cycled, during which time it can be played by sending the byte 153. From this point on, the 152 *length* and 153 commands will be left out for clarity, and I will just list the scripts themselves.

5 The Wait Commands

When programming the Create with an XBC, you might use sleep() or a while loop to wait for something to happen. The Create has a set of equivalents, which, although less flexible, is sufficient for many uses. These commands are the wait commands. To wait a specified time before continuing the script, send the command 155 *ds*, where *ds* is the number of deciseconds you wish to wait. To wait for a certain traveled distance (positive is forward), send 156 *mm_high mm_low*, where *mm_high* and *mm_low* are the high and low bytes of a 16-bit number of millimeters. To wait for a certain angle turned (positive is counterclockwise), send 157 *deg_high deg_low*, where *deg_high* and *deg_low* are the high and low bytes of a 16-bit number of degrees. Finally, to wait for a sensor reading, send 158 *event*, where event is one of the following:

Byte	Sensor	Byte	Sensor	Byte	Sensor
1	Wheel drop	8	Virtual wall	15	Home Base
2	Front wheel drop	9	Wall	16	Advance Button
3	Left wheel drop	10	Cliff	17	Play Button
4	Right wheel drop	11	Left Cliff	18-21	Digital Inputs 0-3
5	Bump	12	Front Left Cliff	22	OI Mode = Passive
6	Left bump	13	Front Right Cliff		
7	Right Bump	14	Right Cliff		

Table 1

To wait for the logical inverse of an event (e.g. waiting until the cliff sensors are reading false), simply replace the event with 256-*event* (which is the same as multiplying the signed byte by -1).

Note that the Create will not respond to input until a wait command finishes. This also means that if you accidentally have an incorrect sign while waiting for a distance or angle (e.g. telling it to wait for a distance of 100mm while going backward), the create script will hang. If this happens while the Create was driving forward, it will keep driving forward until someone hits

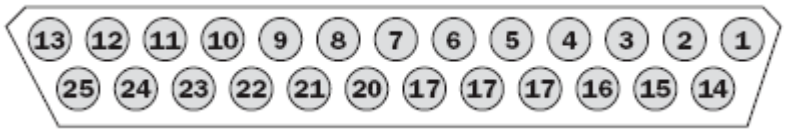
the Power switch. This can be dangerous if no one is present to stop it from driving headfirst into a wall or off the table. Be careful!

As an example, the following script drives forward slowly until it bumps into something, and then quickly backs up half a meter.

```
create_drive_straight(100); // Drive forward
serial_write_byte(158); serial_write_byte(5); // Wait for a bump
create_drive_straight(-500); // Drive backward
serial_write_byte(156); serial_write_byte(get_high_byte(-500));
      serial_write_byte(get_low_byte(-500)); // Wait for -500mm
create_stop(); // Stop
```

6 XBC / Handy Board Motors

What good is a robot without motors? Being restricted to just the Create's drive motors seems limiting. Luckily, the Roomba had brush and vacuum motors, and since they are no longer present on the Create, those outputs are now available to power up to 3 motors with XBC / Handy Board connectors. The feature is called Low-Side Drivers. Motors connect to the DB25 Cargo Bay Connector (See Figure 1 and Table 2).



Driver Number	PWR Pin	GND Pin	Max Current Draw (mA)
0	8	22	100
1	9	23	500
2	10	24	1500
XBC (for reference)			1000

Table 1

Figure 1 (Source: iRobot)

To turn the Low-Side Drivers on or off, simply use the create_low_side_drivers(int toggle2, int toggle1, int toggle0) command. You can also use PWM [5] to control their power level using the create_pwm_low_side_drivers(int pwm2, int pwm1, int pwm0) command (128 = full power).

7 Starting Lights and Time Limits

Readers who have tried out this feature will notice that the script starts running immediately after the 153 byte is sent by the XBC. But in Botball, robots have to start and stop on their own. This presents two problems which need a solution: starting and stopping at the correct time. The solution for the first is quite simple: wait for a sensor reading. The Create doesn't have a light sensor, but it does have a bump sensor. Simply have an XBC robot hit the Create's bump sensor when the game starts.

Stopping at the correct time is trickier. Create scripts don't support multitasking like IC does, so a function like shut_down_in() is out of the question. The best result I can come up with is to make certain that the script doesn't contain any wait commands which could never terminate if the bot makes an error. For example, you have a nice 45-second Create script. But you have it

wait until the left bump sensor hits a wall. What happens when the Create misses the wall by 2 inches and continues driving until it hits a wall on the right side, which won't trigger the left bump sensor? The Create will simply drive forever, not stopping when the time expires, and disqualifying your team – not a good thing when you're in Double Elimination and need to win a match. Test your bots over and over, and think about what could cause an event wait command to not terminate. Distance, angle, and time are safer than events for this reason; it's hard for a slight error to stop the wheels from spinning or cause time to stop! Therefore, events should be used with care.

8 Restarting a Finished Script

Another issue that can be annoying is that the script has to be reloaded onto the Create with an XBC prior to each time it is executed. This is quite easy to solve. You can trick a Create into looping a script by making a 153 byte (the Play Script command) the last command in the script. But you don't want your bot to loop its program in a Botball round. The solution is to also wait for the Advance button to be pressed at the beginning of the script. This way, once your script is finished, you can press the Advance button on the Create to play it again. It also has the helpful effect of not starting the script until you hit Advance, which can be useful in the event that you're setting up the bots and you don't want your Create to run away when you accidentally hit the bump sensor.

9 Limitations of Scripting

As cool as Create scripts are, they're not going to replace XBC robots anytime soon. This is because Create scripting is not a real programming language. Some of the many features missing from Create scripts are variables, math of any kind, if statements, complex loops, and multitasking. In addition, a maximum program size of 100 bytes isn't sufficient for a complex strategy. Its actuator support is shaky as well. XBC / Handy Board motors are unidirectional when used with the Create, so if you need something to be able to move in two directions, you'll need two motors, which unnecessarily complicates the engineering as well as eating up your kit pieces. RC Servos [6] are also not supported, and use of XBC-compatible sensors requires modifying the sensors' pinout (illegal in Botball).

10 Conclusion

Create scripts, despite their limitations, allow you to have a third robot on the table, which can lead to new, possibly more effective strategies. For more information, the best reference on the Create's features is the Create Open Interface documentation [7]. If you've done something cool with Create scripts that wasn't covered here, I'd like to hear about it! E-mail me at the address given at the top of this paper. Note that I can't provide full-fledged technical support for your Create.

11 References

[1] iRobot. iRobot Create. http://www.irobot.com/sp.cfm?pageid=289, 2008.
[2] KIPR. Botball. http://www.botball.org, May 2008.
[3] R. LeGrand et al. The XBC: a Modern Low-Cost Mobile Robot Controller. http://www.kipr.org/papers/xbc-iros05.pdf, July 2005.
[4] KIPR. Interactive C. http://www.botball.org/educational-resources/ic.php, March 2008.

[5] Wikipedia. Pulse Width Modulation: Power Delivery. http://en.wikipedia.org/wiki/Pulse-width_modulation#Power_delivery, May 2008.
[6] Wikipedia. Servomechanism: RC Servos. http://en.wikipedia.org/wiki/Servomechanism#RC_servos, May 2008.
[7] iRobot. iRobot Create Open Interface. http://www.irobot.com/filelibrary/create/Create%20Open%20Interface_v2.pdf, January 2007.