Advanced Create OI Scripting:
CBC Motors and Sensors, Subroutines,
Longer Scripts, and More – No CBC Required
Jeremy Rand
Norman High School
jeremy@asofok.org

Advanced Create OI Scripting: CBC Motors and Sensors, Subroutines, Longer Scripts, and More – No CBC Required

1. Introduction

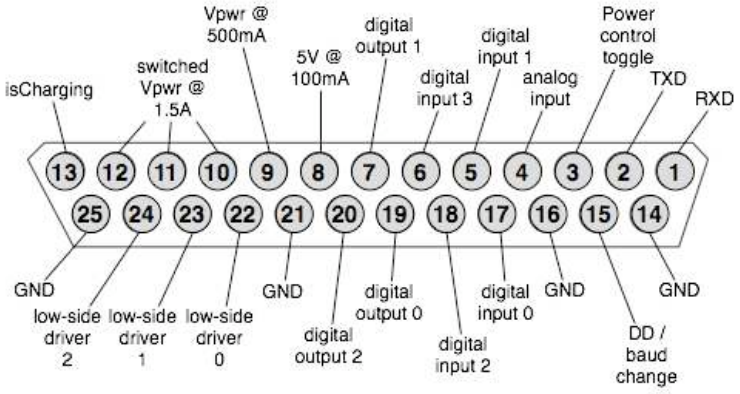
When the Create was new to Botball in 2008, only one team (mine) used Create Open Interface (OI) Scripting. Subsequent to OI Scripting being featured in a GCER 2008 paper [1], at least three other teams have utilized the technique in the 2009 season. While OI Scripts are a very simplistic language (if statements, variables, math, multitasking, loops, and many of the Create sensors are not available, and there is a 100-byte limit on the size of a script), there are a surprising number of advanced techniques for using OI Scripting, which can help push the Create to its maximum potential. This paper will address a few of these techniques.

Readers are encouraged to read my GCER 2008 paper [1] prior to reading this paper, since this paper assumes a working knowledge of Create OI Scripting.

2. The Cargo Bay Connector

Many of the techniques in this paper rely on the Create's Cargo Bay Connector (the female DB25 connector) being available. This means that the 2009 CBC-Create cable will not work for you here, since it uses the Cargo Bay Connector. Instead, you should use the 2008 XBC-Create cable, which connects to the Create's Mini-DIN connector.

This paper will frequently refer to numbered pins on the Cargo Bay Connector. For reference, here is the complete Cargo Bay Connector pinout (source: HackingRoomba.com [2]):



3. Use an XBC

The XBC IC functions for the Create are open-source, and as such it is quite easy to see what data they are sending to the Create. The CBC KISS-C functions for the Create are closed-source, and therefore are much harder to work with when talking to the Create at a low level. As a result, if you are using OI Scripting for a robot that doesn't need a CBC, I highly recommend loading the script with an XBC. For this reason, I have not tested any of this code with a CBC, nor do I plan to, since it remains perfectly legal to bring an XBC to the game table to load a script onto a Create.

4. Driving Motors (Simple Pinout)

The Roomba's vacuum motor and two brush motors are not present on the Create. Rather than remove the circuitry, iRobot decided to allow users to access these connections in the Cargo Bay Connector. These pins are called the Low-Side Drivers. Low-Side Drivers can only be switched to GND, meaning that you need to connect the other motor pin to a PWR pin. This also means that motors can only run in one direction.

Even though there are three Low-Side Driver pins, the pinout of motors with a HB/XBC/CBC connector (used in Botball) means that you can only connect to Drivers 0 and 1 (pins 22 and 23), not Driver 2 (pin 24). Connect your first motor to pins 22 and 11, and your second motor to pins 23 and 12. Each motor will have access to 500mA of power (half the current that the XBC motor driver can handle). You can control the motors (either on/off or with PWM control) using the `create_low_side_drivers()` or `create_pwm_low_side_drivers()` functions in Interactive C.

5. Driving Motors (Advanced Pinout)

If you want a third motor, or if you want more power than 500mA, you can try modifying the pinout. The only way to do this while following Botball construction rules is to use the motor extension cable. Plug one extension cable into pins 10-12, and another into pins 22-24. Once you have done that, plug each motor into one pin of each extension cable. If you want one of your motors to have access to 1500mA (50% more than an XBC), make sure that its Low-Side Driver connection is plugged into the extension cable pin that is connected to pin 24 (Low-Side Driver 2). The other two motors will still be restricted to 500mA.

6. Digital Sensors

The Create includes some digital input pins on the Cargo Bay Connector. Using this, you can attach a digital sensor (e.g. a touch sensor) to the Create, and access it in your script. The problem is that digital sensors don't simply have a data pin; they also have PWR and GND pins. And unfortunately, there is no sequence of pins on the Cargo Bay Connector that consists of PWR, GND, another pin, and a digital input. As a result, some creativity is required. The best solution I have found is to use the Create's Digital Outputs as PWR or GND as needed. Using this technique, two digital sensors can be simultaneously plugged into the Create.

The first digital sensor's PWR and GND pins (the two that are close together) can be plugged into into pins 8 and 7 (5V and Digital Output 1), and the data pin into pin 5 (Digital Input 1). Make sure that Digital Output 1 is set to 0 prior to accessing the sensor (using the `create_digital_output()` IC function), as this will allow that pin to act as a GND. The second digital sensor can be plugged into pins 20, 19, and 17 (Digital Output 2, Digital Output 0, and Digital Input 0). For this pinout, make sure that Digital Output 2 is set to 1, and Digital Output 0 is set to 0. You will be able to read the sensors using the Wait Event command, with event ID's 19 and 18 for the first and second digital sensors (see my 2008 paper for more details).

7. Light Sensors

A light sensor works well as a digital sensor when plugged into either digital sensor pinout. This can be an effective way to start a robot faster than the bump sensor I suggested in my 2008 paper. Unfortunately, the Top Hat and ET sensors require more current than the Create's Digital Output pins can provide (because they need to power an IR LED), so they cannot be used in Create OI Scripts.

8. Wall Sensor

If you've seen a Roomba in action, you've probably noticed that it can tightly follow a wall. This functionality is achieved by an infrared reflectance sensor mounted on the right side of the Roomba's bumper. The wall sensor is also present on the Create, and can be used by OI Scripts (event ID 9). Its range is a few inches (comparable to the Botball Top Hat sensor), and it responds quite well to a piece of white paper held in front of it.

9. The Reboot Command

At the end of your script, you would usually put the command `create_stop()` to stop your motors. The problem is that this takes 5 bytes of script to work. If you also need to turn off the Low-Side Drivers, you'll also need `create_low_side_drivers(0, 0, 0)`, which adds another 2 bytes. Wouldn't it be nice if you could stop all your Create bot's effectors with a single byte? Well, the undocumented command `serial_write_byte(7)` will do just that. The `7` byte tells the Create's bootloader to reboot itself, shutting down everything in the process – wheel motors, Low-Side Drivers, and everything. Note that your script will no longer be present in the Create's RAM after the reboot, so this command must be at the end of your script, and your script will not be able to loop.

10. Input Buffer Scripting

Most processors which utilize a serial port contain an input buffer, so that if the processor is busy while new data arrives on the serial port, the data doesn't get eaten. Typically, receiving a serial byte triggers a hardware interrupt, which then copies the byte into memory for the processor to read after it is no longer occupied. The Create is no exception to this principle. The Create's

serial port operates by default at 57.6kbaud. This works out to about 17.4 microseconds for each bit, or about 138.9 microseconds per byte. Since the Create's main processor loop only runs every 15ms, clearly an input buffer is used to avoid data loss. The Create's input buffer, according to my testing, is 256 bytes. Luckily, we can abuse this knowledge to store extra data in the Create's memory.

To take advantage of the input buffer, simply send the Create up to 256 bytes of commands in quick succession. This works just like a conventional script; all you have to do differently is not call the Store Script and Play Script commands. As soon as you send the data to the Create, the script will begin executing. Note that sending more than 256 bytes will cause the first bytes you send to be eaten until the script fits into 256 bytes.

11. Mixing Input Buffer Scripts with Conventional Scripts

If 256 bytes still aren't enough for your application, you can mix an Input Buffer Script with a Conventional Script. Load a Conventional Script as described in my 2008 paper, but do not send the Play Script command. Instead, wait a short time period for the script to be stored, and then load an Input Buffer Script. Make the last command in the Input Buffer Script the Play Script command. The Input Buffer Script will execute, followed by the Conventional Script. Since you have 255 bytes of Input Buffer Script (one byte is already taken to call the Conventional Script) plus 100 bytes of Conventional Script to work with, the effective maximum size of your script is 355 bytes. This is enough for 43 encoder-measured circular arc moves – 31 from the Input Buffer Script plus 12 from the Conventional Script.

12. `while(1)` Style Looping of the End of a Script

If you'd like your script to run one behavior once, and then loop another behavior forever, combining Input Buffer Scripting with Conventional Scripting as described above is an effective way to accomplish this. Make the Input Buffer Script consist of the run-once behavior, and make the Conventional Script contain the looped behavior. Make the last command in your Conventional Script the Play Script command

13. Subroutines

Maybe your script is longer than the 355-byte limit, but it's very repetitive. If removing the repetition would save more bytes than are used by the Conventional Script portion, you may wish to use the Conventional Script as a subroutine. Find a single sequence of functions that occur frequently in your script, and store them as a Conventional Script. Do not include the Play Script command at the end of either the Input Buffer Script or the Conventional Script. You can then use the Input Buffer Script as your main program, and use the Play Script command anywhere within the Input Buffer Script to call the Subroutine Script.

I have found that when the Play Script command is called from within an Input Buffer Script, the command immediately after the Play Script command is executed before the Subroutine Script. I'm not entirely sure why this is, but an easy workaround is to follow every Play Script command

with a Wait command that will return immediately, e.g. a Wait Time command with a time of 0 deciseconds. Also, remember that you can only have one Subroutine Script using this technique.

14. Multiple Subroutines

If you need to have more than one Subroutine Script, this is possible. Remember that the Store Script command is itself a Create OI command, and you can therefore embed it in an Input Buffer Script. When you need to redefine the Subroutine Script, insert a Store Script command in your Input Buffer Script, followed by the new contents of the Subroutine Script. This can be effective in compressing your script, since the only overhead of redefining the Subroutine Script compared with executing the new commands once is only 2 bytes (the Store Script and Play Script commands), but future invocation of the Subroutine Script will only take 1 byte per call (compared with, for example, 5 bytes to set wheel motor speeds), saving large amounts of space.

15. `for` Style Looping of Part of a Script

If you want to simply loop a subset of your script a predetermined number of times, you can do this similarly to the Subroutine Script technique above. Make the contents of the loop your Subroutine Script, and in the location in your script where you would like to execute a loop, insert a number of Play Script commands equal to the number of times you would like to loop the Subroutine Script.

16. Conclusion

Create OI Scripting is surprisingly powerful. It is my hope that teams reading this paper will consider using OI Scripting in their strategies next year. If you'd like to contact me, I frequent the Botballer's Chat and Botball Forums on the Botball Community website [3]. Please don't e-mail me with questions unless you've tried but haven't had luck contacting me on one of those, or if it's an emergency.

References

[1] J. Rand. Create Open Interface Scripts: Using the Create Without an XBC. *Proceedings of the 2008 Global Conference on Educational Robotics*, 2008.
[2] T. E. Kurt. iRobot Create Robot Review. Hacking Roomba. http://hackingroomba.com/projects/irobot-create-robot-review/ , 2007.
[3] Botball Youth Advisory Council. Botball Community. http://community.botball.org/ , 2009.