Jade Robot™ syscall API List

Myke Predko

Last Updated: December 12, 2014

License and Warranty

This document and code was written for the Mimetics Jade Robot™ and follow on products.

This document and code is considered Mimetics Proprietary and may not be released outside of Mimetics except by permission of Mimetics, Inc.

Software Compatibility

The script language described in this document was written to be supported by:
Robot Software Release 39 or later, except where noted
Robot Tokenizer Version 0.11.10 or later
_start.s defined as _Header4.script or later

Conventions, Options and Selections

Example code will be put in monospace font like:

```
A = B + C
```

In the language definition, there are a number of instances where there are optional parameters or multiple parameters for the same task. To make these situations more obvious, the following convention is used:

[] – Optional parameter
| - One parameter or another
... – Previous parameter can be repeated
<none> - indicates that nothing is a possible option

Overview

The primary method of accessing the Jade Robot™ hardware features is the "syscall" API interface which takes the form:

```
stringReturn = syscall(integerAPI, stringArgument);
```

Note that data passed to the specific API function as well as returned from it are strings. Integers can be converted to an argument or converted from an argument using the "itos" or "stoi" like:

```
integerReturn = stoi(syscall(integerAPI, itos(integerArgument)));
```

For more information regarding programming the Jade Robot™, see the "Script Language Outline" document.

Errors and Warnings

As following the convention of the Jade Robot script language, there are no warnings that will be posted by the tokenizer or script engine.

Errors will cause execution of the current .script (.s) file to stop with the error message displayed on the Jade Robot's display.

APIs

getsoftwareversion – 0	
Return string with Jade Robot Software Version. Each major block of the version is represented in the string with the software release number being the first two digits and normally used to refer to the release version of the software running on the Jade Robot.	

`//  Read software version information and save release version number`
`str releaseVersion = syscall(getSoftwareVersion, "");`
`    releaseVersion = syscall(toString, "2:" + releaseVersion);`	

Passed Argument: None	**Returned Data:** String with version information of the major blocks of software used in the Jade Robot
Error: None	

playwavefile – 1	
Play the specified .wav file over the Jade Robot's speaker. The wave file is an 8 bit, 11,025 samples per second and downloaded into the Jade Robot's file system using the Jade Support tool. Play commands are serialized, which means that the current .wav file will finish before any additional .wav files are played.	
`syscall(playwavefile, "fileName.w");`	
Passed Argument: Filename of wave file to play	**Returned Data:** None
Error: File not found (not loaded into Jade Robot's file system)	

setvolume – 2	
Set the output volume of the Jade Robot. Valid operating values are "0" (Off) to "3" (Full Volume).	
`// Set a medium volume for the robot's output` `syscall(setvolume, itos(2));`	
Passed Argument: Integer Volume level from 0 to 3 as a String	**Returned Data:** None
Error: Invalid volume specified	

getvolume – 3	
Return the current speaker volume level as a string from "0" (Off) to "3" (Full Volume).	
`// Get the integer value of the current Jade Robot's speaker volume` `int currentVolume = stoi(syscall(getvolume, ""));`	
Passed Argument: None	**Returned Data:** String "0" to "3"
Error: None	

getbutton – 4	
Return the state of the five buttons as a string of "udlre" – when released, each button returns lower case, when pressed upper case. "u"/"U" – Up Button, "d"/"D" – Down Button, "l"/"L" – Left Button, "r"/"R" – Right Button, "e"/"E" – "Enter" or "Red" Button	
`while ("udlre" == syscall(getbutton, "")); // Wait for key press`	
Passed Argument: None	**Returned Data:** five character string with upper/lower for button state
Error: None	

getpanelstatus – 5	
Return a numeric string with the Active Panel Driver Status: "0" – Ready, no panel loaded "1" – Error encountered with load or execution "2" – Panel is active "3" – "Enter"/Red button pressed and panel driver has stopped **NOTE:** It is recommended that getActivePanelStatus is used instead of this API due to its more complete information.	
`int panelStatus = stoi(syscall(getpanelstatus, ""));`	
Passed Argument: None	**Returned Data:** Numeric String, Defined above
Error: None	

cleardisplay – 6	
Clear the OLED display	
`syscall(cleardisplay, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

setcursorstartx – 7	
Specify the initial "X" position for graphic and Text draw commands. Note that range is from 0 to 127 for a full OLED display (anything outside this range will result in the graphics being clipped).	
`syscall(setcursorstartx, itos(xPosition));`	
Passed Argument: String from integer X position	**Returned Data:** None
Error: None	

setcursorstarty – 8	
Specify the initial "Y" position for graphic and Text draw commands. Note that range is from 0 to 63 (anything outside this range will result in the graphics being clipped).	
`syscall(setcursorstarty, itos(yPosition));`	
Passed Argument: String from integer Y position	**Returned Data:** None
Error: None	

setcursorendx – 9	
Specify the final "X" position for graphic lines and rectangles. Note that range is from 0 to 127 (anything outside this range will result in the graphics being clipped).	
`syscall(setcursorendx, itos(xPosition));`	
Passed Argument: String with X position	**Returned Data:** None
Error: None	

setcursorendy – 10	
Specify the final "Y" position for graphic lines and rectangles. Note that range is from 0 to 63 (anything outside this range will result in the graphics being clipped).	
`syscall(setcursorendy, itos(yPosition));`	
Passed Argument: String with Y position	**Returned Data:** None
Error: None	

getcursorstartx – 11	
Return the current initial "X" position for graphic and Text draw commands. Note that range is from 0 to 127 for a full OLED display (anything outside this range will result in the graphics being clipped).	
`int currentX = stoi(syscall(getcursorstartx, ""));`	
Passed Argument: None	**Returned Data:** String with current "X" position
Error: None	

getcursorstarty – 12	
Return the current initial "Y" position for graphic and Text draw commands. Note that range is from 0 to 63 for a full OLED display (anything outside this range will result in the graphics being clipped).	
`int currentY = stoi(syscall(getcursorstarty, ""));`	
Passed Argument: None	**Returned Data:** String with current "Y" position
Error: None	

setcolor – 13	
Set the colour for a graphic or text object draw. "1" means pixels drawn to are lit, "0" means they remain dark.	
`syscall(setcolor, itos(1)); // Make the drawing color "on"`	
Passed Argument: String with "1" (on) or "0" (off)	**Returned Data:** None
Error: Invalid String passed	

getcolor – 14	
Return the current active color. "1" means pixels drawn to are lit, "0" means they remain dark.	
`int currentColor = stoi(syscall(getcolor, "″));`	
Passed Argument: None	**Returned Data:** String with "1" (on) or "0" (off)
Error: None	

drawdot – 15	
Draw dot at the current "setcursorstartx/y" position with the current "setcolor".	
`syscall(drawdot, "″);`	
Passed Argument: None	**Returned Data:** None
Error: None	

getdot – 16	
Return the color ("0" for black, "1" for lit) of the pixel at the current "setcursorx/y" position. If the "setcursorstartx/y" position is outside the range of the OLED, "0" is returned.	
`int dotValue = stoi(syscall(getdot, "″));`	
Passed Argument: None	**Returned Data:** Integer String with "0" or "1"
Error: None	

drawtext – 17	
Draw 5x7 text at the current "setcursorstartx/y" position with the "setcolor". If any portion of the text lies outside the visible range of the OELD, text will be clipped. Note that the actual text size is 6x8 as each character has an empty top row of pixels and empty right column of pixels.	
`syscall(drawtext, "Write this text string″);`	
Passed Argument: String to text to be written to the OLED	**Returned Data:** None
Error: None	

drawbitmap – 18	
Draw bitmap file (ending in ".b") on the OLED with the top left hand corner at the current "setcursorstartx/y" position. If any portion of the bitmap is outside the visible range of the OLED, it will be clipped.	
`syscall(drawbitmap, "sample.b″);`	
Passed Argument: Filename of bitmap + ".b"	**Returned Data:** None
Error: Bitmap not found in file system	

drawline – 19	
Draw a line from "setcursorstartx/y" to "setcursorendx/y" with "setcolor". If any portion of the line is outside the visible range of the OLED, it will be clipped.	
`syscall(drawLine, "″);`	
Passed Argument: None	**Returned Data:** None
Error: None	

fillrectangle – 20	
Draw a rectangle of "setcolor" from "setcursorstartx/y" to "setcursorendx/y" with "setcolor". If any portion of the rectangle is outside the visible range of the OLED, it will be clipped.	
`syscall(fillrectangle, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

stopupdate – 21	
Do not carry out any screen updates until "doupdate". Normally, each OLED update API will cause a redraw of the OLED, but executing "stopupdate" before doing any OLED update APIs when "doupdate" is executed, all the updates will be displayed without the need for multiple OLED redraws (which take roughly 25ms).	
`syscall(stopupdate, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

doupdate – 22	
Draw all pending updates to the OLED that were stopped due to "stopupdate" API.	
`syscall(doupdate, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

drawsmalltext – 23	
Draw 3x5 text at the current "setcursorstartx/y" position with the "setcolor". If any portion of the text lies outside the visible range of the OELD, text will be clipped. Note that the actual text size is 6 pixels high as each character has an empty top row of pixels. The width is proportional and based on the character and ranges from 1 to 5 wide with an empty column of pixels to the right.	
`syscall(drawsmalltext, "Write this small text string");`	
Passed Argument: String to text to be written to the OLED	**Returned Data:** None
Error: None	

drawoutlinebitmap – 24	
Draw bitmap file (ending in ".b") on the OLED with the top left hand corner at the current "setcursorstartx/y" position with a lighted square outline around the outside row of pixels. If any portion of the bitmap is outside the visible range of the OLED, it will be clipped.	
`syscall(drawoutlinebitmap, "sample.b");`	
Passed Argument: Filename of bitmap + ".b"	**Returned Data:** None
Error: Bitmap not found in file system	

drawreversebitmap – 25	
Draw bitmap file (ending in ".b") on the OLED with the top left hand corner at the current "setcursorstartX/Y" position reversed (black pixels lighted and visa-versa). If any portion of the bitmap is outside the visible range of the OLED, it will be clipped.	
`syscall(drawreversebitmap, "sample.b");`	
Passed Argument: Filename of bitmap + ".b"	**Returned Data:** None
Error: Bitmap not found in file system	

drawroundedoutlinebitmap – 26	
Draw bitmap file (ending in ".b") on the OLED with the top left hand corner at the current "setcursorstartx/y" position with a rounded, lighted square outline around the outside row of pixels. If any portion of the bitmap is outside the visible range of the OLED, it will be clipped.	
`syscall(drawroundedoutlinebitmap, "sample.b");`	
Passed Argument: Filename of bitmap + ".b"	**Returned Data:** None
Error: Bitmap not found in file system	

drawroundedreversebitmap – 27	
Draw bitmap file (ending in ".b") on the OLED with the top left hand corner at the current "setcursorstartx/y" position as reversed with a rounded, lighted square outline around the outside row of pixels. If any portion of the bitmap is outside the visible range of the OLED, it will be clipped.	
`syscall(drawroundedreversebitmap, "sample.b");`	
Passed Argument: Filename of bitmap + ".b"	**Returned Data:** None
Error: Bitmap not found in file system	

getusbstatus – 28	
Return current USB status	
`syscall(getusbstatus, "");`	
Passed Argument: None	**Returned Data:** "usbPluggedIn\r" or "NOusbPluggedIn\r"
Error: None	

loadrunpanelfile – 29	
Load panel file and start executing	
`syscall(loadrunpanelfile, "anypanel.p"`	
Passed Argument: filename of panel (".p") file	**Returned Data:** none
Error: File not found or invalid	

getactivepanelstatus – 30	
Return the active panel information in the format "Panel Driver Status:Current control". The return values are: "READY" – No panel loaded "ERROR:control" – Error encountered with control "RUN:control" – Panel Driver is active with the specified control. If no controls on the panel, nothing after the colon(:) "END:control" – User has pressed "Enter"/Red button and the Panel Driver has stopped at the specified control. If no controls on the panel, nothing after the colon(:)	
`str panelstatus = syscall(getactivepanelstatus, "");`	
Passed Argument: None	**Returned Data:** String in format shown above
Error: None	

clearoledchangeflag – 31	
The "OLEDChangeFlag" will be set when a program changes the OLED or the user presses a button, moving the active control. Clearing the flag sets a place where the updated OLED can be checked – this flag is primarily used in system software debugging.	
`syscall(clearoledchangeflag, "");`	
Passed Argument: None	**Returned Data: None**
Error: None	

getoledchangeflag – 32	
Return the current "OLEDChangeFlag" state. The OLEDChangeFlag will be set when a program changes the OLED or the user presses a button, moving the active control. Clearing the flag sets a place where the updated OLED can be checked – this flag is primarily used in system software debugging.	
`int OLEDChangeState = stoi(syscall(clearoledchangeflag, ""));`	
Passed Argument: None	**Returned Data:** "1" OLED has been updated/"0" OLED not updated
Error: None	

fileavail – 33	
Return the amount of file space that is available.	
`int fileSpaceFree = stoi(syscall(fileavail, ""));`	
Passed Argument: None	**Returned Data:** Number of bytes free in the file system
Error: None	

varavail – 34	
Return the amount of variable space that is currently available. This API is primarily used during system test – the variable system automatically performs garbage collection on the unused variable space so the need for monitoring variable space is minimized	
`int fileSpaceFree = stoi(syscall(fileavail, ""));`	
Passed Argument: None	**Returned Data:** Number of bytes free in the variable stack
Error: None	

filegarbage – 35	
Perform a garbage collection operation on the file system. **WARNING:** The only safe script file for performing a file system garbage collection is at the bottom level and the first file (of any type) in the system. For this reason, it is highly recommended that file system garbage collection is never used.	
`syscall(filegarbage, "");`	
Passed Argument: None	**Returned Data:** None
Error: None – although if current .script or .function file is relocated due to the garbage collection, execution will not properly return.	

vargarbage – 36	
Perform a garbage collection on the variable heap. This operation is generally not required as the script engine performs this action automatically. There may be some instances where the user will want to execute this function if the following code's timing cannot tolerate the time required for a system garbage collection.	
`sycall(vargarbage, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

delay – 37	
Delay execution a set number of milliseconds. It is recommended that delays are less than 500ms to prevent the application from being unresponsive. During debug, when single stepping, the delay is ignored.	
`syscall(delay, itos(125));  //  Delay 125ms`	
Passed Argument: Decimal ASCII string with the time delay	**Returned Data:** None
Error: Not a Decimal ASCII string passed to API	

setsynch – 38	
Enable at set the "synch" delay. The "synch" delay is used to ensure that repeated operations start at the same point, ensuring the application executes repeatedly and simplifying the code.	

```
syscall(setsynch, itos(500));  //  Synch every 500ms

do
{
    syscall(waitforsynch, "");
    //  Execute synch'd code
}
```

Passed Argument: Decimal ASCII string with synch delay	**Returned Data:** None
Error: Not a Decimal ASCII string passed to API	

stopsynch – 39	
Stop the synch timer.	
`syscall(stopsynch, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

waitforsynch – 40	
Halt execution until the synch timer as completed. Execution resumes at this point.	
`Syscall(waitforsynch, "");`	
Passed Argument: None	**Returned Data:** None
Error: Synch timer not started with "setsynch" API/synch timer has ended while execution still active	

motorleftset – 41	
Set the direction and power level of the left motor from "-100" to "100".	
`syscall(motorleftset, itos(-50));  //  Left Motor 50% backwards`	
Passed Argument: Signed Decimal String from -100 to 100	**Returned Data:** None
Error: Invalid Decimal String	

motorrightset – 42	
Set the direction and power level of the right motor from "-100" to "100".	
`syscall(motorrighttset, itos(75));  //  Right Motor 75% forwards`	
Passed Argument: Signed Decimal String from -100 to 100	**Returned Data:** None
Error: Invalid Decimal String	

motorleftget – 43	
Return the current operating level of the left motor (from "-100" to "100").	
`int leftMotor = stoi(syscall(motorleftget, ""));`	
Passed Argument: None	**Returned Data:** Signed Decimal ASCII String
Error: None	

motorrightget – 44	
Return the current operating level of the right motor (from "-100" to "100").	
`int rightMotor = stoi(syscall(motorrightget, ""));`	
Passed Argument: None	**Returned Data:** Signed Decimal ASCII String
Error: None	

servooff – 45	
Turn servo drivers off/put in a low current state. The servo PWMs are also disabled.	
`syscall(servooff, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

gripservoset – 46	
Set the gripper value from 0 – 100 using a Decimal ASCII string. This function is repeated as "grippersetpos".	
`syscall(gripservoset, itos(60));  //  Open the Gripper 60%`	
Passed Argument: Decimal to ASCII string	**Returned Data:** None
Error: Invalid Decimal ASCII string	

elevservoset – 47	
Set the elevation value from 0 – 100 using a decimal ASCII string. This function is repeated as "grippersetelev".	
`syscall(elevservoset, itos(30));  //  Raise the Gripper 30%`	
Passed Argument: Decimal to ASCII string	**Returned Data:** None
Error: Decimal to ASCII string	

gripservoget – 48	
Return the current servo gripper value (from 0 to 100).	
`int gripperPos = stoi(syscall(gripservoget, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

elevservoget – 49	
Return the current elevation servo gripper value (from 0 to 100).	
`int elevPos = stoi(syscall(elevservoget, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

battread – 50	
Return the current battery level as a Decimal ASCII string in the rage of 0 -100. Value is roughly proportional, but it should be noted the robot has very limited battery life if the value returned is 30 or less.	
`int batteryLevel = stoi(syscall(battread, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

leftlight – 51	
Read the left front light sensor with "0" to "100" returned.	
`int leftLightValue = stoi(syscall(leftlight, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

rightlight – 52	
Read the right front light sensor with "0" to "100" returned.	
`int rightLightValue = stoi(syscall(rightlight, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

rearlight – 53	
Read the rear front light sensor with "0" to "100" returned.	
`int rearLightValue = stoi(syscall(rearlight, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

leftline – 54	
Turn on the appropriate spectrometer LED to illuminate the bottom of the robot and read the left line sensor on the bottom of the robot with "0" to "100" returned. **NOTE:** No LEDs are turned on by this command.	
`int leftLineValue = stoi(syscall(leftline, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

rightline – 55

Turn on the appropriate spectrometer LED to illuminate the bottom of the robot and read the right line sensor on the bottom of the robot with "0" to "100" returned. **NOTE:** No LEDs are turned on by this command.

```
int rightLineValue = stoi(syscall(rightline, ""));
```

Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

centerline – 56

Turn on the appropriate spectrometer LED to illuminate the bottom of the robot and read the center line sensor on the bottom of the robot with "0" to "100" returned. **NOTE:** No LEDs are turned on by this command.

```
int centerLineValue = stoi(syscall(centerline, ""));
```

Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

spectroread – 57

Perform a spectrometer read operation. Seven values are returned in an ASCII string with dashes ("-" ASCII 0x2D) in between them. Each value is a Decimal ASCII string from "0"-"100" and represents:

1. No LED illumination to gauge light level underneath the robot
2. Blue (470nm) response
3. Green (525nm) response
4. Yellow-Green (574nm) response
5. Red (660nm) response
6. 880nm Infrared response
7. 940 Infrared response

```
str spectrometerRead = syscall(spectroread, "");
```

Passed Argument: None	**Returned Data:** ASCII String Described above
Error: None	

dir – 58

Search for a specific file or a number of files meeting some criteria. The passed argument is in the form "fileName.ext" which can have the '?' (any character in this position) and '*' (any character from this position and following) wildcards for both the fileName and extension. "dir" always returns an empty string – "dirnext" is required to retrieve the only or first filename.

```
//  Look for _start.s in the file system.
    syscall(dir, "_start.s");
    if ("" != syscall(dirnext, ""))
    {
        //  Have the file "_start.s"
    }
```

Passed Argument: ASCII string with filename + wildcards	**Returned Data:** None
Error: Invalid fileName.ext format specified	

dirnext – 59	
Retrieve the next file that meets the fileName.ext specified by a "dir" command. When all the files that meet the conditions set by "fileName.ext", "dirnext" returns an empty string.	

`//  List all the script (.s) files loaded in robot:`
`str currentFileName;`
`  syscall(dir, "*.s");`
`  while ("" != (currentFileName = syscall(dirnext, "")))`
`  {`
`  //  Process "currentFileName"`
`  }`	

Passed Argument: None	**Returned Data:** fileName.ext or None
Error: No previous "dir" command	

exec – 60	
Start executing another .script (.s) file.	
`syscall(exec, "_start.s");`	
Passed Argument: fileName.s of script to execute	**Returned Data:** None
Error: Invalid fileName.Ext specified/fileName.s not found	

done – 61	
End current .script (.s). If current script was initiated by "exec" then execution will return to that .script. Else, all execution stops. This API executes the same way as if the end of the mainline execution of a script is encountered.	
`syscall(done, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

bton – 62	
Turn power on Bluetooth module.	
`syscall(bton, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

btoff – 63	
Turn power off to Bluetooth module.	
`syscall(btoff, "");`	
Passed Argument: None	**Returned Data:** None
Error: Error if attempt made to shut off Bluetooth power while it is connected to a PC	

btstatus – 64	
Return current status of Bluetooth module. A single ASCII Decimal is returned consisting of: "2" – Bluetooth Power Off "3" – Bluetooth Power On and Not Connected "4" – Bluetooth Power On and Connected	
`int btStatus = syscall(btstatus, "");`	
Passed Argument: None	**Returned Data:** ASCII Decimal String
Error: None	

gripperclose – 65	
Close gripper by 10 units unless gripper is closed (at value “0”).	
`syscall(gripperclose, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

gripperopen – 66	
Open gripper by 10 units unless gripper is fully open (at value “100”).	
`syscall(gripperopen, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

gripperup – 67	
Move gripper up by 10 units until it is at the fully up position (at value “100”).	
`syscall(gripperup, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

gripperdown – 68	
Move gripper down by 10 units until it is at the fully down position (at value “0”).	
`syscall(gripperdown, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

grippersetelev – 69	
Set the gripper elevation from “0” (fully down) to “100” (fully up).	
`syscall(grippersetelev, itos(55));`	
Passed Argument: Decimal String from “0” to “100”	**Returned Data:** None
Error: None	

grippergetelev – 70	
Return the current gripper elevation from “0” (fully down) to “100” (fully up).	
`int gripperElev = stoi(syscall(grippersetelev, ""));`	
Passed Argument: None	**Returned Data:** Decimal String from “0” to “100”
Error: None	

grippersetpos – 71	
Set the open/close position of the gripper from “0” (fully closed) to “100” (fully open).	
`syscall(grippersetpos, itos(25));`	
Passed Argument: Decimal ASCII String from “0” to “100”	**Returned Data:** None
Error: None	

grippergetpos – 72	
Return the open/close position of the gripper from “0” (fully closed) to “100” (fully open).	
`int gripperPos = stoi(syscall(grippergetpos, "″));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

grippercenter – 73	
Move the gripper and its elevation to the middle position (“50”).	
`syscall(grippercenter, "″);`	
Passed Argument: None	**Returned Data:** None
Error: None	

gripperon – 74	
Turn power on to gripper servos.	
`syscall(gripperon, "″);`	
Passed Argument: None	**Returned Data:** None
Error: None	

gripperoff – 75	
Turn power off to gripper servos.	
`syscall(gripperoff, "″);`	
Passed Argument: None	**Returned Data:** None
Error: None	

setled – 76	
Specify the state of the LEDs on the top PCB of the Jade Robot. LEDs are numbered as: ‘r’ – Data Receive LED, inside Blue LED. ‘t’ – Data Transmit LED, outside Blue LED. ‘0’ – Green LED at the 1:00 Position ‘1’ – Green LED at the 3:00 Position ‘2’ – Green LED at the 4:00 Position ‘3’ – Green LED at the 8:00 Position ‘4’ – Green LED at the 9:00 Position ‘5’ – Green LED at the 10:00 Position Command sent is “L-C” where “L” is the LED, one of the eight listed above and “C” is the command: ‘0’ – LED off ‘1’ – LED on ‘f’ – Flash LED **NOTE:** the ‘r’ and ‘t’ LEDs are used by the Jade Robot for indicating data transfers. They are not recommended for use in applications as their operation may result in user confusion when the Jade Robot is connected to a host PC via Bluetooth or USB.	
`syscall(settled, "1-f"); // Flash the rightmost LED`	
Passed Argument: 3 Character ASCII String Defined above	**Returned Data:** None
Error: Invalid LED or Command specified	

irleftfront – 77	
Return the distance to the left front object sensor (“0” being in contact, “100” nothing detected)	
`int leftFrontObject = stoi(syscall(irleftfront, "″));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

irRightfront – 78	
Return the distance to the right front object sensor (“0” being in contact, “100” nothing detected)	
`int rightFrontObject = stoi(syscall(irrightfront, "″));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

irleftside – 79	
Return the distance to the left side object sensor (“0” being in contact, “100” nothing detected)	
`int leftSideObject = stoi(syscall(irsidefront, "″));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

irrightside – 80	
Return the distance to the right side object sensor (“0” being in contact, “100” nothing detected)	
`int rightsideObject = stoi(syscall(irrightside, "″));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

ircenterrear – 81	
Return the distance to the center rear object sensor (“0” being in contact, “100” nothing detected)	
`int centerRearObject = stoi(syscall(ircenterrear, "″));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

random – 82	
Return a decimal ASCII string with a random number from “0” to “100”.	
`int randomNumber = stoi(syscall(random, "″));`	
Passed Argument: none	**Returned Data:** Decimal ASCII string
Error: None	

geticon – 83	
Get “Icon” bitmap associated with a .script file. The icon is specified using the “meta” statement described in the Jade Robot Script Language Outline. **NOTE:** If no meta “icon” statement is present in the .script file, the tokenizer automatically inserts the “_2script.b” file as the .script file’s icon.	
`str fileIcon = syscall(geticon, "_start.s″);`	
Passed Argument: None	**Returned Data:** ASCII Filename string
Error: File not found/.script (.s) file icon not requested	

error – 84	
Force an error in the current script and display a message on the Jade Robot's display. The current .script file ends and execution returns to the caller (if appropriate).	
`syscall(error, "File Not Found");`	
Passed Argument: string message	**Returned Data:** None
Error: None	

substring – 85	
Return the substring at the specified starting and ending position. Format for the string passed to the API is "S:E:String" where "S" is the starting position for the substring and "E" is the ending position. **NOTE:** the string position is zero based.	
`str fullString = "subStringMessage";` `str subString  = syscall(substring, "3:8:" + fullString); //"String"`	
Passed Argument: String as described above	**Returned Data:** String
Error: Invalid start or end values	

tostring – 86	
Return all the characters in a string to the specified number. The number of characters to return is specified as "#:" and appended to the start of the string passed to the API. **NOTE:** the string position is zero based.	
`str fullString = "toStringMessage";` `str toString   = syscall(tostring, "2:" + fullString);  // "to"`	
Passed Argument: String as described above	**Returned Data:** String
Error: Invalid start value for string	

fromstring – 87	
Return all the characters after the specified position. The start position is specified as "#:" and appended to the start to the string passed to the API. **NOTE:** the string position is zero based.	
`str fullString = "fromString";` `str fromString = syscall(fromString, "4:" + fullString);  // "String"`	
Passed Argument: String as described above	**Returned Data:** None
Error: Invalid start value for string	

getmetadesc – 88	
If a "meta" "desc" statement was included in a .s file, then return it else an empty string.	
`str descString = syscall(getmetadesc, " start.s");`	
Passed Argument: .script Filename to get description	**Returned Data:** String
Error: File not found/File not .s	

fileread – 89	
Return the contents of the current .txt file. No other file type can be accessed – the .txt file type is designed to provide non-volatile storage to the Jade Robot.	
`str testFileContents = syscall(fileread, "test.txt");`	
Passed Argument: Filename, always with .txt extension	**Returned Data:** Contents of File
Error: File Note Found/Invalid File Type	

filewrite – 90	
Write string data to the .txt file. The file name is specified by appending it to the start of the string followed by a colon (":"). This API cannot overwrite an existing file. No other file type can be accessed – the .txt file type is designed to provide non-volatile storage to the Jade Robot.	
`syscall(filewrite, "test.txt:" + testTxtData);`	
Passed Argument: Filename appended to data string	**Returned Data:** None
Error: Filename already in use/Invalid File Type/Insufficient File Space Available	

filedelete – 91	
Delete the specified .txt file. No other file type can be accessed – the .txt file type is designed to provide non-volatile storage for the Jade Robot.	
`syscall(filedelete, "test.txt");`	
Passed Argument: Filename to be deleted	**Returned Data:** None
Error: Filename not found	

loadpanelfile – 92	
Load the specified .panel (.p) file into the panel driver but do not execute it.	
`syscall(loadpanelfile, "_demo.p");`	
Passed Argument: .Panel (.p) file	**Returned Data:** None
Error: .Panel file not found/bmp (.b) file specified in .panel file not found	

runpanelfile – 93	
Start executing the loaded panel file. This command can be used to re-run the current panel file loaded in the Jade Robot (ie after processing an "Enter"/Red button press).	
`syscall(runpanelfile, "");`	
Passed Argument: None	**Returned Data:** None
Error: No panel file loaded	

stoppanelfile – 94	
Stop execution of the current panel file. For the user, the only indication they will have the panel file is not running is that the display will not change due to button presses.	
`syscall(stoppanelfile, "");`	
Passed Argument: None	**Returned Data:** None
Error: No panel file loaded/active	

setpanelicon – 95	
Change a specific icon control on the panel. The format for the data passed is "i:fileName.b" where "i" is the control name of the icon to be changed and "fileName.b" is the new icon filename.	
`syscall(setpanelicon, "iconControl:newBMP.b");`	
Passed Argument: String as described above	**Returned Data:** None
Error: Panel not loaded/Control not found/Wrong type of control/Invalid bmp file	

getpanelicon – 96	
Return the current bitmap (.bmp/.b) being used for an icon.	
`str currentIcon = syscall(getpanelicon, "iconControl");`	
Passed Argument: String with icon label	**Returned Data:** bitmap being displayed
Error: Panel not loaded/Control not found/Invalid control type	

setpaneltext – 97	
Set the text value for a text control. The passed information consists of "c:newText" where "c" is text control and "newText" is the text string. Valid text consists of characters from ASCII space (' ' 0x20) to '~' (0x7E).	
`syscall(setpaneltext, "textControl:New Text");`	
Passed Argument: String as described above	**Returned Data:** None
Error: Panel not loaded/Control not found/Invalid control type	

getpaneltext – 98	
Return the current text for the specified control.	
`str textValue = syscall(getpaneltext, "");`	
Passed Argument: None	**Returned Data:** Text value for Control
Error: Panel not loaded/Control not found/Invalid control type	

setpanelvalue – 99	
Each control has a value associated with it for information storage, to display a specific value on the control (as in the case of dials and sliders) or indicate that a specific control has been selected (radio buttons). The value is specified using the format "c:#" where "c" is the control name and "#" is the value (ranging from "0" to "100" or the maximum number of radio buttons).	
`syscall(setpanelvalue, "control:55");`	
Passed Argument: String as described above	**Returned Data:** None
Error: Panel not loaded/Control not found/Invalid control type	

getpanelvalue – 100	
Return the value for the specified panel control.	
`str controlValue = syscall(getpanelvalue, "control");`	
Passed Argument: Control name	**Returned Data:** Control Value
Error: Panel not loaded/Control not found	

showpanelcontrol – 101	
Change the operation parameters of the control so it will be displayed on the Jade Robot Display.	
`syscall(showpanelcontrol, "control");`	
Passed Argument: Control name	**Returned Data:** None
Error: Panel not loaded/Control not found	

noshowpanelcontrol – 102	
Change the operation parameters of the control so it will NOT be displayed on the Jade Robot Display.	
`syscall(noshowpanelcontrol, "control");`	
Passed Argument: Control name	**Returned Data:** None
Error: Panel not loaded/Control not found	

stoppanelcontrol – 103	
Change the control parameters so that the user can navigate to the control.	
`Syscall(stoppanelcontrol, "control");`	
Passed Argument: Control name	**Returned Data:** None
Error: Panel not loaded/Control not found	

nostoppanelcontrol – 104	
Change the control parameters so the user can NOT navigate to the control.	
`syscall(nostoppanelcontrol, "Control");`	
Passed Argument: Control name	**Returned Data:** None
Error: Panel not loaded/Control not found	

loadsubpanelfile – 105	
Load a sub-panel into the current panel. A sub-panel file has the same extension (.p) as a panel file and can be used as a panel file, but when used as a sub-panel file, it's dimensions must match those of the primary panel file.	
`syscall(loadsubpanelfile, "subPanel.p");`	
Passed Argument: sub-panel filename	**Returned Data:** None
Error: Panel not loaded/Sub-panel file not found/Sub-Panel dimensions are invalid for panel	

setactivepanelcontrol – 106	
Set the currently active panel control. When the panel driver first starts up, the first control in the .panel file is selected as the active panel control – this API allows the program to select which one is active.	
`syscall(setactivepanelcontrol, "control");`	
Passed Argument: Control name	**Returned Data:** None
Error: Control not found/Control not "stoppable"	

scriptbreak – 107	
Stop execution of the .script at the specified location. This API is normally used when debugging an application.	
`syscall(scriptbreak, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

btser – 108	
Return the Bluetooth's module serial number. This serial number is robot's unique identifier. The serial number is in the format "####-##-######" where "#" is a hexadecimal digit.	
`str btSerial = syscall(btser, "");`	
Passed Argument: None	**Returned** Data: 14 character string as defined above
Error: None	

gethardwareversion – 109	
Return the version information for the robot hardware. For the first release of the robot it is "Robot:1.00"	
`str hwVersion = syscall(gethardwareversion, "");`	
Passed Argument: None	**Returned Data:** Robot hardware version string
Error: None	

irdetecton – 110	
Turn on the IR object detection and ranging detect hardware. This hardware also includes the ability to receive remote control codes as well detect other robots – the IR object detection hardware power control is primarily used during sleep mode to minimize current drain.	
`syscall(irdetecton, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

irdetectoff – 111	
Turn off the IR object detection and ranging detect hardware. This hardware also includes the ability to receive remote control codes as well detect other robots – the IR object detection hardware power control is primarily used during sleep mode to minimize current drain.	
`syscall(irdetectoff, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

irremotepanasonic – 112	
Set the remote which controls the robot to "Panasonic DVD" – the arrow keys and "Enter"/"OK" button will act like the buttons on the robot.	
`syscall(irremotepanasonic, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

irremotetoshiba – 113	
Set the remote which controls the robot to "Toshiba DVD" – the arrow keys and "Enter"/"OK" button will act like the buttons on the robot.	
`syscall(irremotetoshiba, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

irremotelg – 114	
Set the remote which controls the robot to "LG DVD" – the arrow keys and "Enter"/"OK" button will act like the buttons on the robot.	
`syscall(irremotelg, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

irgetremote – 115	
Return the currently active remote control. This API returns a 12 character string with "PANASONIC ", "TOSHIBA " or "LG ".	
`str currentRemote = syscall(irgetremote, "");`	
Passed Argument: None	**Returned Data:** String as defined above
Error: None	

setsleep – 116	
Set time delay before robot enters sleep mode (which is the low-power mode) in minutes (from "1" to "60").	
`syscall(setsleep, itos(15)); // Put Robot to sleep after 15 minutes`	
Passed Argument: Decimal string from "1" to "60"	**Returned Data:** None
Error: Invalid sleep time specified	

getsleep – 117	
Return the length of time before the robot enters sleep mode.	
`int sleepTime = stoi(syscall(getsleep, ""));`	
Passed Argument: None	**Returned Data:** Decimal string
Error: None	

sleepoff – 118	
Turn off the sleep mode function	
`syscall(sleepoff, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

setsensorled – 119	
Set the state of one of the "sensor LEDs" on the bottom of the robot (used for line sensor and spectrometer). The format of the command is the same as the "setled" API: "L-V" where "L" is LED number and "V" is the value (in this case just "1" or "0"). The LEDs are defined as: 0. Blue (470nm) response 1. Green (525nm) response 2. Yellow-Green (574nm) response 3. Red (660nm) response 4. 880nm Infrared response 5. 940 Infrared response **NOTE:** Line Sensor and Spectrometer read APIs will turn off any active LEDs before their operation and leave them turned off.	
`syscall(setsensorLED, "1-1"); // Green "Neon" underneath Jade Robot`	
Passed Argument: String as defined above	**Returned Data:** None
Error: Invalid command format/Invalid LED specified/Invalid LED value	

getsensorled – 120	
Return the current state of the specified "sensor LED" (on the bottom of the robot used for the line sensor and spectrometer). The command passed to the API is the single digit string with the LED number and the state ("1" for on and "0" for off) is returned. The LEDs are defined as: 0. Blue (470nm) response 1. Green (525nm) response 2. Yellow-Green (574nm) response 3. Red (660nm) response 4. 880nm Infrared response 5. 940 Infrared response	
`int blueSensorLEDState = syscall(getsensorled, "1");`	
Passed Argument: Single digit string	**Returned Data:** Single digit string
Error: Invalid command/Invalid LED specified	

messageboxok – 121	
Put a message box on the OLED display with "OK" as the user instructinon.	
`syscall(messageboxok, "Message to Display");`	
Passed Argument: Message to Display	**Returned Data:** None
Error: None	

messageboxcancel – 122	
Put a message box on the OLED display with "Cancel" as the user instruction.	
`syscall(messageboxcancel, "Message to Display");`	
Passed Argument: Message to Display	**Returned Data:** None
Error: None	

messageboxyesno – 123	
Put a message box on the OLED display with "Yes/No" as the user instruction with "Yes" highlighted.	
`if ("YES" == syscall(messageboxyesno, "User Question")) {`	
Passed Argument: Message for User to Reply to	**Returned Data:** "YES"/"NO"
Error: None	

messageboxnoyes – 124	
Put a message box on the OLED display with "Yes/No" as the user instruction with "No" highlighted.	
`if ("YES" == syscall(messageboxnoyes, "User Question")) {`	
Passed Argument: Message for User to Reply to	**Returned Data:** "YES"/"NO"
Error: None	

getname – 125	
Return the BT Name of the Jade Robot.	
`str btName = syscall(getname, "");`	
Passed Argument: None	**Returned Data:** BT Name of Robot
Error: None	

setname – 126	
Specify the BT Name of the Jade Robot. The name can be up to 16 characters in length with no " " (space) or "-" characters as the first or last characters. It is recommended that just characters ("a" to "z" and "A" to "Z"), numbers ("0" to "9"), space and dash ("-") are the only characters used for Jade Robot's name.	
`syscall(setname, "New Jade Name");`	
Passed Argument: New Jade Name	**Returned Data:** None
Error: Invalid Name Format	

btstatusname – 127	
Return the Bluetooth status as a string: - SETUP (Should never be seen with the Jade Robot as set up must complete before scripts can start running) - NO BT (Should never be seen with Jade Robot) - OFF (Bluetooth power off) - NO CONNECT (Bluetooth power on but not connected) - CONNECTED (Bluetooth power on and connected to a host) - GET RSSI (Bluetooth power on, connected to host and host has requested a RSSI Check)	
`if ("CONNECTED" == syscall(btstatusname, ""))  {`	
Passed Argument: None	**Returned Data:** String listed above
Error: None	

expsendrx – 128	
Send a message to an I2C device in an expansion slot and (optionally) get reply. Parameters sent to the I2C device are: - Hex Device Address (NOTE: this will be shifted right by 1 to get a 7 bit address) - Hex Device Register - Decimal Number of Bytes to Send (Note that "Register" counts as one) - Decimal Number of Bytes to Receive - Decimal Delay time between sending data and expecting data back - String data to device	
`str value = syscall(expsendrx, "42:43:1:0:10:");`	
Passed Argument: As noted above	**Returned Data:** String from device
Error: Invalid Data formatting	

expansion1poweron – 129	
Turn power on to "Expansion 1"	
`syscall(expansion1poweron, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

expansion1poweroff – 130	
Turn power off to "Expansion 1"	
`syscall(expansion1poweroff, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

expansion1getpower – 131	
Return the current power state of “Expansion 1”	
`if (1 == stoi(syscall(expansion1poweron, “”))) { // Power On`	
Passed Argument: None	**Returned Data:** “1” (On) or “0” (Off)
Error: None	

expansion2poweron – 132	
Turn power on to “Expansion 2”	
`syscall(expansion2poweron, “”);`	
Passed Argument: None	**Returned Data:** None
Error: None	

expansion2poweroff – 133	
Turn power off to “Expansion 2”	
`syscall(expansion2poweroff, “”);`	
Passed Argument: None	**Returned Data:** None
Error: None	

expansion2getpower – 134	
Return the current power state of “Expansion 2”	
`if (1 == stoi(syscall(expansion2poweron, “”))) { // Power On`	
Passed Argument: None	**Returned Data:** “1” (On) or “0” (Off)
Error: None	

allfiledelete – 135	
Delete the specified file. This command can be used on any file type (“filedelete” only works on .txt files). **WARNING:** This API should not be used except in a system operations capacity. There is too much danger that this command can leave the Jade Robot in a non-working state and will require a file system reload.	
`syscall(allfiledelete, “cDemo1.s”);`	
Passed Argument: Filename to be deleted	**Returned Data:** None
Error: Filename not found, Invalid Filename Specified	

drawtextupdate – 136: Release 40 and later	
Draw 5x7 text at the current “startcursorx/y” position with “setcolor” and update “startcursorx” at the end of the position. If any portion of the text lies outside the visible range of the OELD, text will be clipped. Note that the actual text size is 6x8 as each character has an empty top row of pixels and empty right column of pixels.	
`syscall(drawtextupdate, “Write this text string”);`	
Passed Argument: String to text to be written to the OLED	**Returned Data:** None
Error: None	

fullspectroread – 137: Release 40 and later	
Perform a spectrometer read operation. Seven values are returned as three decimal characters in an ASCII string with dashes ("-" ASCII 0x2D) in between them. Each value is a Decimal ASCII string from "0"-"100" and represents: 1. No LED illumination to gauge light level underneath the robot 2. Blue (470nm) response 3. Green (525nm) response 4. Yellow-Green (574nm) response 5. Red (660nm) response 6. 880nm Infrared response 7. 940 Infrared response	
`str spectrometerRead = syscall(fullspectroread, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

motorset – 138: Release 40 and later	
Specify the values for both motors in the form "#:#" where the first "#" is the left motor value (from "-100" to "100") and the second "#" is the right motor value (from "-100" to "100").	
`syscall(motorset, "-75:80"); // Sweeping left turn`	
Passed Argument: Motor Values	**Returned Data:** None
Error: Invalid Motor value string	

setcursorstart – 139: Release 40 and later	
Specify the values for the "X" and "Y" of the "start" cursor position in the form "#:#" where the first "#" is the "X" value (Integer string) and the second "#" is the "Y" value (Integer string).	
`syscall(setcursorstart, "0:0"); // start at top left corner`	
Passed Argument: Cursor Coordinates	**Returned Data:** None
Error: Invalid Coordinate string	

setcursorend – 140: Release 40 and later	
Specify the values for the "X" and "Y" of the "start" cursor position in the form "#:#" where the first "#" is the "X" value (Integer string) and the second "#" is the "Y" value (Integer string).	
`syscall(setcursorend, "127:63"); // end at bottom right corner`	
Passed Argument: Filename to be deleted	**Returned Data:** None
Error: Filename not found, Invalid Filename Specified	

wavefilestatus – 141: Release 41 and later	
Return the status of the audio task playing a file. "-1" means file is playing, "0" means no file playing.	
`int audioStatus = stoi(syscall(wavefilestatus, ""));`	
Passed Argument: None	**Returned Data:** "-1" (Active)/"0" (Stopped)
Error: None	

stutter – 142: Release 42 and later	
Cause the Jade Robot to start vibrating back and forth.	
`syscall(stutter, "");`	
Passed Argument: None	**Returned Data:** None
Error: None	

leftlineraw – 143: Release 42 and later	
Return the current value read from the Left Line Sensor Photocell without any processing. Value returned is from 0 to 3300. **NOTE:** No LEDs are turned on by this command.	
`int leftLineValue = stoi(syscall(leftlineraw, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

rightlineraw – 144: Release 42 and later	
Return the current value read from the Right Line Sensor Photocell without any processing. Value returned is from 0 to 3300. **NOTE:** No LEDs are turned on by this command.	
`int rightLineValue = stoi(syscall(rightlineraw, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

centerlineraw – 145: Release 42 and later	
Return the current value read from the Left Line Sensor Photocell without any processing. Value returned is from 0 to 3300. **NOTE:** No LEDs are turned on by this command.	
`int centerLineValue = stoi(syscall(centerlineraw, ""));`	
Passed Argument: None	**Returned Data:** Decimal ASCII string
Error: None	

getprocessor – 146: Release 42 and later	
Return the information regarding the current processor in format: "K##-##(#)P-R(#)#".	
`Str procInfo = syscall(getprocessor, "");`	
Passed Argument: None	**Returned Data:** ASCII string
Error: None	

Expected Future APIs

Supporting Documents

Jade Robot™ Support Software Installation and Introduction - TBD
Jade Robot™ Script Language Outline – available at http://www.mimetics.ca/knowledge-base-2/
Jade Robot™ Panel Driver Outine - TBD

Glossary

ASCII – Standard 8 bit character set. See http://en.wikipedia.org/wiki/ASCII

Document Updates

Date	Changes	Author
2013.12.28	Initial Release	Myke Predko
2014.01.15	Fixed “clearcisplay” in “cleardisplay” example	Myke Predko
2014.04.26	• Updated requirements of this document • Added additional Syscall APIs (#121 to #134) • Strike through expected future APIs • Added Error for “btoff” if there was a connection	Myke Predko
2014.05.09	Updated with “allfiledelete” and Expansion Port APIs	Myke Predko
2014.08.30	Updated with Release 40 APIs for “Jade Scratch” Programming	Myke Predko
2014.09.01	Added “wavefilestatus” API as part of Release 41	Myke Predko
2014.10.22	Added “stutter” API as part of Release 42	Myke Predko
2014.12.02	Added Line Sense “Raw” APIs as part of Release 42 Updated basic Line Sense APIs to note that they don’t turn on the bottom LEDs	Myke Predko
2014.12.12	Added “getprocessor” API as part of Release 42	Myke Predko