Automation and Robotics

5150 Robotics System
Software Development Kit

User Guide
39644-E0

Printed in Canada

3 0 3 9 6 4 4 E 0 0 7 0 1 ~

Lab-Volt®

AUTOMATION AND ROBOTICS

5150 ROBOTICS SYSTEM
SOFTWARE DEVELOPMENT KIT

by
the Staff
of
Lab-Volt Ltd.

Legal Deposit – First Trimester 2007

ISBN 978-2-89640-105-5

FIRST EDITION, JANUARY 2007

Printed in Canada
January 2007

Foreword

The Lab-Volt 5150 Software Development Kit is a USB DLL library (5150USB.DLL) provided as an abstraction layer between the end-user application and the actual low-level communication protocol from and to the USB Controller.

This library provides simple function calls to control the Lab-Volt 5150 Robot. It also manages the USB low-level communications, as well as error handling.

The prototypes of all the exported functions and some required constants are provided in the file "5150USB.h".

A *.LIB file named "5150USB.LIB" is also provided to allow static linking of the library to the end user application.

Index

DLLCloseDevice . . . 4
DLLDefaultCalib . . . 37
DLLDisableMotors . . . 14
DLLEnableMotors . . . 13
DLLGetAllCurrentPosition . . . 34
DLLGetDesiredPosition . . . 19
DLLGetDevicePortNumber . . . 1
DLLGetFrequency . . . 23
DLLGetHardwareInfo . . . 27
DLLGetPosition . . . 17
DLLGetSelectStatus . . . 11
DLLGetSolenoidOutput . . . 10
DLLGetSwitch . . . 8
DLLGetTTLInput . . . 7
DLLGetTTLOutput . . . 6
DLLGetUpdateStatus . . . 30
DLLHardHome . . . 26
DLLIsDeviceOpen . . . 2
DLLMotorRunStatus . . . 33
DLLMoveMotor . . . 15
DLLMoveToCalibPos . . . 35
DLLOpenDevice . . . 3
DLLResetBoard . . . 12
DLLResetDesiredPosition . . . 20
DLLRunMotors . . . 24
DLLSaveCalib . . . 36
DLLSetDesiredOffset . . . 21
DLLSetDesiredPosition . . . 18
DLLSetFrequency . . . 22
DLLSetPosition . . . 16
DLLSetSelectMonitor . . . 32
DLLSetSolenoidOutput . . . 9
DLLSetTTLOutput . . . 5
DLLStopMotors . . . 25
DLLTerminateUpdate . . . 31
DLLUpdateFirmware . . . 29

REVISION HISTORY . . . 38

DLLGetDevicePortNumber

Retrieves the COM port number of the first CP210x USB device connected to the computer.

long __stdcall DLLGetDevicePortNumber(void);

Parameters

This function has no parameters.

Return Values

The COM port number (e.g.: 6, device on COM6).

Returns (-1), no device found.

Header: 5150USB.h

! Requirements
None

+ See Also
DLLOpenDevice, DLLCloseDevice

DLLIsDeviceOpen

Tests if a device has already been opened by a previous call to DLLOpenDevice.

long __stdcall DLLGetDevicePortNumber(void);

Parameters

This function has no parameters.

Return Values

DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_SUCCESS	Device open and available

Header: 5150USB.h

! Requirements
None

+ See Also
DLLOpenDevice, DLLCloseDevice

DLLOpenDevice

Tries to find a 5150 USB controller connected to the system and enables the communication with it for later calls.

long __stdcall DLLOpenDevice(void);

Parameters

This function has no parameters.

Return Values

DLL_ERROR_NO_DEVICE	No device found
DLL_SUCCESS	Device open and available

Header: 5150USB.h

! Requirements
None

+ See Also
DLLIsDeviceOpen, DLLCloseDevice

DLLCloseDevice

Closes the handle open to a 5150 USB controller device. You should call DLLCloseDevice at the end of your application to free allocated system resources.

```
long __stdcall DLLCloseDevice( void );
```

Parameters

This function has no parameters.

Return Values

DLL_ERROR_NO_DEVICE	No device open
DLL_SUCCESS	Device now closed

Header: 5150USB.h

! Requirements
None

+ See Also
DLLIsDeviceOpen, DLLOpenDevice

DLLSetTTLOutput

Updates the output pattern of one of the four TTL outputs on the 5150 Robot.

long __stdcall DLLSetTTLOutput(BYTE byOutputId, BYTE byState);

Parameters

byOutputId	The Output ID (0…3) The Output IDs are: 0x00 = TTL Output 1 (BROWN) 0x01 = TTL Output 2 (RED) 0x02 = TTL Output 3 (ORANGE) 0x03 = TTL Output 4 (YELLOW)
byState	The Desired State TRUE 0x01 FALSE 0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLGetTTLOutput

DLLGetTTLOutput

Retrieves the current state of one of the four TTL outputs on the 5150 Robot.

long __stdcall DLLGetTTLOutput(BYTE byOutputId, BYTE *pbyState);

Parameters

byOutputId	The Output ID (0…3) The Output IDs are: 0x00 = TTL Output 1 (BROWN) 0x01 = TTL Output 2 (RED) 0x02 = TTL Output 3 (ORANGE) 0x03 = TTL Output 4 (YELLOW)
*pbyState	A pointer to a BYTE (8-bit) variable for the state TRUE 0x01 FALSE 0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLSetTTLOutput

DLLGetTTLInput

Retrieves the current state of one of the four TTL inputs on the 5150 Robot.

long __stdcall DLLGetTTLInput(BYTE byInputId, BYTE *pbyState);

Parameters

byInputId The Input ID (0…3)

The Input IDs are:

0x00 = TTL Input 1 (BROWN)

0x01 = TTL Input 2 (RED)

0x02 = TTL Input 3 (ORANGE)

0x03 = TTL Input 4 (YELLOW)

*pbyState A pointer to a BYTE (8-bit) variable for the state

TRUE	0x01
FALSE	0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLSetTTLOutput

DLLGetSwitch

Retrieves the current state of one of the limit switches of the 5150 Robot.

long __stdcall DLLGetSwitch(BYTE bySwitchId, BYTE *pbyState);

Parameters

bySwitchId The Limit Switch ID (0x09…0x0C)

The Switch IDs are:

0x09 = WRIST Switch

0x0A = FOREARM Switch

0x0B = UPPER ARM Switch

0x0C = BASE Switch

*pbyState A pointer to a BYTE (8-bit) variable for the state

TRIGGED	0x01
UNTRIGGED	0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

DLLSetSolenoidOutput

Updates one of the four selected solenoid outputs on the 5150 Robot.

long __stdcall DLLSetSolenoidOutput(BYTE bySolenoidId, BYTE byState);

Parameters

bySolenoidId The Solenoid ID (0…3)

The Solenoids IDs are:

0x00 = Solenoid Output 1

0x01 = Solenoid Output 2

0x02 = Solenoid Output 3

0x03 = Solenoid Output 4

byState The Desired State

TRUE	0x01
FALSE	0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLGetSolenoidOutput

DLLGetSolenoidOutput

Retrieves the current state of one of the 4 solenoid outputs on the 5150 Robot.

long __stdcall DLLGetSolenoidOutput(BYTE bySolenoidId, BYTE *pbyState);

Parameters

bySolenoidId   The Solenoid ID (0…3)

The Solenoids IDs are:

0x00 = Solenoid Output 1

0x01 = Solenoid Output 2

0x02 = Solenoid Output 3

0x03 = Solenoid Output 4

*pbyState   A pointer to a BYTE (8-bit) variable for the state

TRUE	0x01
FALSE	0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice

+ See Also
DLLSetSolenoidOutput

DLLGetSelectStatus

Returns the state of the LAST SELECTED switch input (the one selected by a previous call to DLLGetSwitch).

long __stdcall DLLGetSelectStatus(BYTE *pbyState);

Parameters

*pbyState	A pointer to a BYTE (8-bit) variable for the state	
	TRIGGED	0x01
	UNTRIGGED	0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.
You should select the switch to monitor by calling DLLGetSwitch at least once.

+ See Also
DLLGetSwitch

DLLResetBoard

Performs a reset of the motor interface board and verifies that it is ready and in working order.

long __stdcall DLLResetBoard(void)

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply or interface not responding

Header: 5150USB.h

! Requirements
None

DLLEnableMotors

Enables the motor drivers on the motor interface board.

long __stdcall DLLEnableMotors(void)

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
The communication must first be opened with DLLOpenDevice

+ See Also
DLLDisableMotors

DLLDisableMotors

Disables the motor drivers on the motor interface board.

long __stdcall DLLDisableMotors(void);

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
The communication must first be opened with DLLOpenDevice

+ See Also
DLLEnableMotors

DLLMoveMotor

Makes a motor turn a single step in the desired direction.

long __stdcall DLLMoveMotor(BYTE byMotorId, BYTE byDirection);

Parameters

byMotorId	The ID of the motor (0…6) The Motor IDs are: 0x00 = MOTOR_GRIPPER 0x01 = MOTOR_WRIST1 0x02 = MOTOR_WRIST2 0x03 = MOTOR_FOREARM 0x04 = MOTOR_SHOULDER 0x05 = MOTOR_BASE 0x06 = MOTOR_EXTERNAL
byDirection	The direction flag 0x00 = DIR_BACKWARD 0x01 = DIR_FORWARD

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

DLLSetPosition

Sets the current position counter of a specific motor.

long __stdcall DLLSetPosition(BYTE byMotorId, int sPosition);

Parameters

byMotorId	The ID of the motor (0…6) The Motor IDs are: 0x00 = MOTOR_GRIPPER 0x01 = MOTOR_WRIST1 0x02 = MOTOR_WRIST2 0x03 = MOTOR_FOREARM 0x04 = MOTOR_SHOULDER 0x05 = MOTOR_BASE 0x06 = MOTOR_EXTERNAL
sPosition	A 32-bit signed position value

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLGetPosition, DLLSetDesiredPosition, DLLGetDesiredPosition

DLLGetPosition

Returns the current position counter of a specific motor.

long __stdcall DLLGetPosition(BYTE byMotorId, int *psPosition);

Parameters

byMotorId	The ID of the motor (0…6) The Motor IDs are: 0x00 = MOTOR_GRIPPER 0x01 = MOTOR_WRIST1 0x02 = MOTOR_WRIST2 0x03 = MOTOR_FOREARM 0x04 = MOTOR_SHOULDER 0x05 = MOTOR_BASE 0x06 = MOTOR_EXTERNAL
*psPosition	A pointer to a 32-bit signed variable to receive the position

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLSetPosition, DLLSetDesiredPosition, DLLGetDesiredPosition

DLLSetDesiredPosition

Defines the position you want a motor to go to.

long __stdcall DLLSetDesiredPosition(BYTE byMotorId, int sPosition);

Parameters

byMotorId The ID of the motor (0…6)

The Motor IDs are:

0x00 = MOTOR_GRIPPER

0x01 = MOTOR_WRIST1

0x02 = MOTOR_WRIST2

0x03 = MOTOR_FOREARM

0x04 = MOTOR_SHOULDER

0x05 = MOTOR_BASE

0x06 = MOTOR_EXTERNAL

sPosition A 32-bit signed value for the target position

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLGetDesiredPosition

DLLGetDesiredPosition

Returns the target position for a specific motor.

long __stdcall DLLGetDesiredPosition(BYTE byMotorId, int *psPosition);

Parameters

byMotorId	The ID of the motor (0…6) The Motor IDs are: 0x00 = MOTOR_GRIPPER 0x01 = MOTOR_WRIST1 0x02 = MOTOR_WRIST2 0x03 = MOTOR_FOREARM 0x04 = MOTOR_SHOULDER 0x05 = MOTOR_BASE 0x06 = MOTOR_EXTERNAL
*psPosition	A pointer to a 32-bit signed variable to receive the target position

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLSetDesiredPosition

DLLResetDesiredPosition

Clears the previously defined motor motions.

long __stdcall DLLResetDesiredPosition(void);

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLSetDesiredPosition, DLLGetDesiredPosition

DLLSetDesiredOffset

Lets you specify a new position based on an offset from the current location.

long __stdcall DLLSetDesiredOffset(BYTE byMotorId, int sOffset);

Parameters

byMotorId	The ID of the motor (0…6) The Motor IDs are: 0x00 = MOTOR_GRIPPER 0x01 = MOTOR_WRIST1 0x02 = MOTOR_WRIST2 0x03 = MOTOR_FOREARM 0x04 = MOTOR_SHOULDER 0x05 = MOTOR_BASE 0x06 = MOTOR_EXTERNAL
sOffset	A 32-bit signed offset value

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

DLLSetFrequency

Sets the pulse frequency for a specified motor.

long __stdcall DLLSetFrequency(BYTE byMotorId, unsigned short usFrequency);

Parameters

byMotorId	The ID of the motor (0…6) The Motor IDs are: 0x00 = MOTOR_GRIPPER 0x01 = MOTOR_WRIST1 0x02 = MOTOR_WRIST2 0x03 = MOTOR_FOREARM 0x04 = MOTOR_SHOULDER 0x05 = MOTOR_BASE 0x06 = MOTOR_EXTERNAL
usFrequency	A 16-bit unsigned value for the frequency

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLGetFrequency

DLLGetFrequency

Returns the pulse frequency for a specified motor.

long __stdcall DLLGetFrequency(BYTE byMotorId, unsigned short *pusFrequency);

Parameters

byMotorId	The ID of the motor (0…6) The Motor IDs are: 0x00 = MOTOR_GRIPPER 0x01 = MOTOR_WRIST1 0x02 = MOTOR_WRIST2 0x03 = MOTOR_FOREARM 0x04 = MOTOR_SHOULDER 0x05 = MOTOR_BASE 0x06 = MOTOR_EXTERNAL
*pusFrequency	A pointer to a 16-bit unsigned value for the returned frequency value

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLSetFrequency

DLLRunMotors

Starts all motors that have a target position different than their current position.

long __stdcall DLLRunMotors(void);

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.
You should first set the destination position and frequency of the motors with DLLSetDesiredPosition and DLLSetFrequency.

+ See Also
DLLStopMotors, DLLSetDesiredPosition, DLLSetFrequency

DLLStopMotors

Stops the motors. This function will return only when the movement has stopped.

long __stdcall DLLStopMotors(void);

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLRunMotors

DLLHardHome

One-step command used to return the robot to its home position. This function returns only once the procedure is completed, which could take as long as 60 seconds.

long __stdcall DLLHardHome(void);

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

DLLGetHardwareInfo

Returns the firmware revision number and the numerous diagnostic codes. You should call DLLGetHardwareInfo to get more information about the problem(s) that might cause a DLL_ERROR_BAD_REPLY to be returned in response to any other function calls.

```
long __stdcall DLLGetHardwareInfo(void *pvData );
```

Parameters

*pvData A pointer to a BYTE array of at least 20 bytes

*pvData+0 USB controller firmware major revision number

*pvData+1 USB controller firmware minor revision number

*pvData+2 Last recorded error code producing a NAK

0x00	No error yet!
0x01	Firmware in one or more time base generators (PIC12F629) not matching the reference code in memory
0x02	Unable to reprogram the 7 time base generators (PIC12)
0x03	MIB interface not connected or OFF
0x04	Received command frame not RECOGNIZED
0x05	MotorID too high (>6)
0x06	Problem encountered in attempting to start INT_EXT for motor run
0x07	Unable to do a MIB_Port_Write (a time-out error occurred)
0x08	MIB_Port_Wait_For_Busy_Low() function timed-out

*pvData+3 Each bit represents a faulty PIC12 time base generator, when any.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

DLLUpdateFirmware

Opens the specified INTEL-HEX file, and then starts the update thread that will do the actual update. After calling DLLUpdateFirmware, you should call DLLGetUpdateStatus to get information about the update process.

```
long __stdcall DLLUpdateFirmware( char *pszFileName );
```

Parameters

*pszFileName	A pointer to a string containing the complete PATH and FILENAME of the updated INTEL-HEX file

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_FILE_READ	Could not open the file. Update aborted.

Header: 5150USB.h

! Requirements
None

+ See Also
DLLGetUpdateStatus

DLLGetUpdateStatus

Gets the status of the firmware update process. By checking the value returned in pnStatus, you can check if the update is completed. PnLineCount and pnMaxCount can be used to compute the completion of the update process.

```
long __stdcall DLLGetUpdateStatus( int *pnStatus, int *pnLineCount, int
*pnMaxCount );
```

Parameters

*pnStatus   A pointer to an integer for a status code:

DLL_STATUS_UPDATE_WORKING

Update thread still running

DLL_STATUS_UPDATE_ERROR

Firmware update error

DLL_STATUS_UPDATE_SUCCESS

Firmware update completed

*pnLineCount   Last line updated by the thread

*pnMaxCount   Total number of lines in the file

Return Values

DLL_SUCCESS	Always

Header: 5150USB.h

! Requirements
None

+ See Also
DLLUpdateFirmware

DLLTerminateUpdate

Aborts the firmware update process. Normally, you should not call this function. The firmware update will complete by itself and the status returned by DLLGetUpdateStatus will change to DLL_STATUS_UPDATE_SUCCESS.

Il you abort the firmware update, the USB controller firmware will have to be successfully updated to be operational again.

```
long __stdcall DLLTerminateUpdate( void )
```

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	Always

Header: 5150USB.h

! Requirements
None

+ See Also
DLLUpdateFirmware, DLLGetUpdateStatus

DLLSetSelectMonitor

Sets the state of the SELECT line (limit switches) that will stop the motors.

long __stdcall DLLSetSelectMonitor(BYTE byState);

Parameters

byState The state to monitor:

TRIGGED	0x01
UNTRIGGED	0x00

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLGetSwitch, DLLGetSelectStatus

DLLMotorRunStatus

Returns the state of the "Running-Flag" byte, this flag indicating if one or more motors are running.

long __stdcall DLLMotorRunStatus(BYTE *pbyStatus);

Parameters

*pbyStatus	Pointer to a BYTE to receive the status: 0 All motors stopped n Each bit set to "1" represents a motor that is still running.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLRunMotors

DLLGetAllCurrentPosition

Returns the "Running-Flag" state (TRUE/FALSE) and the current position of all 7 motors.

```
long __stdcall DLLGetAllCurrentPosition(BYTE * runningMotorFlag, int * positions);
```

Parameters

*runningMotorFlag — Pointer to a BYTE to receive the running flag state.

0x00	All motors stopped
0x01	At least one motor running

*positions — Pointer to a 32-bit signed variable to receive the positions of the 7 motors.

position[0]	Position of motor 1
position[1]	Position of motor 2
...	
position[6]	Position of motor 7

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply
DLL_ERROR_EMPTY_COMMAND	One of the function parameters is NULL

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.

+ See Also
DLLMotorRunStatus

DLLMoveToCalibPos

Moves the robot to a predefined calibration position. This method is useful in helping the user to put the robot in the calibration position.

long __stdcall DLLMoveToCalibPos(void);

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.
The robot must first be hard homed before using this function.

+ See Also
DLLSaveCalib, DLLDefaultCalib

DLLSaveCalib

This function saves the calibration offsets from the robot's current calibration position. It will first move the robot to the ideal home position, then do a hard home, and finally save the offsets in the controller memory.

```
long __stdcall DLLSaveCalib( void );
```

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	If successful
DLL_ERROR_NO_DEVICE	No device open
DLL_ERROR_FILE_WRITE	Write operation failed
DLL_ERROR_FILE_READ	Read operation failed
DLL_ERROR_BAD_REPLY	Bad first character reply

Header: 5150USB.h

! Requirements
Communication must first be opened with DLLOpenDevice.
The robot must first be hard homed before using this function.

+ See Also
DLLMoveToCalibPos, DLLDefaultCalib

DLLDefaultCalib

Resets the calibration offsets to 0.

long __stdcall DLLDefaultCalib(void);

Parameters

This function has no parameters.

Return Values

DLL_SUCCESS	Always

Header: 5150USB.h

! Requirements
None

+ See Also
DLLMoveToCalibPos, DLLSaveCalib

REVISION HISTORY

JANUARY 18th, 2005: Initial Release

SEPTEMBER 12th, 2005: Modified documentation for the DLLMotorRunStatus command

JUNE 20th, 2006: Calibration methods added

AUTOMATION AND ROBOTICS
5150 ROBOTICS SYSTEM
SOFTWARE DEVELOPMENT KIT
39644-E0
First Edition: January 2007
Printed: January 2007

We Value Your Opinion!

COMMENTS

If you have any comments on this guide, please take a few minutes to write them down in the following lines and return this sheet to the address indicated on verso. Thank you! Your comments will enable us to produce better manuals.

____

____

____

____

____

____

____

PUBLICATION ERRORS AND COMMENTS

Please enclose photocopies of pages where errors were found and indicate the modifications that should be carried out.

If you want to receive the corrected pages, please fill in the identification section.

BACKGROUND INFORMATION

☐ Instructor ☐ Student

☐ High School ☐ Vocational ☐ Technical Institute ☐ University

IDENTIFICATION

NAME ____

ADDRESS ____

____

PHONE ______________________ FAX ___________________________

Fold and tape Fold and tape

LAB-VOLT LTD.
Technical Publications Department
675, rue du Carbone
Québec (Québec) G2N 2K7
Canada

Cut along line

Fold and tape Fold and tape