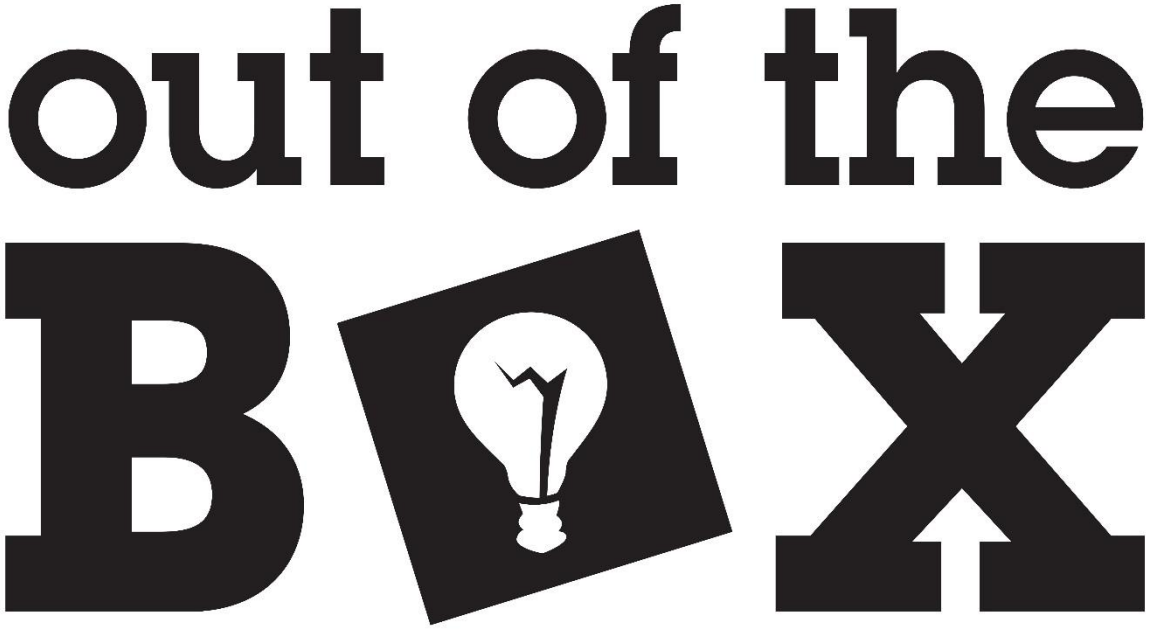

SADI LIB 3.0 Reference

out of the BOX
Electronics and Robotics LLC
ootbrobotics.com

Table of Contents

SADI Software Overview

The interface of the SADI is handled by the DLL provided with the device. This library contains all the function calls used for communicating with the device. SADI was designed to interface with LabView however, C/C++, Python, Matlab/Simulink or any other software platform capable of interfacing with external DLLs will likely work.

Prerequisites

Instance

New in V3 of the SADI DLL there is the capability of connecting multiple SADI devices. This is handled by using an instance number for the device. Nearly all functions use this instance number so it should be stored as a program variable.

SADI States

SADI has various states that can be queried using **getState**.

Opened_state

getState = 1

This is the default state, and is set when sadiSendReset. Before this time the state cannot be queried. So a variable should be

Ready_state

getState = 2

This state is reached when the SADI replies to **requestReady_LV**. In This state the SADI is able to be initialized.

InitSent_state

getState = 4

This is an intermediary state that happens as soon as an initialization has been sent. The device will proceed to Setup_state as soon as a acknowledgement of the initialization being successful has been received.

Setup_state

getState = 3

once the device is initialized and the SADI has acknowledged. This state means the SADI is sitting idle until a start command is sent.

Start_initiated_state

getState = 5

This state is reached once **sendStart_LV** has been called.

Run_state

getState = 6

This state is reached once **sendStart_LV** has been acknowledged by SADI. During Run_state the SADI will report data per the initialization of the device and may also be written to.

StopInitiated_state

getState = 7

This state is reached once **sendStop_LV** has been called.

Stopped_state

getState = 8

This state is reached once **sendStop_LV** has been acknowledged. This state is equivalent to Setup state

Error_state

getState = 9

This state indicates an error has occurred.

SADI Connection and Info

getNumberOfSADI()

Use this function/VI to determine how many SADI devices are connected to the system.

Return Type: unsigned int

getDeviceNumber (int instanceNumber)

Use this function/VI to determine the device number of a device instance.

The device number is the number assigned by the system OS while Instance is related to programming environment utilizing SADI_FW3.dll to communicate with the SADI(s). These numbers likely the same but this function exists in case they are not.

Return Type: unsigned int

openSADI_LV (int instance,int deviceNumber)

This function/VI opens a SADI for communication. To verify that the device was opened successfully verify that the return value is 0 which corresponds to FT_OK.

set **instance** and device number to the value of the SADI you want to open. If only one SADI device is connected use 0.

Return Type: FT_STATUS

getDllVersionNumber()

This function/VI returns the DLL version number.

Return Type: float

getFWVersionNumber_LV (int instance)

This function/VI returns the firmware version of the SADI.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: float

closeSADI_LV (int instance)

This function/VI closes communication with a SADI. Use the same instance number used to open the device to close it.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: FT_STATUS

requestReady_LV (int instance)

This call is used to query the SADI to see if it is ready. If it is this will cause a state change in **getState.**

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

sendReset_LV (int instance)

Calling this will reset the SADI.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

SADI Configuration

Predefined Initializations

SADI is designed to be highly configurable. However to simplify initialization some predefined configurations exist. Currently the two options are as follows.

Pendulum mode

Mode pre-configured for the interface of the SADI to the Out of the Box pendulum plant.

MOM mode

Mode pre-configured for use in MOM Lab 3301C.

sendPredefInit_LV (int instance, uint16_t selection, uint16_t freq)

Call to initialize the SADI for pre-defined configurations. Simply enter the mode value into the selection value and set the loop frequency (freq) in Hz.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

General Initialization

The following functions should be used when creating a custom SADI configuration.

setupADC_LV (int instance, uint8_t adcNum, uint8_t adcSetting)

Use this function/VI to setup the sensitivity of the ADC. adcNum is the channel number 0-3. Acceptable values for adcSetting are as follows.

Range	Value
OFF	0
± 10.24V	1
± 5.12V	2
± 2.56V	3
± 1.28V	4
± 0.64V	5
0 to 10.24V	9
0 to 5.12V	10
0 to 2.56V	11
0 to 1.28V	12

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

A value of 0 corresponds to a successful configuration.

setupLoopFrequency_LV (int instance, uint16_t frequency)

Use this function/VI to set the loop frequency for the SADI. For read only operation frequencies can be as high as 20KHz. For read and write operation 10KHz is the max.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

A value of 0 corresponds to a successful configuration.

setupEncoder_LV (int instance, uint8_t encoderNum, uint8_t pinLocation, uint8_t index, uint8_t ABS, uint8_t filterDepth, uint16_t resolution)

encoderNum corresponds to one of two encoders 0 or 1. **pinLocaion** sets the location for CHA, CHB, and INDEX signals if applicable. Follow the chart for applicable locations respective to the **pinLocation** value. Values of 6 and 10 don't support INDEX.

pinLocation (value)	CHA Pin	CHB Pin	INDEX Pin
0	0	1	2
1	1	2	3
2	2	3	4
3	3	4	5
4	4	5	6
5	5	6	7
6	6	7	N/A
8	8	9	10
9	9	10	11
10	10	11	N/A

index if set to 1 will configure the encoder to use an index signal. Note from the above not all pin locations support an index signal. **ABS** if set to 1 means the encoder will be absolute. This means that the encoder value will always be positive, and it will only as high as **resolution**. The **filterDepth** value is used to prevent erroneous counting a value of 3 is recommended and a value of 7 is the maximum. Last, resolution is the resolution of the encoder to be connected. The math for **resolution** of a motor output shaft is:
resolution = gear_ratio*encoder_resolution*4

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

A value of 0 corresponds to a successful configuration.

setupServoPWM_LV (int instance, uint8_t servoEnables, uint16_t servoFrequency)

servoEnables is a 4-bit mask

Servo Number	Mask value
0	1, 0b0001
1	2, 0b0010
2	4, 0b0100
3	8, 0b1000

Add together the masks of all the servos to be enabled and set to **servoEnables**. Currently **servoFrequency** is fixed so this parameter can be ignored.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

A value of 0 corresponds to a successful configuration.

setupMotorPWM_LV (int instance, uint8_t motorEnables, uint16_t motorFrequency)

motorEnables is a 4-bit mask

Motor Number	Mask value
0	1, 0b0001
1	2, 0b0010
2	4, 0b0100
3	8, 0b1000

Add together the masks of all the motors to be enabled and set to **motorEnables**. Currently **motorFrequency** is fixed so this parameter can be ignored.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0

Return Type: unsigned int

A value of 0 corresponds to a successful configuration.

setupOutputPinMsk_LV (int instance,uint16_t outputPinMsk)

The outputPinMsk parameter is a 12-bit mask used to configure all desired pins to be outputs. The corresponding values are 2^n where n is the pin of Dn.

Pin	Mask value
D0	1, 0x001, 0b0000 0000 0001
D1	2, 0x002, 0b0000 0000 0010
D2	4, 0x004, 0b0000 0000 0100
D3	8, 0x008, 0b0000 0000 1000
D4	16, 0x010, 0b0000 0001 0000
D5	32, 0x020, 0b0000 0010 0000
D6	64, 0x040, 0b0000 0100 0000
D7	128, 0x080, 0b0000 1000 0000
D8	256, 0x100, 0b0001 0000 0000
D9	512, 0x200, 0b0010 0000 0000
D10	1024, 0x400, 0b0100 0000 0000
D11	2048, 0x800, 0b1000 0000 0000

Add all the desired output pin masks together for the **outputPinMsk** parameter.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

A value of 0 corresponds to a successful configuration.

sendInit_LV (int instance)

Once all the desired setup functions/Vis have been called, and they have been deemed successful by their return values **sendInit_LV** can be called to configure the SADI accordingly.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

Runtime Operations

Start/Stop Acquisition

sendStart_LV (int instance)

Calling this function/VI will trigger acquisition. Reference SADI States for details.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

sendStop_LV (int instance)

Calling this function/VI will stop acquisition. Reference SADI States for details.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

Loop Maintenance

sadiMaintenance_LV (int instance)

This function/VI needs to be called each loop to handle incoming data from the SADI hardware.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

The output value can be disregarded.

Reading

getData (int instance)

not currently functional use getDataLatest

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

getDataLatest (int instance)

this function/VI pulls the latest data packet from the stack. Once pulled use "extractFromInputPacket" calls to retrieve specific values.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: unsigned int

extractFromInputPacket_ADC (int instance,int channel)

Use this function/VI to retrieve ADC data from the current data set. The parameter channel 0-3 corresponds to the ADC the respective ADC channels of the SADI. To get a voltage more easily use the function/VI **convertToVoltage** on the value returned by **extractFromInputPacket_ADC**.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: uint16_t

extractFromInputPacket_ENCPos (int instance,int channel)

Use this function/VI to retrieve encoder data from the current data set. The parameter **channel** can be either 0 or 1 corresponding to the two encoders possibly connected.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: int32_t

extractFromInputPacket_ENCVel (int instance,int channel)

Use this function/VI to retrieve encoder velocity data from the current data set. Encoder velocity is a count of pulses per a full period of the CHA encoder signal. As of v3.22 the clock frequency used to sample the pendulum motor encoder velocity is 8MHz.

Example math:

100 rpm of a 30:1 gear motor with an encoder resolution of 12 pulses per revolution of the back shaft.

100 outputRevs/min *1 min/60 sec *30 revs/1 outputRevs*12 pulsesQuad/revs 1pulseSingleChannel/4 pulsesQuad = 150Hz

(1/150)/(1/8,000,000) = 53,333

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0

Return Type: uint16_t

extractFromInputPacket_inputCapture (int instance)

This function/VI is used specifically for the pendulum plant. The value corresponds to the pendulum encoder value.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: uint16_t

extractFromInputPacket_Inputs (int instance)

This function/VI returns a 12-bit value respective to the 12 i/o pins. A '1' represents a "HIGH" (5V) voltage at the pin.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: uint16_t

extractFromInputPacket_msgCNT (int instance)

This function/VI returns the count value of the given data set. This can be used to determine how much time has elapsed since the last data set was sampled based on the loop frequency the SADI was set to.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: uint32_t

Writing

setupWrite functions/Vis are used to set the next desired values of various writable settings. These functions don't have any effect to the SADI by themselves. The data is only written once sendWrite_LV is called. This methodology is used to keep USB throughput to the device as high as possible.

setupWrite_AOUT_LV(int instance, uint8_t aoutNum, uint16_t AOUT)

This function/VI is used to setup the analog output value of a given channel **aoutNum** (values 0-1). The output value is assigned by **AOUT**. The value will take effect after **sendWrite_LV** is called.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

setupWrite_OutCLR_LV(int instance, uint16_t outMsk)

This function/VI is used to clear (0V LOW) the output of pins as defined by **outMsk** . Use the expression
$\sum 2^n = outMsk$
where n is between 0 and 11. Only sum the expression for pins you wish to clear**.** The value will take effect after **sendWrite_LV** is called.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

setupWrite_OutSET_LV (int instance, uint16_t outMsk)

This function/VI is used to set (5V HIGH) the output of pins as defined by **outMsk** . Use the expression
∑ 2^n = outMsk
where n is between 0 and 11. Only sum the expression for pins you wish to set. The value will take effect after **sendWrite_LV** is called.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

setupWrite_OutTGL_LV (int instance, uint16_t outMsk)

This function/VI is used to toggle the output of pins as defined by **outMsk** . Use the expression
∑ 2^n = outMsk
where n is between 0 and 11. Only sum the expression for pins you wish to toggle. The value will take effect after **sendWrite_LV** is called.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

setupWrite_Servo_LV (int instance, uint8_t servoNum, uint16_t value)

This function/VI assigns **value** to the servo **servoNum** (value 0-3). The value will take effect after **sendWrite_LV** is called.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

setupWrite_Motor_LV (int instance, uint8_t motorNum, uint16_t value)

This function/VI assigns **value** to the motor **motorNum** (value 0-3). The value will take effect after **sendWrite_LV** is called.

Note this function/VI serves a special function in pendulum mode. In this mode the motorNum parameter controls the direction of the motor.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

void sendWrite_LV(int instance)

This function/VI sends all the write data from all the setupWrite function/VI calls.

Note the same data for a given setupWrite function/VI will be retained if it is not overwritten by a setupWrite before a subsequent call of **sendWrite_LV**. For this reason, take care when using **setupWrite_OutTGL_LV**.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0.

Return Type: void

Utilities

getLatestStatus (int instance)

This function/VI returns the latest status of the USB transceiver. This call

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0

Return Type: FT_STATUS

getState (int instance)

This function/VI returns the current state of the connected SADI. For reference consult the following.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0

Return Type: sadiState

convertToVoltage (int instance, unsigned int adcCh, uint16_t value)

This function/VI returns the voltage of a give ADC value of channel **adcCh** (values 0-3) and **value**. Connect this function/VI with **extractFromInputPacket_ADC** where function return is value of **convertToVoltage**.

Instance is the number relating to a specific SADI connected to the system. If only one device is connected this number will be 0

Return Type: double

Appendix I

FT_STATUS

Error Code	Value
FT_OK	0
FT_INVALID HANDLE	1
FT_DEVICE_NOT_FOUND	2
FT_DEVICE_NOT_OPENED	3
FT_IO_ERROR	4
FT_INSUFFICIENT_RESOURCES	5
FT_INVALID_PARAMETER	6
FT_INVALID_BAUD_RATE	7
FT_DEVICE_NOT_OPENED_FOR_ERASE	8
FT_DEVICE_NOT_OPENED_FOR_WRITE	9
FT_FAILED_TO_WRITE_DEVICE	10
FT_EEPROM_READ_FAILED	11
FT_EEPROM_WRITE_FAILED	12
FT_EEPROM_ERASE_FAILED	13
FT_EEPROM_NOT_PRESENT	14
FT_EEPROM_NOT_PROGRAMMED	15
FT_INVALID_ARGS	16
FT_NOT_SUPPORTED	17
FT_OTHER_ERROR	18
FT_DEVICE_LIST_NOT_READY	19