

Numeric Data Types Table

Updated 2026-03-31 3 minute(s) read # LabVIEW # User Manual

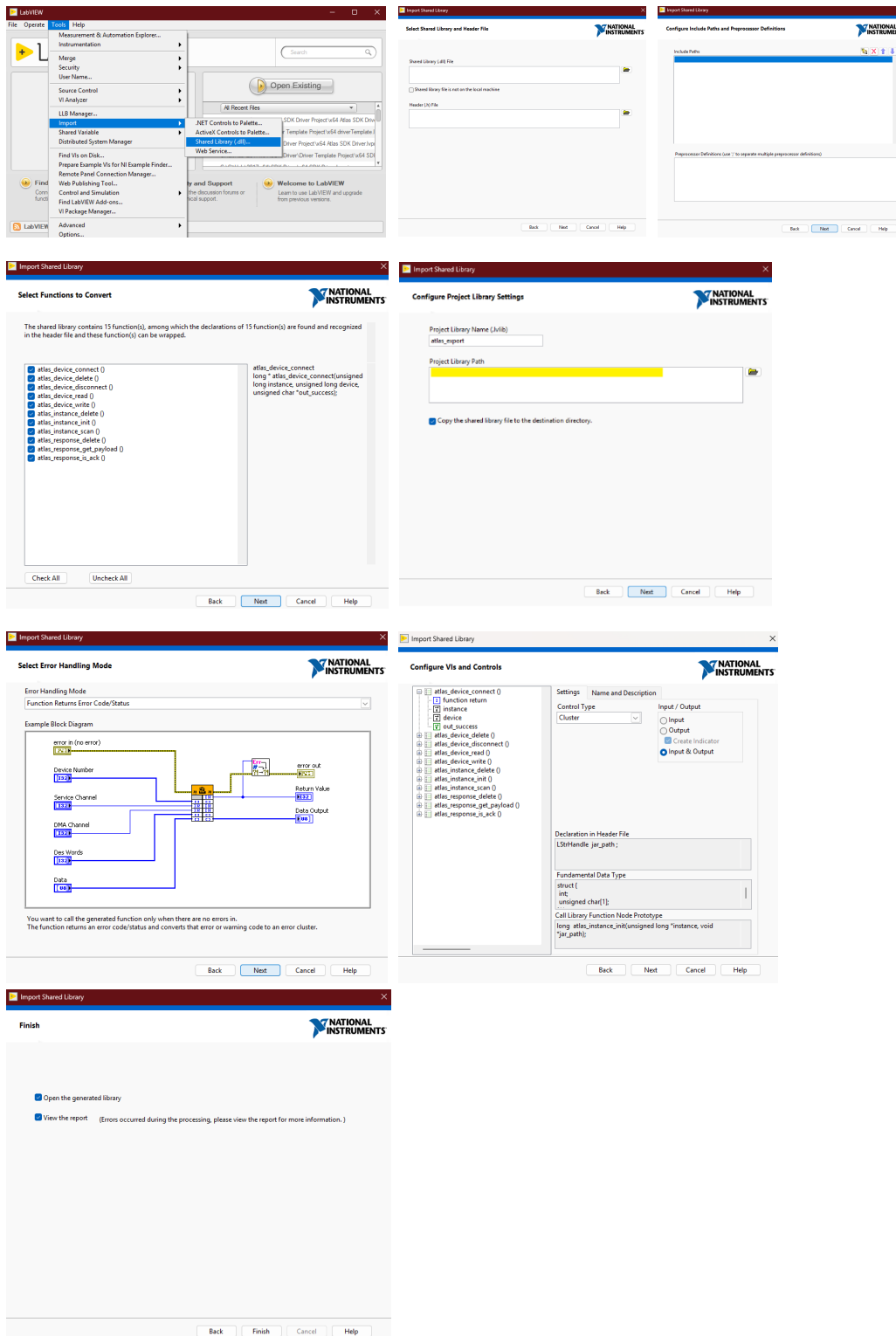
The following table displays the numeric data types available in LabVIEW. LabVIEW stores each data type in a different way.

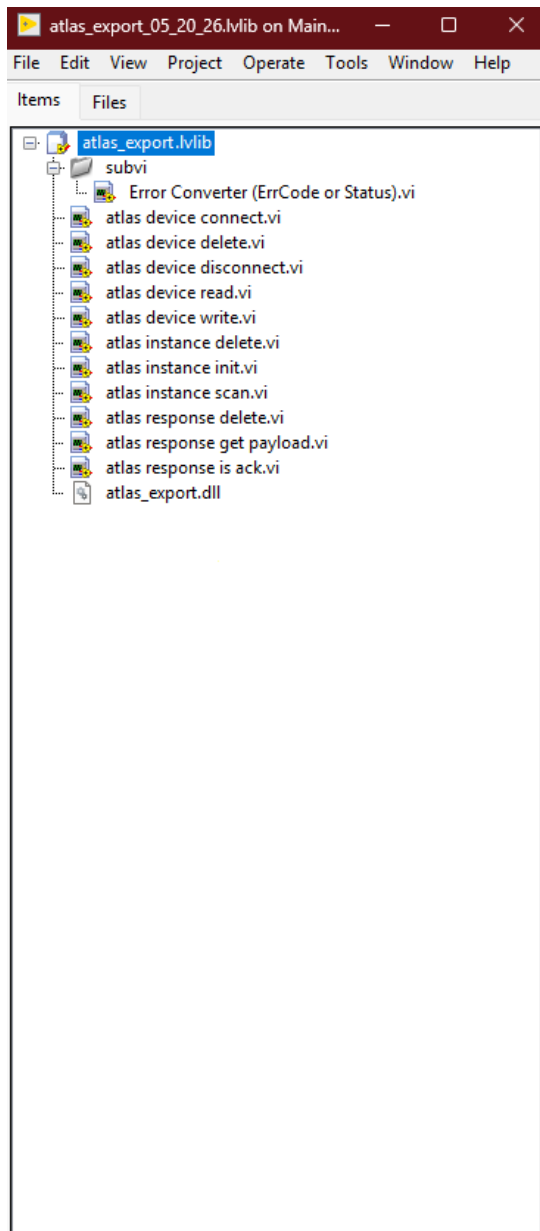
Terminal	Numeric Data Type	Bits of Storage on Disk	Approximate Number of Decimal Digits	Approximate Range
	Single-precision, floating-point	32	6	Minimum positive number: 1.40e-45 Maximum positive number: 3.40e+38 Minimum negative number: -1.40e-45 Maximum negative number: -3.40e+38
	Double-precision, floating-point	64	15	Minimum positive number: 4.94e-324 Maximum positive number: 1.79e+308 Minimum negative number: -4.94e-324 Maximum negative number: -1.79e+308
	Extended-precision, floating-point	128	varies from 15 to 20 by platform	Minimum positive number: 6.48e-4966 Maximum positive number: 1.19e+4932 Minimum negative number: -6.48e-4966 Maximum negative number: -1.19e+4932
	Complex single-precision, floating-point	64	6	Same as single-precision, floating-point for each (real and imaginary) part
	Complex double-precision, floating-point	128	15	Same as double-precision, floating-point for each (real and imaginary) part
	Complex extended-precision, floating-point	256	varies from 15 to 20 by platform	Same as extended-precision, floating-point for each (real and imaginary) part
	Fixed-point	64, or 72 if you include an overflow status	varies by user configuration	varies by user configuration
	Byte signed integer	8	2	-128 to 127

	Word signed integer	16	4	-32,768 to 32,767
	Long signed integer	32	9	-2,147,483,648 to 2,147,483,647
	Quad signed integer	64	18	-1e19 to 1e19
	Byte unsigned integer	8	2	0 to 255
	Word unsigned integer	16	4	0 to 65,535
	Long unsigned integer	32	9	0 to 4,294,967,295
	Quad unsigned integer	64	19	0 to 2e19
	128-bit time stamp	128	19	Minimum time: 01/01/1600 00:00:00 UTC maximum time: 01/01/3001 00:00:00 UTC

Atlas LabVIEW vi configuration settings (v1.0.0 5-14-2006)

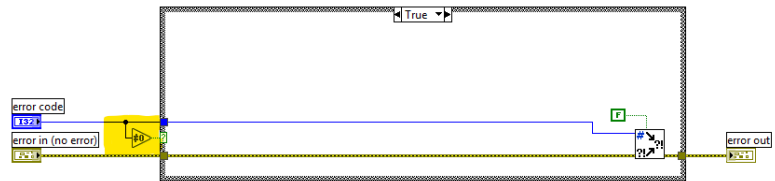
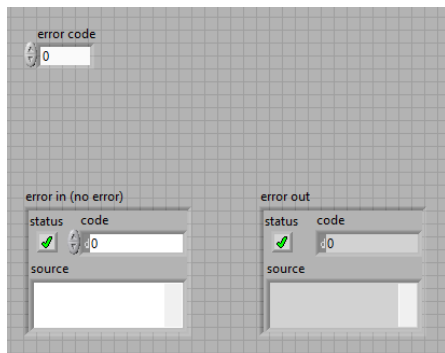
Importing the Atlas dll



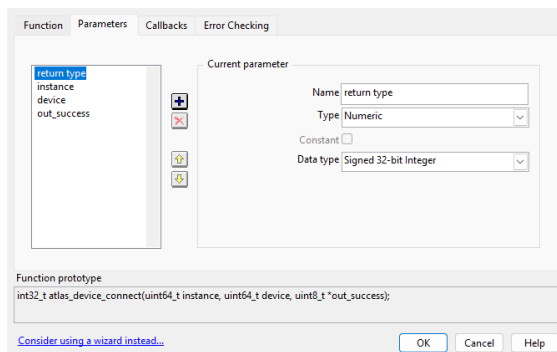
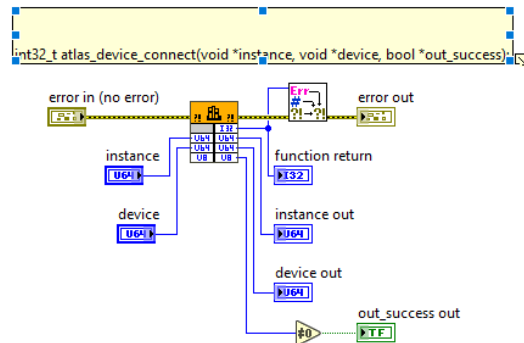
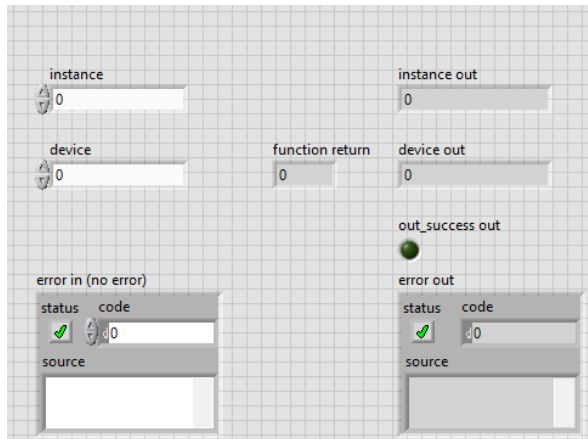


For Simplicity – just import everything and modify each vi – below are working vis

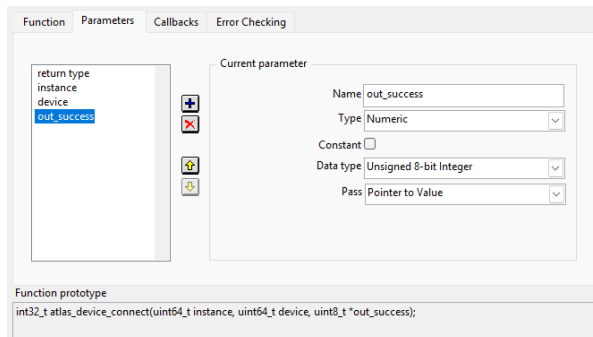
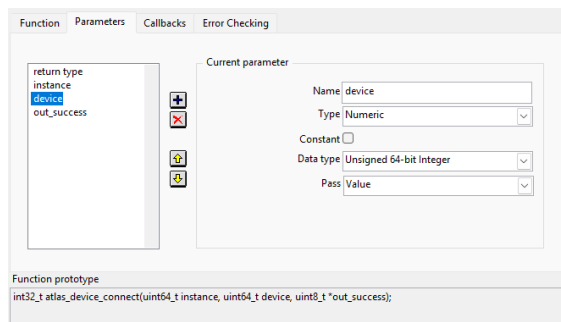
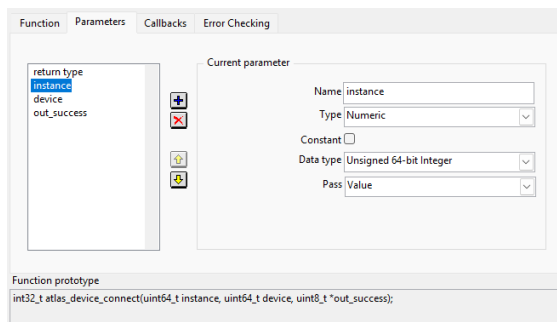
Error Converter (ErrCode or Status).vi “True” case as below



atlas device connect.vi



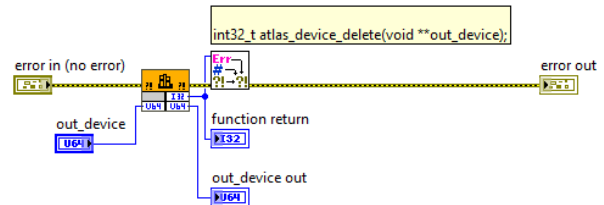
Return Type is the same configuration for all atlas vis



atlas device delete.vi

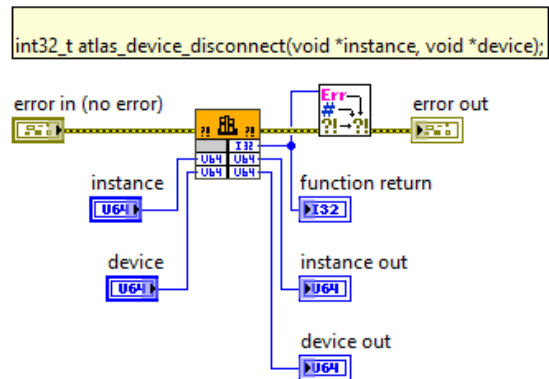
out_device		out_device out	
0		0	
error in (no error)		function return	
0		0	
status code		status code	
0		0	
source		source	

Function	Parameters	Callbacks	Error Checking
return type out_device	Current parameter Name: out_device Type: Numeric Constant: ☐ Data type: Unsigned 64-bit Integer Pass: Pointer to Value		
Function prototype int32_t atlas_device_delete(uint64_t *out_device);			



atlas device disconnect.vi

instance	0	instance out	0
device	0	device out	0
function return		0	
error in (no error)		error out	
status	code	status	code
✓	0	✓	0
source		source	



Function	Parameters	Callbacks	Error Checking
return type	Current parameter		
instance	Name: instance		
device	Type: Numeric		
	Constant: ☐		
	Data type: Unsigned 64-bit Integer		
	Pass: Value		
Function prototype			
int32_t atlas_device_disconnect(uint64_t instance, uint64_t device);			

Function	Parameters	Callbacks	Error Checking
return type	Current parameter		
instance	Name: device		
device	Type: Numeric		
	Constant: ☐		
	Data type: Unsigned 64-bit Integer		
	Pass: Value		
Function prototype			
int32_t atlas_device_disconnect(uint64_t instance, uint64_t device);			

atlas device read.vi

instance: 0

device: 0

address: [] length: 0

local: mem access

routing bit: tool

error in (no error): status: [] code: 0 source: []

instance out: 0

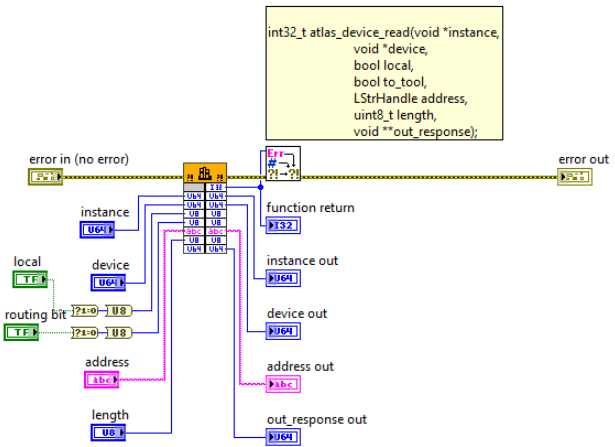
device out: 0

address out: []

out_response out: 0

function return: 0

error out: status: [] code: 0 source: []



Function: Parameters: Callbacks: Error Checking:

return type: instance, device, local, to_tool, address, length, out_response

Current parameter: Name: instance, Type: Numeric, Constant: [], Data type: Unsigned 64-bit Integer, Pass: Value

Function prototype: int32_t atlas_device_read(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, uint8_t length, uint64_t *out_response);

Function: Parameters: Callbacks: Error Checking:

return type: instance, device, local, to_tool, address, length, out_response

Current parameter: Name: device, Type: Numeric, Constant: [], Data type: Unsigned 64-bit Integer, Pass: Value

Function prototype: int32_t atlas_device_read(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, uint8_t length, uint64_t *out_response);

Function: Parameters: Callbacks: Error Checking:

return type: instance, device, local, to_tool, address, length, out_response

Current parameter: Name: local, Type: Numeric, Constant: [], Data type: Unsigned 8-bit Integer, Pass: Value

Function prototype: int32_t atlas_device_read(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, uint8_t length, uint64_t *out_response);

Function: Parameters: Callbacks: Error Checking:

return type: instance, device, local, to_tool, address, length, out_response

Current parameter: Name: to_tool, Type: Numeric, Constant: [], Data type: Unsigned 8-bit Integer, Pass: Value

Function prototype: int32_t atlas_device_read(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, uint8_t length, uint64_t *out_response);

Function: Parameters: Callbacks: Error Checking:

return type: instance, device, local, to_tool, address, length, out_response

Current parameter: Name: address, Type: String, Constant: [], String format: String Handle

Function prototype: int32_t atlas_device_read(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, uint8_t length, uint64_t *out_response);

Function: Parameters: Callbacks: Error Checking:

return type: instance, device, local, to_tool, address, length, out_response

Current parameter: Name: length, Type: Numeric, Constant: [], Data type: Unsigned 8-bit Integer, Pass: Value

Function prototype: int32_t atlas_device_read(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, uint8_t length, uint64_t *out_response);

The screenshot shows the 'Parameters' tab in the IDE. The 'Current parameter' section displays 'out_response' with a 'Numeric' type. The 'Function prototype' section shows the function signature: `int32_t atlas_device_read(uint8_t *instance, uint64_t device, uint8_t *local, uint8_t *to_tool, LStrHandle address, uint8_t *length, uint64_t *out_response);`

atlas device write.vi

instance 0

device 0

address

payload

local mem access

routing bit tool

function return 0

address out

payload out

out_response out 0

error in (no error)

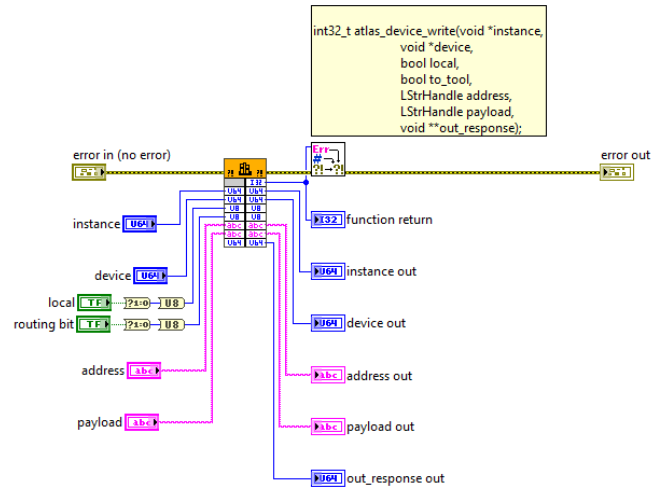
status code

source

error out

status code

source



Function Parameters Callbacks Error Checking

return type

instance

device

local

to_tool

address

payload

out_response

Current parameter

Name instance

Type Numeric

Constant

Data type Unsigned 64-bit Integer

Pass Value

Function prototype

int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

Function Parameters Callbacks Error Checking

return type

instance

device

local

to_tool

address

payload

out_response

Current parameter

Name device

Type Numeric

Constant

Data type Unsigned 64-bit Integer

Pass Value

Function prototype

int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

Function Parameters Callbacks Error Checking

return type

instance

device

local

to_tool

address

payload

out_response

Current parameter

Name local

Type Numeric

Constant

Data type Unsigned 8-bit Integer

Pass Value

Function prototype

int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

Function Parameters Callbacks Error Checking

return type

instance

device

local

to_tool

address

payload

out_response

Current parameter

Name to_tool

Type Numeric

Constant

Data type Unsigned 8-bit Integer

Pass Value

Function prototype

int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

FunctionParametersCallbacksError Checking

return type

instance

device

local

to_tool

address

payload

out_response

+

×

↕

↕

Current parameter

Nameaddress

TypeString

Constant☐

String formatString Handle

Function prototype

int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

FunctionParametersCallbacksError Checking

return type

instance

device

local

to_tool

address

payload

out_response

+

×

↕

↕

Current parameter

Namepayload

TypeString

Constant☐

String formatString Handle

Function prototype

int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

FunctionParametersCallbacksError Checking

return type

instance

device

local

to_tool

address

payload

out_response

+

×

↕

↕

Current parameter

Nameout_response

TypeNumeric

Constant☐

Data typeUnsigned 64-bit Integer

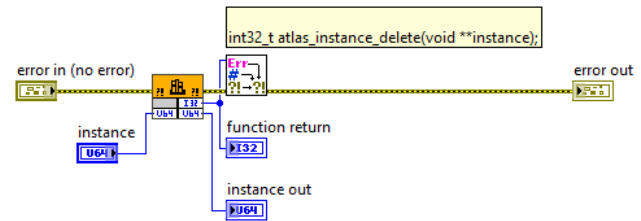
PassPointer to Value

Function prototype

int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

atlas instance delete.vi

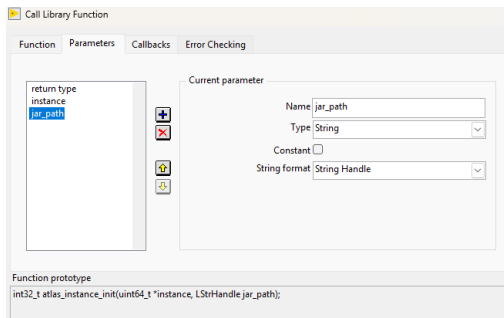
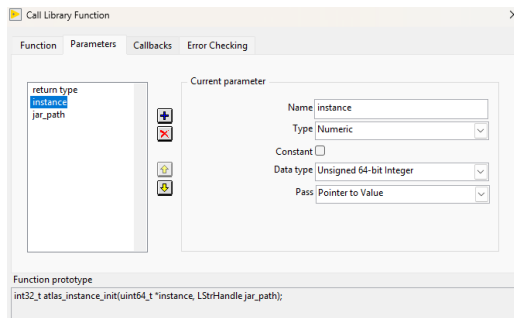
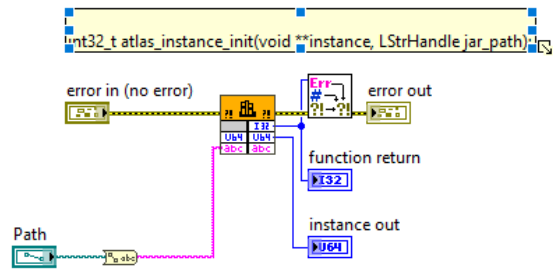
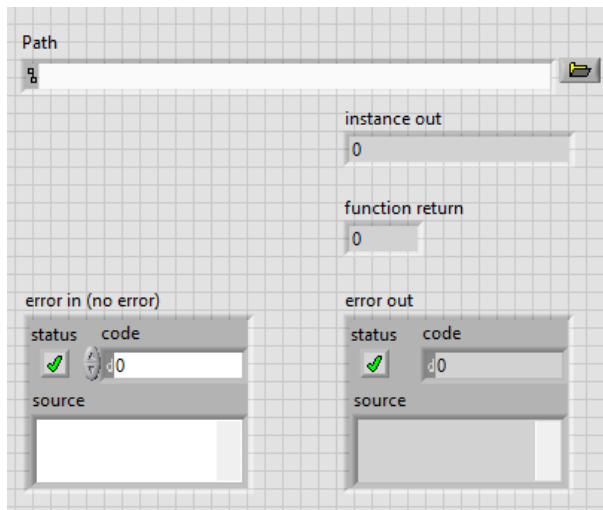
instance		instance out	
0		0	
error in (no error)		error out	
status	code	status	code
✓	0	✓	0
source		source	



Function	Parameters	Callbacks	Error Checking
return type instance			
Current parameter			
Name: instance			
Type: Numeric			
Constant: ☐			
Data type: Unsigned 64-bit Integer			
Pass: Pointer to Value			
Function prototype <code>int32_t atlas_instance_delete(uint64_t *instance);</code>			

atlas instance init.vi

- added Path and path to string



atlas instance scan.vi

instance: 0

instance out: 0

mpbid:

mpbid out:

out_device out: 0

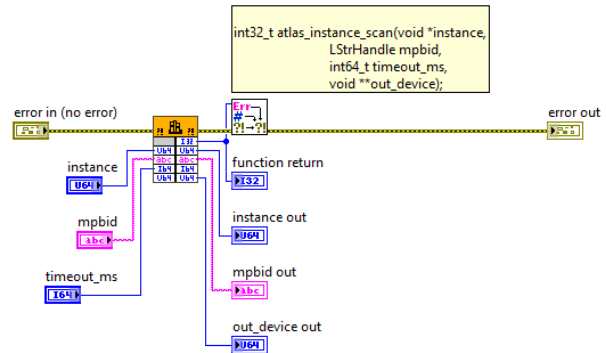
timeout_ms: 3000

function return: 0

error in (no error): status 0, code 0

error out: status 0, code 0

source:



Function Parameters Callbacks Error Checking

return type: instance, mpbid, timeout_ms, out_device

Current parameter: Name: instance, Type: Numeric, Constant: ☐, Data type: Unsigned 64-bit Integer, Pass: Value

Function prototype: `int32_t atlas_instance_scan(uint64_t instance, LStrHandle mpbid, int64_t timeout_ms, uint64_t *out_device);`

Function Parameters Callbacks Error Checking

return type: instance, mpbid, timeout_ms, out_device

Current parameter: Name: mpbid, Type: String, Constant: ☐, String format: String Handle

Function prototype: `int32_t atlas_instance_scan(uint64_t instance, LStrHandle mpbid, int64_t timeout_ms, uint64_t *out_device);`

Function Parameters Callbacks Error Checking

return type: instance, mpbid, timeout_ms, out_device

Current parameter: Name: timeout_ms, Type: Numeric, Constant: ☐, Data type: Signed 64-bit Integer, Pass: Value

Function prototype: `int32_t atlas_instance_scan(uint64_t instance, LStrHandle mpbid, int64_t timeout_ms, uint64_t *out_device);`

Function Parameters Callbacks Error Checking

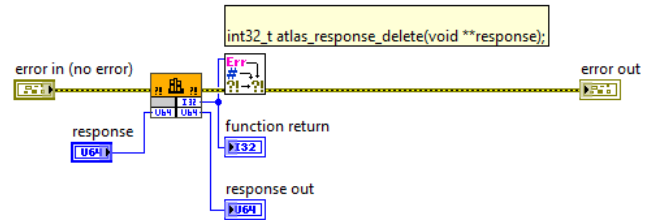
return type: instance, mpbid, timeout_ms, out_device

Current parameter: Name: out_device, Type: Numeric, Constant: ☐, Data type: Unsigned 64-bit Integer, Pass: Pointer to Value

Function prototype: `int32_t atlas_instance_scan(uint64_t instance, LStrHandle mpbid, int64_t timeout_ms, uint64_t *out_device);`

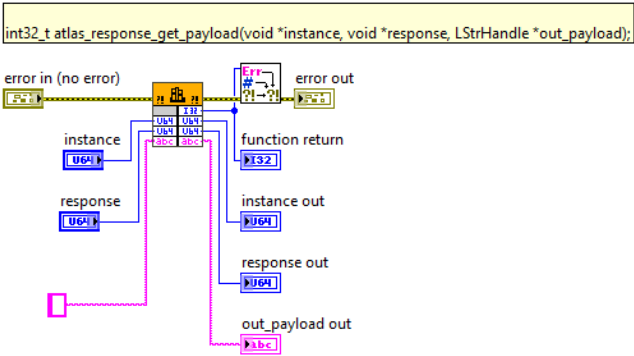
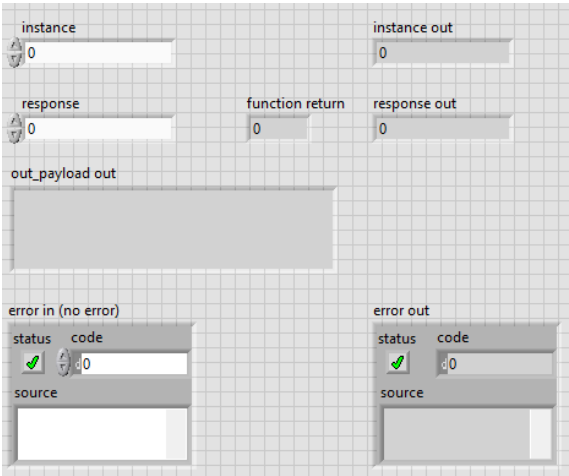
atlas response delete.vi

response		response out	
0		0	
error in (no error)		error out	
status	code	status	code
☒	0	☒	0
source		source	



Function	Parameters	Callbacks	Error Checking
return type response			
Current parameter			
Name: response			
Type: Numeric			
Constant: ☐			
Data type: Unsigned 64-bit Integer			
Pass: Pointer to Value			
Function prototype <code>int32_t atlas_response_delete(uint64_t *response);</code>			

atlas response get payload.vi



Function Parameters Callbacks Error Checking

return type
instance
response
out_payload

Current parameter

Name: instance
Type: Numeric
Constant: ☐
Data type: Unsigned 64-bit Integer
Pass: Value

Function prototype
int32_t atlas_response_get_payload(uint64_t instance, uint64_t response, LStrHandle *out_payload);

Function Parameters Callbacks Error Checking

return type
instance
response
out_payload

Current parameter

Name: response
Type: Numeric
Constant: ☐
Data type: Unsigned 64-bit Integer
Pass: Value

Function prototype
int32_t atlas_response_get_payload(uint64_t instance, uint64_t response, LStrHandle *out_payload);

Function Parameters Callbacks Error Checking

return type
instance
response
out_payload

Current parameter

Name: out_payload
Type: String
Constant: ☐
String format: String Handle Pointer

Function prototype
int32_t atlas_response_get_payload(uint64_t instance, uint64_t response, LStrHandle *out_payload);

atlas response is ack.vi

instance

0

response

0

function return

0

response out

0

out_is_ack out

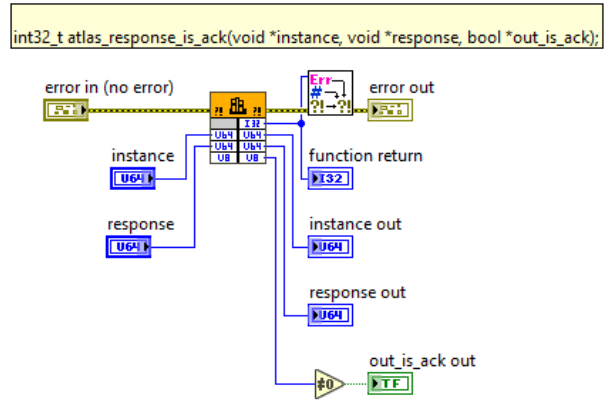
error in (no error)

status code

✓ 0

source

source



The screenshot shows the 'Parameters' tab in the IDE. The 'Current parameter' section has the following settings:

- Name: instance
- Type: Numeric
- Constant: ☐
- Data type: Unsigned 64-bit Integer
- Pass: Value

The 'Function prototype' section shows the following signature:

```
int32_t atlas_response_is_ack(uint64_t instance, uint64_t response, uint8_t *out_is_ack);
```

Function Parameters Callbacks Error Checking

return type
instance
response
out_is_ack

Current parameter

Name: response

Type: Numeric

Constant: ☐

Data type: Unsigned 64-bit Integer

Pass: Value

Function prototype

```
int32_t atlas_response_is_ack(uint64_t instance, uint64_t response, uint8_t* out_is_ack);
```

Atlas variable definitions

instance – user handle for atlas instance

device – atlas device handle tied to atlas instance

response – atlas response handle tied to atlas device

Order of Execution

- 1) Initialize an instance of Atlas
 - a. Call “atlas instance init.vi”
- 2) Scan for a tool
 - a. Call “atlas instance scan.vi”
 - i. using “instance” returned from init vi above as input
 - ii. using valid mpbid in nearby area as input
 - iii. using timeout value in mS as input
- 3) Establish a connection
 - a. Call “atlas device connect.vi”
 - i. using “instance” from above and “device” returned from scan vi
- 4) Read data – follow all steps to ensure pc memory is properly deallocated
 - a. Call “atlas device read.vi”
 - i. using “instance” and “device” from above
 - ii. using valid address as input (hex format)
 - iii. using valid length for address as input (hex format)
 1. example mpbid <address = 0004 / length = 05>
 - iv. set “local” input for “mem access” or “object” oriented mem read
 1. True = mem access
 2. False = object oriented
 - v. set “routing bit” input for “tool” or “bgm”
 1. True = tool
 2. False = bgm
 - b. Call “atlas response is ack.vi”
 - i. using “instance” from above and “response” returned from atlas device read vi
 - c. Call “atlas response get payload.vi”
 - i. using “instance” and “response” from above
 - ii. payload out is data returned
 - d. Call “atlas response delete.vi”
 - i. using “response” from above

- 5) Write data - follow all steps to ensure pc memory is properly deallocated
 - a. Call "atlas device write.vi"
 - i. using "instance" and "device" from above
 - ii. using valid address as input (hex format)
 - iii. using valid payload as input (hex format)
 1. example blink LED <address = A023 / payload = 01>
 - iv. set "local" input for "mem access" or "object" oriented mem write
 1. True = mem access
 2. False = object oriented
 - v. set "routing bit" input for "tool" or "bgm"
 1. True = tool
 2. False = bgm
 - b. Call "atlas response is ack.vi"
 - i. using "instance" from above and "response" returned from atlas device read vi
 - c. Call "atlas response delete.vi"
 - i. using "response" from above
- 6) Disconnect
 - a. Call "atlas device disconnect.vi"
 - i. using "instance" and "device" from above
- 7) Teardown – to destroy atlas instance
 - a. Call "atlas device delete.vi"
 - i. using "device" from above
 - b. Call "atlas instance delete.vi"
 - i. using "instance" from above