Numeric Data Types Table

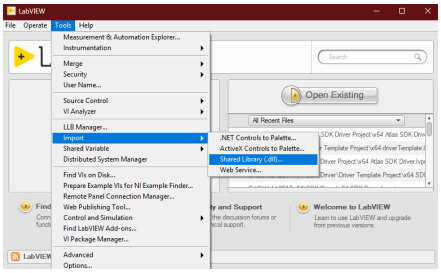
Updated 2026-03-31 | 3 minute(s) read | # LabVIEW # User Manual

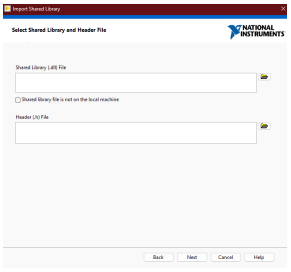
The following table displays the numeric data types available in LabVIEW. LabVIEW stores each data type in a different way.

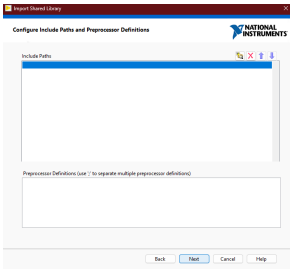
Terminal	Numeric Data Type	Bits of Storage on Disk	Approximate Number of Decimal Digits	Approximate Range
SGL	Single-precision, floating-point	32	6	Minimum positive number: 1.40e−45 Maximum positive number: 3.40e+38 Minimum negative number: −1.40e−45 Maximum negative number: −3.40e+38
DBL	Double-precision, 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
EXT	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
CSG	Complex single-precision, floating-point	64	6	Same as single-precision, floating-point for each (real and imaginary) part
CDB	Complex double-precision, floating-point	128	15	Same as double-precision, floating-point for each (real and imaginary) part
CXT	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
FXP	Fixed-point	64, or 72 if you include an overflow status	varies by user configuration	varies by user configuration
I8	Byte signed integer	8	2	−128 to 127
I16	Word signed integer	16	4	−32,768 to 32,767
I32	Long signed integer	32	9	−2,147,483,648 to 2,147,483,647
I64	Quad signed integer	64	18	−1e19 to 1e19
U8	Byte unsigned integer	8	2	0 to 255
U16	Word unsigned integer	16	4	0 to 65,535
U32	Long unsigned integer	32	9	0 to 4,294,967,295
U64	Quad unsigned integer	64	19	0 to 2e19
TS	128-bit time stamp	128	19	Minimum time: 01/01/1600 00:00:00 UTC maximum time: 01/01/3001 00:00:00 UTC

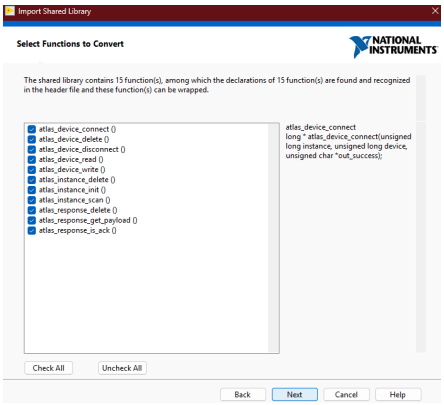
Atlas LabVIEW vi configuration settings (v1.0.0 5-14-2006)

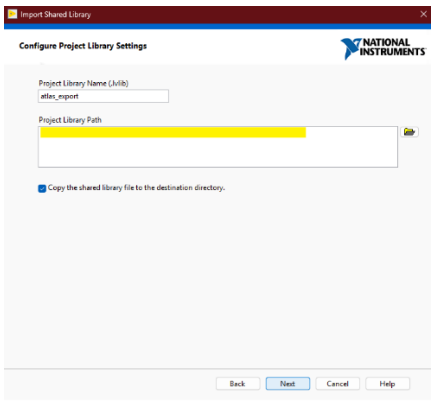
Importing the Atlas dll

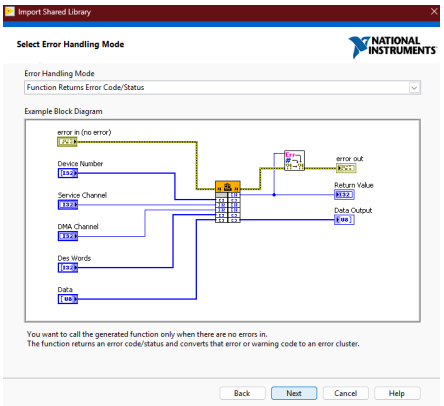


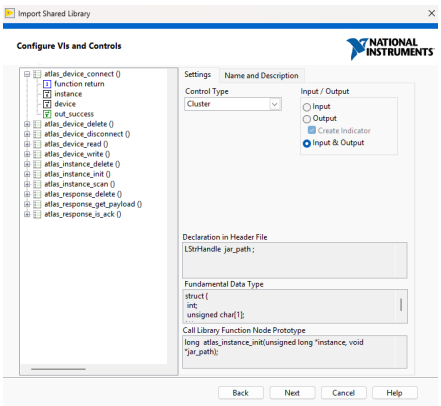


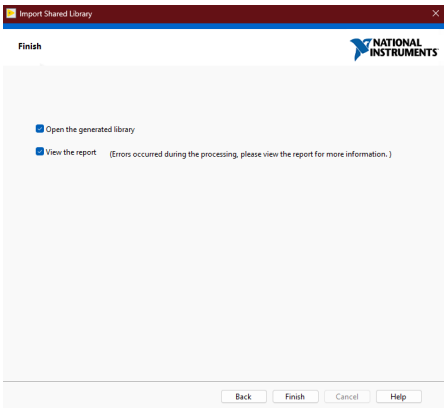


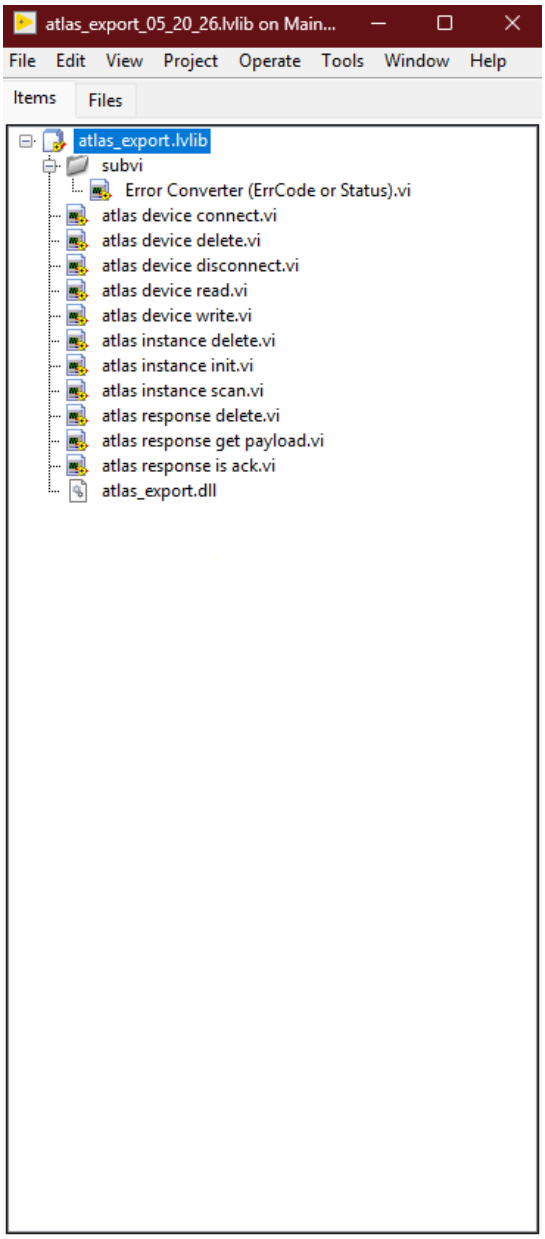






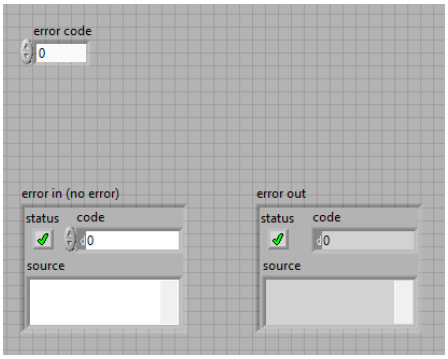


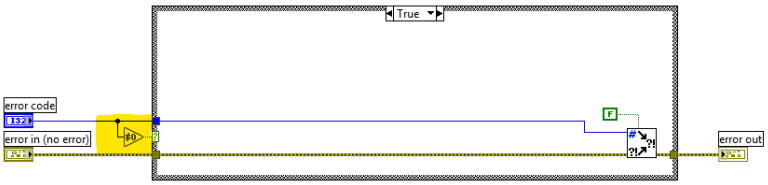



atlas_export_05_20_26.lvlib on Main...
File Edit View Project Operate Tools Window Help
Items Files
atlas_export.lvlib
subvi
Error Converter (ErrCode or Status).vi
atlas device connect.vi
atlas device delete.vi
atlas device disconnect.vi
atlas device read.vi
atlas device write.vi
atlas instance delete.vi
atlas instance init.vi
atlas instance scan.vi
atlas response delete.vi
atlas response get payload.vi
atlas response is ack.vi
atlas_export.dll

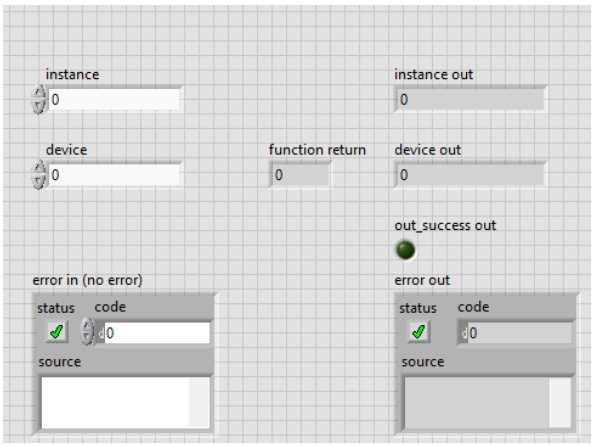
For Simplicity – just import everything and modify each vi – below are working vis

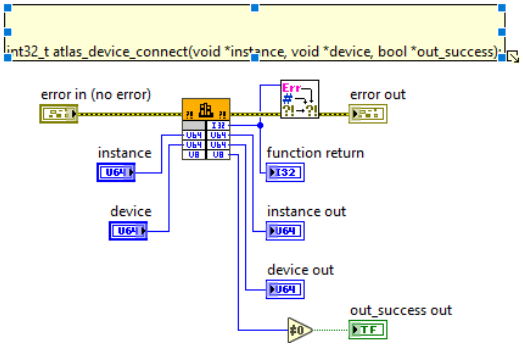
Error Converter (ErrCode or Status).vi “True” case as below

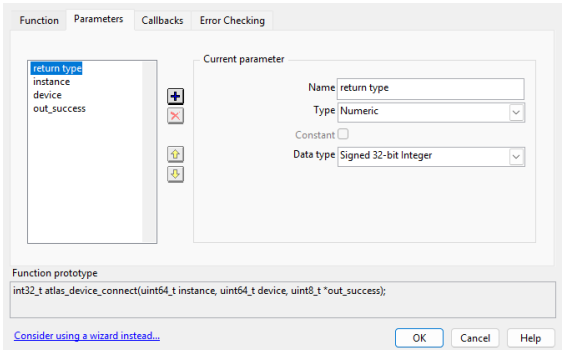




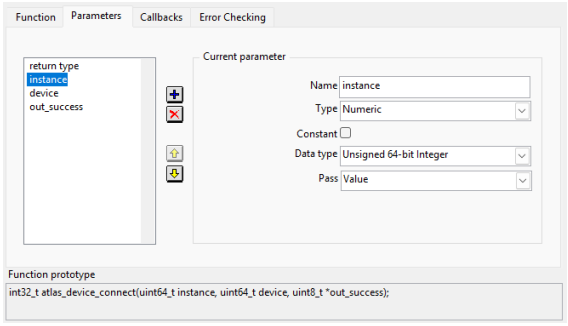
atlas device connect.vi

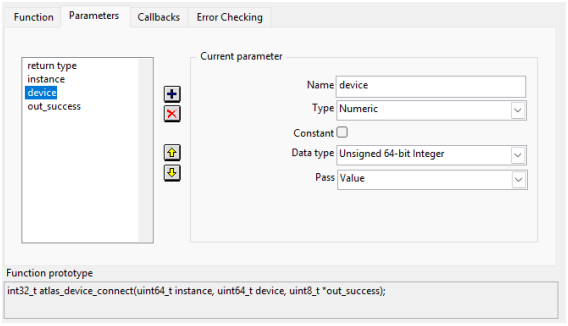


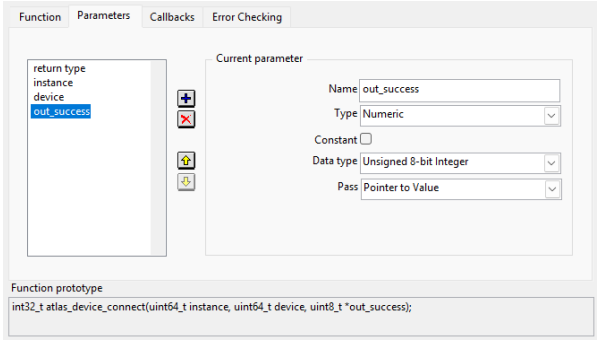




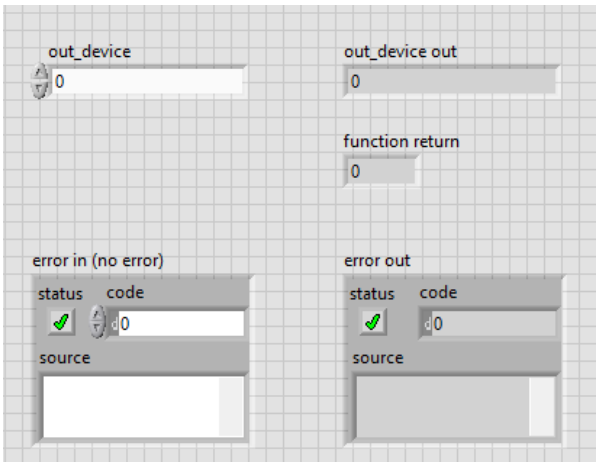
Return Type is the same configuration for all atlas vis

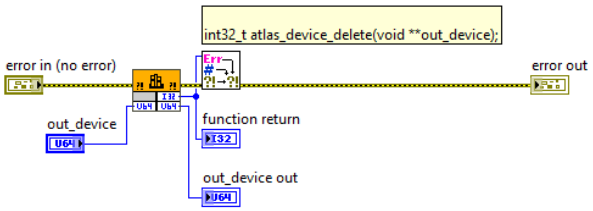


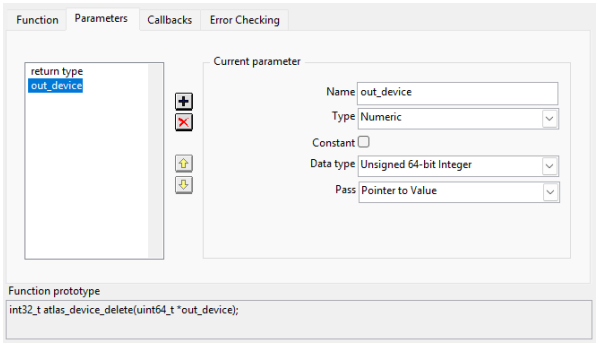




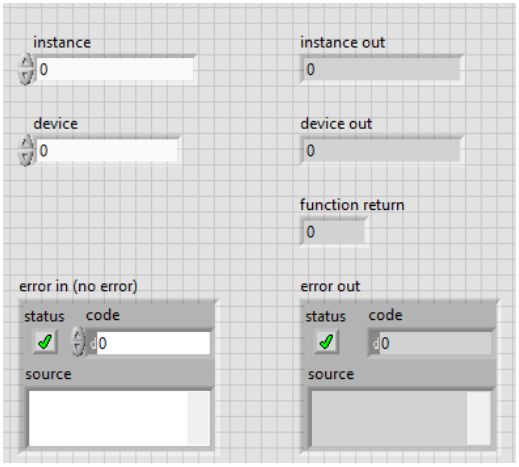
atlas device delete.vi

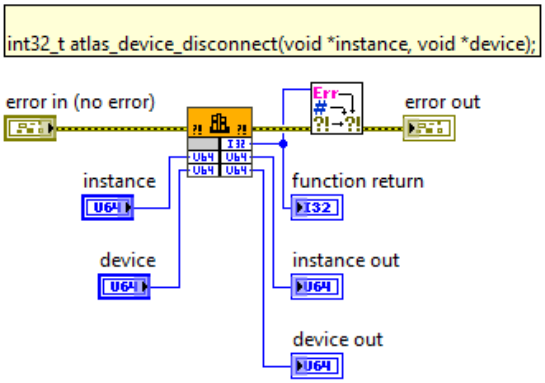


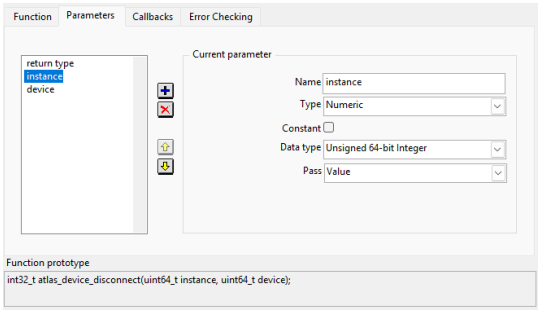


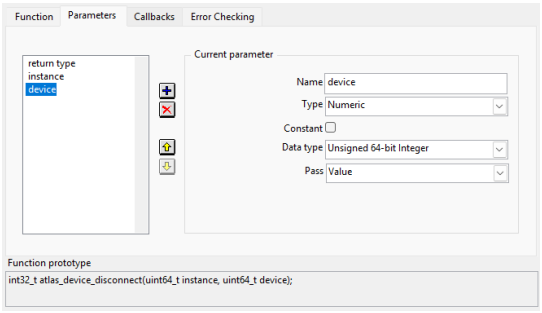


atlas device disconnect.vi

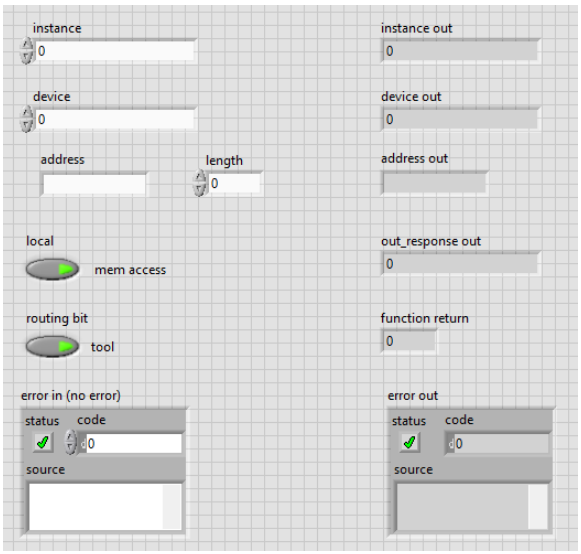


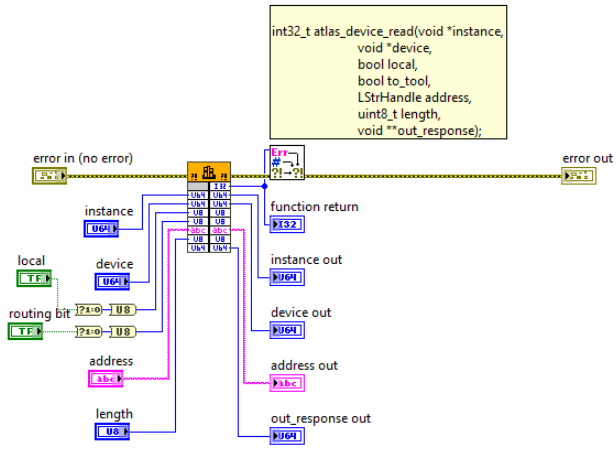


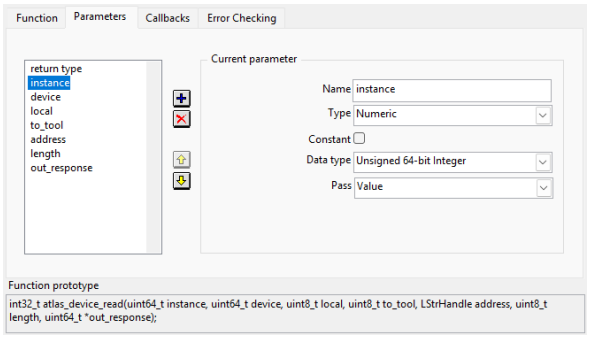


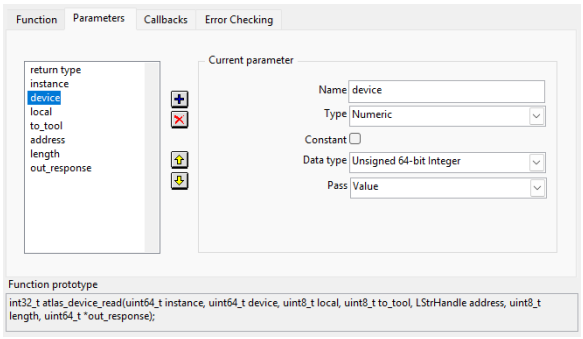


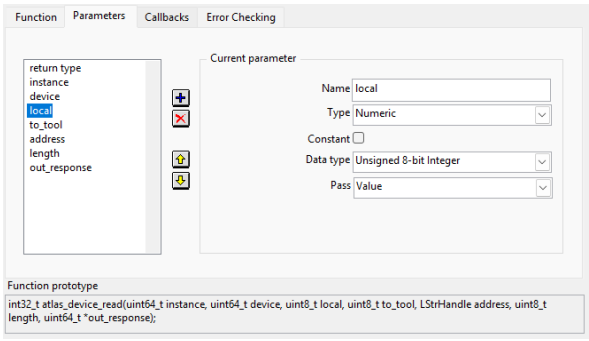
atlas device read.vi

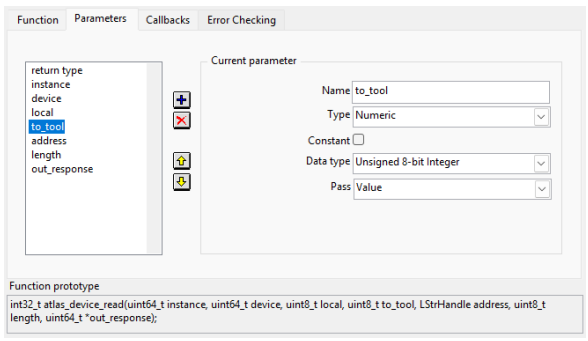


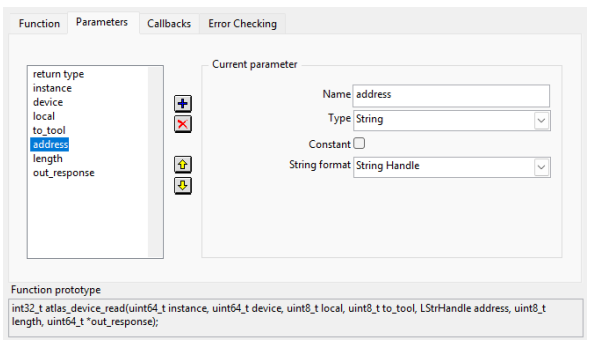


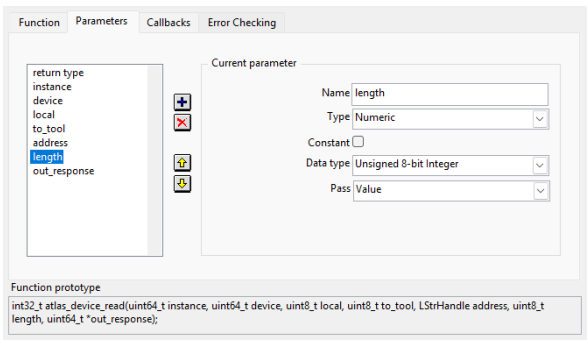


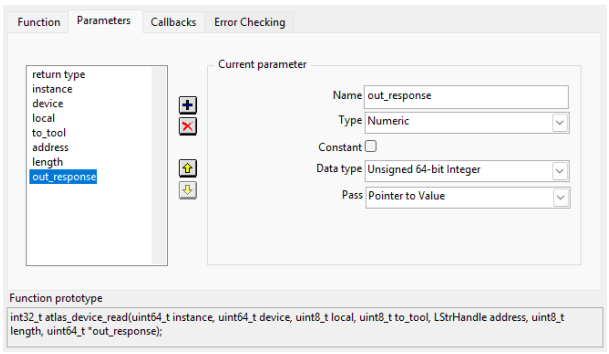




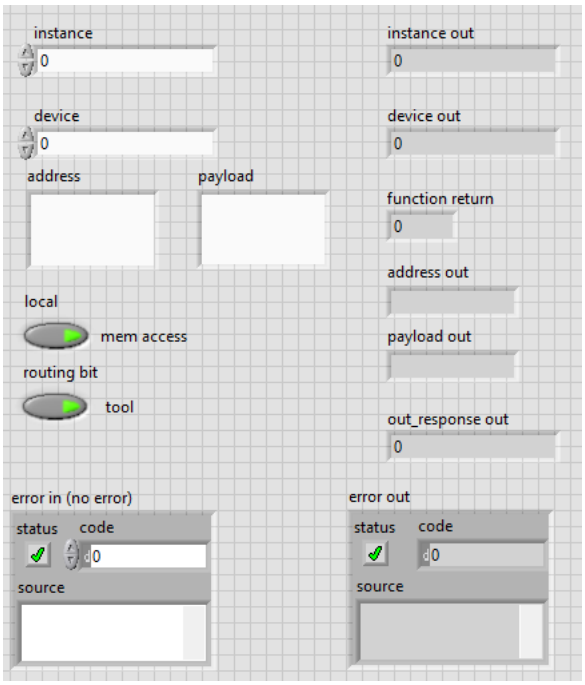


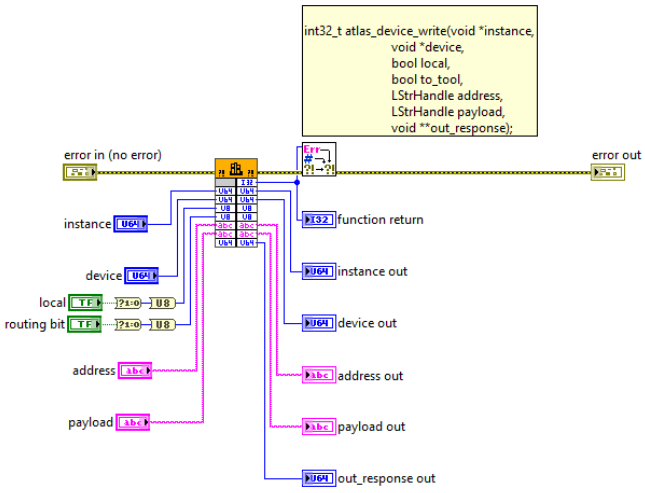


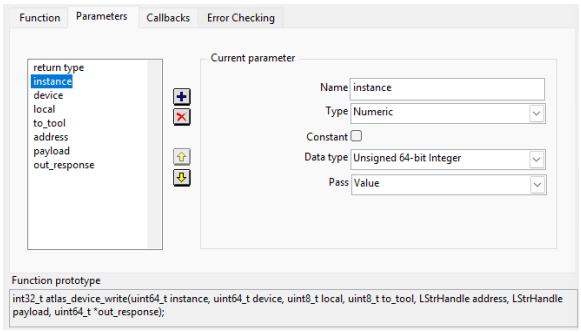


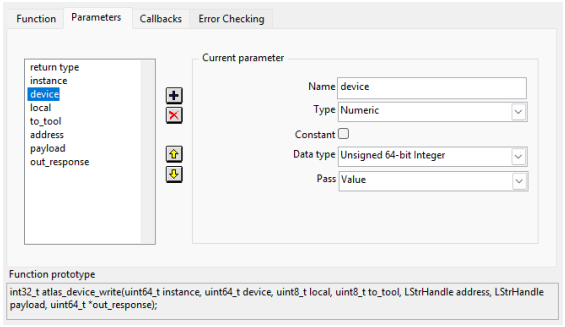

Function
Parameters
Callbacks
Error Checking
return type
instance
device
local
to_tool
address
length
out_response
Current parameter
Name
out_response
Type
Numeric
Constant
Data type
Unsigned 64-bit Integer
Pass
Pointer to Value
Function prototype
int32_t atlas_device_read(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, uint8_t length, uint64_t *out_response);

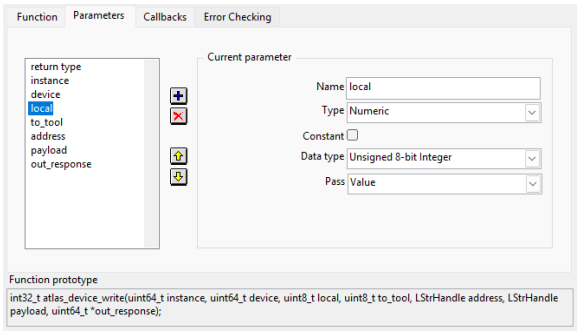
atlas device write.vi

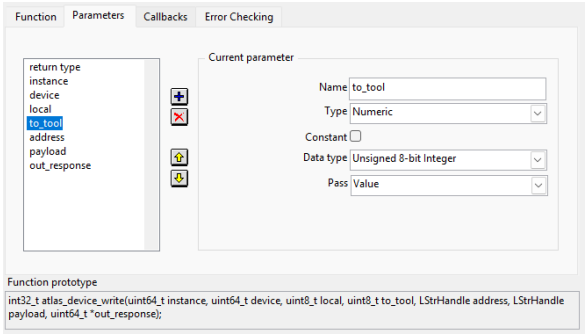


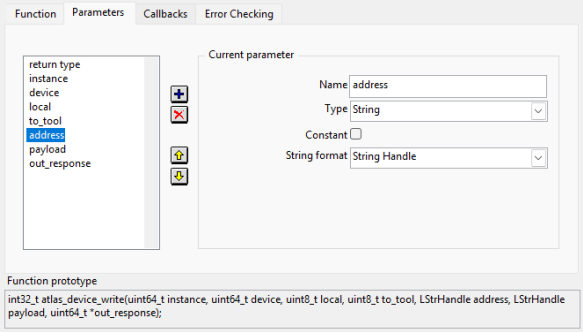


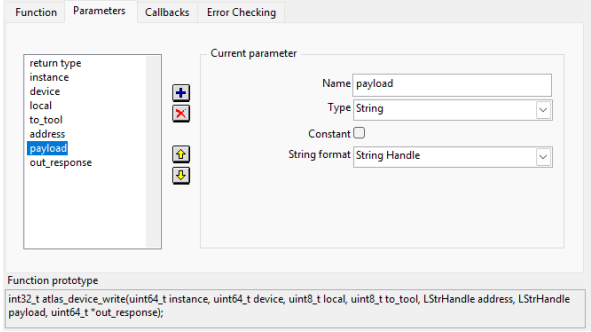


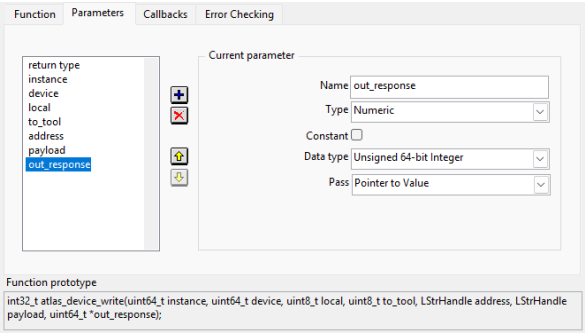




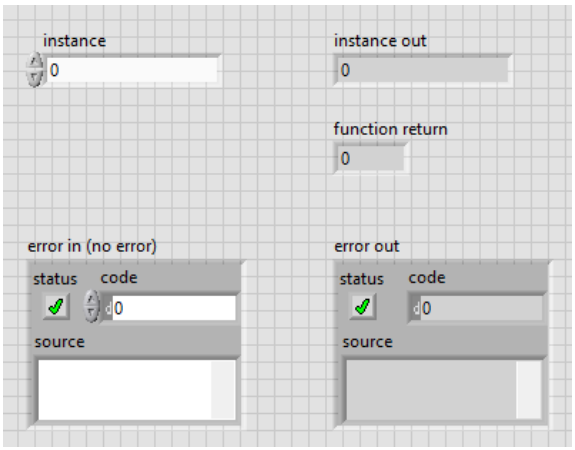


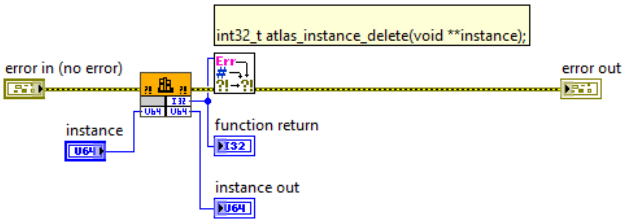

Function
Parameters
Callbacks
Error Checking
return type
instance
device
local
to_tool
address
payload
out_response
Current parameter
Name address
Type String
Constant
String format String Handle
Function prototype
int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);

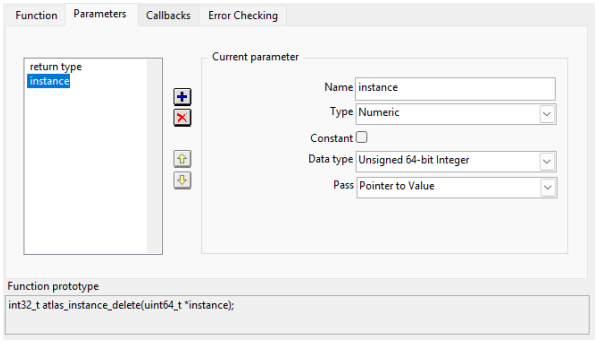

Function
Parameters
Callbacks
Error Checking
return type
instance
device
local
to_tool
address
payload
out_response
Current parameter
Name payload
Type String
Constant
String format String Handle
Function prototype
int32_t atlas_device_write(uint64_t instance, uint64_t device, uint8_t local, uint8_t to_tool, LStrHandle address, LStrHandle payload, uint64_t *out_response);


Function
Parameters
Callbacks
Error Checking
return type
instance
device
local
to_tool
address
payload
out_response
Current parameter
Name out_response
Type Numeric
Constant
Data type Unsigned 64-bit Integer
Pass Pointer 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

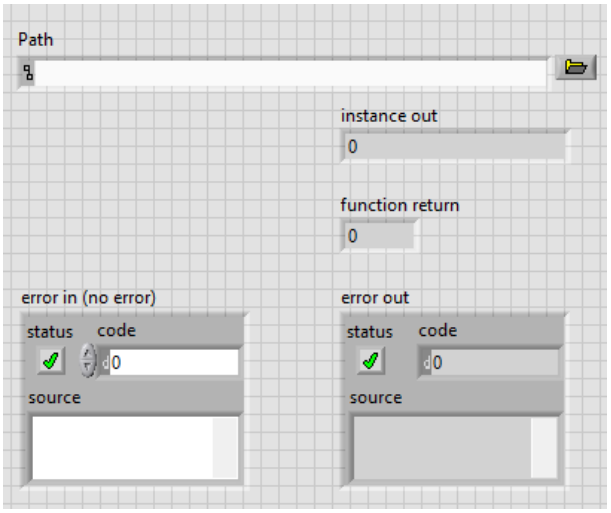


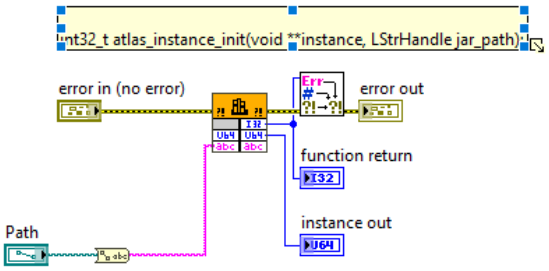


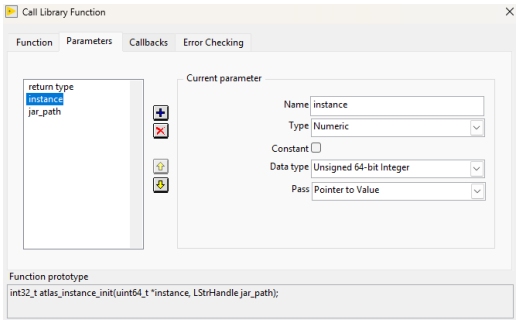


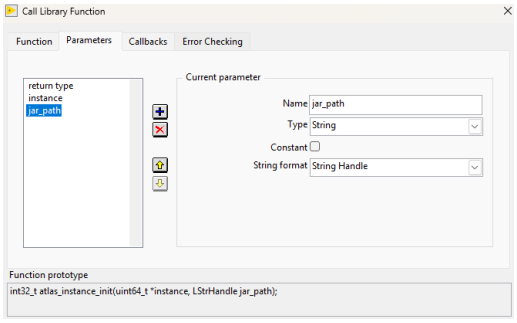
atlas instance init.vi

- added Path and path to string

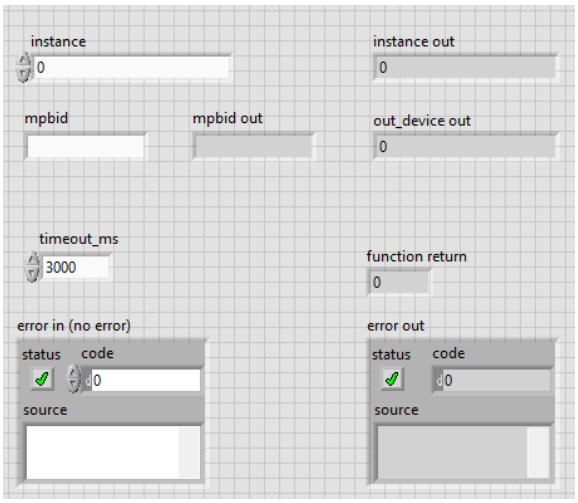


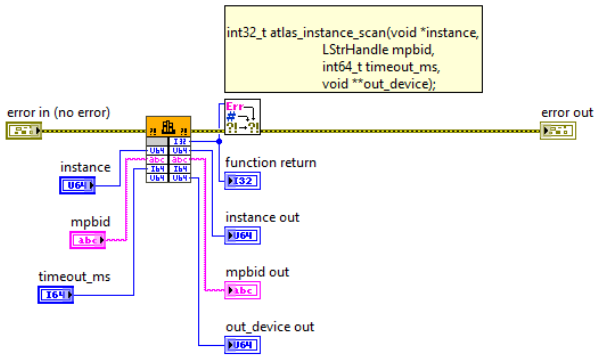


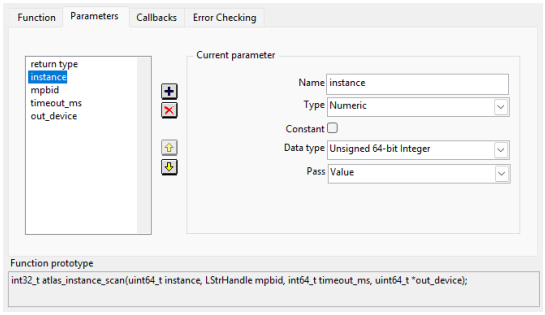


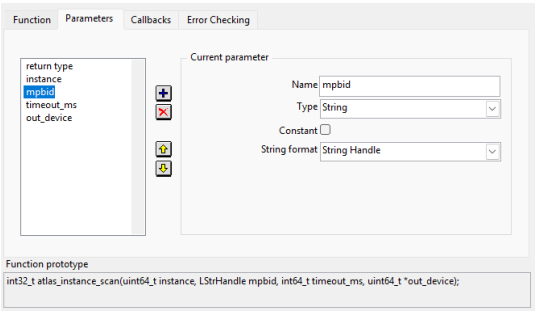


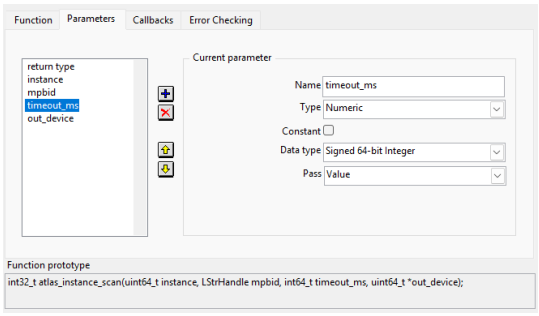
atlas instance scan.vi

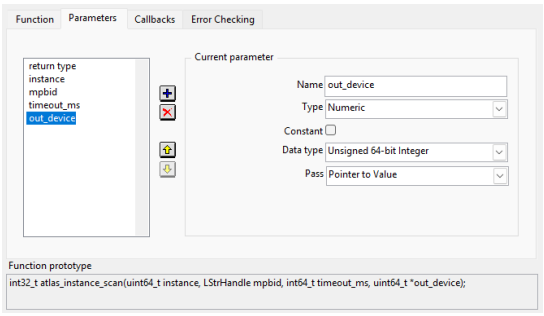




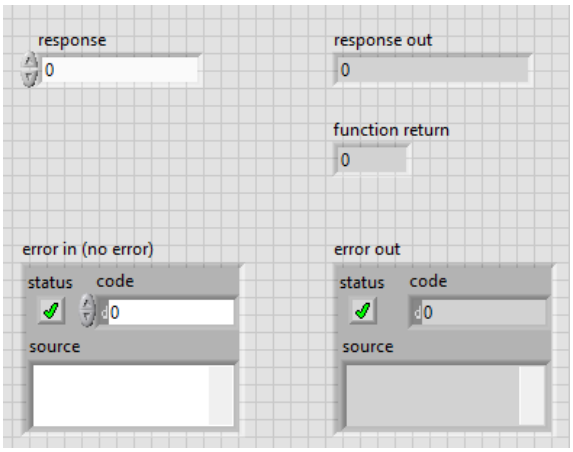


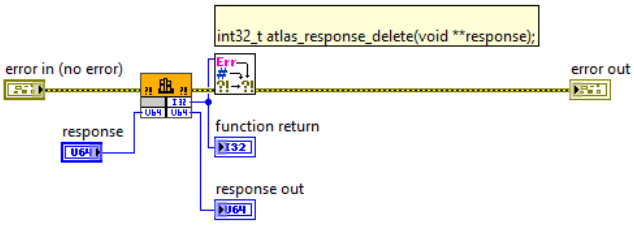


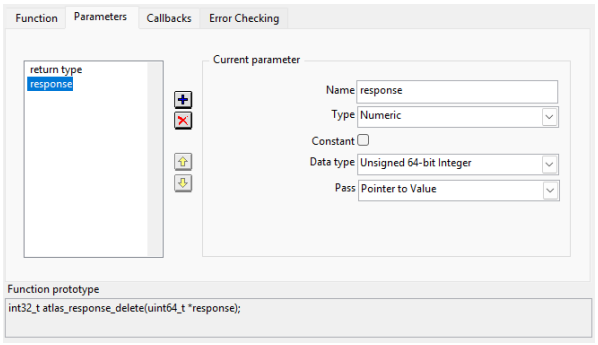




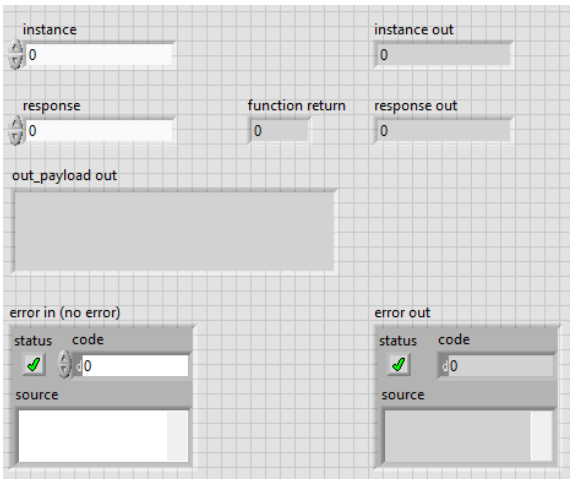
atlas response delete.vi

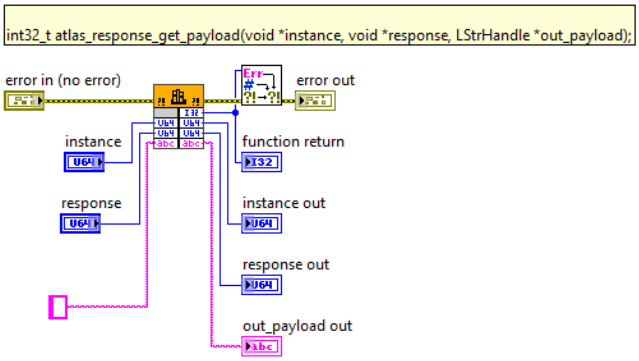


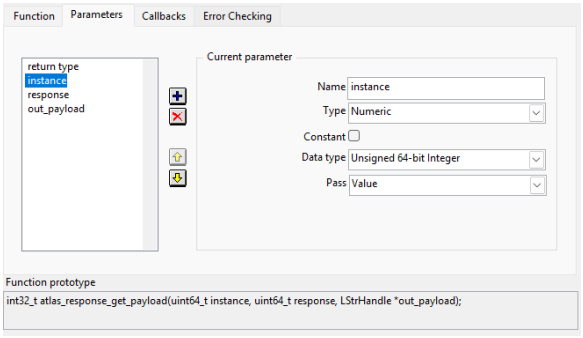


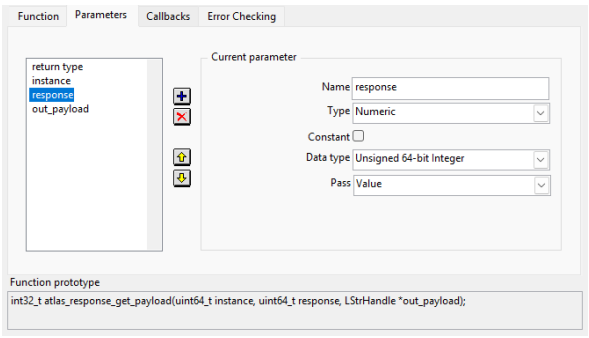


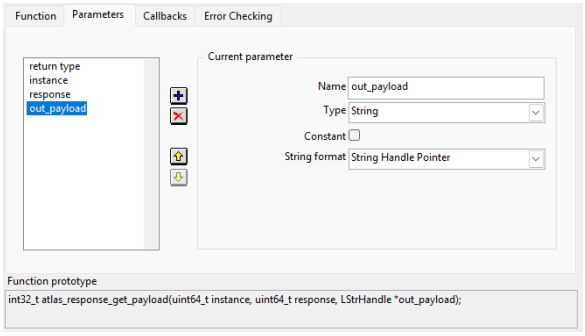
atlas response get payload.vi



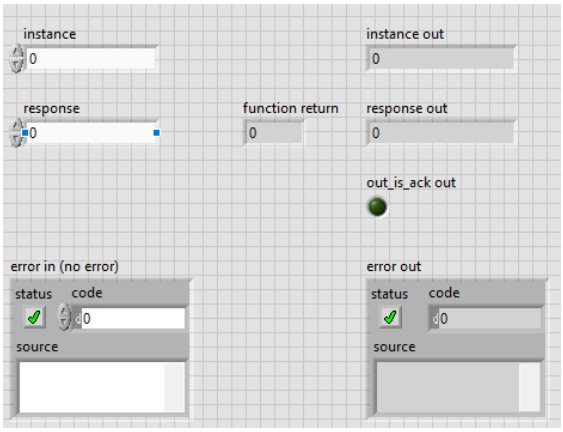


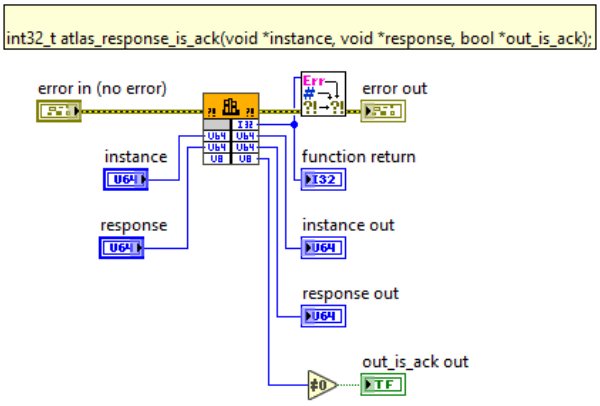


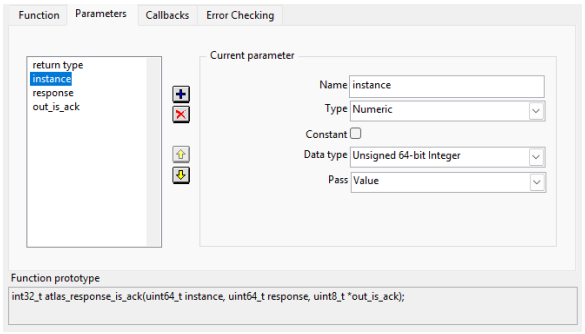


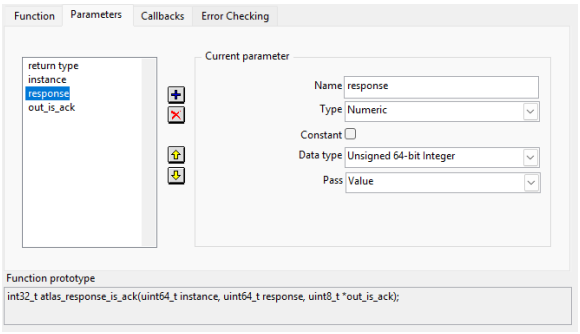


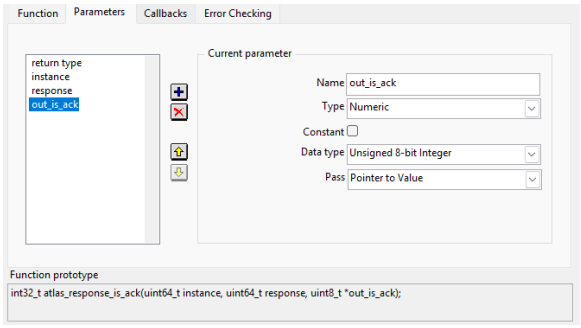
atlas response is ack.vi











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