Your Global Automation Partner

ARGEE 3
Libraries

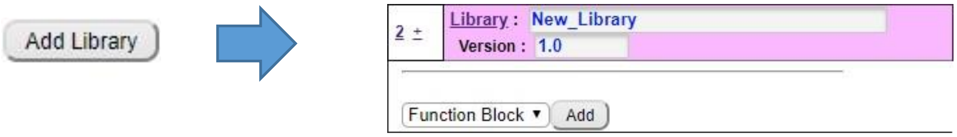
0819A

1 Library Basics

A library in ARGEE 3 is simply any collection of function blocks and states which are grouped together to improve the readability and structure of the program. Libraries can easily be created in ARGEE PRO and then exported and imported to use in other projects. Libraries are a great way to add functionality to ARGEE without making changes to the underlying programming environment. Turck already offers some libraries for download to help with simple common tasks (MISC library) or simplify more complex task like RFID and IO-Link specific data handling (IO-Library).

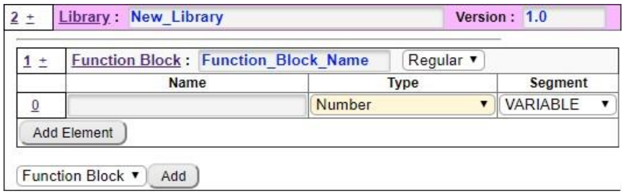
1.1 Creating a Library

First click on the *Add Library* button.



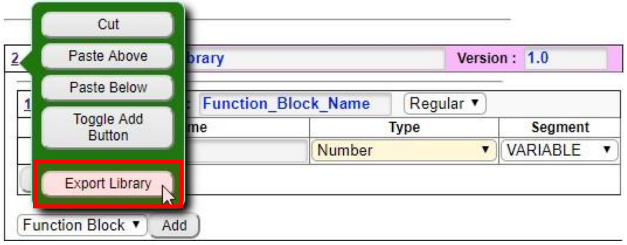
Library : New_Library
Version : 1.0

Then create the desired function blocks.



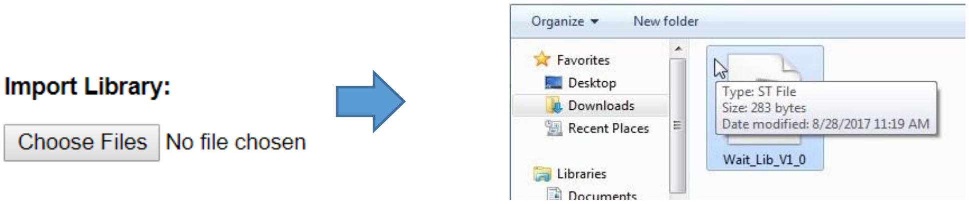
Library : New_Library
Version : 1.0
Function Block : Function_Block_Name
Regular ▾
Number ▾
VARIABLE ▾

Once the library is complete, select *Export Library* from the library context menu.



1.2 Importing a Library

Import an already pre-built library by clicking on the *Choose Files* button.



NOTE

If the user try's to import a library with the same name as an already installed library, ARGEE will ask the user remove the first library before importing the second.

2 Downloadable Libraries

In addition to creating libraries, the user can download official ARGEE libraries from the Turck Website

2.1 MISC Library

The MISC library contains commonly used functions that can greatly simply more advanced ARGEE projects

2.1.1 MISC library Available Function Blocks

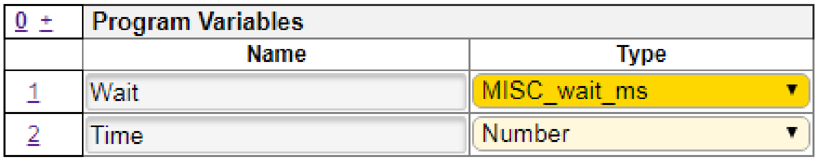
- Wait_ms – Delays program execution for x-number of ms
- Number_ST – used to pass arguments to functions by reference (similar to Pointer)
- Copy_byte_arr – Copies the data from one byte array to another byte array
- copy_byte_to_word_arr – Copies the data from a byte array to a word array
- copy_word_to_byte_arr – Copies the data from a word array to a byte array
- compare_byte_arrays – Compares two arrays and sets a response to 0 if they aren't the same
- convArrToString – Converts the data in a byte array into a string
- compare_and_copy – Copies a array into another array and sets a response to 1 if they arrays are different
- STR_AddSpecialChar – Adds a character to the end of a string
- copy_byte_to_long_arr – Copies a byte array to a number
- copy_long_to_byte_arr – Copies a number to a byte array

2.1.2 wait_ms

Function: When wait_ms is called it halts the task execution for the designated amount of time.

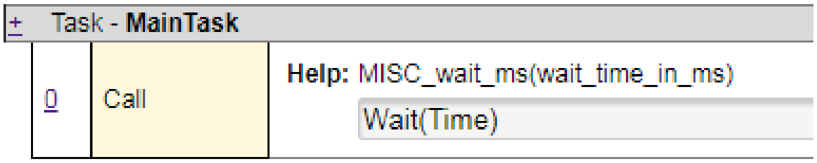
Scope	Name	Type	Comment
Input	wait_time_in_ms	Number	Delay time in ms

Program Variables: A wait_ms program variable is needed to call the function.



0 ±	Program Variables	
	Name	Type
1	Wait	MISC_wait_ms
2	Time	Number

How to Call: The Call needs a wait time in ms argument. This can be a static number or a number program variable.



Help: MISC_wait_ms(wait_time_in_ms)

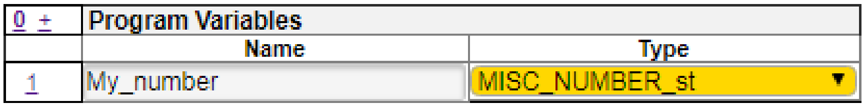
Wait(Time)

2.1.3 NUMBER_st

Function: The function of NUMBER_st is to pass a number to a function block by reference (similar to using pointers)

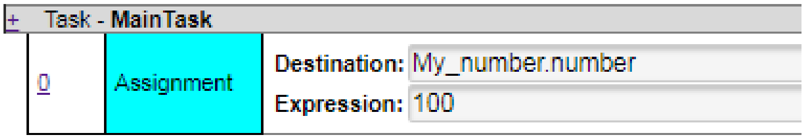
*This function block has no arguments.

Program Variables: The only variable needed is a NUMBER_st variable.



0 +	Program Variables	
	Name	Type
1	My_number	MISC_NUMBER_st ▼

How to Call: This function block is not really called instead a number is assigned to the variable in the function block as shown below.



+ Task - **MainTask**

0 Assignment

Destination: My_number.number

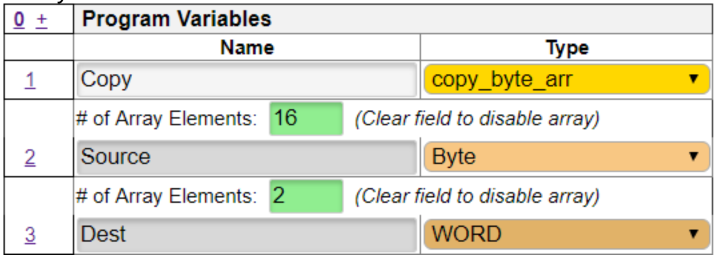
Expression: 100

2.1.4 Copy_byte_arr

Function: copy_byte_arr copies the data from a source byte array to a destination byte array, and the data from the source and to the destination can both be offset.

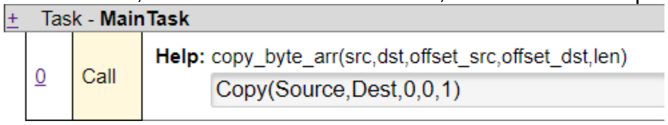
Scope	Name	Type	Comment
Input	src	Byte Array	Source that will be copied
Input	offset_src	Number	Offset of the source array
Input	offset_dst	Number	Offset of the destination array
Input	len	Number	Length of the data being moved in bytes
Output	dst	Byte Array	Destination of the data being copied

Program Variables: The program variables needed are the copy_byte_to_word_arr to call, the length of the arrays (in this case 32), and source and destination arrays. Offset values are also used but they do not need to be variables.



0 +	Program Variables	
	Name	Type
1	Copy	copy_byte_arr ▼
	# of Array Elements: 16	*(Clear field to disable array)*
2	Source	Byte ▼
	# of Array Elements: 2	*(Clear field to disable array)*
3	Dest	WORD ▼

How to Call: To call copy_byte_arr the user needs the source array, the destination array, a source offset number, a destination offset number, and number to represent the length of the arrays.



+ Task - **MainTask**

0 Call

Help: copy_byte_arr(src,dst,offset_src,offset_dst,len)

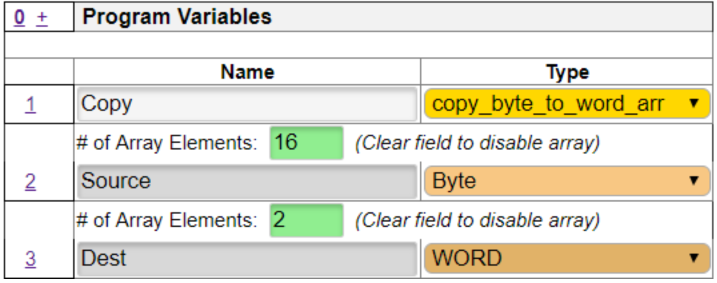
```
Copy(Source,Dest,0,0,1)
```

2.1.5 copy_byte_to_word_arr

Function: copy_byte_to_word_arr copies the data from a source byte array to a destination word array, and the data from the source and to the destination can both be offset.

Scope	Name	Type	Comment
Input	src	Byte Array	Source that will be copied
Input	offset_src	Number	Offset of the source array
Input	offset_dst	Number	Offset of the destination array
Input	len_words	Number	Length of the data being moved in words
Output	dst	Word Array	Destination of the data being copied

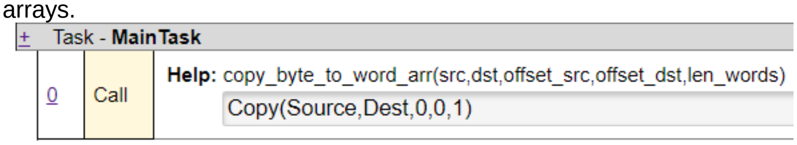
Program Variables: The program variables needed are the copy_byte_to_word_arr to call, the length of the arrays (in this case 32), and source and destination arrays. Offset values are also used but they do not need to be variables.



0 + **Program Variables**

	Name	Type
1	Copy	copy_byte_to_word_arr ▼
	# of Array Elements: 16	*(Clear field to disable array)*
2	Source	Byte ▼
	# of Array Elements: 2	*(Clear field to disable array)*
3	Dest	WORD ▼

How to Call: To call copy_byte_to_word_arr the user needs the source array, the destination array, a source offset number, a destination offset number, and number to represent the length of the arrays.



+ Task - **MainTask**

0 Call **Help:** copy_byte_to_word_arr(src,dst,offset_src,offset_dst,len_words)

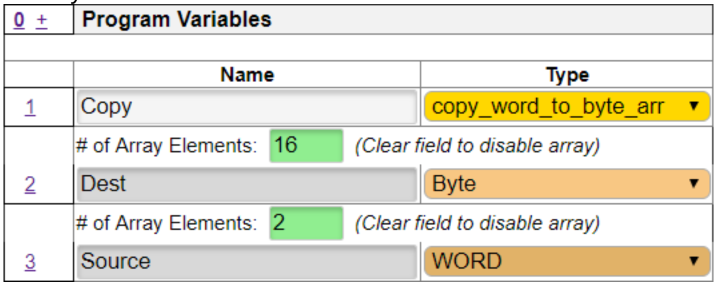
```
Copy(Source,Dest,0,0,1)
```

2.1.6 copy_word_to_byte_arr

Function: copy_word_to_byte_arr copies the data from a source word array to a destination byte array, and the data from the source and to the destination can both be offset.

Scope	Name	Type	Comment
Input	src	Byte Array	Source that will be copied
Input	offset_src	Number	Offset of the source array
Input	offset_dst	Number	Offset of the destination array
Input	len_words	Number	Length of the data being moved in words
Output	dst	Word Array	Destination of the data being copied

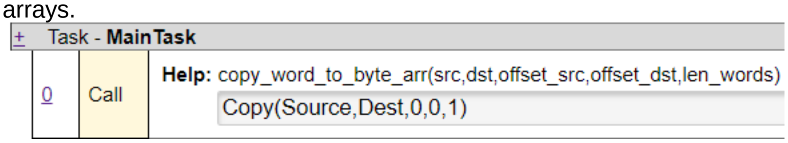
Program Variables: The program variables needed are the copy_word_to_byte_arr to call, the length of the arrays (in this case 32), and source and destination arrays. Offset values are also used but they do not need to be variables.



0 + **Program Variables**

	Name	Type
1	Copy	copy_word_to_byte_arr ▼
	# of Array Elements: 16	*(Clear field to disable array)*
2	Dest	Byte ▼
	# of Array Elements: 2	*(Clear field to disable array)*
3	Source	WORD ▼

How to Call: To call copy_word_to_byte_arr the user needs the source array, the destination array, a source offset number, a destination offset number, and number to represent the length of the arrays.



+ Task - **MainTask**

0 Call **Help:** copy_word_to_byte_arr(src,dst,offset_src,offset_dst,len_words)

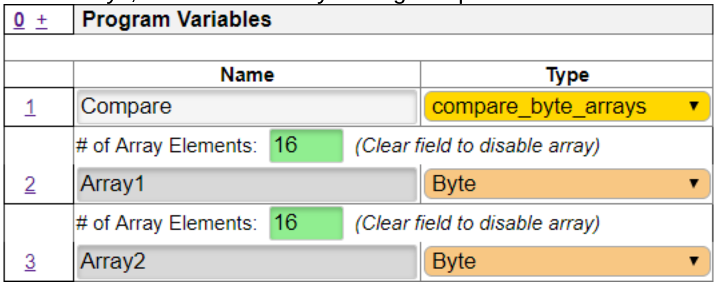
```
Copy(Source,Dest,0,0,1)
```

2.1.7 compare_byte_arrays

Function: compare_byte_arrays looks at two byte arrays and sets a response (res) variable to 0 if the arrays aren't the same.

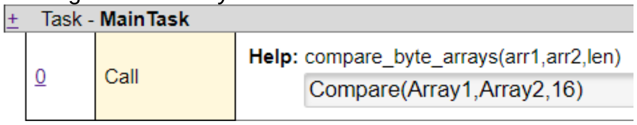
Scope	Name	Type	Comment
Input	arr1	Byte Array	One of the arrays to be compared
Input	arr2	Byte Array	Other array to be compared
Input	len	Number	Number of bytes being compared

Program Variables: The program variables needed are the compare_byte_arrays to call, the length of the arrays, and the two arrays being compared.



0 +	Program Variables	
	Name	**Type**
1	Compare	compare_byte_arrays ▼
	# of Array Elements: 16 (Clear field to disable array)	
2	Array1	Byte ▼
	# of Array Elements: 16 (Clear field to disable array)	
3	Array2	Byte ▼

How to Call: To call compare_byte_arrays the user needs the two arrays, and number to represent the length of the arrays.



+ Task - **Main Task**

0 Call **Help:** compare_byte_arrays(arr1,arr2,len)

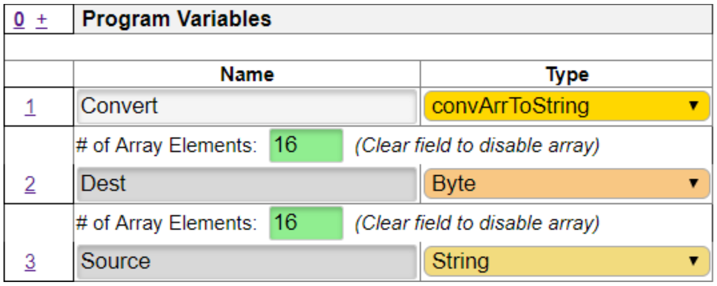
Compare(Array1,Array2,16)

2.1.8 convArrToString

Function: convArrToString copies the data from a source byte array to a destination string, and the data from the source or the destination can both be offset.

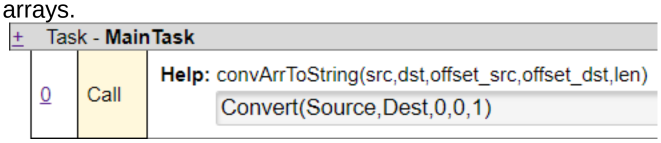
Scope	Name	Type	Comment
Input	src	Byte Array	Source that will be converted
Input	offset_src	Number	Offset of the source array
Input	offset_dst	Number	Offset of the destination array
Input	len	Number	Length of the data being moved in words
Output	dst	String	String where the data is being output

Program Variables: The program variables needed are the convArrToString to call, the length of the arrays, and source and destination arrays. Offset values are also used but they do not need to be variables.



0 +	Program Variables	
	Name	**Type**
1	Convert	convArrToString ▼
	# of Array Elements: 16 (Clear field to disable array)	
2	Dest	Byte ▼
	# of Array Elements: 16 (Clear field to disable array)	
3	Source	String ▼

How to Call: To call copy_word_to_byte_arr the user needs the source array, the destination array, a source offset number, a destination offset number, and number to represent the length of the arrays.



+ Task - **Main Task**

0 Call **Help:** convArrToString(src,dst,offset_src,offset_dst,len)

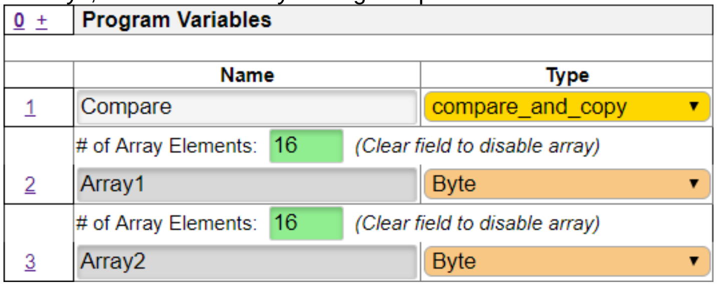
Convert(Source,Dest,0,0,1)

2.1.9 compare_and_copy

Function: compare_and_copy looks at two byte arrays (the current array that is being copied and the previous array that is being copied to) and sets a changed variable to 1 if the arrays aren't the same and copies one of the arrays to the other.

Scope	Name	Type	Comment
Input	curr_arr	Byte Array	Source that will be copied and compared
Output	prev_arr	Byte Array	Output array that is also compared
Input	num_elems	Number	Number of bytes being compared/ copied

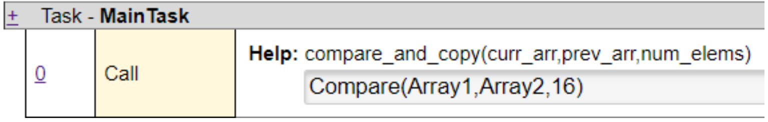
Program Variables: The program variables needed are the compare_and_copy to call, the length of the arrays, and the two arrays being compared.



Program Variables

	Name	Type
1	Compare	compare_and_copy
	# of Array Elements: 16	(Clear field to disable array)
2	Array1	Byte
	# of Array Elements: 16	(Clear field to disable array)
3	Array2	Byte

How to Call: To call compare_and_copy the user needs the two arrays, and number to represent the length of the arrays being compared/ copied.



Task - **Main Task**

0 Call

Help: compare_and_copy(curr_arr,prev_arr,num_elems)

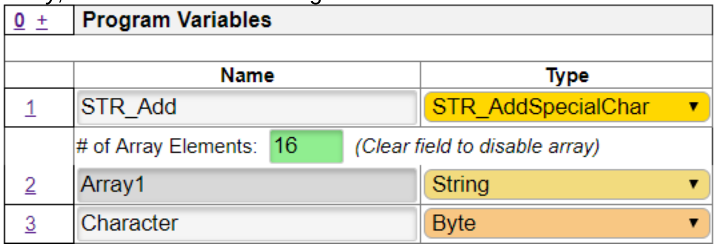
Compare(Array1,Array2,16)

2.1.10 STR_AddSpecialChar

Function: STR_AddSpecialChar looks at a string array and adds a character to the end of it.

Scope	Name	Type	Comment
Input	char	Byte	Character that will be added to the string
Output	str1	String	String that char will be added to

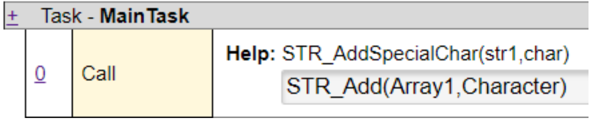
Program Variables: The program variables needed are the STR_AddSpecialChar to call, the string array, and the character being added.



Program Variables

	Name	Type
1	STR_Add	STR_AddSpecialChar
	# of Array Elements: 16	(Clear field to disable array)
2	Array1	String
3	Character	Byte

How to Call: To call STR_AddSpecialChar the user needs a string, and the character (as a byte) being added to it.



Task - **Main Task**

0 Call

Help: STR_AddSpecialChar(str1,char)

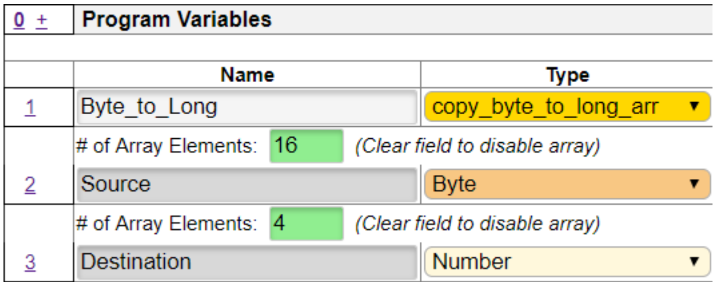
STR_Add(Array1,Character)

2.1.11 copy_byte_to_long_arr

Function: copy_byte_to_long_arr copies the data from a source byte array to a destination number array, and the data from the source and to the destination can both be offset.

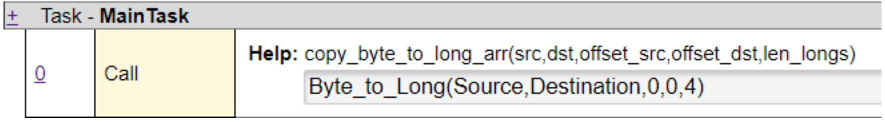
Scope	Name	Type	Comment
Input	src	Byte Array	Source that will be copied
Input	offset_src	Number	Offset of the source array
Input	offset_dst	Number	Offset of the destination array
Input	len_longs	Number	Length of the data being moved
Output	dst	Number	Destination of the data being copied

Program Variables: The program variables needed are the copy_byte_to_long_arr to call, the length of the arrays, and source and destination arrays. Offset values are also used but they do not need to be variables.



	Name	Type
1	Byte_to_Long	copy_byte_to_long_arr
	# of Array Elements: 16 (Clear field to disable array)	
2	Source	Byte
	# of Array Elements: 4 (Clear field to disable array)	
3	Destination	Number

How to Call: To call copy_byte_to_long_arr the user needs the source array, the destination array, a source offset number, a destination offset number, and number to represent the length of the bytes.



Help: copy_byte_to_long_arr(src,dst,offset_src,offset_dst,len_longs)

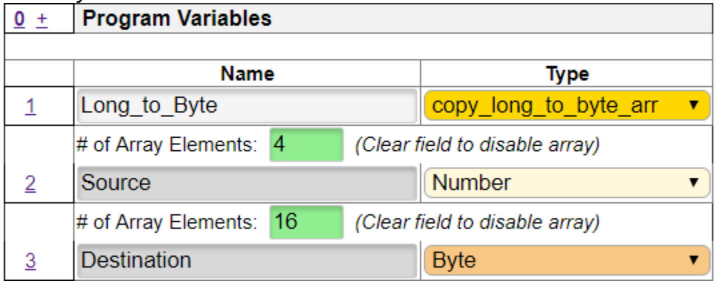
```
Byte_to_Long(Source,Destination,0,0,4)
```

2.1.12 copy_long_to_byte_arr

Function: copy_long_to_byte_arr copies the data from a source number array to a destination byte array, and the data from the source and to the destination can both be offset.

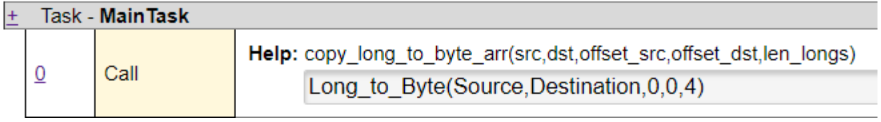
Scope	Name	Type	Comment
Input	src	Number	Source that will be copied
Input	offset_src	Number	Offset of the source array
Input	offset_dst	Number	Offset of the destination array
Input	len_longs	Number	Length of the data being moved
Output	dst	Byte Array	Destination of the data being copied

Program Variables: The program variables needed are the copy_word_to_byte_arr to call, the length of the arrays (in this case 32), and source and destination arrays. Offset values are also used but they do not need to be variables.



	Name	Type
1	Long_to_Byte	copy_long_to_byte_arr
	# of Array Elements: 4 (Clear field to disable array)	
2	Source	Number
	# of Array Elements: 16 (Clear field to disable array)	
3	Destination	Byte

How to Call: To call copy_word_to_byte_arr the user needs the source array, the destination array, a source offset number, a destination offset number, and number to represent the length of the arrays.



Help: copy_long_to_byte_arr(src,dst,offset_src,offset_dst,len_longs)

Long_to_Byte(Source,Destination,0,0,4)

2.2 IO-Library

The IO-Library contains IO-Link and RFID functions that can greatly simply more advanced ARGEE projects

2.2.1 IO Library Available Function Blocks

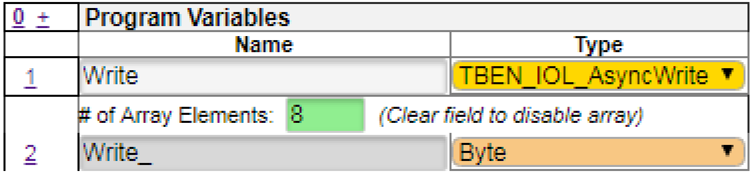
- TBEN_IOL_AsyncWrite – Delays program execution for x-number of ms
- TBEN_IOL_AsyncRead – used to pass arguments to functions by reference (similar to Pointer)
- TBEN_S2_RFID_READ – Copies the data from one byte array to another byte array
- TBEN_S2_RFID_WRITE – Copies the data from a byte array to a word array
- BLCEN_RFIDS_Read – Copies the data from a word array to a byte array
- BLCEN_RFIDS_Write – Compares two arrays and sets a response to 0 if they aren’t the same

2.2.2 TBEN_IOL_AsyncWrite

Function: When TBEN_IOL_AsyncWrite is called the data from a byte array is written into a chosen index and sub index.

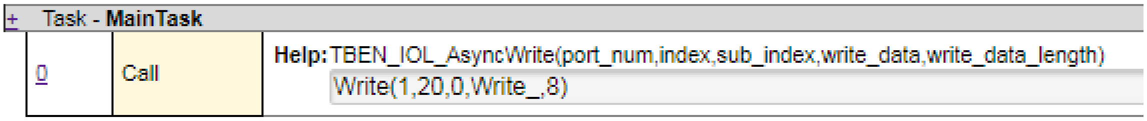
Scope	Name	Type	Comment
Input	port_num	Number	Port that the data is being written to
Input	index	Number	IO-Link index being written to
Input	sub_index	Number	IO-Link sub index being written to
Input	write_data	Byte Array	IO-Link data being written
Input	write_data_length	Number	Length of the IO-Link index being written to

Program Variables: The only program variables needed are a TBEN_IOL_AsyncWrite variable, and a byte array variable.



	Name	Type
1	Write	TBEN_IOL_AsyncWrite ▼
	# of Array Elements: 8 *(Clear field to disable array)*	
2	Write_	Byte ▼

How to Call: To call TBEN_IOL_AsyncWrite the following arguments need to be satisfied, the port that is being used, the parameter index that the user is trying to write into, the sub index that the user is trying to write into, the byte array that is being written, and the length of the array being written.



Help: TBEN_IOL_AsyncWrite(port_num,index,sub_index,write_data,write_data_length)

Write(1,20,0,Write_,8)

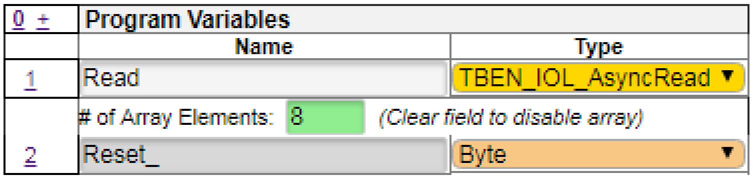
2.2.3 TBEN_IOL_AsyncRead

Function: When TBEN_IOLAsyncRead is called the parameter data from a chosen index and sub index is read into the ds_tx_array and ds_rx_array.

Scope	Name	Type	Comment

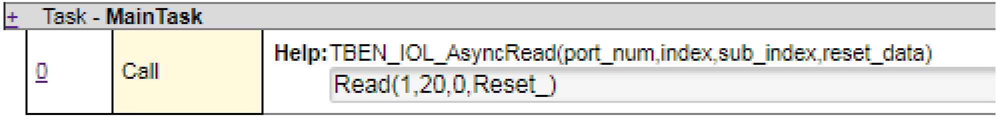
Input	port_num	Number	Port that the data is being read from
Input	index	Number	IO-Link index being read from
Input	sub_index	Number	IO-Link sub index being read from
Output	reset_data	Byte Array	Read data is stored here

Program Variables: The variables needed to call TBEN_IOL_AsyncRead are a TBEN_IOL_AsyncRead variable, and a byte array variable.



0 +	Program Variables	
	Name	Type
1	Read	TBEN_IOL_AsyncRead ▼
	# of Array Elements: 8	(Clear field to disable array)
2	Reset_	Byte ▼

How to Call: To call TBEN_IOL_AsyncRead the following arguments need to be filled; the port that is being used, the parameter index that the user is trying to read, the sub index that the user is trying to read, and a reset byte array.



Help: TBEN_IOL_AsyncRead(port_num,index,sub_index,reset_data)

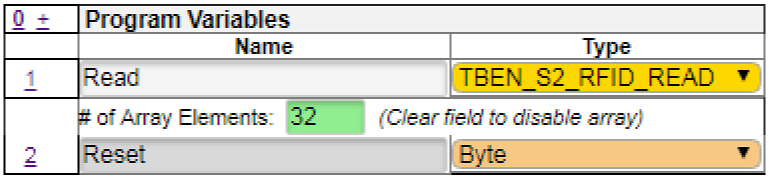
```
Read(1,20,0,Reset_)
```

2.2.4 TBEN_S2_RFID_READ

Function: TBEN_S2_RFIDS_READ when called waits for the next tag to be presented and reads it, and that data is held in the input read data.

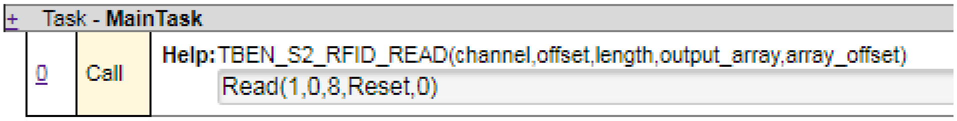
Scope	**Name**	**Type**	**Comment**
Input	channel	Number	Port that the data is being read from
Input	offset	Number	Offset in bytes of the data being read
Input	length	Number	Number of bytes being read
Input	array_offset	Number	Offset of the data being stored
Output	output_array	Byte Array	Where the read data is stored

Program Variables: To call TBEN_S2_RFIDS_READ a TBEN_S2_RFIDS_READ variable, and a byte array are needed.



0 +	Program Variables	
	Name	Type
1	Read	TBEN_S2_RFID_READ ▼
	# of Array Elements: 32	(Clear field to disable array)
2	Reset	Byte ▼

How to Call: When calling TBEN_S2_RFIDS_READ the following arguments need to be fulfilled; which channel is being used, how much the data being read should be offset, the number of bytes that are being read from the tag, the reset data byte array, and how much the array data should be offset.



Help: TBEN_S2_RFID_READ(channel,offset,length,output_array,array_offset)

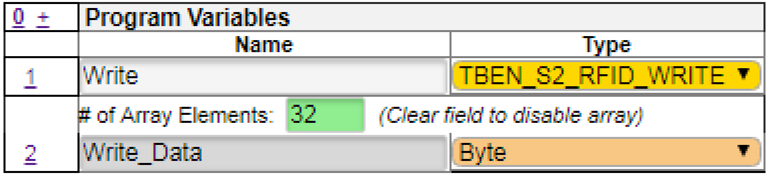
```
Read(1,0,8,Reset,0)
```

2.2.5 TBEN_S2_RFID_WRITE

Function: The function of the TBEN_S2_RFID_WRITE when called writes the data from a byte array is written onto the next tag that is presented into the transceiver's field.

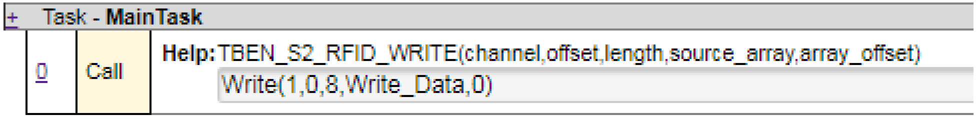
Scope	Name	Type	Comment
Input	channel	Number	Port that the data is being written to
Input	offset	Number	Offset in bytes of the data being written
Input	length	Number	Number of bytes being written
Input	array_offset	Number	Offset of the data being written
Output	res_data	Byte Array	Data being written

Program Variables: To call TBEN_S2_RFID_WRITE a TBEN_S2_RFID_WRITE variable is needed, and a Byte array that holds the data that is being written is needed.



0 +	Program Variables	
	Name	Type
1	Write	TBEN_S2_RFID_WRITE ▼
	# of Array Elements: 32 (Clear field to disable array)	
2	Write_Data	Byte ▼

How to Call: The arguments needed to call TBEN_S2_RFID_WRITE are, which channel is being used, how much the data being written should be offset onto the tag, the length of the array being written onto the tag, the data array that is being written onto the tag, and how much the array data being written should be offset.



+ Task - **MainTask**

0 Call

Help: TBEN_S2_RFID_WRITE(channel,offset,length,source_array,array_offset)

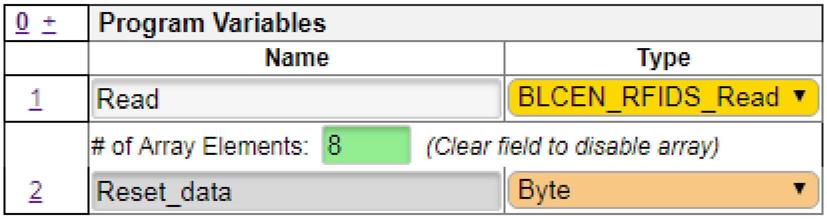
```
Write(1,0,8,Write_Data,0)
```

2.2.6 BLCEN_RFIDS_Read

Function: BLCEN_RFIDS_Read when called waits for the next tag to be presented to read, and that data is held in the input read data.

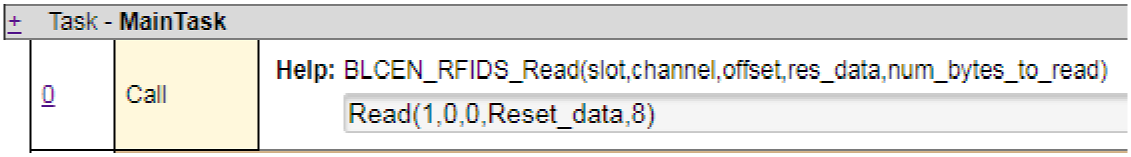
Scope	Name	Type	Comment
Input	slot	Number	Slot on the BLCEN being read from
Input	channel	Number	Port that the data is being read from
Input	offset	Number	Offset in bytes of the data being read
Input	num_bytes_to_read	Number	Number of bytes being read
Output	output_array	Byte Array	Where the read data is stored

Program Variables: To call BLCEN_RFIDS_Read a BLCEN_RFIDS_Read variable, and a byte array are needed.



0 +	Program Variables	
	Name	Type
1	Read	BLCEN_RFIDS_Read ▼
	# of Array Elements: 8 (Clear field to disable array)	
2	Reset_data	Byte ▼

How to Call: When calling BLCEN_RFIDS_Read the following arguments need to be fulfilled; what slot of the BLCEN has the 2RFID channels, which channel is being used, how much the data being read should be offset, the reset data byte array, and the number of bytes that are being read from the tag.



+ Task - **MainTask**

0 Call **Help:** BLCEN_RFIDS_Read(slot,channel,offset,res_data,num_bytes_to_read)

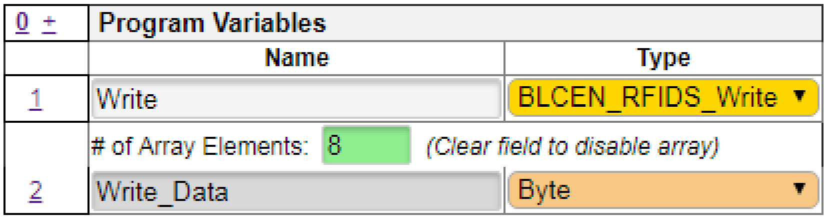
Read(1,0,0,Reset_data,8)

2.2.7 BLCEN_RFIDS_Write

Function: When BLCEN_RFIDS_Write is called the data from an outp_data is written onto the next tag that is put into the transvers field.

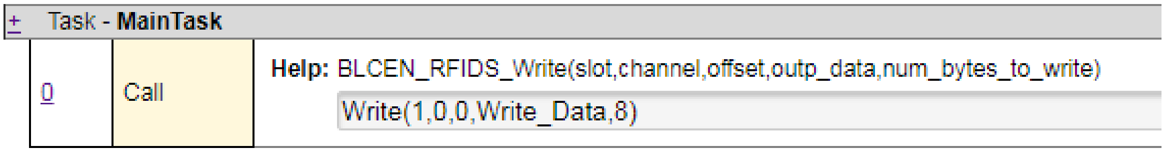
Scope	Name	Type	Comment
Input	slot	Number	The BLCEN slot being written to
Input	channel	Number	Port that the data is being written to
Input	offset	Number	Offset in bytes of the data being written
Input	num_bytes_to_write	Number	Number of bytes being written
Output	outp_data	Byte Array	Data being written

Program Variables: To call BLCEN_RFIDS_Write a BLCEN_RFIDS_Write variable is needed, and a Byte array that holds the data that is being written is needed.



0 +	Program Variables	
	Name	**Type**
1	Write	BLCEN_RFIDS_Write ▼
	# of Array Elements: 8 *(Clear field to disable array)*	
2	Write_Data	Byte ▼

How to Call: The arguments needed to call BLCEN_RFIDS_Write are, what slot of the BLCEN has the 2RFID channels, which channel is being used, how much the data being written should be offset onto the tag, the data array that is being written onto the tag, and the number of bytes that are being written onto the tag.



+ Task - **MainTask**

0 Call **Help:** BLCEN_RFIDS_Write(slot,channel,offset,outp_data,num_bytes_to_write)

Write(1,0,0,Write_Data,8)

TURCK sells its products through Authorized Distributors. These distributors provide our customers with technical support, service and local stock. TURCK distributors are located nationwide –
Including all major metropolitan marketing areas
For Application Assistance or for the location of your nearest TURCK distributor, call:
1-800-544-7769

Specifications in this manual are subject to change without notice. TURCK also reserves the right to make modifications and makes no guarantee of the accuracy of the information contained herein.