Quicksort

IEC 61131 Library for ACSELERATOR RTAC® Projects

SEL Automation Controllers

Table of Contents

RTAC LIBRARY

Quicksort

Introduction

This library implements the Quicksort algorithm for sorting arrays of objects. Due to the strongly typed nature of IEC 61131, the sorting is provided with multiple functions, one for each of the following datatypes:

- USINT
- SINT
- UINT
- INT
- UDINT
- DINT
- ULINT
- LINT
- REAL
- LREAL

This library also allows the sorting of arbitrary objects implementing the *I_Comparable* interface. See the ACSELERATOR RTAC Library Extensions Instruction Manual (LibraryExtensionsIM) for explanation of the concepts used by the object-oriented extensions to the IEC 61131-3 standard. All functions sort in ascending order.

Supported Firmware Versions

You can use this library on any device configured using ACSELERATOR RTAC® SEL-5033 Software with firmware version R143 or higher.

Version 3.5.0.0 can be used on RTAC firmware version R132 and higher.

Functions

The following functions are provided by this library.

fun_SortUSINT (Function)

This function takes a pointer to an array of USINTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO USINT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortSINT (Function)

This function takes a pointer to an array of SINTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO SINT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortUINT (Function)

This function takes a pointer to an array of UINTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO UINT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortINT (Function)

This function takes a pointer to an array of INTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO INT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortUDINT (Function)

This function takes a pointer to an array of UDINTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO UDINT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortDINT (Function)

This function takes a pointer to an array of DINTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO DINT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortULINT (Function)

This function takes a pointer to an array of ULINTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO ULINT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortLINT (Function)

This function takes a pointer to an array of LINTs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO LINT	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.

➤ If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortREAL (Function)

This function takes a pointer to an array of REALs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO REAL	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

➤ The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.

➤ If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.

➤ If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortLREAL (Function)

This function takes a pointer to an array of LREALs and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO LREAL	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function uses the Quicksort algorithm to perform an in-place sorting of the array.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, sorting of the array occurs through use of the Quicksort algorithm, and the function returns the value TRUE.

fun_SortI_Comparable (Function)

This function takes a pointer to an array of *I_Comparable* objects and sorts the array in-place by value.

Inputs

Name	IEC 61131 Type	Description
arrayPointer	POINTER TO I_Comparable	Pointer to the array to sort.
arraySize	UDINT	The number of elements in the array.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the sort completed successfully. FALSE if an error occured.

Processing

This function sorts the array in-place using the Quicksort algorithm.

- The function returns false and does not attempt to sort the array if the *arrayPointer* is invalid.
- The function returns false and does not attempt to sort the array if any members of the array pointed to by *arrayPointer* are invalid.
- If *arrayPointer* is valid and *arraySize* is less than two, the function returns true without sorting the array. Specifically, arrays of size zero or one are considered already sorted.
- If *arrayPointer* is valid and *arraySize* is greater than or equal to two, the array is sorted using an in-place Quicksort algorithm and true is returned.

Interfaces

This library provides the following interfaces.

I_Comparable (Interface)

This interface is implemented by any class needing to be sortable. Any libraries needing to sort arbitrary objects can create a class that implements this interface.

Properties

Name	IEC 61131 Type	Access	Description
pt_Self	POINTER TO BYTE	R	Provides the THIS pointer for the class.

Properties are internal values made visible through Get and Set accessors. Access is defined as R (read), W (write), or R/W (read/write).

CompareTo (Method)

This method compares this object to another object of the same type. If the *compare* object is not of the same class, exceptions will occur.

Inputs

Name	IEC 61131 Type	Description
compare	I_Comparable	The object to compare.

Return Value

IEC 61131 Type	Description
INT	Returns a signed integer representing the sorting order. A negative value means this object goes before the *compare* object. Zero means this object is equivalent to the *compare* object. A positive value means this object goes after the *compare* object.

Processing

Compares this object to another object of the same type.

- Returns a negative value if this object goes before the *compare* object.
- Returns zero if this object is equivalent to the *compare* object.
- Returns a positive value if this object goes after the *compare* object.

Benchmarks

Benchmark Platforms

The benchmarking tests recorded for this library are performed on the following platforms.

- SEL-3530
  - R134 firmware
- SEL-3354
  - Intel Pentium 1.4 GHz
  - 1 GB DDR ECC SDRAM
  - SEL-3532 RTAC Conversion Kit
  - R132 firmware
- SEL-3555
  - Dual-core Intel i7-3555LE processor
  - 4 GB ECC RAM
  - R134-V0 firmware

Benchmark Test Descriptions

All benchmark tests are based on an average of 100 calls in the described environment containing 1000 elements for each DATATYPE having a Quicksort method where DATATYPE is selected from the following types:

- SINT
- USINT
- INT
- UINT
- DINT
- UDINT
- LINT
- ULINT
- REAL
- LREAL

Sorting performance any I_Comparable object will depend on the implementation, so this document does not attempt to characterize that behavior.

DATATYPE InOrderSort

The average time for completion of the sort method when all data are presorted.

DATATYPE ReverseOrderSort

The average time for completion of the sort method when all data are in reverse order.

DATATYPE RandomOrderSort

The average time for completion of the sort method when data are in random order.

Benchmark Results

Operation Tested	Platform (time in μs)		
	SEL-3530	SEL-3354	SEL-3555
SINT InOrderSort	1483	258	86
SINT ReverseOrderSort	1519	269	95
SINT RandomOrderSort	1756	343	141
USINT InOrderSort	1472	257	87
USINT ReverseOrderSort	1499	268	96
USINT RandomOrderSort	1728	345	142
INT InOrderSort	1317	233	78
INT ReverseOrderSort	1471	275	97
INT RandomOrderSort	1789	376	160
UINT InOrderSort	1327	233	76
UINT ReverseOrderSort	1476	271	94
UINT RandomOrderSort	1799	367	156
DINT InOrderSort	1317	229	76
DINT ReverseOrderSort	1469	272	94
DINT RandomOrderSort	1819	368	155
UDINT InOrderSort	1313	232	76
UDINT ReverseOrderSort	1461	271	92
UDINT RandomOrderSort	1800	368	157
LINT InOrderSort	1549	282	93
LINT ReverseOrderSort	1720	323	112
LINT RandomOrderSort	2159	444	181
ULINT InOrderSort	1552	275	91
ULINT ReverseOrderSort	1730	316	111
ULINT RandomOrderSort	2133	458	200
REAL InOrderSort	1460	242	78
REAL ReverseOrderSort	1634	283	97
REAL RandomOrderSort	2042	392	170
LREAL InOrderSort	1674	280	78
LREAL ReverseOrderSort	1863	314	97
LREAL RandomOrderSort	2323	444	168

Examples

These examples demonstrate the capabilities of this library. Do not mistake them as suggestions or recommendations from SEL.

Implement the best practices of your organization when using these libraries. As the user of this library, you are responsible for ensuring correct implementation and verifying that the project using these libraries performs as expected.

Sorting an Array of Integers

Objective

Code Snippet 1 demonstrates sorting a simple array of INTs.

Solution

Code Snippet 1 prg_SortInt

```
PROGRAM prg_SortInt
VAR
    intArray : ARRAY [1..5] OF INT;
END_VAR
```

```
// Populate the array with unsorted values.
intArray[1] := 5;
intArray[2] := 2;
intArray[3] := 0;
intArray[4] := -1;
intArray[5] := -4;
// Sort the array.
fun_SortINT(ADR(intArray), 5);
```

Release Notes

Version	Summary of Revisions	Date Code
3.5.1.0	➤ Allows new versions of ACSELERATOR RTAC to compile projects for previous firmware versions without SEL IEC types "Cannot convert" messages. ➤ Replaced the deprecated POINTER_TO_ANY type with POINTER_TO_BYTE. ➤ Must be used with R143 firmware or later.	20180921
3.5.0.0	➤ Initial release.	20140812