SELServerSimulators

IEC 61131 Library for ACSELERATOR RTAC® Projects

SEL Automation Controllers

Table of Contents

RTAC LIBRARY

SELServerSimulators

Introduction

This library provides simulators for various SEL devices using the SEL Fast Meter protocol. Since this library is primarily a testing and debugging tool, not all commands may be supported for a particular device. There may also be variances in the responses generated when compared to the real device.

See the ACSELERATOR RTAC Library Extensions Instruction Manual (LibraryExtensionsIM) for an explanation of the concepts used by the object-oriented extensions to the IEC 61131-3 standard.

Special Considerations

Classes in this library have memory allocated inside them. As such, they should only be created in environments of permanent scope (e.g., Programs, Global Variable Lists, or VAR_STAT sections).

Copying classes from this library causes unwanted behavior. This means the following:

1. The assignment operator ":=" must not be used on any class from this library; consider assigning pointers to the objects instead.

```
// This is bad and in most cases will provide a compiler error such
    as:
// "C0328: Assignment not allowed for type class_Object"
myObject := otherObject;
```

```
// This is fine
someVariable := myObject.value;
// As is this
pt_myObject := ADR(myObject);
```

2. Classes from this library must never be VAR_INPUT or VAR_OUTPUT members in function blocks, functions, or methods. Place them in the VAR_IN_OUT section or use pointers instead.

Supported Devices

This library contains simulators for the following devices:

- SEL-351

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 3.5.0.1 can be used on RTAC firmware version R132 and higher.

Enumerations

Enumerations make code more readable by allowing a specific number to have a readable textual equivalent.

enum_PointType

This enumeration defines all possible point types for a relay.

Enumeration	Description
DIGITAL	A digital (Boolean) point.
INT16	An analog point stored as a 16-bit integer value.
FLOAT32	An analog point stored as a 32-bit floating-point value.
FLOAT64	An analog point stored as a 64-bit floating-point value.
STRING80	A point stored as a string of as many as 80 characters.

Structures

Structures provide a means to group together several memory locations (variables), making them easier to manage.

Because of the significant number of values in these structures, this document does not provide a complete list of the values.

struct_SEL351_Analog

This structure defines the analog points available on an SEL-351. For example, this structure might contain the following:

Name	IEC 61131 Type	Description
_FREQ	class_SELServerFloat32	System frequency
_IA	class_SELServerFloat32	Phase A current magnitude
_IB	class_SELServerFloat32	Phase B current magnitude
_IC	class_SELServerFloat32	Phase C current magnitude

struct_SEL351_Binary

This structure defines the digital points available on an SEL-351.

struct_SEL351_Strings

This structure defines the string points available on an SEL-351.

Interfaces

I_Session

This interface defines a set of methods for interacting with a client session. A session is a generic object that allows for reading and writing of bytes.

Properties

Name	IEC 61131 Type	Access	Description
Active	BOOL	R	TRUE if the session is active and able to send and receive bytes.
Echo	BOOL	R	TRUE if the session is expecting ASCII characters sent to the server to be echoed back.

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

Close (Method)

Closes the session to the client.

Inputs

Name	IEC 61131 Type	Description
forceClose	BOOL	Close the session without waiting for confirmation.

ReadData (Method)

Read data from the client.

Inputs

Name	IEC 61131 Type	Description
pt_destination	POINTER TO BYTE	Pointer to where the incoming data are written.
numBytes	UDINT	The maximum number of bytes to read.

Return Value

IEC 61131 Type	Description
UDINT	The number of bytes read from the session.

WriteData (Method)

Write data to the client.

Inputs

Name	IEC 61131 Type	Description
pt_source	POINTER TO BYTE	Pointer from where the outgoing data are read.
numBytes	UDINT	The number of bytes to read from *pt_source* and write to the session.

Return Value

IEC 61131 Type	Description
UDINT	The number of bytes written to the session.

I_SessionManager

This interface defines a set of methods for an object that is able to manage multiple I_-SessionProvider objects and their sessions.

Properties

Name	IEC 61131 Type	Access	Description
numSessions	UDINT	R	The total number of sessions managed by this object.

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

AddSessionProvider (Method)

Adds an I_SessionProvider to this session manager.

Inputs

Name	IEC 61131 Type	Description
sessionProvider	I_SessionProvider	The object that is providing sessions to be managed.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the session provider was successfully added to the session manager. FALSE if an error occurred.

NextSession (Method)

Iterates over all I_Session objects referenced within the session manager and returns the next session.

Return Value

IEC 61131 Type	Description
I_Session	Reference to the next session object.

Run (Method)

This method processes all session activity and must be called once per scan.

I_SessionProvider

This interface defines a set of methods for an object that can provide I_Session objects to an I_SessionManager object.

GetSessions (Method)

Provides pointers to all I_Session objects available from this provider.

Inputs/Outputs

Name	IEC 61131 Type	Description
sessionVector	class_PointerVector	All I_Session references in the session provider are written to this vector.

Run (Method)

This method processes all session activity and must be called once per scan.

I_SELServerPoint

This interface defines common methods for all analog and digital points.

Properties

Name	IEC 61131 Type	Access	Description
PointName	STRING	R	The name of the point.
PointType	enum_PointType	R	The type of the point.
stValAsString	STRING	R	The point value as a string.

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

I_SELServerAnalogPoint

This interface defines common methods unique to analog points. This interface extends `I_SELServerPoint`.

Properties

Name	IEC 61131 Type	Access	Description
ChannelType	BYTE	R	The channel type of the point.
ScaleFactorOffset	WORD	R	The scale factor offset of the point.
ScaleFactorType	BYTE	R	The scale factor type of the point.
stValAsInt	INT	R	The point value as an integer.
stValAsLreal	LREAL	R	The point value as a long real.
stValAsReal	REAL	R	The point value as a real.

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

Classes

class_TelnetServer

This object implements a Telnet server. This server will create and manage an internal object for each Telnet session. These internal objects implement I_Session and can be accessed via the I_SessionProvider interface. This server is not intended to be used as a generic Telnet server and should only be used within the context of this library.

This Telnet server automatically attempts to negotiate the following Telnet features:

- Binary Transmission
- Suppress Go Ahead
- Echo

No other Telnet features are supported.

Implemented Interfaces

An interface defines a required set of functionality as methods and properties. As an implementer of any interface all methods and properties declared in that interface must exist as members of this class. This allows multiple generally unrelated classes to be used interchangeably for a specific feature set.

- I_SessionProvider

Initialization Inputs

Name	IEC 61131 Type	Description
listenIP	STRING(15)	The local IP address for the server to listen on. Set to 0.0.0.0 to listen on all available interfaces.
listenPort	UINT	The local port number on which the server listens.
numSessions	USINT	The maximum number of Telnet sessions to support.

Destroy (Method)

This method deallocates all memory allocated by this object. This method must be called before the object goes out of scope, otherwise a memory leak will result.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if all memory was successfully deallocated.

class_SELServerInt16

This class defines an analog point represented as a 16-bit integer value.

Implemented Interfaces

An interface defines a required set of functionality as methods and properties. As an implementer of any interface all methods and properties declared in that interface must exist as members of this class. This allows multiple generally unrelated classes to be used interchangeably for a specific feature set.

➤ I_SELServerAnalogPoint

Properties

Name	IEC 61131 Type	Access	Description
stVal	INT	R/W	The value of the point.

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

Initialize (Method)

This method initializes the point.

Inputs

Name	IEC 61131 Type	Description
pointName	STRING	The name of the point.
initialStVal	INT	The initial *stVal* of the point.
scaleFactorType	BYTE	The scale factor type for the point.
scaleFactorOffset	WORD	The scale factor offset for the point.

class_SELServerFloat32

This class defines an analog point represented as a 32-bit floating point value.

Implemented Interfaces

An interface defines a required set of functionality as methods and properties. As an implementer of any interface all methods and properties declared in that interface must exist as members of this class. This allows multiple generally unrelated classes to be used interchangeably for a specific feature set.

➤ I_SELServerAnalogPoint

Properties

Name	IEC 61131 Type	Access	Description
stVal	REAL	R/W	The value of the point.

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

Initialize (Method)

This method initializes the point.

Inputs

Name	IEC 61131 Type	Description
pointName	STRING	The name of the point.
initialStVal	REAL	The initial *stVal* of the point.
scaleFactorType	BYTE	The scale factor type for the point.
scaleFactorOffset	WORD	The scale factor offset for the point.

class_SELServerFloat64

This class defines an analog point represented as a 64-bit floating point value.

Implemented Interfaces

An interface defines a required set of functionality as methods and properties. As an implementer of any interface all methods and properties declared in that interface must exist as members of this class. This allows multiple generally unrelated classes to be used interchangeably for a specific feature set.

- I_SELServerAnalogPoint

Properties

Name	IEC 61131 Type	Access	Description
stVal	LREAL	R/W	The value of the point.

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

Initialize (Method)

This method initializes the point.

Inputs

Name	IEC 61131 Type	Description
pointName	STRING	The name of the point.
initialStVal	LREAL	The initial *stVal* of the point.
scaleFactorType	BYTE	The scale factor type for the point.
scaleFactorOffset	WORD	The scale factor offset for the point.

class_SELServerDigital

This class defines a digital point.

Implemented Interfaces

An interface defines a required set of functionality as methods and properties. As an implementer of any interface all methods and properties declared in that interface must exist as members of this class. This allows multiple generally unrelated classes to be used interchangeably for a specific feature set.

➤ I_SELServerPoint

Properties

Name	IEC 61131 Type	Access	Description
stVal	BOOL	R/W	The value of the point.

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

Initialize (Method)

This method initializes the point.

Inputs

Name	IEC 61131 Type	Description
pointName	STRING	The name of the point.
initialStVal	INT	The initial *stVal* of the point.
index	UINT	The index value for the digital point. This is used for determining the order of the Relay Word bits. The index is zero for the MSB in Row 0, 8 for the MSB in Row 1, etc.

class_SELServerString

This class defines a point represented as a string.

Implemented Interfaces

An interface defines a required set of functionality as methods and properties. As an implementer of any interface all methods and properties declared in that interface must exist as members of this class. This allows multiple generally unrelated classes to be used interchangeably for a specific feature set.

- I_SELServerPoint

Properties

Name	IEC 61131 Type	Access	Description
stVal	STRING	R/W	The value of the point.

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

Initialize (Method)

This method initializes the point.

Inputs

Name	IEC 61131 Type	Description
pointName	STRING	The name of the point.
initialStVal	STRING	The initial *stVal* of the point.

class_SEL351ServerSimulator

This class implements a server simulating an SEL-351 relay with firmware version R515. The following ASCII commands are supported:

- **2AC**
- **ACC**
- **BNA**
- **CAL**
- **CAS**
- **CEV**
- **CHI**
- **DNA**
- **EXIT**
- **ID**
- **QUIT**
- **SNS**

The following binary commands are supported:

- A546
- A5B9
- A5C0
- A5C1
- A5C2
- A5C3
- A5CD
- A5CE
- A5D1
- A5D2
- A5D3

Some commands have optional parameters that may not be supported by the simulator.

Properties

Name	IEC 61131 Type	Access	Description
RefNum	UINT	R/W	The reference number for the next event. This value will be incremented by the simulator each time an event is created.

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

Input

Inputs

Name	IEC 61131 Type	Description
Analog	struct_SEL351_Analog	Analog points for an SEL-351.
Binary	struct_SEL351_Binary	Relay Word bits for an SEL-351.
Strings	struct_SEL351_String	Strings for an SEL-351.

Outputs

Name	IEC 61131 Type	Description
ENO	BOOL	TRUE if the simulator is initialized and enabled. FALSE if initialization failed or an error occurred during runtime.

bootstrap_AddSessionManager

This method assigns the session manager that will manage all I_Session objects for this simulator. This method assumes that the session manager already contains all desired sessions. Once a session manager is added here, more session providers should not be added to the session manager.

Inputs

Name	IEC 61131 Type	Description
sessionManager	I_SessionManager	The session manager containing I_Session objects for the simulator.

CreateEvent (Method)

This method creates a new event within the simulator. When the event is created, it will be automatically assigned the next available record number. The event created by this method will contain four samples per cycle. The analog values, Relay Word bit values, and relay settings will consist of internal hard-coded values and will not match the values already set in the simulator.

Inputs

Name	IEC 61131 Type	Description
refNum	UINT	Reference number of the event
event	STRING(8)	Type of event
location	REAL	Location of the fault
current	UINT	Peak current detected during the fault
group	USINT	Group number
shot	UINT	Number of reclose events
targets	STRING	Target elements for the event

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the event was successfully created.

DeleteEvent (Method)

This method deletes an event from the simulator that was created with `CreateEvent()`.

Inputs

Name	IEC 61131 Type	Description
recNum	UINT	The record number for the event as provided by the **CHI** command.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if the event was successfully deleted. FALSE if the event did not exist or an error occurred.

Destroy (Method)

This method deallocates all memory allocated by this object. This method must be called before the object goes out of scope; otherwise, a memory leak will result.

Return Value

IEC 61131 Type	Description
BOOL	TRUE if all memory was successfully deallocated.

Run (Method)

This method handles processing communication between the server and all connected clients. This method must be called once per scan.

Processing

The `Run()` method does the following:

- Reads any pending requests from clients.
- If a valid command is received, the applicable response is generated and sent to the client.

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.

Simulating an SEL-351

Objective

This example creates a simulator for an SEL-351 Relay on an RTAC. Changing analog and digital values is also demonstrated.

Assumptions

This example assumes that an Access Point has been defined for TCP Port 23 on the RTAC. This is necessary for the simulator to communicate via Telnet to other RTACs or to a user's interactive Telnet session.

Solution

The program shown in *Code Snippet 1* shows a minimal example of running a simulator of the SEL-351. This example instantiates a Telnet server that can handle as many as ten simultaneous connections. On the first scan, the system frequency of the simulator is set to 60 Hz and the daylight-saving time Relay Word bit is set to TRUE.

Code Snippet 1 prg_SEL351

```
PROGRAM prg_SEL351
VAR
    // Create a Telnet server. The server will bind to any available IP
    // address. The server will listen on port 23 and allow up to 10
    // simultaneous connections.
    TelnetServer : class_TelnetServer('0.0.0.0', 23, 10);
    // Create the session manager for the simulator.
    SessionManager : class_SessionManager();
    // The SEL-351 simulator object.
    Sel351 : class_SEL351ServerSimulator();
    FirstScan : BOOL := TRUE;
END_VAR
```

```
IF FirstScan then
    // Add the Telnet server as a session provider to the session manager.
    SessionManager.AddSessionProvider(TelnetServer);
    // Add the session manager to the SEL-351 simulator.
    Sel351.bootstrap_AddSessionManager(SessionManager);
    // Set the simulator frequency to 60 Hz.
    Sel351.Analog._FREQ.stVal := 60;
    // Set the daylight saving time Relay Word bit to true.
    Sel351.Binary._DST.stVal := TRUE;
    FirstScan := FALSE;
END_IF

// Process communication on each scan.
Sel351.Run();
```

Simulating Multiple SEL-351 Relays

Objective

This example creates two simulators for SEL-351 Relays on a single RTAC.

Assumptions

This example assumes that an Access Point has been defined for TCP Port 23 on the RTAC. This is necessary for the simulators to communicate via Telnet to other RTACs or to a user's interactive Telnet session. For this example, it is necessary to use two network interfaces on the RTAC, one with the IP address 192.168.0.100 and the other with 192.168.1.100.

Solution

The program shown in *Code Snippet 2* shows an example of running two distinct SEL-351 simulators on the same RTAC. One simulator listens on the network interface with the IP address 192.168.0.100 and the other listens on the second interface with address 192.168.1.100.

Code Snippet 2 prg_SEL351

```
PROGRAM prg_SEL351
VAR
    // Create a Telnet server for each relay. Each server binds to a
        specific
    // IP address. Both servers listen on port 23 and allow up to 10
    // simultaneous connections.
    TelnetServer1 : class_TelnetServer('192.168.0.100', 23, 10);
    TelnetServer2 : class_TelnetServer('192.168.1.100', 23, 10);
    // Create a session manager for each simulator.
    SessionManager1 : class_SessionManager();
    SessionManager2 : class_SessionManager();
    // The SEL-351 simulator objects.
    Sel351_1 : class_SEL351ServerSimulator();
    Sel351_2 : class_SEL351ServerSimulator();
    FirstScan : BOOL := TRUE;
END_VAR
```

```
IF FirstScan then
    // Add a Telnet server as a session provider to the respective session
        manager.
    SessionManager1.AddSessionProvider(TelnetServer1);
    SessionManager2.AddSessionProvider(TelnetServer2);
    // Add the respective session manager to the SEL-351 simulator.
    Sel351_1.bootstrap_AddSessionManager(SessionManager1);
    Sel351_2.bootstrap_AddSessionManager(SessionManager2);
    // Set the first simulator frequency to 60 Hz.
    Sel351_1.Analog._FREQ.stVal := 60;
    // Set the second simulator frequency to 50 Hz.
    Sel351_2.Analog._FREQ.stVal := 50;
    FirstScan := FALSE;
END_IF

// Process communication on each scan.
Sel351_1.Run();
Sel351_2.Run();
```

Simulating an SEL-351 on Multiple Ports

Objective

This example creates a simulator for an SEL-351 relay on an RTAC that listens on multiple network ports.

Assumptions

This example assumes that an Access Point has been defined for TCP Ports 23 and 8023 on the RTAC. This is necessary for the simulator to communicate via Telnet to other RTACs or to a user's interactive Telnet session.

Solution

The program shown in *Code Snippet 3* shows an example of running a simulator of the SEL-351 that listens on Ports 23 and 8023. This example instantiates two Telnet servers, one on each port, that can handle as many as ten simultaneous connections each. Connections to these simulators will be allowed on all network interfaces.

Code Snippet 3 prg_SEL351

```
PROGRAM prg_SEL351
VAR
    // Create a Telnet server. The server will bind to any available IP
    // address. The server will listen on port 23 and allow up to 10
    // simultaneous connections. Connections will be accepted on all
        interfaces
    // due to the 0.0.0.0 IP address.
    TelnetServer1 : class_TelnetServer('0.0.0.0', 23, 10);
    // Create another server to listen on port 8023.
    TelnetServer2 : class_TelnetServer('0.0.0.0', 8023, 10);
    // Create the session manager for the simulator.
    SessionManager : class_SessionManager();
    // The SEL-351 simulator object.
    Sel351 : class_SEL351ServerSimulator();
    FirstScan : BOOL := TRUE;
END_VAR
```

```
IF FirstScan then
    // Add both Telnet servers as session providers to the session manager.
    SessionManager.AddSessionProvider(TelnetServer1);
    SessionManager.AddSessionProvider(TelnetServer2);
    // Add the session manager to the SEL-351 simulator.
    Sel351.bootstrap_AddSessionManager(sessionManager);
    FirstScan := FALSE;
END_IF

// Process communication on each scan.
Sel351.Run();
```

Creating Events on an SEL-351 Simulator

Objective

This example creates a simulator and an event for an SEL-351 Relay on an RTAC. After adding an event, it can be viewed by issuing a **CHI** or **CEV** command via Telnet.

Assumptions

This example assumes that an Access Point has been defined for TCP Port 23 on the RTAC. This is necessary for the simulator to communicate via Telnet to other RTACs or to a user's interactive Telnet session.

Solution

The program shown in *Code Snippet 4* shows a minimal example of running a simulator of the SEL-351 and adding an event. This example instantiates a Telnet server that can handle as many as ten simultaneous connections. On the first scan, an event is created. The event date and time will be set to the present date and time of the RTAC system.

Code Snippet 4 prg_SEL351

```
PROGRAM prg_SEL351
VAR
    // Create a Telnet server. The server will bind to any available IP
    // address. The server will listen on port 23 and allow up to 10
    // simultaneous connections.
    TelnetServer : class_TelnetServer('0.0.0.0', 23, 10);
    // Create the session manager for the simulator.
    SessionManager : class_SessionManager();
    // The SEL-351 simulator object.
    Sel351 : class_SEL351ServerSimulator();
    FirstScan : BOOL := TRUE;
END_VAR
```

```
IF FirstScan then
    // Add the Telnet server as a session provider to the session manager.
    SessionManager.AddSessionProvider(TelnetServer);
    // Add the session manager to the SEL-351 simulator.
    Sel351.bootstrap_AddSessionManager(SessionManager);
    // Add an event to the simulator. The reference number is 1, type is
        TRIG,
    // location of zero, maximum current of zero, group 1, shot 1, and no
    // targets. Other event values are set automatically.
    Sel351.CreateEvent(1, 'TRIG', 0, 0, 1, 1, '');
    FirstScan := FALSE;
END_IF

// Process communication on each scan.
Sel351.Run();
```

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. ➤ Must be used with R143 firmware or later.	20180921
3.5.0.1	➤ Initial release.	20150511