Modbus Server Stack

Quick Start Guide

Version 1.0

THIS PAGE IS INTENTIONALLY LEFT BLANK

Table of Contents

1. Package Information

Extract **RS_MODBUS_Server_Stack_VXX.zip** into a folder.

${PACKAGE_DIR} - <destination>/RS_MODBUS_Server_Stack_VXX/

Directory	Description
${PACKAGE_DIR}/rs_common/	Common header files (rs_config.h, rs_lib.h, rs_typedef.h …)
${PACKAGE_DIR}/rs_modbus/inc/	RS Modbus Server/Client public header files
${PACKAGE_DIR}/rs_modbus/libs/release/v1_0/full_version/	Pre-built release static libraries (x64 / x86)
${PACKAGE_DIR}/rs_modbus/libs/debug/v1_0/full_version/	Pre-built debug static libraries (x64 / x86)
${PACKAGE_DIR}/rs_modbus/apps/modbus_server_demo_app/	Modbus server demo application source
${PACKAGE_DIR}/rs_modbus/build/windows_x86_vs_ws/	Visual Studio workspace for Windows
${PACKAGE_DIR}/rs_modbus/build/linux_x64_eclipse_ws/	Eclipse workspace for Linux
${PACKAGE_DIR}/rs_modbus_src/src/	Modbus stack source files (reference)
${PACKAGE_DIR}/rs_hal_windows/ or rs_hal_linux/	Platform HAL package — choose matching your OS
${PACKAGE_DIR}/rs_platform/	Platform library (inc + pre-built libs)

2. HAL (Hardware Abstraction Layer)

Two platform-specific HAL packages are provided. Add the library for your target OS to your project linker settings:

Package	Library (release / x64)
rs_hal_windows	${PACKAGE_DIR}/rs_hal_windows/lib/release/x64/RSHAL.lib
rs_hal_linux	${PACKAGE_DIR}/rs_hal_linux/lib/release/x64/libRSHal.a

3. Windows Environment Setup

Open the Visual Studio workspace at:

${PACKAGE_DIR}/rs_modbus/build/windows_x86_vs_ws/Modbus_Server_Demo_App/

Follow:

https://www.rapidseasuite.com/documentation/rapidsea/guide/windows_pro_installation.html

Link the following static libraries in your project:

Library	Path
RSHAL.lib	${PACKAGE_DIR}/rs_hal_windows/lib/release/x64/RSHAL.lib
RSPlatform.lib	${PACKAGE_DIR}/rs_platform/libs/release/v1_0/x64/RSPlatform.lib
RSModbusServer.lib	${PACKAGE_DIR}/rs_modbus/libs/release/v1_0/full_version/x64/RSModbusServer.lib

Add to project include paths:

```
${PACKAGE_DIR}/rs_common/inc
```

4. Linux Environment Setup

Open the Eclipse workspace at:

${PACKAGE_DIR}/rs_modbus/build/linux_x64_eclipse_ws/modbus_server_demo/

Follow:
https://www.rapidseasuite.com/documentation/rapidsea/guide/linux_pro_installation.html

Link the following static libraries in your project:

Library	Path
libRSHal.a	${PACKAGE_DIR}/rs_hal_linux/lib/release/x64/libRSHal.a
libRSPlatform.a	${PACKAGE_DIR}/rs_platform/libs/release/v1_0/x64/libRSPlatform.a
libRSModbusServer.a	${PACKAGE_DIR}/rs_modbus/libs/release/v1_0/full_version/x64/libRSModbusServer.a

Add to project include paths:

```
${PACKAGE_DIR}/rs_common/inc
```

5. Prerequisite: Enable Modbus Server in rs_config.h

Before compiling, enable the Modbus Server feature in **${PACKAGE_DIR}/rs_common/inc/rs_config.h**:

```
#define RS_ENABLE_PROTOCOL_MODBUS_SERVER  1
```

6. Application Startup Code

#include "rs_lib.h" in your application. Use one of the templates below depending on your communication mode.

Modbus RTU Server

```
//! Modbus RTU Server info structure
static rs_modbus_rtu_server_instance_t rtu_server_instance;
//! Server configuration structure
static rs_modbus_server_config_t server_config = {
      .server_id   = 1,
      .comm_type   = RS_MODBUS_COMM_TYPE_RTU,
      .addr        = "COM1",
      .baud_rate   = RS_SERIAL_PORT_BAUD_115200,
      .parity      = RS_SERIAL_PORT_PARITY_DISABLED,
      .flow_ctrl   = RS_SERIAL_PORT_FLOW_CTRL_DISABLED,
      .data_length = RS_SERIAL_PORT_DATA_LENGTH_8BITS,
      .stop_bits   = RS_SERIAL_PORT_STOP_BITS_ONE,
};
int main() {
      rs_handle_t mb_server_handle = 0;
      int32_t s32_ret_val = 0;
      rs_modbus_server_instance_t *ptr_instance =
(rs_modbus_server_instance_t *)&rtu_server_instance;
      rs_modbus_server_config_t   *ptr_config   = &server_config;
      mb_server_handle = rs_modbus_server_open(ptr_instance, ptr_config);
      if (mb_server_handle > 0) {
            rs_modbus_server_run(mb_server_handle, MODBUS_SERVER_RUN);
            while (1) { rs_modbus_server_process(mb_server_handle); }
      }
      return s32_ret_val;
}
```

Modbus ASCII Server

```
//! Modbus ASCII Server info structure
static rs_modbus_ascii_server_instance_t ascii_server_instance;
static rs_modbus_server_config_t server_config = {
      .server_id   = 1,  .comm_type = RS_MODBUS_COMM_TYPE_ASCII,
```

```
      .addr        = "COM1",        .baud_rate =
RS_SERIAL_PORT_BAUD_115200,
      .parity      = RS_SERIAL_PORT_PARITY_DISABLED,
      .flow_ctrl   = RS_SERIAL_PORT_FLOW_CTRL_DISABLED,
      .data_length = RS_SERIAL_PORT_DATA_LENGTH_8BITS,
      .stop_bits   = RS_SERIAL_PORT_STOP_BITS_ONE,
};
int main() {
      rs_handle_t mb_server_handle = 0;
      int32_t s32_ret_val = 0;
      rs_modbus_server_instance_t *ptr_instance =
(rs_modbus_server_instance_t *)&ascii_server_instance;
      rs_modbus_server_config_t   *ptr_config   = &server_config;
      mb_server_handle = rs_modbus_server_open(ptr_instance, ptr_config);
      if (mb_server_handle > 0) {
            rs_modbus_server_run(mb_server_handle, MODBUS_SERVER_RUN);
            while (1) { rs_modbus_server_process(mb_server_handle); }
      }
      return s32_ret_val;
}
```

Modbus TCP Server

```
//! Modbus TCP Server info structure
static rs_modbus_tcp_server_instance_t tcp_server_instance;
static rs_modbus_server_config_t server_config = {
      .server_id       = 1,
      .comm_type       = RS_MODBUS_COMM_TYPE_TCP_IP,
      .addr            = "192.168.29.10",          // bind to all
interfaces
      .port            = 502,
      .num_conn_accept = RS_MODBUS_MAX_NUM_OF_CLIENT,
};
int main() {
      rs_handle_t mb_server_handle = 0;
      int32_t s32_ret_val = 0;
```

```
	rs_modbus_server_instance_t *ptr_instance =
(rs_modbus_server_instance_t *)&tcp_server_instance;
	rs_modbus_server_config_t   *ptr_config   = &server_config;
	mb_server_handle = rs_modbus_server_open(ptr_instance, ptr_config);
	if (mb_server_handle > 0) {
		rs_modbus_server_run(mb_server_handle, MODBUS_SERVER_RUN);
		while (1) {
			rs_modbus_server_tcp_comm_handler(mb_server_handle); //
accept clients
			rs_modbus_server_process(mb_server_handle
		}
	}
	return s32_ret_val;
}
```

7. Conclusion

This guide covered the steps to integrate the RAPIDSEA Modbus Server stack into your project. Starting from the package directory layout and selecting the appropriate HAL library for your platform (rs_hal_windows or rs_hal_linux), through configuring the project include paths and linker settings, to enabling the Modbus Server feature in rs_config.h — your project is now ready to build.

Three communication modes are supported out of the box: RTU and ASCII for serial-based deployments, and TCP/IP for Ethernet-connected devices. The startup code templates above can be used directly as the entry point for your Modbus server application.

For detailed API reference, demo walkthroughs, and advanced configuration options, visit https://www.rapidseasuite.com/documentation/rapidsea/protocols/modbus_server.html