DAKOTASOFT

FUNCTIONAL MANUAL FOR THE

MODBUS RTU PROTOCOL STACK

VERSION: 1.00

March 18, 2021

1.Contents

2. Revision History

Rev 1.00.00
Date: 3/18/21
Information:
* Initial Release.

3. Scope and Background Information

The purpose of this document is to provide documentation for the implementation of the DakotaSoft Modbus Stack. This stack can be implemented as a modbus master or slave and contains drivers for both UART RS485 and ethernet.

4. Getting Started

In order for the stack to operate properly with the UART hardware several functions must be completed by the user of the stack. These functions, which reside in the module *Modbus RS485.c*, perform the initialization of the hardware and basic I/O of data between the stack and the hardware.

4.1 Modbus RS485.c Functions That Need To Be Completed

fnDeviceModbusRS485Init – This function needs to be called once during initialization. The purpose of the function is to initialize the UART for the baud rate that is to be used. It also initializes the timer that needs to be used to determine sentence termination.

HAL_UART_MspInit – This function does the low level initialization of the UART peripheral pins.

HAL_TIM_Base_MspInit – This function does the low level initialization of the TIMER peripheral.

HAL_TIM_PeriodElapsedCallback – This function is called by the TIMER peripheral when it interrupts at the end of a sentence.

HAL_UART_TxCpltCallback – The function is called by the UART peripheral when it interrupts at the end of a sentence being sent.

HAL_UART_RxCpltCallback – This function is called by the UART peripheral when it interrupts as each byte is received.

HAL_UART_ErrorCallback – This function is called by the UART peripheral when it interrupts due to an error condition.

fnDeviceModbusRS485Tx – This function is called by the application software to cause a message to be transmitted on the UART. The UART port and a pointer to the data that is to be sent is passed into the function.

4.2 Modbus Stack Initialization

Call the following functions at powerup:

fnDeviceModbusRS485Init() – Initialize the UART peripheral

fnDeviceModbusInit() – Initialize the modbus map device registers

fnMasterModbusInit() – Initialize the modbus map master registers

4.3 Modbus Master Stack At Run Time

The following function needs to be called cyclically in order for the modbus master stack to work:

fnModbusMasterEngine() – This function requests the register data from the slave. Each time the function is called it requests data for the next register in its list. Once last register in its list is updated it starts from the beginning of the list.

4.4 Modbus Slave Stack At Run Time

There are no functions that need to be called cyclically. All requests made of the slave are responded to immediately from the callback function (interrupt).

4.5 Modules

Modbus RS485.c.h – contains the low level drivers for interfacing with the UART and Timer peripherals.

Modbus Ethernet.c.h – contains the low level drivers for interfacing with the ethernet peripheral.

Modbus Device RTU.c.h – contains the functions for handling the different modbus slave message requests.

Modbus Master RTU.c.h – contains the functions for handling the different modbus master message requests.

Modbus Includes.h – contains typedef definitions that allow for the software to be more easily ported to different hardware.

4.6 Modbus map

The structure variable: static tMODBUS_REGISTERS gstMap, contains pointers to the slave map register data.

The structure variable: static tMODBUS_MASTER_REGISTERS gstMap, contains pointers to the master map register data.