APPLICATION NOTE

AT8547: RS-485 Communications using the SAM C21 Xplained Pro and RS-485 Xplained Pro

SMART ARM Based Microcontroller

Introduction

The SAM C21 introduces an enhanced SERCOM peripheral, configurable as a number of different serial communication modes. There are up to six configurable modules that can act as an enhanced USART, UART, SPI, I2C, and LIN.

This application note addresses the use of the SERCOM in UART mode using the hardware Transmitter Enable (TE) function to set the driver enable and receiver enable pins on an RS-485/422 transceiver, resulting in half-duplex operation.

The RS-485 Xplained Pro board, along with the SAM C21 Xplained Pro, are used to demonstrate RS-485 communications via the SAM C21 SERCOM. The RS-485 Xplained Pro has numerous jumpers to enable Full-Duplex, Half-Duplex, and Loopback operation.

Features

- SAM C21 Hardware Driver/Receiver Enable via RTS/TE pin
- XPro wing board with RS-485/422 transceiver
- Screw terminals for easy prototyping
- Half/Full Duplex operation via jumpers

Table of Contents

1 Prerequisites

The example firmware requires the following:

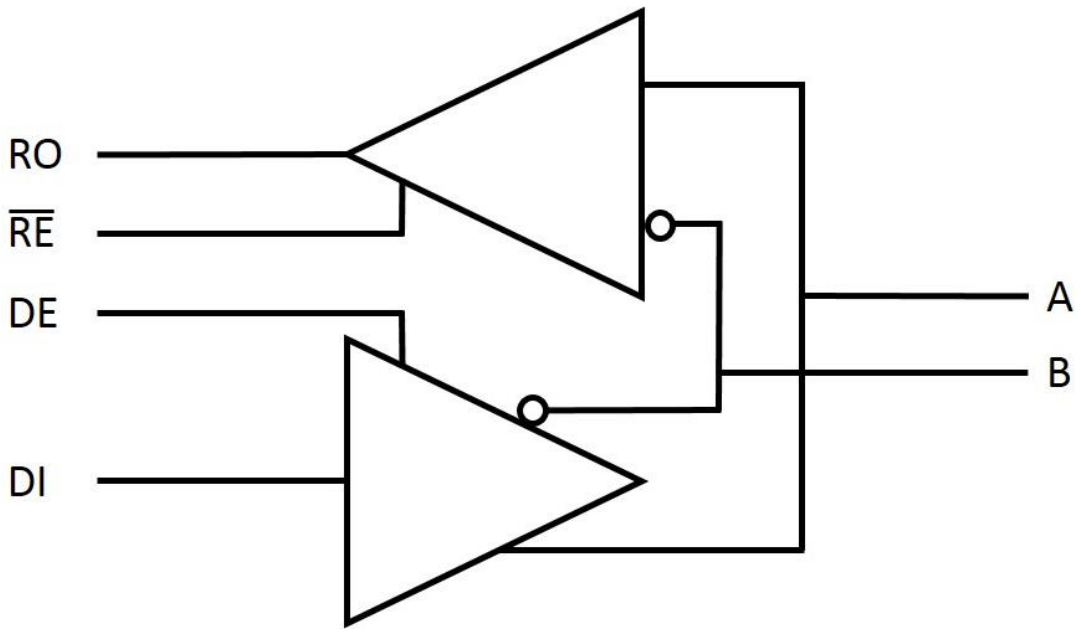
1. Atmel Studio version 6.2.1563 – Service Pack 2 or higher.
2. Atmel Software Framework (ASF) version 3.21.8 or higher.
3. Xplained Pro SAM C21 development board (2x).
4. RS-485 Xplained Pro Wing Board (2x).

2 Module Overview

2.1 Description

This application note and accompanying application example uses a SERCOM module configured as an asynchronous UART with hardware TE control for RS-485 communications.

Figure 2-1. RS-485 Half Duplex Configuration



In half-duplex applications, the Receiver Enable (RE) and Driver Enable (DE) on the physical transceiver are connected together as they are opposite polarity inputs. When the connection is held logic low, the receiver is active and the driver is forced to a high impedance state. Alternatively, when the connection is driven logic level high, the receiver is forced into a high impedance state and the driver is enabled.

2.2 Register Interface

There are two registers associated with RS-485 communications using the SERCOM UART. The CTRLA register configures the SERCOM pad(s) to include the TE pin (Same as RTS). Configuration is accomplished by writing 0x03 to CTRLA.TXPO[1:0]. The second register is the CTRLC and is used for implementing a guard time, where the TE pin will remain high n bit counts after the frame has been transmitted. Writing to CTRLC.GTIME[2:0] will set the guard time.

Additional registers and features are outlined in the ASF Programmers Manual application note: AT03256: SAM Serial USART Driver (SERCOM USART).

3 Functional Description

RS-485, or EIA-485, is a physical layer standard used for differential balanced line digital communication systems. The standard supports multi-drop links and can span distances >1000m using twisted pair cable. The unique feature of this standard is the use of a Driver Enable (listed as DE on most interfaces) to explicitly enable drivers. Thus, linear bus topologies are possible using a simple two-wire, half-duplex, interface. In practice, however, many applications implement a 3-wire interface to limit the common mode voltage on the receiver inputs.

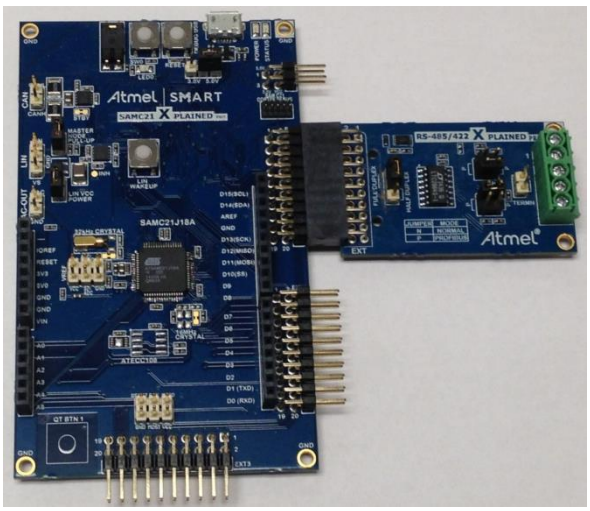
Although it is possible to manage the DE pin using standard I/O, it is much simpler to implement in the USART hardware on devices that support this like the SAM C21. This also has the added benefit of meeting critical timing specifications associated with some fieldbus networks that utilize the EIA-485 physical layer, i.e. Modbus and Profibus.

The example firmware included with this application note uses two SAM C21 Xplained Pro development boards along with two RS-485 wing boards. The wing boards contain a transceiver that convert the TTL level UART RX and TX lines to differential signals. The onboard transceiver (LTC2864) has the outputs routed to screw terminals where half-duplex 2-wire or full-duplex 4-wire can be connected. The RE and DE pins are controlled via the UART using the GPIO1_RTS line on pin 5 of the wing board.

3.1 Basic Operation

3.1.1 Hardware Setup

Figure 3-1. RS-485 Xplained Pro Board Connected to SAM C21 XPro



To configure the RS-485 board for this application example:

1. Install jumper between pins 2 and 3 on J101.
   a. This enables the TE pin to manage both DE and RE for Half-Duplex operation.
2. Install jumper between pins 3 and 4 on J102.
   a. This removes the Profibus Termination for A/B.
3. Install jumper between pins 1 and 2 on J103.
   a. This removes the Profibus Termination for Z/Y.
4. Install jumpers on J106 and J107.

a. This connects A/Y and B/Z to form half-duplex A/B differential pair.

b. Alternatively, this can be done by jumpering RXP to TXP and RXN to TXN, or pins 2 and 5 and pins 3 and 4 on the screw terminal block.

Next, connect the RS-485 board to the SAM C21 Xplained Pro board using Extension 1. Repeat for second test board and connect the two RS-485 boards via a single twisted pair cable.

4 Firmware Implementation

SERCOM drivers are a part of Atmel Studio Framework. Easy to use APIs have been provided to use the peripheral. There are two example projects included in the ASF as part of the firmware support package for the SAM C21 Xplained Pro Development Board. These examples demonstrate the basic use of the SERCOM in polled and interrupt, or callback, modes. Both these implementations are demonstrated in the application example accompanying this application note.

4.1 Peripheral APIs

Peripheral APIs provided in the Atmel ASF allow for configuration, initialization, enabling and reading the SERCOM. The ASF drivers provide APIs to set/reset each bit in the SERCOM configuration registers. The APIs provided by the driver are:

- `void usart_get_config_defaults(struct usart_config *const config)`
- `enum status_code usart_init(struct usart_module *const module,Sercom *const hw, const struct usart_config *const config);`
- `void usart_enable(const struct usart_module *const module)`
- `void usart_disable(const struct usart_module *const module)`
- `void usart_reset(const struct usart_module *const module)`

4.2 Callback APIs

In addition to the peripheral APIs, there are a number of callback APIs to support interrupt-based usage. These APIs include:

- `void usart_register_callback(struct usart_module *const module,usart_callback_t callback_func,enum usart_callback callback_type)void sdadc_unregister_callback(struct sdadc_module *module,enum sdadc_callback callback_type)`
- `void usart_unregister_callback(struct usart_module *const module,enum usart_callback callback_type)void sdadc_disable_callback(struct sdadc_module *const module, enum sdadc_callback callback_type)`
- `void usart_enable_callback(struct usart_module *const module, enum usart_callback callback_type)`
- `void usart_disable_callback(struct usart_module *const module, enum usart_callback callback_type)`
- `enum status_code usart_write_job(struct usart_module *const module,const uint16_t *tx_data)`
- `enum status_code usart_read_job(struct usart_module *const module,uint16_t *const rx_data)void sdadc_abort_job(struct sdadc_module *module_inst,enum sdadc_job_type type)`
- `enum status_code usart_write_buffer_job(struct usart_module *const module, uint8_t *tx_data, uint16_t length)`

- `enum status_code usart_read_buffer_job(struct usart_module *const module, uint8_t *rx_data, uint16_t length)`

4.3 Support APIs

The SERCOM can be mapped for STDIO usage to support built-in standard I/O functions, such as printf and scanf:

- `void stdio_serial_init(struct usart_module *const module, usart_inst_t const hw, const struct usart_config *const config)`

4.4 RS-485 Half-Duplex Example

This application note is accompanied by example firmware. This application has been developed using Atmel Studio using the Atmel Software Framework, or ASF.

4.4.1 Description

The firmware for this utilizes two serial communication channels per board. The first SERCOM (SERCOM 4) is mapped to use the on-board EDBG CDC channel and is connected via the USB debug connector. SERCOM 3 is routed the Extension 1 header on the SAM C21 Xplained Pro and is used for RS485 communications.

SERCOM 4 is configured using the built-in DEFINES found in the header file samc21_xplained_pro.h. See Table 4-1. Instead of calling the usart_init function, the stdio_serial_init function is used (see support APIs). This will map SERCOM 4 to the EBDG CDC module and allow the use of STDIO functions on this port.

Table 4-1. STDIO UART DEFINES

- `#define CONF_STDIO_USART EDBG_CDC_MODULE`
- `#define CONF_STDIO_MUX_SETTING EDBG_CDC_SERCOM_MUX_SETTING`
- `#define CONF_STDIO_PINMUX_PAD0 EDBG_CDC_SERCOM_PINMUX_PAD0`
- `#define CONF_STDIO_PINMUX_PAD1 EDBG_CDC_SERCOM_PINMUX_PAD1`
- `#define CONF_STDIO_PINMUX_PAD2 EDBG_CDC_SERCOM_PINMUX_PAD2`
- `#define CONF_STDIO_PINMUX_PAD3 EDBG_CDC_SERCOM_PINMUX_PAD3`
- `#define CONF_STDIO_BAUDRATE 9600`

SERCOM 3 is configured using the built-in DEFINES found in the same header file. Beyond the standard MUX and PAD settings, the guard time is set using the configuration structure. See Table 4-2. For this application, the guard time is set at 1 bit time. Also, notice that PAD 2 is routed to PA20 function D, which is the TE pin connected to the RS-485 transceiver.

Table 4-2. RS-485 SERCOM Configuration

- `config_usart.baudrate = 9600;`
- `config_usart.mux_setting = EXT1_RS485_SERCOM_MUX_SETTING;`
- `config_usart.pinmux_pad0 = EXT1_RS485_SERCOM_PINMUX_PAD0;`
- `config_usart.pinmux_pad1 = EXT1_RS485_SERCOM_PINMUX_PAD1;`
- `config_usart.pinmux_pad2 = EXT1_RS485_SERCOM_PINMUX_PAD2;`
- `config_usart.pinmux_pad3 = EXT1_RS485_SERCOM_PINMUX_PAD3;`
- `config_usart.rs485_guard_time = RS485_GUARD_TIME_1_BIT;`

SERCOM 3 (RS485) is also configured to use interrupts and callbacks for both transmit and receive. Using these callback functions, the application will automatically echo characters and display debug information.

To start, open a terminal program for each board connected to the respective CDC ports. Configure for 9600, 8, N, 1, and set up local echo and CR + LF on receive carriage return. Next, configure one board as a slave by pressing 2. In the master terminal window, type a letter from a-z or A-Z and it will be sent to the slave and echoed back.

Figure 4-1. Terminal Windows Start-Up Menu

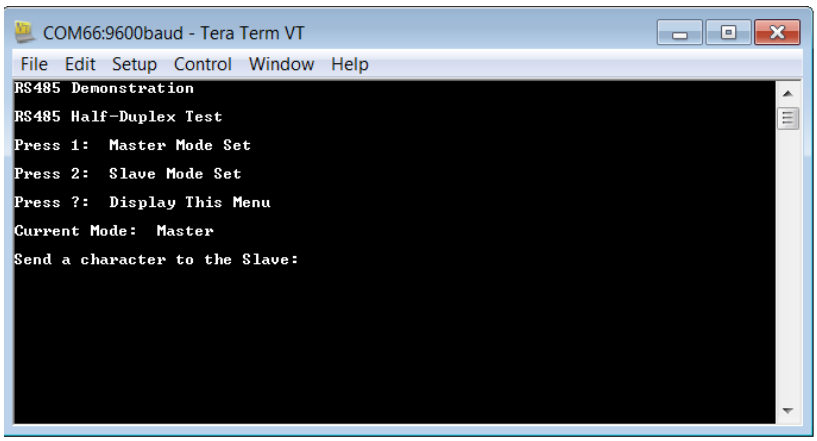
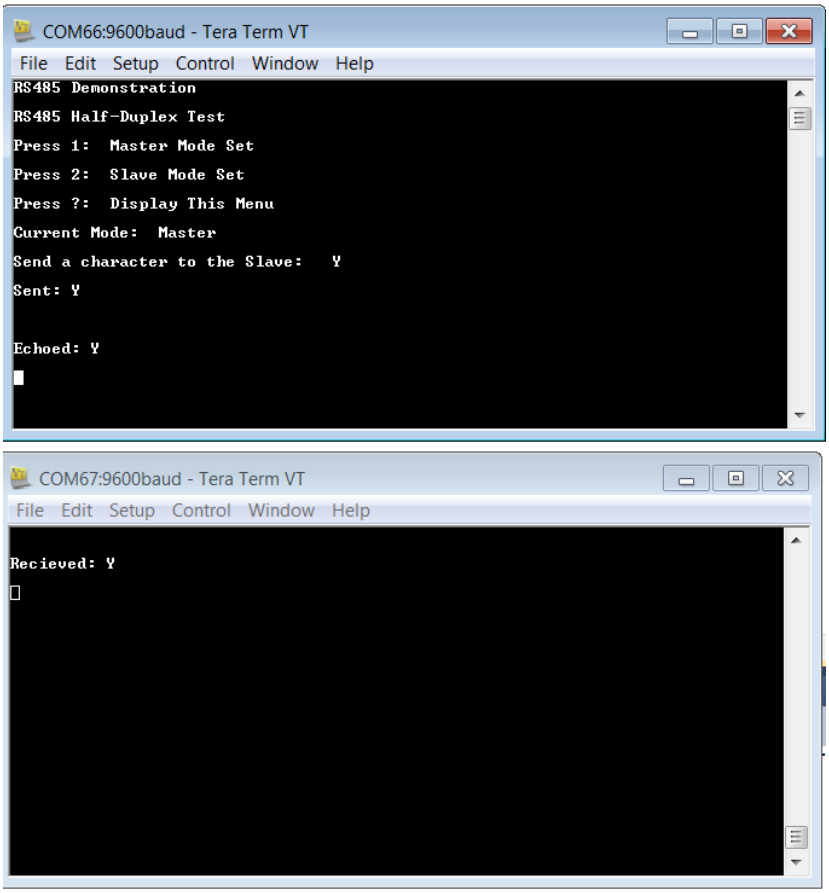


Table 4-3. Terminal Master Slave Exchange



5 Software License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6 Revision History

Doc Rev.	Date	Comments
42468A	06/2015	Initial document release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

© 2014 Atmel Corporation. / Rev.:Atmel-42468A-RS-485-Communications-using-the-SAM-C21-Xplained-Pro-and-RS-485-Xplained-Pro_ApplicationNote_062015.

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected® logo, and others are the registered trademarks or trademarks of ARM Ltd. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.