

1 Trademarks

Stellaris, StellarisWare are registered trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

Software UART for Stellaris® Microcontrollers

Chris Lande

ABSTRACT

The universal asynchronous receiver transmitter (UART) is a widely used serial communications peripheral used in many applications for connection to legacy devices. Many systems use multiple instances of legacy devices and modern microcontrollers generally support this. However, in some cases additional instances beyond the hardware limit may be required. This application report looks at an implementation of a software-based UART that, using general-purpose input/outputs (GPIO), allows you to overcome the hardware limitations or lack of dedicated peripherals. Project collateral discussed in this app report can be downloaded from the following URL: <http://www.ti.com/lit/zip/SPMA017>.

Contents

2	Software UART Initialization.....	2
3	Software UART Performance Limitations.....	3
4	Software UART Receiver	4
5	Software UART Transmitter	4
6	API Functional Description	4
7	Conclusion	8
8	References	8

2 Software UART Initialization

This implementation uses two GPIOs, one for transmit (TX), and one for receive (RX) and two timers, a single 32-bit timer configured as two individual 16-bit timers. One 16-bit timer is used to generate the transmit tick and the other is used for the receive tick. Each timer interrupt function is loaded into the nested vectored interrupt controller (NVIC) table in the `startup.c` file. The GPIOs used in this example both belong to the same GPIO bank, for example, BANK 'C' IO 0, 1.

2.1 Inputs and Outputs

The GPIO bank and individual GPIOs used for TX, RX are configured in the `sw_uart_basic.c` file by way of definitions. TX and RX GPIOs are configured by the `SWU_GPIO_TX` and `SWU_GPIO_RX` definitions, respectively. Note that in this implementation, both RX and TX GPIOs use the same GPIO bank via the `SWU_GPIO_PORT_BASE` definition.

2.2 Software UART Port Configuration

The baud rate, number of data-bits, parity, and stop-bits are also configured by the `SW_UARTConfigSetExpClk` function. It has been tested with the following configurations. ⁽¹⁾

Baud Rate:	9600, 19200, 38400, 57600, 115200
Data Bits:	8
Parity:	NONE
Stop Bits:	1

⁽¹⁾ For running clock frequency limitations, see [Section 3](#).

There are two functions used to configure and initialize the software UART:

- `SW_UARTConfigSetExpClk`
Sets up baud-rate, number of data-bits, and number of stop-bits.
- `SW_UARTEnable`
Enables the timers and interrupts used by software UART.

Each of the above functions is detailed in [Section 6](#).

3 Software UART Performance Limitations

The software UART has been tested through a range of system clock frequencies for both data transmission and reception. Both the receive and transmit functions are independently interrupt-driven. The receive functionality has a higher priority than transmit. Therefore, there will be some cases, typically seen at lower system frequencies, where transmission can be interrupted by a received event resulting in the transmission failing.

[Table 1](#) shows the tested system frequencies and attainable baud rates for character transmission.

Table 1. Transmit Performance Versus System Clock Frequency

System Clock Frequency (MHz)	Baud Rate				
	9600	19200	38400	57600	115200
50	√	√	√	√	√
40	√	√	√	√	√
33	√	√	√	√	√
25	√	√	√	√	√
20	√	√	√	√	√
12.5	√	√	√	√	√
8	√	√	√	√	√

[Table 2](#) shows the tested system frequencies and attainable baud rates for character reception.

Table 2. Receive Performance Versus System Clock Frequency

System Clock Frequency (MHz)	Baud Rate				
	9600	19200	38400	57600	115200
50	√	√	√	√	√
40	√	√	√	√	√
33	√	√	√	√	√
25	√	√	√	√	√
20	√	√	√	√	√
12.5	√	√	√	√	-
8	√	√	√	-	-

4 Software UART Receiver

The RX GPIO is configured as an input that interrupts on a falling edge. The interrupt is used to detect a valid start condition.

There are three functions available to handle character reception:

- `SW_UARTCharsAvail`
Returns the number of characters in the receive buffer.
- `SW_UARTCharGet`
Returns a character if there is one available in the receive buffer. If there is no character available, it blocks until there is.
- `SW_UARTCharGetNonBlocking`
Returns a character if there is one available in the receive buffer. If there is no character available, it returns an error code.

Each of the above functions is described in detail in [Section 6](#).

4.1 Receive Buffer Configuration

A circular buffer is implemented for received characters. The default size of this buffer is 16 characters. The size is configured by a definition in the `sw_uart.c` file in the section labeled “Receive buffer/Variables.” The `#define` is labeled as below. This value must be 1 or greater to ensure correct operation.

```
#define RX_BUFFER_SIZE  
    16
```

5 Software UART Transmitter

The TX GPIO is configured as an output. There are three functions available to handle character transmission:

- `SW_UARTSpaceAvail`
Checks whether there is space available in the transmit buffer.
- `SW_UARTCharPut`
Puts a character in the transmit buffer if there is space available. If there is no space available, it blocks until there is.
- `SW_UARTCharPutNonBlocking`
Puts a character in the transmit buffer if there is space available. If there is no space available, it returns an error code.

Each of the above functions is described in detail in [Section 6](#).

5.1 Transmit Buffer Configuration

A circular buffer is implemented for characters to be transmitted. The default size of this buffer is 16 characters. The size is configured by a definition in the `sw_uart.c` file in the section labeled “Transmit buffer/Variables.” The `#define` is labeled as below. This value must be 1 or greater to ensure correct operation.

```
#define TX_BUFFER_SIZE 16
```

6 API Functional Description

6.1 Existing Compatibility

To maintain compatibility with the existing StellarisWare® UART API functionality, all of the functions used in the software UART have the `ulBase` parameter. In the software UART implementation, this parameter is currently unused.

6.2 Functions

```

long SW_UARTCharGet (unsigned long ulBase);
long SW_UARTCharGetNonBlocking (unsigned long ulBase);
void SW_UARTCharPut (unsigned long ulBase, unsigned char ucData);
tBoolean SW_UARTCharPutNonBlocking (unsigned long ulBase, unsigned char ucData);
tBoolean SW_UARTCharsAvail (unsigned long ulBase);
void SW_UARTConfigSetExpClk (unsigned long ulBase, unsigned long ulUARTClk, unsigned long
ulBaud, unsigned long ulConfig, unsigned long ulPortBase, unsigned long ulTxPin, unsigned
long ulRxPin);
void SW_UARTDisable (unsigned long ulBase);
void SW_UARTEnable (unsigned long ulBase);
tBoolean SW_UARTSpaceAvail (unsigned long ulBase);

```

6.3 API Function Documentation

Table 3. API Functions

Title	Page
SW_UARTCharGet — Gets a character from the software UART receiver.	5
SW_UARTCharGetNonBlocking — Receives a character from the software UART.	5
SW_UARTCharPut — Sends a character via the software UART, blocking if necessary.	6
SW_UARTCharPutNonBlocking — Sends a character via the software UART if space exists in the transmit buffer.	6
SW_UARTCharsAvail — Determines if there are any characters in the receive buffer.	6
SW_UARTConfigSetExpClk — Sets the configuration of a UART.	7
SW_UARTDisable — Disables transmission and reception.	7
SW_UARTEnable — Enables transmission and reception.	7
SW_UARTSpaceAvail — Determines if there is any space in the transmit buffer.	8

SW_UARTCharGet *Gets a character from the software UART receiver.*

Prototype long
SW_UARTCharGet(unsigned long ulBase)

Parameters ulBase is not used and is for compatibility with HW UART StellarisWare API calls.

Description Gets a character from the receive buffer for the software UART receiver. If there are no characters available, this function blocks until a character is received before returning.

Returns Returns the character read from the software UART, cast as a long.

SW_UARTCharGetNonBlocking *Receives a character from the software UART.*

Prototype long
SW_UARTCharGetNonBlocking(unsigned long ulBase)

Parameters ulBase is not used and is for compatibility with HW UART StellarisWare API calls.

Description Gets a character from the receive buffer for the software UART.

Returns Returns the character read from the software UART, cast as a long. -1 is returned if there are no characters present in the receive buffer.

SW_UARTCharPut — *Sends a character via the software UART, blocking if necessary.*

www.ti.com

SW_UARTCharPut *Sends a character via the software UART, blocking if necessary.*

Prototype	void SW_UARTCharPut(unsigned long ulBase, unsigned char ucData)
Parameters	ulBase is not used and is for compatibility with HW UART StellarisWare API calls. ucData is the character to be transmitted.
Description	Sends the ucData character to the transmit buffer for the software UART. If there is no space available in the transmit buffer, this function blocks until there is space available before returning.
Returns	None.

SW_UARTCharPutNonBlocking *Sends a character via the software UART if space exists in the transmit buffer.*

Prototype	tBoolean SW_UARTCharPutNonBlocking(unsigned long ulBase, unsigned char ucData)
Parameters	ulBase is not used and is for compatibility with HW UART StellarisWare API calls. ucData is the character to be transmitted.
Description	Writes the ucData character to the transmit buffer for the software UART. This function does not block, so if there is no space available, then false is returned and the application must retry the function later.
Returns	Returns true if the character is successfully placed in the transmit buffer, or false if there is no space available in the transmit buffer.

SW_UARTCharsAvail *Determines if there are any characters in the receive buffer.*

Prototype	tBoolean SW_UARTCharsAvail(unsigned long ulBase)
Parameters	ulBase is not used and is for compatibility with HW UART StellarisWare API calls.
Description	This function returns a flag indicating whether there is data available in the receive buffer.
Returns	Returns true if there is data in the receive buffer, and false if there is no data in the receive buffer.

SW_UARTConfigSetExpClk *Sets the configuration of a UART.*

Prototype	<pre>void SW_UARTConfigSetExpClk(unsigned long ulBase, unsigned long ulUARTClk, unsigned long ulBaud, unsigned long ulConfig, unsigned long ulPortBase)</pre>
Parameters	<ulbase (number="" <="" and="" api="" baud="" bits,="" calls.="" clock="" compatibility="" data="" desired="" for="" format="" hw="" is="" module.="" not="" number="" of="" parity).="" port="" rate="" rate.="" stellarisware="" stop="" supplied="" td="" the="" to="" uart="" ulbaud="" ulconfig="" uluartclk="" used="" with=""> </ulbase>
Description	<p>This function configures the UART for operation in the specified data format. The baud rate is provided in the ulBaud parameter and the data format in the ulConfig parameter. The ulConfig parameter is the logical OR of three values: the number of data bits, the number of stop bits, and the parity. UART_CONFIG_WLEN_8, UART_CONFIG_WLEN_7, UART_CONFIG_WLEN_6, and UART_CONFIG_WLEN_5 select from eight to five data bits per byte. UART_CONFIG_STOP_ONE and UART_CONFIG_STOP_TWO select one or two stop bits, respectively. UART_CONFIG_PAR_NONE, UART_CONFIG_PAR_EVEN, UART_CONFIG_PAR_ODD, UART_CONFIG_PAR_ONE, and UART_CONFIG_PAR_ZERO select the parity mode (no parity bit, even parity bit, odd parity bit, parity bit always one, and parity bit always zero, respectively). The peripheral clock is the same as the processor clock. This is the value returned by SysCtlClockGet(), or it can be explicitly hard-coded if it is constant and known (to save the code/execution overhead of a call to SysCtlClockGet()).</p>
Returns	None.

SW_UARTDisable *Disables transmission and reception.*

Prototype	<pre>void SW_UARTDisable(unsigned long ulBase)</pre>
Parameters	ulBase is not used and is for compatibility with HW UART StellarisWare API calls.
Description	Clears the UARTEN, TXE, and RXE bits, then waits for the end of transmission of the current character, and flushes the transmit buffer.
Returns	None.

SW_UARTEnable *Enables transmission and reception.*

Prototype	<pre>void SW_UARTEnable(unsigned long ulBase)</pre>
Parameters	ulBase is not used and is for compatibility with HW UART StellarisWare API calls.
Description	Sets the UARTEN, TXE, and RXE bits, and enables the transmit and receive buffers.
Returns:	None.

SW_UARTSpaceAvail — *Determines if there is any space in the transmit buffer.*

www.ti.com

SW_UARTSpaceAvail *Determines if there is any space in the transmit buffer.*

Prototype	tBoolean SW_UARTSpaceAvail(unsigned long ulBase)
Parameters	ulBase is not used and is for compatibility with HW UART StellarisWare API calls.
Description	This function returns a flag indicating whether there is space available in the transmit buffer.
Returns	Returns true if there is space available in the transmit buffer, or false if there is no space available in the transmit buffer.

7 Conclusion

The software-based UART can be used on any Stellaris product to extend the number of available UARTs available to an application. When the number of hardware-based UARTs is not sufficient, the software-based UART gives the application access to additional UARTs. Special care should be taken when using the software-based UART to ensure that enough processing time is given to the software UART in order to allow it to function at all baud rates. As the required baud rates get higher, the processing time required to handle the baud rate will increase.

8 Refernces

- *Tiva™ C Series Data Sheet* (individual device-specific documents available through the [product folders](#))
- *Tiva™ C Series Errata* (individual device-specific documents available through the [product folders](#))
- *Tiva™ C Series ROM User's Guide* (individual device-specific documents available through the [product folders](#))
- [TivaWare™ Peripheral Driver Library for C Series](#)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2019, Texas Instruments Incorporated