Software UART for Stellaris® Microcontrollers

Application Note

AN01270-01

Copyright

Texas Instruments
108 Wild Basin, Suite 350
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
http://www.luminarymicro.com

Table of Contents

Introduction

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 notes looks at an implementation of a software-based UART which, using general-purpose input/outputs (GPIO), allows the user to overcome the hardware limitations or lack of dedicated peripherals.

Software UART Initialization

This implementation uses two GPIOs, one for transmit (TX), one for receive (RX) and two timers, in this implementation 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.

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

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

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 the “API Functional Description” on page 7.

1.See “Software UART Performance Limitations” on page 5 for running clock frequency limitations.

Software UART Performance Limitations

DescriptionThe 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

		Baud Rate				
		9600	19200	38400	57600	115200
System Clock Frequency (MHz)	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

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

Software UART Receiver

The RX GPIO is configured as an input which will interrupt 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 the “API Functional Description” on page 7.

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
```

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 the “API Functional Description” on page 7.

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
```

API Functional Description

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.

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);
```

API Function Documentation

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.

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:

```
void
SW_UARTConfigSetExpClk(unsigned long ulBase,
unsigned long ulUARTClk,
unsigned long ulBaud,
unsigned long ulConfig,
unsigned long ulPortBase)
```

Parameters:

`ulBase` is not used and is for compatibility with HW UART StellarisWare™ API calls.
`ulUARTClk` is the rate of the clock supplied to the UART module.
`ulBaud` is the desired baud rate.
`ulConfig` is the data format for the port (number of data bits, number of stop bits, and parity

Description:

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()`).

Returns:

None.

SW_UARTDisable

Disables transmission and reception.

Prototype:

```
void
SW_UARTDisable(unsigned long ulBase)
```

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:

```
void
SW_UARTEnable(unsigned long ulBase)
```

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.

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.

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.

References

The following documents and source code are available for download at www.luminarymicro.com:

- *Stellaris LM3Snnnn microcontroller data sheet* (where nnnn is the device number), Publication Number DS-LM3Snnnn
- Stellaris Microcontrollers Soft UART software, Order Number SW01270
- StellarisWare® Driver Library, Order number SW-DRL
- *StellarisWare® Driver Library User's Manual*, publication number SW-DRL-UG

Important Notice

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless