**Introduction**

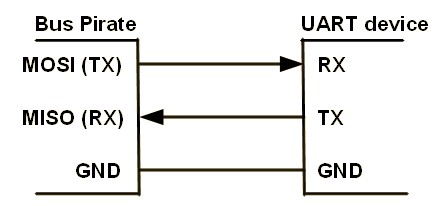
In the electronic world, a lot of thing that can be controlled by electronic and the purpose is to make human have convenient and better live. If the thing was controlled by electronic, that must have a sense and controller to monitor the situation. Furthermore, in between the controlled and sense, another important thing that is communication interface. The communication interface is a hardware device that usually to assist hardware to communicate another hardware. For example, if the file or photo in smart phone that need send to computer, the USB protocol is one of way to assist smart phone send the file to computer. To fulfill the market demand, the communication interface was developed to produce different kind of capability protocol such as USB, UART, SPI and so no.

In this report, the UART communication interface is discussed and investigated to have a better understanding and learn how to apply UART device. The UART was investigated in STM32F429 board.

The UART is term for “Universal Asynchronous Receiver Transmitter”. The “Universal” indicates that data format and transmission speeds are configurable and “Asynchronous” indicates transmission of data without the use of an external clock signal. The UART performs serial-to-parallel conversion on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU. The UART includes control capability and processor interrupt system that can be customized by programmer to service the high-priority instruction.

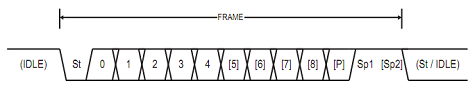
Besides that, the UART also includes a programmable baud generator that can be configure the speeds of transmission by use the equation that is given by data manual to calculate programmed of value.

**Theory**



Basically, the UART is bidirectional communication that requires a minimum of two pin such as transmission line (TX) and reception line (RX) as show in diagram X. At the same diagram, the ground pin (GND) is also common pin for any UAST because this pin to ensure the ground level between master and slave is the same. Otherwise, when transmission is occurred, the transmit signal is received by slave that can not to read.

**UAST Data Formal Description**



The UAST transmits or received data in serial form. In diagram XX, that is show the common data frame formal. At begin, the first bit is called “start bit” (St) at low state to warn other peripheral device that should started to receive data.

From 0th bit to 8th bit in frame, these data was set by software. The word length may be selected as being either 8 or 9 bit by programming the master. After end of data (8th bit), the parity bit is appeared to let other peripheral receiver checks the received data that is correct or not.

At the end, the last bit is called “stop bit” (Sp) at high state to warn other peripheral device that should stop to receive data. The length of stop bit may be selected 1 or 2 bit by programmer.

**Transmission**

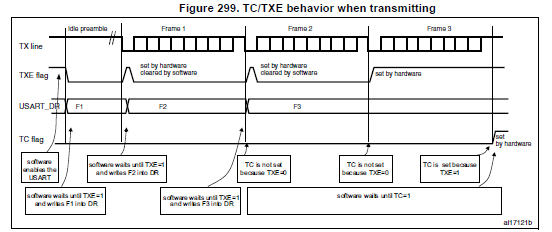
This instruction is referred by STM32F4XX data manual. Some of the specific term that is used in the STM32F4XX data manual.

The UART transmitter section includes a Transmit Data Register (TDR) and a Transmit Shift Register. When a transmission is taking place, a write instruction to the UAST Data Register (DR) stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the UAST Data Register place the data directly in the shift register, the data transmission starts, and the TXE bit is immediately.

If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART\_CR1 register.

After writing the last data into the USART\_DR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low-power mode.



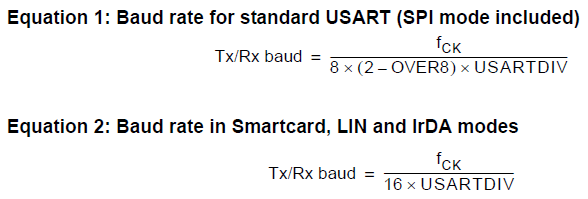
**Reception**

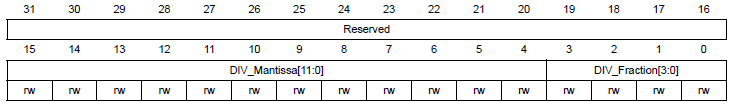
The UART receiver section includes a Receive Data Register (RDR) and a Receive Shift Register. When a character is received, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read.

The error flags can be set if a frame error, noise or an overrun error has been detected during reception.

**Result and Discussion**

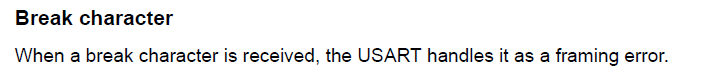
Baud Rate Algorithm



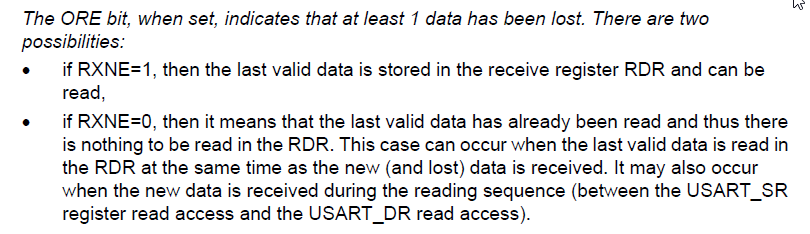


Word Length

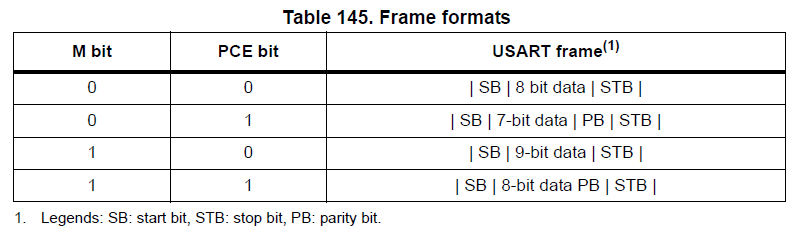
Framing Error



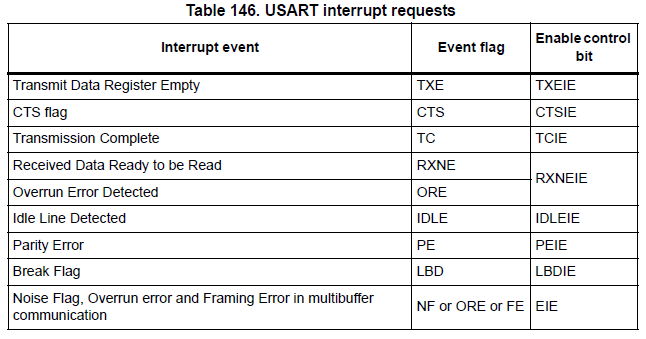
Overrun Error



Parity Control



Interrupt Event



Continuous Communication Using DMA

This section that will discuss about how the DMA is implemented in the UART peripheral. When the UART is implemented, the receiver and transmitter will be received and send the data. The CPU that should to execute the instruction in order or interrupt. If the CPU need to handle multi task, the execution of reception and transmission will reduce the processing power and lower down the efficiency of CPU. So, the DMA peripheral is used to improve the weakness for the UAST execution.

The DMA is stand for “Direct Memory Access” that is used for provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be transferred by DMA without any CPU action. The benefit for CPU that is keeps CPU resources free and focus for other operations.

In the STM32F429, that include 2 of DMA those contain 8 of stream and each stream includes 8 channel. In this experiment, The UART5 is used to implement the DMA for transmission and reception. Following the STM32F429 data sheet, the DMA1 was decided to service the UART5. The channel 4 is selected in stream 7 for UART transmitter. The channel 4 is selected in stream 0 for UART receiver.

Configuration

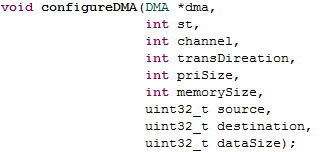


Figure XX shows the configure function for DMA in C language. This function that is more user friendly for configure DMA. This is because whatever peripherals implement the DMA, the function provide user who can conveniently select DMA1 or DMA2. Table XX shows the parameter in the “configureDMA” function.

|  |  |
| --- | --- |
| Parameter | Explanation |
| dma | To configure which DMA is used to service the peripheral |
| st | To configure which stream is decided for peripheral |
| channel | To configure which channel is decided for peripheral |
| transDireation | To configure the data transfer between peripheral and memory or between memory and memory. |
| priSize | To select the size of data in peripheral |
| memorySize | To select the size of data in memory |
| source | To configure the source in DMA |
| destination | To configure the destination in DMA |
| dataSize | To configure how many data need to transfer |

Figure XX shows the example for configure the DMA in “configureDMA” function. In the Figure XX, the first row shows the configuration for UART5 receiver and the second row shows configuration for UART5 transmitter.



Figure XX shows the function which to enable DMA to service the UASR transmitter and receiver. Notice, the DMA is configured for UART transmission and reception while the enable bit must set to 0.



Figure XX shows the function which to allow the UART is serviced by DMA. This function can used for other UART.

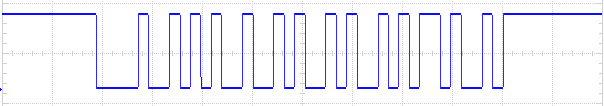


Discover

Transmission

Following the STM32F429 data sheet, the size of data register is 32-bit in UART but only 8-bit of size of data register can be availed to store the data for transmission. This experiment shows when the DMA is used to service the UAST transmitter, the peripheral data size (what kind of data size is used to transfer the data between UART data register and DMA FIFO register) is set by different data size such as byte (8-bit), half-word (16-bit) and word (32-bit) to observe what happen is take place. The data memory size is maintain byte (8-bit). In the configuration of DMA, the UART data register is assign for destination and the certain memory is assign for source.

MSIZE is byte and PSIZE is byte



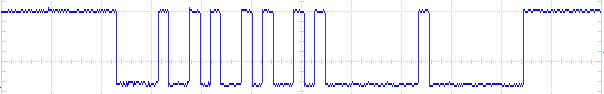
‘H’

‘I’

‘J’

‘K’

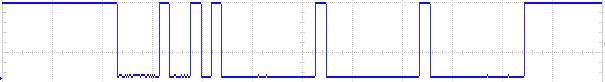
MSIZE is byte and PSIZE is half-word



‘J’

‘H’

MSIZE is byte and PSIZE is word



‘H’

Figure XX, XX, and XX show the results for the peripheral data size is set by different kind of data size and the memory data size is maintain. This experiment the UART transmitter is requested to send 4 of data character such as ‘H’, ‘I’ , ‘J’ and ‘K’.

In Figure XX, the result shows UART transmitter was perfectly send the 4 of data character and compare with Figure XX and Figure XX there show UART transmitter did not send the 4 of data character completely. The reason is the UART data register that only can store one byte at one times. If the DMA FIFO register transfer data that is more than one byte at one times, only the first byte that is detected and store into the UART data register so other byte that will be disappear. However, the number of transfer times is already set in the DMA configuration, the DMA will transfer ‘0’ to UART data register until reach the number of transfer times as show in Figure XX and Figure XX.

Reception

This experiment shows when the UART receiver received data that store in to data register, the data transfers to memory that memory size is selected by different kind of data size such as byte, half-word and word.

MSIZE is byte and PSIZE is byte

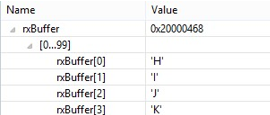
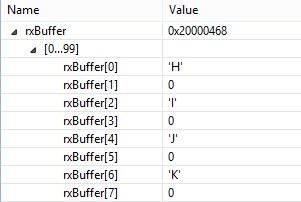
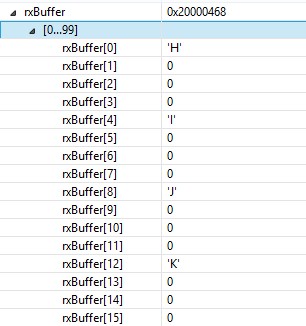


Figure XX shows the data that store into array from UART data register to

MSIZE is half-word and PSIZE is byte



MSIZE is word and PSIZE is byte



Discussion