

**BAME2123 Microcontroller Peripherals**

**Universal Asynchronous Receiver Transmitter**

**(UART)**

Course: Microelectronic with Embedded Technology (RMB2)

Name: LAI HWA NENG

ID: 15WAR10799

Tutor: Dr. POH TZE WEN

**Contents**

**CHARTER 1 : INTRODUCTION** **2**

**CHARTER 2 : THEORY** **2**

UART Data Formal Description3

UART Transmission3

UART Reception4

**CHARTER 3 : CONFIGURATION** **5**

GPIO Configuration For Peripherals5

UART Configuration 5

Baud Rate Configuration For The UART Transmitter and Receiver6

UART Interrupt Configuration7

**CHARTER 4 : RESULT AND DISCUSSION** **8**

The Baud Rate for UART transmitter8

The Word Length of UART Frame9

The Number of Stop Bits10

The Parity Control11

USART Frame Formats 11

Even Parity12

Odd Parity12

The Break Characters 13

**CHARTER 5 : THE UART WITH DMA PERIPHERAL**  **14**

DMA Configuration14

Result and Discussion16

The UART Transmission with DMA Peripheral16

The UART Reception with DMA Peripheral18

**Introduction**

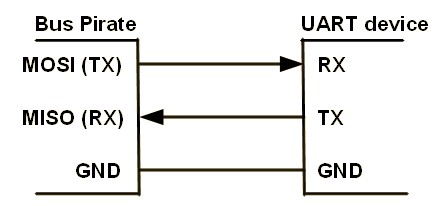
In the electronic world, a lot of things that can be controlled by electronic and the purpose is to make human have convenient and better live. If the thing was controlled by electronics, that must have a sense and controller to monitor the situation. Furthermore, in between the controlled and sense, another important thing that is communication interface. The communication interface is a hardware device that usually to assist hardware to communicate other hardware. For example, if the file or photo in smart phone that need send to the computer, the USB protocol is one of ways to assist smart phone send the file to the 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 the UART device. The UART was developed in STM32F429 board.

The UART is the term for “Universal Asynchronous Receiver Transmitter”. The “Universal” indicates that the 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.

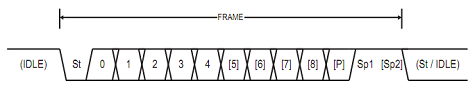
Besides that, the UART also includes a programmable baud generator that can be configured the speeds of transmission by using the equation that is given by data manual to calculate programmed of value.

**Theory**



Basically, the UART is bidirectional communication that requires a minimum of two pins such as transmission line (TX) and reception line (RX) as shown 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 occurring, the transmit signal is received from a slave that can not to read.

**UAST Data Formal Description**



In the UART protocol, the data formal is built in serial form and it is called frame as shown in figure XX. In the figure XX, The first bit at low-level state in frame is called it “start bit” (St) that is used for differentiate when is the receiver should start to collect the data.

After the start bit, the bits from 0th to 8th bit there was called it “data frame” in figure XX. Actually, the length of data can be selected by users. Following STM32F429 data sheet, the length of data was provided that only can be selected either 8 bits or 9 bits. After the end of data frame, the parity bit will be generate to let the receiver to checks the data that is correct or not. The parity bit can be enable or disable depend on the users. At the end, the last bits at high-level state in dame is called it “stop bit” (Sp) that is used for warn when is the receiver should stop to collect the data.

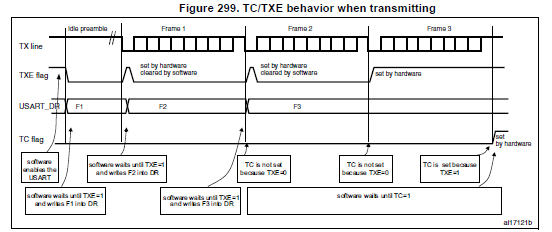
**Transmission**

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, the 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 the reception.

**Configuration**

**GPIO Configuration for Peripherals**



Figure XX shows the function that is configuration for GPIO when the GPIO is used for some of peripherals. Table XX shows the parameter in the “configureAlterFunctPin” function.

|  |  |
| --- | --- |
| Parameter | Explanation |
| pinNum | To configure the pin is used for peripherals |
| port | To configure the port is used for peripherals |
| ValueAFR | To configure the alternation function for port X and pin Y |



In this experiment, the UART5 is used to communicate with other peripherals. Following the STM32F429 data sheet, the transmitted pin of UART5 is the 12 pin of port C and the received pin of UART5 is the pin 2 of port D. Figure XX shows the example that configure the transmitted pin of UART5 and received pin of UART5.

**UART Configuration**

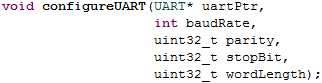


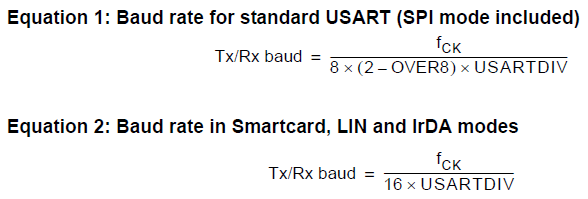
Figure XX shows the function that is used for UART configuration. This function provide users some of option about the frame and baud rate of transmit data and received data. Table XX shows the parameter in the “configureUART” function.

|  |  |
| --- | --- |
| Parameter | Explanation |
| uartPtr | To configure which UART is used to do transmission and reception |
| baudRate | To configure which the baud rate of transmit and received |
| Parity | To enable the parity control |
| stopBit | To configure the size of stop bit |
| wordLeng | To select the size of data frame |



Figure XX shows the function is used to configure the UART that include the baud rate is set to 9600 bit per second, the party control is disabled, the size of stop bit is one bit and the word length of the data frame is 8-bit.

**Baud Rate Configuration for the UART Receiver and Transmitter**



Following the STM32F429 data sheet, the equations about baud rate is provided as shown in Figure XX. In figure XX, the equation 1 is used for standard USART and the equation 2 is used in smartcard, LIN and IrDA modes. In this experiment, only the equation 1 is used for calculate the baud rate of UART. So, the OVER8 should be set in the UART\_CR1.



The baud rate for the UART receiver and transmitter are both set to the same value as programed in the Mantissa and Fraction values of USARTDIV. The values of USARTDIV is passed in the USART\_BRR register for baud rate configuration.



Figure XX shows the function that can calculate the values of USARTDIV. This function is more fixable at different PCLK. Table XX shows the parameter in the “baudRateAlgorithm” function.

|  |  |
| --- | --- |
| Parameter | Explanation |
| baudRate | baud rate that is set by users |
| fpclk | The Pclk that is get from HAL\_RCC\_GetPCLK1Freq() |
| Over8 | To select the equation 1 or equation 2 to calculate the USARTDIV |



Figure XX shows the function that calculate the values of USARTDIV and return to the USART\_BBR register for baud rate configuration.

**UART Interrupt configuration**

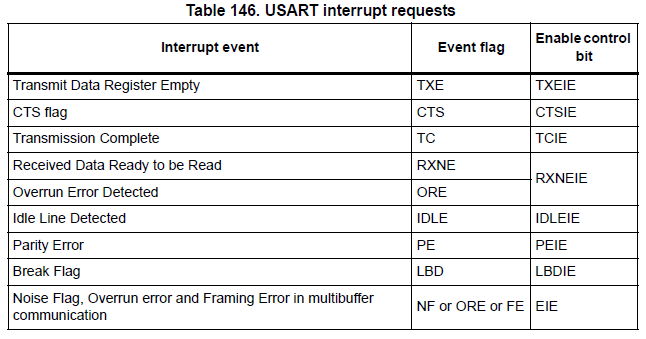


Figure XX shows the interrupt event there are used in UART transmission and reception. These interrupts can be differentiate into two of part: During transmission and While receiving.

During Transmission

|  |  |  |
| --- | --- | --- |
| Interrupt Event | Event Flag | Enable control bit |
| Transmission Complete | TC | TCIE |
| Clear to Send or Transmit Data Register empty interrupt | TXE | TXEIE |

While receiving

|  |  |  |
| --- | --- | --- |
| Interrupt Event | Event Flag | Enable control bit |
| Idle Line detection | IDLE | IDLEIE |
| Receive Data register not empty | RXNE | RXNEIE |
| Overrun error | ORE | EIE |
| Noise Flag | NF | EIE |
| Framing Error | FE | EIE |
| Parity error | PE | PEIE |
| LIN break detection | LBD | LBDIE |



Figure XX shows the function is used for UART Interrupt configuration. Table XX shows the parameter in the “configureUART\_IE” function.

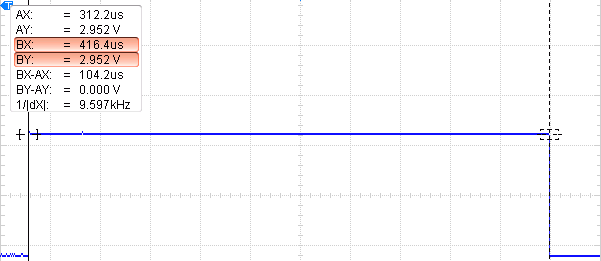
|  |  |
| --- | --- |
| Parameter | Explanation |
| uartPtr | To configure which UART is used for transmission and reception |
| txIE | To enable Transmit Data Register empty interrupt (TXEIE) |
| tcIE | To enable Transmission Complete interrupt (TCIE) |
| rxIE | To enable Receive data register not empty interrupt (RXNEIE) |
| parityIE | To enable Parity Error interrupt (PEIE) |
| ErrIE | To enable interrupt generation in case of a framing error, overrun error or noise flag |



Figure XX shows the function is used to configure all of UART interrupt.

**RESULT AND DISCUSSION**

**Buad rate for UART transmiter**



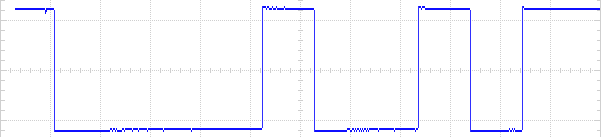
AY

BY

Figure XX shows the result about the baud rate is set by 9600 bit in transmission. In Figure XX, the difference between BY and AY is 104.2us that inverses to became 9.597 kHz. The 9597 of frequency is equal to 9.597k bit per second. The percentage discrepancy between the 9600 of baud rate and the 9597 of baud rate get from experiment is 0.0003%. This show the result is considered accurate.

**The Word Length of UART Frame**

The word length is 8 bit with no parity bit



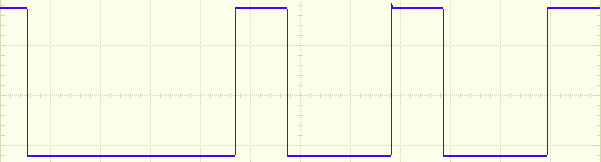
SB

STB

8-bit Data Frame

Figure XX show the result about the word length of UART frame is set by 8 bit with no parity bit.

The word length is 9 bit with no parity bit

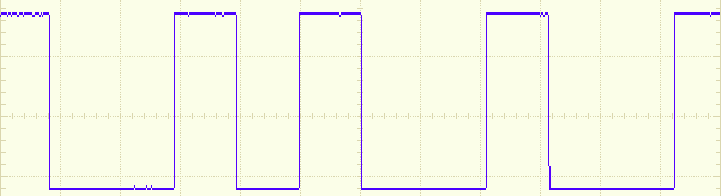


SB

STB

9-bit

8-bit Data Frame

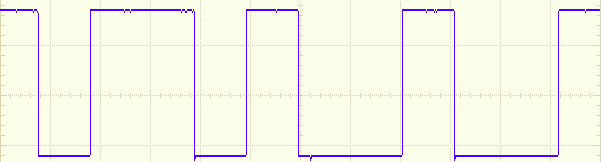


9-bit

8-bit Data Frame

SB

STB



9-bit

8-bit Data Frame

SB

STB

SB: Start bit, PB: Parity bit, STB: Stop bit.

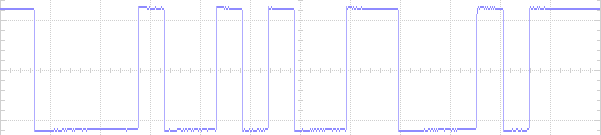
Figure XX, XX and XX show the results about the word length of UART frame is set by 9 bit with no parity. By looking at Figure XX, YY and ZZ, the length of data frame is 8-bit because the UART transmitter only can send the 8-bit size of data at once. So, the extra bit that is 9th bit is sent by 0 that depend on the design of microcontrollers.

**The Number of Stop bits**

The number of stop bits is 1 in 8-bit data frame

1-bit

1-bit



STB

STB

SB

SB

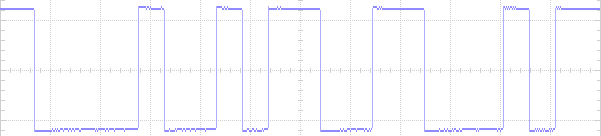
8-bit Data Frame

8-bit Data Frame

The number of stop bits is 2 in 8-bit frame

2-bit

2-bit



STB

STB

SB

SB

8-bit Data Frame

8-bit Data Frame

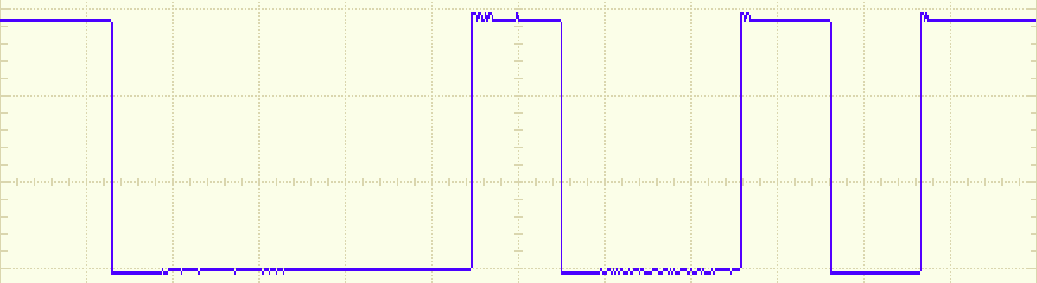
SB: Start bit, PB: Parity bit, STB: Stop bit.

By looking at Figure XX and Figure YY, the number of stop bit between these two figures are different. These want to show the number of stop bits to be transmitted with every character can be programmed in USART\_CR2. The following stop bits are supported by USART includes 0.5, 1, 1.5 and 2 stop bit. This experiment is only for the observation of the number of stop bit is 1 and 2. The 0.5 and 1.5 of stop bit were not show in the results as they were used for other function mode.

**Parity Control**

**USART Frame Formats**

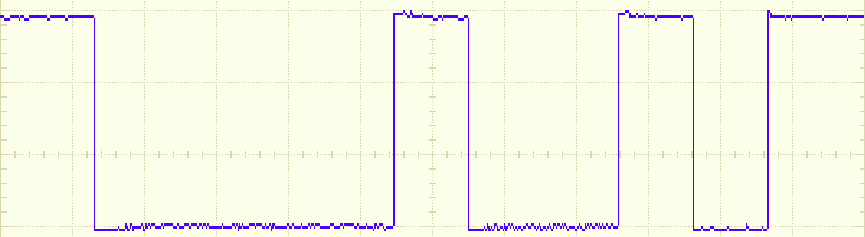
M bit is 0 and PCE bit is 0



SB

8-bit Data Frame

M bit is 0 and PCE bit is 1

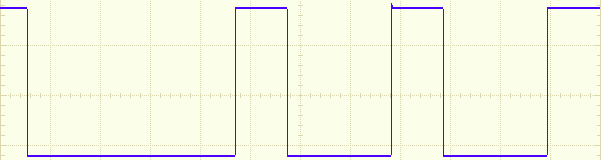


PB

SB

7-bit Data Frame

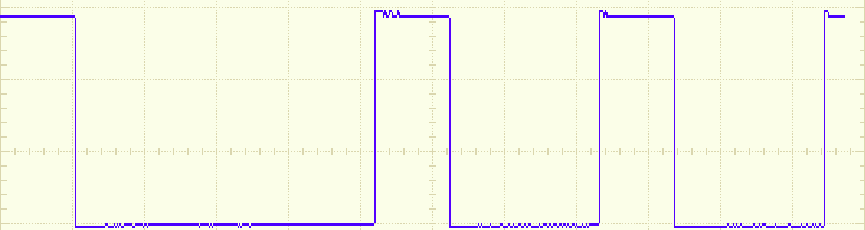
M bit is 1 and PCE bit is 0



SB

9-bit Data Frame

M bit is 1 and PCE bit is 1



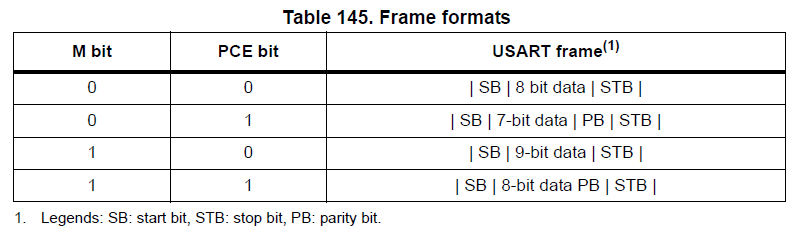
PB

SB

8-bit Data Frame

SB: Start bit, PB: Parity bit, STB: Stop bit.

Figure XX, XX, XX and XX show the results about possible UART frame formats are created by setting the PCE bit and M bit. The parity control can be enabled by setting the PCE bit and the frame length is defined by setting the M bit in USART\_CR1.

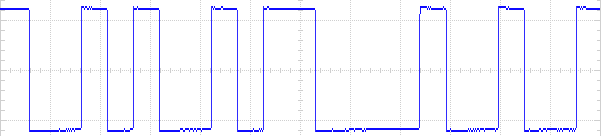


The results that were obtained through experiment is the same as the expected results as shown in Table 145.

Even Parity

Low-Level State

High-Level State



PB

PB

SB

STB

STB

SB

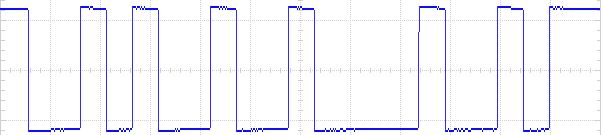
Data Frame

Data Frame

Odd Parity

High-Level State

Low-Level State



PB

PB

STB

SB

STB

SB

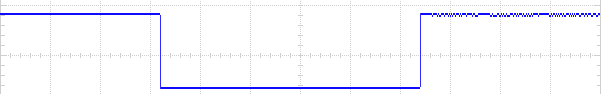
Data Frame

Data Frame

SB: Start bit, PB: Parity bit, STB: Stop bit.

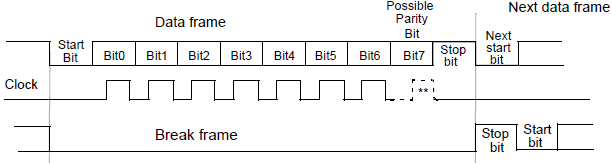
Figure XX and XX show the results about the parity bit was generated in even parity and odd parity. When the parity selection (PS) is set by even parity, the parity bit is calculated to obtain an even number of “1s” inside the frame made of the 7 or 8 LSB bits and the parity bit. When the parity selection (PS) is set by odd parity, the parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 7 or 8 LSB bit and the parity bit.

**Break Characters**



The length of Frame break

Figure XX shows the result about the break character is sent by the UART transmitter. A break transmission that sent a long length of low bits. A break character was transmitted by setting the SBK bit in USART\_CR1.

****

The length of frame break is depend on the word length. At the end of break frame the transmitter insert either 1 or 2 stop bits to acknowledge the start bit. In the figure XX, the word length is set by 8-bit and the transmitter sent the 10-bit of frame. When the SBK bit is sent by programmer, the 11-bit of frame break is sent by the transmitter to acknowledge the start bits.

**The UART peripheral with DMA peripheral**

This section will discuss about how the DMA is implemented in the UART peripheral. When the UART is implemented, the receiver and transmitter can be received and send the data to other devices. The CPU executes the instruction of UART in order or interrupt. If the CPU is required to handle a task while the execution of reception or transmission is occurred, there will cause the CPU reduce the processing power and lower down the efficiency of CPU. So, the purpose of DMA peripheral is used to improve the weakness for the CPU in that situation.

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. The data can be transferred by DMA and without effect the execution of CPU. The benefit is keeps the resources of CPU free for other operation.

In the STM32F429, that include the two of DMA and those contain the eight of stream and each stream includes the eight of 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.

**DMA Configuration**

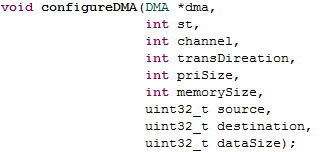


Figure XX shows the configuration function for DMA in C language. This function is more users friendly to do configuration for DMA. This is because whatever peripherals implement the DMA, the function can provides users to select DMA1 or DMA2. Table XX shows the parameter in the “configureDMA” function.

|  |  |
| --- | --- |
| 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 that can enable the 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.



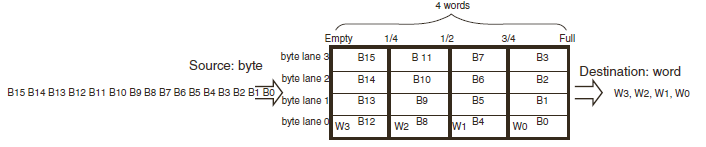
Figure XX shows the function that can enable the UART transmitter and receiver are serviced by DMA.



**The Result and Discussion**

**Transmission with DMA peripheral**

This experiment shows when the DMA is used to service the UAST transmitter, the peripheral data size is selected 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 memory data size is always fixed at byte (8-bit) because whatever size of memory that does not effected UART transmission.

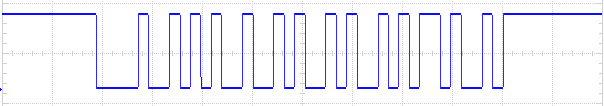


When the DMA service the UART transmission, the Data transfer direction (DIR) is set to Memory-to-Peripheral in DMA configuration. So that, the source is memory that assigns by the programmer and the destination is the UART data register in the DMA process.

In this case, the meaning of memory data size (MSIZE) is the size of the data store into the FIFO register from memory at once. The meaning of peripheral data size (PSIZE) is the size of the data pass out from the FIFO register to the data register of peripheral at once times.

Figure XX shows the concept about how the data is stored into FIFO register depend on the memory data size and how the data is pass out from FIFO register depend on the peripheral data size.

MSIZE is byte and PSIZE is byte



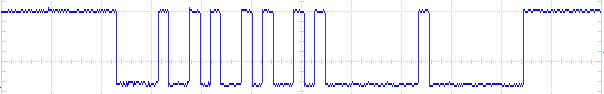
‘K’

‘J’

‘I’

‘H’

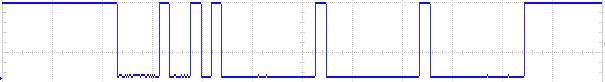
MSIZE is byte and PSIZE is half-word



‘H’

‘J’

MSIZE is byte and PSIZE is word



‘H’

Figure XX, XX, and XX show the results for the peripheral data size is selected by different kind of data size and the memory data size is always byte. The UART transmitter was requested by the DMA to send 4 of data character such as ‘H’, ‘I’, ‘J’ and ‘K’.

In Figure XX, the result shows the UART transmitter perfectly sent the 4 of data character and compare with the Figure XX and Figure XX that show the 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 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.

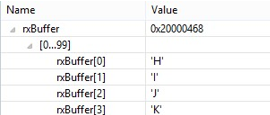
**Reception with DMA peripheral**

This experiment shows when the DMA is used to service the UART receiver, the memory size is selected 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 peripheral data size is always fixed at byte because whatever size of data that does not effected the UART reception.

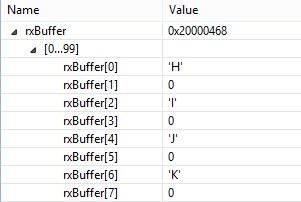
When the DMA service the UART reception, the Data transfer direction (DIR) is set to Peripheral -to-Memory in DMA configuration. So that, the source is the UART data register and the destination is memory that assigns by the programmer in the DMA process.

In this case, the meaning of the peripheral data size (PSIZE) is the size of data store into the FIFO register from the data register of peripheral at once. The meaning of memory data size (MSIZE) is the size of data pass out from the FIFO register to memory that was assigned by the programmers at once.

MSIZE is byte and PSIZE is byte



MSIZE is half-word and PSIZE is byte



MSIZE is word and PSIZE is byte

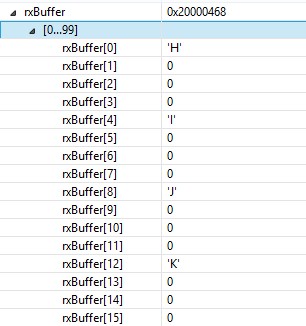


Figure XX, XX and XX show the results from the memory data size is selected by different kind of the data size and the peripheral data size is always byte. The UART receiver received the 4 of data character such as ‘H’, ‘I’ , ‘J’ and ‘K’ and the data was transferred by the DMA and store into the byte size array as called it “rxBuffer”.

In Figure XX, the data was sequentially arranged into the array and compare with the Figure XX and Figure XX the data was arranged into array with some of intervals. This is because the DMA reserve more than 8-bit size location for store the 8-bit data so that only first 8-bit location will fill the 8-bit data in and other location will be assign to 0. The reserve location that will depend on the MSIZE.

When the data is received and stored into a location of memory, the memory address pointer is automatically increased to point the next address of location for new data. The configuration can be set at the Memory Increment Mode (MINC) in DMA\_SxCR. The number of location address is increased at once time that depend on the memory data size (MSIZE) that is selected in the DMA configuration. For instance, the MSIZE is selected by half-word (16-bit) so the number of increase of address is 16 in every time. In Figure XX and Figure XX, the DMA reserves a location that the size of location will depend on the MSIZE.