

**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** **3**

**CHARTER 2 : THEORY** **3**

2.1 UART Fundamental Structure3

2.2 UART Data Formal Description4

2.3 UART Transmission4

2.4 UART Reception5

**CHARTER 3 : CONFIGURATION** **6**

3.1 GPIO Configuration For Peripherals6

3.2 UART Configuration 6

3.3 Baud Rate Configuration For The UART Transmitter and Receiver7

3.4 UART Interrupt Configuration8

3.5 DMA Configuration For UART10

**CHARTER 4 : INTERRUPT** **12**

4.1 Interrupt For Transmission12

4.2 Interrupt For Reception12

4.3 Interrupt For DMA13

**CHARTER 5 : RESULT AND DISCUSSION** **14**

5.1 The Baud Rate for UART transmitter14

5.2 The Word Length of UART Frame15

5.3 The Number of Stop Bits16

5.4 The Parity Control17

5.4.1 USART Frame Formats 17

5.4.2 Even Parity19

5.4.3 Odd Parity19

5.5 The Break Characters 20

5.6 The UART Communicate with PL-2303HX21

5.7 The Transmission with DMA Peripheral22

5.8 The Reception with DMA Peripheral23

**CHARTER 6 : REFERENCE**2**4**

**CHARTER 7 : APPENDIX**2**4**

7.1 Main Program26

7.2 UART.c File27

7.3 DMA.c File29

7.4 GPIO.c File30

**CHARTER 1: 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 by 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.

**CHARTER 2: THEORY**

**2.1 UAST fundamental structure**

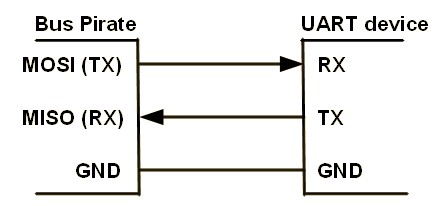


Figure 2.0 – UART Fundamental Block Diagram

Basically, the UART is bidirectional communication that requires a minimum of two pins such as transmitted line (TX) and received line (RX) as shown in Figure 2.0. 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, the communication between master and slave cannot communicate anymore.

**2.2 UAST Data Formal Description**

In the UART protocol, the data formal is built in serial form and it is called by “frame” as shown in Figure 2.1.

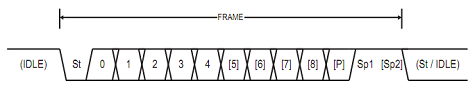


Figure 2.1 – UART Frame Formal

In the figure 2.1, the first bit of frame at low state is called “start bit” (St) that is used to indicate the actual timing to read the data for the receiver.

After the start bit, the bits from 0th to 8th bit those was called by “data frame”. Actually, the length of data can be selected by users. Referring STM32F429 data sheet, the length of data only can be selected either 8 bits or 9 bits.

After the end of data frame, the parity bit will be generated for the receiver to check the data that is correct or not. The parity bit can be enabled or disabled depend on the users. At the end, the last bit of frame at high state is called “stop bit” (Sp) that is used to indicate the data has finished sending to receiver.

**2.3 UART Transmission**

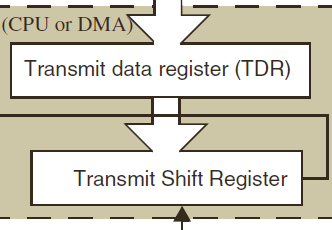


Figure 2.2 – Data Register for UART Transmitter

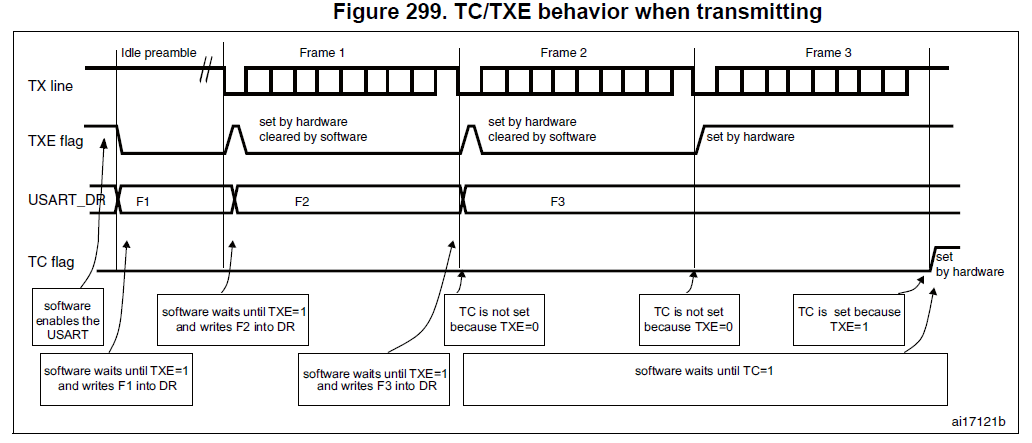
In the section of UART transmitter, that involves the Transmit Data Register (TDR) and the Transmit Shift Register as shown in Figure 2.2. When a transmission is occurring, a write instruction to the UAST Data Register (USART\_DR) stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

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

If a frame is transmitted and the TXE bit is set, the TC bit is set 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 to become high before disabling the USART or causing the microcontroller to enter the low-power mode.

Figure 299 shows the UART transmitter behavior when transmitting.



**2.4 UART Reception**

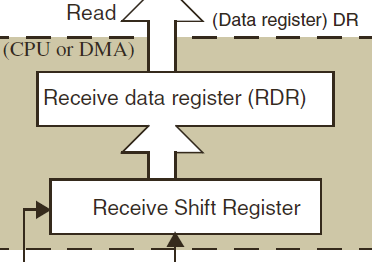


Figure 2.3 – Data Register for UART Receiver

In the section of UART receiver, that involves the Receive Data Register (RDR) and the Receive Shift Register as shown in Figure 2.3. When a character is received, the RXNE flag is set. It indicates that the data of the shift register is transferred to the RDR. In other words, the data has been received and can be read.

**CHARTER 3: CONFIGURATION**

**3.1 GPIO Configuration for Peripherals**



Figure 3.1.0 – GPIO Configuration Function

Figure 3.1.0 shows the function that is used for GPIO configuration when the GPIO is applied by peripherals.

Table 3.1.0 – Parameter of “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 |



Figure 3.1.1 – Applied “configureAlterFunctPin” Function.

Figure 3.1.1 shows the function that is configured the transmitted pin of UART5 and received pin of UART5 in real. In this experiment, the UART5 is used to communicate with other peripherals. Referring 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.

**3.2 UART Configuration**

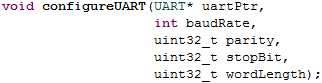


Figure 3.2.0 – UART Configuration Function

Figure 3.2.0 shows the function that is used for UART configuration. This function provides user option to select the frame format and baud rate of transmitted and received.

Table 3.2.0 – Parameter of “configureUART” Function

|  |  |
| --- | --- |
| Parameter | Explanation |
| uartPtr | To configure which UART is used to do transmission and reception |
| baudRate | To configure which the baud rate of transmited 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 3.2.1 – Applied “configureUART” Function.

Figure 3.2.1 shows the function that is applied to configure the UART in real. The configuration includes 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.

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

The meaning of baud rate is how fast data sent over a serial line. The unit of baud rate is expressed in bit-per-second (bps). Nowadays, the standard of baud rate for communication interface such as 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200.

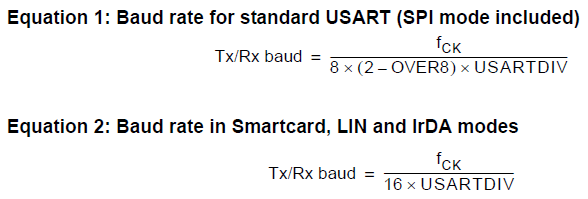


Figure 3.3.0 – UART Baud Rate Equations

Referring the STM32F429 data sheet, the baud rate equations are provided as shown in Figure 3.3.0. In figure 3.3.0, the Equation 1 is used for standard USART and the Equation 2 is used in smartcard, LIN and IrDA modes. In this experiment, the Equation 1 is used for calculating the values of baud rate and the OVER8 should be set in the UART\_CR1.



Figure 3.3.1 – UART Baud Rate Register

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 are passed in the USART\_BRR register for baud rate configuration.



Figure 3.3.1 – UART Baud Rate Configuration Function

Figure 3.3.1 shows the function that can calculate the values of USARTDIV. This function is more convenient for configuring UART baud rate. This is because any values of PCLK are provided by PLL and the values of USARTDIV still can be calculated without referring the STM32F420 data sheet. Table XX shows the parameter in the “baudRateAlgorithm” function.

Table 3.3.0 – Parameter of “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 3.3.2 – Applied “baudRateAlgorithm” function

Figure 3.3.2 shows the function that calculates the values of USARTDIV and return to the USART\_BBR register for baud rate configuration.

**3.4 UART Interrupt configuration**

Interrupts are events that require attention by the processor. When the interrupt event occurs the processor stop its current task and attend to the interrupt by executing an Interrupt Service Routine (ISR) at the end of the ISR the microcontroller returns to the task it had stopped and continue its normal operation.

In UART peripheral, the interrupt event are used for the UART transmission and reception. The UART interrupt event can be differentiated into two sections. The first section is “During Transmission” that is mean when the transmitter is enabled by doing transmission. Table 3.4.0 shows the interrupt event about the transmitter during transmission.

The second section is “While receiving” that is mean when the receiver is enabled by doing reception. Table 3.4.1 shows the interrupt event about the receiver while receiving.

During Transmission

Table 3.4.0 – Interrupt Event for 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

Table 3.4.1 – Interrupt Event for 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 |

Interrupt Service Routine (ISR) is a software routine that hardware invokes in response to an interrupt. In the STM32F429, the ISR is called IRQ that stand for Interrupt Request.

For this experiment, the UART5 is operated for doing communication so the name of the IRQ handler function is “UART5\_IRQHandler”. The name of the IRQ handler function can be searched in the “startup\_stm32f429xx. s” file. Before operating the IRQ handler function, the IRQ handler must be enabled by calling the “HAL\_NIVC\_EnableIRQ (XXX\_IRQn)” in the main program as shown in the appendix. “XXX\_IRQn” is an identifier that can be searched “stm32f429xx. h” file.



Figure 3.4.0 – UART Interrupt Configuration Function

Figure 3.4.0 shows the function is applied to configure UART Interrupt.

Table 3.4.2 – Parameter of “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 (EIE) |

**3.5 The DMA Configuration for UART**

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 occurring, 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 provides high-speed data transfer between peripherals and memory and between memory and memory. The data can be transferred by DMA and without affect 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.

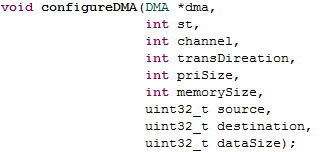


Figure 3.5.0 – DMA Configuration Function

Figure 3.5.0 shows the function is used for DMA configuration. This function is more user-friendly and convenient because it can configure the DMA that is used for any peripherals in different stream and channel.

Table 3.5.0 – Parameter of “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 3.5.1 – Applied “configureDMA” function

Figure 3.5.1 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 3.5.2 –DMA Enable Transmitter Function and DMA Enable Receiver Function

Figure 3.5.2 shows the function that can enable the DMA to service the UASR transmitter and receiver. Notice, when DMA is doing configuration, the enable bit must be set to 0 first.



Figure 3.5.3 – DMA Enable Function

Figure 3.5.3 shows the function that can enable the UART transmitter and receiver are serviced by DMA.

**CHARTER 4: INTERRUPT**

**4.1 Interrupt for UART Transmission**

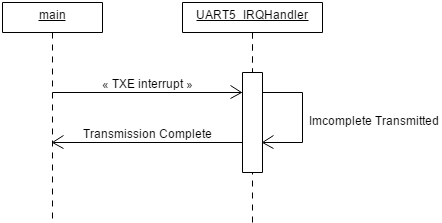


Figure 4.1 – Sequence Diagram for UART Interrupt Transmission

Figure 4.1 shows the sequence diagram about the TXE interrupt is operated in the main program. The TXE interrupt can be referred in Table 3.4.0.

When the TXE interrupt is set by hardware, the processor pause its current task in the main program and attend to UART5\_IRQHandler function by executing the transmission. The processor will continuing transmitted data until all of data is sent.

When the transmission is complete, the processor returns the task it had paused and continues its operation. The program in UART5\_IRQHandler as shown in the appendix.

**4.2 Interrupt for UART Reception**

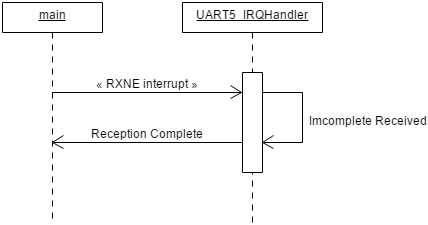


Figure 4.2 – Sequence Diagram for UART Interrupt Receiver

Figure 4.2 shows the sequence diagram about the RXNE interrupt is operated in the main program. The TXE interrupt can be referred in Table 3.4.1.

When the RXNE interrupt is set by hardware, the processor pause its current task in the main program and attend to “UART5\_IRQHandler” function by executing the reception.The procosser will continue the received data until no more data is received in UART receiver.

When the reception is complete, the processor returns the task it had paused and continues its operation. The program in UART5\_IRQHandler as shown in the appendix.

**4.3 Interrupt for DMA**

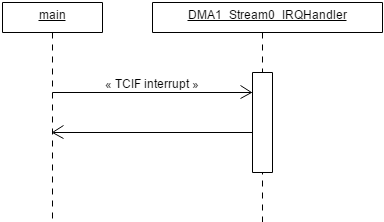


Figure 4.3 – Sequence Diagram for DMA Interrupt

Figure 4.3 shows the sequence diagram about the TCIF interrupt is operated in the main program. The TCIF is stand for “Transfer Complete Interrupt Flag”. It is used for getting the attention of processor when the DMA done transfered.

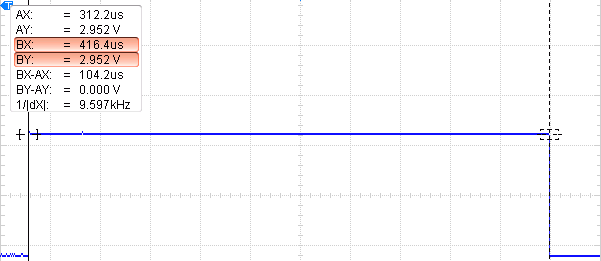
When TCIF interrupt is set by hardware, the processor pause its current task in the main program and attend to “DMA1\_StreamX\_IRQHandle” function by executing the task that is required by the programmer. When all of task is done in the “DMA1\_StreamX\_IRQHandle” function , the processor return the task it had paused and continues its operation.

The name of DMA IRQ handler function can be searched in the “startup\_stm32f429xx.s” file.

**CHARTER 5: RESULT AND DISCUSSION**

**5.1 Baud rate for UART transmiter**

9600 bit per second



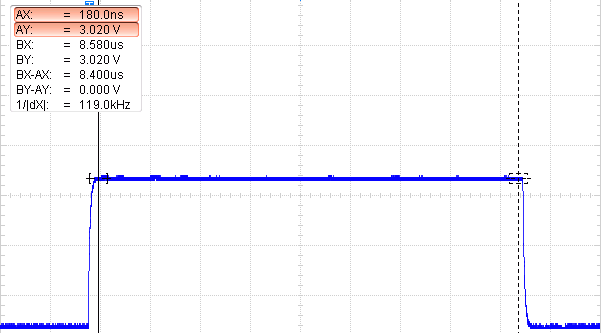
BY

AY

Figure 5.1.0 – Baud Rate is set at 9600 bit per second

Figure 5.1.0 shows the result about the baud rate is set by 9600 bit per second for transmission. In Figure 5.1.0, the difference between BY and AY is 104.2us that inverses to became 9.597 kHz. The meaning of 9.597 kHz is the data is transmitted in 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%.

11520 bit per second



BY

AY

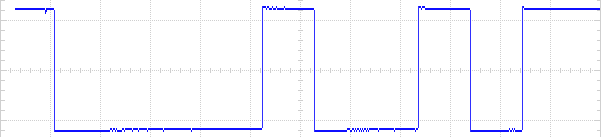
Figure 5.1.1 – Baud Rate is set at 11520 bit per second

Figure 5.1.1 shows the result about the baud rate is set by 11520 bit per second for reception. In Figure 5.1.1, the difference between BY and AY is 8.4us that inverses to became 119.0 kHz. The meaning of 119.0 kHz is the data is transmitted in 119.0k bit per second. The percentage discrepancy between the 11520 of baud rate and the 11900 of baud rate get from experiment is 0.0324%.

**5.2 The Word Length of UART Frame**

In the UART peripheral, the term word-length is used to describe the length of data frame. The length of data frame is set by programmers.

The word length is 8 bit with no parity bit



SB

STB

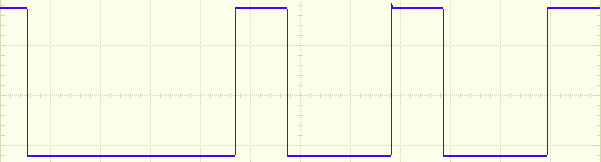
8-bit Data Frame

Figure 5.2.0 – The ‘H’ character signal is sent in 8-bit of data frame

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

Figure 5.2.0 shows 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



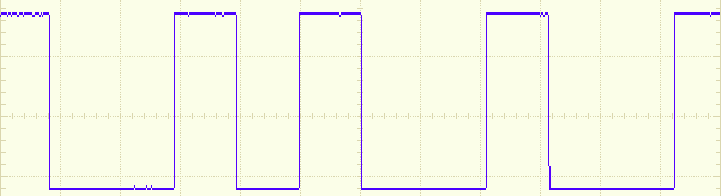
SB

STB

9-bit

8-bit Data Frame

Figure 5.2.1 – The ‘H’ character signal is sent in 9-bit of data frame



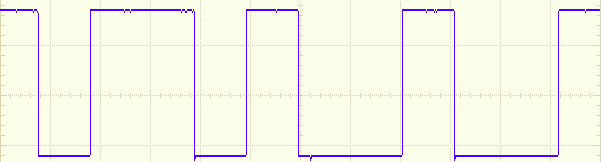
9-bit

8-bit Data Frame

SB

STB

Figure 5.2.2 – The ‘J’ character signal is sent in 9-bit of data frame



9-bit

8-bit Data Frame

SB

STB

Figure 5.2.3 – The ‘K’ character signal is sent in 9-bit of data frame

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

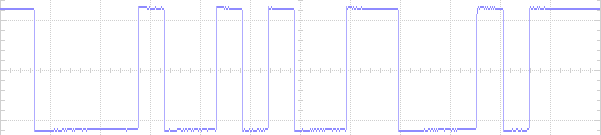
Figure 5.2.0, 5.2.1 and 5.2.2 show the results about the word length of UART frame is set by 9 bit with no parity. By looking at Figure 5.2.0, 5.2.1 and 5.2.2, the length of data frame is 8-bit because the UART transmitter only can send the 8-bit size of data at once. So, the extra bit that is 9th bit is sent by 0 that depend on the design of microcontrollers.

**5.3 The Number of Stop bits**

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

1-bit

1-bit



STB

STB

SB

SB

8-bit Data Frame

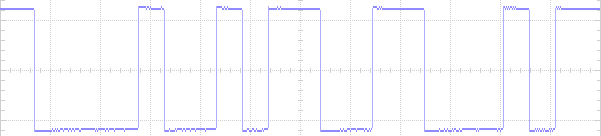
8-bit Data Frame

Figure 5.3.0 – The stop bit is selected in 1-bit size.

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

2-bit

2-bit



STB

STB

SB

SB

8-bit Data Frame

8-bit Data Frame

Figure 5.3.1 – The stop bit is selected in 2-bit size.

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

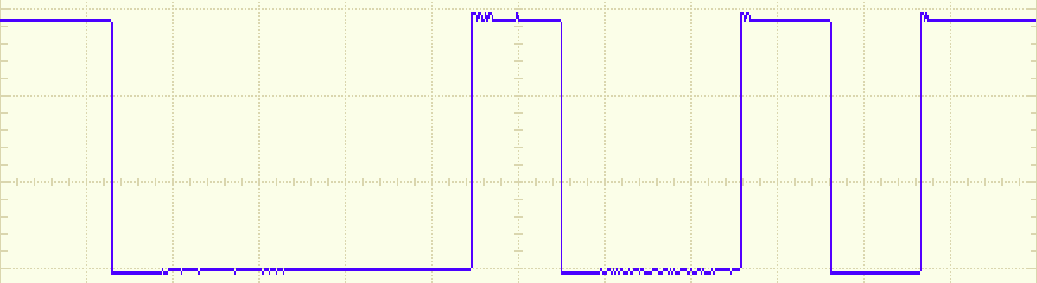
By looking at Figure 5.3.0 and 5.3.1, the difference between Figure 5.3.0 and 5.3.1 is their number of stop bit. The stop bit is transmitted with every end of character in programmed in UART configuration function as referred in section 3.4.

Referring the STM32F429 data sheet, the number of stop bit is provided such as 1-bit, 2-bit, 0.5-bit and 1.5-bit. In this experiment, only the 1-bit and 2-bit of stop bit are programmed for doing the observation. The 0.5 and 1.5 of stop bit were not show in the results as they were used for other function mode.

**5.4 Parity Control**

**5.4.1 USART Frame Formats**

M bit is 0 and PCE bit is 0

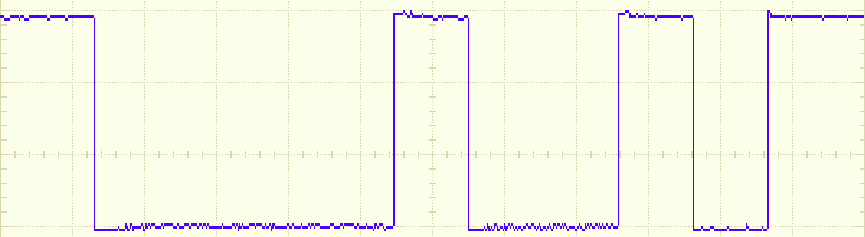


SB

8-bit Data Frame

Figure 5.4.1a – The word length is set at 8-bit and the parity control is disabled.

M bit is 0 and PCE bit is 1



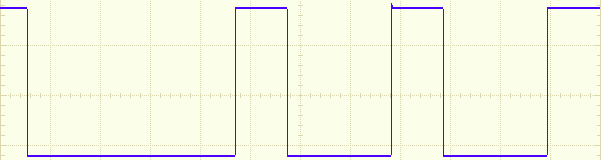
PB

SB

7-bit Data Frame

Figure 5.4.1b – The word length is set at 8-bit and the parity control is enabled.

M bit is 1 and PCE bit is 0

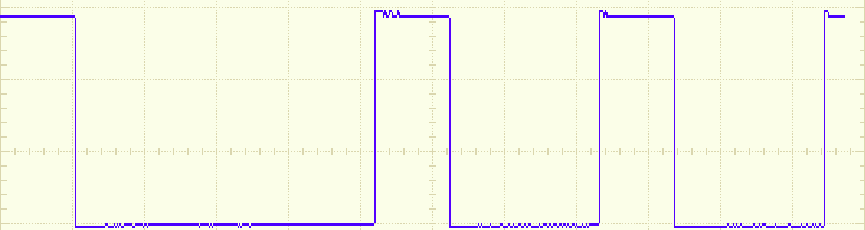


SB

9-bit Data Frame

Figure 5.4.2c – The word length is set at 9-bit and the parity control is disabled.

M bit is 1 and PCE bit is 1



PB

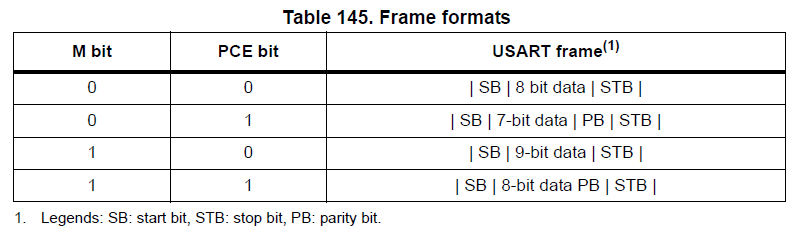
SB

8-bit Data Frame

Figure 5.4.2d – The word length is set at 9-bit and the parity control is enabled.

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

Figure 5.4.2a, 5.4.2b, 5.4.2c and 5.4.2d show the results about possible UART frame formats are configured by setting the PCE bit and M bit. The parity control and word length can be configured by programming the UART Configuration Function as refer in section 3.2.

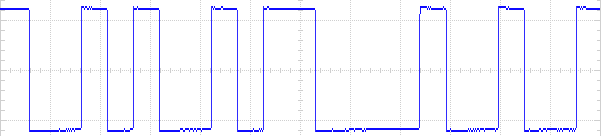


In conclusion, the results that were obtained through experiment is the same as the expected results as shown in Table 145.

**5.4.2 Even Parity**

Low-Level State

High-Level State



PB

PB

SB

STB

STB

SB

Data Frame

Data Frame

Figure 5.4.2a – Parity Control work in Even Parity mode

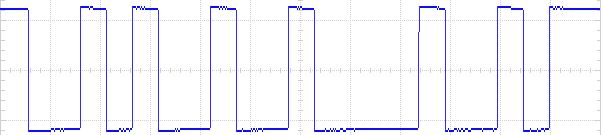
SB: Start bit, PB: Parity bit, STB: Stop bit.

Figure 5.4.2a show the results about the parity bit was generated in even parity mode. 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.

**5.4.3 Odd Parity**

High-Level State

Low-Level State



PB

PB

STB

SB

STB

SB

Data Frame

Data Frame

Figure 5.4.3a – Parity Control work in Odd Parity mode

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

Figure XX show the results about the parity bit was generated in odd parity. 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.

**5.5 Break Characters**

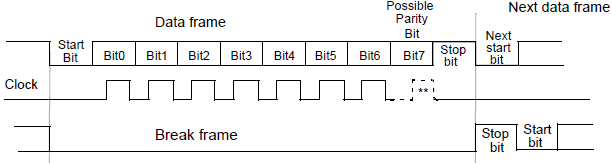
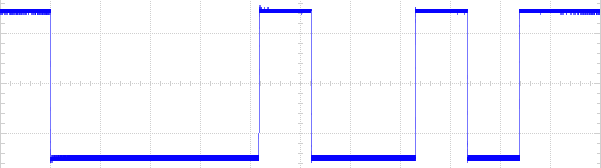
****

Figure 5.5.0 – Break Character

Break character basically is a length of low bits. It is used to inform the peripherals which communicate with the master for stopping the communication.

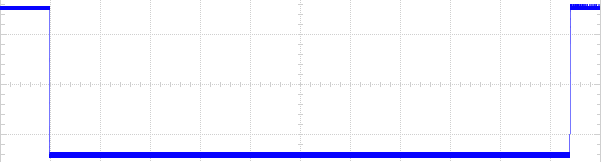
Figure 5.5.0 shows the concept diagram is obtained from the STM32F249 data sheet, It presents the length of break frame is set depending on the length of data frame. In the figure 5.5.0, the 10-bit of frame is transmitted by UART transmitter. If the SBK bit is set, the 11-bit of break frame is transmitted. At the end of break frame the transmitter insert 1 stop bits to acknowledge the start bit.



10-bit of frame

Figure 5.5.1 – The ‘H’ character signal is sent by transmitter in 10-bit frame

Figure 5.5.1 shows the result about the ‘H’ character is sent by the UART transmitter. The length of frame is 10-bit with no parity bit and the one bit size of stop bit.



10-bit of break frame

Figure 5.5.2 – The break character is sent by transmitter in 10-bit of break frame

Figure 5.5.2 shows the result about the 10-bit of break character is sent by the UART transmitter. A break character was transmitted by setting the SBK bit in USART\_CR1.

In conclusion, the results that were obtained through experiment is the same as the expected results as shown in Figure 5.5.0.

**5.6 The UART communicate with** **PL-2303HX**

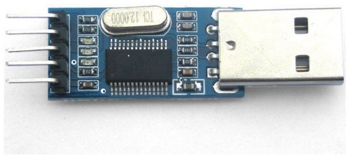
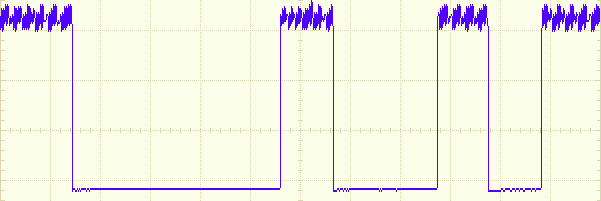


Figure 5.6.0 – PL-2303HX USB-to-Serial Bridge Controller

The PL-2303HX is cheaper and good performance USB-to-Serial Bridge Controller as shown in Figure 5.6.0. The PL-2303HX provides a convenient solution for connecting an RS-232 full-duplex asynchronous serial device to any Universal Serial Bus (USB) capable host.

In this experiment, the PL-2303 is connected by using USB to the computer for programing. The received pin (RX) of PL-2303HX is connected the pin 2 of port D that is in STM32F429. The transmitted pin (TX) of PL-2303HX is connected the pin 12 of port C that is in STM32F429. Notice, both the baud rate of device must be set at the same value. Otherwise, the communication between master and slave cannot communicate anymore.

PL-2303HX transmission



8-bit data frame

SB

Figure 5.6.1 – The ‘H’ character signal is sent by PL-2303HX transmitter

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

Figure 5.6.1 shows the character ‘H’ signal that is the transmitted by PL-2303 to the UART5 of STM32F429. This result is captured for showing the signal is received by the UART5 receiver. In the Figure X, the signal transmit the high level state ‘1’ that will generate the violation. Its violation may be due to the internal circuit problem.

PL-2303HX reception



Figure 5.6.2 – Transmitted String Function

Figure 5.6.2 shows the function is used for transmitting a string from UART5 transmitter to the receiver of PL-2303.

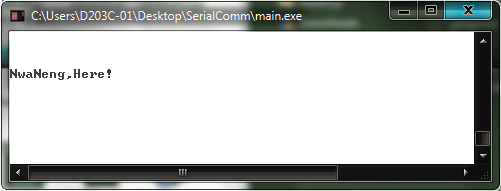


Figure 5.6.3 – Display Window

Figure X shows the receiver of PL-2303 received the string is sent by the UART5 of STM32F429 as display in the window of computer.

**5.7 The 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.

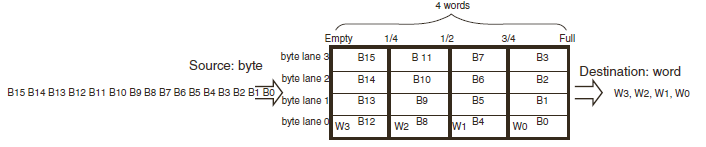


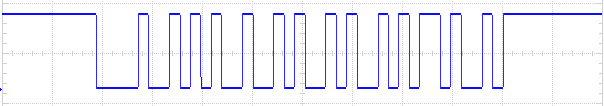
Figure 5.7.0 – FIFO structure

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 5.7.0 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’

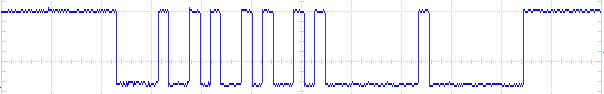
‘J’

‘I’

‘H’

Figure 5.7.1 – UART Transmission using DMA when PSIZE is byte

MSIZE is byte and PSIZE is half-word

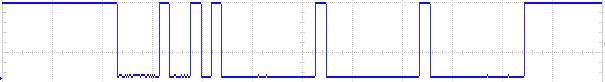


‘H’

‘J’

Figure 5.7.2 – UART Transmission using DMA when PSIZE is half-word

MSIZE is byte and PSIZE is word



‘H’

Figure 5.7.3 – UART Transmission using DMA when PSIZE is word

Figure 5.7.1, 5.7.2, and 5.7.3 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 characters such as ‘H’, ‘I’, ‘J’ and ‘K’.

In Figure 5.7.1, the result shows the UART transmitter perfectly sent the 4 of data characters and compare with the Figure 5.7.2 and Figure 5.7.3 that show the UART transmitter did not send the 4 of data characters completely.

The reason is the UART data register that only can store one byte at once. If the FIFO register transfer data that is more than one byte at once, 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 5.7.2 and 5.7.3.

**5.8 The 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 affect 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 a 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

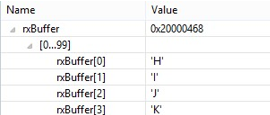


Figure 5.8.0 – UART reception using DMA when MSIZE is byte

MSIZE is half-word and PSIZE is byte

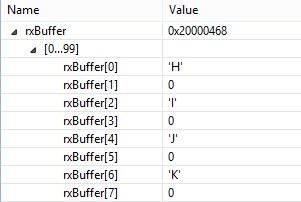


Figure 5.8.1 – UART reception using DMA when MSIZE is half-word

MSIZE is word and PSIZE is byte

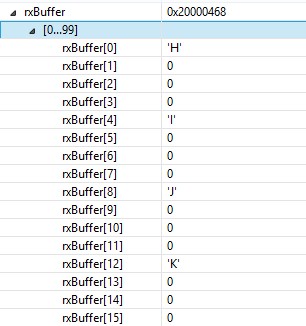


Figure 5.8.2 – UART reception using DMA when MSIZE is word

Figure 5.8.0, 5.8.1 and 5.8.2 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 5.8.0, the data was sequentially arranged into the array and compare with the Figure 5.8.1 and Figure 5.8.2 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 5.8.1 and 5.8.0, the DMA reserves a location that the size of location will depend on the MSIZE.

**CHARTER 6 : REFERENCE**

1. Dangerous Prototypes (2015) *Bus Pirate UART guide* [Online] Available

<http://dangerousprototypes.com/bus-pirate-manual/bus-pirate-uart-guide/> [Accessed: 25th December 2015].

1. AliExpress (2015) *PL2303(PL-2303HX) USB to RS232 UART Board* [Online] Available

<http://www.aliexpress.com/item/5pcs-lot-New-Waveshare-PL2303-PL-2303HX-USB-to-RS232-UART-Board-type-A-USB-to/1924133906.html> [Accessed: 25th December 2015]

1. ST life augmented (2015) *RM0090 Reference Manual* [e-book]

<http://www.st.com/web/en/catalog/mmc/FM141/SC1169/SS1577/LN1806?ecmp=stm32f429-439_pron_pr-ces2014_nov2013&sc=stm32f429-439-pr> [Accessed: 25th December 2015]

**CHARTER 7 : APPENDIX**

7.1 Main program

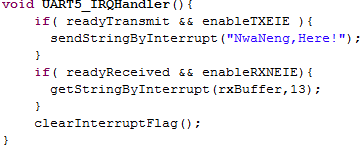


Figure 7.1.0 – UART5 IRQ Handler function



Figure 7.1.1 – DMA1 IRQ Handler function for UART Transmitter



Figure 7.1.2 – DMA1 IRQ handler function for UART Receiver

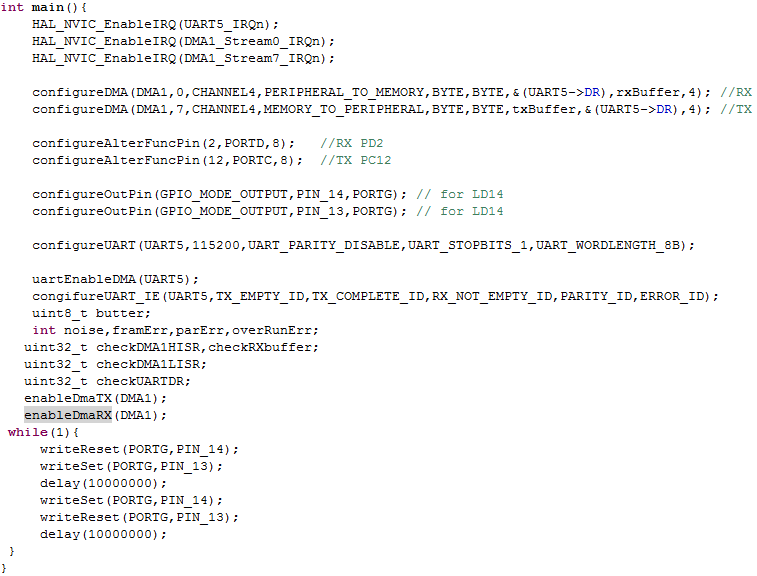


Figure 7.1.3 – Main Program

7.2 UART.c file

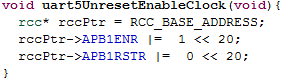


Figure 7.2.0 –Does not Reset and enable Clock for UART5

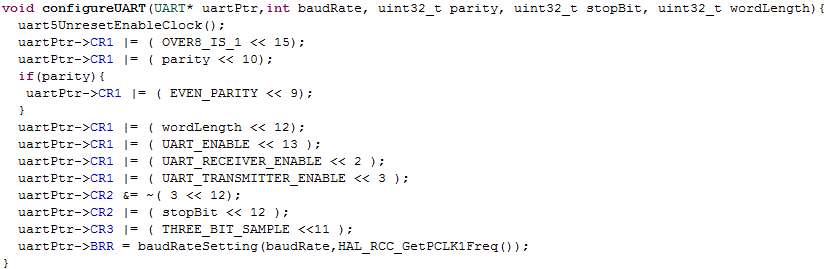


Figure 7.2.1 – UART Configuration Function

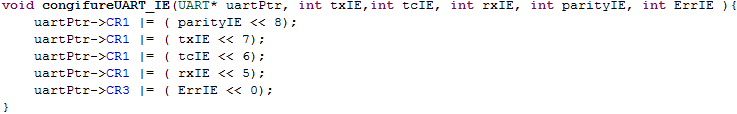


Figure 7.2.2 – UART Interrupt Configuration Function



Figure 7.2.3 – Transmitted Data function

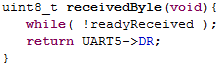


Figure 7.2.4 – Received Data Function



Figure 7.2.5 – Send Break Character Function

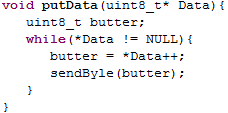
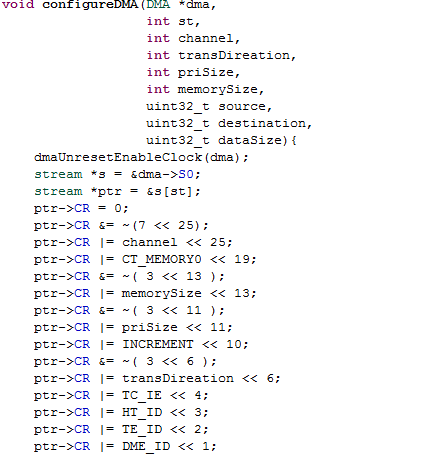


Figure 7.2.6 – Transmitted String function

7.3 DMA.c file



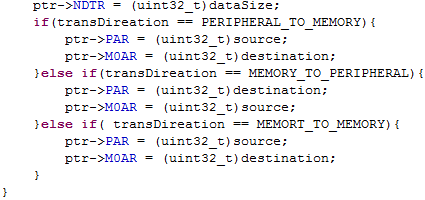


Figure 7.3.0 – DMA configuration Function

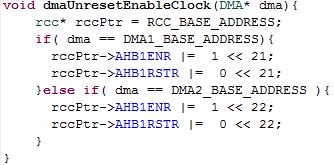


Figure 7.3.1 – Does not Reset and Enable Clock for DMA



Figure 7.3.2 – DMA Enable Transmitter Function



Figure 7.3.3 –DMA Enable Receiver Function

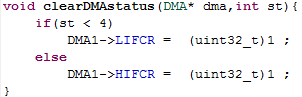


Figure 7.3.4 – Clear DMA Status Register Function

7.4 GPIO.c file

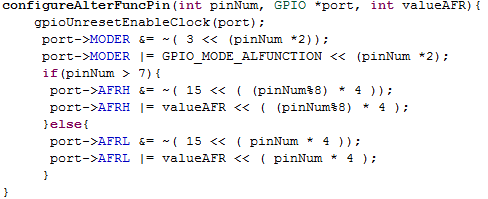


Figure 7.4.0 – GPIO Configuration for peripherals