UART Interrupt

Example of program

```
u8 TxBuffer[100]  = “\rUSART1 Transmission using interrupt \r\n”;
u8 RxMessage[2] = [0];
u8 RxUpdate=0;
 u8 TxReady = 1;
int main(){
  RCC_setup();
  GPIO_setup();
  NVIC_setup();
  USART1_setup();
  usart1_puts(“USART1 Test \r\n”);
 while(1){
     if(RxUpdate==1){
        RxUpdate = 0;
        usart1_puts(“You press: “);
        usart1_putc(RxMessage);
        usart1_puts(“\r\n”);
     }
  }
}
```

```
void GPIO_setup(){
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC |
  RCC_APB2Periph_AFIO,ENABLE);  // Enable GPIOA, GPIOC, and AFIO channel
  // Configure USART1 Tx (PA9) as alternate function push-pull
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // Configure USART1 RX (PA10 ) as Input floating
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

```
void NVIC_setup(){
  NVIC_InitTypeDef NVIC_InitStructure;
  // Enable the USART1 interrupt
  NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQChannel;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
void USART1_setup(){
  USART_InitTypeDef USART_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE); //Enable USART clock
  USART_InitStructure.USART_BaudRate = 115200;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_Stopbits = USART_StopBits_1;
  USART_InitStructure.USART_Priority = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  USART_Init(USART1, &USART_InitStructure);
  USART_ITConfig(USART1, USART_IT_TXE, DISABLE);
  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
  USART_Cmd(USART1,ENABLE);
}
```

```
void usart1_puts(unsigned char *s){
  while(TxReady==0);
  TxCounter=0;
  strcpy(TxBuffer,s);
  USART_ITConfig(USART1,USART_IT_TXE,ENABLE);
  TxReady = 0;
}
void usart1_putc(unsigned char c){
  while(TxReady==0);
  TxCounter=0;
  TxBuffer[0] = c;
  TxBuffer[1] = '\0';
  USART_ITConfig(USART1,USART_IT_TXE,ENABLE);
  TxReady = 0;
}
```

```
void USART1_IRQHandler(void){
  if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET){ // check RX interrupt
      RxMessage = USART_ReceiveData(USART1);
      RxUpdate = 1;
      USART_ClearITPendingBit(USART1,USART_IT_RXNE);
  }
 if(USART_GetITStatus(USART1,USART_IT_TXE)!=RESET){ // check TX interrupt
      USART_SendData(USART1TxBuffer[TxCounter++]);
      USART_ClearITPendingBit(USART1 USART_IT_TXE);
      if(TxBuffer[TxCounter] == 0 | TxCounter > TxBufferSize){
        USART_ITConfig(USART1,USART_IT_TXE,DISABLE); // disable transmit interrupt
         TxReady = 1;
      }
  }
}
```

UART DMA

Directed Memory Access

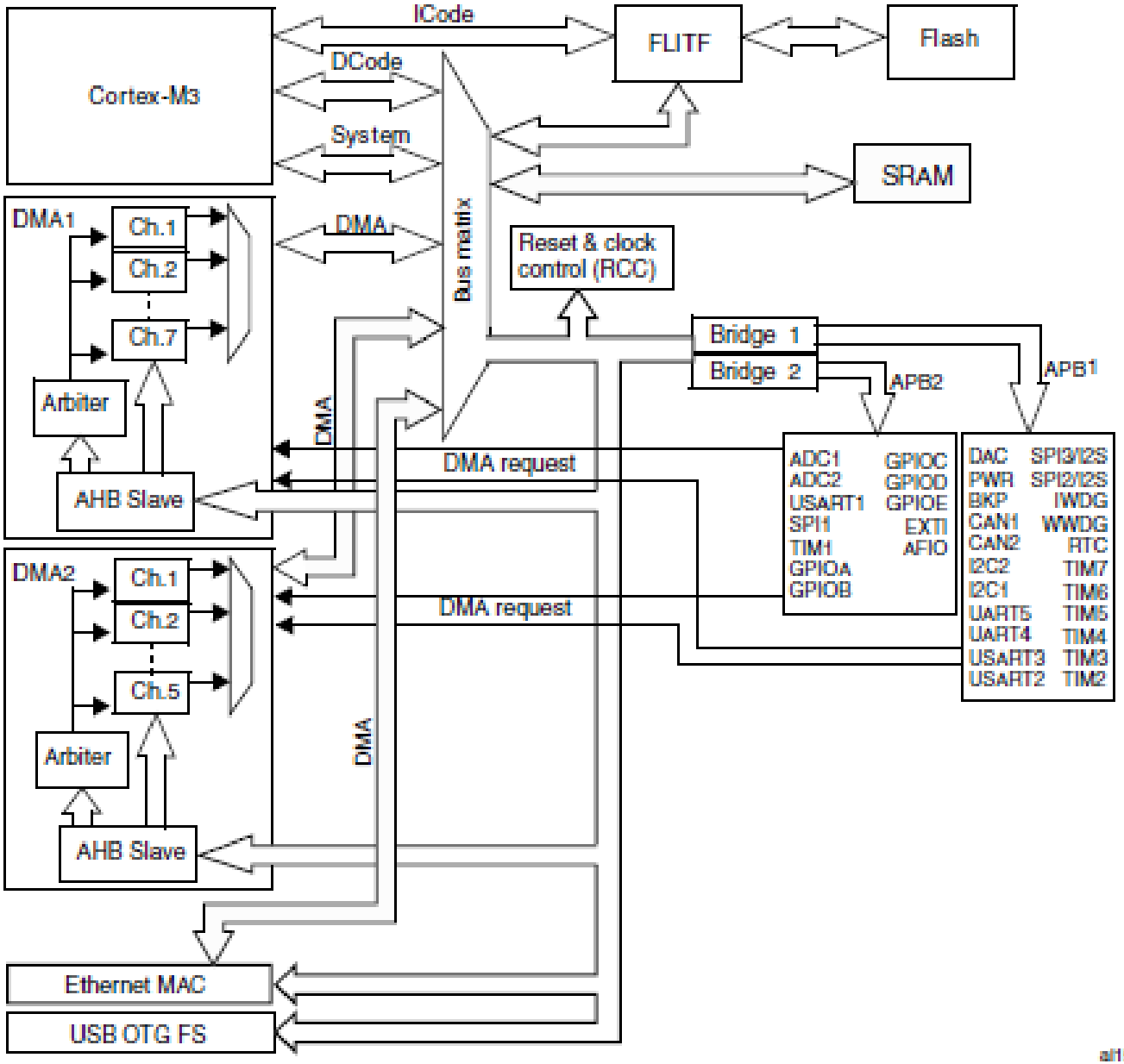
- DMA provides high speed data access between peripherals and memory or memory and memory

- Data can be moved quickly by DMA without CPU actions

ARM Cortex DMA

- ARM Cortex M-3 provides 2 DMA controllers with 12 channels in total (7 for DMA1 and 5 for DMA2)
- Independent source and destination transfer size (byte, half word, word)
- Support 4 level of priority: very high, high, medium, and low

DMA block diagram



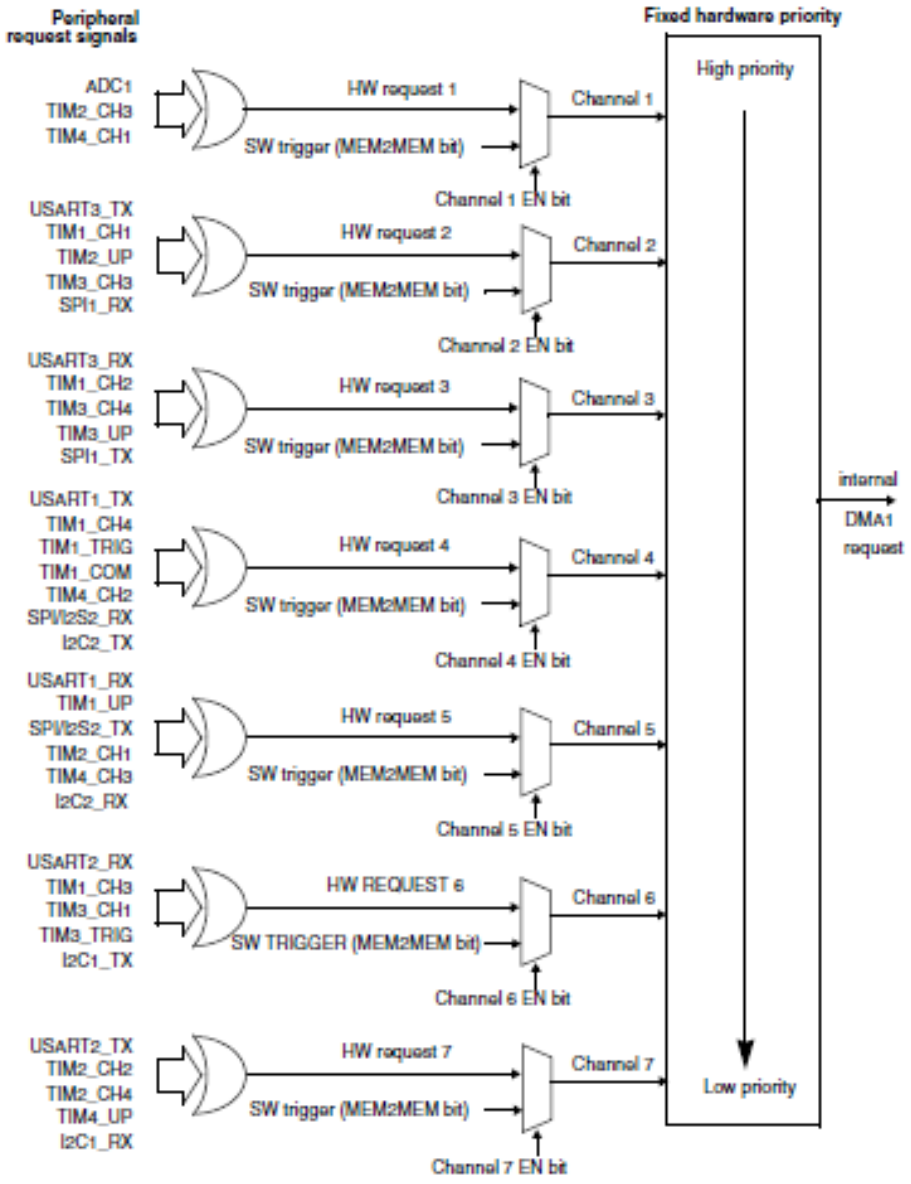
DMA transaction

- The peripheral sends a request signal to the DMA controller
- The DMA controller serves the request depending on the channel priorities
- As soon as DMA accesses the peripheral, the Acknowledge is sent to the peripheral by DMA controller
- The peripheral releases the request
- The DMA controller releases the Acknowledge

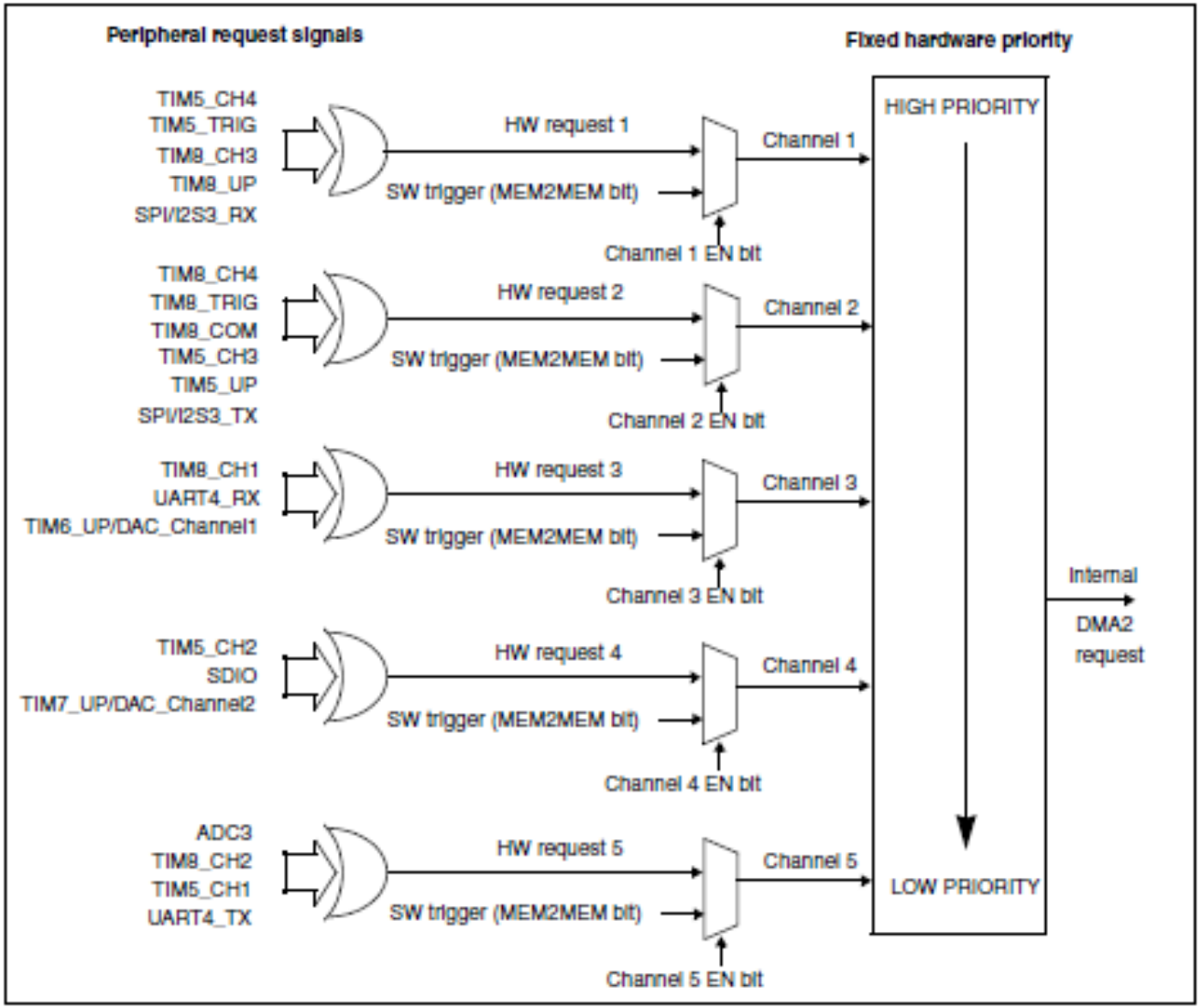
Arbiter

- DMA controller manages the channel requests based on their priority and launches the peripheral/memory access sequences
- The priorities are managed in two stages:
  - Software:
    - Very high, high, medium, and low priorities
  - Hardware:
    - If 2 requests have the same software priority level, the channel with lowest number will get the priority

DMA1 request mapping



DMA2 request mapping



Example of program

```
u8 TxBuffer[100]  = “\rUSART1 Transmission using DMA \r\n”;
u8 RxMessage[2] = [0];
int main(){
  RCC_setup();
  GPIO_setup();
  NVIC_setup();
  DMA_setup();
  USART1_setup();
  usart1_puts(“Test Message 1 \r\n”);
  while(1);
}
```

```
void GPIO_setup(){
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC |
  RCC_APB2Periph_AFIO,ENABLE);  // Enable GPIOA, GPIOC, and AFIO channel
  // Configure USART1 Tx (PA9) as alternate function push-pull
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // Configure USART1 RX (PA10 ) as Input floating
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOA, &GPIO_InitStructure);
 // Configure  PC6, PC7, PC8 and PC9 as Output push-pull connect with LED
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8 | GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

```
void NVIC_setup(){
  NVIC_InitTypeDef NVIC_InitStructure;
  // Enable the DMAChannel5 Interrupt (for RX USART1 interrupt)
  NVIC_InitStructure.NVIC_IRQChannel = DMAChannel5_IRQChannel;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
void USART1_setup(){
  USART_InitTypeDef USART_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE); //Enable USART clock
  USART_InitStructure.USART_BaudRate = 115200;
  USART_InitStructure.USART_WordLength = USART)WordLength_8b;
  USART_InitStructure.USART_Stopbits = USART_StopBits_1;
  USART_InitStructure.USART_Priority = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  USART_InitStructure.USART_Clock = USART_Clock_Disable;
  USART_InitStructure.USART_CPOL = USART_CPOL_Low; // clock active low
  USART_InitStructure.USART_CPHA = USART_CPHA_2Edge; // data on 2nd edge
  USART_InitStructure.USART_LastBit = USART_LastBit_Disable;
```

```
  USART_Init(USART1, &USART_InitStructure);
  USART_DMACmd(USART_DMAReq_Rx | USART_DMAReq_Tx, ENABLE);
  USART_Cmd(USART1,ENABLE);
}

void  DMA_setup(){
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA, ENABLE); // DMA clock enable
  DMA_DeInit(DMA_Channel4);
  DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART1->DR;
  DMA_InitStructure.DMA_MemoryBaseAddr = (u32) TxBuffer;  // transmit data
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
  DMA_InitStructure.DMA_BufferSize =  strlen(TxBuffer);
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MeoryInc_Enable;
  DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
  DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
  DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
  DMA_Init(DMA_Channel4,&DMA_InitStructure);
  DMA_Cmd(DMA_Channel4, ENABLE);
```

```
  DMA_DeInit(DMA_Channel5);
  DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART1->DR;
  DMA_InitStructure.DMA_MemoryBaseAddr = (u32) RxMessage; // receive data
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
  DMA_InitStructure.DMA_BufferSize =  strlen(TxBuffer);
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MeoryInc_Disable;
  DMA_InitStructure.DMA_DMA_BufferSize = 1;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
  DMA_Init(DMA_Channel5,&DMA_InitStructure);
  DMA_ITConfig(DMA_Channel5,DMA_IT_TC_ENABLE); // Enable interrupt when
     receive data
  DMA_Cmd(DMA_Channel5, ENABLE);
}

/* CPOL =  clock polarity (if 1, clock has high level idle state
    CPHA = clock phase (if 1, falling edge. If 0, rising edge
    M2M = memory to memory transfer
    Last_bit = If 1 is o use last bit for sending data (have to use NACK, otherwise using it
      for control signal) */
```

```
void usart1_puts(unsigned char *s){
 while(DMA_GetFlagStatus(DMA_FLAG)TC4) == RESET); // wait until send ready
 DMA_Cmd(DMA_Channel4, DISABLE);
 strcpy(TxBuffer,s);
 DMA_DeInit(DMA_Channel4);
 DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&USART1->DR;
 DMA_InitStructure.DMA_MemoryBaseAddr = (u32) TxBuffer;
 DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
 DMA_InitStructure.DMA_BufferSize =  strlen(TxBuffer);
 DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
 DMA_InitStructure.DMA_MemoryInc = DMA_MeoryInc_Enable;
 DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
 DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
 DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
 DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
 DMA_Init(DMA_Channel4,&DMA_InitStructure);
 DMA_Cmd(DMA_Channel4, ENABLE);

}
```

```
void DMAChannel5_IRQHandler(void){
  if(DMA_GetFlagStatus(DMA_IT_TC5)==SET){ // ensure check interrupt flag for receive
      data
     if(RxMessage[0]=='1'){  // Receive character 1
        GPIO_WriteBit(GPIOC, GPIO_Pin_6, (BitAction)((1-
      GPIO_ReadOutputDataBit(GPIOC,GPIO_Pin_6)));  // Togge PC6 pin
     }
     if(RxMessage[0]=='2'){  // Receive character 2
        GPIO_WriteBit(GPIOC, GPIO_Pin_7, (BitAction)((1-
      GPIO_ReadOutputDataBit(GPIOC,GPIO_Pin_7)));  // Togge PC7 pin
     }
     if(RxMessage[0]=='3'){  // Receive character 3
        GPIO_WriteBit(GPIOC, GPIO_Pin_8, (BitAction)((1-
      GPIO_ReadOutputDataBit(GPIOC,GPIO_Pin_8)));  // Togge PC8 pin
     }
     if(RxMessage[0]=='4'){  // Receive character 4
        GPIO_WriteBit(GPIOC, GPIO_Pin_9, (BitAction)((1-
      GPIO_ReadOutputDataBit(GPIOC,GPIO_Pin_9)));  // Togge PC9 pin
     }
      DMA_ClearITPendingBit(DMA_IT_TC,5); /// Clear Interrupt pending bit
   }
}
```

Questions?