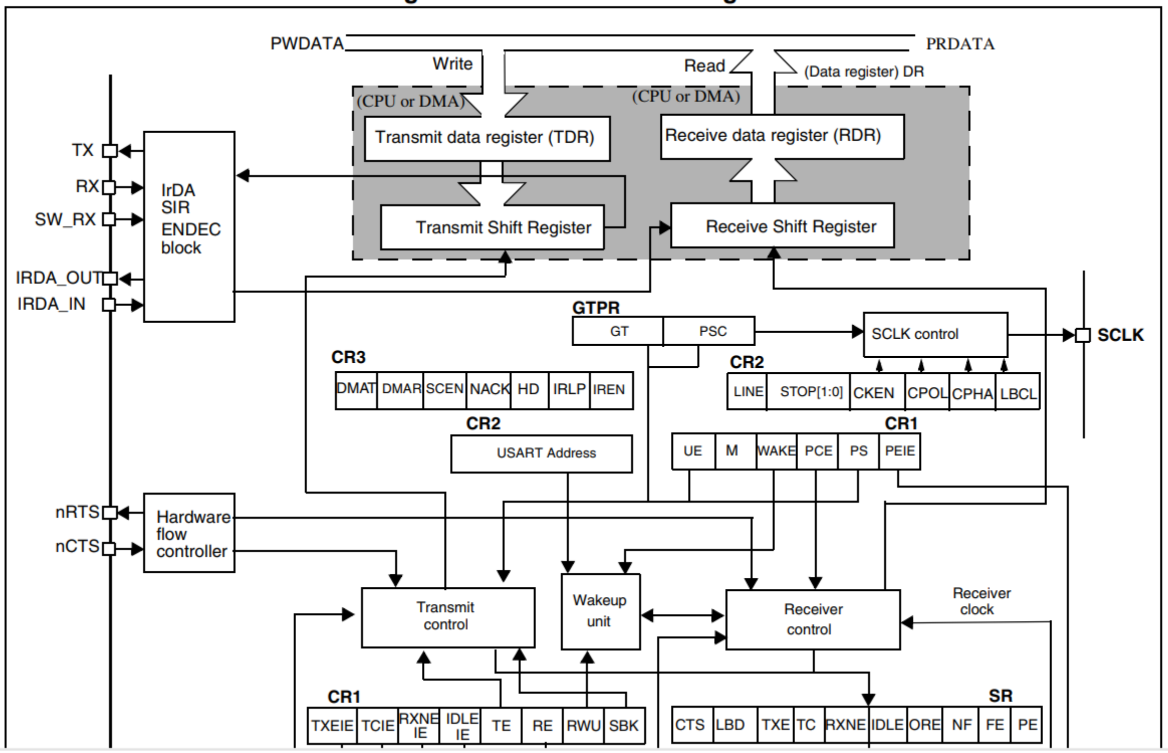
UART is a on-chip serial communication peripheral on the microcontroller. By which it can output serial data to a particular pin of the microcontroller.

UART stands for “Universal Asynchronous Receiver / Transmitter”, and it is a very simple serial communication interface. In its most basic form, it only uses two data signals: “Receive” (RX) and “Transmit” (TX). Since it is asynchronous (no clock signal), both devices need to use the same “baud rate”.

One of the most common uses of UART is to transmit strings of text or binary data between devices

Block diagram of USART/UART is as follows

Figure 278. USART block diagram



As shown in the figure

Now this UART can be connected to PA2(TxD where uart transmits) and PA3(RxD where UART receives).

Now to program the UART

Following steps need to be carried out for **configuring UART2.**

1. Enable the clock for UART and GPIO thru its relevant Bus registers.
2. Configure UART pins PA2 and PA3 for alternate function thru GPIOA->MODER.(see STM32 data sheet for alternate functions).
3. Set the speed of the PORTA pins PA2 and PA3 to high speed using OSPEEDR registers.
4. Configure PA2 and PA3 for alternate functions using AFRL and AFRH registers of GPIO, (see alternate functionality pins on STM32F446re data sheet 57 page table for ref).
5. Enable UE bit on USART->CR1 register to further configure USART related registers.
6. Program M bit in USART->CR register to configure data length, UART data(our data is 8 bit)
7. Then set the baud rate for USART using USART->BRR according to the given formula on page of the RM 0390 manual.(our baud rate is 115200)(like 7 binary on first 4 bits of BRR and 24 on second 4 bits of BRR for a baud rate of 115200)
8. Now Enable both Transmitter and receiver on USART->CR1 register.

Following are the steps to use the **USART2 as a transmitter**

After configuring the USART2

1. Write your 1 byte data on the USART2->DR register.
2. After writing the UART starts transmitting data serial to the port PA2 to putty. In the process of transmission of data if the TD register becomes empty then it indicates a flag on USART->SR register.
3. We wait for this status flag TxC after transmitting using a while loop.
   ```
   while (!(USART2->SR & (1<<6)));
   ```
4. then we move further in the program.

Following are the steps to use the **USART2 as a Receiver**

After configuring the USART2

1. Wait for the status register of UART to become high on RxNE bit of USART2->SR register using a while loop like this `while (!(USART2->SR & (1<<5)));`
2. Then collect the `USART2->DR` in a temporary variable.
3. Move further in the program.

Requirements for this program

Use the following function and call it before configuring and using USART in the program

This function is meant for raising the CPU clock to 180 Mhz which we will look into it later in the course.

```
void SysClockConfig (void)
{

	#define PLL_M 	4
	#define PLL_N 	180
	#define PLL_P 	0  // PLLP = 2

	// 1. ENABLE HSE and wait for the HSE to become Ready
	RCC->CR |= RCC_CR_HSEON;
	while (!(RCC->CR & RCC_CR_HSERDY));

	// 2. Set the POWER ENABLE CLOCK and VOLTAGE REGULATOR
	RCC->APB1ENR |= RCC_APB1ENR_PWREN;
	PWR->CR |= PWR_CR_VOS;

	// 3. Configure the FLASH PREFETCH and the LATENCY Related
Settings
	FLASH->ACR = FLASH_ACR_ICEN | FLASH_ACR_DCEN | FLASH_ACR_PRFTEN |
FLASH_ACR_LATENCY_5WS;

	// 4. Configure the PRESCALARS HCLK, PCLK1, PCLK2
	// AHB PR
	RCC->CFGR |= RCC_CFGR_HPRE_DIV1;

	// APB1 PR
	RCC->CFGR |= RCC_CFGR_PPRE1_DIV4;

	// APB2 PR
	RCC->CFGR |= RCC_CFGR_PPRE2_DIV2;


	// 5. Configure the MAIN PLL
	RCC->PLLCFGR = (PLL_M <<0) | (PLL_N << 6) | (PLL_P <<16) |
(RCC_PLLCFGR_PLLSRC_HSE);

	// 6. Enable the PLL and wait for it to become ready
	RCC->CR |= RCC_CR_PLLON;
	while (!(RCC->CR & RCC_CR_PLLRDY));

	// 7. Select the Clock Source and wait for it to be set
	RCC->CFGR |= RCC_CFGR_SW_PLL;
	while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL);
}
```