USART

Programming by Registers

Any thoughts ?

```c
#include "stm32f103x6.h"

void usartInit () {
  RCC->APB2ENR |= RCC_APB2ENR_USART1EN | RCC_APB2ENR_IOPAEN;
  GPIOA->CRH &= 0x44444004;
  GPIOA->CRH |= 0x000008A0;
  USART1->BRR = 0x341;                  // set baudrate
  USART1->CR1 |= USART_CR1_RE | USART_CR1_TE | USART_CR1_UE;
  }


int sendChar (char ch)  {
  while (!(USART1->SR & USART_SR_TXE));
  USART1->DR |= (ch & 0xFF);
  return (ch);
}

int getChar (void)  {
  while (!(USART1->SR & USART_SR_RXNE));
  return ((int)(USART1->DR & 0xFF));
}

int main()
{
  usartInit();
  while (1)
  {
    sendChar(getChar());
  }
}
```

Set UARTS Pin

```
void usartInit () {
  RCC->APB2ENR |= RCC_APB2ENR_USART1EN | RCC_APB2ENR_IOPAEN;
  GPIOA->CRH &= 0x44444004;
  GPIOA->CRH |= 0x000008A0;
```

9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

Set Baud Rate

```
USART1->BRR = 0x341;
```

$$BAUD = \frac{F_{ck}}{(16 \cdot USARTDIV)}$$

$$USARTDIV = \frac{72000000}{(16 \cdot 9600)} = 468.75$$

$468 = 0x1D4$ <-----

$0.75 \cdot 16 = 12 = 0xC$ <---------

USARTDIV = 0x1D4C

27.3.4 Fractional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value a
programmed in the Mantissa and Fraction values of USARTDIV.

$$\text{Tx/ Rx baud} = \frac{f_{CK}}{(16*\text{USARTDIV})}$$

legend: f_{CK} - Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

Enable USART, RX, and TX

```
USART1->CR1 |= USART_CR1_RE | USART_CR1_TE | USART_CR1_UE;
```

27.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Programmer

10 0000 0000 1100

HEX	200C
DEC	8.204
OCT	20 014
BIN	0010 0000 0000 1100

sendChar

```
int sendChar (char ch)  {
  while (!(USART1->SR & USART_SR_TXE));
  USART1->DR |= (ch & 0xFF);
  return (ch);
}
```

getChar

```
int getChar (void)  {
  while (!(USART1->SR & USART_SR_RXNE));
  return ((int)(USART1->DR & 0xFF));
}
```

main()

```
int main()
{
  usartInit();
  while (1)
  {
    sendChar(getChar());
  }
}
```

Practices :

- Turning on/off led by command ‘1’ or ‘0’
- Turning on/off led by command ‘ON’ or ‘OFF’