Application Note

32-bit Cortex™-M0 MCU
NuMicro® Family

How to print information into PC via UART?

Table of Contents-

1 INTRODUCTION

This MCU provides three channels of Universal Asynchronous Receiver/Transmitters (UART) to communicate easily with PC and other device. UART0 supports High Speed UART and UART1~2 perform Normal Speed UARTs, besides, only UART0 and UART1 support flow control function.

1.1 Feature for PC communicate

- The UART control supports three channels, UART0, UART1 and UART2.
- UART0/UART1/UART2 supports 64/16/16 bytes entry FIFO for received and transmitted data payloads.
- Auto flow control/flow control function (/CTS, /RTS) are supported in UART0 and UART1.
- Individual programmable baud-rate generator for each channel.
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit character.
  - Even, odd, or no-parity bit generation and detection.
  - 1-, 1&1/2, or 2-stop bit generation.
  - Baud rate generation.
  - False start bit detection.

1.2 Structure

The UART clock control and block diagram are shown as following.

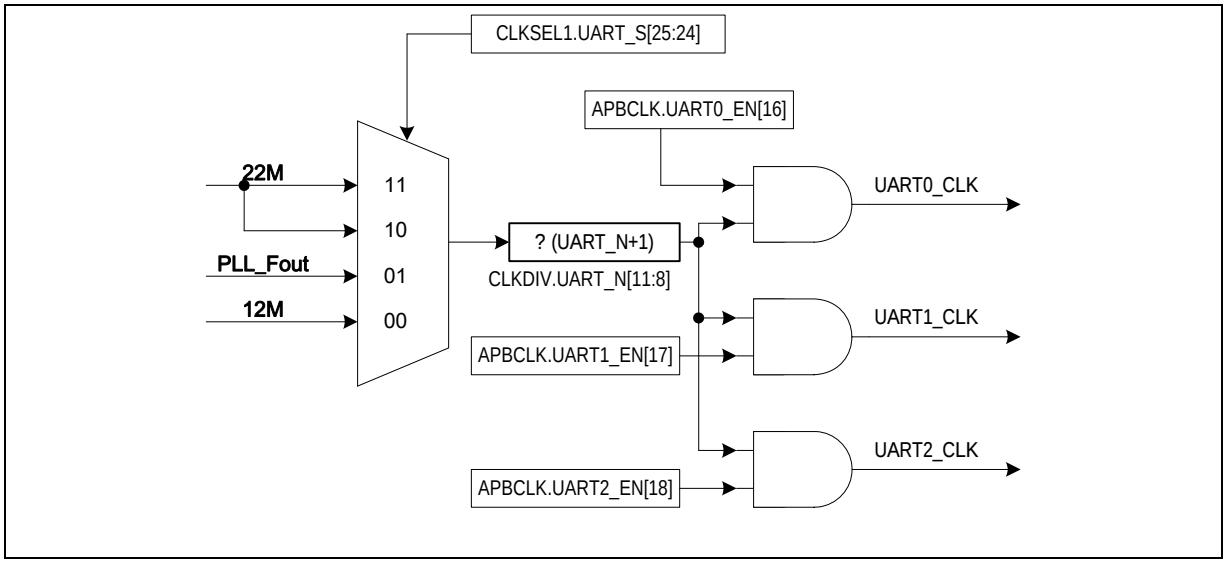


Figure 1 UART Clock Control Diagram

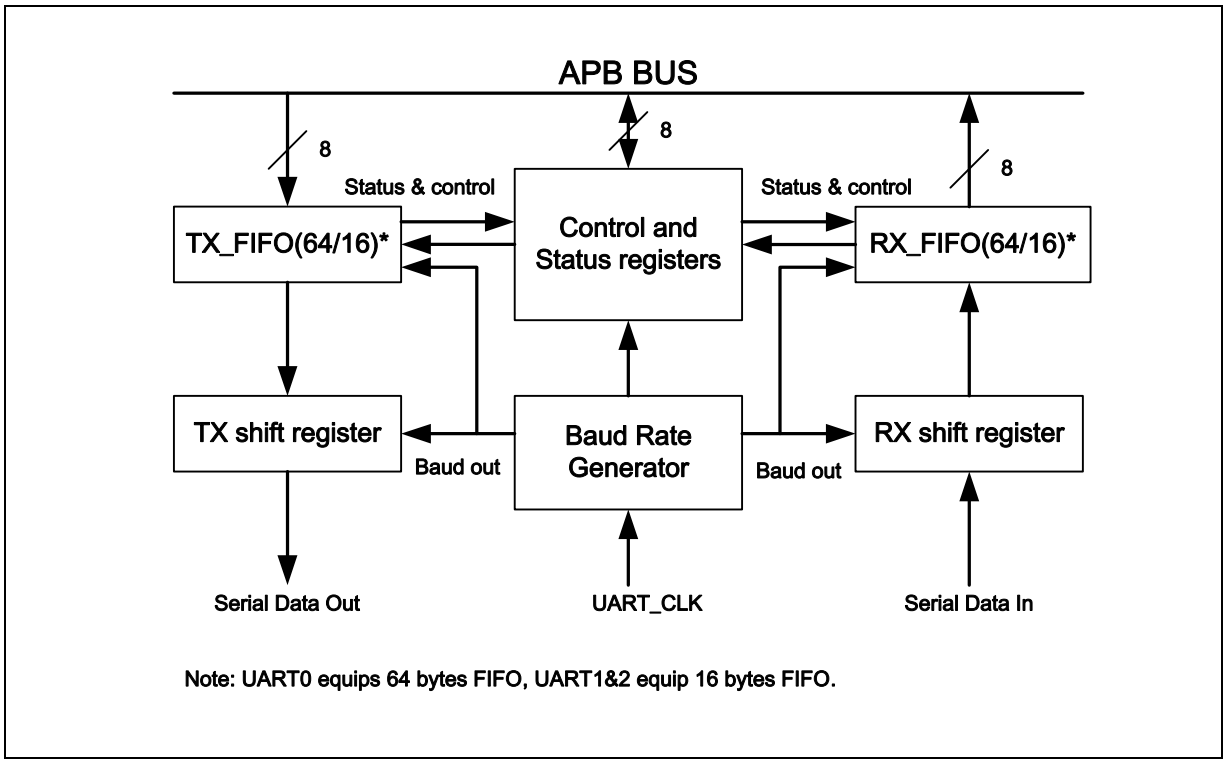


Figure 2 UART Block Diagram

TX_FIFO
The transmitter is buffered with a 64/16 byte FIFO to reduce the number of interrupts presented to the CPU.

RX_FIFO
The receiver is buffered with a 64/16 byte FIFO (plus three error bits per byte) to reduce the number of interrupts presented to the CPU.

TX shift Register
Shifting the transmitting data out serially

RX shift Register
Shifting the receiving data in serially

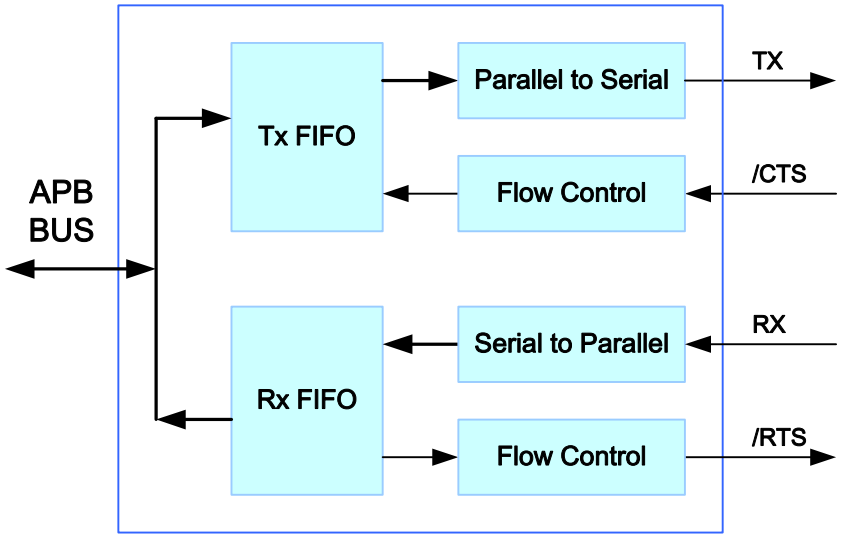
Modem Control Register
This register controls the interface to the MODEM or data set (or a peripheral device emulating a MODEM).

Baud Rate Generator
Dividing the external clock by the divisor to get the desired baud rate clock.

Control and Status Register
This is a register set, including the FIFO control registers (FCR), FIFO status registers (FSR), and line control register (LCR) for transmitter and receiver. The time out control register (TOR) identifies the condition of time out interrupt. This register set also includes the interrupt enable register (IER) and interrupt status register (ISR) to enable or disable the responding interrupt and to identify the occurrence of the responding interrupt. There are **six types of interrupts**, transmitter FIFO empty interrupt(INT_THRE), receiver threshold level reaching interrupt (INT_RDA), line status interrupt

(overrun error or parity error or framing error or break interrupt) (INT_RLS) , time out interrupt (INT_Tout), MODEM/Wakeup status interrupt (INT_Modem) and Buffer error interrupt (INT_Buf_Err).

The following diagram demonstrates the auto-flow control block diagram.



Note: Only supported in UART0 and UART1

Figure 3 Auto Flow Control Block Diagram

2 HOW TO PROGRAM UART TO PC

2.1 PROGRAM FLOW OF UART to PC

1. Prepare a retarget function to replace ANSI library function
2. Set **UART0_RX** and **UART0_TX** bit in **GPBMFP** register to initiate GPIO for UART function
3. Select **UART_S** bit in **CLKSEL1** register and **UART_N** bit in **CLKDIV** register and then set **UART0_EN** bit in **APBCLK** register
4. Select **RFITL** bit in **FCR** register, **PBE** bit in **LCR** register, **WLS** bit in **LCR** register and **NSB** bit in **LCR** register to define UART format.
5. Select **DIVX_EN** bit, **DIVX1** bit and **DIV** bit in **BAUD** register to define BaudRate.

2.2 Sample code

2.2.1 Smpl_UART2PC.c

```
#include <stdio.h>
#include "NUC1xx.h"
/*---------------------------------------------------------------------------
  MAIN function
  ---------------------------------------------------------------------------*/
int32_t main (void)
{

        /* Step 1. GPIO initial */
        SYS->GPBMFP.UART0_RX      =1;
        SYS->GPBMFP.UART0_TX      =1;

        /* Step 2. Enable and Select UART clock source*/
        UNLOCKREG();
        SYSCLK->PWRCON.XTL12M_EN = 1;
        LOCKREG();
        SYSCLK->APBCLK.UART0_EN = 1;      //Enable UART clock
        SYSCLK->CLKSEL1.UART_S = 0;       //Select 12Mhz for UART clock source
        SYSCLK->CLKDIV.UART_N = 0;        //UART clock source = 12Mhz;

        /* Step 3. Select Operation mode */
        UART0->FCR.TFR =1;                //Reset Tx FIFO
        UART0->FCR.RFR =1;                //Reset Rx FIFO
        UART0->FCR.RFITL = 0;/            /Set Rx Trigger Level -1byte FIFO
```

```
        UART0->LCR.PBE      = 0;            //Disable parity
        UART0->LCR.WLS      = 3;            //8 data bits
        UART0->LCR.NSB      = 0;            //Enable 1 Stop bit

        /* Step 4. Set BaudRate */
        UART0->BAUD.DIVX_EN = 1;
        UART0->BAUD.DIVX1   = 1;
        UART0->BAUD.DIV = 12000000 / 115200  -2;

        do
        {
                printf("\nUART Sample Demo. (Press 'ESC' to exit)\n");
        }while(GetChar()!=0x1B);
        printf("Exit\n");
}
```

2.2.2 retarget.c

```
#include <stdio.h>
#include "NUC1xx.h"

#if defined ( __CC_ARM   )
#if (__ARMCC_VERSION < 400000)
#else
/* Insist on keeping widthprec, to avoid X propagation by benign code in C-lib */
#pragma import _printf_widthprec
#endif
#endif

/*---------------------------------------------------------------------------------------------------------*/
/* Global variables
/*---------------------------------------------------------------------------------------------------------*/
struct __FILE { int handle; /* Add whatever you need here */ };
FILE __stdout;
FILE __stdin;

/*---------------------------------------------------------------------------------------------------------*/
/* Routine to write a char
/*---------------------------------------------------------------------------------------------------------*/
void SendChar(int ch)
{
            while(UART0->FSR.TX_FULL == 1);
            UART0->DATA = ch;
    if(ch == '\n')
    {
        while(UART0->FSR.TX_FULL == 1);
        UART0->DATA = '\r';
    }
}
/*---------------------------------------------------------------------------------------------------------*/
/* Routine to get a char                                                                                   */
```

```
/*---------------------------------------------------------------------------------------------------------*/
char GetChar()
{
        while (1)
        {
                if(UART0->FSR.RX_EMPTY == 0 )
                {
                        return (UART0->DATA);
           }
        }
}
/*---------------------------------------------------------------------------------------------------------*/
/* C library retargeting                                                                                   */
/*---------------------------------------------------------------------------------------------------------*/
void _ttywrch(int ch)
{
  SendChar(ch);
  return;
}

int fputc(int ch, FILE *f)
{
  SendChar(ch);
  return 0;
}

int fgetc(FILE *f) {
  return (GetChar());
}

int ferror(FILE *f) {
  return EOF;
}
```

2.2.3 Baud-rate setting for 22.1184Mhz

System clock = 22.1184Mhz			
Baudrate	Mode0	Mode1	Mode2
921600	x	A=0,B=11	A=22
460800	A=1	A=1,B=15 A=2,B=11	A=46
230400	A=4	A=4,B=15 A=6,B=11	A=94
115200	A=10	A=10,B=15 A=14,B=11	A=190
57600	A=22	A=22,B=15 A=30,B=11	A=382
38400	A=34	A=62,B=8 A=46,B=11 A=34,B=15	A=574
19200	A=70	A=126,B=8 A=94,B=11 A=70,B=15	A=1150
9600	A=142	A=254,B=8 A=190,B=11 A=142,B=15	A=2302
4800	A=286	A=510,B=8 A=382,B=11 A=286,B=15	A=4606

3 REVISION HISTORY

REV.	DATE	DESCRIPTION
1.00	March 1, 2010	1. Initially issued.
1.01	April 8, 2010	1. Remove register description

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Further more, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

Please note that all data and specifications are subject to change without notice. All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.