

应用指南

32-bit Cortex™-M0 MCU NuMicro® Family

怎样将信息从UART打印到PC?

目录

1	简介	2
1.1	与PC通讯特性	2
1.2	结构	2
2	UART到PC的编程	5
2.1	UART到PC编程流程	5
2.2	示例程序	5
2.2.1	Smp1_UART2PC.c	6
2.2.2	retarget.c	8
2.2.3	22.1184Mhz的波特率设定	10
3	修订历史	11

1 简介

NUC1xx系列MCU提供3组通用异步收发器(UART)来与PC或其他设备进行通讯，UART0支持高速UART，UART1,2是普通UART，UART0,1有流控制功能。

1.1 与PC通讯特性

- UART控制器支持3通道，UART0，UART1和UART2
- UART0/UART1/UART2提供64/16/16字节FIFO作为接收和发射数据的缓冲
- UART0和UART1支持自动流控制功能（/CTS, /RTS）
- 每个通道拥有独立的可编程波特率产生器
- 完全可编程串行接口特性
 - . 5, 6, 7或8位字符
 - . 奇 / 偶校验位或无校验位产生及侦测
 - . 1, 1&1/2, 或2位停止位
 - . 波特率产生器
 - . 起始位错误检测

1.2 结构

UART时钟控制如下图所示：

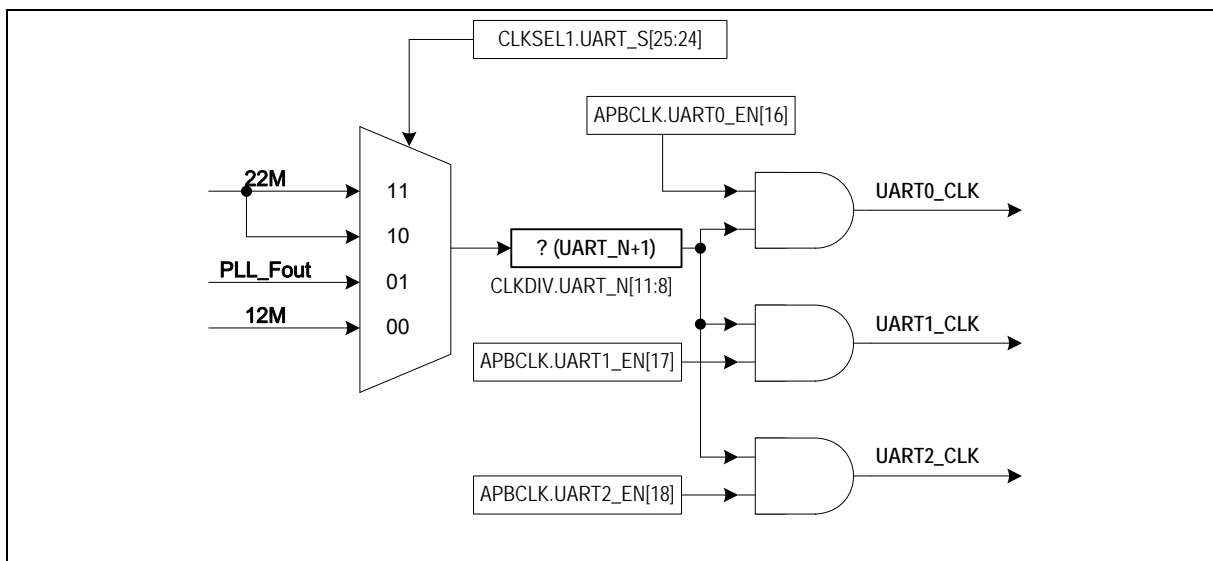


图1 UART时钟控制图

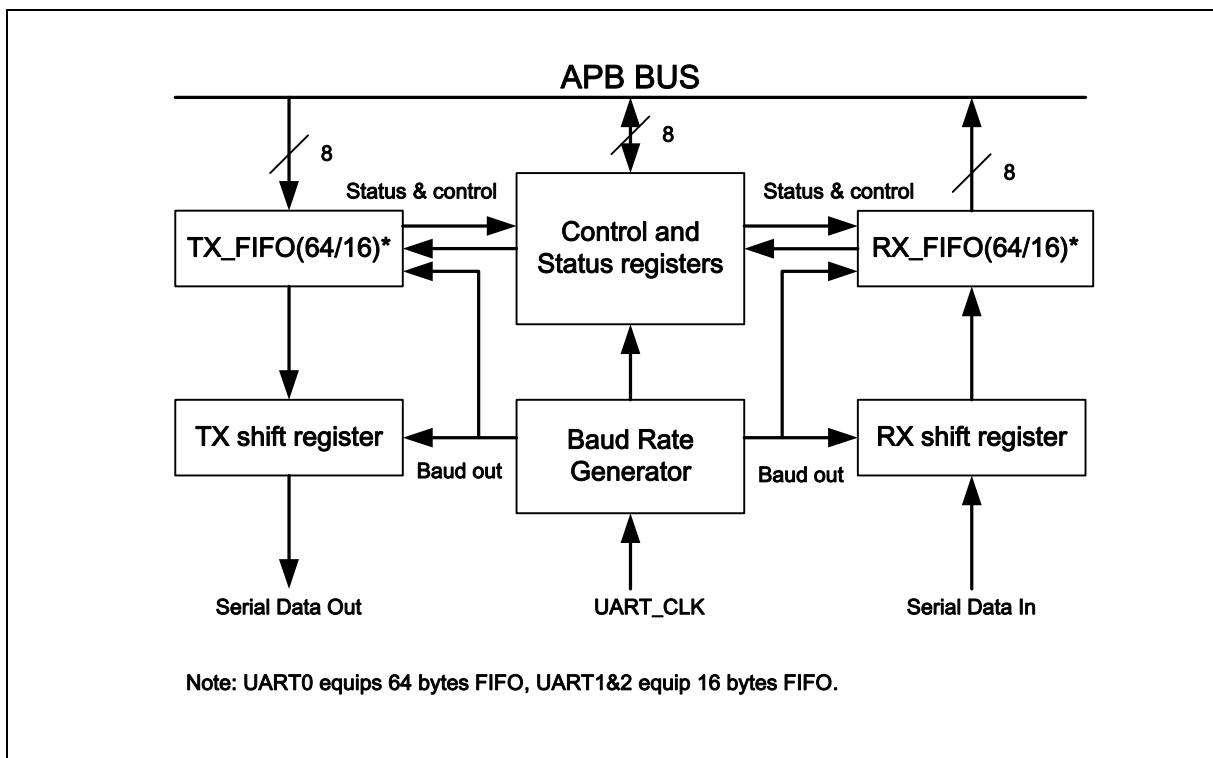


图2 UART框图

TX_FIFO

发送器拥有64/16字节的缓冲器，可以减少CPU的发送中断次数

RX_FIFO

接收器拥有64/16字节的缓冲器（每字节加3位错误检测位），可以减少CPU的接收中断次数

TX移位寄存器

将要发送的数据连续移出

RX移位寄存器

将接收的数据连续移入

调制解调器控制寄存器

用来控制调制解调器或数据设备(或者终端设备模拟调制解调器)的接口

波特率产生器

将外部时钟通过除法器得到需要的波特率时钟

控制及状态寄存器

这是一组寄存器，包括FIFO控制寄存器（FCR），FIFO状态寄存器（FSR），线性控制寄存器（LCR）。时间溢出控制寄存器（TOR）标识了时间溢出中断的状态。这组寄存器也包括中断使能寄存器（IER）和中断状态寄存器（ISR），这用来开/关相应的中断以及判断中断源。共有6种类型的中断：发送器FIFO空中断（INT_THRE），接收数据中断（INT_RDA），线性状态中断（数据覆盖错误，奇偶校验错误，帧错误，暂停中断）（INT_RLS），时间溢出中断（INT_TOUT），调制解调器/唤醒状态中断（INT_MODEM）以及缓冲区错误中断（INT_BUF_ERR）。下图描述了自动流量控制：

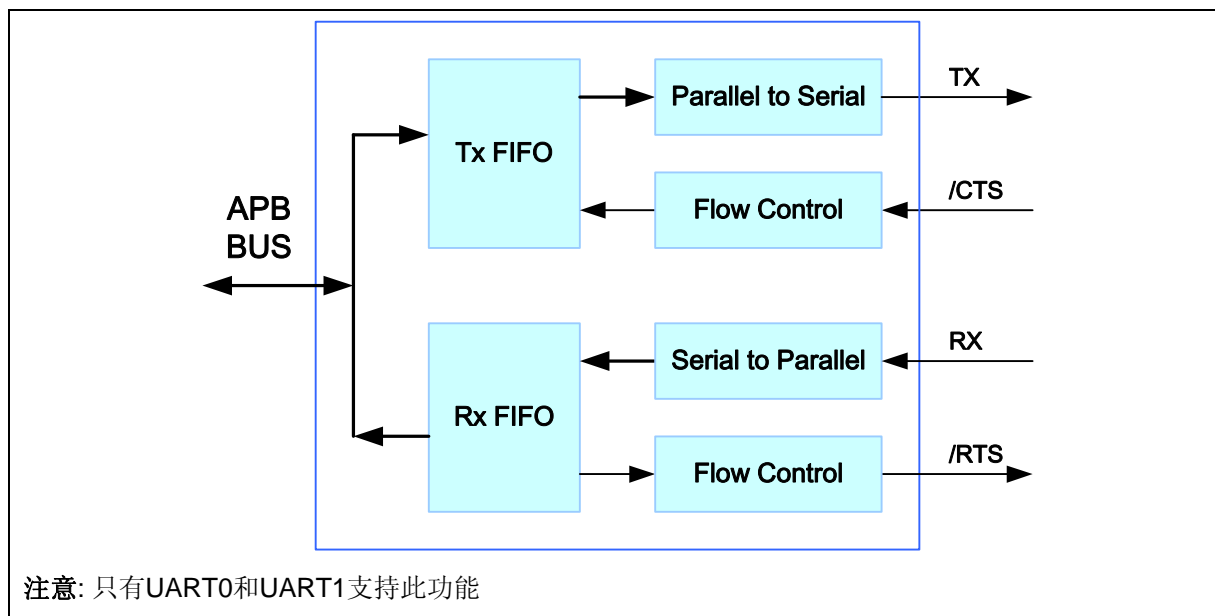


图3 自动流量控制框图

2 UART到PC的编程

2.1 UART到PC编程流程

1. 重写UART的输出输入函数代替ANSI中的库函数功能
2. 设定寄存器 GPBMFP. UART0_RX 和 GPBMFP. UART0_TX 初始化 GPIO 的 UART 功能
3. 根据所要的时钟来设置寄存器 CLKSEL1. UART_S, CLKDIV. UART_N 以及 APBCLK. UART0_EN
4. 通过设置寄存器 FCR. RFITL, LCR. PBE, LCR. WLS 以及 LCR. NSB 来定义 UART 的数据格式
5. 根据波特率来设定 UA_BAUD. DIV_X_EN, UA_BAUD. DIV_X_ONE 以及 UA_BAUD. Divider_X 的值

2.2 示例程序

2.2.1 Smp1_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 22.1184Mhz的波特率设定

System clock = 22.1184Mhz			
波特率	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 修订历史

版本.	日期	描述
1.00	2010-3-1	1. 初次发布
1.01	2010-4-8	2. 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.