Application Note

32-bit Cortex™-M0 MCU NuMicro® Family

Power Management and Power Down/IDLE mode

Table of Contents-

1 INTRODUCTION

IDLE/Power Down mode provides the minimum power consumption. When system is not working or just waiting for an external interrupt/event, software makes system to enter IDLE/Power Down mode. In this state, all peripherals are also in sleep mode since the clock source is stopped. System exits IDLE/Power Down state when external interrupt/event is detected.

This document explains the sample code, “Smpl_PowerManagement” which is included in the ***AN_1007_EN.ZIP*** file and demonstrates how to enter to system IDLE/Power Down state and wake up from IDLE/Power Down through the related libraries.

Power Down Mode Control Table

Register/Instruction / **Mode**	**PWR_DOWN_EN**	**PD_WAIT_CPU**	**CPU run WFE/WFI instruction**	**Clock Gating**
Normal Running Mode	1’b0	1’b0	NO	All Clock Gating by control register
IDLE Mode **(CPU entry Sleep Mode)**	1’b0	1’b0	YES	Only CPU clock gating
Power_down Mode	1’b1	1’b0	NO	Most Clock are gating except 10K/32K and some RTC/WDT/Timer/PWM/ADC peripheral clock are still active.
Power_down Mode **(CPU entry deep sleep mode)**	1’b1	1’b1	YES	Most Clock are gating except 10K/32K and some RTC/WDT/Timer/PWM/ADC peripheral clock are still active.

1.1 Features

- Support disable the selected single IP clock for power saving.
- Support system IDLE mode and wake up by RTC/GPIO/UART/USB/CAN interrupt event.
- Support system Power Down mode and wake up by RTC/GPIO/UART/USB/CAN interrupt event.

2 CODE SECTION

Most the verified functions are declared in Smpl_PowerManagement.c.

Refer chapter of Calling Sequence for the detail calling sequence

2.1 Main function

In the main function, we set the external 12MHz as the system clock for the later test.

After the hardware initialization is finished, open UART0 and configure baud rate to 115200 for output debug message. Then, you can input the item which you want to test.

```
// Poll and handle USB events.
DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
......
DrvUART_Open(UART_PORT0, &sParam);
……
While (i32Loop)
{
        ……
             printf("| [1] IP Clock Control Test                |\n");
            printf("| [2] Idle Mode Test                         |\n");
            printf("| [3] Power Down Mode Test             |\n");
            switch (u8Item)
            {
                        case '0':
                                    i32Loop = 0;
                                    break;
                        case '1':
                                    IpClockCtrlTest();
                                    break;
                        case '2':           /* for IDLE Mode */
                        case '3':           /* for Power Down Mode */
                                    PowerDownAndIdleModeTest(u8Item-'2');
                                    break;
            }
}
```

2.2 IpClockCtrlTest function

In this function, user can input character 'a'~'z' to enable or disable the corresponding engine clock or input character '1' to enable or disable all of peripheral engine clock..

```
……
/* Deafult settings are all IP clocks enabled */
_ConfigureIPClock(&u32IPClkCfg);
……
if (u8Item == '1')
{
        if (u32IPClkCfg == ALL_IP_CLOCK)
                u32IPClkCfg = 0;
        else
                u32IPClkCfg = ALL_IP_CLOCK;
        printf("\r%s all IP ...\n\n", (u32IPClkCfg==0)? "Disable":"Enable");
}else
if ((u8Item >= 'a') && (u8Item <= 'z'))
{
        u8IPIndex = u8Item - 'a';
        u32IPClkCfg = u32IPClkCfg ^ (1<<u8IPIndex);
        printf("\rIndex-[%c] IP is %s ...\n\n", u8Item,
                 ((u32IPClkCfg&(1<<u8IPIndex))==0)? "Disable":"Enable");
}else
{
        continue;
}
 ……
/* Apply the stting */
DrvSYS_SetIPClock(E_SYS_WD_CLK,     ((u32IPClkCfg&WD_CLOCK)? 1:0));
DrvSYS_SetIPClock(E_SYS_RTC_CLK,    ((u32IPClkCfg&RTC_CLOCK)? 1:0));
DrvSYS_SetIPClock(E_SYS_TMR0_CLK,  ((u32IPClkCfg&TMR0_CLOCK)? 1:0));
DrvSYS_SetIPClock(E_SYS_TMR1_CLK,  ((u32IPClkCfg&TMR1_CLOCK)? 1:0));
DrvSYS_SetIPClock(E_SYS_TMR2_CLK,  ((u32IPClkCfg&TMR2_CLOCK)? 1:0));
……
```

2.3 PowerDownAndIdleModeTest function

About the PowerDownAndIdleModeTest function, when the argument is 0 means for IDLE mode test and 1 for Power Down mode test.

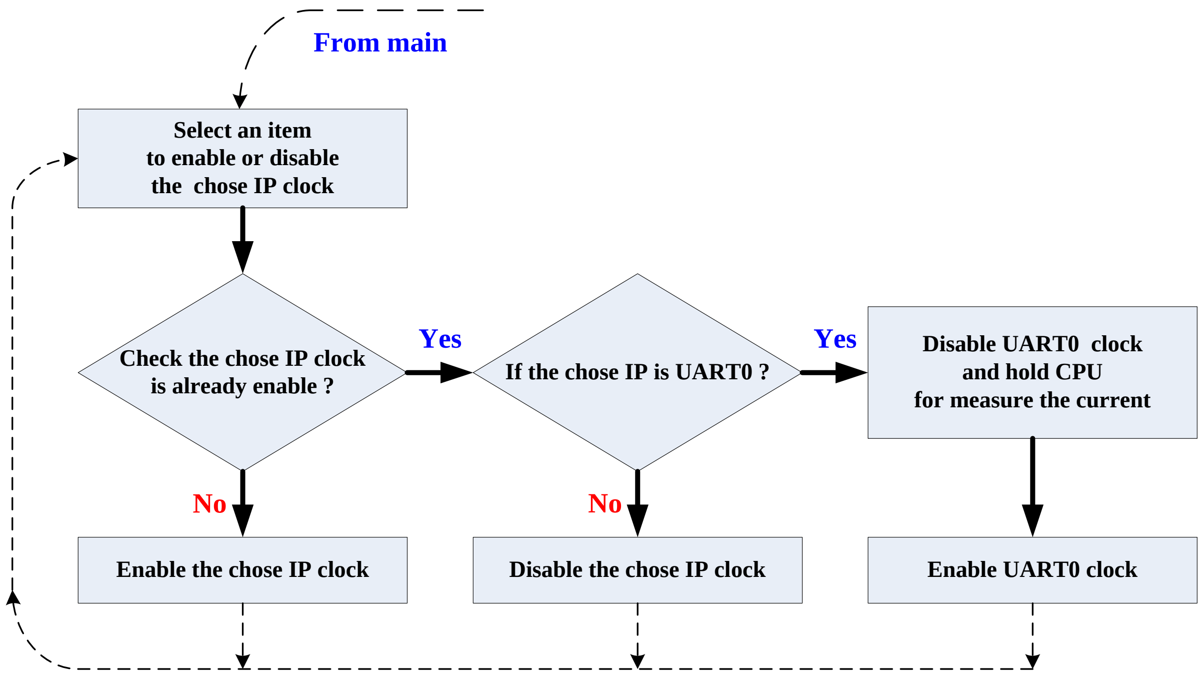
After choosing the test mode in main function, user must input ‘1’ ~ ‘5’ to select the wakeup type, RTC, GPIO, UART, USB or CAN interrupts.

```
……
printf("| [1] RTC wake up Test                              |\n");
printf("| [2] GPIO wake up Test …. (GPA1 toggle)            |\n");
printf("| [3] UART wake up Test ….(GPB3 toggle)             |\n");
printf("| [4] USB wake up Test                              |\n");
printf("| [5] CAN0 wake up Tes ….t(GPD6 toggle)             |\n");
……
switch (u8Item)
{
        case '1':
                if (InitRTCWakeupFunction() == FALSE)
                        continue;
                break;
        case '2':
                if (InitGPIOWakeupFunction() == FALSE)
                        continue;
                break;
        case '3':
                if (InitUARTWakeupFunction() == FALSE)
                        continue;
                break;
        case '4':
                if (InitUSBWakeupFunction() == FALSE)
                        continue;
                break;
        case '5':
                if (InitCANWakeupFunction() == FALSE)
                        continue;
                break;
        case '0':
                return;
        default :
                continue;
}
……
DrvSYS_SetPllSrc(E_DRVSYS_EXT_12M);
DrvSYS_SetPLLPowerDown(0);
DrvSYS_SetHCLKSource(2);
Delay(1000);
FMC->FATCON.FPSEN = 1;

if (u8Type == 0)
{
        /* For IDLE Mode */
        DrvSYS_SetPowerDownWaitCPU(0);

        DrvSYS_EnablePowerDown(0);

        /* wait for interrupt, enter in IDLE Mode */
        Delay(1000);
        __WFI();
}else
if (u8Type == 1)
{
        /* For PowerDown Mode */
        SCB->SCR = 4;
        DrvSYS_SetPowerDownWaitCPU(1);

        /* Disable crystal to enter power down */
        DrvSYS_EnablePowerDown(1);

        /* wait for interrupt, enter in Power Down mode */
        Delay(1000);
        __WFI();
}
……
```

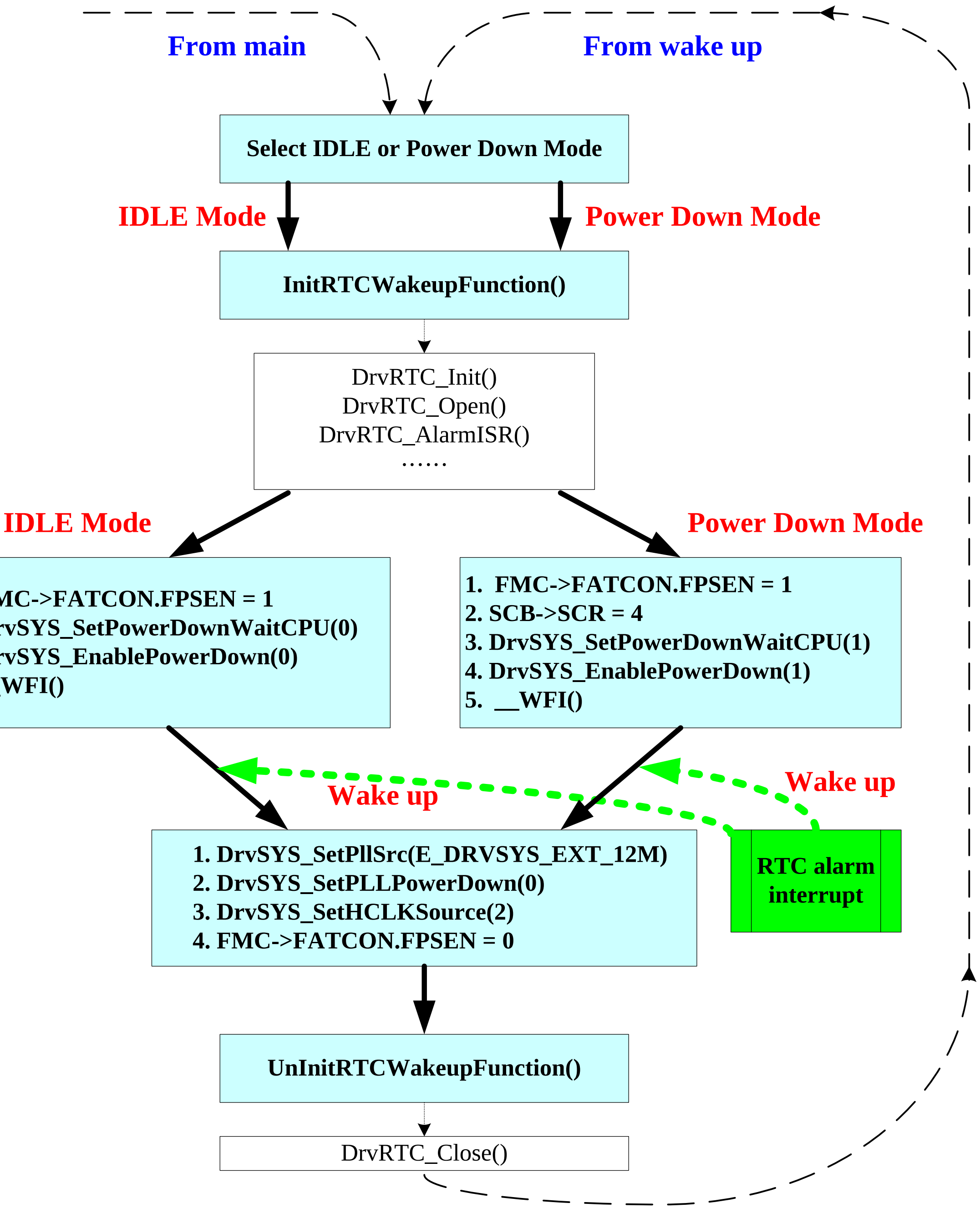
```
DrvSYS_SetPllSrc(E_DRVSYS_EXT_12M);
DrvSYS_SetPLLPowerDown(0);
DrvSYS_SetHCLKSource(2);
Delay(1000);
FMC->FATCON.FPSEN = 0;

switch (u8Item)
{
        case '1':
                UnInitRTCWakeupFunction();
                break;
        case '2':
                UnInitGPIOWakeupFunction();
                break;
        case '3':
                UnInitUARTWakeupFunction();
                break;
        case '4':
                UnInitUSBWakeupFunction();
                break;
        case '5':
                UnInitCANWakeupFunction();
                break;
        default :
                break;
}
……
```

3 CALLING SEQUENCE

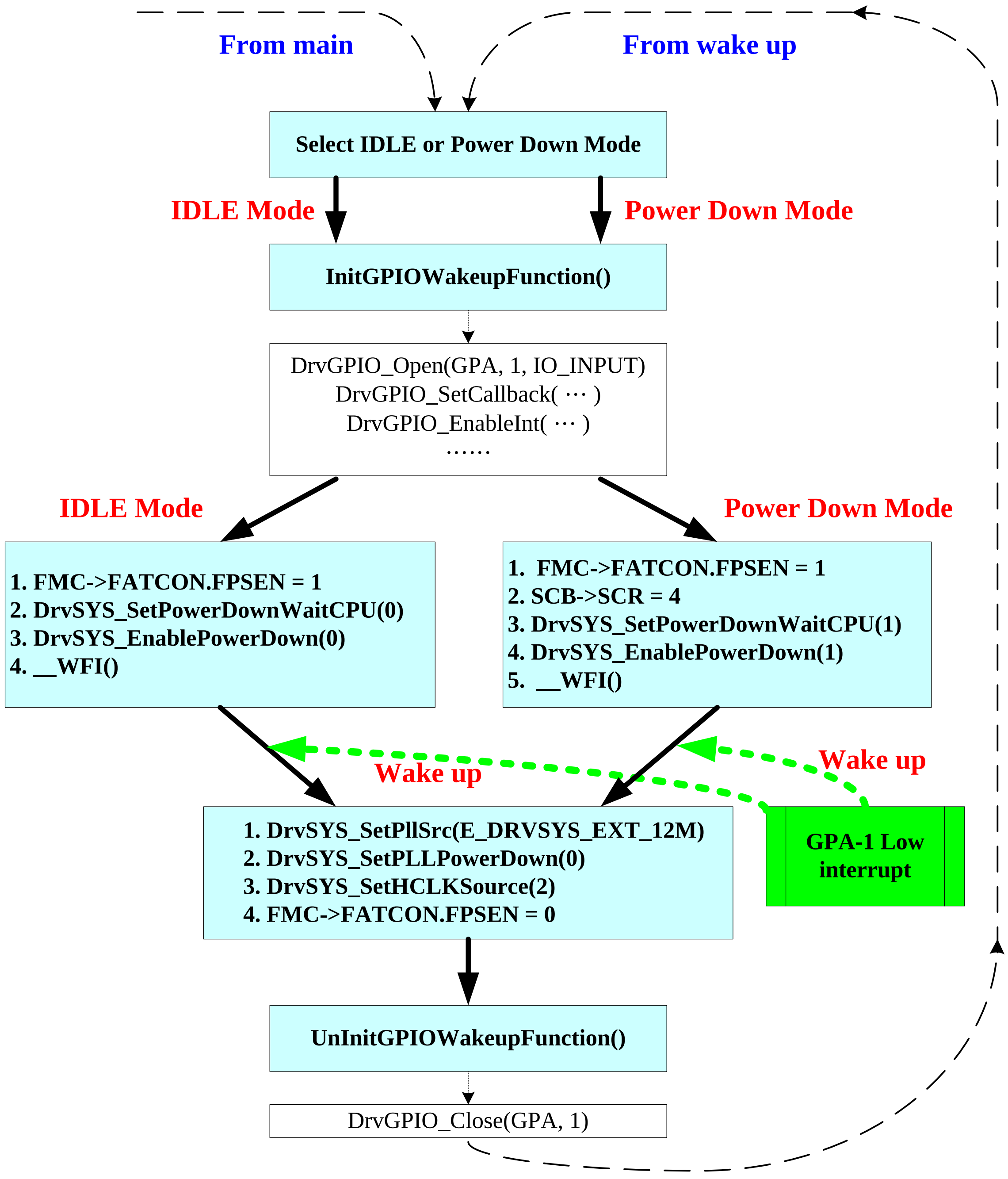
3.1 Enable/Disable chose IP clock


From main
Select an item to enable or disable the chose IP clock
Check the chose IP clock is already enable ?
Yes
If the chose IP is UART0 ?
Yes
Disable UART0 clock and hold CPU for measure the current
No
Enable the chose IP clock
No
Disable the chose IP clock
Enable UART0 clock

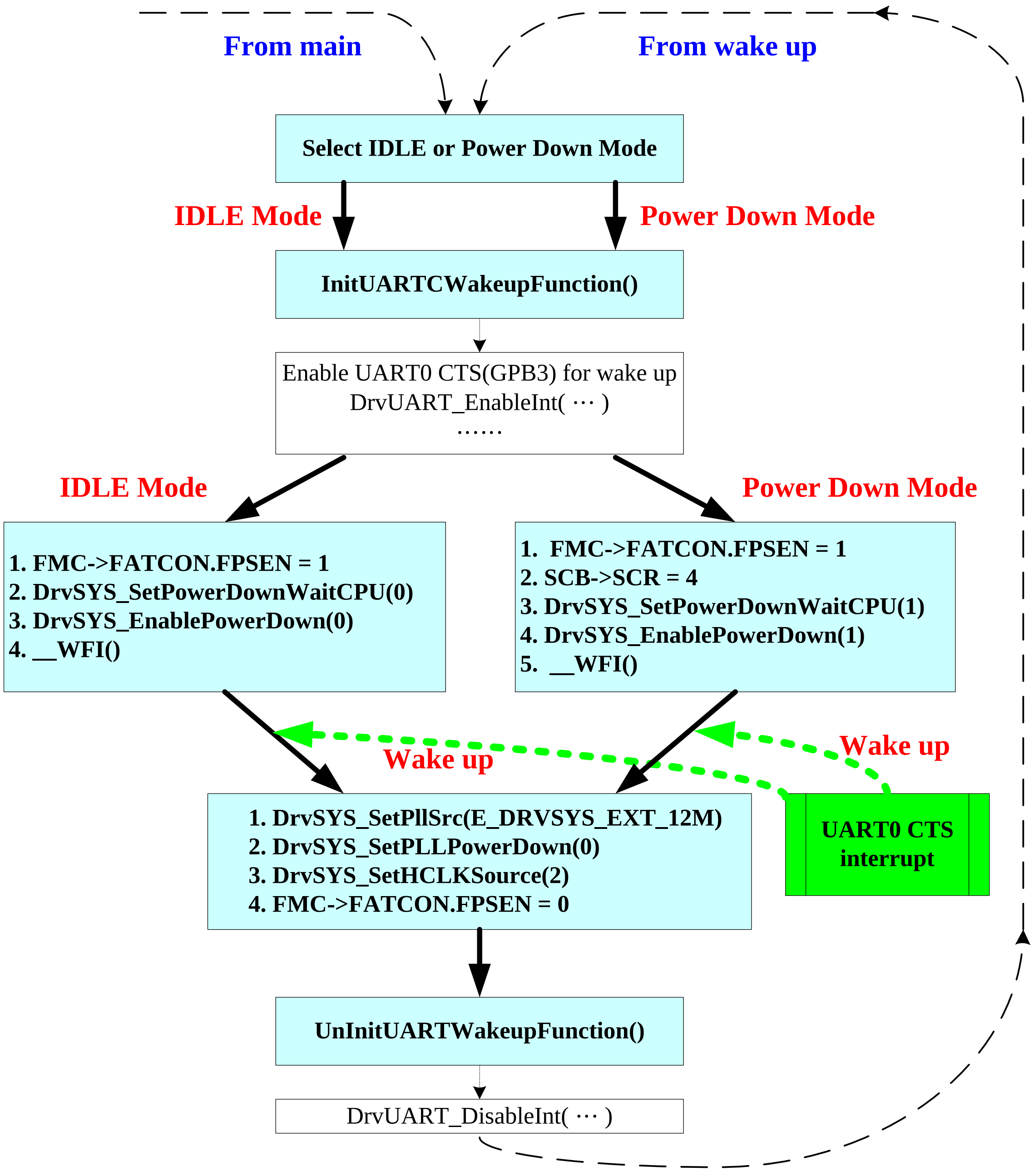
3.2 Enter in Power Down/IDLE mode and wakeup by RTC



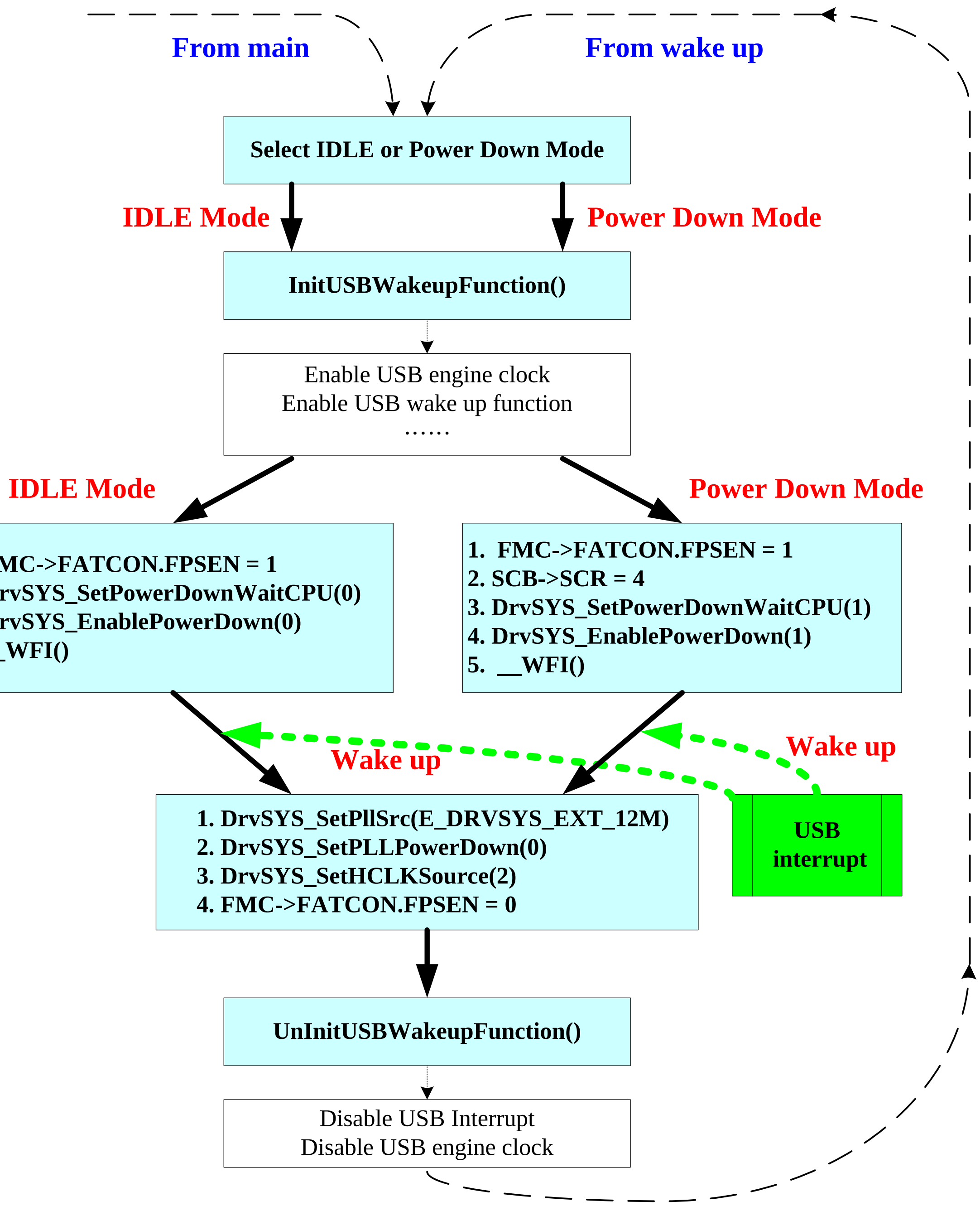
3.3 Enter in Power Down/IDLE mode and wakeup by GPIO



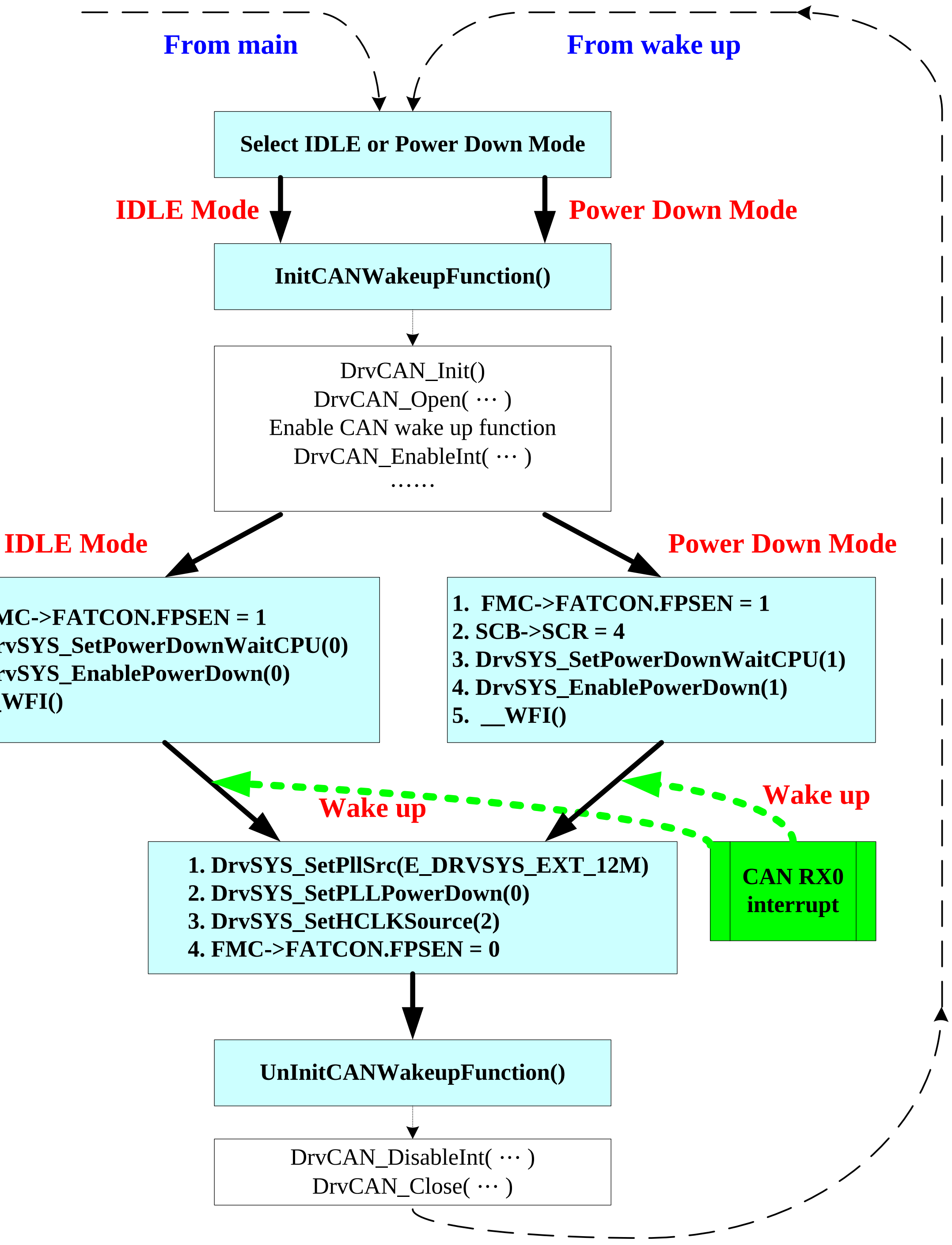
3.4 Enter in Power Down/IDLE mode and wakeup by UART


From main
From wake up
Select IDLE or Power Down Mode
IDLE Mode
Power Down Mode
InitUARTCWakeupFunction()
Enable UART0 CTS(GPB3) for wake up
DrvUART_EnableInt(⋯)
……
IDLE Mode
Power Down Mode
1. FMC->FATCON.FPSEN = 1
2. DrvSYS_SetPowerDownWaitCPU(0)
3. DrvSYS_EnablePowerDown(0)
4. __WFI()
1. FMC->FATCON.FPSEN = 1
2. SCB->SCR = 4
3. DrvSYS_SetPowerDownWaitCPU(1)
4. DrvSYS_EnablePowerDown(1)
5. __WFI()
Wake up
Wake up
1. DrvSYS_SetPllSrc(E_DRVSYS_EXT_12M)
2. DrvSYS_SetPLLPowerDown(0)
3. DrvSYS_SetHCLKSource(2)
4. FMC->FATCON.FPSEN = 0
UART0 CTS interrupt
UnInitUARTWakeupFunction()
DrvUART_DisableInt(⋯)

3.5 Enter in Power Down/IDLE mode and wakeup by USB



3.6 Enter in Power Down/IDLE mode and wakeup by CAN



4 EXECUTION ENVIRONMENT SETUP AND RESULT

4.1 Current consumption table

Use external crystal 12MHz and system clock runs up to 48MHz@3.3v

IP Enable/ Disable	Current Consumption(mA)
Watch Dog Timer	0.08
RTC	0.45
Timer 0	0.2
Timer 1	0.2
Timer 2	0.29
Timer 3	0.24
I2C 0	0.29
I2C 1	0.29
SPI 0	0.81
SPI 1	0.84
SPI 2	0.84
SPI 3	0.83
UART 0	0.94
UART 1	0.94
UART 2	x
PWM 10	0.23
PWM 32	0.24
PWM 54	x
PWM 76	x
CAN	1.11
USB	1.94
ADC	1.23
ACMP	0.17
PS2	0.59
PDMA	0.79
Flash ISP	0.4

4.2 Power Down/IDLE mode current and Wake up methods

Use external crystal 12MHz and system clock runs up to 48MHz@3.3v

System Mode	Current Consumption
IDLE mode	15.47 mA
Power Down mode	15.2 uA
Power Down mode and enable UART0 for show debug message	34.8 uA

4.2.1 RTC alarm wakeup

Setup new RTC date/time and alarm time for 10s then enter in Power Down/IDLE mode.

After 10s, system will exit Power Down/IDLE mode and enter in normal mode by RTC alarm interrupt.

4.2.2 GPIO interrupt wake up

Set GPIOA-1 as input mode(pin status is HIGH) and interrupt falling trigger, then enter in Power Down /IDLE mode.

When toggle GPIOA-1 to LOW, system will exit Power Down/IDLE mode and enter in normal mode by GPIO interrupt.

4.2.3 UART CTS wake up

Enable UART0 wake up function and configure GPB3 as CTS0 then enter in Power Down/IDLE mode.

As toggle the CTS0(GPB3), system will exit Power Down/IDLE mode and enter in normal mode.

4.2.4 USB plugged-in wake up

Enable USB engine clock and wake up function the enter in Power Down/IDLE mode.

When USB cable plugged-in, system will exit Power Down/IDLE mode and enter in normal mode.

4.2.5 CAN RX wake up

Enable relative CAN function for CAN wake up and GPIOD-6 as input mode for CAN_RX pin then enter in Power Down/IDLE mode.

When toggle GPIOD-6(CAN_RX) to LOW, system will exit Power Down/IDLE mode then enter in normal mode.

4.2.6 Time period of wake up

System spent about 1.11ms to exit Power Down mode when use external 12MHz crystal. That included 12MHz crystal running time and NUC1xx internal 4096 clock cycle delay to wait crystal stable. And spent about 680ns from IDLE mode to normal mode when wake up event occurred.

5 REVISION HISTORY

REV.	DATE	DESCRIPTION
0.01	March 09, 2010	1. Initially issued.
0.02	April 21, 2010	1. Update wake up time.
0.03	May 20, 2010	1. Update wake up time for IDLE mode

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.