Application Note, V1.1, Jul. 2007

AP08056

XC800 Family

Power Saving Features - Achieving Low Power Requirements on Applications

Microcontrollers

Never stop thinking

Edition 2007-07

Published by
Infineon Technologies AG
81726 München, Germany

XC800 Family

Revision History: V1.1, 2007-07

Previous Version(s): V1.0

Section	Subjects (major changes since last revision)
Section 3	Included Section 3.1 to highlight the lower limit of slow-down frequency that guarantees flash programming and erasing operations in slow-down mode.
Section 4	Error in the WKSEL value in Table 8. It has been swapped.
Section 6	Section 6.1 and Section 6.2 updated with the main limitations of PLL-Base mode and Prescaler mode. Example code in Section 6.2.1 to restore PLL mode from prescaler mode included an instruction to open SCU page 1.

1 Overview

This application note describes the various power saving features in the XC800 architecture. It is intended to provide users of the XC866 and XC886/888 with information and guidelines on how to apply these features to achieve low power consumption on applications via a combination of techniques which include:

- Stopping the CPU clock (Idle Mode)
- Reducing clock speed (slow-down Mode)
- Power down of the entire system (Power-Down Mode)
- Stopping the clocks of individual system components (Peripheral Management)
- Operating on a lower clock frequency

In this document, users can find measurements of the current consumed by the XC866 and XC886/888 devices operating in different power modes. There is also a breakdown of the current consumed by the peripherals at different clock speeds. With these measurements, the users will have a better understanding of how the different power saving features can affect the power consumption of the devices and therefore apply the power saving features effectively. However, these results were meant to serve as a reference only as the measurements will vary from device to device. The power consumption can be affected by various factors such as the type of instructions used in the code, program flow, peripheral settings, measuring equipment, power supply, temperature and frequency of the oscillator etc.

The measurements in this document were taken from four different devices, XC866-1FR, XC866-4FR, XC886 and XC888. The XC866-4FR results can be used as a reference for XC866-2FR.

Unless otherwise stated, all the results were measured under the following conditions:

1. Power supply of 5 V
2. Room temperature of approximately 25 ^{o}C
3. Port 2 of the devices were set to output to disable the analog driver
4. All the other port pins were configured with internal pull-down enabled
5. Peripherals were not programmed to perform any operations

2 Stopping the CPU Clock (Idle Mode)

The microcontroller can reduce power consumption by stopping the CPU's clock. This can be achieved by putting the device in idle mode. In this mode, the oscillator continues to run, but the CPU is stopped with its clock disabled. Peripherals whose input clocks are not disabled are still functional.

Note: If the watchdog timer (WDT) is still active when the device goes into idle mode, it will generate an internal reset when an overflow occurs. It is therefore necessary to disable the WDT before entering idle mode.

The CPU status is preserved in its entirety; the stack pointer, program counter, program status word, accumulator, and all other registers maintain their data during idle mode. The port pins hold the logical state they had at the time the idle mode was activated.

2.1 Entering and Exiting Idle Mode

Idle mode can be entered by setting the bit PCON.IDLE.

```
PCON |= 0x01;
```

Table 1 look at the savings in current consumption between active mode and idle mode of the different devices.

Table 1 Active and Idle mode current measurements

Products	CPU Clock (MHz)	Current (mA)		
		Active [1)]	Idle	Savings
XC866-1FR	26.8	15.50	11.87	3.63
XC866-4FR	26.6	17.84	14.16	3.72
XC886	24.4	23.46	20.02	3.44
XC888	24.1	23.22	19.82	3.40

1) Program waits in a continuos loop. i.e. while(1);.

Consider a program that is constantly waiting to service a Timer interrupt. The program can:

a) wait in an endless loop for the interrupt event to occur, or
b) wait for the interrupt event to occur while the CPU is disabled (idle mode) and then enable the CPU to service the interrupt routine when an interrupt occurs.

Method (b) will consume less power as indicated in **Table 1** and **Table 16**.

The device in idle mode can be exited to active mode on the occurrence of any of the two conditions:

1. Hardware reset. The device is put into the hardware reset state.
2. An interrupt has occurred from an enabled interrupt source. The device will service the interrupt routine and continue its operation from the next instruction after the instruction that sets the PCON.IDLE bit to 1.

2.2 Ineffective Idle Mode

Idle mode may not always be useful in reducing power consumption. If interrupt events keep occurring, the device will not be able to stay in idle mode and therefore is unable to reduce power consumption. In the below example, the peripherals of the devices were configured to operate as follows:

- Timers 0/1/2/21 in autoreload mode to overflow after 0xFFFF counts
- ADC to perform 10-bit parallel conversion in autoscan mode
- CCU to generate PWM output upon every period match
- UART0 and UART1 to transmit data at 19.2 kBaud upon each UART interrupt
- SSC to transmit at 3 MBaud upon each SSC interrupt
- CORDIC to start computation in circular rotation mode upon each CORDIC interrupt
- MDU to perform unsigned multiplication upon each MDU interrupt
- MCAN to start 8 bytes transmission in loopback mode upon each MCAN interrupt

Note: UART1, CORDIC, MDU and MCAN were configured for XC886/888 only.

The current consumption of the devices is as follow:

Table 2 Current consumed by devices with peripheral activities

Product	CPU Clock (MHz)	Current (mA)		
		Active	Idle	Savings[1]
XC866-1FR	26.8	15.88	15.89	- 0.01
XC866-4FR	26.6	18.52	18.53	- 0.01
XC886	24.4	25.01	25.07	-0.06
XC888	24.1	24.82	24.89	- 0.07

1) Negative value indicates that Idle mode current > Active mode current.

Looking at the amount of actives, the devices will be too busy serving the interrupt routines that they may never be able to stay in idle mode (i.e. turn off the CPU). It can be observed from **Table 2** that idle mode has no effect on the current consumption since the CPU is always active.

3 Reducing Clock Speed (Slow-Down Mode)

The slow-down mode is used to reduce the power consumption by dividing the CPU clock and the peripherals clock with a programmable factor. The slow-down mode is activated by setting the bit PMCON0.SD after the programmable factor in the bitfield CMCON.CLKREL has been selected. The programmable factors may differ for different XC800 products.

The slow-down mode is terminated by clearing bit PMCON0.SD, which is a protected bit. It cannot be written directly when the protection scheme is activated. An example below shows a software example of a XC866 device entering slow-down mode with system clock reduced by a factor of 32.

```
SYSCON0 &= 0xFE;//Access standard SFR region (RMAP = 0)
SCU_PAGE = 0x01;//Open SCU page 1
CMCON &= 0xF0;  //Clear CLKREL
CMCON |= 0x06;  //Select clock divider = fsys/32
PASSWD = 0x98;  //Open access to writing of all protected bits
PMCON0 |= 0x08; //Set SD to enter slow-down mode
PASSWD = 0xA8;  //Close access to writing of all protected bits
```

Similarly, to exit the slow-down mode,

```
SYSCON0 &= 0xFE;//Access standard SFR region (RMAP = 0)
SCU_PAGE = 0x01;//Open SCU page 1
PASSWD = 0x98;  //Open access to writing of all protected bits
PMCON0 &= 0xF7; //Clear SD to exit slow-down mode
PASSWD = 0xA8;  //Close access to writing of all protected bits
```

3.1 Flash Programming and Erasing in Slow-Down Mode

In slow-down mode, flash programming and erasing operations are guaranteed only if the Clock Divider (CLKREL) value does not exceed the slow-down factor listed in Table 3. If application runs at a lower frequency, user should exit the slow-down mode before performing any flash programming or erasing operations. Flash reading can be performed at any frequency, even below the described limit.

Table 3 Maximum Clock Divider Value to Guarantee Flash Programming or Erasing Operations

Product	System Frequency	CLKREL	CPU Clock Frequency in Slow-Down Mode
XC866	80 MHz	0100_B	1.7 MHz 1)
XC886/888	96 MHz	1000_B	1.5 MHz 2)

1) System frequency of 80 MHz divided by the slow-down factor of 16 and then a fixed divider of 3
2) System frequency of 96 MHz divided by the slow-down factor of 32 and then a fixed divider of 2

3.2 Combining slow-down Mode with Idle Mode

The slow-down mode can be combined with the idle mode (slow-down-idle mode). This can be done by performing the following sequence:

1. Select the slow-down frequency in the bitfield CMCON.CLKREL.
2. Enter the slow-down mode by setting the bit PMCON0.SD.
3. Activate idle mode by setting the PCON.IDLE mode.

The slow-down-idle mode can be terminated by first exiting from idle mode and then followed by clearing the bit PMCON0.SD. It will also be terminated by a hardware reset.

3.3 Slow-Down Mode vs. Slow-Down-Idle Mode Current Measurements

This section illustrates the current measurements taken from XC866-1FR, XC866-4FR, XC886 and XC888, and shows the effect of the slow-down mode and slow-down-idle mode. The measurements are tabulated in **Table 4**, **Table 5**, **Table 6** and **Table 7** respectively.

From the tables, it can be observed that the amount of current that can be saved in slow-down mode reduces with the peripheral frequency. This can be seen in the graphical representations of each table. The peripherals require less power to operate when the peripheral clock is reduced in slow-down mode. At low frequencies, the current drawn by the peripherals become relatively constant and therefore less current can be saved when the peripheral clock speed is further reduced. In any case, the devices draw the least current at the lowest possible frequency.

The slow-down-idle mode does not necessary draw the least current at the lowest possible frequency. This can be observed in **Table 4**, **Table 5**, **Table 17** (XC866) and **Table 18** (XC888). The effectiveness of the slow-down-idle mode will depend on factors such as peripheral activities and the operating frequency. It is therefore necessary to perform measurements manually to determine whether slow-down-idle mode is more effective than slow-down mode in reducing current consumption.

Table 4 Current comparison between slow-down mode and slow-down-idle mode of a XC866-1FR device

f_{SYS} = 80.4 MHz		Current (mA)		
Peripheral [1] Frequency (MHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference [2] X - Y**
26.8	0000_B	15.50	11.87	3.63
6.7	0010_B	7.12	6.41	0.71
3.4	0011_B	5.70	5.47	0.23
1.7	0100_B	4.99	5.01	0.02
0.42	0110_B	4.45	4.66	-0.21
0.21	0111_B	4.36	4.36	-0.00
0.11	1000_B	4.32	4.57	-0.25
0.052	1001_B	4.30	4.56	-0.26
0.026	1010_B	4.28	4.55	-0.27
0.013	1011_B	4.28	4.54	-0.26

1) Peripheral frequency = f_{SYS}/CLKREL/Fixed divider of 3 (refer to user's manual on Clock System)

2) Negative value indicates that slow-down-idle mode current > slow-down mode current.

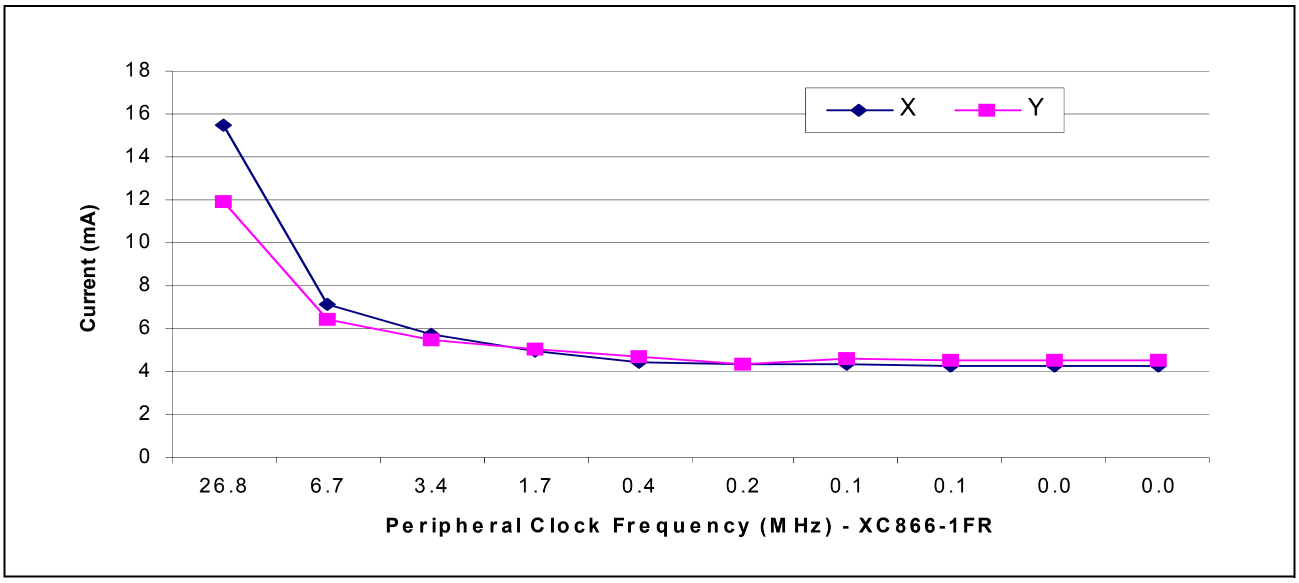

Figure 1 Graphical representation of the X and Y series in Table 4

Table 5 Current comparison between slow-down mode and slow-down-idle mode of a XC866-4FR device

f_{SYS} = 79.8 MHz		Current (mA)		
Peripheral [1] Frequency (MHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference [2] X - Y**
26.6	0000_B	17.84	14.16	3.68
6.7	0010_B	8.62	7.70	0.92
1.7	0100_B	6.19	6.02	0.17
0.83	0101_B	5.78	5.75	0.03
0.42	0110_B	5.58	5.60	-0.02
0.21	0111_B	5.48	5.53	-0.05
0.10	1000_B	5.43	5.50	-0.07
0.052	1001_B	5.40	5.48	-0.08
0.026	1010_B	5.39	5.47	-0.08
0.013	1011_B	5.38	5.47	-0.09

1) Peripheral frequency = f_{SYS}/CLKREL/Fixed divider of 3 (refer to user's manual on Clock System)

2) Negative value indicates that slow-down-idle mode current > slow-down mode current.

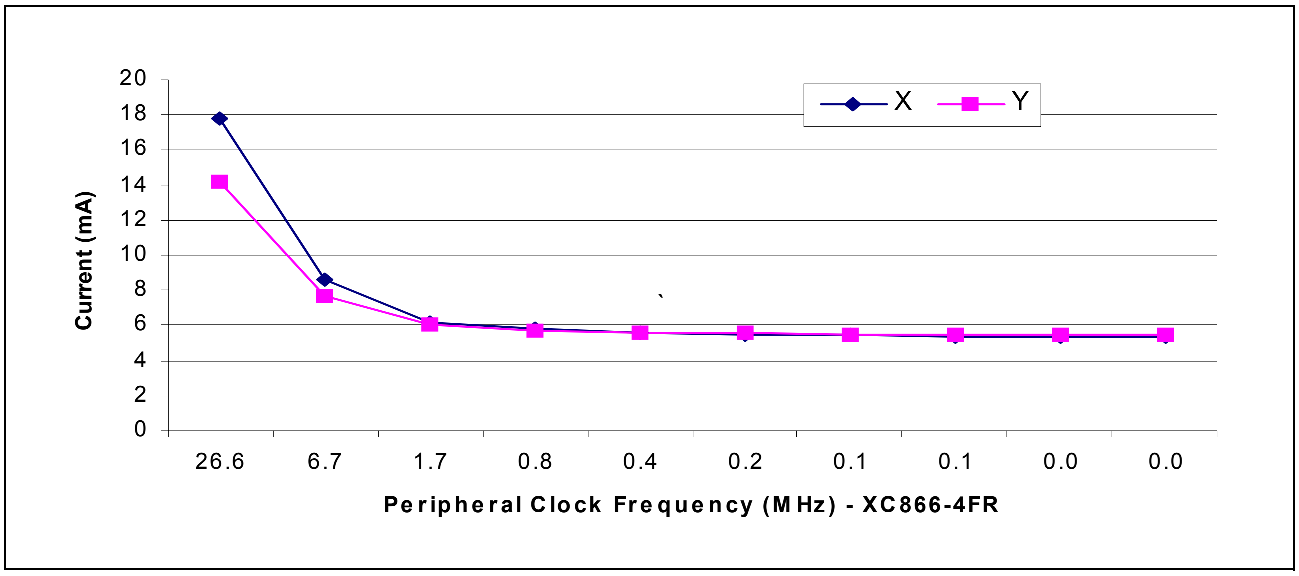

Figure 2 Graphical representation of the X and Y series in Table 5

Table 6 **Current comparison between slow-down mode and slow-down-idle mode of a XC886 device**

f_{SYS} = 97.5 MHz		**Current (mA)**		
Peripheral [1) Frequency (MHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference X - Y**
24.4	0000_B	23.46	20.02	3.44
12.2	0010_B	14.87	13.05	1.82
6.1	0100_B	10.43	9.45	0.98
3.0	0110_B	8.19	7.65	0.54
1.5	1000_B	7.07	6.74	0.33
0.76	1010_B	6.50	6.29	0.21
0.38	1100_B	6.22	6.06	0.16
0.19	1110_B	6.08	5.95	0.13
0.13	1111_B	6.03	5.91	0.12

1) Peripheral frequency = f_{SYS}/CLKREL/Fixed divider of 2 (refer to user's manual on Clock System)

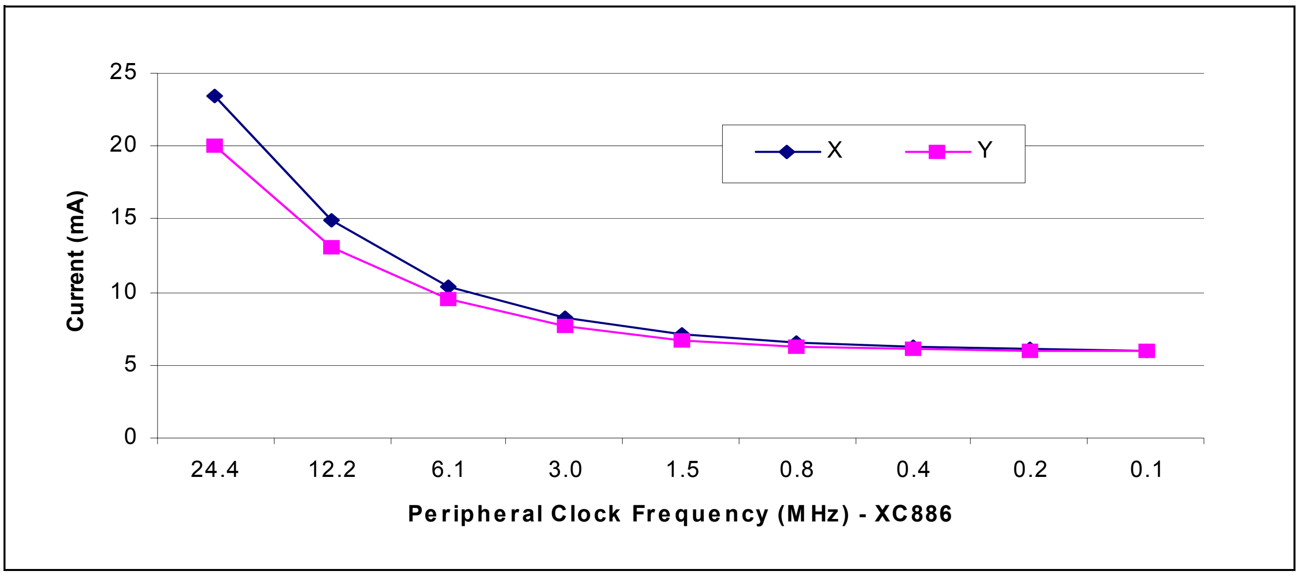

Figure 3 **Graphical representation of the X and Y series in Table 6**

Table 7 Current comparison between slow-down mode and slow-down-idle mode of a XC888 device

f_{SYS} = 96.5 MHz		Current (mA)		
Peripheral [1)] Frequency (MHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference X - Y**
24.1	0000_B	23.22	19.82	3.40
12.1	0010_B	14.78	12.91	1.87
6.0	0100_B	10.33	9.36	0.97
3.0	0110_B	8.12	7.58	0.54
1.5	1000_B	7.01	6.68	0.33
0.75	1010_B	6.45	6.24	0.21
0.38	1100_B	6.17	6.01	0.16
0.19	1110_B	6.03	5.90	0.13
0.13	1111_B	5.98	5.86	0.12

1) Peripheral frequency = f_{SYS}/CLKREL/Fixed divider of 2 (refer to user's manual on Clock System)

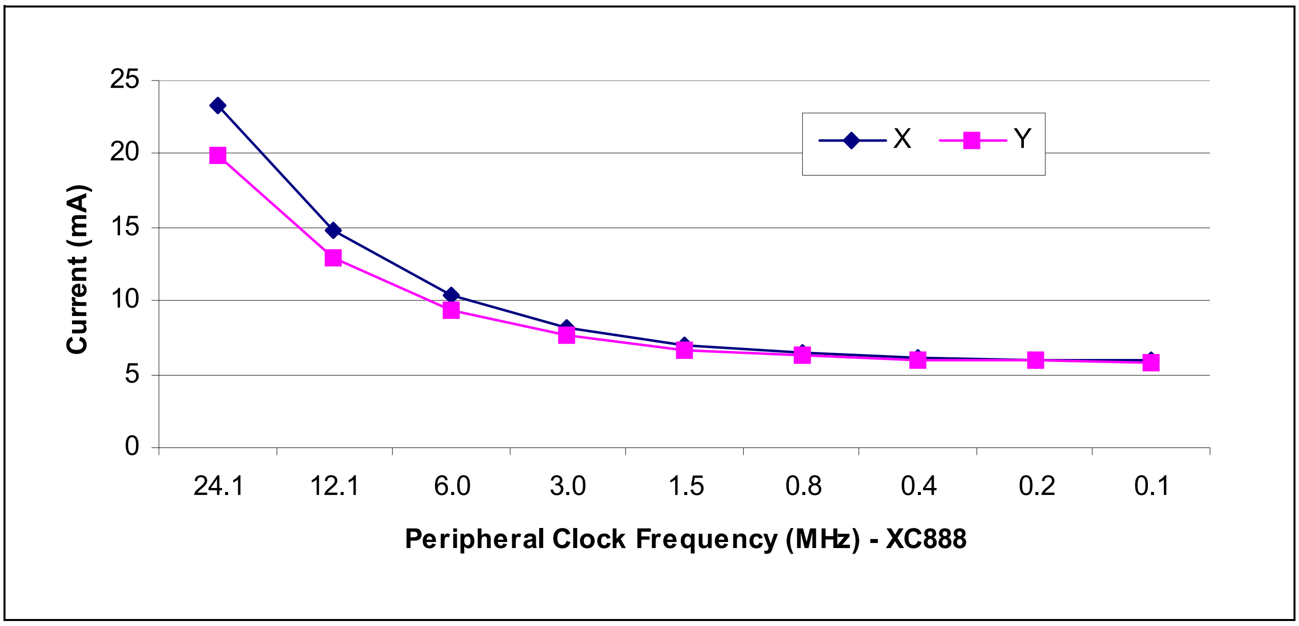

Figure 4 Graphical representation of the X and Y series in Table 7

4 Power Down of the Entire System (Power-Down Mode)

Power-down of the entire system can be achieved by setting the device in power-down mode. In this mode, all functions of the microcontroller are stopped and only the contents of the FLASH, on-chip RAM, XRAM and the SFRs are maintained. The port pins hold the logical state they had when the power-down mode was activated. For the digital ports, the user should ensure that the ports are not left floating. Floating input port pins should enable pull-down or set as output. Output pins that are pulled-up should be set to high. For the analog Port 2, the user should disable the input driver of the port by setting the port direction to output.

In power-down mode, the clock is turned off. Hence, it cannot be awakened by an interrupt or by the WDT. It is awakened only when it receives an external wake-up signal or reset signal.

4.1 Entering Power-Down Mode

Software requests power-down mode by setting the bit PMCON0.PD (protected bit) to 1. Two NOP instructions must be inserted after the bit PMCON0.PD is set to 1. The device will not enter power-down mode immediately after executing the instruction to set PMCON0.PD and therefore the two NOPs is to ensure the first instruction (after two NOP instructions) is executed correctly after wake-up from power-down mode.

If the external wake-up from power-down is used, software must select the EXINT0 pin, the RXD pin, or either of these two pins as the wake-up source by selecting the WS bit of the PMCON0 register. Exit from power-down mode can be achieved when a falling edge trigger is detected at the selected source(s). Bit MODPISEL.URRIS is used to select one of the two RXD inputs and bit MODPISEL.EXINT0IS is used to select one of the two EXINT0 inputs. The wake-up with reset or without reset is selected by bit PMCON0.WKSEL. The wake-up source and wake-up type must be selected before the system enters the power-down mode.

In order for the program to resume its operation from where it was stopped after the reset, it is necessary to save the contents of the relevant registers to the stack including the current stack pointer. The example below shows how the power-down mode can be entered correctly with the port status stored to stack.

```
//... SP was initialized as 0x41
PORT_PAGE = 0x00;//Open port page 0 to access Px_DATA
_push_(P3_DATA);//Save port 3, current SP = 0x42.
                //_push_() function is defined in intrins.h
```

```
STACKSP = SP;    //Save current stack pointer at an absolute memory
                 //location. e.g. unsigned char idata STACKSP _at_
                 //0xe0; iram location 0xe0 = 0x42
SCU_PAGE = 0x01;//Open SCU page 1 to access PMCON0 and PASSWD
PASSWD = 0x98;   //Open access to writing of all protected bits
PMCON0 |= 0x17;  //Enable power-down with wake up source RXD or
                 //EXINT0 selected and select wake-up with reset
_nop_();         //2 NOPs necessary for proper execution of
_nop_();         //code upon wakeup. Function is defined in
                 //intrins.h.
```

Note: Some IRAM locations will be corrupted after a reset. The stack and the constant variable location must avoid these locations. Please refer to the respective product's Errata Sheet for the detailed locations.

Note: START_XC.A51 will clear IRAM locations ranging from 0 to the number defined in the symbol IDATALEN. It is therefore necessary to redefine the data assigned to this symbol to avoid destroying the stack and the constant variable data.

4.2 Exiting Power-Down Mode

Power-down mode can be exited in two ways:

1. Hardware reset. The device is put into the hardware reset state.
2. The EXINT0 pin, the RXD pin or either of these two pins detects a falling edge. The wake-up source(s) must be selected prior to entering power-down mode.

When a wake up source is selected, the device will wake-up from power-down mode upon detecting a falling edge trigger at that source. In power-down mode, EXINT0 pin/RXD pin must be held at high level. Power-down mode is exited when EXINT0 pin/RXD pin goes low for at least 100 ns.

The device can wake-up with or without reset depending on the setting of the PMCON0.WKSEL bit. If the WKSEL bit was set to 1 prior to entering power-down mode, the system will execute a reset sequence similar to the power-on reset sequence. Therefore, all port pins and SFRs are put into their reset state and will remain in this state until they are affected by program execution. The only exception is that hardware will set the wake-up indication bit PMCON0.WKRS to indicate that the device is woken up from power-down reset.

If bit WKSEL was cleared to 0 before entering power-down mode, a fast wake-up sequence is used. The port pins continue to hold their state which was valid during power-down mode until they are affected by program execution.

The example below shows how the previous example can restore the port 3 status after exiting the power-down mode with reset.

```
//... after initialization
SCU_PAGE = 0x01;//Open SCU page 1 to access PMCON0
if (PMCON0 & 0x20)//Check PMCON0.WKRS for wakeup indication
{
     SP = STACKSP;//Restore stack pointer, SP = 0x42
     PORT_PAGE = 0x00; //Open port page 0 to access Px_DATA
     _pop_(P3_DATA); //Restore P3 from stack
}
//... continue...
```

Table 8 illustrates the timing for a device to wake-up from power-down mode to active mode for both wake-up with/without reset.

Table 8 Wakeup timing from power-down mode

	XC866-1FR		XC866-4FR and XC886/888	
Reset Sequence	**WKSEL = 0**	**WKSEL = 1**	**WKSEL = 0**	**WKSEL = 1**
EVR is stable	150 µs (Typical)		150 µs (Typical)	
PLL is locked	200 µs (Maximum)		200 µs (Maximum)	
Flash ready to read state	0	160 µs	0	160 µs
BROM execution[1)]	0	215 µs (Typical)	0	250 µs (Typical)
Total wake-up time	350 µs	725 µs	350 µs	760 µs

1) Timing is based on the system frequency of the devices. XC866 operates at 80 MHz while XC886/888 operates at 96 MHz.

In power-down mode, the typical amount of current drawn by the devices will be approximately 1 µA at 25 °C.

5 Stopping the Clocks of Individual System Components (Peripheral Management)

Peripherals which are not required for a particular functionality can be disabled by programming the assigned register bits in PMCON1 and PMCON2 which would gate off clock inputs. This further reduces overall power consumption of the device.

Table 9, Table 10 and Table 11 shows the amount of current that can be saved when each/all of the peripheral module(s) are disabled in a XC866-1FR, XC866-4FR and a XC886 device at different peripheral clock frequency (f_{PCLK}). It can be observed that when the peripheral clock is reduced, the power consumed by the peripherals will also reduce, which means that less current can be saved when the peripheral is disabled.

Software example to disable all peripherals on a XC886/888 device:

```
SCU_PAGE = 0x01;     //Open SCU page 1 to access PMCON1
PMCON1 |= 0x7F;      //Disable ADC, CCU, SSC, T2, MDU, CAN and
                     //CORDIC
SCU_PAGE = 0x03;     //Open SCU page 3 to access PMCON2
PMCON2 |= 0x03;      //Disable T21 and UART1
```

For XC866, only ADC, CCU, SSC and T2 can be disabled by setting the corresponding bits in PMCON1.

Table 9 Current consumption by the peripherals of a XC866-1FR device at different peripheral clock (PCLK) frequencies

	Current (mA)				
Peripheral Frequency (MHz)	**26.8**	**6.7**	**0.84**	**0.21**	**0.013**
CLKREL	**0000_B**	**0010_B**	**0101_B**	**0111_B**	**1011_B**
ADC	0.80	0.20	0.03	0.01	0[1]
CCU	1.45	0.36	0.05	0.02	0
SSC	0.51	0.12	0.02	0.01	0
T2	0.16	0.04	0.01	0	0
All	2.95	0.73	0.10	0.03	0

1) < 10 µA

Table 10 Current consumption by the peripherals of a XC866-4FR device at different peripheral clock (PCLK) frequencies

	Current (mA)				
Peripheral Frequency (MHz)	**26.6**	**6.7**	**0.83**	**0.21**	**0.013**
CLKREL	**0000_B**	**0010_B**	**0101_B**	**0111_B**	**1011_B**
ADC	0.95	0.26	0.04	0.01	0[1)]
CCU	1.44	0.39	0.05	0.02	0
SSC	0.37	0.10	0.02	0.01	0
T2	0.17	0.05	0.01	0	0
All	3.01	0.80	0.11	0.03	0

1) < 10 µA

Table 11 Current consumption by the peripherals of a XC888 device at different peripheral clock (PCLK) frequencies

	Current (mA)				
Peripheral Frequency (MHz)	**24.1**	**12.1**	**3.0**	**0.75**	**0.13**
CLKREL	**0000_B**	**0010_B**	**0110_B**	**1010_B**	**1111_B**
ADC	0.65	0.35	0.09	0.03	0[1)]
CCU	1.14	0.60	0.15	0.04	0.01
SSC	0.38	0.19	0.05	0.01	0
T2	0.13	0.07	0.02	0.01	0
T21	0.13	0.07	0.02	0.01	0
CAN	4.81	2.51	0.64	0.16	0.03
CORDIC	0.60	0.31	0.08	0.02	0
MDU	0.24	0.14	0.04	0.01	0
UART1	0.25	0.13	0.04	0.01	0
All	8.54	4.42	1.12	0.28	0.04

1) < 10 µA

6 Operating on a Lower Clock Frequency

The normal system clock runs in PLL Mode. The system frequency (f_{SYS}) is defined as:

$$f_{SYS} = \frac{N}{P \cdot K} \cdot f_{OSC}$$

The N factor is selected by the bitfield PLL_CON.NDIV. The P factor is fixed to 1. The K factor is fixed at 2 for XC866. It can be selected by the bit CMCON.KDIV (protected bit) for XC886/888. The internal oscillator frequency (f_{OSC}) for XC866 is 10 MHz. The internal oscillator frequency for XC886/888 is 9.6 MHz.

This system frequency is further divided to derive the CPU clock, peripheral clock and flash clock. By selecting a different clock mode, the system frequency will be changed and thus affecting all the other clocks of the entire system. The PLL Base Mode and the Prescaler Mode are two of the other clock modes that can be used to derive the system frequency.

It is important to take note that PLL-Base mode and Prescaler are not the normal recommended mode of operations and have their limitations. However, these two modes may offer potential power-saving capabilities for applications where the limitations are not critical. More details of these two clock modes and its limitations will be discussed in the following sections.

6.1 Device Clock in PLL Base Mode

In the PLL base mode, the oscillator is disconnected from the PLL and the system frequency is derived from the VCO base (free running) frequency clock divided by the K factor:

$$f_{SYS} = \frac{1}{K} \cdot f_{VCObase}$$

The ranges of the VCO base frequency is shown in **Table 12**.

Table 12 VCO base ($f_{VCObase}$) and system frequency (f_{SYS}) ranges

Device	**VCOSEL**	**$f_{VCObase}$** Range (MHz)	**f_{SYS}** Range when K = 1 (MHz)	**f_{SYS}** Range when K = 2 (MHz)
XC866	0	10 - 80	-	5 - 40
	1	20 - 80	-	10 - 40
XC886/888	0	20 - 80	20 - 80	10 - 40
	1	10 - 80	10 - 80	5 - 40

The VCO base frequency of different devices will range between 10 MHz to 80 MHz depending on the VCOSEL setting.

The main limitations of the PLL Base Mode are as follow:

1. PLL Base Frequency may vary between devices

Users will need to manually check the system frequency of each device by outputting the device's system clock at the CLKOUT pin. The code example to output the system clock is as follows:

```
SCU_PAGE = 0x01;      //open SCU page 1 to access COCON
COCON = 0x37;         //select clock output frequency/10
PORT_PAGE = 0x02;     //open port page 2 to access Px_ALTSELx
P0_ALTSEL0 = 0x01;    //select clock out function for P0.0
P0_ALTSEL1 = 0x00;
PORT_PAGE = 0x00;     //open port page 0 to access Px_DIR
P0_DIR = 0x01;        //set P0.0 as output port
```

At the output of P0.0, the clock output frequency is half of the frequency that is chosen by the bitfield COCON.COREL. In the above example, the system clock output at P0.0 has been divided by a factor of 10 based on the setting of the bitfield COCON.COREL. The system frequency can hence be obtained by multiplying the measured frequency by a factor of 20.

2. VCO Base Frequency Deviation

The VCO base frequency that was determined could vary approximately up to +/- 10% depending on the stability of the power supply to the PLL (i.e. power supply from EVR) and the temperature.

3. Long Recovery Time

In order to restore the clock to PLL mode, the oscillator has to be re-locked to the PLL. This require quite a number of instruction cycles to carry out the task (see example in **Section 6.1.1**) and the recovery time will depend on the frequency of the CPU operating in PLL-Base mode. The slow recovery time may render it unsuitable for applications where events require a fast response time.

4. Flash Programming and Erasing Operations are not Supported

The flash clocks will not meet the required frequency to guarantee flash programming and erasing operations when the device operates in PLL Base mode. As such, user has to restore the clock to PLL mode before flash programming and erasing operations can be carried out.

Table 13 are some results showing the current consumption by a XC866-1FR device, a XC866-4FR device and a XC886 device operating in PLL base mode.

Table 13 Current consumption in PLL base mode

	XC866-1FR		**XC866-4FR**		**XC886**	
VCOSEL	**0**	**1**	**0**	**1**	**0**	**1**
$f_{VCObase}$ (MHz)	22.5	38.9	22.0	37.7	36.2	21.2
f_{SYS} (MHz) with K = 2	11.24	19.47	11.00	18.85	18.11	10.62
PLL base mode current (mA)	2.74	4.25	3.07	4.75	5.12	3.37
PLL base mode [1] current with slow-down mode enabled (mA)	1.01	1.35	1.26	1.65	1.60	1.23

1) CLKREL = 1011_B for XC886 and CLKREL = 1111_B for XC886

6.1.1 Software Example for PLL Base Mode

In order to disconnect the oscillator, the VCO bypass mode must be selected first by setting the bit PLL_CON.VCOBYP. This is to prevent the system from detecting a PLL loss of lock condition which sets the NMICON.NMIPLL bit. The internal oscillator can be powered down in PLL base mode to save power by setting bit OSC_CON.OSCPD.

The following example shows how the PLL base mode is selected.

```
SCU_PAGE = 0x01;//Open SCU page 1
PLL_CON |= 0x08;//Select VCO Bypass Mode. This step is to
                //prevent PLL loss of lock bit from being set
PLL_CON |= 0x04;//Disconnect the oscillator
OSC_CON |= 0x10;//Power-down the oscillator
//... device is running in PLL Base Mode
```

The next example will show how to restore the clock in PLL mode from PLL base mode.

```
//... PLL_CON.VCOBYP = 1
SCU_PAGE = 0x01;//Open SCU page 1
OSC_CON &= 0xEF;//Power up the oscillator
OSC_CON |= 0x02;//Reset and restart oscillator detection logic
while(!(OSC_CON & 0x01));//Wait for oscillator to run
PLL_CON &= 0xFB;//Connect oscillator to PLL
PLL_CON |= 0x02;//Restart PLL lock detection
while(!(PLL_CON & 0x01));//Wait for PLL lock bit to set
PLL_CON &= 0xF7;//Deselect VCO bypass mode
//... device is running in PLL Mode
```

6.2 Device Clock in Prescaler Mode

In this mode, the system clock is derived from the oscillator clock divided by the P and K factor:

$$f_{SYS} = \frac{1}{P \cdot K} \cdot f_{OSC}$$

The oscillator frequency is close to its designated frequency (10 MHz for XC866 and 9.6 MHz for XC886/888) after it has been trimmed by the factory.

The main limitations of the Prescaler Mode are as follow:

1. No Oscillator Fail-Safe Mechanism

In Prescaler Mode, the clock is derived directly from the oscillator clock. If the oscillator clock breaks down during the operation, the device will stall. It will not be able to operate on the free running frequency as the PLL is bypassed.

2. Long Recovery Time

The recovery time will depend on the CPU speed. Although it requires less instruction cycles to recover to PLL mode as compared to PLL-Base mode, the significance still lies in the speed of the CPU operating in prescaler mode.

3. Flash Programming and Erasing Operations are not Supported

The flash clocks will not meet the required frequency to guarantee flash programming and erasing operations when the device operates in Prescaler mode. As such, user has to restore the clock to PLL mode before Flash programming and Erasing operations can be carried out.

Below are some results showing the current consumption by the four devices operating in prescaler mode.

Table 14 Current consumption in prescaler mode

	XC866-1FR	XC866-4-FR	XC886	XC888
Internal oscillator frequency (f_{osc})	10.5 MHz	10.0 MHz	9.8 MHz	9.7 MHz
System frequency (f_{sys}) [1)]	5.3 MHz	5.0 MHz	4.9 MHz	4.8 MHz
Active mode current	3.13 mA	3.53 mA	3.85 mA	3.84 mA
Slow-down mode current[2)]	2.28 mA	2.70 mA	2.89 mA	2.89 mA

1) $f_{sys} = 1/(P*K)*f_{osc}$ where P = 1 and K = 2

2) CLKREL = 1011_B for XC866 and CLKREL = 1111_B for XC886/888

The current consumption is much lower than the device operating in slow-down mode in **Section 3.3**. The combination of slow-down and idle mode may not be as efficient as slow-down alone at low frequencies. For example for the XC888 device, it will draw 3.35 mA of current for the combined slow-down and idle mode compared to 2.89 mA for slow-down mode alone when bitfield CMCON.CLKREL = 1111_B.

6.2.1 Software Example for Prescaler Mode

The system clock can operate in prescaler mode by setting the bit PLL_CON.VCOBYP. The following example shows how the prescaler mode is selected.

```
SCU_PAGE = 0x01;//Open SCU page 1
PLL_CON |= 0x08;//Select VCO Bypass Mode
```

PLL mode can be restored from prescaler mode simply by clearing the bit PLL_CON.VCOBYP.

```
SCU_PAGE = 0x01;//Open SCU page 1
PLL_CON &= 0xF7;//Deselect VCO bypass mode
```

7 Reset Current

This section describes the current consumption by the devices in reset state, i.e. when the reset pin is held low. All the port pins are left floating except the reset pin which is grounded.

Table 15 Current drawn by devices in reset state

Devices	Current (µA)
XC866-1FR	~ 750 to 900
XC866-4FR	~ 850 to 1000
XC886	~ 1050 - 1150
XC888	~ 1200 - 1350

8 A Timer 2 Example

In this section, the XC866-4FR device and the XC888 device were configured to start timer 2 in auto-reload mode which generates an overflow in every 20 ms. The overflow will trigger an interrupt service routine that will toggle port 3.0. The current drawn by the devices in active, idle, slow-down, slow-down-idle and prescaler mode were measured.

8.1 Active and Idle Mode Current

The current drawn by the devices in active mode (i.e. device wait in an endless loop for the interrupt event) and idle is shown in the table below. It can be observed that the savings ("Difference") that can be achieved by putting the device in idle mode is comparable to **Table 1**.

Table 16 Current consumed by devices when timer 2 is active

Product	CPU Clock (MHz)	Active (mA)	Idle (mA)	Difference (mA)
XC866-4FR	26.6	18.04	14.23	3.64
XC888	24.1	24.00	20.01	3.59

8.2 Slow-Down and Slow-Down-Idle Current

The program is now programmed to run in slow-down mode and slow-down-idle mode. For every different slow-down factor, the timer reload value was changed accordingly such that the timer will overflow in every 20 ms.

Table 17 Current consumed by XC866-4FR device when timer 2 is active

f_{SYS} = 79.8 MHz		Current (mA)		
Peripheral Frequency (MHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference [1)] X - Y**
26.6	0000_B	18.04	14.23	3.81
6.7	0010_B	8.69	7.76	0.93
1.7	0100_B	6.23	6.07	0.16
0.83	0101_B	5.82	5.80	0.02
0.42	0110_B	5.62	5.66	-0.04
0.21	0111_B	5.51	5.59	-0.08

Table 17 Current consumed by XC866-4FR device when timer 2 is active

f_{SYS} = 79.8 MHz		**Current (mA)**		
0.10	1000_B	5.46	5.55	-0.09
0.052	1001_B	5.43	5.53	-0.10
0.026	1010_B	5.42	5.52	-0.10
0.013	1011_B	5.41	5.51	-0.10

1) Negative value indicates that slow-down-idle mode current > slow-down mode current.

Table 18 Current consumed by XC888 device when timer 2 is active

f_{SYS} = 96.5 MHz		**Current (mA)**		
Peripheral Frequency (MHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference** [1)] **X - Y**
24.1	0000_B	24.00	20.41	3.59
12.1	0010_B	15.16	13.51	1.65
6.0	0100_B	10.57	9.95	0.62
3.0	0110_B	8.26	8.16	0.10
2.0	0111_B	7.50	7.57	-0.07
1.5	1000_B	7.11	7.26	-0.15
0.75	1010_B	6.52	6.81	-0.29
0.38	1100_B	6.23	6.59	-0.36
0.19	1110_B	6.09	6.47	-0.38
0.13	1111_B	6.04	6.43	-0.39

1) Negative value indicates that slow-down-idle mode current > slow-down mode current.

It can be noticed that by reducing the clock frequency of the CPU and peripheral, the devices will require less current to run.

8.3 Prescaler Mode Current

The program is now programmed to run in prescaler mode with slow-down mode and slow-down-idle mode. The timer reload value for every measurement was changed such that the timer will overflow in every 20 ms.

Table 19 Current consumed by XC866-4FR device when timer 2 is active

f_{SYS} = 5.0 MHz		Current (mA)		
Peripheral Frequency (KHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference X - Y**
1663.3	0000_B	3.58	3.40	0.18
831.7	0001_B	3.17	3.13	0.04
415.8	0010_B	2.95	2.98	-0.03
104.0	0100_B	2.78	2.87	-0.09
52.0	0101_B	2.76	2.85	-0.09
26.0	0110_B	2.74	2.84	-0.10
13.0	0111_B	2.73	2.83	-0.10

Table 20 Current consumed by XC888 device when timer 2 is active

f_{SYS} = 4.8 MHz		Current (mA)		
Peripheral Frequency (KHz)	**CLKREL**	**Slow-Down (X)**	**Slow-Down-Idle (Y)**	**Difference X - Y**
1206.3	0000_B	4.11	3.60	0.51
603.1	0010_B	3.55	3.23	0.32
301.6	0100_B	3.26	3.04	0.22
150.8	0110_B	3.12	2.95	0.17
100.5	0111_B	3.08	2.92	0.16
75.4	1000_B	3.05	2.90	0.15
37.7	1010_B	3.02	2.89	0.13
18.9	1100_B	3.00	2.88	0.12

The current consumed by the devices were even lower than those measured in **Table 17** and **Table 18**. This is because the system frequency is much lower in prescaler mode than in PLL mode.

9 Summary

- Idle mode, slow-down mode and power-down mode are the three power saving modes offered in the XC800 product family.
- A combination of idle and slow-down mode can help to reduce power consumption further, but it is not a certainty. It will depend on factors such as the peripheral activities and the operating frequency.
- One of main factors that affects the power consumption of a device is the system frequency. Reducing the system frequency will reduce the power consumption of the device.
- Slow-down mode can be used to reduce the system frequency that is supplied to the CPU and the peripherals thus reducing the power consumption of the devices when maximum bandwidth is not required. Users need to take note of the lower limit of the slow-down frequency that guarantees the flash programming and erasing operations.
- In PLL base mode, the system frequency can range between 5 MHz and 80 MHz depending on the setting of CMCON.VCOSEL and the K factor. At the lower range, the system frequency is very much reduced thus lowering the power consumption of the device. The system frequency can be determined by outputting the system clock signal on the CLKOUT pin.
- In prescaler mode, the system frequency is operating at half or equivalent of the oscillator frequency depending on the K factor.
- PLL-Base mode and Prescaler may offer potential power-saving capabilities for applications but users must be aware of their limitations.
- Unused peripheral can be disabled by gating off it’s clock input to save power.

10 Power Saving Checklist

- Disable the unused analog pins (Port 2) by setting them as output pins.
- Input port pins should not be left floating. Floating port pins should enable pull-down or set as output port. Output pins that are pulled-up should be set to high or have the pull-up disabled.
- The analog part of the ADC module can be disabled by resetting the bit GLOBCTR.ANON. This feature causes the generation of f_{ADCI} to be stopped and allows a reduction in power consumption when no conversion is needed.
- When the on-chip oscillator is used, XTAL should be powered down by setting bit OSC_CON.XPD (this bit is set by default). When the external oscillator is used, the on-chip oscillator should be powered down by setting bit OSC_CON.OSCPD.
- Disable peripherals that are not used.
- Use idle mode instead of polling loops (see **Section 3.3**).
- Reduce the operating frequency of the system, CPU or peripherals via the selection of power saving modes and clock modes when the maximum bandwidth is not required. User must also consider the response time of the system to interrupt events. When the system frequency is reduced, the interrupt latency will increase accordingly thus making it unsuitable for applications that require quick response time.