AN204775

F²MC-16FX Family, Standby Modes and Power Management

This application note shows the necessary steps for transition to standby mode and wake-up

Contents

1 Introduction

All Cypress 16FX-Microcontrollers offer a wide range of standby modes. In the standby modes "timer mode" and "sleep mode" CPU operation is stopped. However, timers can be used to provide periodic wake-up. Also single events may wake-up the MCU.

This application note shows the necessary steps for transition to standby mode and wake-up.

Used abbreviations:

CPU	Central Processing Unit
C-Unit	Clock Unit
EI	External Interrupt
HW	Hardware
I	Interrupt Enable-Flag of PS
ICR	Interrupt Control Register
IE	Interrupt Enable (Resource)
IL	Interrupt Level (ICR)
ILM	Interrupt Level Mask (PS)
INT	Interrupt (CPU)
IR	Interrupt Request (Resource)
IRQ	Interrupt Request
ISR	Interrupt Service Routine
MCT	Main Clock Timer
MCU	Microcontroller Unit
MUX	Multiplexer
PLL	Phase Locked Loop
PS	Processor Status
RTC	Real-Time Clock
SW	Software
WDT	Watchdog Timer

2 Standby Modes

The current consumption of the MCU can be reduced by selecting the number of running resources on chip.

2.1 Overview

In general, the normal run modes provide higher performance but also higher current consumption. The standby modes have lowest current consumption. Hence, the following table is somehow sorted from top to bottom, from high performance to low performance, from normal current consumption to low current consumption.

Table 1. Operation Modes

Operation		Clock supply				watchdog
		PLL	CPU	Peripherals 1 & 2	Source Oscillators	
Normal Run modes	PLL run mode	factor 1..25	active	active	active	active
	Main run mode	inactive				
Low Power Standby modes	PLL Sleep mode	factor 1..25	inactive			
	Main Sleep mode	inactive				
	PLL Timer mode	factor 1..25		inactive		
	Main Timer mode	inactive				
	Stop				inactive	inactive

In standby modes the CPU is not running. Only resources are supplied with clock. Those resources can be used to wake-up the MCU from standby mode. Depending on standby mode, different clock scheme is used.

Because all oscillators are stopped, the stop mode has lowest current consumption. However, only external interrupts or a reset can wake-up from stop mode.

2.2 Wake-up from standby modes

To wake up from a standby mode an event from a resource is necessary. The possible resources, which can wake-up an MCU, depend on the standby mode.

Table 2. Wake-Up Sources in Standby Modes

Standby Mode	Wake-Up Source		
	External Interrupts	Source Clock Timers, RTC	Other Resources
Sleep modes	✓	✓	✓
Timer modes	✓	✓	-
Stop mode	✓	-	-

The wake-up request is similar to interrupt requests. The major difference to interrupts is that CPU operation is stopped. Unlike interrupt requests, wake-up requests are accepted regardless of the processor status in PS-register. Hence, wake-up is possible, even if interrupts are disabled or interrupt priority is not sufficient for interrupts.

Consequently, wake-up conditions are enabled by a subset of interrupt settings. Because of these conditions, interrupt configuration is a possible condition but not a necessary condition.

Rule 1: Standby wake-up conditions are a subset of interrupt conditions.

If the wake-up condition is already present when changing to standby mode, the MCU does not enter standby mode. In this case, it disregards the standby mode transition operation and continues with normal program execution.

Rule 2: MCU will not enter standby mode, if a wake-up/interrupt request is present.

2.2.1 Wake-up by interrupt request

An interrupt is executed by adequate configuration of resource, interrupt controller and CPU:

Table 3. Conditions for Accepting Interrupt Request

Module	Flag/Register	Value
Resource	Interrupt Request Flag	requested (IR = 1)
	Interrupt Enable Flag	enabled (IE = 1)
Interrupt Controller	Interrupt Level of resource	resource level (7 > ICR:IL ≥ 0)
CPU processor status	Interrupt Level Mask	level mask (PS:ILM > ICR:IL)
	Global Interrupt Enable	enabled (PS:I = 1)

- The request flag of resource is set (e.g. "buffer full") and interrupts enabled.
- The level (IL-bits) for this interrupt was set to a value smaller than 7.
- The level has to pass the level mask and enable flag in Processor Status Register (PS).

In addition, a valid interrupt vector has to be set in the program code. If all conditions are met, the Interrupt Service Routine (ISR) is entered.

If an interrupt request occurs during standby mode, the controller will wake-up first. After that, the interrupt service routine is executed.

After wake up, the first instructions that are executed are the instructions that were in the pipeline when the standby mode was requested. This holds true even when the interrupt is serviced by the ISR.

2.2.2 Wake-up by wake-up request only

Unlike interrupts, the wake-up conditions do not depend on dedicated CPU-register PS (Processor Status), because CPU-operation is stopped.

Table 4. Conditions for Accepting Wake-Up Request

Module	Flag/Register	Value
Resource	Interrupt Request Flag	requested (IR = 1)
	Interrupt enable Flag	enabled (IE = 1)
Interrupt Controller	Interrupt level of resource	enabled (7 > ICR:IL ≥ 0)

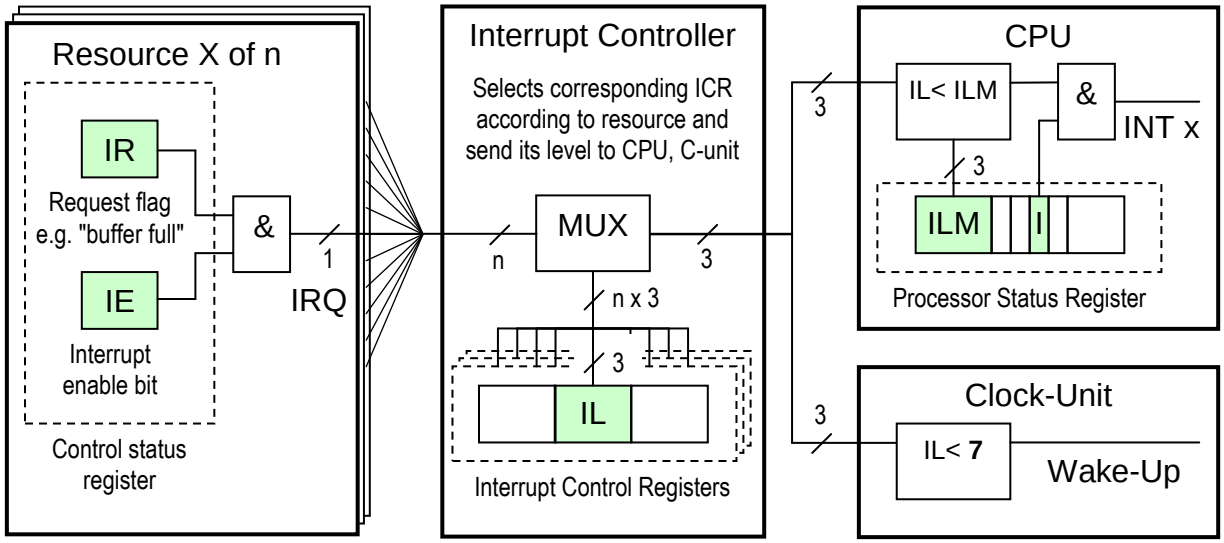
Wake-up is not priority-dependent. All resources, which are valid for that standby mode and which are activated, may wake-up the MCU, if their level in interrupt control register is less than seven.

Furthermore, there is no dependency on global interrupt enable flag (PS:I). Hence, wake-up is possible, even if interrupts are disabled.

Rule 3: Wake-up from standby mode is independent from CPU status PS (Interrupt Level Mask PS:ILM, Interrupt Enable flag PS:I)

After wake-up from standby mode, the interrupt request is still pending. As soon as all remaining conditions are met (e.g. by enabling interrupts), the ISR can be executed.

Figure 1. Simplified Wake up Signal Flow from Resource to CPU and Clock-Unit



3 Transition to and wake-up from Standby Mode

Transition to standby mode and wake-up from standby mode requires setting only a few registers. However, there might be side effects on other parts of the software. Following sequences do not apply to all applications but describe a general scenario.

The possible interrupt handling of events, which also wake-up the controller, should be separated from wake-up sequence.

Interrupts should be disabled during the whole sequence of…

- checking software conditions whether to change to standby mode,
- transition to standby mode,
- re-transition to standby
- and final release.

This avoids that interrupt service routines will interfere with pre-standby or post-standby operation.

Rule 4: Standby-transition and wake-up should be performed while interrupts are disabled.

The transition to standby mode is initiated by writing SMCR:SMS bits.

If the PLL is disabled in standby mode, it needs to start-up again, after wake-up. During PLL-stabilisation time (a few milliseconds), the application may perform any operations, which do not rely on CPU-speed or resource-speed (e.g. UART). In that time, the application could perform post-standby operations.

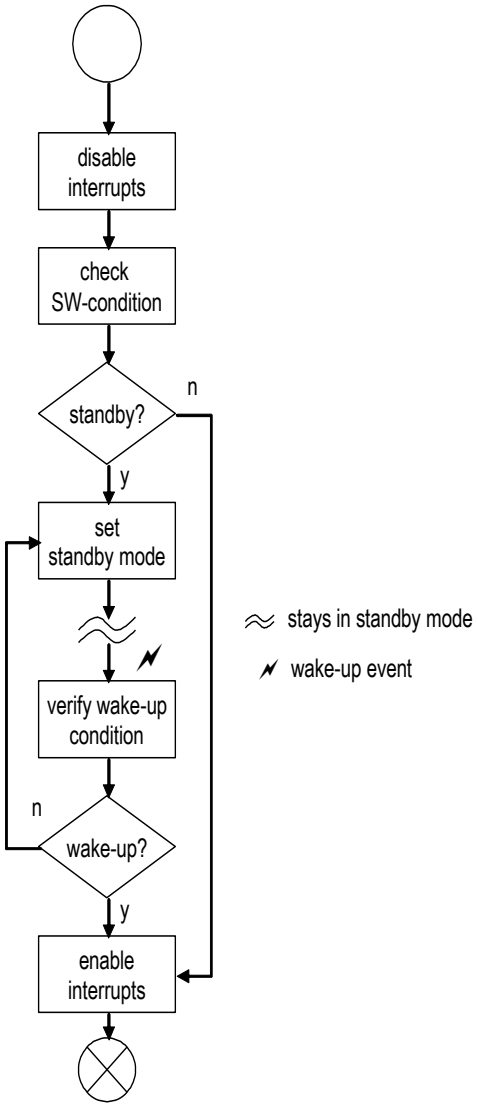
Rule 5: The PLL-stabilisation time should be considered as usable time after wake-up.

3.1 Sequence for transition to standby mode and wake-up by single events

A single event can be e.g. an external interrupt or an UART-RX-interrupt. On those events, the MCU might have to release standby mode. However, there might be some more conditions, which are not covered by the request of the resource (e.g. additional port pins or certain character in UART-buffer).

If those conditions are not met yet, the MCU is sent to standby mode again.

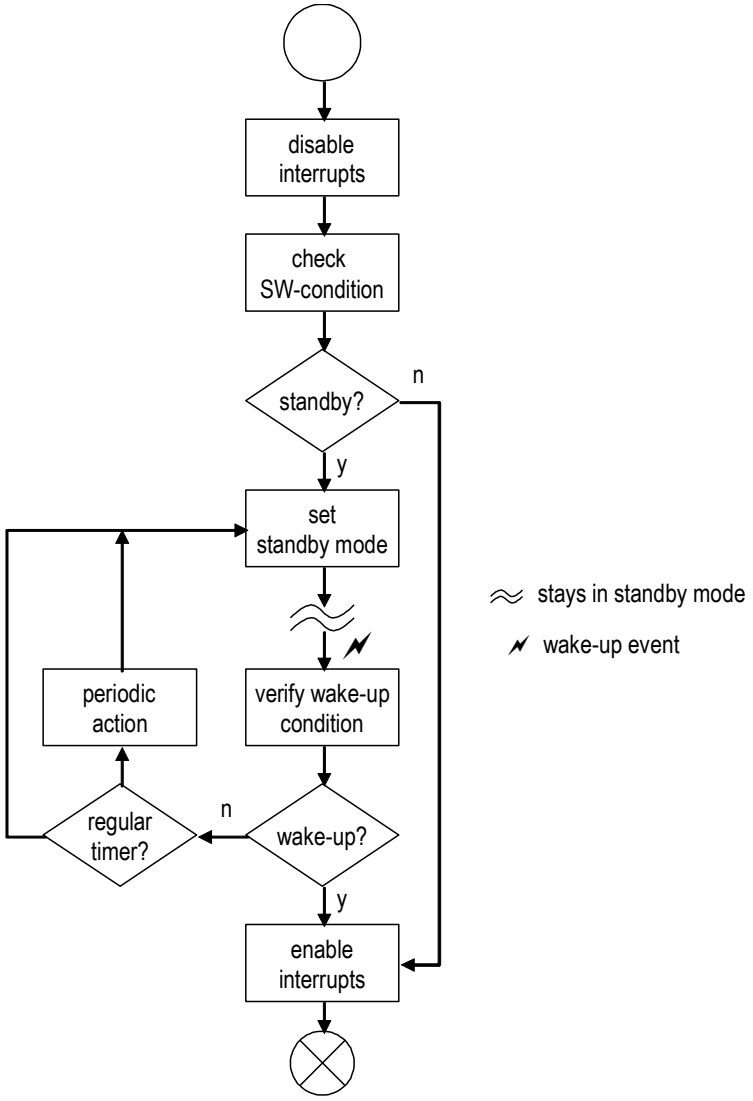
Figure 2. Standby Transition and Wake-Up By Single Event



3.2 Sequence for transition to standby mode and wake-up by cyclic events

This is the same as with single wake-up event. In addition, a cyclic event (timer) can be used to regularly update data (e.g. a watch clock). In this case, the application might have to leave the shown block in order to enable interrupts or to change the clock mode. However, if the amount of code is small, the update operation can be called from the shown block.

Figure 3. Standby Transition and Wake-Up By Single Event and by Cyclic Timer Event



3.3 Sequence for transition to standby mode and wake-up by cyclic events with the need of resetting the watchdog counter.

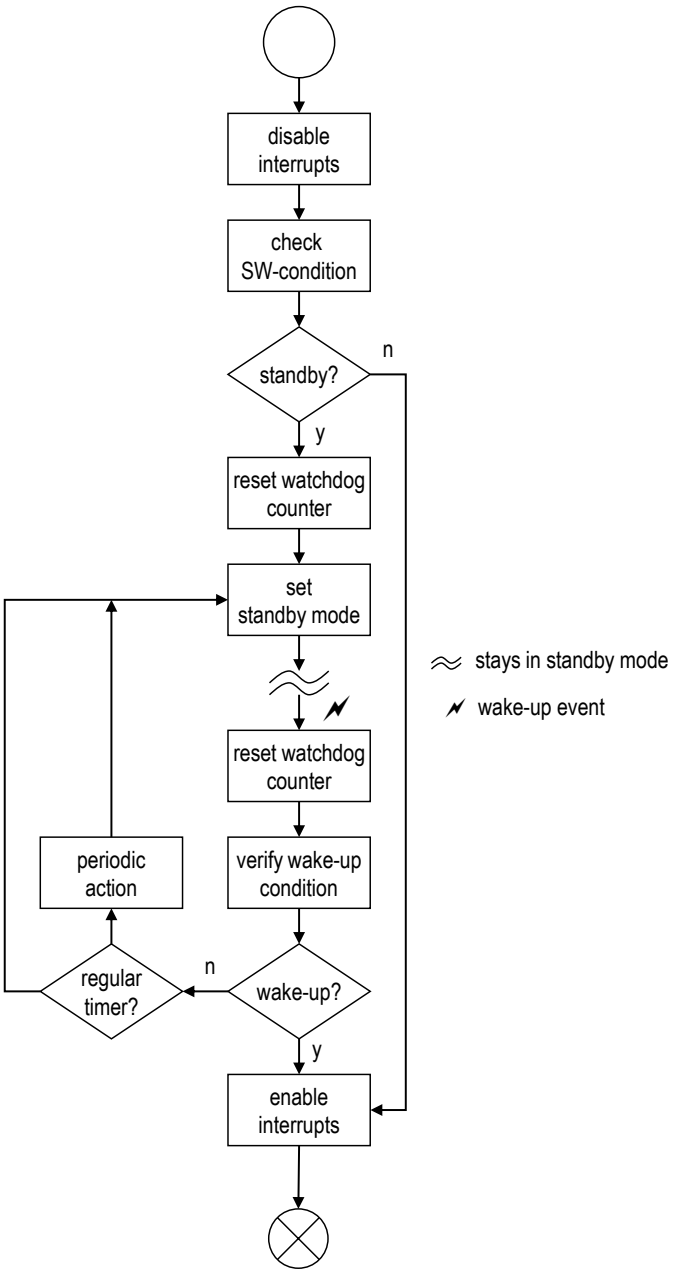
This is same as wake-up by single plus cyclic event. However, it is extended by the operation needed for resetting the watchdog counter.

Furthermore, the SW-wake-up condition is more complex, because it has to distinguish wake-up events for resetting watchdog, wake-up events for cyclic action, and final wake-up of application.

The cyclic wake-up time (e.g. timer) has to be set to a period, which covers both the cyclic update of application data and the watchdog timer period.

The combination and shifted occurrence of those events must be considered.

Figure 4. Standby Transition and Wake-Up By Single Event and by Cyclic Timer Event and the Need to Reset Watchdog



4 Example of Resetting Watchdog Counter in Timer Mode

This example illustrates the scenario that two timers and watchdog counter have to be co-ordinated. Therefore, it covers several possible cases.

4.1 Used Resources

4.1.1 External Interrupts (EI)

External interrupts are used to wake-up on changing external signals as keypad, switches, and others. In this example, the configuration of the external interrupts of a running application differs from the wake-up configuration.

4.1.2 Real Time Clock (RTC)

RTC is used here to provide wake-up events for Seconds and Minutes. Apart from wake-up, the RTC also triggers the appropriate interrupt service routines in order to update a display.

4.1.3 Watchdog Counter (WDT)

Watchdog counter is used to observe the application. If the application does not regularly reset the WDT, it expires and resets the entire MCU. This enables to restart malfunctioning applications (e.g. deadlock).

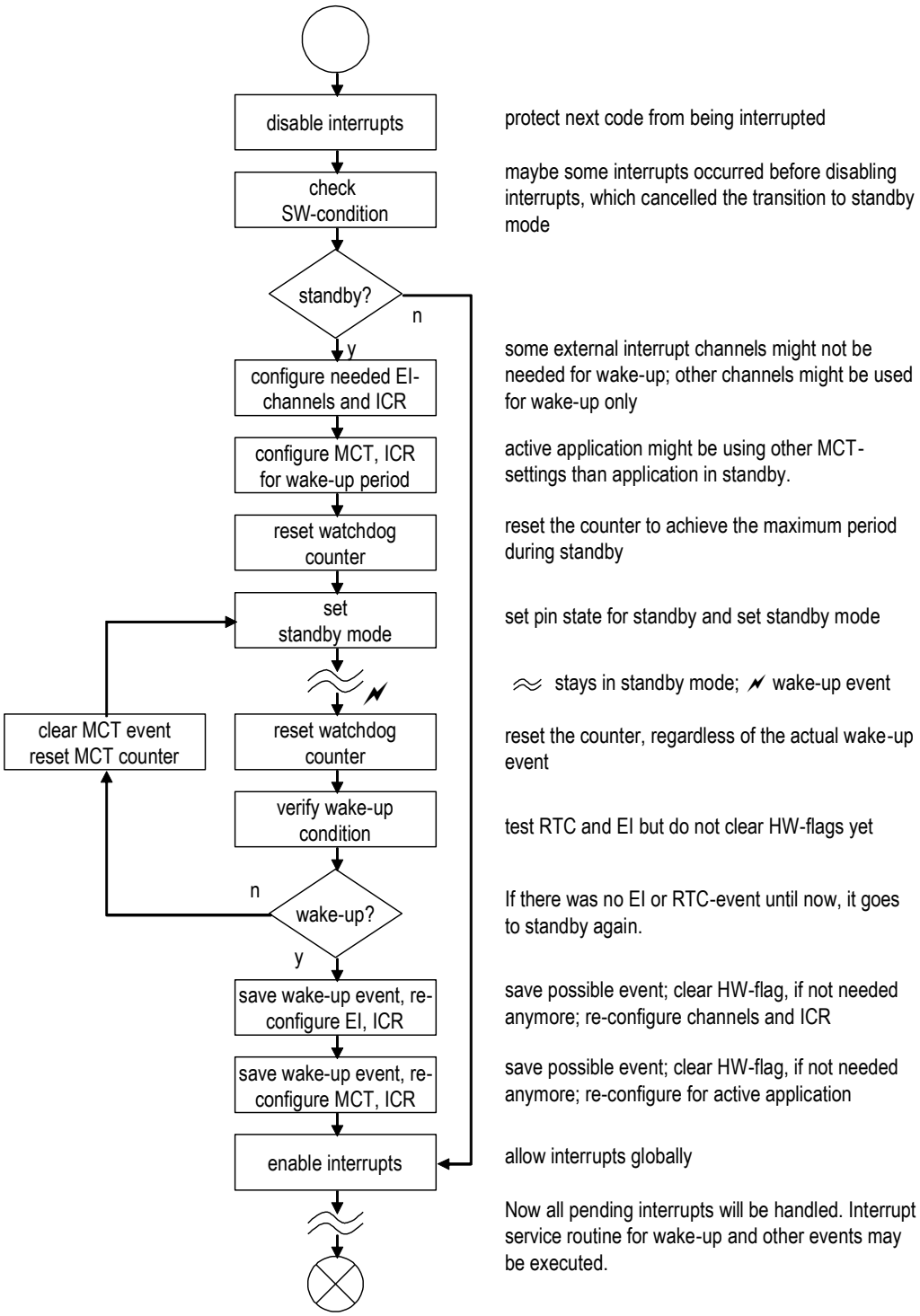
In this example, we assume a WDT, which does not stop in standby mode. In order to clear the WDT, the standby mode has to be released cyclically.

4.1.4 Main Clock Timer (MCT)

Since RTC does not wake-up MCU as often as needed to reset the watchdog counter, the MCT is used to wake-up MCU several times in a Second.

4.2 Standby/wake-up Flow

Figure 5. Example of Transition to Standby and Wake-Up By Single Event, by Cyclic Timer Event, and the need to Wake-Up by Extra Timer for Watchdog Reset



5 Power Management

Power management refers to the selective shutdown or speed down of system components that are idle or underutilized. The most important factor in determining power consumption in any microcontroller design is the system Clock Speed and Core Voltage.

The dynamic power dissipation is proportional to the square of operating voltage and linearly proportional to operating frequency. Therefore, lowering clock frequency will decrease the dynamic power dissipation linearly, while reducing Core Voltage will lower it exponentially.

16FX architecture offers following options to optimize power consumption as per the application need

- Multiple Clock Source

  Having multiple clock sources helps in managing the current consumption of the MCU.

  For example to minimize current consumption when the part goes into standby mode, one can use the external 32 kHz crystal to wake up the MCU using a Real-Time Counter (RTC) with wake-up feature.

- Clock Switching

  The ability to implement variable clocking allows throttling back the frequency for lower current and powering up the clock rate to full speed when math-intensive computing is required. Various clock sources, a PLL, a divider and clock enable/disable gives a very flexible architecture for power management.

  It is better to switch the system to a lower clock frequency if the MCU is in an idle mode.

  For example in a system where MCU is awaiting an input from input device and if there is no input for a pre determined period of time, system should be programmed to switch to low frequency. The clock frequency returns to normal when the system receives an input from an input device.

- Clock Divider

  All peripherals derive their timing from either CLKP1 or CLKP2. For example the Universal Asynchronous Receiver/Transmitter (UART) clock rate is determined by further dividing the CLKP1 signal. CAN core clock rate is determined by further dividing the CLKP2 signal. This frequency can be selected as close as the requirement of bit rate of application to reduce power consumption.

- Power down Modes

  Enabling and disabling peripherals as needed also helps lower power consumption. Periodic wake-up capabilities and the ability to wake up the MCU based on network communications or other outside signals allow the device to remain switched off until a user needs it.

Following tables shows I_{CC} measured for various power-down modes. It can be seen that current consumption is very low in stop mode and in run mode it is proportional to clock frequency.

Table 5. Current Consumption in Power Down Modes (CLKS1 = CLKPLL, Vcc = 5V)

Current Consumption						
CLKB (MHz)	**CLKP1 (MHz)**	**CLKP2 (MHz)**	**PLL Run Mode ICCPLL(mA)**	**PLL Sleep Mode ICCSPLL (mA)**	**PLL Timer Mode ICCTPLL (mA)**	**Stop Mode ICCH (mA)**
56	56	8	33.86	9.20	1.73	0.131
28	56	8	23.26	9.18	1.73	0.131
8	56	8	13.28	9.18	1.73	0.130
56	28	8	28.56	5.76	1.73	0.130
56	8	8	21.42	3.28	1.73	0.130
56	56	28	34.01	9.30	1.73	0.130
56	56	14	33.94	9.23	1.73	0.130
56	56	7	33.90	9.18	1.73	0.130

For the above measurement, MB96F346RSA mounted on SK-16FX-100PMC target board is used. In the measurement test setup PPG0, UART0 and CAN0 are active in PLL RUN mode. MCU Vcc is 5 Vdc.

Table 6. Current Consumption in Power Down Modes (CLKS1 = CLKMC, Vcc = 5V)

Current Consumption						
CLKB (MHz)	CLKP1 (MHz)	CLKP2 (MHz)	Main Run Mode ICCMAIN(mA)	Main Sleep Mode ICCSMAIN (mA)	Main Timer Mode ICCTMAIN (mA)	Stop Mode ICCH (mA)
4	4	4	3.55	1.41	0.61	0.131
4	2	4	3.15	1.18	0.61	0.131
4	1	4	3.02	1.05	0.61	0.130
2	4	4	2.61	1.42	0.61	0.130
1	4	4	2.01	1.41	0.61	0.130
4	4	2	3.42	1.29	0.61	0.130
4	4	1	3.36	1.22	0.61	0.130

For the above measurement, MB96F346RSA mounted on SK-16FX-100PMC target board is used. In the measurement test setup PPG0, UART0 and CAN0 are active in Main RUN mode. MCU Vcc is 5 Vdc.

Table 7. Current Consumption in Power Down Modes (CLKS1 = CLKMC, Vcc = 5V)

Current Consumption						
CLKB (MHz)	CLKP1 (MHz)	CLKP2 (kHz)	Main Run Mode ICCMAIN(mA)	Main Sleep Mode ICCSMAIN (mA)	Main Timer Mode ICCTMAIN (mA)	Stop Mode ICCH (mA)
2	2	250	1.93	0.92	0.61	0.130
2	1	250	1.86	0.80	0.61	0.130
1	2	250	1.49	0.91	0.61	0.130
1	1	250	1.30	0.79	0.61	0.130

For the above measurement, MB96F346RSA mounted on SK-16FX-100PMC target board is used. In the measurement test setup PPG0, UART0 are active in Main RUN mode. CAN0 is off. MCU Vcc is 5Vdc.

Table 8. Current Consumption in Power Down Modes (CLKS1 = CLKMC, Vcc = 3.3V)

Current Consumption						
CLKB (MHz)	CLKP1 (MHz)	CLKP2 (MHz)	Main Run Mode ICCMAIN(mA)	Main Sleep Mode ICCSMAIN (mA)	Main Timer Mode ICCTMAIN (mA)	Stop Mode ICCH (mA)
4	4	4	3.44	1.31	0.52	0.11
4	2	4	3.02	1.08	0.52	0.11
4	1	4	2.90	0.95	0.52	0.11
2	4	4	2.49	1.31	0.52	0.11
1	4	4	1.92	1.32	0.52	0.11
4	4	2	3.30	1.19	0.52	0.11
4	4	1	3.24	1.12	0.52	0.11

For the above measurement, MB96F346RSA mounted on SK-16FX-100PMC target board is used. In the measurement test setup PPG0, UART0 and CAN0 are active in Main RUN mode. MCU Vcc is 3.3 Vdc.

Table 9. Current Consumption in Power Down Modes (CLKS1 = CLKMC, Vcc 3.3V)

Current Consumption						
CLKB (MHz)	**CLKP1 (MHz)**	**CLKP2 (kHz)**	**Main Run Mode ICCMAIN(mA)**	**Main Sleep Mode ICCSMAIN (mA)**	**Main Timer Mode ICCTMAIN (mA)**	**Stop Mode ICCH (mA)**
2	2	250	1.81	0.82	0.52	0.11
2	1	250	1.76	0.71	0.52	0.11
1	2	250	1.38	0.82	0.52	0.11
1	1	250	1.21	0.70	0.52	0.11

For the above measurement, MB96F346RSA mounted on SK-16FX-100PMC target board is used. In the measurement test setup PPG0, UART0 are active in Main RUN mode. CAN0 is off. MCU Vcc is 3.3Vdc.

Please note that for all measurements, the Low Voltage detector was enabled.

Further, these readings are valid for the above test setup only and should not be considered as a reference.

6 Additional Information

Information about Cypress Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

7 Document History

Document Title: AN204775 - F²MC-16FX Family, Standby Modes and Power Management

Document Number: 002-04775

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	MKEA	09/04/2006	Initial Release
			06/22/2007	Updated for Additional Information
			08/03/2007	Updated as per review comments
			08/15/2007	Updated as per review comments
			09/20/2007	Added additional current measurements
			09/28/2007	Added additional current measurements
*A	5060625	MKEA	12/22/2015	Migrated Spansion Application Note from MCU-AN-300226-E-V15 to Cypress format
*B	5835288	AESATMP9	07/27/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709