Freescale Semiconductor
Application Note

Document Number: MPC5668x
Rev. 0, 09/2010

Using Sleep Mode on the MPC5668x

by: David McMenamin
Applications Engineer
East Kilbride, Scotland

Contents

1 Introduction....................1
2 MPC5668x Power Modes....................2
3 Clock, Reset and Power Module (CRP)....................2
4 Using Sleep Mode....................3
5 Configuring a wake up source to exit sleep mode....................3
6 Entering into Sleep Mode....................6
7 Low Power Mode Exit routine....................9
8 Wake up to Flash or RAM....................11
9 Debugging Low Power Mode....................11
10 Examples....................12
11 Conclusion....................14

1 Introduction

The dual core MPC5668x[1] microcontrollers (MCU) have been developed to address the needs of high end automotive gateway and body control applications. The MCU must offer good performance and a range of peripherals that can meet the process and communication intensive demands of the application when the vehicle is running. The applications also require the MCU to support a level of functionality when the automobile is parked and power available from the battery only. It is therefore, important that the MCU offers low power consumption. This helps to maximize the time the vehicle's electronic system can run from the battery.

Traditionally two MCU’s would have been used to achieve this. A high performance MCU to manage a running application and a small low performance low power MCU that will be used in an idle mode. The MPC5668x offers a single MCU solution for this task with its run and sleep modes of operation, helping to reduce system cost and complexity.

This application note will explain how to use the low power mode, the process and options available for transitioning between run and sleep mode on the MPC5668x and provide example scenarios and software. The software used in the examples within this applications note is provided in AN4150SW and can be downloaded from www.freescale.com. This application note must be used in conjunction with the MPC5668x Reference Manual.

1. MPC5668x includes MPC5668E and MPC5668G

© 2010 Freescale Semiconductor, Inc.

2 MPC5668x Power Modes

The MPC5668x supports two modes of operation:

Run Mode

- Default mode out of reset
- Device is running normally, all resources can be used, executing application code
- The estimated room temperature current in this mode is ~220 mA at 116 MHz with all peripherals and both cores running.

Sleep Mode

- Power is removed for large areas of the device to eliminate static leakage from these areas (power gating).
- Essential parts of the silicon remain powered to monitor wake up events and control the entry and exit sequence from Run to Sleep and vice versa.
- Optional amounts of RAM can be maintained to allow program critical information to be maintained during sleep mode.
- A range of external and internal time based events can be used to cause the MCU to exit sleep and return to run mode.

A summary of what remains powered in Run and Sleep mode is shown in Table 1 below:

Table 1. Mode Summary

Features	Run Mode	Sleep Mode
Standard Cell Logic (e200z6, e200z0, DMA, peripherals etc.)	Powered, Running	Powered Down
Flash	Power from Reset Optionally disabled	Powered Down
SRAM	All SRAM Powered	Optional Amounts of SRAM powered: 0: All RAM powered 1: First 32KB powered 2: First 64KB powered 3: First 128KB powered
Output Pins	Active	Disabled
Input Pins	Active	Active for Wakeup
RTC, API	Optional	Optional

3 Clock, Reset and Power Module (CRP)

The primary function of the clock, reset, and power (CRP) module is to maintain all the control logic that requires power when other portions of the MCU are powered down in sleep mode. The CRP manages entry into, operation during and exit from sleep mode.

The CRP consists of the following:

- Input isolation block
- RTC/API
- Wakeup and power status block
- Clock and reset control block
- Low-power state machine
- Bus interface unit.

Figure 1 shows block diagram of the CRP module:

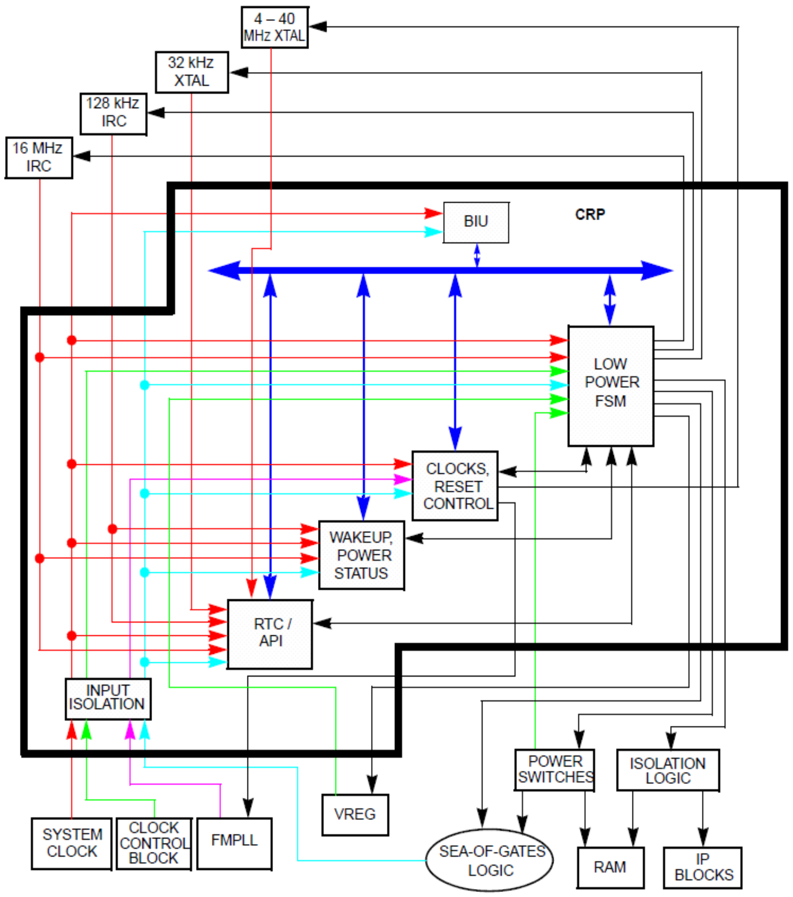


Figure 1. CRP module block diagram

The input isolation block allows inputs from external blocks to be driven to known states when the logic driving the input is powered down. The RTC/API block implements a real-time counter and periodic interrupt. The wakeup and the power status block implements the logic to select power mode operation and wakeup sources. The clock and reset control block implements miscellaneous logic related to PLL and oscillator operation, and reset gating for sleep mode. The low power state machine controls the transitions into and out of sleep mode. The Bus Interface Unit allows for read/write register access from the e200z6 and e200z0 cores.

4 Using Sleep Mode

When in run mode the user application must execute a sequence of events to cause the MCU to enter into sleep mode. It is important to consider the whole picture before entering and exiting low power mode. This section will explain and provide examples of what needs to be done and considered when using sleep mode in the following order:

1. Configuring a wake up source to exit sleep mode
2. Entering into Sleep Mode
3. Exiting Sleep mode
4. Program layout for entry and exit software.

5 Configuring a wake up source to exit sleep mode

It is important to have at least one valid wake up source configured before entering into the sleep mode. If no wakeup source is configured then the only way to exit from sleep mode will be from an external reset or power on reset.

The MPC5668x supports the following wake up sources:

- Real Time Counter (RTC)
- Real Time Counter Roll Over
- Autonomous Periodic Interrupt (API)
- External Pin transition – 32 Pins supported

Multiple wake up sources can be enabled at any time, with the first one to occur causing the MCU to exit sleep.

5.1 RTC/API

The RTC and API is based on a 32-bit free running counter that can be driven from the 32 kHz external oscillator, 128 kHz Internal RC oscillator (IRC), 16 MHz IRC or the 4-40 MHz external oscillator. The MCU can be configured to exit sleep mode in case of a RTC match value and/or when the counter rolls over.

The clock source chosen for the RTC is a trade-off between accuracy, power, and recovery time. The lowest power option is the 32 kHz IRC that consumes approximately 1 µA. The disadvantage is that the clock is accurate to around only 10% deviation. If a pin transition occurs immediately after a clock edge, the edge won’t be latched and acted upon until the next clock edge. At 32 kHz, the clock is over 30 µs. That may not be a problem in most applications. The next lowest power option is using the 32kHz external oscillator facility. This option consumes around 3 µA of current but requires the addition of a 32 kHz crystal. The advantage is that this is of a greater accuracy than the internal IRC and could potentially be used to track the “time of day.”

The 16 MHz IRC is the most accurate internal clock available (<5%), consuming a higher current of 160 µA. This is the default system clock from sleep mode exit, hence allowing near instantaneous recovery. This capability allows registers, modules, etc., to be configured while waiting for the external crystal to stabilize. Alternatively, it supports rapid code execution during short run cycles, enabling the device to return to sleep mode quicker.

There is a fixed 32-bit divide going into the counter that provides a 1ms resolution for the 32 kHz sources. The 16 MHz IRC is another source, and this can be the full 16 MHz or passed through a fixed divide-by-512 counter. This fixed divider again provides a 1 ms resolution on the 32-bit counter. This 1 ms resolution provides up to 1.5 months time period. The following RTC/API block diagram including sources and divides is given below:

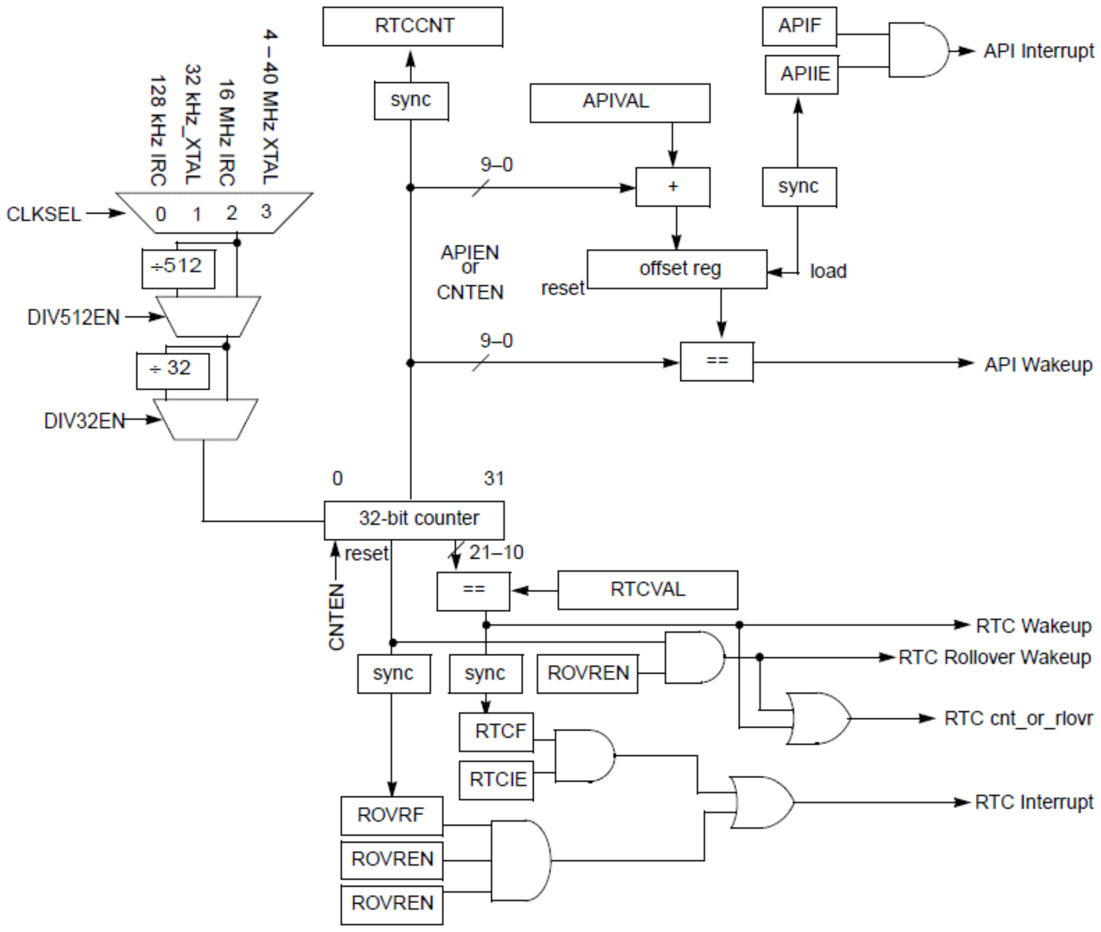


Figure 2. RTC/API block diagram

To use RTC or the API, the user needs to trade off the length of time the timer may have to wait before wakeup against resolution:

- The RTC system with its 12-bit compare gives a 1 s to 1 hour timeout with a 1 s resolution.
- The API system with a 10-bit compare supports wakeup intervals 1 ms to 1 s.

  The match values of these timers can be changed while they are running although if the part is in sleep mode, there is no mechanism to achieve this. The user specifies a RTC match value using the RTCVAL field within the RTC control register. This value is compared to bits 10-21 of the RTC counter. The user must ensure the CRP is configured to keep the chosen clock source active in sleep mode if the RTC is to be used a wakeup source. The RTC must also be enabled and selected as a wakeup source. The code example below provides an example sequence for configuring the RTC as a wakeup source with wake up match and roll over enabled. The RTC is driven from the 128 kHz IRC in the following example:

```
/* Configure RTC as Wake up Source */
CRP.RTCC.B.CLKSEL = 0x01;     /* 128K IRC drives RTC */
CRP.RTCC.B.RTCVAL = 0xFFF;    /* Set Match value for RTC */
CRP.PSCR.B.RTCWKEN = 0x1;/* Enable RTC as wake up source */
CRP.RTCC.B.ROVEN = 0x1;       /* Enable Wakeup on RTC Rollover */
CRP.CLKSRC.B.EN128KIRC = 1;/* enable 128K IRC in Sleep */
CRP.RTCC.B.CNTEN = 1;         /* Enable RTC */
```

The API is enabled separately. To use the API the RTC must be enabled and a clock source running. The user then sets the API compare. Software Example 1 utilises the API and RTC as wake up sources.

5.2 Pin Wakeup

An edge transition on an external pin can be used to generate a wake up event to cause the MCU to exit from sleep mode. When in sleep mode up to 32 external pins can be monitored as a wake up source.

These are shown in Table 2

Table 2. Wake up Pin options

CRP_PWKENL							
PWKn	Pin	PWKn	Pin	PWKn	Pin	PWKn	Pin
0	PB4	8	PD9	16	PE9	24	PJ9
1	PB5	9	PD11	17	PE11	25	PJ10
2	PB6	10	PD13	18	PE13	26	PJ11
3	PB7	11	PD15	19	PF3	27	PJ12
4	PD1	12	PE1	20	PF7	28	PJ13
5	PD3	13	PE3	21	PF11	29	PJ14
6	PD5	14	PE5	22	PF15	30	PK3
7	PD7	15	PE7	23	PJ8	31	PK6

The wake up event can occur on a positive, negative or either edge transition. Each pin must be enabled as a wake up source and the wakeup edge(s) selected within the CRP Pin Wake up enable register (CRP_PWENR). Pins being used for wake up must have their Input buffer enable bit set within the corresponding SIU pin configuration register (IBE = 1: SIU_PCSR).

Either the 16 MHz IRC or the 128 kHz IRC can be used as the logic synchronizer and edge detect clock for pin wakeup. This selection is made from within the CRP Power Status and Control Register Wake up clock select bit (CRP_CSPR WKCLK). If the 128 kHz clock is selected, it must remain enabled when in sleep mode.

To facilitate wake up of the MCU upon reception of data on a communications interface, many of the pins that can be selected as wake up sources are receive pins on the communications interfaces. The first frame received would cause the MCU to exit from sleep mode. This frame would however be lost as the peripheral would not yet be powered and configured, but can be configured ready for reception of the next frame.

6 Entering into Sleep Mode

To allow the MCU to successfully enter into sleep mode, a sequence of events must be followed. A valid wake up source, as discussed, must be configured and an exit routine must be present in the location specified before entering. This is discussed later in this applications note. The flow of events required to enter into sleep mode is given in Figure 3

NOTE

All program/erase operations on the flash should be completed before attempting to enter into sleep mode.

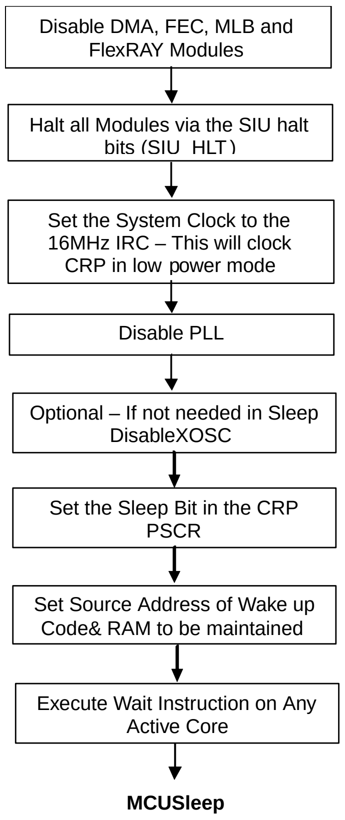

Disable DMA, FEC, MLB and FlexRAY Modules
Halt all Modules via the SIU halt bits (SIU_HLT)
Set the System Clock to the 16MHz IRC – This will clock CRP in low power mode
Disable PLL
Optional – If not needed in Sleep DisableXOSC
Set the Sleep Bit in the CRP PSCR
Set Source Address of Wake up Code& RAM to be maintained
Execute Wait Instruction on Any Active Core
MCUSleep

Figure 3. MCU Sleep mode entry procedure

Appendix A provides low power mode entry software example that follows the above flow diagram.

6.1 Disable DMA, FEC, MLB and FlexRAY Modules & Halt all Modules via the SIU halt bits (SIU_HLT)

The relevant bit in the SIU Halt Register (SIU HLT) enables the halt logic built into the module to perform a controlled shutdown of the module rather than just simply stopping the clock to the module. This allows a module currently performing a task to complete the task and stop gracefully (such as a DSPI in the middle of transmitting a message). Each module also has a corresponding flag in the HLTACK (halt acknowledge) register. When a module is requested to halt, and has completed its halt sequence, it sets the appropriate halt acknowledge flag in the HLTACK register. In this way, software can easily monitor the progress of any modules requested to halt. When entering a low power mode, because all modules are going to be halted, writing 0xFFFFFFFF to each of the 32-bit SIU_HLT registers and then waiting for the flags to assert in confirmation for halting the modules.

However, on the MPC5668x family, the DMA, MLB, Ethernet and the FlexRay modules do not contain the halt logic necessary to perform a graceful halt. This is due to the complex nature of these particular modules. If any of these four modules are used within the application, the user must take care of shutting them down manually. To do this the user must ensure that all the data transfers have completed and that no more are pending.

6.2 Set the System Clock to the 16 MHz IRC

While the MPC5668x is in sleep mode, any operation of the CRP module is clocked by the 16 MHz IRC. Thus, prior to sleep mode entry, the system clock must be set to the 16 MHz IRC. This is achieved by writing 00 to the SYSCLKSEL bits in the SIU_SYSCLK register. If the 16 MHz IRC is not selected prior to execution of the WAIT instruction, the 16 MHz IRC is automatically started by the system to clock the CRP module. In this case, whichever clock source that was clocking the system prior to sleep mode entry (most likely PLL/XOSC) is still active and will be using power.

6.3 Disable PLL

Once the system clock has been configured to be the 16 MHz IRC then the PLL can be disabled as it is not required to clock any logic when in sleep mode. Until sleep mode is entered the core will be clocked by the 16 MHz IRC. All peripherals should be halted by this stage so they are not affected by the clock source. Writing 0b000 to the CLKCFG bits in the ESYNCR1 register disables the PLL. Power to the PLL will be removed upon entry into sleep mode but it is a good practice to disable it before entry.

6.4 Optional – If not needed in Sleep Disable XOSC

If the XOSC is not used to clock the RTC then it can be disabled to save power. This is done by clearing the EN40MOSC bit within the CLKSRC register. If the XOSC is left enabled then the ENLPOSC bit in the same register will determine its behavior when low power mode is entered. If cleared the 4 – 40 MHz OSC clock is disabled to save power, but connection to the external crystal is still active thus supports faster recovery time for availability of 4 – 40 MHz OSC after sleep recovery. This supports the full 4 – 40MHz range of external crystals. If set, then 4 – 40 MHz OSC clock is active and may be used as the clock source for the RTC/API. The external crystal frequency is limited to ≤ 8 MHz.

6.5 Set the Sleep Bit in the CRP_PSCR Register

Set the sleep Bit in the CRP_PSCR Register to tell the CRP to enter sleep mode when the wait instruction is executed on the active core(s).

6.6 Set Source Address of Wake up Code & RAM to be maintained

The CRP module contains the address of the software routine that each core must access when it exits from sleep mode. There is an individual register provided for each core CRP_Z6VEC and CRP_Z0VEC. The e200z0 core sleep recovery code can be located at any 16-bit aligned address in flash or RAM as the 30 most significant address bits can be specified. On the e200z6 core only the 20 most significant bits can be specified in the Z6VEC register, therefore code must be aligned to a 4k boundary with the addition of 0xFFC, the fixed value of the bits in the address field that are not specified by the user.

If only one core is to be used upon exit from reset, or only one core is being utilized in the system then the other core should be put into reset by setting the Z6RST/Z0RST bit in either the Z6VEC or Z0VEC register. This should be done prior to entering into sleep mode. A core will only begin executing code from the vector (Z6VEC/Z0VEC) register address after low power exit if it was not held in reset and executed the Wait instruction.

NOTE

It is not possible to set the RST bit of both cores at the same time when the device is in run mode.

The user must specify the amount of RAM memory that is to be maintained when the device is in sleep mode. This is done from within the CRP_PSCR (PSCR) register using the RAM select bits. The maximum power savings comes from maintaining the minimum amount of RAM possible.

6.7 Execute Wait Instruction on Any Active Core

Execute the Wait Instruction on any active Core. This can be done using the “wait” assembly mnemonic if supported by your compiler or the op code 0x7C00007C.

The MCU must now successfully enter into sleep mode. This can be verified by measuring the current that is being drawn by pass transistor on the VRC.

6.8 Other considerations before entering Sleep Mode

Depending on the nature of the user's software and what state the user wants the system to be at low power sleep mode exit, it may be necessary to save the machine state before entering sleep mode. This allows the application to return to where it was before entering sleep mode. This can be done by saving the contents of the core registers and then restoring them upon exit. This stack must also be maintained when doing this. A special register within the CRP called the Recovery Pointer (CRP_RECPTR) can be used to store the stack pointer. This can then be used by the recovery routine to restore the core register state before sleep mode if they are the last entry on to the stack.

7 Low Power Mode Exit routine

Exit from sleep mode is caused by a reset (power on reset or external reset pin) or the occurrence of a pre-defined wakeup event that was specified prior to entry into sleep. Upon exit, any core that executed the “wait” instruction prior to entering sleep mode will begin to execute code from the address stored in the corresponding Vector Register (CRP_Z6VEC / CRP_Z0VEC).

The code that will be executed from this point onward will depend on the needs of the following application tasks and the location where the wake up code is. Following list provides some guidelines on this:

- Reconfigure the Memory Management Unit (MMU) - (e200z6 core only)
- Restore the context
- Configure and change the system clock source
- Reinitialize RAM if disabled
- Re-enable cache
- Reconfigure peripherals
- Clear the sleep flag to enable i/o
- Establish the cause of the wakeup

7.1 Reconfigure the Memory Management Unit (MMU)

If the e200z6 core has to resume after exit from sleep mode then the MMU must be reconfigured. After exit from sleep mode there will be only one valid MMU entry- TLB0. This will be setup for the flash if the e200z6 sleep recovery vector is located in flash and RAM if it is located in RAM. The VLE bit contained within the Z6VEC register will determine if the VLE bit is set in the MMU entry. The valid address ranges in the MMU are shown depending on the memory location of the MMU.

Table 3. MMU Settings after Sleep Mode Exit

e200z6 VEC Location	Start of Valid Address Range	End of Valid Address Range
RAM	0x40000000	0x40000FFF
Flash	0x00000000	0x00000FFF

As only the 20 most significant bits can be specified in the Z6VEC field and the remaining bits are fixed adding on 0xFFC then the sleep mode exit procedure code cannot be placed sequentially from this address. Only memory is available for one BookE /32bit VLE or 2 16-bit VLE instructions. Therefore, the first instruction must be a branch to a lower numerical address where the low power exit code is located. An example of how this can be achieved with the single MMU entry and e200z6 core vector restrictions is shown below.

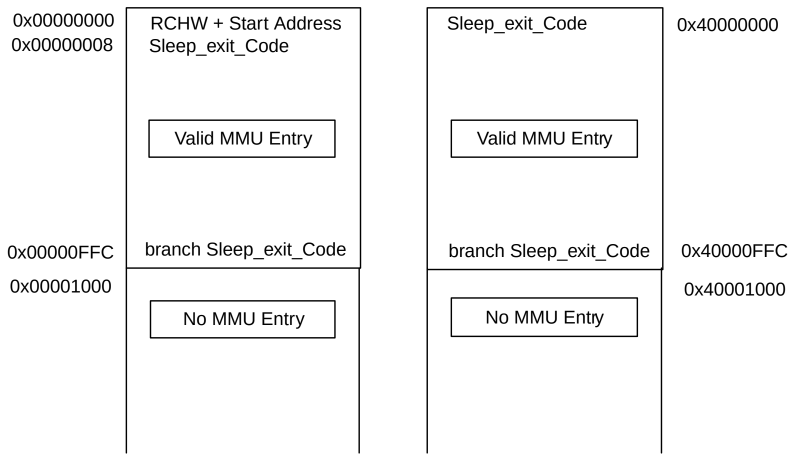


Figure 4. e200z6 Memory and Code location After Sleep Mode Exit

The MMU can then be configured to meet the needs of the application. At least one entry will be required for RAM, flash, Peripheral Bridge A and Peripheral Bridge B if the e200z6 core needs to access all of these resources. Software is included within the examples that show how to configure the MMU after exit from sleep mode. The e200z0 core does not have an MMU and all address map resources are available to the e200z0 after exit from sleep mode.

7.2 Restore Context

If the context of the core was saved to RAM and retained during sleep mode then the application may wish to restore this after exit from sleep mode. This is shown in the examples later in this applications note.

7.3 Configure and Change System Clock Source

The MCU will be clocked from the 16 MHz IRC at this point. It may be desirable to switch the system to a faster clock speed or to full system speed. This will be beneficial as it will speed up the time taken to reconfigure the system. Some peripherals may also need a faster clock speed to meet the needs of the application. If the exit from sleep is only for a short period of time before sleep mode is entered again it may not be desirable to start the faster clock as the external oscillator needs time to stabilize and the PLL needs to lock.

7.4 Reinitialize RAM

Any areas of RAM that were disabled when in sleep mode must be written to again before they can be used to initialize the ECC syndrome bits. RAM that remained powered can be left as these values are retained for these areas.

7.5 Reinitialize Cache

If cache memory is required then it must be re-enabled in the same way that was used previously in the application before entering sleep mode.

7.6 Reinitialize Peripherals

Power is removed from the peripherals when sleep mode is entered and applied again when sleep mode is exited. Therefore after sleep mode exit, all peripherals, except CRP module, are in their reset state and must be fully configured again before they can be used in the application.

7.7 Clear the Sleep flag to enable I/O

The sleep flag, SLEEPF within the CRP_PSCR is set to indicate that the MCU has entered into sleep mode. While SLEEPF is set, the pads remain in a safe state after sleep mode recovery and clearing SLEEPF will return the pads to normal operation. Writing 1 clears this status flag and a writing 0 has no effect.

7.8 Establish the cause of the Wakeup

If there are multiple wake up sources that can cause the MCU to exit from sleep mode then the application might require to identify what wakeup source caused the MCU to exit from sleep mode so it can carry out tasks according to this.

There are two registers that must be read to establish the cause of the wakeup:

CRP PSCR: There are 3 bits in this register that provide information about the cause of the exit from sleep mode.

1. RTC Counter Rollover Wakeup Flag (RTCOVRWKF) :The RTCOVRWKF bit indicates that a RTC counter rollover was the wakeup source
2. RTC Wakeup Flag (RTCWKF): The RTCWKF bit indicates that the RTC match was the wakeup source.
3. API Wakeup Flag (APIWKF): The APIWKF bit indicates the API was the wakeup source.

If the wakeup was caused by an edge transition on an external pin then this can be established by reading the Pin Wake up Source Flag register within the CRP (CRP_PWKSRCF).

8 Wake up to Flash or RAM

The code executed by either core after sleep mode exit can be located in either flash or RAM memory. There are advantages and disadvantages to placing the code in each memory type. The choice will be determined by the needs of the application.

If the sleep mode exit code is located in RAM then the FASTREC bit can be set within the CRP_RECPTR register. This shortens the period of time it takes to exit sleep mode from 1000 clock cycles to 16 clock cycles as there is no delay while the flash initializes. This is ideal if the application needs to wake up quickly and make some assessments before deciding to fully initialise the system or go back to sleep again. The disadvantage of this is that the software must be stored in flash and copied over into RAM at start up. A cost of time and memory duplication.

9 Debugging Low Power Mode

9.1 Debugging Through Low Power Mode

There is a handshaking procedure that is performed by the MCU and the debug hardware to allow the user to debug through sleep mode. Debuggers that support this functionality must be able to indicate the user when the MCU is in sleep mode. The tools vendor can advise if this feature is supported.

9.2 Checking Entry into Low Power Mode

If the debugger does not support sleep mode debug or the MCU that is being used independently from the debugger then the user can attach an ammeter in series with the VRC supply. The current drawn will greatly reduce when low power mode is entered. Please see the expected values in the MPC5668x Data sheet.

9.3 MCU is not entering Low Power Mode

Step the entry code for low power mode and ensure all the required tasks are being carried out correctly.

NOTE

The wait instruction cannot be stepped using debug software.

.

9.4 MCU is not exiting Low Power Mode

If MCU does not exit from sleep mode when it should have, then check that the wake up source is configured correctly and if it is dependant on a clock source, then the clock source is not disabled in sleep mode. If a reset occurs when exiting sleep mode then there might be a software issue in the start up code (for example, MMU region is not configured correctly and the core tries to access it).

The user can then step the code up until the wait instruction is under the control of the debugger. Then change the pc counter to the start up code before executing the wait instruction. Clear the core registers and configure MMU settings to mimic exit from sleep then step the code to see if the cause of the issue can be identified.

10 Examples

The example software that accompanies this applications note contains 2 low power examples for the MPC5668x. These examples have been developed to be run on the MPC5668EVB that is provided with the MPC5668KIT. The following examples are provided:

Example 1

Running from RAM and waking up to RAM after sleep mode. Fast recovery on Wake up on an edge transition on an external pin.

Example 2

Running from flash and waking up to flash. Wake up caused by API.

These examples have been built using the Greenhills Multi v5.05 PPC compiler.

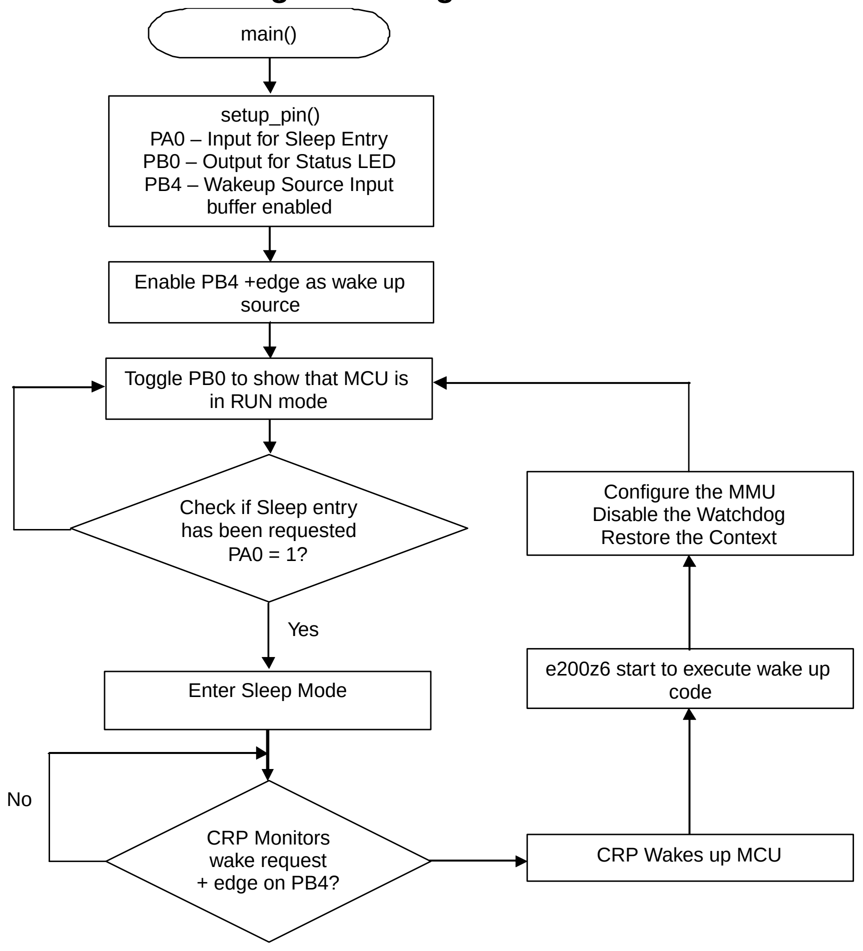
10.1 Example 1

This is a Book E coded RAM only single core (e200z6) example; it runs from RAM and wakes up to RAM. When running the code configures the MCU Pins and then remains in RUN mode toggling Pin PB0 (This can be connected to an LED on the EVB to give a visual representation) and monitoring PA0. A high input on PA0 causes the MCU to execute the sleep entry procedure and enter into sleep mode. Pin PB4 is configured as a wake up source prior to entering sleep mode. A rising edge on this pin will cause the MCU to exit from sleep mode, disable the watchdog reconfigure the MMU and restore the context. When the context is restored the MCU will return to the main function immediately after the enter sleep mode function. PB0 will begin to toggle again and the MCU is ready to re-enter sleep mode.

The wakeup code is located at address 0x40000FFC (A branch to 0x40000000 is located at this address) therefore the fast recovery bit is set as this is a location in RAM.

The flow chart below explains the operation of this example. All codes are executed by the e200z6 core:

Figure 5. Program flow


main()
setup_pin()
PA0 – Input for Sleep Entry
PB0 – Output for Status LED
PB4 – Wakeup Source Input buffer enabled
Enable PB4 +edge as wake up source
Toggle PB0 to show that MCU is in RUN mode
Check if Sleep entry has been requested PA0 = 1?
Yes
Enter Sleep Mode
No
CRP Monitors wake request + edge on PB4?
CRP Wakes up MCU
e200z6 start to execute wake up code
Configure the MMU
Disable the Watchdog
Restore the Context

Following key software files for this example can be found in Appendix A which is included in AN4150SW and can be downloaded from www.freescale.com.

- main.c – Main routine, enter sleep mode and pin setup
- execute_wait.s – saves the core context to the stack and executes the wait instruction
- LPM_Recover.s – sleep recovery functions.
- Z6_ram.ld – linker file

NOTE

To run this example, the user must init the MMU from the debugger before downloading and running the software.

10.2 Example 2

This example runs from flash memory and wakes up to flash memory. Unlike Example 1 one it does not depend on an external input to wake up. The wake up signal is generated by the RTC, using the16 Mhz IRC as the source clock for this. After entering sleep, again upon a high signal on PA0, to wake up the MCU waits for the RTC clock match to occur.

The key software files for this example can be found in Appendix B, which is included in AN4150SW and can be downloaded from www.freescale.com

11 Conclusion

Through this application note the procedure for entering and exiting from sleep mode on the MPC5668x has been discussed. It has shown the options that are available for wake up sources, clock options for wake up sources and code storage options for sleep entry and exit software. The advantages and disadvantages of the options have been discussed as well as the issues that can occur and the methods for debugging these issues. Additionally, example code has been discussed and provided.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and sofware implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor prodcuts are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2010 Freescale Semiconductor, Inc.

Document Number: MPC5668x
Rev. 0, 09/2010