

MC9S08GB/GT Low-Power Modes

by: Donnie Garcia and Scott Pape
8/16 Bit Applications and Systems Engineering
Austin, Texas

Introduction

This application note is a guide for using the MC9S08GB/GT microcontroller to achieve low power consumption. The MC9S08GB/GT is a member of the new low-cost, high-performance HCS08 Family of 8-bit microcontrollers. Some new features of this family include a 40-MHz HCS08 CPU, an enhanced instruction set, and a background debug controller (BDC) that provides an easy interface for in-system real-time debugging. Please see the device data sheet, Freescale document MC9S08GB60/D, for a more complete description of the features of this part. Always refer to the data sheet for the most current specification (<http://freescale.com>)

The MC9S08GB/GT microcontroller has additional features that pertain specifically to achieving low power consumption. These features provide a great deal of flexibility for the user and can be used to provide ideal conditions for many different types of applications.

System Clock Generation

The system clock can be generated from external (crystal, resonator, or square wave) or internal sources. Also, a frequency-locked loop (FLL) stage can be used to boost the external or internal clock source to a higher frequency. The MC9S08GB/GT has the ability to use a low range (32 kHz–100 kHz) or high range (1 MHz–16 MHz) crystal or resonator. Upon any system startup (from stop or reset) the MCU uses the

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
This product incorporates SuperFlash® technology licensed from SST.

Introduction

internal clock source, which eliminates the need for a long startup time. Depending on application requirements, power can be reduced by selecting the best system clock generation option. Table 1 shows the configuration considerations among the different clock modes. For more information about the clock options for the MC9S08GB/GT refer to AN2494/D, *Configuring the System Peripheral Clocks in MC9S08GB/GT*.

Table 1. ICG Configuration Consideration

	Clock Reference Source = Internal	Clock Reference Source = External
FLL Engaged	(FEI) FLL Engaged-Internal Reference $4 \text{ MHz} < f_{\text{BUS}} < 20 \text{ MHz}$. Medium power (will be less than FEE if oscillator range = high) Medium clock accuracy (After IRG is trimmed) Lowest system cost (no external components required) IRG is on. DCO is on. ⁽¹⁾	(FEE) FLL Engaged-External Reference $4 \text{ MHz} < f_{\text{BUS}} < 20 \text{ MHz}$ Medium power (will be less than FEI if oscillator range = low) Good clock accuracy Medium/High system cost (crystal, resonator or external clock source required) IRG is off. DCO is on.
FLL Bypassed	(SCM) Self-Clocked Mode This mode is mainly provided for quick and reliable system startup. $3 \text{ MHz} < f_{\text{BUS}} < 5 \text{ MHz}$ (default). $3 \text{ MHz} < f_{\text{BUS}} < 20 \text{ MHz}$ (via filter bits). Medium power Poor accuracy. IRG is off. DCO is on and open loop.	(FBE) FLL Bypassed-External Clock f_{BUS} range $\leq 8 \text{ MHz}$ when crystal or resonator is used. Lowest power Highest clock accuracy Medium/High system cost (Crystal, resonator or external clock source required) IRG is off. DCO is off.

1. The IRG typically consumes 100 μA . The FLL and DCO typically consumes 0.5 to 2.5 mA, depending upon output frequency. For minimum power consumption and minimum jitter, choose N and R to be as small as possible.

Modes of Operation

After reset, the normal mode of operation is run mode in which the CPU is active and peripherals can be enabled. By executing a WAIT instruction, the MCU enters wait mode. In wait mode, power is reduced because the CPU is not clocked. To reduce power consumption further, stop mode can be used. When a STOP instruction is executed, one of three stop modes will be entered. Stop1, stop2, and stop3 each provide different levels of operation that reduce power consumption. The table below describes stop mode behaviors.

Table 2. Stop Mode Behaviors

Mode	CPU, Digital Peripherals FLASH	RAM	ICG	ATD	KBI	Regulator	I/O Pins	RTI
Stop1	Off	Off	Off	Disabled ⁽¹⁾	Off	Off	Reset	Off
Stop2	Off	Standby	Off	Disabled	Off	Standby	States held	Optionally on
Stop3	Standby	Standby	Standby ⁽²⁾	Disabled	Optionally on	Standby	States held	Optionally on

1. Either ATD stop mode or power-down mode depending on the state of ATDPU.

2. Crystal oscillator can be configured to run in stop3. Please see the ICG registers.

Real-Time Interrupt (RTI)

The RTI can be used to exit stop2 or stop3. In stop3, it can be configured to use an external or internal reference. In stop3, using an internal reference will reduce power consumption further than using the external reference. In stop2, only the internal reference can be used. The RTI module can be configured to achieve real-time interrupts between 8 ms and 1.024 s. The 1-kHz reference has a tolerance of about $\pm 30\%$, thus the wakeup times will be approximate when the internal reference is used by the RTI.

Low-Voltage Detect (LVD)

The MC9S08GB/GT MCU has the ability to enable or disable low voltage detection when in stop3 mode. It is important to note that if low voltage detection is enabled in stop, the only stop mode that can be used is stop3. If the LVDSE bit in SPMSC1 is set, then upon execution of a STOP instruction, stop3 will be entered regardless of the state of the PDC and PPDC bits in SPMSC2.

Operating Voltage Ranges

The MC9S08GB/GT MCU is specified to operate from 3.6 V down to 1.8 V (see [Table 3](#)). For lower than 2.08 V operation, the maximum bus speed should be reduced to 8 MHz or lower. At lower voltages, the resulting power consumption will be reduced for all modes.

Table 3. DC Characteristics (Temperature = -40 to 85°C Ambient)

Parameter	Symbol	Min	Typ	Max	Unit
Supply voltage (run, wait, and stop modes)					
0 < f_{BUS} < 8 MHz	V_{DD}	1.8	—	3.6	V
0 < f_{BUS} < 20 MHz		2.08	—	3.6	

Internal Voltage Regulator

The MC9S08GB/GT uses an internal voltage regulator to provide about 2.4 V for the internal power supplies to the CPU and most peripherals. When V_{DD} falls below 2.4 V, the regulator is bypassed. This regulation keeps operating currents from rising very much as the V_{DD} rises above 2.4 V. The regulator is always on when the MCU is in run and wait modes. In stop2 and stop3 modes, the regulator is put into a state that results in looser regulation, thereby saving power. In stop1, the regulator is turned off.

Description of Low-Power Modes

Stop Modes

With the introduction of the HCS08 Family of MCUs, two new forms of stop mode were introduced, resulting in three total forms of stop. The three modes of stop are referred to as stop1, stop2, and stop3. Stop3 is functionally equivalent to stop mode on the HC08 MCUs, while stop1 and stop2 are new forms.

Description of Low-Power Modes

Stop1

Stop1 is a full power-down mode in which the internal voltage regulator is turned off. This will provide the lowest possible standby current for the MCU. In stop1, all the internal circuitry of the MCU is powered down, including the RAM and all registers. This means that all contents of the registers and RAM will be lost upon entering this mode. Also, no peripherals can be left enabled in stop1, including LVD, RTI, KBI, and ATD. No clocks can be left enabled internal to the device. Internal power to the I/O pin driver is turned off, and output pins will be pulled to ground. I/O pins will revert to high-impedance inputs.

Exiting stop1 is accomplished by asserting either the $\overline{\text{RESET}}$ pin or $\overline{\text{IRQ}}$ pin low. The IRQ will always be active low in this mode regardless of how it was configured before entering stop1. Internal pullups are automatically enabled for IRQ and $\overline{\text{RESET}}$ in stop1 mode.

Upon exiting stop1, the MCU registers will be configured as if a POR had occurred. The CPU will fetch the reset vector to begin code execution whether the $\overline{\text{RESET}}$ or $\overline{\text{IRQ}}$ pin was used to wake up the MCU.

Stop2

Stop2 will provide lower standby currents than stop3, but higher than stop1. Stop2 is a partial power-down mode in which the internal voltage regulator goes into a “loose” regulation mode, thereby reducing the current consumption by reducing the power output of the regulator.

In stop2, the RAM remains powered and the states of all the I/O pins are latched in their state prior to entering stop2. Pins configured as inputs remain inputs and output pins drive the last known state. However, all other peripherals that are powered by the voltage regulator are powered down and cannot be used, such as LVD and KBI. The ATD is also turned off and therefore cannot be used.

Although the I/O pins retain their state in stop2, all registers are powered down. Therefore, the values of any register such as SCI, timer, or port data to be preserved should be copied into RAM before entering stop2.

Exit stop2 by asserting either the $\overline{\text{RESET}}$ pin or $\overline{\text{IRQ}}$ pin low. The $\overline{\text{IRQ}}$ will always be active low in this mode no matter how it was configured before entering stop2. The $\overline{\text{IRQ}}$ pin must be enabled prior to entering stop2 mode.

In addition to the $\overline{\text{RESET}}$ or $\overline{\text{IRQ}}$ pin, in stop2, the RTI can be enabled and used for wakeup without depending on an external input. However, only the internal 1-kHz RTI oscillator can be used as the clock source for the RTI in stop2. When the RTI event occurs, stop2 is exited as if a POR has occurred.

As with stop1, exiting stop2 results in the registers resetting to their POR values with the following exception. The PPDF bit in the SPMSC2 register is set and the I/O pins remain latched in their current state until a logic 1 is written to the PPDACK bit in SPMSC2. The PPDF bit can be used as a flag to branch to a stop2 recovery routine. In order to maintain the current state of the I/O pins, copy the saved register values in RAM back into their respective locations before writing the PPDACK bit. Any register not restored will revert to its POR value and any corresponding I/O pins will also revert to their POR state. Upon stop2 recovery, normal operation of peripherals will not begin until the PPDACK has been written because the I/O will be latched.

Stop3

Stop3 in HCS08 Family devices is functionally equivalent to the stop mode on HC08 Family MCUs. The states of all I/O pins are latched in the state they were in prior to executing the stop command. In stop3, there are a couple of options that are not available in the other stop modes. Stop3 is the only stop mode where LVD protection can be enabled during stop. In fact, if the LVDSE bit in the SPMSC2 register is set, the only stop mode that can be entered is stop3.

Also, the OSCSTEN bit can be set so that the individual clock generators are enabled but the clock to the rest of the MCU is turned off. The OSCSTEN option can be used to avoid long oscillator startup times if necessary. This also allows the RTI to use an external clock source as a reference for the real-time interrupts. For time critical applications using the external reference provides for precise RTI intervals.

Exit from stop3 can be less intrusive than the exits of stop1 and stop2. If an interrupt source such as $\overline{\text{IRQ}}$, KBI, or RTI is used to exit stop3, the MCU services the interrupt and then continues operation at the instruction that follows the stop instruction. It is not necessary to initialize peripherals after exiting stop3. Stop3 can also be exited by asserting the $\overline{\text{RESET}}$ pin. In this case the MCU will fetch the reset vector and registers and peripherals will be placed in their reset state.

Wait

Wait mode consumes less power than run mode. In this mode, clocks to the CPU are turned off to reduce power. All other peripherals can be enabled in wait. In this mode, any interrupt can be used to exit wait. A common application would be to execute a WAIT command and then wait for an SCI or SPI interrupt so that operation can continue. After exit from wait via an interrupt, the MCU services the interrupt and then continues operation at the instruction that follows the WAIT command.

Using the Low-Power Modes

In order to enter any of the three stop modes, three bits in two registers must be configured properly. In the system options register (SOPT), the stop mode enable bit (STOPE) must be set to a logic 1. This register is a write-once after any reset, so care must be taken to configure the other options in the same write. If the STOPE bit is clear and an attempt to execute a STOP instruction is made, the instruction is treated as an illegal opcode and a reset is forced.

In the system power management status and control 2 register (SPMSC2), two bits, power down control (PDC) and partial power down control (PPDC) are used to determine which of the three stop modes is entered when a STOP instruction is executed.

In addition, to be able to use stop2 or stop1 mode, the LVDSE bit in SPMSC1 must be cleared. If this bit is not cleared the only stop mode that can be entered is stop3.

[Table 4](#) summarizes the source of exit and condition upon exit for each of the stop modes.

Table 4. Stop Mode Selection and Source of Exit

Mode	SPMC2		Source of Exit	Condition Upon Exit
	PDC	PDDC		
Stop1	1	0	IRQ or reset	POR
Stop2	1	1	IRQ, reset, or RTI	POR (PPDF bit set in SPMSCR)
Stop3	0	Don't Care	IRQ, reset, RTI, or KBI	If reset is used, then POR; else, normal operation continues from the interrupt vector

Stop1

When the PDC bit is set to a logic 1 and the PPDC bit is set to a logic 0, stop1 is entered upon execution of the STOP instruction. Stop1 will result in the lowest possible current drain by powering off the internal voltage regulator and all on-chip peripherals that are powered by it. Also, the I/O pins and all memories are turned off.

Stop1 is best suited for situations where power consumption is of the greatest importance and the MCU is not required to wake itself up. Only an external falling edge on either RESET or IRQ will wake the MCU up from this mode.

Since the majority of the MCU is powered down during stop1, minimal software is necessary for stop1 entry. The main concerns are to enable the STOP instruction in the SOPT register and to select stop1 with the PDC bit in the SPMSC2 register. There is no need to configure the individual peripherals since they will automatically be powered down upon entry to stop1.

Since the I/O pins are in the reset state, all I/O pins will revert to inputs and all pullups will be disabled. If the external oscillator is enabled in stop mode (OSCSTEN bit in ICG control register 1) and stop1 is entered, this bit is ignored and the clocks will be powered down.

If the LVD module is enabled in stop mode, stop1 cannot be used. Attempting to enter stop1 with the LVD enabled in stop will result in the MCU entering stop3 mode instead.

The RESET and IRQ pins will be automatically configured as wakeup pins for stop1. No software or external pullups are necessary.

Upon waking from stop1, the MCU will start up as if from a POR. Since all registers revert to their POR state and the RAM was powered down, there is no mechanism to indicate that the MCU just woke from stop1.

Since the POR results in the system bus clock being driven by an internal 4-MHz clock, the stop recovery occurs fairly quickly and allows for rapid code execution to perform any register restoration before jumping back into the normal program flow. The longest delay is the time necessary for the internal voltage regulator to turn on from the off state and stabilize. Typical delay times from the falling edge of the wakeup signal to the first instruction being executed are about 50 μ s at $V_{DD} = 3$ V and 80 μ s at $V_{DD} = 2$ V.

Stop2

When the PDC bit and the PPDC bit are both set to a logic 1, stop2 is entered upon execution of the STOP instruction. Stop2 will result in higher current consumption than stop1, but less than stop3. The RAM is kept powered on to maintain its values and the I/O pins are latched in their current state.

There are several considerations when using stop2 to ensure proper operation:

- The $\overline{\text{IRQ}}$ pin must be enabled or pulled up externally.
- The LVD must be disabled in stop (LVDSE = 0).
- If using the RTI, only the internal clock source functions in stop2.
- The OSCSTEN bit has no effect in stop2. This clock reference will always be powered down.
- Only the RAM remains powered; all other I/O registers will be reset upon wakeup.
- The PPDF flag must always be cleared before the I/O pins can be modified from their stop2 entry state.

The $\overline{\text{IRQ}}$ pin must be enabled by writing to the $\overline{\text{IRQ}}$ pin enable bit (IRQPE) in the IRQ status and control (IRQSC) register. Failure to do this will result in the MCU waking from stop2 immediately after entering stop unless an external pullup is placed on the $\overline{\text{IRQ}}$ pin. The IRQ interrupt does not need to be enabled (IRQIE bit in IRQSC).

The $\overline{\text{RESET}}$ pin will automatically be configured as a wakeup pin for stop2. No software or external pullups are necessary.

If the LVD module is enabled in stop mode, stop2 cannot be used. Attempting to enter stop2 with the LVD enabled in stop will result in the MCU entering stop3 mode instead.

When using the RTI module in stop2 as a wakeup source, the internal clock source must be used since the external clock source will not remain powered in stop2.

If the external oscillator is enabled in stop mode (the OSCSTEN bit in ICG control register 1) and stop2 is entered, this bit is ignored and the clocks will be powered down.

As always, the stop instruction must be enabled in the SOPT register and the PDC and PPDC bits in the SPMSC2 register must be set to a logic 1.

The peripherals not already mentioned do not require any special handling since they will automatically be powered down upon entry to stop2.

Stop2 is best suited for situations where the lowest possible power consumption is required, but RAM contents and I/O states must be maintained. Since the RTI module can run in stop2, the MCU can also wake up without external input.

Upon waking from stop2, the MCU will start up as if a POR had occurred. However, unlike in stop1, PPDF in the SPMSC2 register can be used to indicate that the MCU woke up from stop2 instead of a standard POR.

By using PPDF and PPDACK, the user code can save any desired register values into RAM before entering stop2 and restore these values after waking up. If the port registers are saved and restored before the PPDACK is written to a logic 1, then the I/O states will be preserved. Any port pin that is not reconfigured to its latched stop2 state will revert to its reset state. Also, any peripheral not reconfigured to its pre-stop2 state will revert to its reset state.

Using the Low-Power Modes

A typical code execution sequence for stop2 entry and exit would be:

```

; Constant declarations
IRQSCinit:    equ    $10                ; enable the IRQ pin
SOPTinit:     equ    $A0                ; enable COP and STOP
SRTISCinit:   equ    $17                ; enable int and select 1.024 sec timeout
SPMSC2init:   equ    $03                ; PDC & PPDC both set
SPMSC2st2:    equ    $07                ; PPDACK, PDC & PPDC both set
PPDFmask:     equ    $08                ; mask for PPDF bit in SPMSC2 reg
...
1) System initialization after reset
Start:        lda     SPMSC2             ; Check if coming from stop2
              and     #PPDFmask
              bne     Stop2rec           ; If so, branch to recovery code
              lda     #SOPTinit          ; Else, treat as normal POR
              sta     SOPT               ; init the System Options
              lda     #SPMSC2init
              sta     SPMSC2             ; init the SPMSC2 reg
              mov     #IRQSCinit,IRQSC   ; init the IRQ pin
...
2) Entering stop with RTI enabled
              jsr     SaveRegs
              lda     #SRTISCinit        ; Enable RTI module
              sta     SRTISC
              stop
...
3) After RTI times out, a POR will execute and code restarts at reset vector but this time PPDF
will be set

Start:        lda     SPMSC2
              and     #PPDFmask
              bne     Stop2rec
...
Stop2rec:     jsr     LoadRegs
              lda     # SPMSC2st2
              sta     SPMSC2
              bra     Main
...
; Begin Main code execution
Main:

```

Note that the constant SPMSC2st2, in addition to setting the PPDACK bit to clear the PPDF flag, also sets the PDC and PPDC bits to logic 1s. This is because these bits are write-once. Failure to set these bits to 1s in this write would result in the next STOP instruction going into stop3 mode instead of stop2. The user may, of course, elect to enable either stop1 or stop3 instead of stop2 at this point, if so desired.

As is the case with stop1, the POR results in the system bus clock being driven by an internal 4-MHz clock, and stop recovery occurs fairly quickly, allowing for rapid code execution to restore registers. The delay for the regulator to return to the full regulation state is the same as for stop1, about 50 μ s at $V_{DD} = 3$ V and 80 μ s at $V_{DD} = 2$ V.

Stop3

Though stop3 mode does not lead to the lowest possible I_{DDs} , it is very versatile and the least intrusive of all the stop modes. Stop3 is entered as long as the PDC bit in SPMSC2 is 0. Also, it is important to note that if LVD is enabled in stop or entry into background debug mode is enabled (ENBDM bit in BDCSCR

is set), the only stop mode that can be entered will be stop3. When the ENBDM bit is set and a stop instruction is executed, the system clocks to the background debug logic remain active so background debug communication is still possible.

Stop3 should be used when a user is depending on an easy exit from stop mode. Stop recovery time is typically around 100 μs when using the internal clock or the FLL. For applications that use the FLL to boost a reference frequency, stop3 has the advantage of preserving previous DCO settings when recovering from stop3 with an interrupt. This means that upon stop recovery, the DCO will be set up with the system clock configuration predefined.

Unlike the other stop modes, if stop3 is exited with an interrupt, there is no need for any initialization or reconfiguration. When the interrupt occurs, the CPU will begin processing with the stacking operations leading to the interrupt service routine. Upon the RTI command of the interrupt service routine, the CPU will resume at the instruction immediately following the stop command.

Another situation where it may be necessary to use stop3 mode is where the keyboard interrupt (KBI) module must be used. The MC9S08GB/GT has eight KBI pins — any of which can be used to wake the part from stop3. The KBI module cannot be used in stop1 or stop2, and in some applications, it may be necessary to have multiple sources for exit from stop.

Stop Recovery Times

Stop recovery times for stop1 and stop2 are very similar. In recovering from stop1 or stop2, the internal voltage regulator needs time to re-establish its 2.4 V regulation. For this reason, stop1 and stop2 recovery times are dependent on V_{DD} . If V_{DD} is 3 V, stop recovery time is approximately 50 μs . If V_{DD} is 2 V, stop recovery time is approximately 80 μs . The measurements shown in Table 5 were taken using a M68DEMO908GB60 board and the low power modes code.

Table 5. Approximate Stop Recovery Times for Stop1 and Stop2

Mode	$V_{DD} = 3 \text{ V}$	$V_{DD} = 2 \text{ V}$
Stop1	50 μs	80 μs
Stop2	50 μs	80 μs

Stop3 startup time can vary greatly depending on the clock configuration used. The minimum stop3 recovery time will be around 100 μs regardless of V_{DD} . This stop recovery time will be seen except when the ICG is configured for FLL bypassed-external clock mode (FBE). The 100 μs is a typical measurement at room temperature. For FBE mode stop3 recovery time will vary based on the external clock frequency. For example, when a 32-kHz crystal is used, stop recovery will not occur until the crystal has stabilized. From bench measurements, stop3 recovery time when a 32-kHz crystal is used can be approximately 180 ms–300 ms. If the OSCEN bit is set (oscillator enabled in stop mode) then the recovery time becomes 2.42 ms. This delay is due to a 16-cycle count and interrupt fetching overhead. If the external clock frequency is higher, stop recovery times will be reduced. Measurements of stop recovery time with a 32-kHz crystal are worst case because of the long start-up time for a low frequency crystal.

If a reset is used to exit stop3, the stop recovery time will be approximately the same as for stop1 and stop2. See the Table 6 for stop3 recovery time data.

Table 6. Stop3 Approximate Stop Recovery Time Measurements

Clock Source	Recovery Time
Internal clock used as reference	100 μ s
External 32 kHz crystal (OSCTEN = 0)	180 ms–300 ms
External 32 kHz crystal (OSCTEN = 1)	2.4 ms

Typical I_{DD} s for Specific Low-Power Modes

Table 7 and Table 8 contain data taken on an M68DEMO908GB60 board. The code following the tables can be used as a template to enter different stop modes.

Table 7. Typical I_{DD} at Room Temperature $V_{DD} = 3.12$ V

Mode	Current
Stop1	57 nA
Stop2	590 nA
Stop2 with RTI enabled	890 nA
Stop3	750 nA
Stop3 with RTI enabled	1.1 μ A
Stop3 with RTI enabled with external 32-kHz source	14.5 μ A

Table 8. Typical I_{DD} at Room Temperature $V_{DD} = 2.0$ V

Mode	Current
Stop1	17 nA
Stop2	400 nA
Stop2 with RTI enabled	700 nA
Stop3	525 nA
Stop3 with RTI enabled	850 nA
Stop3 with RTI enabled with external 32-kHz source	10.5 μ A

Creating Code to Achieve Low Power Consumption

Here are some important notes about how to create code that can achieve low power consumption. All I/O pins should be initialized to be either outputs driving low or inputs with pullups enabled. This will ensure that there is no extra current consumption due to floating inputs.

To achieve the lowest possible I_{DD} s, the LVD should be disabled in stop mode. In stop mode code is not executed, thus the risk of low voltage causing code runaway is minimal. However, this risk is increased when the RTI is used in stop2 or stop3. Stop1 and stop2 will not be entered if LVD is enabled in stop, but in stop3 the application code should ensure that the LVD is disabled in stop. The same is true for the

OSCSTEN bit. The lowest power consumption will not be reached if the oscillator is allowed to continue to run in stop mode.

In dealing with stop2 recovery, it is important to note that all registers and I/Os will be latched until the PPDACK bit is set. For example, port pins will not be able to change state until PPDACK is set.

Below is an assembly code listing of the software used to take the measurements in this application note.

Metrowerks HC08-Assembler

(c) COPYRIGHT METROWERKS 1987-2003

Rel. Loc Obj. code Source line

```

1      ;*****
2      ;      Copyright (c) Freescale 2003
3      ;*****
4      ;File name:      Low_Power_Modes.asm Current Release Level:  1.0
5      ;Last Edit Date: 06-May-03      Classification:      ES
6      ;
7      ;Include Files:  9S08GB60v1r3.inc  MC68HC9S08GB60 MCU definitions
8      ;Assembler:      CW Assembler V5.0.13      Version:      3.16
9      ;Target Device:  MC68HC9S08GB60
10     ;Documentation:   GB60 Low Power Modes AN2493
11     ;*****
12     ; Author:         Donnie Garcia
13     ; First Release:   06-May-03
14     ;
15     ; Update History:
16     ;
17     ; Rev      Date      Author  Description of Change
18     ; -----
19     ; 1.0      5-06-03      DG      Initial Release
20     ;
21     ;*****
22     ;*****
23     ; This code is used along with M68DEM0908GB60 board to demonstrate
24     ; Stop Modes
25     ; The measurements in AN2493 were taken using this code and the
26     ; Demo Board
27     ; For Measurement purposes all headers/jumpers (Except the Power_Sel
28     ; jumper) were removed from the demo board
29     ; When using Stop2 or Stop1 in order to re-establish BDM connection
30     ; PTG0/BKGD should be held low on power up, then released.
31     ;*****
32     ;      StopSelect and WakeSelect are used to configure the code
33     ; To test Stop1
34     ; StopSelect = %00000001 WakeSelect = Dont Care
35     ;
36     ; To test Stop2
37     ; StopSelect = %00000010 WakeSelect = %00000000
38     ;
39     ; To test Stop2 RTI
40     ; StopSelect = %00000010 WakeSelect = %00000010
41     ;
42     ; To test Stop3
43     ; StopSelect = %00000100 WakeSelect = %00000001
44     ;
45     ; To test Stop3 RTI Internal

```

Typical I_{DD}s for Specific Low-Power Modes

```

46      ;* StopSelect = %00000100 WakeSelect = %00000010
47      ;*
48      ;* To test Stop3 RTI External
49      ;* StopSelect = %00000100 WakeSelect = %00000100
50      ;*****
51      include "gS08GB60v1r3.inc"
52      ;*****
53      ;SELECT STOP MODE AND WAKE UP SOURCE HERE
54      ;
55      0000 0004 StopSelect: equ      %00000100 ;Select Stop Mode Here
56      ;
57      ;          |||
58      ;          ||+---Stop1 Mode selected
59      ;          |+---Stop2 Mode selected
60      ;          +----Stop3 Mode Selected
61      ;If more than 1 mode is selected the lowest stop mode will be set
62      ;If no selection is made stop3 is chosen
63      0000 0002 WakeSelect: equ      %00000010 ;Select Method of wake up (Stop2,3)
64      ;
65      ;          |||
66      ;          ||+---KeyBoard wake selected (For Stop3)
67      ;          ||+---RTI Internal wake selected
68      ;          +----RTI external wake Selected (For Stop3)
69      ;If No selection is made KBI is selected
70      ;*****
71
72      ;IMPORTANT REGISTER INITS
73      ;
74      0000 0014 initSPMSC1: equ      %00010100 ;Disable LVD in stop
75      ;
76      ;          |||||
77      ;          |||||
78      ;          |||||+----LVDE Enable LVD
79      ;          ||||+----LVDS Disable LVD in stop
80      ;          |||+-----LVDSRE Enable LVD reset protection
81      ;          ||+-----LVDIE
82      ;          |+-----LVDACK
83      ;          +-----LVDF
84
85      0000 0000 initSPMSC2: equ      %00000000 ;This register sets stop mode
86      ;
87      ;          |||||
88      ;          |||||+--PPDC
89      ;          |||||+--PDC
90      ;          |||||+---PPDACK
91      ;          |||+----PPDF
92      ;          ||+-----LVWV
93      ;          |+-----LVDV
94      ;          +-----LVWACK
95      ;          +-----LVWF
96
97
98      0000 0063 initSOPT:   equ      %01100011 ;COP and STOP enable controls
99      ;
100     ;          |||  ||
101     ;          |||  |+--RSTPE --- Reset pin enabled
102     ;          |||  +--BKGDPE -- BKGD pin enabled
103     ;          ||+-----STOPE --- STOP allowed
104     ;          |+-----COPT ---- long timeout 2^18
105     ;          +-----COPE ---- COP off

```

```

105
106
107      0000 003C   initICGC1:   equ      %00111100 ;Clock Generator Control 1
108                      ;                      0|||||xx      this setting for 32 kHz xtal
109                      ;                      |||+---OSCSTEN - keep osc on in stop mode
110                      ;                      ||+----CLKS0  - select FLL engaged external
111                      ;                      |+-----CLKS1 /    (FEE) mode
112                      ;                      |+-----REFS ---- enable oscillator amplifier
113                      ;                      +-----RANGE --- 32.768 kHHz crystal
114
115
116      0000 0021   initICGC2:   equ      %00100001 ;Clock Generator Control 2
117                      ;                      |||||      should write MFDx before ICGC1
118                      ;                      |||||+---RFD0 \
119                      ;                      |||||+---RFD1 --- post-PLL divider
120                      ;                      |||||+---RFD2 *
121                      ;                      |||+----LOCRES --- loss of clock doesn't reset
122                      ;                      ||+-----MFD0 \
123                      ;                      |+-----MFD1 --- FLL loop multiplier N
124                      ;                      +-----MFD2 *
125                      ;                      +-----LOLRE --- loss of lock doesn't reset
126
127
128      0000 0000   LED           equ      0
129                      ;*****
130                      org      RamStart
131
132      0080       StopSet        RMB      1          ;Used to select stop mode
133      0081       WakeSet        RMB      1          ;Used to select Wait Mode
134      0082       PTFD_STORE     RMB      1          ;Used to store PTF
135
136
137                      org      RomStart
138
139      START:
140
141      1080 AE 63              ldx      #initSOPT
142      1082 CF 1802           stx      SOPT          ;Disable COP and enable STOP
143
144      1085 AE 14              ldx      #initSPMSC1
145      1087 CF 1809           stx      SPMSC1       ;Disable LVD in stop
146
147      108A C6 180A           lda      SPMSC2       ;how did we get here?
148      108D A4 08             and      #mPPDF       ;was it a wake-up from STOP2?
149      108F 26 1C             bne      Stop2Recovery ;If = 0 was normal reset
150                      ;If = 1 Stop2 Recovery is needed
151
152                      ;This begins the path of a normal reset (Not stop2 recovery)
153
154      INIT:
155
156                      ;*****
157                      ;FIRST setup SPMSC2 to to the proper stop mode
158      1091 A6 04              lda      #StopSelect
159      1093 B7 80              sta      StopSet
160
161      1095 00 80 0E           brset   0,StopSet,Set_Stop1
162      1098 02 80 02           brset   1,StopSet,Set_Stop2
163

```

Typical I_{DD}s for Specific Low-Power Modes

```

164                               Set_Stop3:
165 109B 20 10                   bra      StopSelectDone ;Reset state of SPMSC2 selects stop3
166
167                               Set_Stop2:
168 109D B6 00                   lda      initSPMSC2 ;enable stop2
169 109F AA 03                   ora      #(mPDC|mPPDC)
170 10A1 C7 180A                sta      SPMSC2
171
172
173                               bra      StopSelectDone
174
175                               Set_Stop1:
176 10A6 B6 00                   lda      initSPMSC2 ;enable stop1
177 10A8 AA 02                   ora      #(mPDC)
178 10AA C7 180A                sta      SPMSC2
179
180
181                               StopSelectDone:
182
183                               ;*****
184                               Stop2Recovery:                ;Initialize before PDACK
185                               ;*****
186                               ;Now set up the selected wakeup source
187 10AD A6 02                   lda      #WakeSelect
188 10AF B7 81                   sta      WakeSet
189 10B1 00 81 08                brset   0,WakeSet,InitKBI
190 10B4 02 81 0F                brset   1,WakeSet,InitRTIint
191 10B7 04 81 13                brset   2,WakeSet,InitRTIext
192 10BA 20 1F                   bra      WakeSelectDone
193
194                               InitKBI:
195 10BC 18 17                   bset    KBIPE4,KBIPE ;Enable Keyboard Pin
196 10BE 12 16                   bset    KBIE,KBISC ;Enable Keyboard Interrupts
197 10C0 14 16                   bset    KBACK,KBISC ;Clear Pending Keyboard Interrupts
198 10C2 18 01                   bset    PTAPE4,PTAPE ;Enable Pullup for Keyboard pin
199 10C4 20 15                   bra      WakeSelectDone
200
201                               InitRTIint
202 10C6 AE 17                   ldx     #$17 ;Enable RTI Interrupts 1s timeout
203 10C8 CF 1808                stx     SRTISC
204                               bra      WakeSelectDone
205                               InitRTIext
206 10CD AE 37                   ldx     #$37 ;External clock bit set
207                               ;Enable RTI Interrupts long timeout
208 10CF CF 1808                stx     SRTISC
209 10D2 6E 21 49                mov     #initICGC2,ICGC2 ;sets MFD divider
210 10D5 6E 3C 48                mov     #initICGC1,ICGC1 ;32.768 kHz -> 4.166MHz bus rate
211 10D8 07 4A FD                brclr   LOCK,ICGS1,*
212                               WakeSelectDone
213                               ;*****
214                               ;Initialize all I/O to achieve Low Power
215                               Init_IO
216
217 10DB 6E 10 14                mov     #mIRQPE,IRQSC ;pull-up and enable IRQ
218                               ;Make All unused I/O Outputs Driving low
219 10DE 6E EF 03                mov     #$EF,PTADD ;ADDR
220 10E1 6E FF 07                mov     #$ff,PTBDD ;BDDR
221 10E4 6E FF 0B                mov     #$ff,PTCDD ;CDDR
222 10E7 6E FF 0F                mov     #$ff,PTDDD ;DDDR

```

```

223 10EA 6E FF 13      mov    #$ff,PTEDD ;EDDR
224 10ED 6E FF 43      mov    #$ff,PTFDD ;FDDR
225 10F0 6E FF 47      mov    #$ff,PTGDD ;GDDR
226
227 10F3 6E 00 00      mov    #$00,PTAD ;ADR
228 10F6 6E 00 04      mov    #$00,PTBD ;BDR
229 10F9 6E 00 08      mov    #$00,PTCD ;CDR
230 10FC 6E 00 0C      mov    #$00,PTDD ;DDR
231 10FF 6E 00 10      mov    #$00,PTED ;EDR
232 1102 6E 01 40      mov    #$01,PTFD ;FDR
233 1105 6E 00 44      mov    #$00,PTGD ;GDR
234
235 1108 C6 180A        lda     SPMSC2 ;how did we get here?
236 110B A4 08          and     #mPPDF ;was it a wake-up from STOP2?
237 110D 27 13          beq     MainLoop
238
239 110F 4E 82 40        mov    PTFD_STORE,PTFD ;Replace PTF with stored info
240 1112 C6 180A        lda     SPMSC2 ;acknowledge Stop2 recovery
241 1115 AA 07           ora     #(mPPDACK|mPDC|mPPDC)
242 1117 C7 180A        sta     SPMSC2
243
244 111A B6 40           lda     PTFD ;Toggle LED here for Stop2
245 111C A8 01           eor     #mPTFD0
246 111E B7 40           sta     PTFD
247 1120 2E FE          bil     * ;Wait while IRQ is low (Debounce)
248
249                                MainLoop
250 1122 B6 40           lda     PTFD
251 1124 B7 82           sta     PTFD_STORE ;Store PTF state into RAM
252 1126 8E              stop
253 1127 20 F9          bra     MainLoop
254
255 ;*****Interrupt Service Routines*****
256 kbi_isr
257 1129 14 16          bset    KBACK,KBISC ;Acknowledge KB Interrupt
258 112B B6 40          lda     PTFD ;Toggle LED Here
259 112D A8 01          eor     #mPTFD0
260 112F B7 40          sta     PTFD
261 1131 80             rti
262 rti_isr
263 1132 C6 1808        lda     SRTISC
264 1135 AA 40          ora     #mRTIACK
265 1137 C7 1808        sta     SRTISC ;Acknowledge RTI Interrupt
266 113A B6 40          lda     PTFD ;Toggle LED Here
267 113C A8 01          eor     #mPTFD0
268 113E B7 40          sta     PTFD
269 1140 80             rti
270
271 ;*****Vectors*****
272 org     Vrti
273 FFCC 1132           fdb     rti_isr
274 org     Vkeyboard
275 FFD2 1129           fdb     kbi_isr
276 org     Vreset
277 FFFE 1080           fdb     START

```


Conclusion

The MC9S08GB/GT can be configured in a variety of ways to achieve low power consumption. The three different stop modes offer a variety of solutions for a user's application. The I_{DD} data shows that in all stop modes, operational current can be kept very low. The flexibility of the different stop modes combined with other features such as RTI, operating voltage range and clock configuration options make the MC9S08GB/GT ideal in low-power applications.

Below is a summary of the features of each of the low-power modes:

- Stop1 is the complete power-down mode with 20 nA typ at 2 V
 - All RAM and register content is lost
 - Exit stop1 with IRQ or reset; exit requires an external event
- Stop2 is partial power-down mode with 400 nA typ at 2 V
 - RAM contents are maintained with I/O states latched
 - Exit with IRQ, reset, or internal auto wakeup timer
 - Requires initialization of any peripheral used
- Stop3 is equivalent to M68HC08 MCU's stop mode with 500 nA typ at 2 V
 - Exit with IRQ, KBI, LVD, internal auto wakeup timer, or reset
 - Allows clock generators to be enabled but not driven to peripherals so external clock references can be used
 - Does not require initialization of peripherals
- Wait mode disables the clock to the CPU but can clock peripherals with 420 μ A typ at 1 MHz and 2 V (FBE)
 - Typically used when waiting for an interrupt such as an SPI receive interrupt
 - Immediate processing of interrupt service routine
- Run mode with 640 μ A typ at 1 MHz and 2 V (FBE)

Two zip files accompany this application note, AN2493SW1.zip and AN2493SW2.zip. AN2493SW1.zip contains just the assembly and include file for the low-power modes software.

AN2493SW2.zip contains the complete CodeWarrior project folder. Inside the first level of the project folder is a CodeWarrior project file with a ".mcp" filename extension. Double-clicking this file will open the project if CodeWarrior has been installed. The project has been assembled and the listing (".lst" file extension) is available in the "bin" subfolder. Also, the s record (".s19" file extension) is available in the same folder.

How to Reach Us:

Home Page:

www.freescale.com

E-mail:

support@freescale.com

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor 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 and all 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 products 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 the 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 and hold 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 claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

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

© Freescale Semiconductor, Inc. 2004. All rights reserved.