Section 18 Power-Down State

18.1 Overview

The H8/532 has a power-down state that greatly reduces power consumption by stopping the CPU functions. The power-down state includes three modes:

1. Sleep mode— a software-triggered mode in which the CPU halts but the rest of the chip remains active
2. Software standby mode— a software-triggered mode in which the entire chip is inactive
3. Hardware standby mode— a hardware-triggered mode in which the entire chip is inactive

The sleep mode and software standby mode are entered from the program execution state by executing the SLEEP instruction under the conditions given in table 18-1. The hardware standby mode is entered from any other state by a Low input at the $\overline{\text{STBY}}$ pin.

Table 18-1 lists the conditions for entering and leaving the power-down modes. It also indicates the status of the CPU, on-chip supporting modules, etc., in each power-down mode.

Table 18-1 Power-Down State

Mode	Entering Procedure	Clock	CPU	CPU Reg's.	Sup. Mod's.	RAM	I/O Ports	Exiting Methods
Sleep mode	Execute SLEEP instruction	Run	Halt	Held	Run	Held	Held	• Interrupt • $\overline{\text{RES}}$ Low • $\overline{\text{STBY}}$ Low
Software standby mode	Set SSBY bit in SBYCR to 1, then execute SLEEP instruction*	Halt	Halt	Held	Halt and partly initialized	Held	Held	• NMI • $\overline{\text{RES}}$ Low • $\overline{\text{STBY}}$ Low
Hardware standby mode	Set $\overline{\text{STBY}}$ pin to Low level	Halt	Halt	Not held	Halt and partly initialized	Held	High impedance state	• $\overline{\text{STBY}}$ High, then $\overline{\text{RES}}$ Low → High

* The watchdog timer must also be stopped.

Notes: SBYCR Software standby control register
SSBY Software standby bit

18.2 Sleep Mode

18.2.1 Transition to Sleep Mode

Execution of the SLEEP instruction causes a transition from the program execution state to the sleep mode. After executing the SLEEP instruction, the CPU halts, but the contents of its internal registers remain unchanged. The functions of the on-chip supporting modules do not stop in the sleep mode.

18.2.2 Exit from Sleep Mode

The chip wakes up from the sleep mode when it receives an internal or external interrupt request, or a Low input at the $\overline{\text{RES}}$ or $\overline{\text{STBY}}$ pin.

1. **Wake-Up by Interrupt:** An interrupt releases the sleep mode and starts either the CPU's interrupt-handling sequence or the data transfer controller (DTC).

   If the interrupt is served by the DTC, after the data transfer is completed the CPU executes the instruction following the SLEEP instruction, unless the count in the data transfer count register (DTCR) is 0.

   If an interrupt on a level equal to or less than the mask level in the CPU's status register (SR) is requested, the interrupt is left pending and the sleep mode continues. Also, if an interrupt from an on-chip supporting module is disabled by the corresponding enable/disable bit in the module's control register, the interrupt cannot be requested, so it cannot wake the chip up.

2. **Wake-Up by $\overline{\text{RES}}$ pin:** When the $\overline{\text{RES}}$ pin goes Low, the chip exits from the sleep mode to the reset state.

3. **Wake-Up by $\overline{\text{STBY}}$ pin:** When the $\overline{\text{STBY}}$ pin goes Low, the chip exits from the sleep mode to the hardware standby mode.

18.3 Software Standby Mode

18.3.1 Transition to Software Standby Mode

A program enters the software standby mode by setting the standby bit (SSBY) in the software standby control register (SBYCR) to 1, then executing the SLEEP instruction. Table 18-2 lists the attributes of the software standby control register.

Table 18-2 Software Standby Control Register

Name	Abbreviation	R/W	Initial Value	Address
Software standby control register	SBYCR	R/W	H'7F	H'FFFB

In the software standby mode, the CPU, clock, and the on-chip supporting module functions all stop, reducing power consumption to an extremely low level. The on-chip supporting modules and their registers are reset to their initial state, but as long as a minimum necessary voltage supply is maintained (at least 2V), the contents of the CPU registers and on-chip RAM remain unchanged. The I/O ports also remain in their current states.

18.3.2 Software Standby Control Register (SBYCR)

Bit	7	6	5	4	3	2	1	0
	SSBY	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

The software standby control register (SBYCR) is an 8-bit register that controls the action of the SLEEP instruction.

Bit 7—Software Standby (SSBY): This bit enables or disables the transition to the software standby mode.

Bit 7 SSBY	Description
0	The SLEEP instruction causes a transition to the sleep mode. (Initial value)
1	The SLEEP instruction causes a transition to the software standby mode.

The watchdog timer must be stopped before the chip can enter the software standby mode. To stop the watchdog timer, clear the timer enable bit (TME) in the watchdog timer's timer control/status register (TCSR) to 0. The SSBY bit cannot be set to 1 while the TME bit is set to 1.

When the chip is recovered from the software standby mode by a nonmaskable interrupt (NMI), the SSBY bit is automatically cleared to 0. It is also cleared to 0 by a reset or transition to the hardware standby mode.

Bits 6 to 0—Reserved: These bits cannot be modified and are always read as 1.

18.3.3 Exit from Software Standby Mode

The chip can be brought out of the software standby mode by an input at one of three pins: the NMI pin, $\overline{\text{RES}}$ pin, or $\overline{\text{STBY}}$ pin.

1. **Recovery by NMI Pin:** When an NMI request signal is received, the clock oscillator begins operating but clock pulses are supplied only to the watchdog timer (WDT). The watchdog timer begins counting from H'00 at the rate determined by the clock select bits (CKS2 to CKS0) in its timer status/control register (TCSR). This rate should be set slow enough to allow the clock oscillator to stabilize before the count reaches H'FF. When the count overflows from H'FF to H'00, clock pulses are supplied to the whole chip, the software standby mode ends, and execution of the NMI interrupt-handling sequence begins.

   The clock select bits (CKS2 to CKS0) should be set as follows.

   (1) Crystal oscillator: Set CKS2 to CKS0 to a value that makes the watchdog timer interval equal to or greater than 10ms, which is the clock stabilization time.

   (2) External clock input: CKS2 to CKS0 can be set to any value. The minimum value (CKS2 = CKS1 = CKS0 = 0) is recommended.

2. **Recovery by $\overline{\text{RES}}$ Pin:** When the $\overline{\text{RES}}$ pin goes Low, the clock oscillator starts. Next, when the $\overline{\text{RES}}$ pin goes High, the CPU begins executing the reset sequence.

   When the chip recovers from the software standby mode by a reset, clock pulses are supplied to the entire chip at once. Be sure to hold the $\overline{\text{RES}}$ pin Low long enough for the clock to stabilize.

3. **Recovery by $\overline{\text{STBY}}$ Pin:** When $\overline{\text{STBY}}$ the pin goes Low, the chip exits from the software standby mode to the hardware standby mode.

18.3.4 Sample Application of Software Standby Mode

In this example the chip enters the software standby mode on the falling edge of the NMI input and recovers from the software standby mode on the rising edge of NMI. Figure 18-1 shows a timing chart of the transitions.

The nonmaskable interrupt edge bit (NMIEG) in the port 1 control register (P1CR) is originally cleared to 0, selecting the falling edge as the NMI trigger. After accepting an NMI interrupt in this condition, software changes the NMIEG bit to 1, sets the SSBY bit to 1, and executes the SLEEP instruction to enter the software standby mode. The chip recovers from the software standby mode on the next rising edge at the NMI pin.

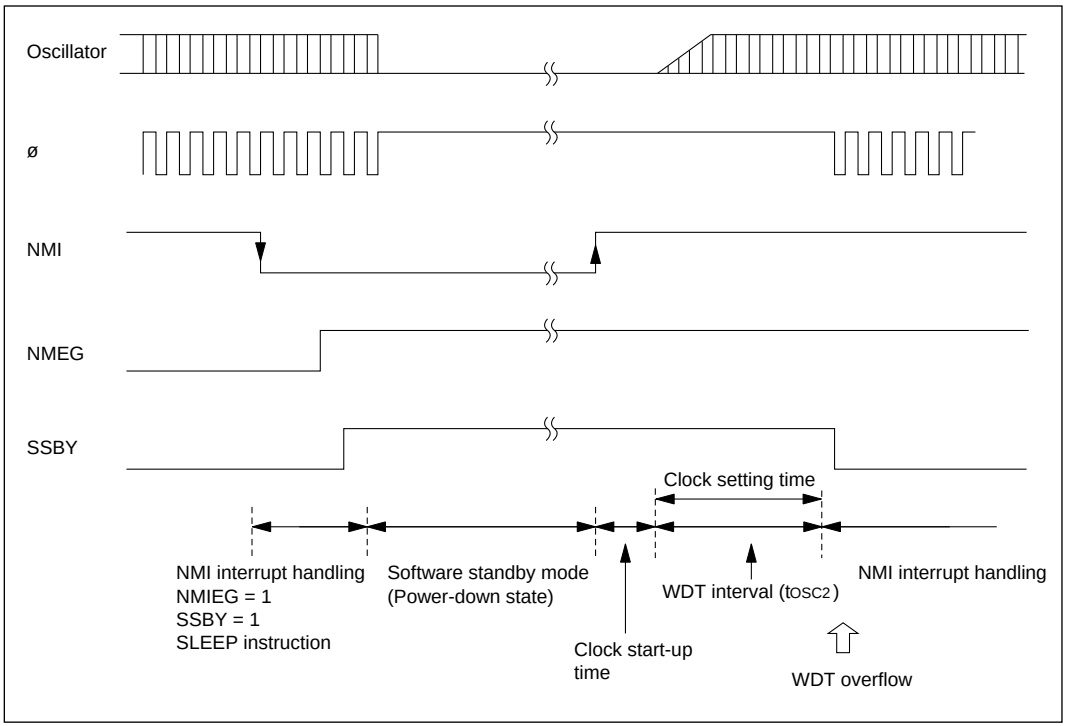

Figure 18-1 NMI Timing of Software Standby Mode (Application Example)

18.3.5 Application Notes

(1) The I/O ports retain their current states in the software standby mode. If a port is in the High output state, its output current is not reduced in the software standby mode.

(2) If the software standby mode is entered under either condition ① or condition ② below in a ZTAT version of the H8/532, current dissipation is greater than in normal standby mode (I_{CC} = 100 to 300µA). This problem does not occur in H8/532 versions with masked ROM.

① In single-chip mode (mode 3): if software standby mode is entered after even one instruction not stored in on-chip ROM has been fetched (e.g. from on-chip RAM).

② In expanded mode with on-chip ROM enabled (mode 2): if software standby mode is entered after even one instruction not stored in on-chip ROM has been fetched (e.g. from external memory or on-chip RAM).

This problem does not occur in the expanded mode when on-chip ROM is disabled (mode 1).

In applications in which the additional standby current must be avoided, take one of the following actions:

- Store program code only in on-chip ROM.
- Use the hardware standby mode. There is never any additional current in hardware standby mode.

18.4 Hardware Standby Mode

18.4.1 Transition to Hardware Standby Mode

Regardless of its current state, the chip enters the hardware standby mode whenever the $\overline{\text{STBY}}$ pin goes Low.

The hardware standby mode reduces power consumption drastically by halting the CPU, stopping all the functions of the on-chip supporting modules, and placing I/O ports in the high-impedance state.

The registers of the on-chip supporting modules are reset to their initial values. Only the on-chip RAM is held unchanged, provided the minimum necessary voltage supply is maintained (at least 2V).*

Notes: 1 The RAME bit in the RAM control register should be cleared to 0 before the $\overline{\text{STBY}}$ pin goes Low, to disable the on-chip RAM during the hardware standby mode.
2 Do not change the inputs at the mode pins (MD2, MD1, MD0) during hardware standby mode. Be particularly careful not to let all three mode inputs go low, since that would place the chip in PROM mode, causing increased current dissipation.

18.4.2 Recovery from Hardware Standby Mode

Recovery from the hardware standby mode requires inputs at both the $\overline{\text{STBY}}$ and $\overline{\text{RES}}$ pins.

When the $\overline{\text{STBY}}$ pin goes High, the clock oscillator begins running. The $\overline{\text{RES}}$ pin should be Low at this time and should be held Low long enough for the clock to stabilize. When the $\overline{\text{RES}}$ pin changes from Low to High, the reset sequence is executed and the chip returns to the program execution state.

18.4.3 Timing Sequence of Hardware Standby Mode

Figure 18-2 shows the usual sequence for entering and leaving the hardware standby mode.

First the $\overline{\text{RES}}$ pin goes Low, placing the chip in the reset state. Then the $\overline{\text{STBY}}$ pin goes Low, placing the chip in the hardware standby mode and stopping the clock. In the recovery sequence first the $\overline{\text{STBY}}$ pin goes High; then after the clock stabilizes, the $\overline{\text{RES}}$ pin is returned to the High level.

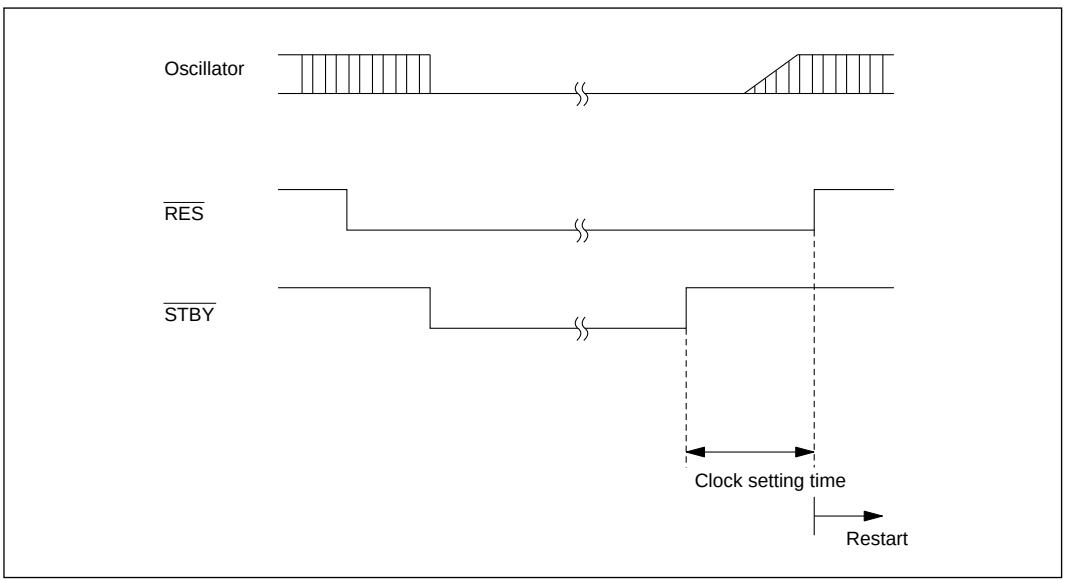

Figure 18-2 Hardware Standby Sequence