Application Note

Low Power Modes

of MB91F361

History

13th Oct. 99	MM	V1.0	New Format, new updated version
05th Jul. 00	MEN	V1.1	Updated Application

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, application Notes, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment**.

1. Fujitsu Mikroelektronik GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Mikroelektronik GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

2. Should a Product turn out to be defect, Fujitsu Mikroelektronik GmbH´s entire liability and the customer´s exclusive remedy shall be, at Fujitsu Mikroelektronik GmbH´s sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Mikroelektronik GmbH in original packing and without further defects resulting from the customer´s use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Mikroelektronik GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Mikroelektronik GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Mikroelektronik GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.

4. To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH´s and its suppliers´ liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Mikroelektronik GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

0 Introduction

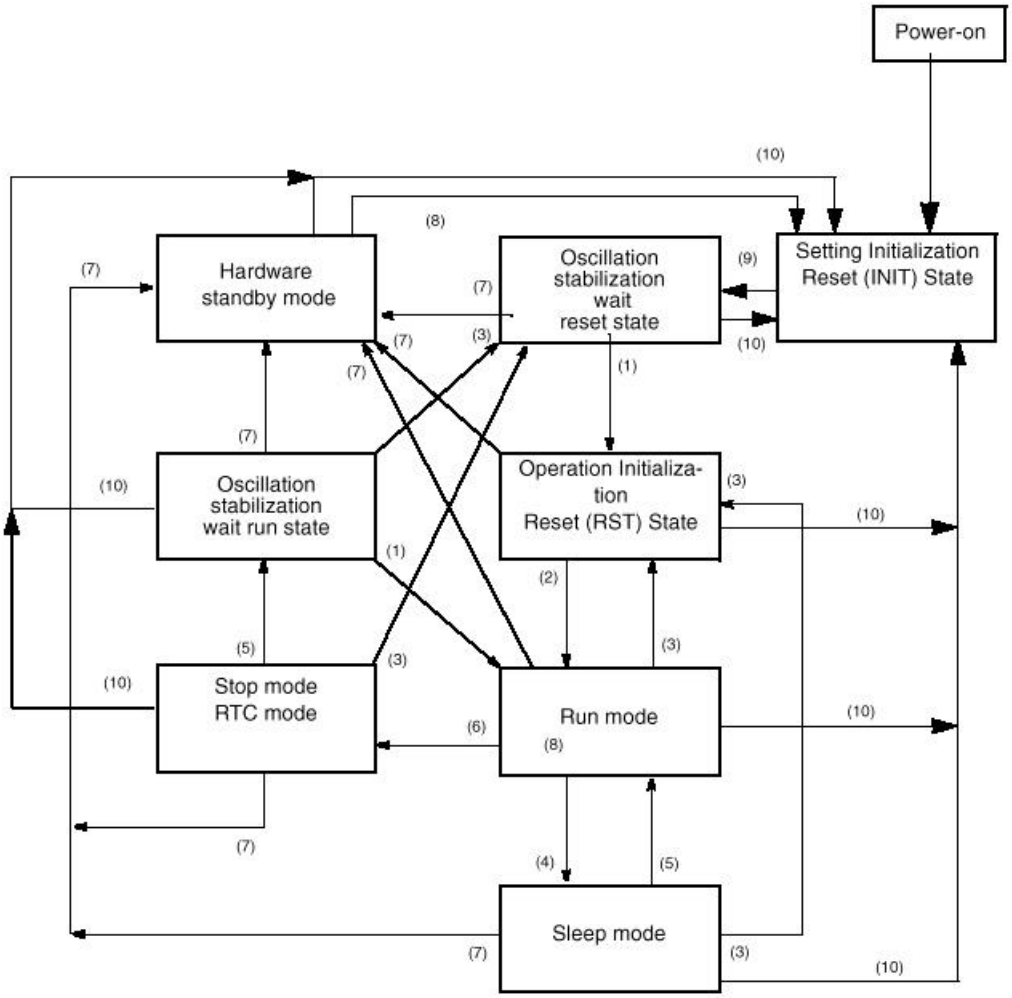
This application note shows how to set the MB91F361 (or related devices) into a certain low-power mode. The available modes are :

- Run : The normal operation mode (default after wake-up from reset or any low-power mode). CPU-core, all peripherals and the clock are active. Note that the R-Bus (16-Bit peripheral bus) stops whenever there is no access to any peripheral register.
- Sleep : The CPU-core will be switched off (program execution is stopped), but peripherals and clock-supply will continue to run.
- RTC : In this mode, only the Real-time clock (RTC) and the clock supply are alive. The CPU-core and the rest of the peripherals are switched off.
- Stop : Everything will be shut down. Even the oscillator stops. Wakeup from this mode is only possible by external interrupts by high-level detection.
- Hardware Standby : This mode can only be entered by applying a low-level to the external HSTX-pin. It has similar properties as the Stop-State.

To make a transition from Run to any of the above low-power modes, the user has to take some defined steps in order to avoid unwanted effects (e.g. the device will be woken up again immediately after entering sleep or stopmode). Also, to wake-up the device again from a low-power mode, the user should be aware of the conditions.

1 Transitions

Diagram 1 shows the different device states and the possible transitions between them. Note that only some of then can be reached by software interactions.



(1) Oscillation stabilization delay time complete

(2) Wakeup from reset

(3) Reset (RST)

(4) SLEEP= “1” in the STCR register

(5) Interrupt input

(6) STOP= “1” in the STCR register :
OSCD1=”0” causes transition to RTC mode (a) ,OSCD1 =”1” causes transition to STOP mode (b)

(7) Hardware standby input

(8) Wakeup from hardware standby

(9) Release of INITX

(10) Reset (INIT)

Diagram 1 : CPU-States and transitions of MB91F361

2 Software Settings

To enter one of the low-power modes, the following settings has to be done in Software :

From Run to Sleep Mode (4) :

1. Disable all interrupt except the ones that you want to use as wakeup condition. Make sure that no interrupt requests are currently pending at this time (no Interrupt flags set in the resouce-registers).
2. Set the Sleep-Bit in the Standby Control Register (`STCR_SLEEP = 1;`)

From Run to RTC Mode (6a) :

1. Disable all interrupt except the ones that you want to use as wakeup condition. Make sure that no interrupt request are currently pending at this time (no Interrupt flags set in the resouce-registers).
2. Reset the Oscillator Disable Bit in the Standby Control Register (`STCR_OSCD1 = 0;`)
3. Set the Stop-Bit in the Standby Control Register (`STCR_STOP = 1;`)

From Run to Stop Mode (6b) :

1. Disable all interrupt except the ones that you want to use as wakeup condition. Make sure that no interrupt request are currently pending at this time (no Interrupt flags set in the resouce-registers).
2. Determine the Pin-State in Stop-mode using the HIZ-Bit in the Standby Control Register (1 means all Pins go into input mode (high impedeance) ; 0 means the current pin state is maintained in Stop mode)
3. Disable the PLLs by switching to Oscillator/2 in the CLKR Register.
4. Set the Oscillator Disable Bit in the Standby Control Register (`STCR_OSCD1 = 1;`)
5. Set the Stop-Bit in the Standby Control Register (`STCR_STOP = 1;`)

Note : If you want to reduce power consumption, disable the PLL (switch to Oscillation/2) as an option before switching to Sleep or RTC mode.

3 Wakeup

Wakeup from Sleepmode is simply done by the acceptance of an interrupt (5). The interrupt must be enabled and accepted by the interrupt controller in terms of priority and level. If the PLL was not switched off, no further preparation is necessary. The CPU continues operation after `STCR_SLEEP = 1;` .

Wakeup from RTC mode can be initiated by an interrupt of the Real Time Clock or one of the external interrupts (5).

Wakeup from Stop mode can only be done by an external interrupt at high level detection(5). You have to re-enable the PLL in this case. Make sure you have included the wait loop for the PLL lock time (see Startup.Asm as an example).

Notes :

A detection of any higher interrupt causes (Reset) is always possible and causes the transitions (3), (7) or (10).

The user can choose between synchronous and asynchronous standby by the appropriate setting of the `TBCR_SYNCR` bit. This function allows entering a low-power mode only, when all accesses to internal registers are finished. This guarantees to have a consistant memory contents. However, when using this function, make sure the CPU can execute some instructions (`NOPs`) to let any memory or register access finish in the background. See the example project "`LOWPOWER.PRJ`" for details.

For further details refer to the MB91360 Series Hardware-Manual chapter "5.10 Device State Control".

4 Sample Software

This is the main routine of LOWPOWER.PRJ :

```
void main(void)
{
        int i;
        setPLL(PLLFREQ);      /* acivate PLL clock */
        LCDinitdisp();

        ICR00 = 20;                  /* set interrupt control register */
        EIRR_ER0 = 0;                /* clear old and currently not def. int-request */
        ICR01 = 21;                  /* set interrupt control register */
        EIRR_ER1 = 0;                /* clear old and currently not def. int-request */

        __EI();                      /* enable interrupts */
        __set_il(30);                /* set global int-level */

        choose = 1;
        while(1)
        {
           LCDprint("Choose mode !");
            ENIR_EN0 = 0;                    /* disable int0 */
            ELVR = 0x000F;                   /* falling edges for selection */
            EIRR = 0;
            ENIR_EN1 = 1;                    /* enable int1 */

            while (PDRK_PDK0 == 1);    /* do selection until USER0 pressed */

            ENIR_EN1 = 0;                    /* disable int1 */
            EIRR = 0;
            ELVR = 0x0001;                   /* high level for wake-up */
            ENIR_EN0 = 1;                    /* enable int0 */

            CLKR &= 0xfc;                    /* switch clock source to X-clock (2MHz) */
            CLKR &= 0xf8;                    /* turn PLLs off */

            TBCR_SYNCS = 0;     /* SYNC  0:ordanary standby  0:asynchr. standby */
```

```
        switch(choose)
        {
              case 1 :
                    LCDprint("STOP-mode");
                    STCR_OSCD1 = 1;
                    STCR_HIZ = 1;
                    STCR_STOP = 1;                        /* go to STOP-mode now */
                    break;
              case 2 :
                    LCDprint("SLEEP-mode");
                    STCR_SLEEP = 1;                       /* go to SLEEP-mode now */
                    break;
              case 3 :
                    LCDprint("RTC-mode");
                    STCR_OSCD1 = 0;
                    STCR_HIZ = 1;
                    STCR_STOP = 1;               /* go to RTC-mode now */
                    break;

            }

            #pragma asm                           /* 6 NOPs for pipeline */
                    nop
                    nop
                    nop
                    nop
                    nop
                    nop
            #pragma endasm

         setPLL(PLLFREQ);   /* activate PLL again */


      }

}
```