Application Note

FUJITSU

Procedure to switch the PLL

MB91F365G,MB91F366G

MB91F367G,MB91F368G

MB91F362,MB91F361G

History

20th Nov. 00	AG	V1.0	started
25th. May 01	AG	V1.1	measurement results added
13th Aug. 01	AG	V1.2	sequence optimised clock modulator switched off FWMT set to 1 wait state for flash access when oscillator is clock source

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.

.

Contents

1. Concerned devices

The following description is concerned to these devices:
MB91F361G,
MB91F362,
MB91F365G,
MB91F366G,
MB91F367G,
MB91F368G

2. The obesrved behavior

The MB91360- devices have an voltage regulator inernal. This regulator generates the voltage of 3.3 V for the core. The core needs a voltage in the specified range for a correct functionality. Changing the clock settings can cause a power consumption the voltage regulator is not able to supply with a voltage level in the specified range.
A core voltage outside the specified range can consequence to a system crash. There are two events which can induce an unspecified level of the internal core voltage on the VCC3C pin of the MCU:

- enabling PLL as clock source
- disabling PLL as clock source

Possible situation when clock settings can be changed:

- at startup (configure and select PLL as clock source)
- before entering low power modes (RTC mode, sleep mode)
- after returning from low power modes (RTC mode, SLEEP mode) to RUN mode clock settings are reconfigured

3. The cause of the voltage variation

The internal voltage regulator generates the 3V core voltage form the 5V supply. During enabling/disabling the PLL as clock source the regulator can't guaranty a constant core voltage of 3V. The voltage drops or exceeds the specified limits of the voltage range obviously. In that case a correct execution of the appplication can't be guarantied.
Therefore it's necessary to keep the core voltage inside the specified range.

4. Method to reduce the voltage variation

The intensity of the voltage variation depends on the power consumption of the whole controller at the moment of changing the clock source settings. Then lower the current power consumtion at the critical moment then lower is the intensity of the voltage variation. It isn’t possible to avoid the voltage variation complete. But the intensity of the voltage variation can be reduced and be kept inside the specified limits. The absolute value of the voltage variation is influenced by the frequency of internal components and what components are running at the moment of enabling or disabling the PLL as clock source. Then more components of the controller are disabled at the moment of changing the clock source then lower is the intensity of the voltage variation.

So we propose the following to take care when the clock settings are changed:

- reduction of frequency of external and peripheral bus before settings are changed
- entering to sleep mode at the moment PLL is enabled/disabled as clock source

FME provides a routine should be used to change the clock settings. Figure 1 shows the flowchart of this routine.

change PLL settings :

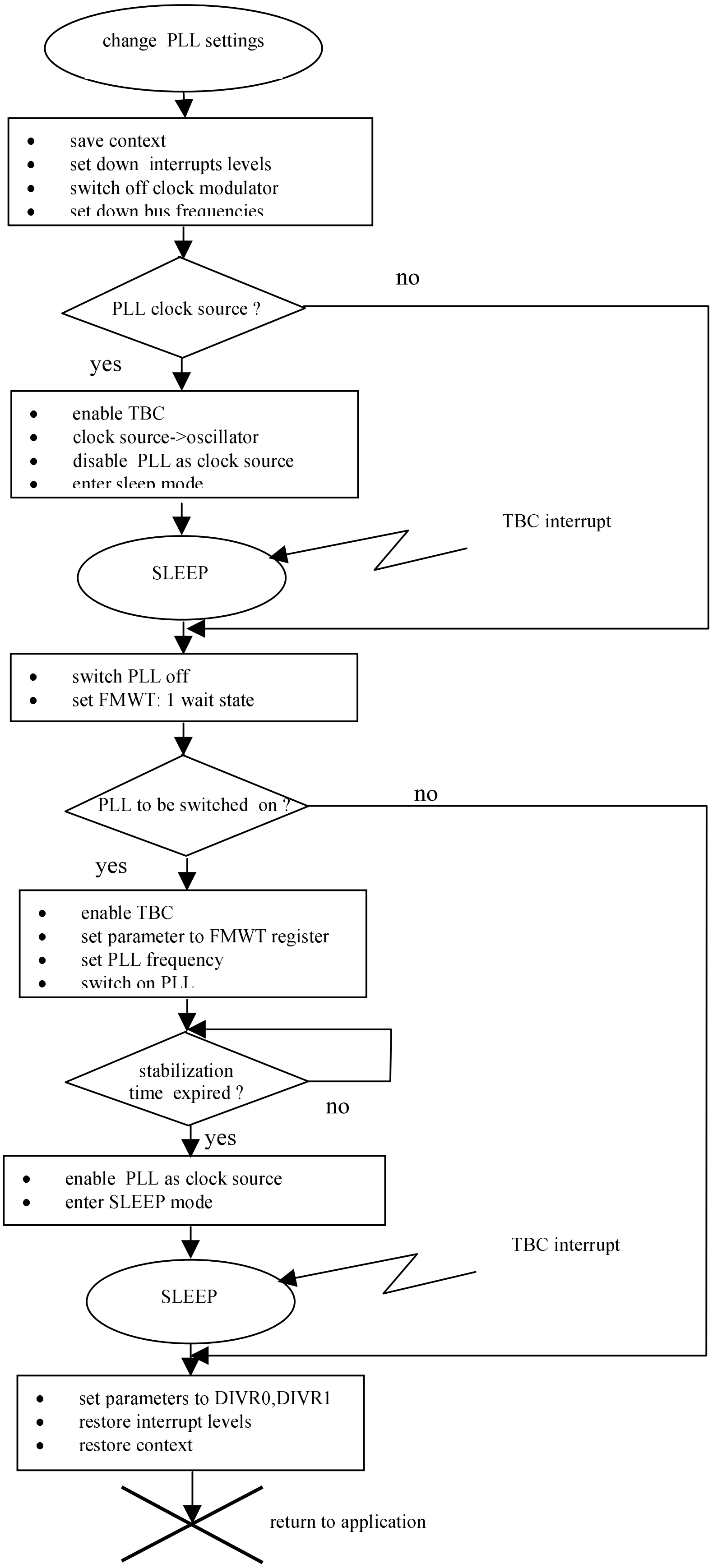


Figure 1 flowchart

5. Interface of the routine

prototype of the function:
void switch_pll (BYTE register_CLKR, BYTE register_DIVR0,BYTE register_DIVR1, BYTE register_FMWT);

The function “switch_pll” is used to reduce the voltage variation when clock settings are changed. Four parameter are passed to this function. The value of these parameters are set to the MCU registers CLKR, DIVR0, DIVR1 and FMWT.

parameter “register_CLKR”:
This parameter mirrors the MCU register CLKR and defines the PLL multiplier will be set to this register.

register CLKR:

PLL2S0	PLL1S2	PLL1S1	PLL1S0	PLL2EN	PLL1EN	CLKS1	CLKS0

This register is used to configure the PLL. The bits PLL1S2, PLL1S1 and PLL1S0 define the frequency of the PLL and only these bits of the parameter „register_CLKR“ needs to be set by the application. If these three bits of the parameter are set to null the PLL is deselected as clock source (CLKS1 = 0, CLKS0 = 0) and switched off (PLL1EN = 0). All other values of this parameter are taken over to CLKR register without testing. The application side needs to take care of the validility of the settings.

parameter “register_DIVR0”:
This parameter mirrors the MCU register DIVR0 and defines the value will be set to this register.

register DIVR0:

B3	B2	B1	B0	P3	P2	P11	P0

Bits B0 –B3 set the division ratio for CPU clock. These bits are always set to null.
Bits P0-P3 set the division the division ratio for clock of resource bus. Only these bits of parameter register_DIVR0 need to be set by the application.

parameter “register_DIVR1”:

register DIVR1:

T3	T2	T1	T0	S3	S2	S1	S0

Bits T0 –T3 set the division ratio for clock of external bus. Only these bits of parameter register_DIVR1 need to be set by application.
Bits S0-S3 are unused.
The application side needs to take care of the valid setting of all three parameter to avoid unspecified settings of the concerned register which can block the MCU.

parameter “register_FMWT”:
This parameter mirrors the MCU register FMWT and defines the value will be set to this register.

register FMWT:

--------	---------	FAC1	FAC0	EQINH	WTC2	WTC1	WTC0

This register is contained only in devices with flash on the f-bus.
The bits WTC0-WTC2 define the count of wait states for flash accesses.
The application side needs to take care of the valid setting.

6. Results

Measurement of the voltage variation when clock settings are changed

The voltage variation could not be avoided totally by this procedure.
But the intensity of the variation could be reduced obviously and could be kept inside the speciefied range by using this procedure.

The following pictures show the results of the measurement at the VCC3C pin during clock settings are changed. There are more then one variation visible, because the voltage varies at disabling the PLL as clock source, enabling the PLL as clock source and setting up/down the bus frequencies.

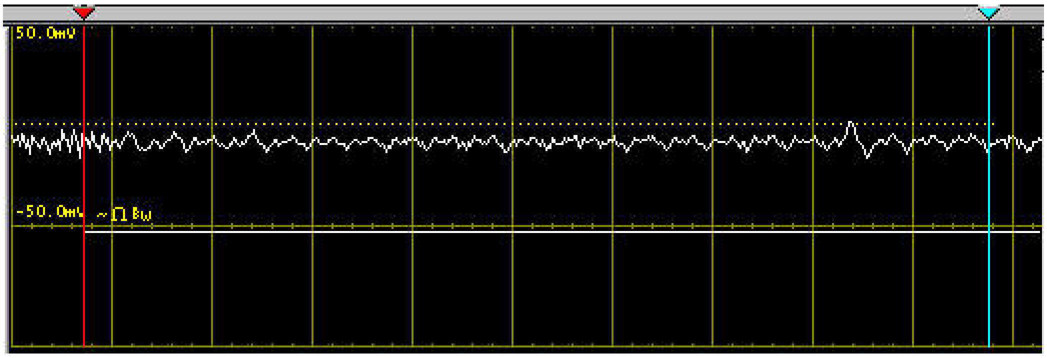


Figure 2 PLL -> switch off , CPU clocked by Oszillator

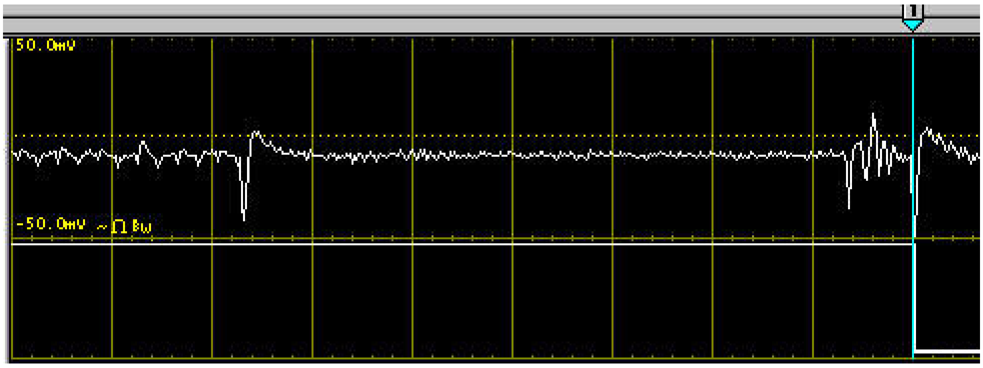


Figure 3 PLL -> switch on to 16 MHz

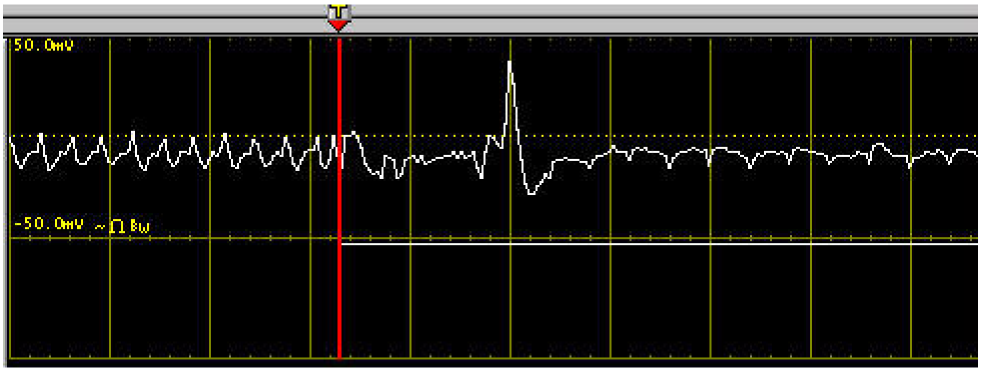


Figure 4 PLL -> switch off , CPU clocked by PLL 16 MHz

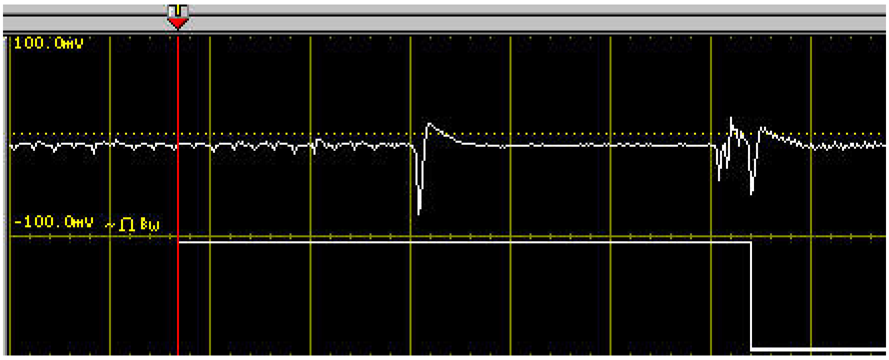
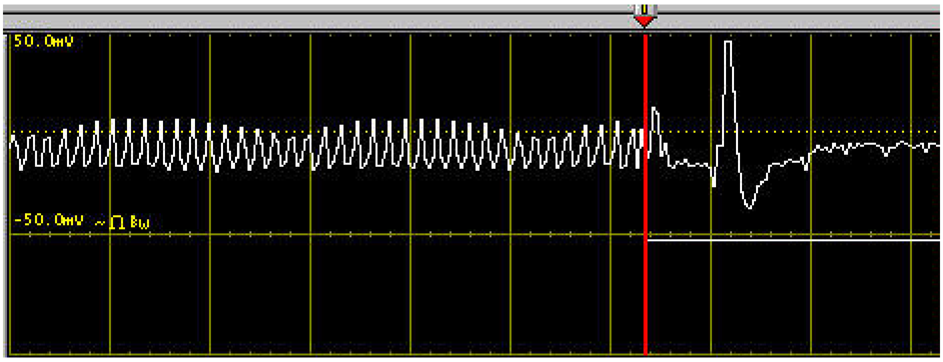


Figure 5 PLL -> switch to 32 MHz)



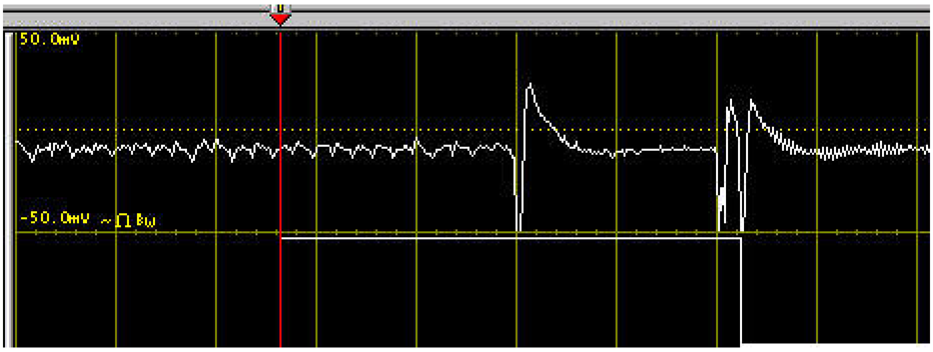


Figure 7 PLL -> switch on to 48 MHz

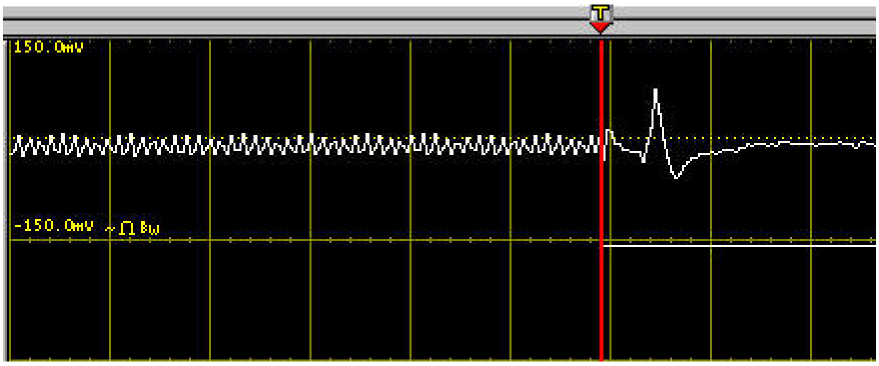


Figure 8 PLL -> switch off , CPU clocked by PLL 48 MHz

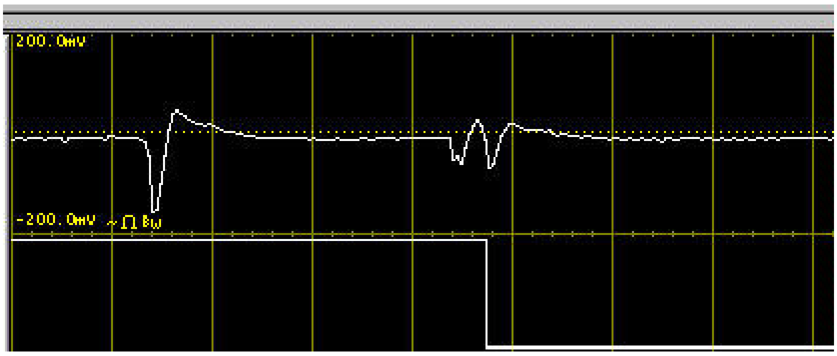


Figure 9 PLL -> switch on to 64 MHz

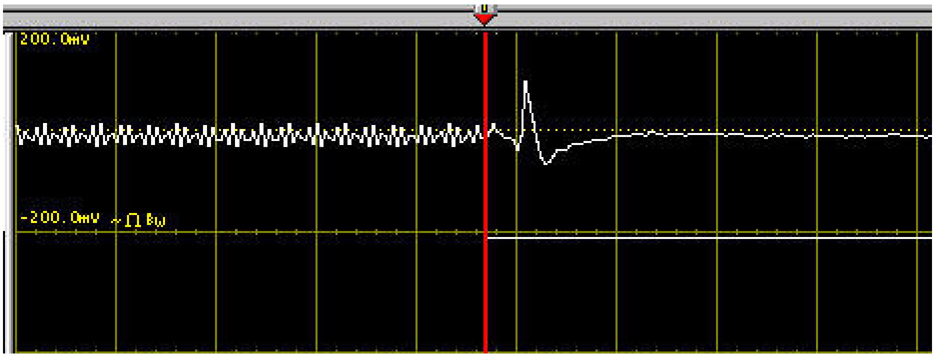


Figure 10 PLL -> switch off ,CPU clocked by PLL 64 MHz

Runtime of this routine

Table 1 shows the runtime of this procedure depending on the clock rate.

CPU clock before call	CPU clock after call	duration
2 MHz, clocked by oscillator	16 MHz, clocked by PLL	2.6 ms
16 MHz, clocked by PLL	32 MHz, clocked by PLL	2.22 ms
64 MHz, clocked by PLL	2 MHz, clocked by oscillator	1.58 ms

Table 1 runtime of the procedure

7. The source code of the routine

```
void switch_pll(BYTE register_CLKR, BYTE register_DIVR0,BYTE register_DIVR1, BYTE register_FWMT)
{
BYTE local_register_TBCR;
BYTE local_register_ICR[48], *ptr,count;

/*        save status register           */

#pragma asm
        st ps,@-r15
#pragma endasm

/*disable all interrupts*/
__DI();

/*save TBCR */
local_register_TBCR = TBCR;

/* save all ICR's and set lowest priority */
ptr=(BYTE*) 0x440;
for (count=0;count<=47;count++)
{
local_register_ICR[count]= *ptr;
*ptr++ = 0x1F;
}

/* disable clock modulator */
if(CMCR & 0x0001)
{
CMCR &= 0xBF;
CMCR &= 0xAC;
}

/* disable TBC interrupt */
TBCR &= 0x3F;

#pragma asm
PLL_SWITCH:

/*   set R-Bus to PLL /16 MHz : */
```

```
          ldi #_divr0, R12     ; R-Bus clock :
          ldi #0x0f, r1        ; R-Bus = PLL /16
          stb r1,@r12

/*   set ext Bus to PLL /16 MHz : */
          ldi #_divr1, R12     ; R-Bus clock :
          ldi #0xFF, r1        ; R-Bus = PLL /16
          stb r1,@r12

/*        set interrupt level of tbc */
          ldi:8 #20,r0
          ldi #_icr31,r1
          stb r0,@r1

/* enable interrupt */
   stilm #30

/* PLL clock source ? */
          ldi:20 #_clkr,R12
          ldub @R12,R1
          ldi:8 #0x02,R0
          and R0,R1
          beq PLL_NOT_CLOCK_SOURCE

/*        configuration and start of TBC */
          ldi:8     #0xa5,r0
          ldi:8     #0x5a,r1
          ldi:32    #_ctbr,r12
          stb       r0,@r12
          stb       r1,@r12
          ldi:8     #0x40,r0          /* TBIF=0,TBIE=1,TBC=000,SYNCR/SYNCS=0*/
          ldi:32    #_tbcr,r12
          stb       r0,@r12

/*  clock source -> oscillator */
          ldi #_clkr, R12  ; PLL lock time elapsed :
   ldi #0xFC,r2

/*        go to SLEEP mode */
          ldi:8     #0x50,r0
          ldi:32    #_stcr,r11
          stb       r0,@r11
          andb r2,@r12     ; deselect PLL as clock source

/* now sleeping...*/
lock_time:
   ldi #_tbcr,R11
   btsth #0x8, @R11                ; Check interrupt flag
   beq lock_time                   ; time elapsed when set

PLL_NOT_CLOCK_SOURCE:
/* switch off PLL */
          ldi #0xF8,r2
          andb r2,@r12


/* disable tbc interrupt */
          ldi:32    #_tbcr,r12
          ldi:8 #0x3F,r1
          andb      r1,@r12
```

```
/* set FWMT, CLK source oscillator, 1 wait states */

#if defined __CPU_MB91FV360G__
            ldi #_fmwt,R3
            ldi #0x01,R4
            stb R4,@R3
#endif

PLL_SWITCHED_OFF:
#pragma endasm
/*   prepare configuration main PLL : */
/* mask the multiplier for CLKR-register; if !0, then PLL switch PLL on*/
if (register_CLKR &= 0x70)
{
            CLKR &= 0x8F;
            CLKR |= register_CLKR;

#if defined __CPU_MB91FV360G__
/* set parameter value -> FWMT */
FMWT= register_FWMT;
#endif

#pragma asm
/* configuration and start of TBC; no interrupt; timer underflow is polled */
            ldi:8       #0xa5,r0
            ldi:8       #0x5a,r1
            ldi:32      #_ctbr,r12
            stb         r0,@r12
            stb         r1,@r12
            ldi:8       #0x00,r0            /* TBIF=0,TBIE=0,TBC=000,SYNCR/SYNCS=0*/
            ldi:32      #_tbcr,r12
            stb         r0,@r12

/* switch on PLL*/
            ldi #_clkr,R12
            ldi #0x04,R1
            orb R1,@R12
/*awaiting stabilisation time*/
lock_time_2nd:
            ldi #_tbcr,R12
            btsth #0x8, @R12                        ; Check interrupt flag
   beq lock_time_2nd                    ; time elapsed when set

;PLL lock time elapsed :
/* configuration and start of TBC */
            ldi:8       #0xa5,r0
            ldi:8       #0x5a,r1
            ldi:32      #_ctbr,r12
            stb         r0,@r12
            stb         r1,@r12
            ldi:8       #0x40,r0            /* TBIF=0,TBIE=1,TBC=000,SYNCR/SYNCS=0*/
            ldi:32      #_tbcr,r12
            stb         r0,@r12

            ldi #_clkr, R12
            ldi #0x02,r2

/* go to SLEEP mode*/
            ldi:8       #0x50,r0            /* SLEEP=1,HIZ=0,OS=00,OSCD2=0,OSCD1=0*/
            ldi:32     #_stcr,r11
            stb        r0,@r11
            orb r2,@r12     ; select PLL as clock source

/* now sleeping...*/

lock_time_3rd:
            ldi #_tbcr,R12
            btsth #0x8, @R12                ; Check interrupt flag
            beq lock_time_3rd               ; time elapsed when set

/* TBT off */
            ldi:8      #0x00,r0
            ldi:32     #_tbcr,r12
            stb        r0,@r12
#pragma endasm
}
/* end of PLL switching on*/

/*   set R-Bus frequency :  */
            DIVR0 = register_DIVR0;

/*   set ext.Bus frequency : */
            DIVR1 = register_DIVR1;

/* end SMOOTH_PLL */

/* restore ICR's*/
ptr=(BYTE*) 0x440;
for (count=0;count<=47;count++)
{
 *ptr++ = local_register_ICR[count];
}

TBCR = local_register_TBCR;

#pragma asm
            ld @r15+,ps
#pragma endasm

return;
}
```