AN12905
Low-power Mode Usage on KM35 MCU

Rev. 0 — June 2020

Application Note

Contents

1 Introduction

Multiple power modes are implemented in the KM35Z75 MCU to meet various kinds of power requirements. This application note describes the details of each power mode and shows the method to enter the low power modes. The document mentions several modules, including:

- Power Management Controller (PMC)
- System Mode Controller (SMC)
- Low Leakage Wakeup Unit (LLWU)

2 Power modes on KM35Z75 MCU

2.1 Basic power modes in Cortex-M0+ core

The Arm® Cortex®-M0+ uses the basic power modes of Arm Cortex-M architecture: Run, Sleep, and Deep Sleep. The Cortex-M0+ processor sleep modes reduce the power consumption:

- A sleep mode, that stops the processor clock.
- A deep sleep mode, that stops the system clock and switches off the PLL and flash memory.

The system can generate spurious wakeup events, for example, a debug operation wakes up the processor. For this reason, software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back into sleep mode. To enter the low power modes (Sleep/Deep Sleep), there are three instructions to inform the processor:

- The Wait For Interrupt (WFI) instruction causes immediate entry to Sleep mode. When the processor executes a WFI instruction, it stops executing instructions and enters Sleep mode.
- The Wait For Event (WFE) instruction causes entry to Sleep mode conditional on the value of a one-bit event register (set by SEV instruction). When the processor executes a WFE instruction, it checks the value of the event register:
    - 0 = The processor stops executing instructions and enters Sleep mode.
    - 1 = The processor sets the register to zero and continues executing instructions without entering Sleep mode.
- The Send the EVent (SEV) instruction causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register.

In the Cortex-M0+ core, the SCB register controls the behavior of entering low power modes after the WFI/WFE instruction.

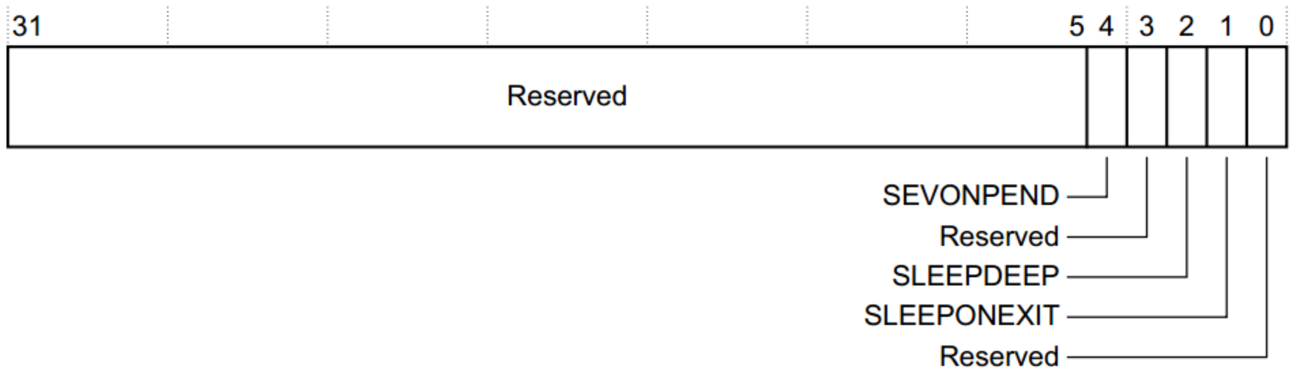

Figure 1. SCB register in Arm core

- The `SCB[SLEEPDEEP]` bit controls whether the processor uses Sleep or Deep Sleep as its low power mode:
  - — 0 = Sleep
  - — 1 = Deep Sleep
- The `SCB[SLEEPONEXIT]` bit indicates sleep-on-exit when returning from the Handler mode (Interrupt Service Routine) to the Thread mode (`main()` function):
  - — 0 = Do not sleep when returning to Thread mode, back to `main()` function directly.
  - — 1 = Enter Sleep or Deep Sleep again, on return from an Interrupt Service Routine (ISR) to Thread mode, never back to the `main()` function any more. Setting this bit to **1** enables an interrupt driven application to avoid returning to an empty main application.
- The `SCB[SEVONPEND]` bit sends Event on Pending bit:
  - — 0 = Only the enabled interrupts or events can wake up the processor, and the disabled interrupts are excluded.
  - — 1 = The enabled events and all interrupts, including the disabled interrupts, can wake up the processor. When an event or interrupt becomes pending, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event.

WFI is generally used to enter the low power modes to achieve the lower idle power consumption. For additional information about **WFE** and **SEV**, see *Armv6-M Architecture Reference Manual*.

2.2 Extending power modes in KM35Z75 MCU

In the KM35Z75 MCU, this core uses WFI or WFE instruction to invoke Sleep and Deep Sleep modes, but also extends power modes.

NOTE

PMC contains a RUN and a STOP mode regulator. RUN regulator is used in normal RUN, WAIT and STOP modes. STOP mode regulator is used during all very low power modes (VLPR, VLPW, VLPS) and low leakage modes (VLLS3/2/1/0). When the STOP mode regulator is used, the frequencies in the system are limited in the very low level.

Table 1 lists all available power modes. Arm CPU documentation uses Sleep and Deep Sleep, while the Kinetis MCU documentation normally uses WAIT and STOP.

Table 1. Power modes on KM35Z75 MCU

Power mode	Description
RUN	The MCU runs at full speed and the internal supply is fully regulated by RUN regulator.
WAIT (Sleep)	The core clock is gated off. The system clock continues to operate. Bus clocks continue to operate if enabled.
STOP (Deep Sleep)	The core clock is gated off. System clocks to other masters and bus clocks would be gated off after all the stop acknowledge signals from supporting peripherals are valid.
VLPR	The core, system, bus, and flash clock maximum frequencies are restricted, the the STOP mode regulator is used.
VLPW (Sleep)	The core clock is gated off. The system, bus, and flash clocks continue to operate, but their maximum frequency is restricted.
VLPS (Deep Sleep)	The core clock is gated off. System clocks to other masters and bus clocks would be gated off after all the stop acknowledge signals from supporting peripherals are valid.
VLLS3 (Deep Sleep)	Act almost the same as VLPS. The MCU is placed in a low leakage mode as the internal logic is powered down. Internal logic states are not retained. All system RAM contents are retained and I/O states are held.
VLLS2 (Deep Sleep)	Almost same as VLLS3, but with lower power consumption.
VLLS1(Deep Sleep)	Almost same as VLLS2, but additionally, the system RAM is powered down as well additionally, for lower consumption.
VLLS0 (Deep Sleep)	Almost same as VLLS1, but additionally, the 1 kHz LPO clock is disabled and the power on reset (POR) circuit can be optionally enabled using `STOPCTRL[PORPO]`, for lower consumption.

The power mode can switch to each other, as shown in Figure 2.

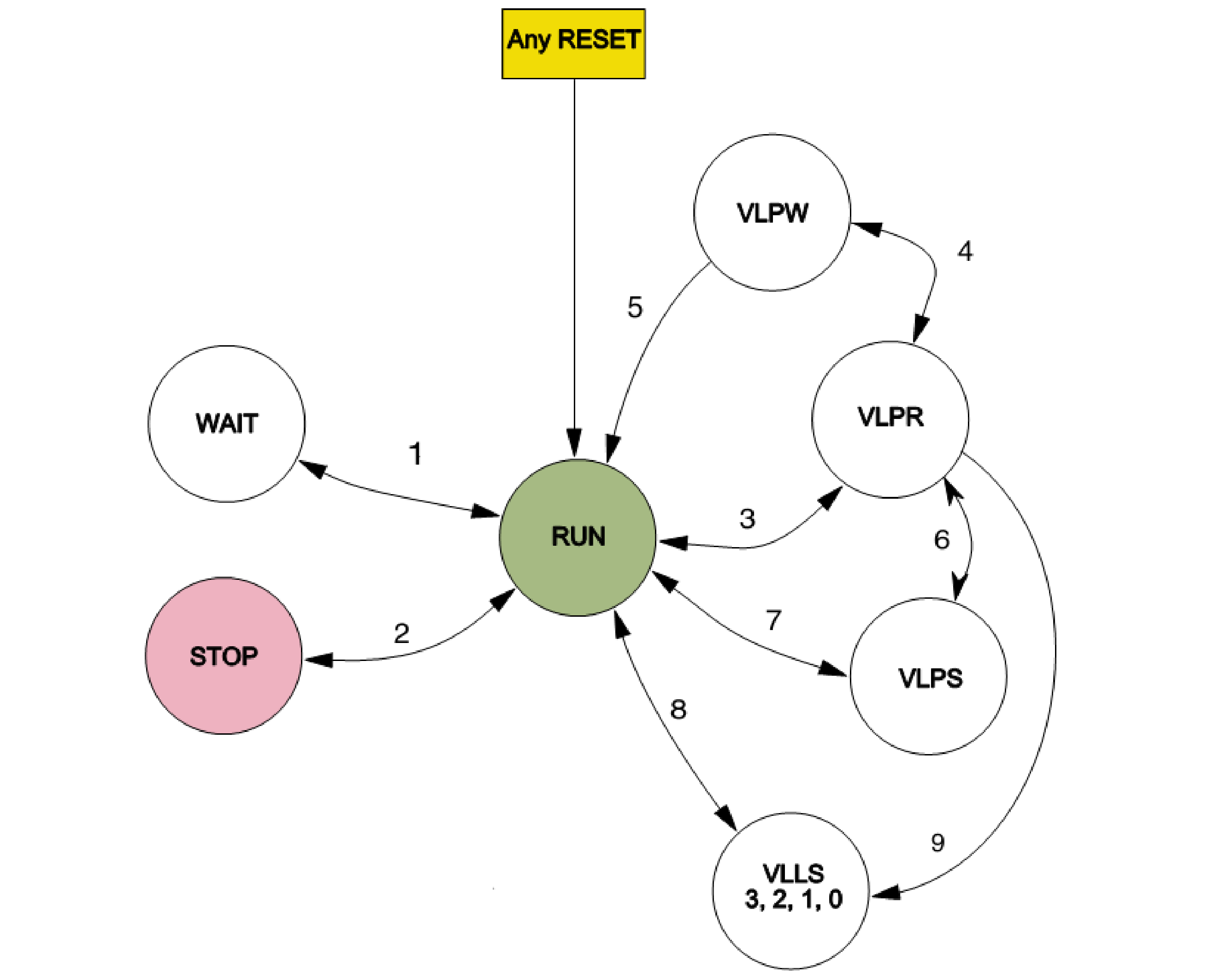

Figure 2. Power mode switch diagram

Generally, most of the modes can switch to others directly, but for wakeup from the VLLS3/2/1/0 modes, it needs the only LLWU input source (from AWIC) or RESET pin, while the normal interrupt (from NVIC) is unavailable during these modes.

To enter the extend power modes, set the target mode to SMC before it is invoked into the Arm Sleep/Deep Sleep modes. For more detailed description about hardware, see **Chapter 15 System Mode Controller (SMC)** in *Kinetis KM35 Sub-Family Reference Manual* (document KM35P144M75SF0RM). A simplest demo will be created to show the operations of entering the low power mode, as shown in The simplest demo of entering low power mode.

3 The simplest demo of entering low power mode

Although the MCUXpresso SDK provides the **power*mode*switch** demo to show the switching from one mode to another, in this application note, for an easy understanding, we create the simplest project to tell the details step by step.

The simplest demo is created to tell the steps of entering the low power mode and waken up. VLLS3 mode is selected as a typical target low power mode, while other VLLS modes work almost the same with it. It is easy for users to use other low-power modes in the same way.

3.1 Steps to enter low power mode

The whole `main()` function in the demo project is implemented as the following code:

```
int main(void)
{
    /* disable the low voltage detection/warning. */
    disable_lvd_lvm();

    /* clear the io pin lock. */
    if (    (RCM->SRS0 == 0x41)  /* VLLS mode wake up due to RESET pin assertion. */
        || (RCM->SRS0 == 0x01)) /* VLLS mode wake up due to other wake up sources. */
    {
        PMC->REGSC |= PMC_REGSC_ACKISO_MASK;
        NVIC_ClearPendingIRQ(LLWU_IRQn);
    }

    /* pending before enter the low power modes. */
    wait_user_button();

    /* disable the modules. */
    disable_pins();
    disable_clock_gates();

    /* setup wake up source. */
    setup_wakeup_button();

    /* enter the low power vlls3 mode. */
    enter_vlls3();

    /* the code should never run to here. */
    while (1);
}
```

1. The Low Voltage Detection (LVD) and Low Voltage Warning (LVW) features are disabled when entering the very low power mode like in VLLS modes. They are enabled by default after booting up, and would cause interrupts to ask CPU to protect the running condition before reset. However, we are not willing to wake up the MCU by these interrupts during the low power modes which would survive in low voltage level. So here we need to disable them using the following operations.

   ```
   void disable_lvd_lvm(void)
   {
       PMC->LVDSC1 = 0x00; /* lvd. */
       PMC->LVDSC2 = 0x00; /* lvw. */
   }
   ```

   a. Reading the PMC*REGSC*ACKISO field indicates whether certain peripherals and the I/O pads are in a latched state as a result of having been in a VLLS mode. Writing 1 to this field when it is set releases the I/O pads and certain peripherals to their normal run mode state.

   b. After recovering from a VLLS mode, users should restore chip configuration before clearing PMC*REGSC*ACKISO. In particular, pin configuration for enabled LLWU wakeup pins should be restored to avoid any LLWU flag from being falsely set when ACKISO is cleared.

   ```
       /* clear the io pin lock. */
       if (    (RCM->SRS0 == 0x41)  /* VLLS mode wake up due to RESET pin assertion. */
           || (RCM->SRS0 == 0x01)) /* VLLS mode wake up due to other wake up sources. */
       {
           PMC->REGSC |= PMC_REGSC_ACKISO_MASK;
   ```

```
        NVIC_ClearPendingIRQ(LLWU_IRQn);
    }
```

c. After waking up from VLLS, the device returns to normal RUN mode with a pending LLWU interrupt. If VLLS was entered from VLPR mode, then the MCU will begin VLPR entry shortly after wakeup. LLWU will retain the flags to indicate the source of the last wakeup until user clears them. In the LLWU ISR after the system reset, the user can poll the LLWU module wake-up flags to determine the source of the wake-up.

```
/* LLWU interrupt handler. */
void LLWU_IRQHandler(void)
{
    /* if waken up by external pin. */
    if (LLWU_GetExternalWakeupPinFlag(LLWU, 15u))
    {
        /* clear llwu flag. */
        LLWU_ClearExternalWakeupPinFlag(LLWU, 15u);
    }
    __DSB();
}
```

d. In this demo, to reduce the unnecessary current leakage, the debug port pins would be disabled before entering the low power mode. But the debugger could no longer connect to the MCU then. There would be a short slice between booting up and entering the low power mode, but it can be caught occasionally. For ease of debugging, here use a button to pend the routine before entering the low power mode. Once the MCU is reset, before manually pressing the button, the debugger can connect the MCU as it will. Then we have the following function:

```
/* SW2: PTD1. */
void wait_user_button(void)
{
    CLOCK_EnableClock(kCLOCK_PortD);

    /* gpio pin with pull-up. */
    PORTD->PCR[1] = PORT_PCR_MUX(1) | PORT_PCR_PE_MASK | PORT_PCR_PS(1);

    gpio_pin_config_t gpio_pin_config;
    gpio_pin_config.pinDirection = kGPIO_DigitalInput;
    GPIO_PinInit(GPIOD, 1u, &gpio_pin_config);

    while (1u == GPIO_PinRead(GPIOD, 1u))
    {}
    while (0u == GPIO_PinRead(GPIOD, 1u))
    {}

    PORTD->PCR[1] = PORT_PCR_MUX(0); /* gpio pin disabled. */
    CLOCK_DisableClock(kCLOCK_PortD);
}
```

2. Disable the port pins as much as possible, and disable the clock gate as much as possible to save power. The debug port pins (SWCLK and SWDIO) are enabled by default. Disable them here as well.

```
void disable_pins(void)
{
    CLOCK_EnableClock(kCLOCK_PortE);
    PORTE->PCR[6] = PORT_PCR_MUX(0); /* swdio. */
    PORTE->PCR[7] = PORT_PCR_MUX(0); /* swclk. */
    CLOCK_DisableClock(kCLOCK_PortE);
}

void disable_clock_gates(void)
```

```
{
    SIM->SCGC4 = 0x0;
    SIM->SCGC5 = 0x0;
    SIM->SCGC6 = SIM_SCGC6_FTFA_MASK;
    SIM->SCGC7 = 0x0;
}
```

The last step before entering the low power modes is to set an available wekeup source so that the MCU can return to RUN mode on indicated condition. The interrupt managed by NVIC can be the wake up source for the normal STOP modes. But for the ultra low power mode like VLLSx, the LLWU should be used.

```
/* SW1: PTA4/LLWU_P15. */
void setup_wakeup_button(void)
{
    CLOCK_EnableClock(kCLOCK_PortA);
    PORTA->PCR[4] = PORT_PCR_MUX(1); /* PTA4/LLWU_P15. */
    CLOCK_DisableClock(kCLOCK_PortA);

    /* setup for vlls3/2/1/0. */
    LLWU_SetExternalWakeupPinMode(LLWU, 15u, kLLWU_ExternalPinRisingEdge); /* LLWU_P15 */
    NVIC_EnableIRQ(LLWU_IRQn);
}
```

3. Set the target mode and invoke into it.

```
void enter_vlls3(void)
{
    /* unlock the modes. */
    SMC->PMPROT = SMC_PMPROT_AVLLS_MASK;

    /* prepare to enter the STOP modes. */
    SMC->PMCTRL = SMC_PMCTRL_RUNM(0x0) | SMC_PMCTRL_STOPM(0x4);
    SMC->STOPCTRL = SMC_STOPCTRL_PSTOPO(0x0) /* 2'b00 normal stop; 2'b01 pstop1; 2'b02 pstop2 */
                  | SMC_STOPCTRL_VLLSM(0x3); /* vlls3. */

    /* stop entry. */
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
    asm
    (
        "wfi;"
        "nop;"
    );
}
```

3.2 Hardware rework

To reduce the unnecessary current leakage, disable or disconnect the port pins as much as possible.

- The measure socket is J6, including the VDD, VDDA, and VDDA_AFE for MCU. VBAT is excluded since the RTC is not used in this demo. See Figure 3.

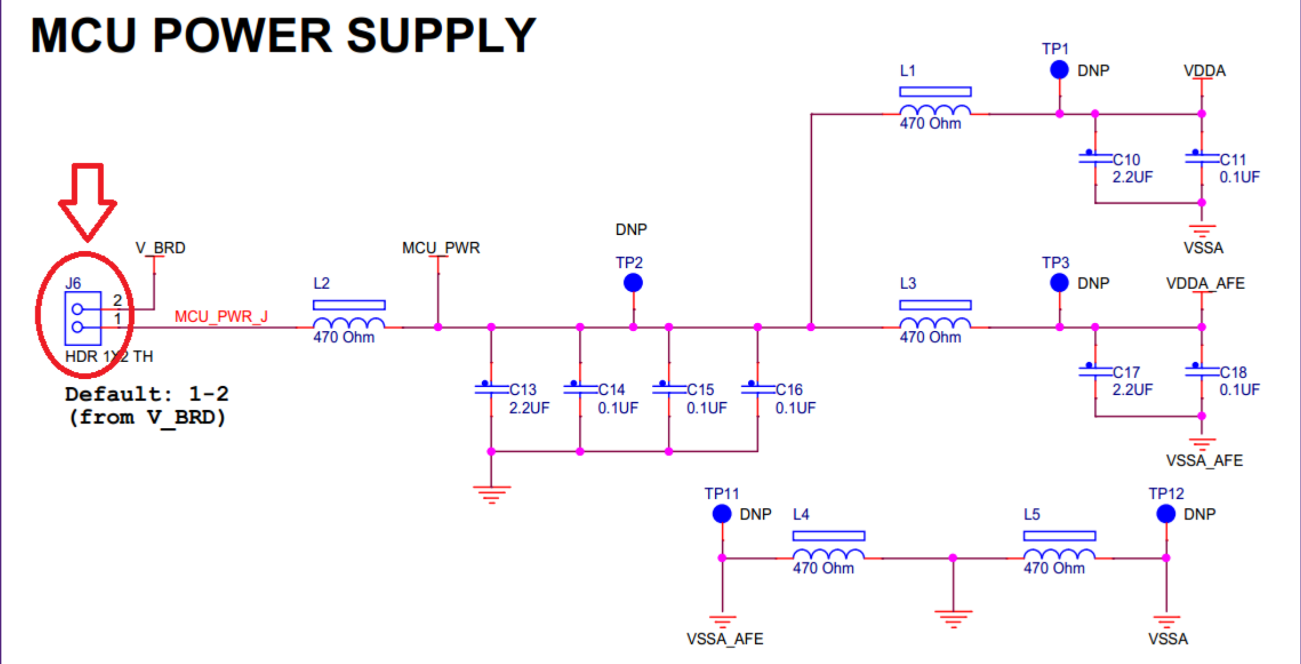

Figure 3. Jumper J6 for measurement

- The RESET pin should never be disabled since it would be used as the final wakeup source. However, the debugger connects the RESET wire on the TWR-KM35Z75M board now. So the jumper J28 is required to be removed from the board. See Figure 4.

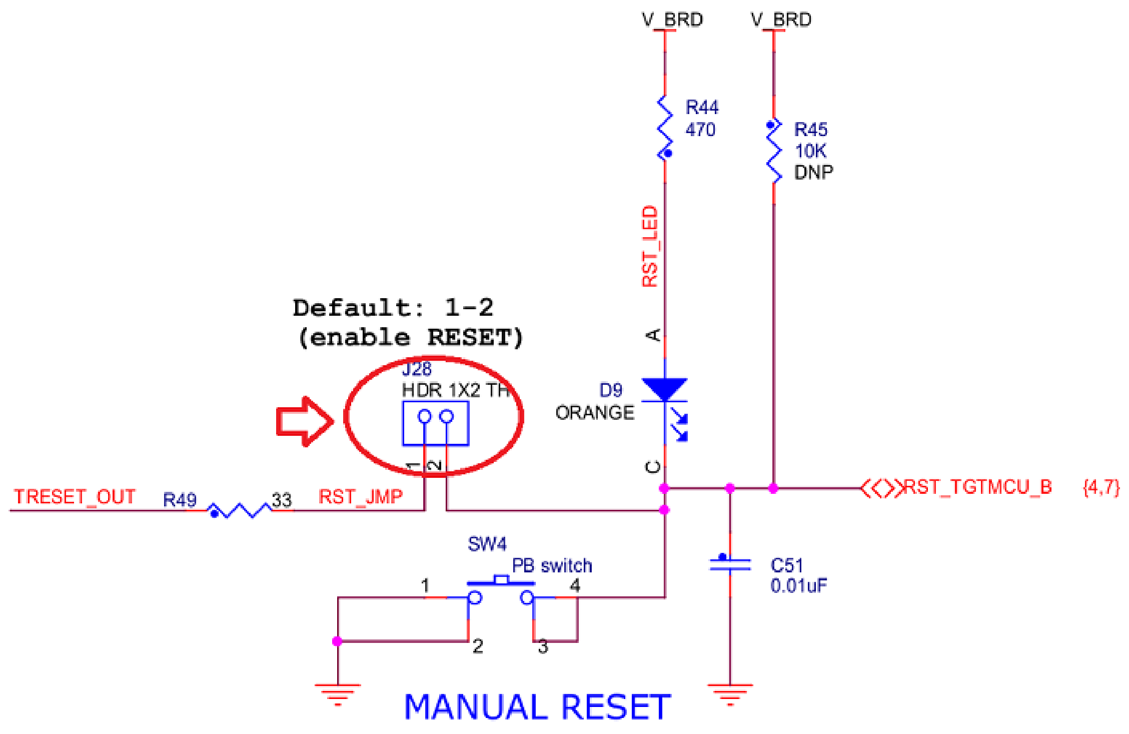

Figure 4. Jumper J28 for RESET from debugger

- Remove the jumper J1 power supply to VBAT if the RTC is not used.

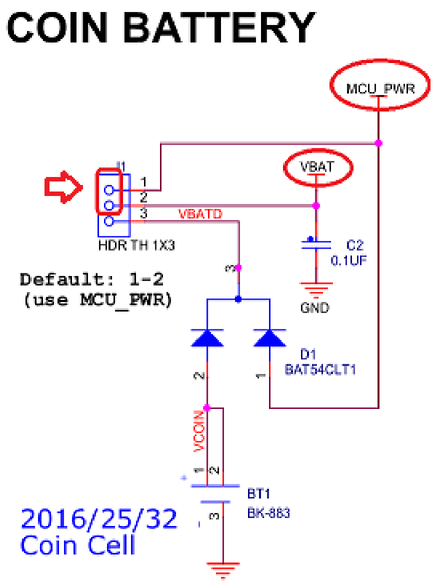

Figure 5. Jumper J1 for VBAT

- Remove all the other jumpers on the board since no external devices, like SPI, I^{2}C, ADC, and Crystal, are used in this demo project. Only keep the jumper J3 to make sure the power supply is available to the board.

3.3 Running the demo and measuring the current

Build the project and download the image to TWR-KM35Z75M board before removing the J28. Inject the amperemeter into the J6.

- Press the **RESET** button. Read the current value on 4 MHz core clock at RUN mode. The value would be about 1.8 mA.
- Press the **SW2** button to enter the low power. Read the current value on VLLS3 mode. The value would be about 2.6 uA.
- Press the **SW1** button to wake up to RUN mode. Read the current values again, like in Step 1.

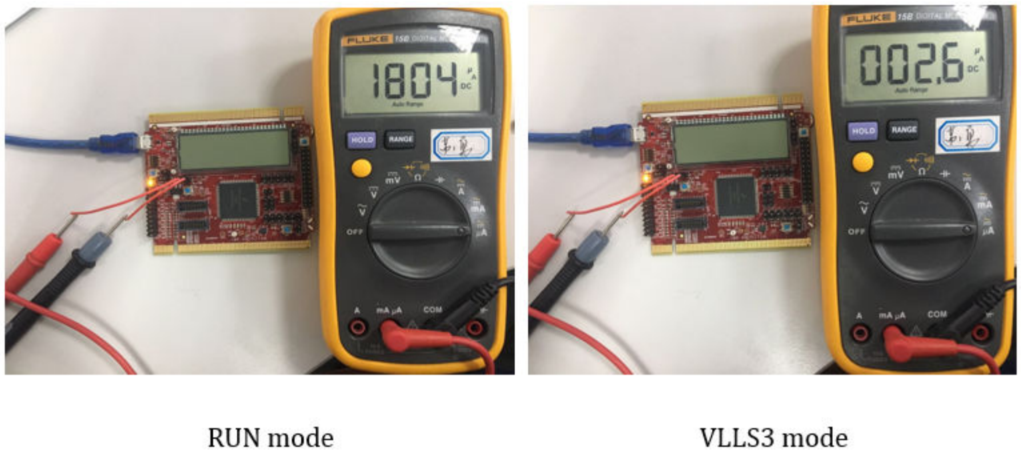

Figure 6. Reading the measuring values when switching mode

For other measurement result, please refer to the records listed in the datasheet document.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: June 2020

Document identifier: AN12905