Attached you can find an example for CW10.4 HID device software that is up and running using the baremetal USB stack enabling suspend mode and VLPS. We use the FRDM-KL25
Once you unzip the file the project is located at the following path …\USB wakeup from VLPS\Source\Device\app\hid\cw10\kinetis_l2k

The current during suspend mode is around 800uA, please note that this is not an optimal current we didn't implement all the low power best practices. We mainly focus on being able to go into VLPS – RUN using SUSPEND-RESUME .

Our major lesson learned of this implementation is that you need to go into the low power mode outside the USB stack, which is why we did it on the main loop, after the USB task ends.

main_kinetis.c

```
while(TRUE)
{
    Watchdog_Reset();
    /* Call the application task */
    TestApp_Task();

    /** If the suspend interrupt && enumeration was complete send to Low power mode */
    if (u8USBSleepFlag)
    {
        u8USBSleepFlag = FALSE;

        /** Turn-off LED */
        GPIOB_PSOR |= (1<<18);
        GPIOB_PSOR |= (1<<19);

        /** Go to VLPS */
        Enter_LowPowerMode(VLPS);

        /* Go back to Run mode with the PLL engaged */
        SIM_CLKDIV1  = SIM_CLKDIV1_OUTDIV4(1);
        SIM_CLKDIV1 |= SIM_CLKDIV1_OUTDIV1(1);
        exit_vlpr();
        pbe_pee(CLK0_FREQ_HZ);

        /** Turn-on LED */
        GPIOB_PCOR |= (1<<18);
    }
```

To being able to separate the Enumeration suspend, we include more logic inside the SLEEP (SUSPEND) interrupt, if the device already passed the enumeration process then you enable the RESUME interrupt, both the synchronous and the asynchronous. Inside this interrupt the software flag u8USBSleepFlag is turned on.

usb_dci_kinetis.c

```
if(SLEEP_FLAG(intr_stat))
{
    if (u8Enumeration)
    {
        /* Clear RESUME Interrupt if Pending */
        USB0_ISTAT = USB_ISTAT_RESUME_MASK;

        /* Clear SLEEP Interrupt */
        USB0_ISTAT = USB_ISTAT_SLEEP_MASK;

        /* Notify Device Layer of SLEEP Event*/
        (void)USB_Device_Call_Service(USB_SERVICE_SLEEP, &event);

        /* Set Low Power RESUME enable */
        USB0_USBTRC0 |= USB_USBTRC0_USBRESMEN_MASK;

        /* Set SUSP Bit in USB_CTRL */
        USB0_USBCTRL |= USB_USBCTRL_SUSP_MASK;

        /* Enable RESUME Interrupt */
        USB0_INTEN |= USB_INTEN_RESUMEEN_MASK;

        /* Enter Stop3 Mode  */
        u8USBSleepFlag = TRUE;
    }
    else
    {
        /* Clear SLEEP Interrupt */
        USB0_ISTAT = USB_ISTAT_SLEEP_MASK;

        /* Notify Device Layer of SLEEP Event */
        (void)USB_Device_Call_Service(USB_SERVICE_SLEEP, &event);
    }
}
```

To test stop you can go to the main loop, change the Enter_LowPowerMode parameter to STOP and comment the software related to go back to Run mode.

```
while(TRUE)
{
    Watchdog_Reset();
    /* Call the application task */
    TestApp_Task();

    /** If the suspend interrupt && enumeration was complete send to Low power mode */
    if (u8USBSleepFlag)
    {
        u8USBSleepFlag = FALSE;

        /** Turn-off LED */
        GPIOB_PSOR |= (1<<18);
        GPIOB_PSOR |= (1<<19);

        /** Go to STOP */
        Enter_LowPowerMode(STOP);

        /* Go back to Run mode with the PLL engaged
        SIM_CLKDIV1  = SIM_CLKDIV1_OUTDIV4(1);
        SIM_CLKDIV1 |= SIM_CLKDIV1_OUTDIV1(1);
        exit_vlpr();
        pbe_pee(CLK0_FREQ_HZ);*/

        /** Turn-on LED */
        GPIOB_PCOR |= (1<<18);
    }
```

- Alejandra