AN1043
APPLICATION NOTE

IMPLEMENTING USB SUSPEND MODE ON THE ST92163

by Microcontroller Division

INTRODUCTION

In the absence of bus traffic from any powered state for more than 3 ms, all devices that are supplied power via the Universal Serial Bus (USB) must enter Suspend mode.

Bus-powered devices draw current from the USB. These devices must reduce their power consumption to meet USB specifications. For power consumption values, refer to the USB Specifications datasheet (version 1.1) (see Section 4).

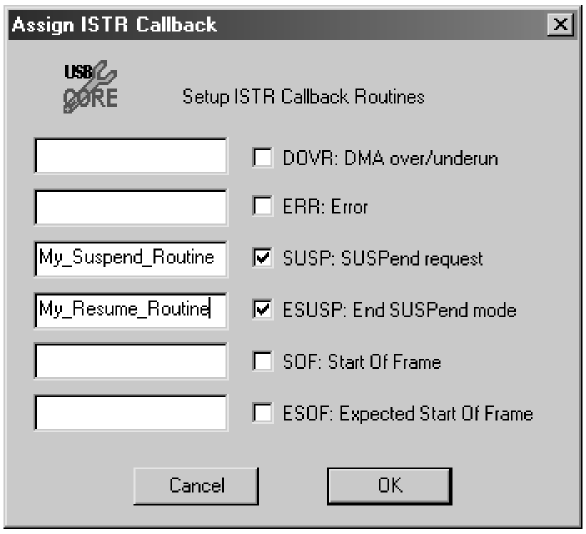
A device in Suspend mode can resume operation when any non-idle signal is received on its upstream port. It can also resume operation from the application via an external interrupt.

This application note describes how to configure Suspend and Resume operations for ST92163 microcontrollers.

1 SUSPEND/RESUME OPERATIONS SETUP

You have to enable the Suspend and End_suspend interrupts to support Suspend and Resume operations. This can be easily done using the `CoreCfg.exe` utility provided by the ST92163 USB library. The following figure shows an example of how to use this utility.

Figure 1. Configuring Suspend and End_suspend Interrupt


Assign ISTR Callback

Setup ISTR Callback Routines

	☐ DOVR: DMA over/underun
	☐ ERR: Error
My_Suspend_Routine	✓ SUSP: SUSPend request
My_Resume_Routine	✓ ESUSP: End SUSPend mode
	☐ SOF: Start Of Frame
	☐ ESOF: Expected Start Of Frame

You have to write the two callback routines associated to Suspend and End Suspend interrupts (`My_Suspend_Routine` and `My_Resume_Routine`) to reduce the power consumption of the application board.

The `CoreCfg.exe` utility generates a `config.h` file. This file contains the interrupt mask for the USBISTR interrupt register that defines the Suspend and End_suspend callback routines. Refer to the “Using the ST92163 USB Library” document provided with the ST92163 USB library for more details on how to use this utility.

Figure 2. Suspend and End_suspend Callback Routines

```
#define SUSP_Callback  My_Suspend_Routine()
#define ESUSP_Callback My_Resume_Routine()


#define IMR_MSK (INT_RESET|INT_ESUSP|INT_SUSP)
```

Note: This code is generated by the `CoreCfg.exe` utility.

To support Resume operations, the WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (WUIMU) has to be configured during the initialization phase of the ST92163 microcontroller.

Figure 3. Wake-Up Unit Configuration

```
/* WAKE-UP Unit configuration*/
spp(WU_PG);
WU_CTLR = 0x1;  /* Set WKUP-INT and reset ID1S bits */
WU_WUMRH = 0x80;  /* Set WUM15 bit (To USB I/F) */
WU_WUMRL = 0x0;
WU_WUTRH = 0x80;  /* Select rising edge for WUT15 */
WU_WUTRL = 0x0;  /* Select rising edge for WUT7 */
WU_WUPRH = 0;     /* Clear pending bits */
WU_WUPRL = 0;
```

Note: This code is included in the ST92163 USB Library `main.c` file.

2 SUSPEND MANAGEMENT

The ST92163 USB Interface features a dedicated interrupt flag for Suspend mode support. When the ST92163 must enter Suspend mode, the SUSP (*Suspend mode request*) bit of the USBISTR (*INTERRUPT STATUS REGISTER*) register is set and an interrupt is generated.

Note: The Suspend condition check is enabled immediately after any USB Reset and is disabled by hardware when Suspend mode is active.

2.1 SUSPEND INTERRUPT ROUTINE

Before entering Suspend mode, the application program must configure the application used to reduce power consumption (switch off on-board components, configure I/O ports, ...).

You have to perform these operations in the `My_Suspend_Routine` routine (see Section 1).

The SUSP interrupt routine will then switch the USB interface into Suspend mode by setting the TIM_SUSP *(Timed Suspend*) bit and then the LP_SUSP (*Low-power suspend*) bit of the USBCTLR (*CONTROL REGISTER*) register.

Finally, the ST92163 CPU clocks are put into Low Power mode (STOP mode) by the SUSP interrupt routine via the WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (WUIMU).

The following flowchart illustrates the Suspend operation.

Figure 4. Suspend Operation Flowchart

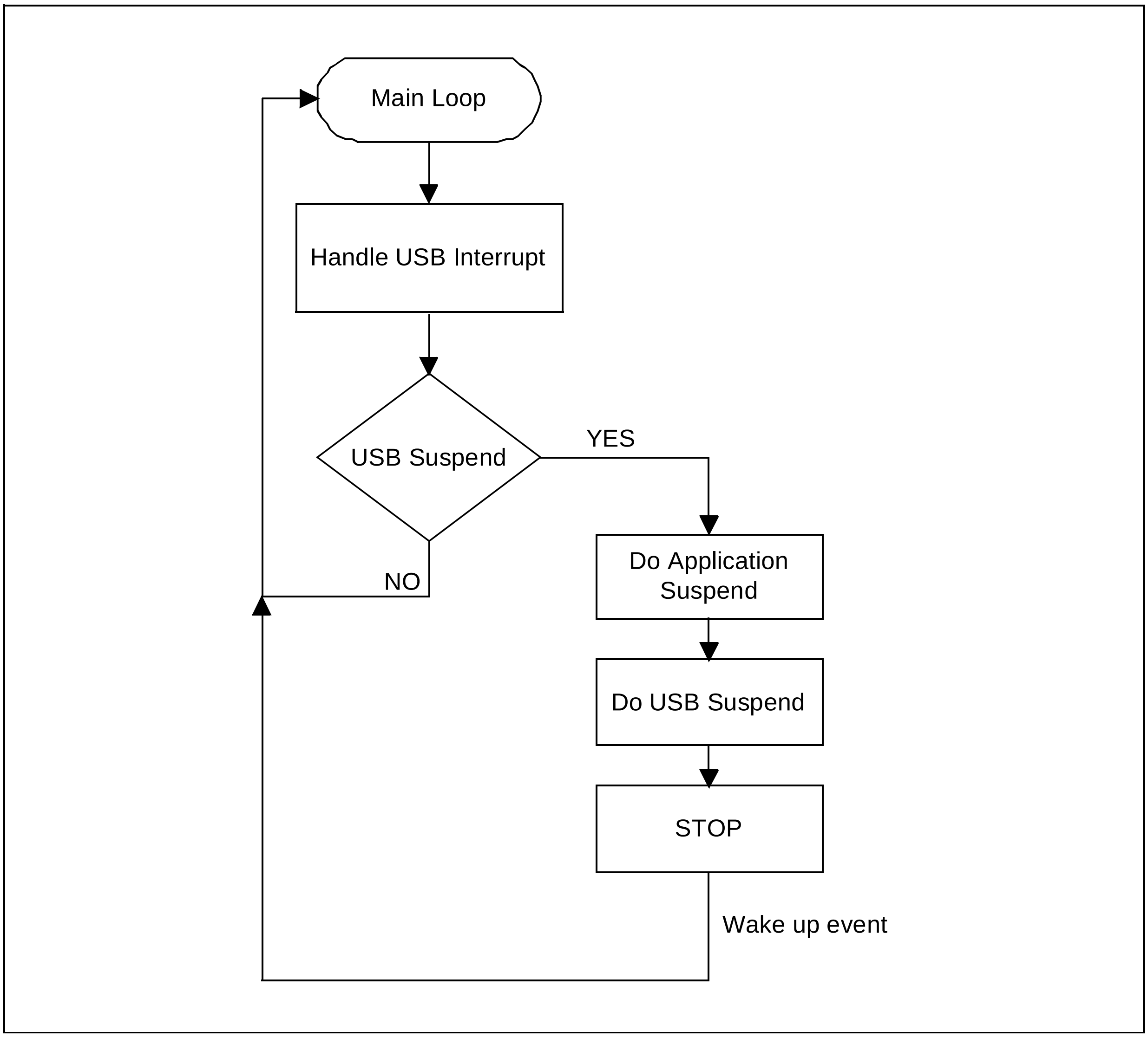


Figure 5. Suspend Interrupt Routine

```
#if(IMR_MSK & INT_SUSP)
if (ISTR & IMR & INT_SUSP) {

#ifdef SUSP_Callback
SUSP_Callback;
#endif
/* Transient suspend */
spp(15);
CTLR |= CTR_TIM_SUSP; /* Set TIM_SUSP bit of CTLR register */

spp(WU_PG);
WU_WUPRH = 0;    /* Clear pending bits */
WU_WUPRL = 0;

/* LP suspend  */
 spp(15);
CTLR |= CTR_LP_SUSP;
/* Enter STOP mode*/
spp(WU_PG);
WU_CTLR = 0x05;/* Set STOP and WKUP-INT bits */
WU_CTLR = 0x01;/* Set WKUP-INT bit */
WU_CTLR = 0x05;/* Set STOP and WKUP-INT bits */
NOP;NOP;NOP;NOP;
 spp(15);
CLEAR_ISTR_FLAG(INT_SUSP);
}
```

Note: This code is included in the ST92163 USB Library `int92163.c` file.

3 RESUME MANAGEMENT

The ST92163 can resume operation by a signal received via its upstream port, and also by the application via an external interrupt.

3.1 RESUME FROM THE USB

When a non-idle signal is received on its upstream port, an End_suspend (Resume) interrupt is generated and the ST92163 will resume operations.

The End_suspend interrupt routine will then check the on-chip PLL locking and resume the ST92163 USB interface.

The following flowchart illustrates the Resume operation.

Figure 6. Resume Operation Flowchart

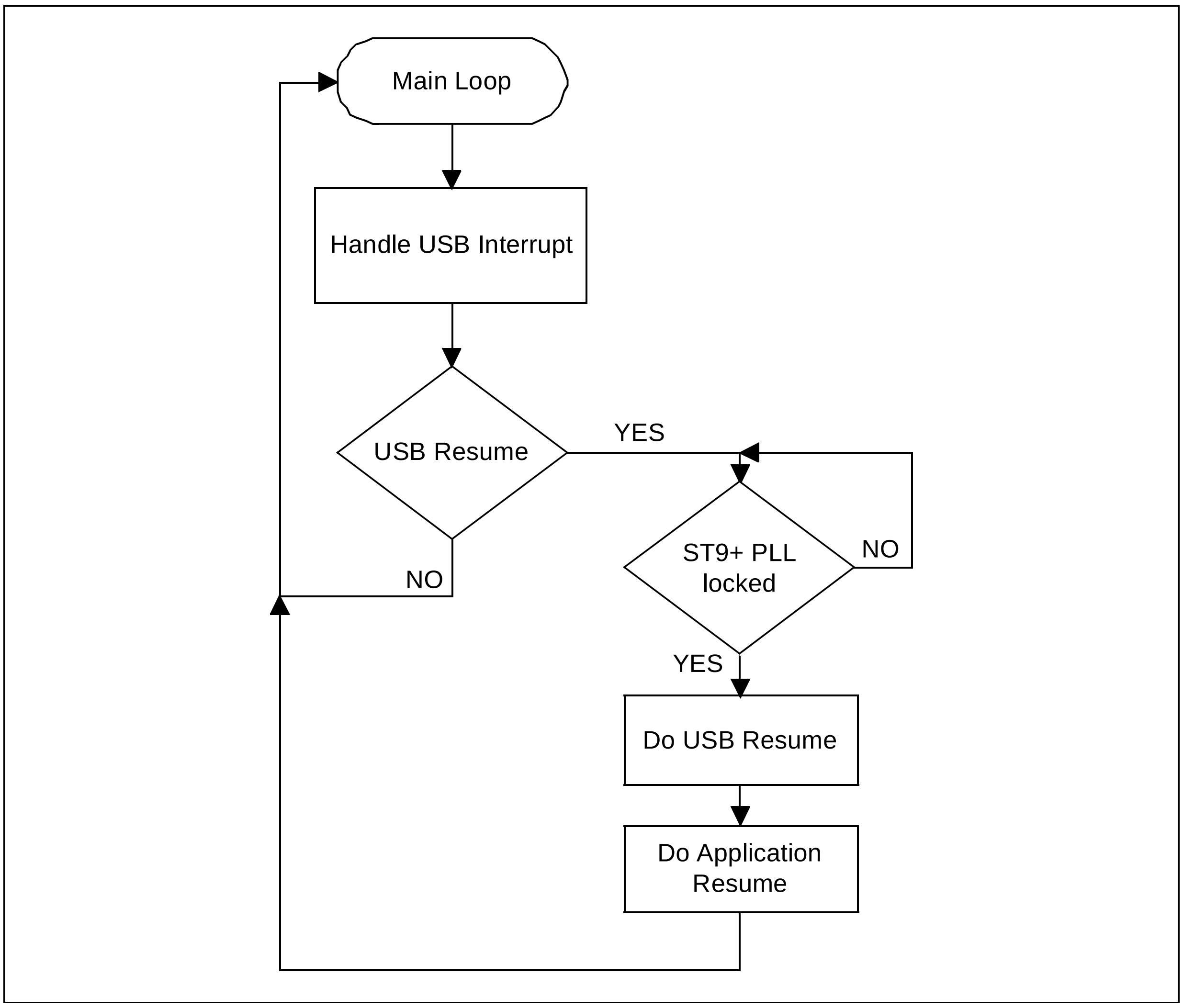


Figure 7. End_suspend Interrupt Routine

```
#if(IMR_MSK & INT_ESUSP)
if (ISTR & IMR & INT_ESUSP) {
spp(RCCU_PG);
while (!(CLK_FLAG & 0x2));/* Test PLL lock-in */
CLK_FLAG |= 0x1;     /* Set CSU_SEL bit */

spp(15);
CTLR &= ~CTR_TIM_SUSP;  /* Clear TIM_SUSP bit of CTLR register */

spp(WU_PG);
if (ValBit(WU_WUPRH, 7))
WU_WUPRH &= ~0x80; /* Clear pending bit */

#ifdef ESUSP_Callback
ESUSP_Callback;
#endif

spp(15);
CLEAR_ISTR_FLAG(INT_ESUSP);
}
#endif
```

Note: This code is included in the ST92163 USB Library `int92163.c` file.

3.2 RESUME FROM APPLICATION BOARD

To resume operations on the ST92163 MCU via an external interrupt by the application, the WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (WUIMU) has to be configured.

The WUIMU wake-up lines are all mapped on the INTD1 external interrupt channel. When a wake-up line is enabled, it can wake up the microcontroller. Refer to the ST92163 Datasheet for more details about the WUIMU.

The following example illustrates the configuration of Port 3.7 as the wake-up event source.

Figure 8. Port 3.7 Configuration

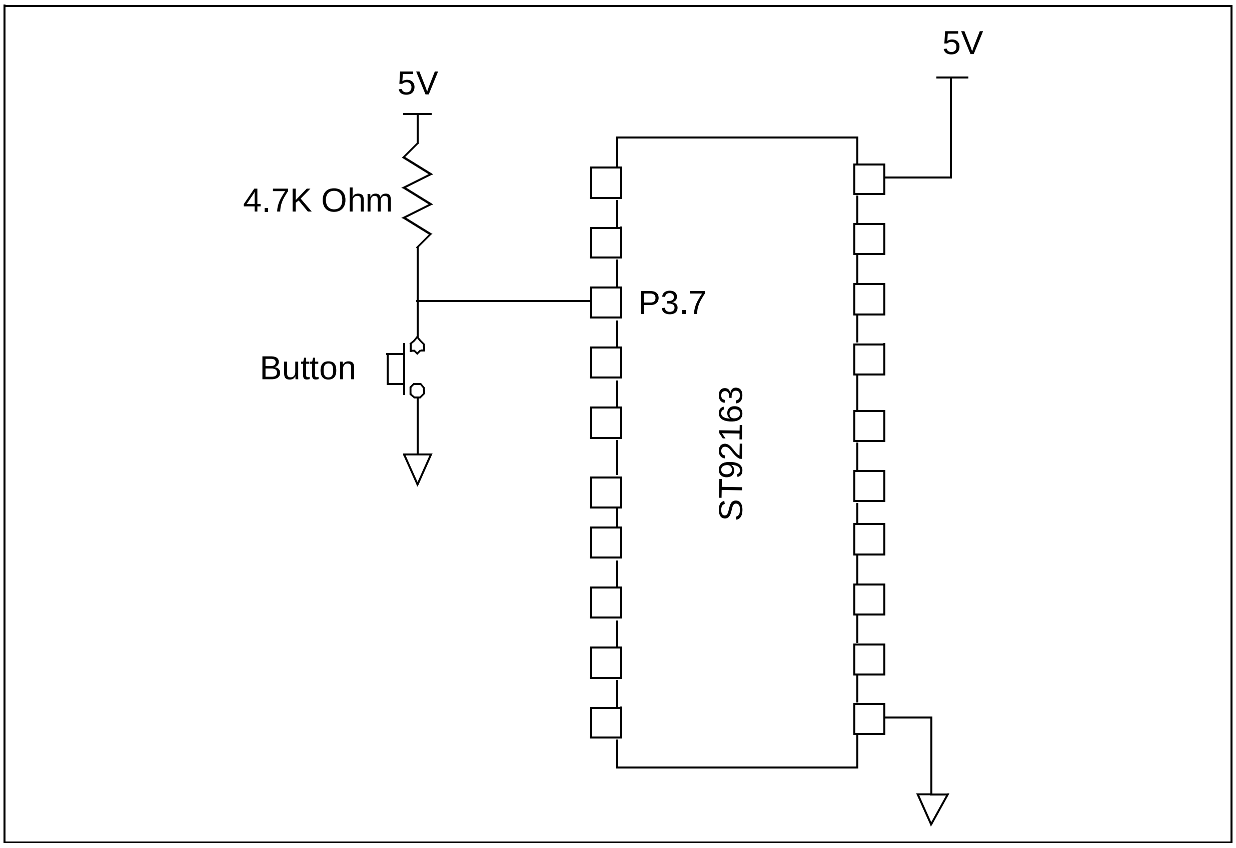


Figure 9. Remote Wake-up Configuration Routine

```
/* Port 3 configuration */
spp(P3C_PG);/*init P3.7 as Input CMOS*/
P3C0R |= 0x80;
P3C1R &= 0x7F;
P3C2R &= 0x7F;
P3DR= 0xFF;
/* WAKE-UP Unit configuration*/
spp(WU_PG);
WU_CTLR = 0x1;  /* Set WKUP-INT and reset ID1S bits */
WU_WUMRH = 0x80;  /* Set WUM15 bit (To USB I/F) */
WU_WUMRL = 0x80;  /* Set WUM7 bit (From SW2 button) */
WU_WUTRH = 0x80;  /* Select rising edge for WUT15 */
WU_WUTRL = 0x80;  /* Select rising edge for WUT7 */
WU_WUPRH = 0;     /* Clear pending bits */
WU_WUPRL = 0;

/* Interrupts configuration */
    spp(EXINT_PG);
    EITR = 0x80;  /* Trigger on rising edge for D1 channel */
    EIPR = 0x0;  /* Clear pending bits */
    EIMR = 0x80;  /* Set external interrupt mask INTD1*/
    EIPLR = 0x7F; /* Set external interrupt priority level for channels 
D1 */
    EIVR = 0x46;  /* Set external interrupt vector to 0x40 */
```

Note: This code is included in the ST92163 USB Library `main.c` file.

When the button on port 3.7 is pressed and the ST92163 is in STOP mode, an external interrupt is generated and the MCU wakes up.

If the remote wake-up capability is enabled on the device, it will have to propagate a resume signal upstream.

The following flowchart illustrates the Resume and Remote Wake-up operations.

Figure 10. Resume and Remote Wake-up Flowchart

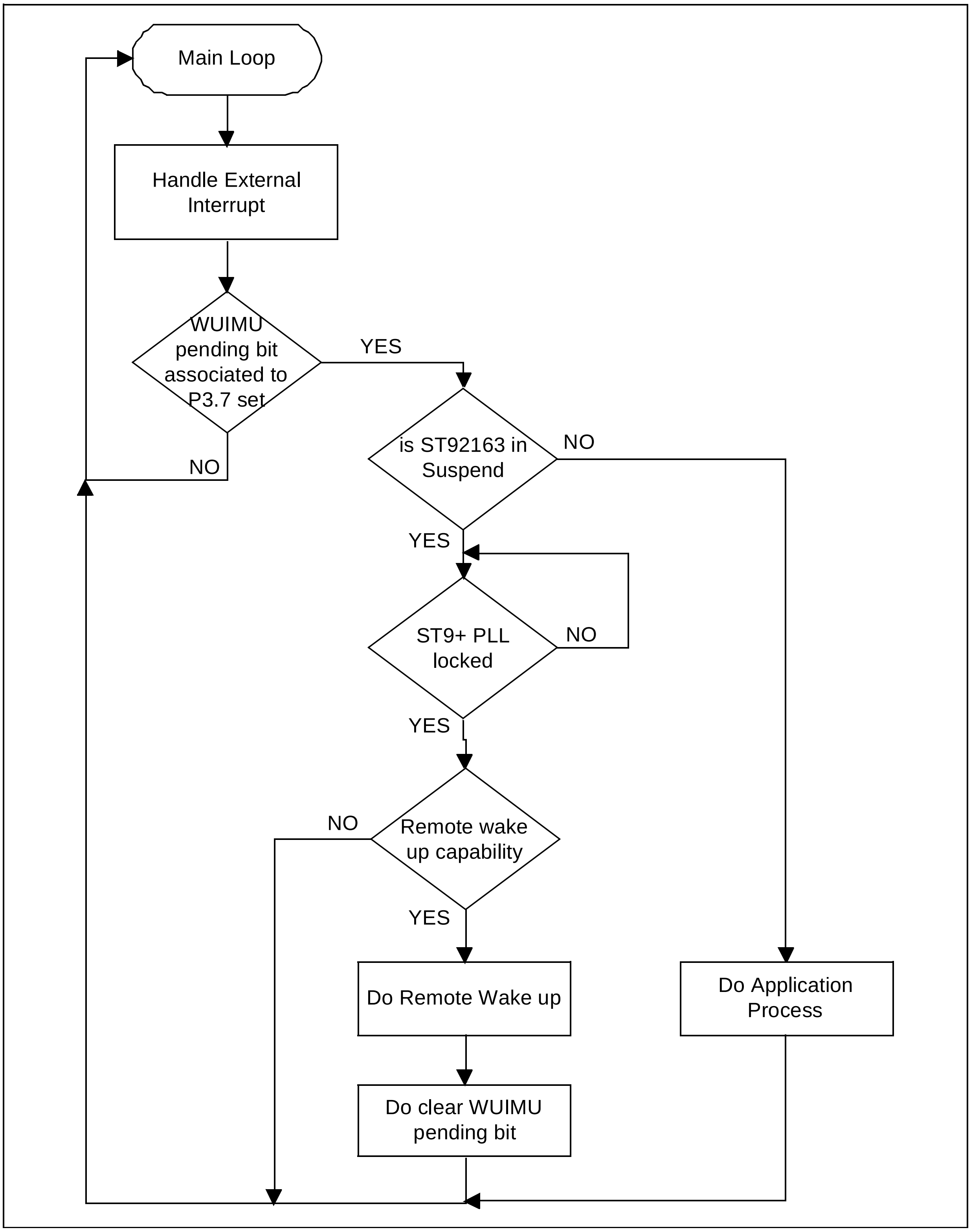

Main Loop
Handle External Interrupt
WUIMU pending bit associated to P3.7 set
YES
NO
is ST92163 in Suspend
NO
YES
ST9+ PLL locked
NO
YES
Remote wake up capability
NO
YES
Do Remote Wake up
Do Application Process
Do clear WUIMU pending bit

Figure 11. Resume and Remote Wake-up Routine

```
#pragma interrupt (ButtonPressed)
void ButtonPressed(void)
{
SAVE_PAGE;
spp(WU_PG);
if (ValBit(WU_WUPRL, 7)) /* If SW2 button pressed */
{
spp(15);
if ((CTLR  & (CTR_TIM_SUSP | CTR_LP_SUSP)) == (CTR_TIM_SUSP |
CTR_LP_SUSP))
{  /* If device is in suspend mode */
spp(RCCU_PG);
while (!(CLK_FLAG & 0x2));/* Test PLL lock-in */
CLK_FLAG |= 0x1;     /* Set CSU_SEL bit */

RemoteWakeup();
}
spp(WU_PG);
WU_WUPRL &= ~0x80; /* Clear pending bit */
}
RESTORE_PAGE;
}
```

4 REFERENCE DOCUMENTS

1. Universal Serial Bus Specification, Compaq Intel Microsoft NEC, Revision 1.1, September 23, 1998

2. ST92163 Datasheet, STMicroelectronics, Revision 1.9, January 2000