Power manager circuit driver for a handy power-pack over ATtiny4

1.4

Generated by Doxygen 1.8.18

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 ATtiny4.h File Reference

This header file contains the important definitions for ATtiny4 MCU.

Macros

- #define **SET_BIT**(REG, BIT) REG |= (1<<BIT) /*Sets the bit value to 1*/
- #define **CLEAR_BIT**(REG, BIT) REG &= ∼(1<<BIT) /*Clears the bit value to 0*/
- #define **GET_BIT**(REG, BIT) ((REG >> BIT) & 0x01) /*Get the bit value*/
- #define **SREG** (*(volatile u8_t*)(0x3F))
- #define **PUEB** (*(volatile u8_t*)(0x03))
- #define **PORTB** (*(volatile u8_t*)(0x02))
- #define **DDRB** (*(volatile u8_t*)(0x01))
- #define **PINB** (*(volatile u8_t*)(0x00))
- #define **TCCR0** (*(volatile u16_t*)(0x2D))
- #define **TCNT0** (*(volatile u16_t*)(0x28))
- #define **OCR0A** (*(volatile u16_t*)(0x26))
- #define **TIMSK0** (*(volatile u8_t*)(0x2B))
- #define **CLKMSR** (*(volatile u8_t*)(0x37))
- #define **CLKPSR** (*(volatile u8_t*)(0x36))
- #define **CCP** (*(volatile u8_t*)(0x3C))
- #define **SREG_IBIT** (7)
- #define **PORTB_PB0** (0)
- #define **PORTB_PB1** (1)
- #define **PORTB_PB2** (2)
- #define **PORTB_PB3** (3)
- #define **DDRB_PB0** (0)
- #define **DDRB_PB1** (1)
- #define **DDRB_PB2** (2)
- #define **DDRB_PB3** (3)
- #define **PINB_PB0** (0)
- #define **PINB_PB1** (1)
- #define **PINB_PB2** (2)
- #define **PINB_PB3** (3)

Typedefs

- typedef unsigned char **u8_t**
- typedef unsigned short **u16_t**

2.1.1 Detailed Description

This header file contains the important definitions for ATtiny4 MCU.

Author

Ahmed Ashraf (ahmedashrafelnaqeeb@gmail.com)

Version

1.2

Date

2020-07-13

Copyright

Copyright (c) 2020

2.2 Functionality.c File Reference

This file contains the interfacing functions logic implementation for the power manager application.

```
#include "ATtiny4.h"
#include "Functionality.h"
#include "util/delay.h"
#include "avr/sleep.h"
```

Macros

- #define **F_CPU** 31250UL
- #define **TIMER0_CTC_MODE_SELECTION** (0x0008)
- #define **TIMER0_50MS_TICK** (1563)
- #define **TIMER0_PRESCALER_1** (0x0001)
- #define **TIMER0_OCR0A_INT_EN** (0x02)
- #define **IO_PINS_DIR_INITIALIZATION** (0x01)
- #define **IO_LOW_LEVEL** (0)
- #define **IO_HIGH_LEVEL** (1)
- #define **IO_PB2_PULLUP_ENABLE** (0x04)
- #define **IO_PB0_LL** (0x00)
- #define **IO_PB0_HL** (0x01)
- #define **SYSTEM_OFF_STATUS** (0xAA)
- #define **SYSTEM_ON_STATUS** (0x55)
- #define **ONE_SECOND** (20)
- #define **TWO_SECONDS** (40)
- #define **TEN_SECONDS** (200)
- #define **INTERNAL_OSC_SELECT_8MHZ** (0x00)
- #define **ENABLE_CHANGE_FOR_IO_REG** (0xD8)
- #define **MAIN_CLK_PRESCALING_BY_256** (0x08)

Functions

- u8_t gu8_systemStatus **__attribute__** ((section(".noinit")))
- void attiny4_init (void)

  This function is responsible for initializing the ATtiny MCU and activate the power down mode.
- void mainApplication (void)

  This function is responsible for applying the state machine of the power manager system and making a transition from state to another
- void **OCR0A_ISR** (void)

Variables

- u16_t **gu16_switchCounter** = 0

2.2.1 Detailed Description

This file contains the interfacing functions logic implementation for the power manager application.

Author

Ahmed Ashraf (`ahmedashrafelnaqeeb@gmail.com`)

Version

1.4

Date

2020-07-13

Copyright

Copyright (c) 2020

2.2.2 Function Documentation

2.2.2.1 attiny4_init()

```
void attiny4_init (
            void  )
```

This function is responsible for initializing the ATtiny MCU and activate the power down mode.

DIO initialization section

IO Pins initialization by: PB0 -> Output PB1 -> Input PB2 -> Input PB3 -> Input

Adjusting the MCU CLK section

Timer initialization section

Definition at line 62 of file Functionality.c.

```
{
    /*Check the current state of the system to turn it OFF or ON*/
    if( gu8_systemStatus == SYSTEM_ON_STATUS )
    {
        /*If the system is already ON then set PB0 to +5v voltage level*/
        PORTB = IO_PB0_HL;
    }
    else if( gu8_systemStatus == SYSTEM_OFF_STATUS )
    {
        /*If the system is already ON then set PB0 to 0v voltage level*/
        PORTB = IO_PB0_LL;
    }
    else
    {
        /*Report that the system is in OFF mode*/
        gu8_systemStatus = SYSTEM_OFF_STATUS;
    }

    DDRB = IO_PINS_DIR_INITIALIZATION;

    /*Enabling the pull up resistor for PB2*/
    PUEB = IO_PB2_PULLUP_ENABLE;


     /*Select the internal oscillator of the MCU with 8MHz*/
     CLKMSR = INTERNAL_OSC_SELECT_8MHZ;

     /*Enable writing to the CLKPSR register*/
     CCP = ENABLE_CHANGE_FOR_IO_REG;

     /*Enable the pre-scaler of the main CLK by 256 which gives 31.25 KHz*/
     CLKPSR = MAIN_CLK_PRESCALING_BY_256;


     /*Selecting CTC mode with OCR0A*/
     TCCR0 = TIMER0_CTC_MODE_SELECTION;

     /*Clearing timer/counter register*/
     TCNT0 = 0;

     /*Adjusting TIMER0 to fire CTC interrupt every 50ms for 8MHz frequency and pre-scaler by 8*/
     OCR0A = TIMER0_50MS_TICK;

     /*Enable CTC mode interrupt*/
     TIMSK0 = TIMER0_OCR0A_INT_EN;


     /*Enable global interrupts*/
     SET_BIT(SREG , SREG_IBIT);

     return;
}
```

2.2.2.2 mainApplication()

```
void mainApplication (
              void  )
```

This function is responsible for applying the state machine of the power manager system and making a transition from state to another

Definition at line 135 of file Functionality.c.

```
{
    /*Check if the switch over PB2 is pressed or not*/
    if( GET_BIT(PINB , PINB_PB2) == IO_LOW_LEVEL )
    {
        /*If the switch is pressed for more than one second and the system is in OFF mode then go to ON mode*/
        if( (gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) && (gu8_systemStatus == SYSTEM_OFF_STATUS) )
        {
            /*Report that the system is in ON mode*/
            gu8_systemStatus = SYSTEM_ON_STATUS;

            /*Set the switch counter to two seconds count*/
            gu16_switchCounter = TWO_SECONDS;

            /*Activate PB0*/
            SET_BIT(PORTB , PORTB_PB0);
        }

        /*If the switch is pressed for more than one second and the system is in ON mode then go to OFF mode*/
        else if( ((gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) && (gu8_systemStatus == SYSTEM_ON_STATUS)) || (gu16_switchCounter >= TEN_SECONDS) )
        {
            /*Report that the system is in OFF mode*/
            gu8_systemStatus = SYSTEM_OFF_STATUS;

            /*De-activate PB0*/
            CLEAR_BIT(PORTB , PORTB_PB0);

            /*Select the power down mode*/
            set_sleep_mode(SLEEP_MODE_PWR_DOWN);

            /*Sleep enable*/
            sleep_enable();

            /*Execute sleep instruction*/
            sleep_cpu();
        }

        /*If the switch counter is reset then enable the timer and increase the switch counter by 1*/
        else if( gu16_switchCounter == 0 )
        {
            /*Turn ON the timer to measure the switch pressing time*/
            TCCR0 |= TIMER0_PRESCALER_1;

            /*Increase the switch counter by 1*/
            gu16_switchCounter++;
        }

        /*If nothing happens then enter IDLE mode until the timer fires its interrupt*/
        else
        {
            /*Select the idle mode*/
            set_sleep_mode(SLEEP_MODE_IDLE);

            /*Sleep enable*/
            sleep_enable();

            /*Execute sleep instruction*/
            sleep_cpu();
        }
    }
    else if( GET_BIT(PINB , PINB_PB2) == IO_HIGH_LEVEL )
    {
        /*Delay to make sure the bouncing has gone*/
        _delay_ms(50);

        /*Select the power down mode*/
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);

        /*Sleep enable*/
        sleep_enable();

        /*Execute sleep instruction*/
        sleep_cpu();
    }
    else
    {
        /*Do nothing*/
    }

    return;
}
```

2.3 Functionality.h File Reference

This header file contains power manager interfacing functions' prototypes.

Macros

- #define **EXTI0_ISR** __vector_1
- #define **OCR0A_ISR** __vector_5

Functions

- void attiny4_init (void)

  This function is responsible for initializing the ATtiny MCU and activate the power down mode.
- void mainApplication (void)

  This function is responsible for applying the state machine of the power manager system and making a transition from state to another

2.3.1 Detailed Description

This header file contains power manager interfacing functions' prototypes.

Author

Ahmed Ashraf (ahmedashrafelnaqeeb@gmail.com)

Version

1.0

Date

2020-07-12

Copyright

Copyright (c) 2020

2.3.2 Function Documentation

2.3.2.1 attiny4_init()

```
void attiny4_init (
               void  )
```

This function is responsible for initializing the ATtiny MCU and activate the power down mode.

DIO initialization section

IO Pins initialization by: PB0 -> Output PB1 -> Input PB2 -> Input PB3 -> Input

Adjusting the MCU CLK section

Timer initialization section

Definition at line 62 of file Functionality.c.

```
63 {
68     /*Check the current state of the system to turn it OFF or ON*/
69     if( gu8_systemStatus == SYSTEM_ON_STATUS )
70     {
71         /*If the system is already ON then set PB0 to +5v voltage level*/
72         PORTB = IO_PB0_HL;
73     }
74     else if( gu8_systemStatus == SYSTEM_OFF_STATUS )
75     {
76         /*If the system is already ON then set PB0 to 0v voltage level*/
77         PORTB = IO_PB0_LL;
78     }
79     else
80     {
81         /*Report that the system is in OFF mode*/
82         gu8_systemStatus = SYSTEM_OFF_STATUS;
83     }
84 
92     DDRB = IO_PINS_DIR_INITIALIZATION;
93 
94     /*Enabling the pull up resistor for PB2*/
95     PUEB = IO_PB2_PULLUP_ENABLE;
96 
97 
102     /*Select the internal oscillator of the MCU with 8MHz*/
103     CLKMSR = INTERNAL_OSC_SELECT_8MHZ;
104 
105     /*Enable writing to the CLKPSR register*/
106     CCP = ENABLE_CHANGE_FOR_IO_REG;
107 
108     /*Enable the pre-scaler of the main CLK by 256 which gives 31.25 KHz*/
109     CLKPSR = MAIN_CLK_PRESCALING_BY_256;
110 
111 
116     /*Selecting CTC mode with OCR0A*/
117     TCCR0 = TIMER0_CTC_MODE_SELECTION;
118 
119     /*Clearing timer/counter register*/
120     TCNT0 = 0;
121 
122     /*Adjusting TIMER0 to fire CTC interrupt every 50ms for 8MHz frequency and pre-scaler by 8*/
123     OCR0A = TIMER0_50MS_TICK;
124 
125     /*Enable CTC mode interrupt*/
126     TIMSK0 = TIMER0_OCR0A_INT_EN;
127 
128 
129     /*Enable global interrupts*/
130     SET_BIT(SREG , SREG_IBIT);
131 
132     return;
133 }
```

2.3.2.2 mainApplication()

```
void mainApplication (
            void  )
```

This function is responsible for applying the state machine of the power manager system and making a transition from state to another

Definition at line 135 of file Functionality.c.

```
{
    /*Check if the switch over PB2 is pressed or not*/
    if( GET_BIT(PINB , PINB_PB2) == IO_LOW_LEVEL )
    {
        /*If the switch is pressed for more than one second and the system is in OFF mode then go to ON
mode*/
        if( (gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) && (gu8_systemStatus
== SYSTEM_OFF_STATUS) )
        {
            /*Report that the system is in ON mode*/
            gu8_systemStatus = SYSTEM_ON_STATUS;

            /*Set the switch counter to two seconds count*/
            gu16_switchCounter = TWO_SECONDS;

            /*Activate PB0*/
            SET_BIT(PORTB , PORTB_PB0);
        }

        /*If the switch is pressed for more than one second and the system is in ON mode then go to OFF
mode*/
        else if( ((gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) &&
(gu8_systemStatus == SYSTEM_ON_STATUS)) || (gu16_switchCounter >= TEN_SECONDS) )
        {
            /*Report that the system is in OFF mode*/
            gu8_systemStatus = SYSTEM_OFF_STATUS;

            /*De-activate PB0*/
            CLEAR_BIT(PORTB , PORTB_PB0);

            /*Select the power down mode*/
            set_sleep_mode(SLEEP_MODE_PWR_DOWN);

            /*Sleep enable*/
            sleep_enable();

            /*Execute sleep instruction*/
            sleep_cpu();
        }

        /*If the switch counter is reset then enable the timer and increase the switch counter by 1*/
        else if( gu16_switchCounter == 0 )
        {
            /*Turn ON the timer to measure the switch pressing time*/
            TCCR0 |= TIMER0_PRESCALER_1;

            /*Increase the switch counter by 1*/
            gu16_switchCounter++;
        }

        /*If nothing happens then enter IDLE mode until the timer fires its interrupt*/
        else
        {
            /*Select the idle mode*/
            set_sleep_mode(SLEEP_MODE_IDLE);

            /*Sleep enable*/
            sleep_enable();

            /*Execute sleep instruction*/
            sleep_cpu();
        }
    }
    else if( GET_BIT(PINB , PINB_PB2) == IO_HIGH_LEVEL )
    {
        /*Delay to make sure the bouncing has gone*/
        _delay_ms(50);

        /*Select the power down mode*/
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);

        /*Sleep enable*/
        sleep_enable();

        /*Execute sleep instruction*/
        sleep_cpu();
    }
    else
    {
        /*Do nothing*/
    }

    return;
}
```

2.4 main.c File Reference

This file contains the starting point (main function) of the power manager application.

```
#include "Functionality.h"
```

Functions

- int main (void)

    This the entry point of the power manager application.

2.4.1 Detailed Description

This file contains the starting point (main function) of the power manager application.

Author

Ahmed Ashraf (ahmedashrafelnaqeeb@gmail.com)

Version

1.0

Date

2020-07-12

Copyright

Copyright (c) 2020

2.4.2 Function Documentation

2.4.2.1 main()

```
int main (
            void  )
```

This the entry point of the power manager application.

Returns

int 0 if everything is good and another value if there's an error

Definition at line 28 of file main.c.

```
{
    /*Initializing the power manager circuit*/
    attiny4_init();

    while(1)
    {
        /*The main operation of the power manager circuit*/
        mainApplication();
    }
    return 0;
}
```

Index