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

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_PB0_LL** (0x00)
- #define **IO_PB0_HL** (0x01)
- #define **IO_PB0_MOSFET** (PORTB_PB0)
- #define **IO_PB2_SWITCH** (PINB_PB2)
- #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)
- #define **DELAY_50MS** (50)

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

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 64 of file Functionality.c.

```
65 {
70     /*Check the current state of the system to turn it OFF or ON*/
71     if( gu8_systemStatus == SYSTEM_ON_STATUS )
72     {
73         /*If the system is already ON then set PB0 to +5v voltage level*/
74         PORTB = IO_PB0_HL;
75     }
76     else if( gu8_systemStatus == SYSTEM_OFF_STATUS )
77     {
78         /*If the system is already ON then set PB0 to 0v voltage level*/
79         PORTB = IO_PB0_LL;
80     }
81     else
82     {
83         /*Report that the system is in OFF mode*/
84         gu8_systemStatus = SYSTEM_OFF_STATUS;
85     }
86
94     DDRB = IO_PINS_DIR_INITIALIZATION;
95
96
101     /*Select the internal oscillator of the MCU with 8MHz*/
102     CLKMSR = INTERNAL_OSC_SELECT_8MHZ;
103
104     /*Enable writing to the CLKPSR register*/
105     CCP = ENABLE_CHANGE_FOR_IO_REG;
106
107     /*Enable the pre-scaler of the main CLK by 256 which gives 31.25 KHz*/
108     CLKPSR = MAIN_CLK_PRESCALING_BY_256;
109
110
115     /*Selecting CTC mode with OCR0A*/
116     TCCR0 = TIMER0_CTC_MODE_SELECTION;
117
118     /*Clearing timer/counter register*/
119     TCNT0 = 0;
120
121     /*Adjusting TIMER0 to fire CTC interrupt every 50ms for 8MHz frequency and pre-scaler by 8*/
122     OCR0A = TIMER0_50MS_TICK;
123
124     /*Enable CTC mode interrupt*/
125     TIMSK0 = TIMER0_OCR0A_INT_EN;
126
127     /*Enable global interrupts*/
128     SET_BIT(SREG , SREG_IBIT);
129
130     return;
131 }
```

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 133 of file Functionality.c.

```
134 {
135     /*Check if the switch over PB2 is pressed or not*/
136     if( GET_BIT(PINB , IO_PB2_SWITCH) == IO_LOW_LEVEL )
137     {
138         /*If the switch is pressed for more than one second and the system is in OFF mode then go to ON
      mode*/
139         if( (gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) && (gu8_systemStatus
      == SYSTEM_OFF_STATUS) )
140         {
141             /*Report that the system is in ON mode*/
142             gu8_systemStatus = SYSTEM_ON_STATUS;
143 
144             /*Set the switch counter to two seconds count*/
145             gu16_switchCounter = TWO_SECONDS;
146 
147             /*Activate Mosfet over PB0*/
148             SET_BIT(PORTB , IO_PB0_MOSFET);
149         }
150 
151         /*If the switch is pressed for more than one second and the system is in ON mode then go to OFF
      mode*/
152         else if( ((gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) &&
      (gu8_systemStatus == SYSTEM_ON_STATUS)) || (gu16_switchCounter >= TEN_SECONDS) )
153         {
154             /*Report that the system is in OFF mode*/
155             gu8_systemStatus = SYSTEM_OFF_STATUS;
156 
157             /*De-activate Mosfet over PB0*/
158             CLEAR_BIT(PORTB , IO_PB0_MOSFET);
159 
160             /*Disable global interrupts*/
161             CLEAR_BIT(SREG , SREG_IBIT);
162 
163             /*Disable the timer*/
164             TCCR0 = 0;
165 
166             /*Select the power down mode*/
167             set_sleep_mode(SLEEP_MODE_PWR_DOWN);
168 
169             /*Sleep enable*/
170             sleep_enable();
171 
172             /*Execute sleep instruction*/
173             sleep_cpu();
174         }
175 
176         /*If the switch counter is reset then enable the timer and increase the switch counter by 1*/
177         else if( gu16_switchCounter == 0 )
178         {
179             /*Turn ON the timer to measure the switch pressing time*/
180             TCCR0 |= TIMER0_PRESCALER_1;
181 
182             /*Increase the switch counter by 1*/
183             gu16_switchCounter++;
184         }
185 
186         /*If nothing happens then enter IDLE mode until the timer fires its interrupt*/
187         else
188         {
189             /*Select the idle mode*/
190             set_sleep_mode(SLEEP_MODE_IDLE);
191 
192             /*Sleep enable*/
193             sleep_enable();
194 
195             /*Execute sleep instruction*/
196             sleep_cpu();
197         }
198     }
199     else if( GET_BIT(PINB , IO_PB2_SWITCH) == IO_HIGH_LEVEL )
200     {
201         /*Delay to make sure the bouncing has gone*/
```

```
202         _delay_ms(DELAY_50MS);
203
204         /*Disable global interrupts*/
205         CLEAR_BIT(SREG , SREG_IBIT);
206
207         /*Disable the timer*/
208         TCCR0 = 0;
209
210         /*Select the power down mode*/
211         set_sleep_mode(SLEEP_MODE_PWR_DOWN);
212
213         /*Sleep enable*/
214         sleep_enable();
215
216         /*Execute sleep instruction*/
217         sleep_cpu();
218     }
219     else
220     {
221         /*Do nothing*/
222     }
223
224     return;
225 }
```

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 64 of file Functionality.c.

```
65 {
70     /*Check the current state of the system to turn it OFF or ON*/
71     if( gu8_systemStatus == SYSTEM_ON_STATUS )
72     {
73         /*If the system is already ON then set PB0 to +5v voltage level*/
74         PORTB = IO_PB0_HL;
75     }
76     else if( gu8_systemStatus == SYSTEM_OFF_STATUS )
77     {
78         /*If the system is already ON then set PB0 to 0v voltage level*/
79         PORTB = IO_PB0_LL;
80     }
81     else
82     {
83         /*Report that the system is in OFF mode*/
84         gu8_systemStatus = SYSTEM_OFF_STATUS;
85     }
86
94     DDRB = IO_PINS_DIR_INITIALIZATION;
95
96
101     /*Select the internal oscillator of the MCU with 8MHz*/
102     CLKMSR = INTERNAL_OSC_SELECT_8MHZ;
103
104     /*Enable writing to the CLKPSR register*/
105     CCP = ENABLE_CHANGE_FOR_IO_REG;
106
107     /*Enable the pre-scaler of the main CLK by 256 which gives 31.25 KHz*/
108     CLKPSR = MAIN_CLK_PRESCALING_BY_256;
109
110
115     /*Selecting CTC mode with OCR0A*/
116     TCCR0 = TIMER0_CTC_MODE_SELECTION;
117
118     /*Clearing timer/counter register*/
119     TCNT0 = 0;
120
121     /*Adjusting TIMER0 to fire CTC interrupt every 50ms for 8MHz frequency and pre-scaler by 8*/
122     OCR0A = TIMER0_50MS_TICK;
123
124     /*Enable CTC mode interrupt*/
125     TIMSK0 = TIMER0_OCR0A_INT_EN;
126
127     /*Enable global interrupts*/
128     SET_BIT(SREG , SREG_IBIT);
129
130     return;
131 }
```

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 133 of file Functionality.c.

```
134 {
135     /*Check if the switch over PB2 is pressed or not*/
136     if( GET_BIT(PINB , IO_PB2_SWITCH) == IO_LOW_LEVEL )
137     {
138         /*If the switch is pressed for more than one second and the system is in OFF mode then go to ON
      mode*/
139         if( (gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) && (gu8_systemStatus
      == SYSTEM_OFF_STATUS) )
140         {
141             /*Report that the system is in ON mode*/
142             gu8_systemStatus = SYSTEM_ON_STATUS;
143 
144             /*Set the switch counter to two seconds count*/
145             gu16_switchCounter = TWO_SECONDS;
146 
147             /*Activate Mosfet over PB0*/
148             SET_BIT(PORTB , IO_PB0_MOSFET);
149         }
150 
151         /*If the switch is pressed for more than one second and the system is in ON mode then go to OFF
      mode*/
152         else if( ((gu16_switchCounter > ONE_SECOND && gu16_switchCounter < TWO_SECONDS) &&
      (gu8_systemStatus == SYSTEM_ON_STATUS)) || (gu16_switchCounter >= TEN_SECONDS) )
153         {
154             /*Report that the system is in OFF mode*/
155             gu8_systemStatus = SYSTEM_OFF_STATUS;
156 
157             /*De-activate Mosfet over PB0*/
158             CLEAR_BIT(PORTB , IO_PB0_MOSFET);
159 
160             /*Disable global interrupts*/
161             CLEAR_BIT(SREG , SREG_IBIT);
162 
163             /*Disable the timer*/
164             TCCR0 = 0;
165 
166             /*Select the power down mode*/
167             set_sleep_mode(SLEEP_MODE_PWR_DOWN);
168 
169             /*Sleep enable*/
170             sleep_enable();
171 
172             /*Execute sleep instruction*/
173             sleep_cpu();
174         }
175 
176         /*If the switch counter is reset then enable the timer and increase the switch counter by 1*/
177         else if( gu16_switchCounter == 0 )
178         {
179             /*Turn ON the timer to measure the switch pressing time*/
180             TCCR0 |= TIMER0_PRESCALER_1;
181 
182             /*Increase the switch counter by 1*/
183             gu16_switchCounter++;
184         }
185 
186         /*If nothing happens then enter IDLE mode until the timer fires its interrupt*/
187         else
188         {
189             /*Select the idle mode*/
190             set_sleep_mode(SLEEP_MODE_IDLE);
191 
192             /*Sleep enable*/
193             sleep_enable();
194 
195             /*Execute sleep instruction*/
196             sleep_cpu();
197         }
198     }
199     else if( GET_BIT(PINB , IO_PB2_SWITCH) == IO_HIGH_LEVEL )
200     {
201         /*Delay to make sure the bouncing has gone*/
```

```
202         _delay_ms(DELAY_50MS);
203
204         /*Disable global interrupts*/
205         CLEAR_BIT(SREG , SREG_IBIT);
206
207         /*Disable the timer*/
208         TCCR0 = 0;
209
210         /*Select the power down mode*/
211         set_sleep_mode(SLEEP_MODE_PWR_DOWN);
212
213         /*Sleep enable*/
214         sleep_enable();
215
216         /*Execute sleep instruction*/
217         sleep_cpu();
218     }
219     else
220     {
221         /*Do nothing*/
222     }
223
224     return;
225 }
```

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.

```
29 {
30     /*Initializing the power manager circuit*/
31     attiny4_init();
32 
33     while(1)
34     {
35         /*The main operation of the power manager circuit*/
36         mainApplication();
37     }
38     return 0;
39 }
```

Index