

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

1.4

Generated by Doxygen 1.8.18

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 ATtiny4.h File Reference	3
2.1.1 Detailed Description	4
2.2 Functionality.c File Reference	4
2.2.1 Detailed Description	5
2.2.2 Function Documentation	5
2.2.2.1 attiny4_init()	6
2.2.2.2 mainApplication()	7
2.3 Functionality.h File Reference	8
2.3.1 Detailed Description	8
2.3.2 Function Documentation	9
2.3.2.1 attiny4_init()	9
2.3.2.2 mainApplication()	10
2.4 main.c File Reference	11
2.4.1 Detailed Description	11
2.4.2 Function Documentation	12
2.4.2.1 main()	12
Index	13

Chapter 1

File Index

1.1 File List

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

ATtiny4.h	This header file contains the important definitions for ATtiny4 MCU	3
Functionality.c	This file contains the interfacing functions logic implementation for the power manager application	4
Functionality.h	This header file contains power manager interfacing functions' prototypes	8
main.c	This file contains the starting point (main function) of the power manager application	11

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.

```
63 {
64     /*Check the current state of the system to turn it OFF or ON*/
65     if( gu8_systemStatus == SYSTEM_ON_STATUS )
66     {
67         /*If the system is already ON then set PB0 to +5v voltage level*/
68         PORTB = IO_PB0_HL;
69     }
70     else if( gu8_systemStatus == SYSTEM_OFF_STATUS )
71     {
72         /*If the system is already ON then set PB0 to 0v voltage level*/
73         PORTB = IO_PB0_LL;
74     }
75     else
76     {
77         /*Report that the system is in OFF mode*/
78         gu8_systemStatus = SYSTEM_OFF_STATUS;
79     }
80
81     DDRB = IO_PINS_DIR_INITIALIZATION;
82
83     /*Enabling the pull up resistor for PB2*/
84     PUEB = IO_PB2_PULLUP_ENABLE;
85
86     /*Select the internal oscillator of the MCU with 8MHz*/
87     CLKMSR = INTERNAL_OSC_SELECT_8MHZ;
88
89     /*Enable writing to the CLKPSR register*/
90     CCP = ENABLE_CHANGE_FOR_IO_REG;
91
92     /*Enable the pre-scaler of the main CLK by 256 which gives 31.25 KHz*/
93     CLKPSR = MAIN_CLK_PRESCALING_BY_256;
94
95     /*Selecting CTC mode with OCR0A*/
96     TCCR0 = TIMER0_CTC_MODE_SELECTION;
97
98     /*Clearing timer/counter register*/
99     TCNT0 = 0;
100
101     /*Adjusting TIMER0 to fire CTC interrupt every 50ms for 8MHz frequency and pre-scaler by 8*/
102     OCR0A = TIMER0_50MS_TICK;
103
104     /*Enable CTC mode interrupt*/
105     TIMSK0 = TIMER0_OCR0A_INT_EN;
106
107     /*Enable global interrupts*/
108     SET_BIT(SREG , SREG_IBIT);
109
110     return;
111 }
```

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 134 of file Functionality.c.

```
135 {
136     /*Check if the switch over PB2 is pressed or not*/
137     if( GET_BIT(PINB , PINB_PB2) == IO_LOW_LEVEL )
138     {
139         /*If the switch is pressed for more than one second and the system is in OFF mode then go to ON
mode*/
140         if( (gul6_switchCounter > ONE_SECOND && gul6_switchCounter < TWO_SECONDS) && (gu8_systemStatus
== SYSTEM_OFF_STATUS) )
141         {
142             /*Report that the system is in ON mode*/
143             gu8_systemStatus = SYSTEM_ON_STATUS;
144
145             /*Set the switch counter to two seconds count*/
146             gul6_switchCounter = TWO_SECONDS;
147
148             /*Activate PB0*/
149             SET_BIT(PORTB , PORTB_PB0);
150         }
151
152         /*If the switch is pressed for more than one second and the system is in ON mode then go to OFF
mode*/
153         else if( ((gul6_switchCounter > ONE_SECOND && gul6_switchCounter < TWO_SECONDS) &&
(gu8_systemStatus == SYSTEM_ON_STATUS)) || (gul6_switchCounter >= TEN_SECONDS) )
154         {
155             /*Report that the system is in OFF mode*/
156             gu8_systemStatus = SYSTEM_OFF_STATUS;
157
158             /*De-activate PB0*/
159             CLEAR_BIT(PORTB , PORTB_PB0);
160
161             /*Disable global interrupts*/
162             CLEAR_BIT(SREG , SREG_IBIT);
163
164             /*Disable the timer*/
165             TCCR0 = 0;
166
167             /*Select the power down mode*/
168             set_sleep_mode(SLEEP_MODE_PWR_DOWN);
169
170             /*Sleep enable*/
171             sleep_enable();
172
173             /*Execute sleep instruction*/
174             sleep_cpu();
175         }
176
177         /*If the switch counter is reset then enable the timer and increase the switch counter by 1*/
178         else if( gul6_switchCounter == 0 )
179         {
180             /*Turn ON the timer to measure the switch pressing time*/
181             TCCR0 |= TIMER0_PRESCALER_1;
182
183             /*Increase the switch counter by 1*/
184             gul6_switchCounter++;
185         }
186
187         /*If nothing happens then enter IDLE mode until the timer fires its interrupt*/
188         else
189         {
190             /*Select the idle mode*/
191             set_sleep_mode(SLEEP_MODE_IDLE);
192
193             /*Sleep enable*/
194             sleep_enable();
195
196             /*Execute sleep instruction*/
197             sleep_cpu();
198         }
199     }
200     else if( GET_BIT(PINB , PINB_PB2) == IO_HIGH_LEVEL )
201     {
202         /*Delay to make sure the bouncing has gone*/
```

```

203     _delay_ms(50);
204
205     /*Disable global interrupts*/
206     CLEAR_BIT(SREG , SREG_IBIT);
207
208     /*Disable the timer*/
209     TCCR0 = 0;
210
211     /*Select the power down mode*/
212     set_sleep_mode(SLEEP_MODE_PWR_DOWN);
213
214     /*Sleep enable*/
215     sleep_enable();
216
217     /*Execute sleep instruction*/
218     sleep_cpu();
219 }
220 else
221 {
222     /*Do nothing*/
223 }
224
225 return;
226 }

```

2.3 Functionality.h File Reference

This header file contains power manager interfacing functions' prototypes.

Macros

- `#define EXTIO_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 {
64     /*Check the current state of the system to turn it OFF or ON*/
65     if( gu8_systemStatus == SYSTEM_ON_STATUS )
66     {
67         /*If the system is already ON then set PB0 to +5v voltage level*/
68         PORTB = IO_PB0_HL;
69     }
70     else if( gu8_systemStatus == SYSTEM_OFF_STATUS )
71     {
72         /*If the system is already ON then set PB0 to 0v voltage level*/
73         PORTB = IO_PB0_LL;
74     }
75     else
76     {
77         /*Report that the system is in OFF mode*/
78         gu8_systemStatus = SYSTEM_OFF_STATUS;
79     }
80
81     DDRB = IO_PINS_DIR_INITIALIZATION;
82
83     /*Enabling the pull up resistor for PB2*/
84     PUEB = IO_PB2_PULLUP_ENABLE;
85
86     /*Select the internal oscillator of the MCU with 8MHz*/
87     CLKMSR = INTERNAL_OSC_SELECT_8MHZ;
88
89     /*Enable writing to the CLKPSR register*/
90     CCP = ENABLE_CHANGE_FOR_IO_REG;
91
92     /*Enable the pre-scaler of the main CLK by 256 which gives 31.25 KHz*/
93     CLKPSR = MAIN_CLK_PRESCALING_BY_256;
94
95     /*Selecting CTC mode with OCR0A*/
96     TCCR0 = TIMER0_CTC_MODE_SELECTION;
97
98     /*Clearing timer/counter register*/
99     TCNT0 = 0;
100
101     /*Adjusting TIMER0 to fire CTC interrupt every 50ms for 8MHz frequency and pre-scaler by 8*/
102     OCR0A = TIMER0_50MS_TICK;
103
104     /*Enable CTC mode interrupt*/
105     TIMSK0 = TIMER0_OCR0A_INT_EN;
106
107     /*Enable global interrupts*/
108     SET_BIT(SREG , SREG_IBIT);
109
110     return;
111 }
```

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 134 of file Functionality.c.

```
135 {
136     /*Check if the switch over PB2 is pressed or not*/
137     if( GET_BIT(PINB , PINB_PB2) == IO_LOW_LEVEL )
138     {
139         /*If the switch is pressed for more than one second and the system is in OFF mode then go to ON
mode*/
140         if( (gul6_switchCounter > ONE_SECOND && gul6_switchCounter < TWO_SECONDS) && (gu8_systemStatus
== SYSTEM_OFF_STATUS) )
141         {
142             /*Report that the system is in ON mode*/
143             gu8_systemStatus = SYSTEM_ON_STATUS;
144
145             /*Set the switch counter to two seconds count*/
146             gul6_switchCounter = TWO_SECONDS;
147
148             /*Activate PB0*/
149             SET_BIT(PORTB , PORTB_PB0);
150         }
151
152         /*If the switch is pressed for more than one second and the system is in ON mode then go to OFF
mode*/
153         else if( ((gul6_switchCounter > ONE_SECOND && gul6_switchCounter < TWO_SECONDS) &&
(gu8_systemStatus == SYSTEM_ON_STATUS)) || (gul6_switchCounter >= TEN_SECONDS) )
154         {
155             /*Report that the system is in OFF mode*/
156             gu8_systemStatus = SYSTEM_OFF_STATUS;
157
158             /*De-activate PB0*/
159             CLEAR_BIT(PORTB , PORTB_PB0);
160
161             /*Disable global interrupts*/
162             CLEAR_BIT(SREG , SREG_IBIT);
163
164             /*Disable the timer*/
165             TCCR0 = 0;
166
167             /*Select the power down mode*/
168             set_sleep_mode(SLEEP_MODE_PWR_DOWN);
169
170             /*Sleep enable*/
171             sleep_enable();
172
173             /*Execute sleep instruction*/
174             sleep_cpu();
175         }
176
177         /*If the switch counter is reset then enable the timer and increase the switch counter by 1*/
178         else if( gul6_switchCounter == 0 )
179         {
180             /*Turn ON the timer to measure the switch pressing time*/
181             TCCR0 |= TIMER0_PRESCALER_1;
182
183             /*Increase the switch counter by 1*/
184             gul6_switchCounter++;
185         }
186
187         /*If nothing happens then enter IDLE mode until the timer fires its interrupt*/
188         else
189         {
190             /*Select the idle mode*/
191             set_sleep_mode(SLEEP_MODE_IDLE);
192
193             /*Sleep enable*/
194             sleep_enable();
195
196             /*Execute sleep instruction*/
197             sleep_cpu();
198         }
199     }
200     else if( GET_BIT(PINB , PINB_PB2) == IO_HIGH_LEVEL )
201     {
202         /*Delay to make sure the bouncing has gone*/
```

```
203     _delay_ms(50);
204
205     /*Disable global interrupts*/
206     CLEAR_BIT(SREG , SREG_IBIT);
207
208     /*Disable the timer*/
209     TCCR0 = 0;
210
211     /*Select the power down mode*/
212     set_sleep_mode(SLEEP_MODE_PWR_DOWN);
213
214     /*Sleep enable*/
215     sleep_enable();
216
217     /*Execute sleep instruction*/
218     sleep_cpu();
219 }
220 else
221 {
222     /*Do nothing*/
223 }
224
225 return;
226 }
```

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

ATtiny4.h, [3](#)

attiny4_init

 Functionality.c, [5](#)

 Functionality.h, [9](#)

Functionality.c, [4](#)

 attiny4_init, [5](#)

 mainApplication, [6](#)

Functionality.h, [8](#)

 attiny4_init, [9](#)

 mainApplication, [9](#)

main

 main.c, [12](#)

main.c, [11](#)

 main, [12](#)

mainApplication

 Functionality.c, [6](#)

 Functionality.h, [9](#)