AVR Timer1

Generated by Doxygen 1.8.6

Sun Aug 9 2015 03:31:10

Contents

1 File Index **1**

1.1 File List 1

2 File Documentation **3**

2.1 timer_1.c File Reference 3

2.1.1 Macro Definition Documentation 4

2.1.1.1 BAUD 4

2.1.1.2 F_CPU 4

2.1.1.3 FOSC 4

2.1.1.4 MYUBRR 4

2.1.2 Function Documentation 4

2.1.2.1 ISR 4

2.1.2.2 main 4

2.1.2.3 timer1_init 5

2.1.2.4 USART_init 5

2.1.2.5 USART_receive 5

2.1.2.6 USART_send 6

2.1.3 Variable Documentation 6

2.1.3.1 n 6

Index **7**

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

timer_1.c 3

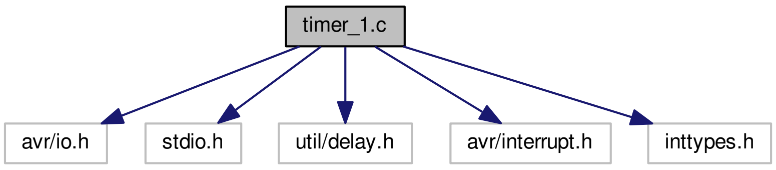
Chapter 2

File Documentation

2.1 timer_1.c File Reference

```
#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <inttypes.h>
```

Include dependency graph for timer_1.c:



Macros

- #define F_CPU 16000000UL
- #define FOSC 16000000
- #define BAUD 9600
- #define MYUBRR FOSC/16/BAUD-1

Functions

- void USART_init (unsigned int ubrr)

  Initialize USART for 8 bit data transmit no parity and 1 stop bit.
- int USART_send (char c, FILE ∗stream)

  Send 8bit data.
- int USART_receive (FILE ∗stream)

  Receive 8bit data.
- ISR (TIMER1_OVF_vect)

  ISR for TIMER1 overflow. Increase value of n.

- void timer1_init ()
- int main ()

  Program entry point.

Variables

- volatile uint32_t n

2.1.1 Macro Definition Documentation

2.1.1.1 #define BAUD 9600

Baud Rate in bps. refer page 179 of 328p datasheet.

2.1.1.2 #define F_CPU 16000000UL

Clock speed for delay functions.

2.1.1.3 #define FOSC 16000000

Clock speed for UBRR calculation. refer page 179 of 328p datasheet.

2.1.1.4 #define MYUBRR FOSC/16/BAUD-1

UBRR = (F_CPU/(16∗Baud))-1 for asynch USART page 179 328p datasheet. Baud rate 9600bps, assuming 16MHz clock UBRR0 becomes 0x0067

2.1.2 Function Documentation

2.1.2.1 ISR (TIMER1_OVF_vect)

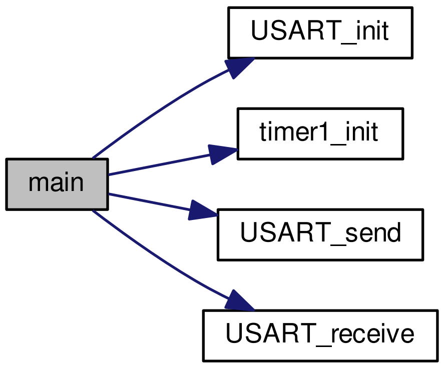
ISR for TIMER1 overflow. Increase value of n.

2.1.2.2 int main ( )

Program entry point.

Initialize the timer1 register. Initialize usart communication. Initialize the standard IO handlers for printf. Initialize the standard IO handlers for scanf. Reset counter. Reset Timer1. Enable global interrupt. run a random loop. print the result. Target Period = ((2 ^ 16) ∗ (Prescale / 16000000) = ((2 ^ 16) ∗ (64 / 16000000) = 0.262144 = 2621 uS (approx.)

Here is the call graph for this function:



2.1.2.3 void timer1_init ( )

Step 1. normal mode

Step 2. 1:64 prescaler, internal clock

Step 3. enable Timer 1 overflow interrupt

2.1.2.4 void USART_init (unsigned int *ubrr*)

Initialize USART for 8 bit data transmit no parity and 1 stop bit.

This is a code snippet from datasheet page 182

Parameters

ubrr	The UBRR value calculated in macro MYUBRR

See Also

MYUBRR

Step 1. Set UCSR0C in Asynchronous mode, no parity, 1 stop bit, 8 data bits

Step 2. Set UCSR0A in Normal speed, disable multi-proc

Step 3. Load ubrr into UBRR0H and UBRR0L

Step 4. Enable Tx Rx and disable interrupt in UCSR0B

2.1.2.5 int USART_receive (FILE ∗ *stream*)

Receive 8bit data.

This is a code snippet from datasheet page 187

Returns

Returns received data from UDR0

Step 1. Wait for Receive Complete Flag is high. Busy waiting

Step 2. Get and return received data from buffer

2.1.2.6 int USART_send (char *c*, FILE ∗ *stream*)

Send 8bit data.

This is a code snippet from datasheet page 184

Parameters

c	The 8 bit data to be sent
FILE	∗stream to receive

Step 1. Wait until UDRE0 flag is high. Busy Waitinig

Step 2. Write char to UDR0 for transmission

2.1.3 Variable Documentation

2.1.3.1 volatile uint32_t n

n to count elapsed time

Index

BAUD
    timer_1.c, 4

F_CPU
    timer_1.c, 4
FOSC
    timer_1.c, 4

ISR
    timer_1.c, 4

MYUBRR
    timer_1.c, 4
main
    timer_1.c, 4

n
    timer_1.c, 6

timer1_init
    timer_1.c, 5
timer_1.c, 3
    BAUD, 4
    F_CPU, 4
    FOSC, 4
    ISR, 4
    MYUBRR, 4
    main, 4
    n, 6
    timer1_init, 5
    USART_init, 5
    USART_receive, 5
    USART_send, 6

USART_init
    timer_1.c, 5
USART_receive
    timer_1.c, 5
USART_send
    timer_1.c, 6