Assignment 1

ENS2257: Microprocessor Systems

David Duke – 10374985

Qj Stout-Spykers – 10223002

2017

Table of Contents

[Tables 2](#_Toc494206457)

[Figures 2](#_Toc494206458)

[Equations 2](#_Toc494206459)

[Task 1 3](#_Toc494206460)

[Single Digit LED Numeric Display 3](#_Toc494206461)

[Atmel ATmega168 Microcontroller 4](#_Toc494206462)

[Additional components 5](#_Toc494206463)

[Circuit Diagram 6](#_Toc494206464)

[Seven segment display interface requirements 7](#_Toc494206465)

[C Subroutine code (display 0 to F by parameter passed to subroutine) 8](#_Toc494206466)

[Assembler Routine that writes 0 to the seven-segment display 9](#_Toc494206467)

[Task 2 9](#_Toc494206468)

[Description of principles of solution 9](#_Toc494206469)

[Description of all timers/counters, registers and interrupts used 9](#_Toc494206470)

[PRR – Power Reduction Register 9](#_Toc494206471)

[TCCR1A/ TCCR1B – Timer/Counter1 Control Register A/B 9](#_Toc494206472)

[OCR1A – Output Compare Register 1 A 10](#_Toc494206473)

[TIMSK1 – Timer/Counter1 Interrupt Mask Register 10](#_Toc494206474)

[TCNT1 – Timer/Counter1 10](#_Toc494206475)

[TIFR1 – Timer/Counter1 Interrupt Flag Register 10](#_Toc494206476)

[Details of configuration information (with justification/explanation/calculations) 11](#_Toc494206477)

[Register configuration 11](#_Toc494206478)

[Configuration justification 11](#_Toc494206479)

[Description of algorithm(s) (with code fragments or pseudocode) 12](#_Toc494206480)

[Global declarations 12](#_Toc494206481)

[Timer delay function 12](#_Toc494206482)

[Initializing the Timer/Counter 13](#_Toc494206483)

[ISR routine 13](#_Toc494206484)

[References 14](#_Toc494206485)

# Tables

[Table 1 : Absolute maximum ratings for LA-601AB (ROHM, 2008) 3](#_Toc494206486)

[Table 2 : Electrical Characteristics for LA-601AB (ROHM, 2008) 3](#_Toc494206487)

[Table 3: Absolute maximum ratings for ATmega168 (Atmel, 2007) 4](#_Toc494206488)

[Table 4: LED pin connection reference table 7](#_Toc494206489)

[Table 5: Truth table for the seven-segment display 7](#_Toc494206490)

[Table 6 : Timer/Counter registers used (Atmel, 2007) 10](#_Toc494206491)

[Table 7: Shows the configuration of all registers relevant to the solution. 11](#_Toc494206492)

[Table 8: Clock Select configuration 11](#_Toc494206493)

[Table 9: Enable CTC mode, from ATmega168 datasheet Table 15-4 (p.g. 133) 12](#_Toc494206494)

# Figures

[Figure 1 : LED circuit under forward bias conditions 5](#_Toc494206495)

[Figure 2 : LED circuit when MCU pin outputs high. 5](#_Toc494206496)

[Figure 3: Circuit Diagram 6](https://ourecuedu-my.sharepoint.com/personal/qstoutsp_our_ecu_edu_au/Documents/ENS2257%20-%20Microprocessor%20Systems/Assessments/ENS2257_MPUS%20-%20Group_Work/Assignment%201/ENS2257%20Assignment%201.docx#_Toc494206497)

[Figure 4 : Seven-segment display layout (ROHM, 2008) 7](#_Toc494206498)

# Equations

[Equation 1: Equation for determining LED circuit resistor. 4](#_Toc494206499)

[Equation 2: Shows minimum required bits for representation of Clock Speed. 11](#_Toc494206500)

[Equation 3: Shows justification of Timer/Counter1 & prescaler option. 11](#_Toc494206501)

[Equation 4: Shows the clock ticks required to count to one second, when prescaling at 1024. 12](#_Toc494206502)

[Equation 5: The maximum time delay with the solution given. 12](#_Toc494206503)

# Task 1

## Single Digit LED Numeric Display

**Model Number: LA-601AB**

The seven-segment display consists of an array of Light Emitting Diodes (LED)s, each sharing a common anode. In this case, the datasheet specifies the common anode pins (COM) as pins 3 and 8 on the display package. The common anode will be connected directly to the 5V DC supply and each cathode will be connected, via a current limiting resistor, to corresponding pins on the Atmel ATmega168 microprocessor. When implemented on the breadboard, the anode pins 3,8 and the decimal point pin 5 on will be connected to Vcc. The anode pins will be connected to avoid issues such as short circuits, whilst pin 5 will be connected so that there is no potential difference between the cathode and anode, effectively removing it from the circuit.

Table 1 : Absolute maximum ratings for LA-601AB (ROHM, 2008)

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Symbol | Value | Units |
| Forward Current |  | 25 | mA |
| Reverse Voltage |  | 5 | V |

**Forward Current:**

The absolute maximum amount of current that the LED can conduct from anode to cathode without risking damage. The forward current will increase exponentially as the forward voltage () difference from anode to cathode exceeds the specified in the LED data sheet. The forward current will be used to calculate the current limiting resistance required to keep the LED display within a stable operational range. Based on the graph in the datasheet of Relative Luminous Intensity versus forward current, for a luminous intensity of 1 (normal brightness), the display will need to operate with a forward current of 10mA.

**Reverse Voltage:**

The maximum voltage that can be applied to the LEDs, in reverse bias. Needs to be accounted for when the MCU output pins output a high signal. It is unlikely that the reverse bias voltage on the LED will reach 5V above the forward bias at any time during operation of the display. However, the current limiting resistors would also function to drop a portion of the voltage of a high output signal if the common anode was mistakenly connected to the negative breadboard rail.

Table 2 : Electrical Characteristics for LA-601AB (ROHM, 2008)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameter | Symbol | Conditions | Value | | Units |
| Typ. | Max |
| Forward Voltage |  | = 20mA | 2.05 | 2.6 | V |

**Forward Voltage:**

The forward voltage versus forward current graph in the datasheet shows that to produce a forward current of 10mA, the forward voltage difference between the LED anode and cathodes will be need to be 2V. The values given for the forward voltage in the data sheet specify that a typical forward voltage of 2.05V is given under the conditions the forward current is 20mA.

Equation 1: Equation for determining LED circuit resistor.

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Symbol | Description | Nominal value |
|  | Vs | Power supply voltage | 5V |
| Vf | Forward voltage of LED | 2V |
| I | Current | 10mA |
| R | Resistance | To be determined |

Thus, the required resistance value required for the current limiting resistors to lower the current flow the LED circuit to 10mA is 300Ω.

## Atmel ATmega168 Microcontroller

Table 3: Absolute maximum ratings for ATmega168 (Atmel, 2007)

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Nominal Value | Condition | Min/Max |
| Output Low Voltage | 0V | = 20mA | Max = 0.7V |
| Output High Voltage | 5V | = -20mA | Min = 4.2V |
| DC Current per I/O Pin | 40.0mA | - | - |
| DC Current per I/O Port | Low = 100mA | - | - |
| High = 150mA |

**Output Low Voltage:**

**The voltage on the I/O pins when the PORTx registers have a logical zero in the corresponding bits.**

**As the LED display is a common anode type, the segments of the display will have to be switched on by outputting an active low signal between 0V and 0.7V on the MCU pins.**

**Output High Voltage:**

**The voltage on a pin when a PORTx bit has a logical high value, with the I/O pins accordingly outputting a logical high value ranging from 4.2V to 5V. Writing to the PORTB/C registers and outputting a high signal will result in the LED segments on the display being turned off.**

**DC Current per I/O Pin:**

**The absolute maximum amount of current the MCU can sink or source per pin. Practically, this amount of current should never be reached on any of the I/O pins. In this, case the MCU is directly controlling the seven-segment display through the output pins, so the pins could be exposed to high currents if there were a short-circuit. The inclusion of current limiting resistors will help to reduce the current in the LED circuit and ensure that the maximum limits for the pins are not reached.**

**DC Current per I/O Port:**

**Whilst the typical input and output current for an I/O pin is 20mA at 5V, the datasheet specifies that the sum of all currents into a port for an output low signal should not exceed 100mA. The LEDs can easily operate at 20mA with a forward bias of 2.05V to 2.6V, therefore the pin assignment will have to be considered to prevent the I/O PORT limits from being exceeded. If the six available pins on PORTC all sink 20mA, then PORTC will be sinking 120mA of current, exceeding the maximum ratings for the MCU and potentially damaging the microcontroller. The datasheet also specifies that the sum of all output high currents should not exceed 150mA for the I/O ports used to control the LED display. As the seven-segment anode is connected to Vcc through the anode, the main current draw when segments are activated will be directly from the power supply rail.**

## Additional components

**300Ω Resistors x7**

As the LA-601AB LED display is common anode type, current limiting resistors will need to be placed in series with the cathodes. Cathode or Anode placement of the resistor wouldn’t affect the operation of individual LEDs, however an array of LEDs sharing a common anode would appear as if they were connected in parallel when a resistor is placed on the anode side. This could result in uneven current flow due to the variability in the forward conduction voltage range of LEDs, potentially exposing one or more of the LEDs on the display to forward currents in excess of the absolute maximum ratings. The alternative approach will be to connect a resistor between the individual cathode and MCU pins, this will result in each LED on the display being wired in series with the corresponding MCU pin and result in a stable current flow throughout the circuit. As all the LEDs are connected in series with the MCU pins, the seven connected LEDs can be represented the sum of seven individual LED circuits.

Pin output set to active low:

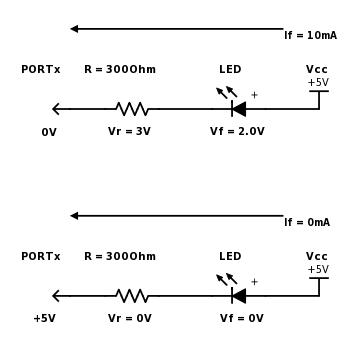


Figure 1 : LED circuit under forward bias conditions

When the pin output is active low, a current of approximately 10mA will flow from Vcc, through the circuit and to the output pins. Assuming all seven MCU pins are active low (all LEDs enabled), the total current the MCU will have to sink is 70mA, with PORTC sinking 40mA and PORTB sinking 30mA.

Pin output set high:

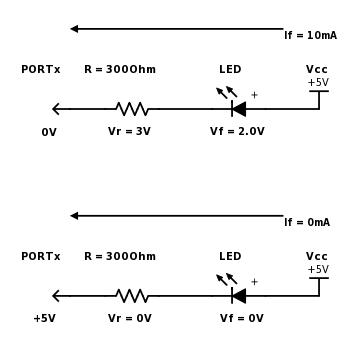


Figure 2 : LED circuit when MCU pin outputs high.

When the pin output is set to a logical high value, there will be no current flow through the LED circuit as there is no potential difference between the MCU pins and Vcc.

## Circuit Diagram

Figure 3: Circuit Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| Element | Description | Nominal value | Function |
| Vcc | Voltage Source | 5V | Provides source voltage for circuit to run. Actual source voltage is 9V, reduced to 5V with a voltage regulator. |
| MCU | ATmega168 | pin high = 5V  pin low = 0V | Used to control the LEDs of the seven segment display. |
| R1-R7 | Current limiting resistors | 300 **Ω** | Reduce current through LED circuit to 10mA |
| 7-SEG | Seven segment display | COM = 5V | COM3/8: Common anode |

## Seven segment display interface requirements

As in the circuit diagram above, the pins on the LA-601AB display will be connected through a 300-ohm current limiting resistor to the listed pins on the ATmega168 microcontroller in the table below.

Table 4: LED pin connection reference table

|  |  |  |  |
| --- | --- | --- | --- |
| Pin No. | Function | MCU Pin | Figure 4 : Seven-segment display layout (ROHM, 2008) |
| 1 | E | PC2 |
| 2 | D | PC1 |
| 3 | COM | - |
| 4 | C | PC0 |
| 5 | D.P | - |
| 6 | B | PB5 |
| 7 | A | PB4 |
| 8 | COM | - |
| 9 | F | PB3 |
| 10 | G | PC3 |

Based on the circuit setup for anode type display, outputting active low on MCU pins will light up the corresponding display segment. Therefore, the output pattern for the pins on the microcontroller will need to clear bits in the PORTB & PORTC registers to turn on the display and set bits to turn off segments. By writing data bits. Alternatively, the PIN(B/C) registers could be used to toggle the output pins.

Table 5: Truth table for the seven-segment display

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hex character | Binary | B (PB5) | A (PB4) | F (PB3) | G (PC3) | E (PC2) | D (PC1) | C (PC0) |
| 0 | 0000 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0001 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 2 | 0010 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 0011 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 4 | 0100 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 5 | 0101 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0110 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0111 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 8 | 1000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1001 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| A | 1010 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| B | 1011 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| C | 1100 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| D | 1101 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| E | 1110 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| F | 1111 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

## C Subroutine code (display 0 to F by parameter passed to subroutine)

//function declarations

void hex\_to\_disp**(**uint8\_t hex\_num**);**

void init\_disp**()**

//Lookup table for decoding the seven segment display.

static const uint8\_t seg\_table**[**15**]** **=**

**{**

//Outputting 0 on a pin activates the segment, a 1 turns off the segment.

//

// [# 0 0 0 0 0 0 0]

// ^ ^ ^ ^ ^ ^ ^ ^

// | b a f g e d c

// |

// Bit seven is unneeded or could be used for the d.p

//

// 0 1 2 3

0b00001000**,** 0b00111110**,** 0b00010001**,** 0b00010100**,**

// 4 5 6 7

0b00100110**,** 0b01000100**,** 0b10000000**,** 0b00011110**,**

// 8 9 A b

0b00000000**,** 0b00000100**,** 0b00000010**,** 0b01100000**,**

// c d E F

0b01110001**,** 0b00110000**,** 0b01000001**,** 0b01000011

**};**

// Initialise the display

void init\_disp**()**

**{**

//Set the 7 pins to output, DDRC|=[00001111]

DDRC **|=** **((**1**<<**PC3**)|(**1**<<**PC2**)|(**1**<<**PC1**)|(**1**<<**PC0**);**

//DDRB|=[00111000]

DDRB **|=** **(**1**<<**PB5**)|(**1**<<**PB4**)|(**1**<<**PB3**);**

//Initialize PORTB/C assuming 7seg display is connected to correct pins.

// Turn on all segments.

PORTC **&=** **~((**1**<<**PC3**)|(**1**<<**PC2**)|(**1**<<**PC1**)|(**1**<<**PC0**));**// PORTC&=[11110000]

PORTB **&=** **~((**1**<<**PB5**)|(**1**<<**PB4**)|(**1**<<**PB3**));**// PORTB&=[11000111]

**}**

// Pass in hexadecimal parameter and set PORTB/C bits.

void hex\_to\_disp**(**hex\_num**)**

**{**

uint8\_t seg\_code **=** seg\_table**[**hex\_num**]**

//Lowest four bits of seg\_code match with PORTC lower bits.

//Bitmask clears high bits to keep OR'd PORTC high bits unchanged

PORTC **|=** **(**seg\_code **&** **((**1**<<**PC3**)|(**1**<<**PC2**)|(**1**<<**PC1**)|(**1**<<**PC0**)))**//0s to 1s

//bitmask the upper four bits as ones to keep AND'd PORTC bits

PORTC **&=** **(**seg\_code **|** **~((**1**<<**PC3**)|(**1**<<**PC2**)|(**1**<<**PC1**)|(**1**<<**PC0**)));**//1s to 0s

//seg\_code bits need to be right shifted one place to match PORTB

//Then bitmask for the PORTB bits

PORTB **|=** **((**seg\_code**>>**1**)** **&** **((**1**<<**PB5**)|(**1**<<**PB4**)|(**1**<<**PB3**)));**//0s to 1s

PORTB **&=** **((**seg\_code**>>**1**)** **|** **~((**1**<<**PB5**)|(**1**<<**PB4**)|(**1**<<**PB3**)));**//1s to 0s

**}**

## Assembler Routine that writes 0 to the seven-segment display

#include "avr/io.h"

.global zero\_to\_disp

zero\_to\_disp:

cbi PORTC, 2 ;Clear bit 2 in PORTC - E

cbi PORTC, 1 ;Clear bit 1 in PORTC - D

cbi PORTC, 0 ;Clear bit 0 in PORTC - C

sbi PORTC, 3 ;Set bit 3 in PORTC - G

cbi PORTB, 5 ;Clear bit 5 in PORTB - B

cbi PORTB, 4 ;Clear bit 4 in PORTB - A

cbi PORTB, 3 ;Clear bit 3 in PORTB - F

ret

.end

# Task 2

## Description of principles of solution

The delay function will accept two 8-bit integers representing the desired delay in minutes and seconds and will then use the ATmega168 internal 16bit timer/counter along with the clock pre-scaler to count time with a resolution of one second. The timer/counter control register will be set to CTC mode so that the timer/counter will reset once it counts to a number specified in the output compare register. The output compare flag will be set upon completion of every timer/counter cycle and the timer compare interrupt will call an ISR every cycle that will decrement the specified minute and seconds integers. Once the seconds variable reaches zero, the minutes variable will be decremented by one and the seconds variable will be set to 60. When both the minutes and seconds variables reach zero, the output compare interrupt mask will be disabled and the program will resume from where the function was last called.

## Description of all timers/counters, registers and interrupts used

### PRR – Power Reduction Register

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| PRR | PRTWI | PRTIM2 | PRTIM0 | - | PRTIM1 | PRSPI | PRUSART0 | PRADC |

This register is used to reduce power consumption by disabling unneeded modules. In relation to the assignment, the Power Reduction Timer/Countern (PRTIMn) bits will disable the associated Timer/Counter when written logic one.

### TCCR1A/ TCCR1B – Timer/Counter1 Control Register A/B

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| TCCR1A | COM1A1 | COM1A0 | COM1B1 | COM1B0 | - | - | WGM11 | WGM10 |
| TCCR1B | ICNC1 | ICES1 | - | WGM13 | WGM12 | CS12 | CS11 | CS10 |

These registers are used for configuring various modes and settings of Timer/Counter1, including Output Compare, Clock Select and Waveform Generation modes.

The Clock Select bits are used to select between no clock source, external clock source, or system clock with various prescaling options.

The WGM bits are used to select between Waveform Generation modes. The *Clear Timer on Compare Match* (CTC) mode, is used in this solution. In CTC, the current clock count (TCNT1) is compared to the value of the OCR1A register and if they are equivalent, the counter is cleared. If the *Compare Match A* interrupt is enabled, the ISR will be called.

### OCR1A – Output Compare Register 1 A

This is a 16-bit register, in *Clear Timer on Compare Match* mode, it is used to generate an interrupt when equal to TCNT1.

### TIMSK1 – Timer/Counter1 Interrupt Mask Register

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| TIMSK1 | - | - | ICIE1 | - | - | OCIE1B | OCIE1A | TOIE1 |

This register is used to enable/disable interrupts specific to Timer/Counter1. When set, the OCIE1A bit enables interrupts to occur when OCR1A and TCNT1 are equal.

### TCNT1 – Timer/Counter1

This 16-bit register is the clock counter for Timer/Counter1. Writing TCNT1 directly can cause the OCR1A comparison condition to be missed, allowing the counter to run to overflow. As our solution does not require manually accessing TCNT1, it was not deemed necessary to implement an overflow interrupt.

### TIFR1 – Timer/Counter1 Interrupt Flag Register

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| TIFR1 | - | - | ICF1 | - | - | OCF1B | OCF1A | TOV1 |

This register contains interrupt flags associated with Timer/Counter1. When a positive comparison is made between OCR1A and TCNT1, the OCF1A flag is set and the Timer/Counter1 Compare Match A interrupt vector is run. After the ISR has completed, OCF1A is cleared.

Table 6 : Timer/Counter registers used (Atmel, 2007)

|  |  |
| --- | --- |
| Register Name | Description |
| PRR | Power Reduction Register |
| TCCR1A/B | Timer/Counter1 Control Register A/B |
| OCR1A | Output Compare Register 1 A |
| TIMSK1 | Timer/Counter1 Interrupt Mask Register |
| TCNT1 | Timer/Counter1 |
| TIFR1 | Timer/Counter1 Interrupt Flag Register |

## Details of configuration information (with justification/explanation/calculations)

### Register configuration

Table 7: Shows the configuration of all registers relevant to the solution.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Register | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| PRR | x | x | x | - | 0 | x | x | x |
| TCCR1A | 0 | 0 | 0 | 0 | - | - | 0 | 0 |
| TCCR1B | 0 | 0 | - | 0 | 1 | 1 | 0 | 1 |
| TIMSK1 | - | - | x | - | - | x | 1 | x |
| TIFR1 | - | - | x | - | - | x | 0 | x |

### Configuration justification

The external crystal oscillator installed on the NerdKit, resonates at 14.7456mHz. As shown below in *Equation 2*, the number of bits required to represent the minimum resolution of one second at a clock speed of 14.7456mHz, are not available in the OCRnA and TCNTn registers of either Timer/Countern. To solve this problem, we used prescaling to lower the counter resolution. As shown in *Equation 3*, even with the maximum prescaling option available (F/1024), the clock speed cannot be represented in the 8-bit register of Timer/Counter0. Due to these considerations, Timer/Counter1 was selected, with a prescaling of F/1024. To ensure that Timer/Counter1 is not disabled, logic zero is written to the PRTIM1 (bit 3) bit of the Power Reduction Register. To select the system clock with a prescaling of 1024, the bit configuration shown in *Table 8* must be made to Timer/Counter Control Register B (TCCR1A).

Equation 2: Shows minimum required bits for representation of Clock Speed.

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Symbol | Description | Value |
|  | F | Oscillator clock speed | 14.7456e6 |
| TC0 | The representation limit of Timer/Counter0 | 2^8 = 256 |
| TC1 | The representation limit of Timer/Counter1 | 2^16 = 65,536 |

Equation 3: Shows justification of Timer/Counter1 & prescaler option.

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Symbol | Description | Value |
|  | F | Oscillator clock speed | 14.7456e6 |
| F1024 | Oscillator clock speed, divided by 1024 | 14,400 |
| TC0 | The representation limit of Timer/Counter0 | 2^8 = 256 |
| TC1 | The representation limit of Timer/Counter1 | 2^16 = 65,536 |

Table 8: Clock Select configuration

|  |  |  |  |
| --- | --- | --- | --- |
| CS12 | CS11 | CS10 | Description |
| 1 | 0 | 1 | CLKI/O/1024 (From prescaler) |

The solution we opted for counts to one second (See *Equation 4*), then clears the TCNT1 counter register. To do this, we set OCR1A to equal the number of clock cycles in one second and we use the Clear Timer on Match (CTC) mode, so that TCNT1 is cleared after each second. Enabling CTC mode requires setting and clearing bits in both TCCR1A and TCCR1B. See *Table 9*.

Equation 4: Shows the clock ticks required to count to one second, when prescaling at 1024.

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Symbol | Description | Value |
|  | F | Oscillator clock speed | 14.7456e6 |
| F1024 | Oscillator clock speed, divided by 1024 | F/1024 = 14,400 |
| CPF/1024 | The Clock period of F1024 | 1/F1024=69.44µs |
| S | The number of cycles per second | |

Table 9: Enable CTC mode, from ATmega168 datasheet Table 15-4 (p.g. 133)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WGM13 | WGM12 | WGM11 | WGM10 | Description |
| 0 | 1 | 0 | 0 | Clear Timer on Comparison (CTC) |

As ‘Compare Output Mode’ was not required, the COM1A1 (bit 7), COM1A (bit 6), COM1B1 (bit 5) and COM1B0 (bit 4) bits, in the TCCR1A register, are cleared.

As ‘Input Capture’ is not required for this solution, the ICNC1 (bit 7) and ICES1 (bit 6) bits, in the TCCR1B register, are cleared.

The maximum delay time is shown in *Equation 5*.

Incrementing the minutes and seconds values in the solution is performed in an interrupt service routine. To enable the correct interrupt the OCIE1A bit in the TIMSK1 register is set, and the SEI function is called to enable global interrupts. To ensure that the interrupt flag is properly cleared before starting the clock, the OCF1A bit in the TIFR1 register, is cleared.

Equation 5: The maximum time delay with the solution given.

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Symbol | Description | Value |
|  | s | The maximum integer representation of the global unit8\_t seconds counter |  |
| S | The maximum amount of time countable in seconds. | 255 = 4min, 15sec |
| m | The maximum integer representation of the global unit8\_t minutes counter |  |
| M | The maximum amount of time countable in minutes. | 255 = 4hr, 15min |
| MD | Maximum delay time | |

## 

## Description of algorithm(s) (with code fragments or pseudocode)

### Global declarations

uint8\_t minutes;//globals to be used by ISA

uint8\_t seconds;

### Timer delay function

void timer\_delay(min, sec)

{

//Both minutes and seconds go up to 255 for a maximum delay of 4 hours, 19 minutes and 15 seconds.

minutes = min;

seconds = sec;

init\_timer();

}

### Initializing the Timer/Counter

void init\_timer();

{

//Set correct bits in timer control register, flags, interrupts, etc.

//Set the timer up so that the OCF1A flag interrupts every second

PRR &= ~(1<<PRTIM1); // Ensure PRTIM1 bit is cleared in the Power Reduction Register

TCCR1A ^= TCCR1A;//Clear TCCR1A register

//WGM12 - Set Timer/Counter Control Register to CTC mode

TCCR1B = (1<<WGM12)|(1<<CS12)|(1<<CS10);//CS12/CS10 - Set clock source to prescale clk/1024

OCR1A = 14400;//Count up to the desired time for the prescaled clock. i.e.(14400/14.7456M/1024)= 1 second

//Set interrupts last

TIMSK1 |= (1<<OCIE1A);//Enable timer counter interrupts for Output

TIFR1 &= ~((1<<OCF1A);

Compare Register 1 A

sei();//Set global interrupts enable.

}

### ISR routine

ISR(TIMER1\_COMPA\_vect)

{

//increment/decrement, whatever we want.

if(minutes!=0)

{

if(seconds==0)

{

seconds = 60;

minutes--;

}

seconds--;

}

// else, if no minutes left to count down

else(minutes==0)

{

// check if seconds are still counting down

if(seconds!=0)

{

seconds--;//decrement seconds

}

else //If no minutes or seconds to count down, disable timer counter

{

TIMSK1 &= ~(1<<OCIE1A);//zero the mask to disable the interrupt.

PRR |= (1<<PRTIM1);//lower power consumption by disabling timer/counter1 in PRR.

}

}

}

# References

Atmel. (2007). *8-bit AVR Microcontroller with 8k bytes In-System Programmable Flash.*

ROHM. (2008). Signle Digit LED Numeric Display. Kyoto, Japan. Retrieved from www.rohm.com/web/eu/products/-/product/LA-601AB