AVR242: 8-bit Microcontroller Multiplexing LED Drive and a 4 x 4 Keypad

Features

- **16 Key Pushbutton Pad in 4 x 4 Matrix**
- **4 Digit Multiplexed LED Display with Flashing Colon**
- **Industrial Real Time Clock/timer**
- **Controls ON/OFF Times for two Loads**
- **Tactile Feedback via Piezo-sounder**
- **Flashing Display to Indicate Power Down Event**
- **Dual Function I/O Pins**
- **Minimum External Components**
- **Efficient Code**
- **Complete Program Included for AT90S1200**
- **Suitable for any AVR MCU with 20 Pins or more**

Introduction

This application note describes a comprehensive system providing a 4 x 4 keypad as input into a real time clock/timer with two outputs. This system control external loads, and a four digit mulitplexed LED display. The application is designed to show the versatility of the AVR port configuration, and the efficiency of the rich instruction set. The application will run on any AVR with 20 pins or more, although due consideration will have to be given to stack initialization and table placement. The program has been structured within the confines of the three level deep hardware stack at the AT90S1200 and could be better structured in the other AVRs with software stack.

Theory of Operation

The connection of a 4 x 4 keypad, a piezo sounder, two LED loads and a four digit multiplexed display, would normally require twenty-three I/O lines. This application shows how this can be reduced to fifteen with a bit of ingenuity, allowing the smaller 20-pin AVR to be used. The circuit diagram is shown in figure 1 and is complete apart from the oscillator components, which have been omitted for clarity.

The four keypad columns are connected to the low nibble of port B and the four keypad rows are connected to the high nibble. The same eight bits also directly drive the segment cathodes of the four digit LED display, via current limit resistors R13-20. The pins thus serve a dual function, acting as outputs when driving the LED display and I/O when scanning the keypad. This is accomplished by using the programmable nature and large current drive capabilities of the AVR ports to good effect.

The majority of the time port B sinks the 9 mA of current, to directly drive the LED segments. Each digit is switched sequentially in 5 ms time slots, to multiplex the displays via the PNP transistors Q1-4. The common anodes of the LED display digits are driven via PNP transistors, since the maximum possible 72 mA (9mA - 8 segments) of current is outside the handling capabilities of the ports.

These can be any PNP type capable of driving 100 mA or so (e.g BC479). This could be modified by paralleling up two port pins for each anode to share the current, but then the number of I/O pins required would necessitate the use of a larger MCU.

Before the start of each display cycle, the port configuration is changed to provide four inputs with internal pull-ups enabled, and four outputs in the low state to scan the keypad. If a key is pressed the nibble configuration is trans-

Multiplexing LED Drive and a 4 x 4 Keypad

Application Note

Rev. 1231A–12/98

posed to calculate the key value with the key number stored in a variable. A short delay is allowed between each port change to allow the port to settle. This method is more code efficient than the conventional "snake" method in this application.

The common anode drives are disabled during this time to avoid interference. The port configuration is then reinstated ready for the multiplexing routine. The main housekeeping function then uses this key variable to take the appropriate action.

The real time clock is interrupt driven, using Timer 0 clocked from the system clock divided by 256. The timer is preloaded with the number 176 and interrupts on overflow every five milliseconds, ensuring high accuracy if a good quality crystal is used. To be accurate a 4.096 MHz clock crystal is employed. The program could be modified to use a 4 MHz crystal with minor modifications.

The interrupt service routine reloads the timer and increments three variables: a counter variable (tock), a keypad debounce variable (bounce) and a counter to maintain the seconds count (second). This is used by the main housekeeping function to update the minutes and hours, which in turn are displayed by the display function.

The housekeeping function checks the two loads for ON or OFF times and controls the outputs on the high nibble of port D accordingly. In this application the loads are simulated by red and green LEDs driven in current sink (active low) configuration. These could be replaced by relay drivers or opto-coupled triacs to drive power loads.

The keypad provides a means of setting up (SET) the real time and the ON/OFF times of each load and also allows the loads to be turned off (CLEAR) at once. A Piezo-sounder, connected to the top bit of port D, provides an audible beep on keypress.

The use of the port B pins requires some careful consideration. Since the pins are used for two functions, it is important that if a key is pressed, it does not short out the display. This is achieved by placing current limit resistors in series with each key. When used as inputs the internal pull-up resistors are employed saving external components. The choice of resistor value (R1-8) is such that the potential division is negligible. With the values chosen, and on a 5V supply, the logic levels are about 0.6V for logic "0" and 4.95V for logic "1". Resistors R21 and R22 are the traditional current limit resistors for the LEDs and can be any suitable value for the supply rail. This note was tested using 330 ohms on a 5V supply. The LEDs are driven in current sink mode ("0" = ON) and provide about 9 mA of forward current with the values specified.

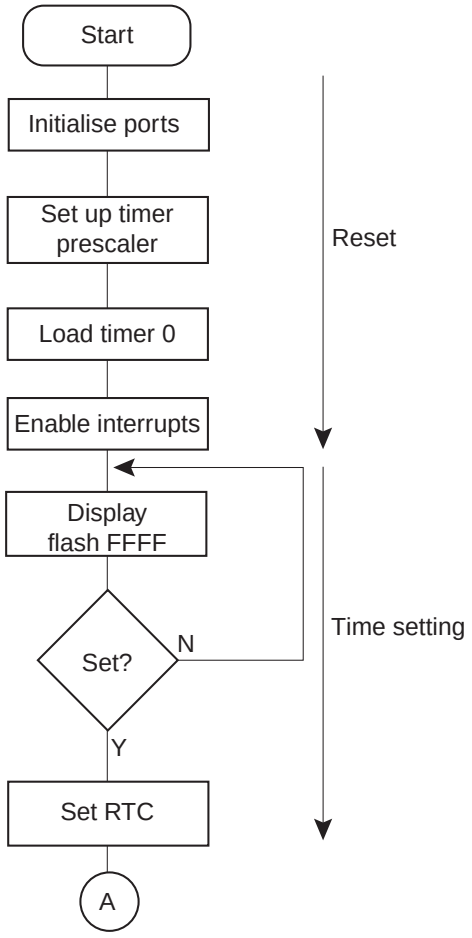
Implementation

The firmware comprises of two main areas, a background function, which is interrupt driven and provides the real-time accuracy, and the foreground processes. These consist of three sections, the reset routine, which sets up the ports, timer and the interrupts, the timesetting routine and the main housekeeping function.

Foreground Process

The foreground process is running for most of the time, only interrupted for 5.127 microseconds (21 cycles) every 5 ms to update the real time clock variables. It consists of three sections, RESET, TIME SETTING and HOUSEKEEPING. The flowchart is shown in Figure 1.

Figure 1. .Foreground process flow chart (Part 1) Continued on Figure 3.



Reset Section

On power up, or reset conditions, a reset routine is entered to initializes the system hardware. The ports are initialized

with their starting directions and all pins set high to turn off any loads. These are fixed as all outputs initially, requiring 255 to be loaded into the data direction registers of both ports. The directions are modified on port B for a short time by the keypad scanning function. The timer prescaler is set up to divide the clock by 256, giving a 5 ms interrupt period when the timer is loaded with 176. The timer overflow interrupt is then enabled followed by Global interrupts.

The equation for the interrupt period is tied to the 4.096 MHz clock, providing an instruction cycle time of 0.2441 microseconds. The number n to be loaded into the timer 0 register TCNT0 is thus given by :-

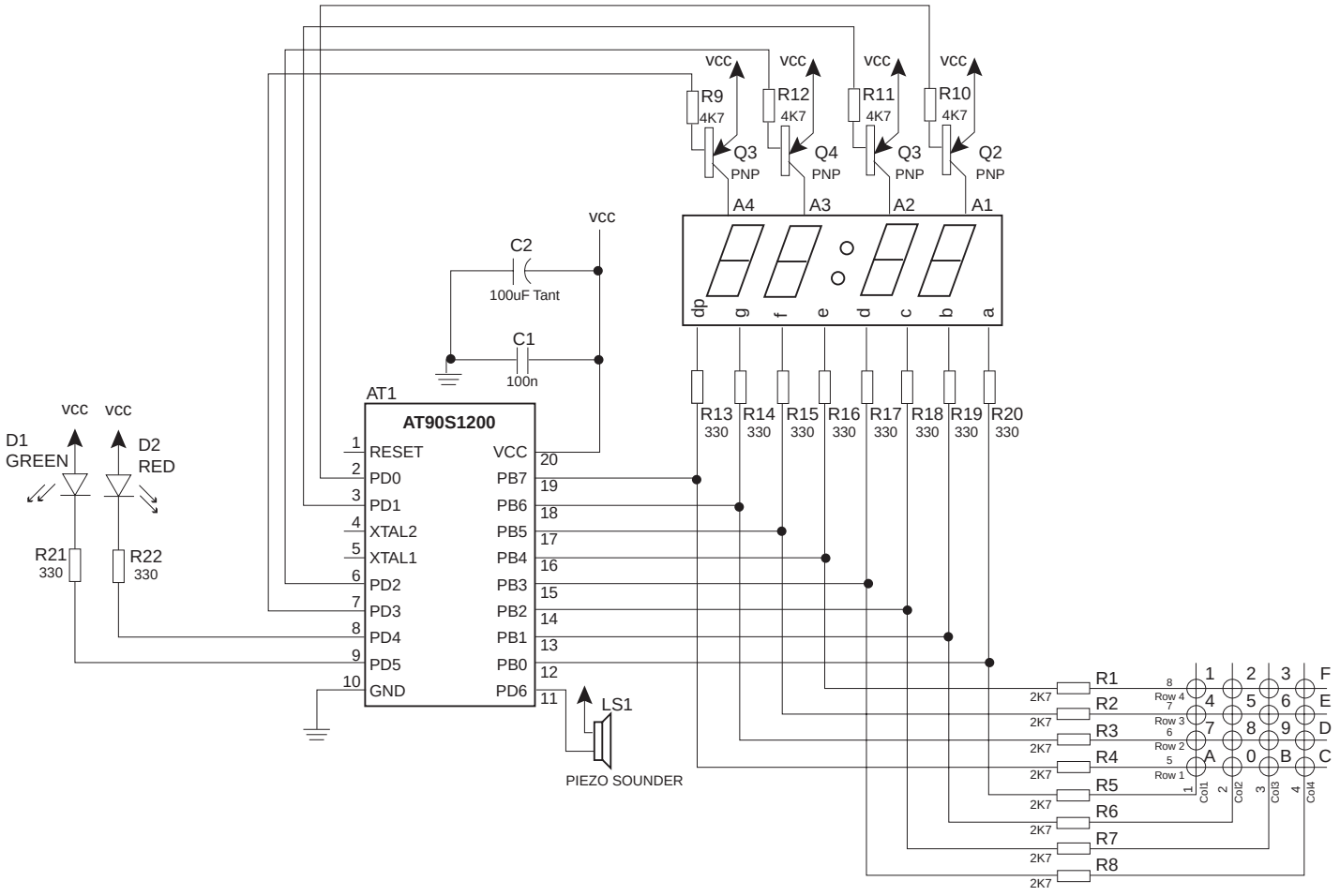
*(256 - n) * 256 * 0.2441 microseconds.*

A value of 176 provides 5 ms exactly , ensuring high RTC accuracy.

Time Setting

The LEDs are now made to flash EEEE to indicate that the time is incorrect and needs resetting. This will continue until the SET key is pressed on the key pad. This calls the "setrtc" function which handles input from the keypad and display feedback. Once the time has been reset, the main housekeeping function handles the updating and driving of the display from the main *"second"* variable, and scans the keypad for commands.

Figure 2. Circuit Diagram for Keypad/Display Unit



Housekeeping

The main housekeeping function does the work of updating the time variables derived from the background process and driving the LED display with the correct time. The key pad is also scanned to allow command inputs and the on/off times are checked for the loads. The flowchart is shown in Figure 3.

The seconds, incremented by the interrupt service routine, are compared with 60. If 60 seconds has passed the minute variable is incremented and the seconds reset to zero. The same procedure is adopted for the hours, with the minute variable compared to sixty and the hour variable incremented accordingly. The hour variable is then compared with twenty-four to check for the start of a new day and the hours and seconds all reset to zero.

To save on the use of RAM storage, the minutes and hours have been confined to one byte each. The low nibble houses the low digit and the high nibble the high digit. This means that it must be treated as BCD and the appropriate error trapping included to ensure correct counting. The minute or hour byte must therefore be split up into nibbles and checked for size on each check.

If no change is encountered during any of the checks on minutes or hours the next section is bypassed and the time is displayed. The clock is a twenty-four hour type and consequently must cause a start of new day when the time is incremented from 23:59. The display routine is a function called “display” which also includes the keyscan routine. This function is explained later.

On return from the display function the key value is checked, followed by the on/off times for the loads and any appropriate action taken before the housekeeping loop is repeated. E.g. If load 1 on time equals the RTC then load 1 is turned on.

A “flag” variable is used to contain single bits to indicate various actions. This is used to pass control from one function to another. For this application NINE flags were required, which is one more than that available in one byte. To save using another register just for one bit, the “T” flag in the status register has been employed for the ninth bit. This is useful because it can be tested using specific branch instructions (BRTC, BRTS) making programming easy, with the SBRS and SBRC instructions used for the main “flag” tests. The flags are active high and are allocated as shown in table 1 below, along with their function: The time taken around the loop does not affect the accuracy of the RTC since it is interrupt driven, with the loop being interrupted four times during one pass of the loop.

Figure 3. -Foreground process flow chart (part 2)

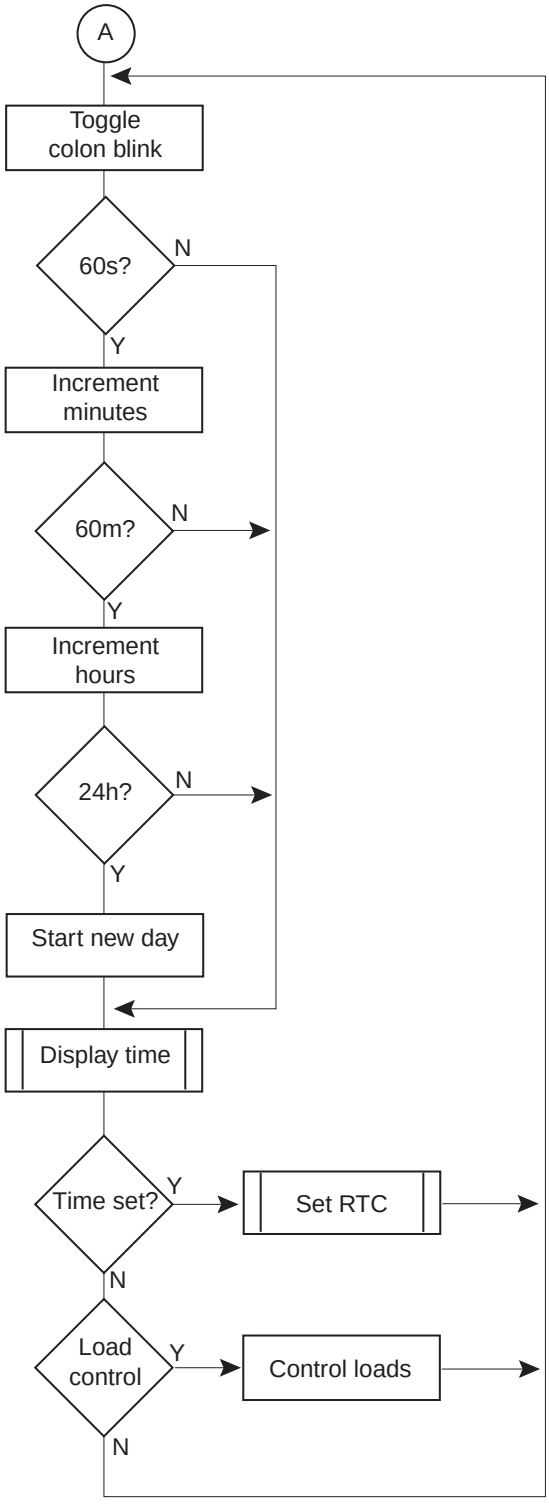


TABLE 1. Flag word usage

"FLAG" bit number	Function
0	Load 1 active
1	Load 2 active
2	Load 1 ON
3	Load 1 OFF
4	Load 2 ON
5	Load 2 OFF
6	Key press OK (debounced)
7	5 ms tick pulse
Status T flag	Time Set encountered

The central colon (dp) is flashed at half second intervals using the "blink" variable incremented by the background interrupt process. This is used to toggle the "flash" variable which is used as a mask by the display function.The load check routine is actually more complex than the single flowchart box would suggest, testing the various control bits in the "flag" word and taking action accordingly. Including this in the flowchart would have made it very difficult to follow.

If it picks up a "set load" command it calls up the "setrtc" function to load in a new on or off time for the load key selected. The same flashing method is employed here, only now the display flashes "n" in the appropriate digit being entered and moves across from high to low as the time is entered. The user is thus sure which number is going where.

A CLEAR command turns off both loads immediately cancelling any previous on/off commands.These processes do not affect the RTC, which still maintains the correct time in the background. The RTC can also be modified, to update the time, at any stage by the same process.

Display Function

The flowchart is shown in Figure 5. This function is called up by the flashing reset routine, the "setrtc" function and the housekeeping routine, and serves to scan the keypad and multiplex the display. If a larger AVR is to be employed it would be worth making the digit drive segments a function and calling it up four times. This can not be done with the AT90S1200, because of the 3 level deep stack.

The first section disables the display anode drives and then scans the keypad. This is done by changing the PORTB configuration to inputs on the row nibble and outputs on the column nibble. The internal pull-ups are also enabled on the four inputs. All four columns bits are taken low and the row inputs read from PINB. This generates either a base number, stored in "key" of 0, 4, 8, or 12 depending on the key row pressed, or the number 0x10 if no key is pressed.

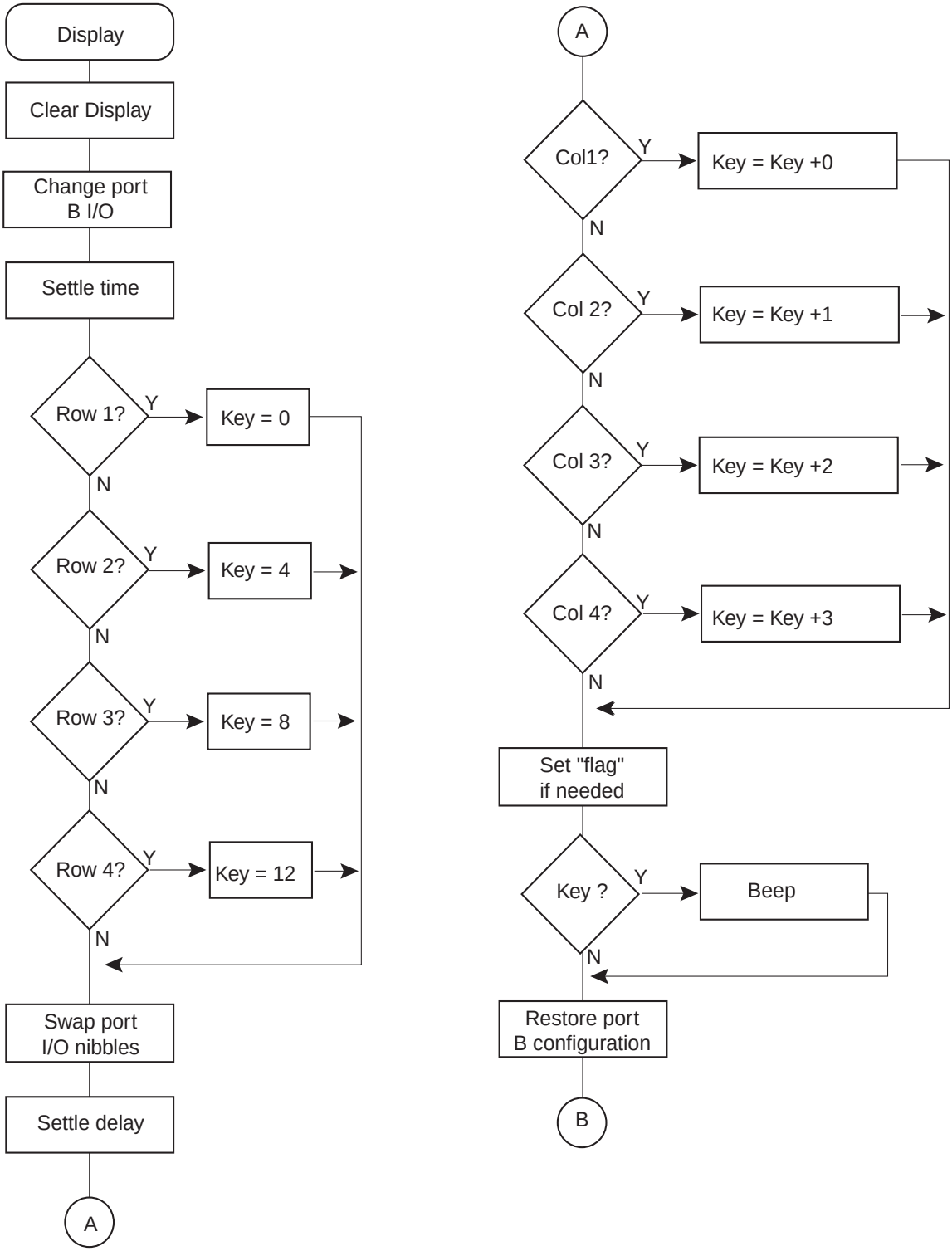
The port configuration is then swapped over to make the row nibble outputs and the column nibble inputs, and the row bits taken low. After a short settling time the column inputs are read from PINB and used to add a small offset of 0, 1, 2, or 3 to the base number depending on the key column pressed. The end result is a number stored in "key" which is used as an index to look up the actual key value required in a table stored in EEPROM. The true key value is written back into "key" and used by the calling functions. This is necessary because the keys are not arranged in a logical order. It also provides greater flexibility for the programmer. The keypad layout and functions are shown in Figure 4.

Figure 4. Keypad Layout and function

1 #1	2 #2	3 #3	F Load 1 ON
4 #4	5 #5	6 #6	E Load 1 OFF
7 #7	8 #8	9 #9	D Load 2 ON
A SetRTC	0 #0	B Clear	C Load 2 OFF

Key values greater than 9 are trapped and used to set the corresponding bits in the "flag" word used by the calling functions. A key value of 0x10 indicates that no key has been pressed.

Figure 5. Flowchart for keyscan part of “display” function



If a key has been pressed a short “beep” is sent to the piezo sounder connected to PORTD bit 6 for tactile feedback to the user.

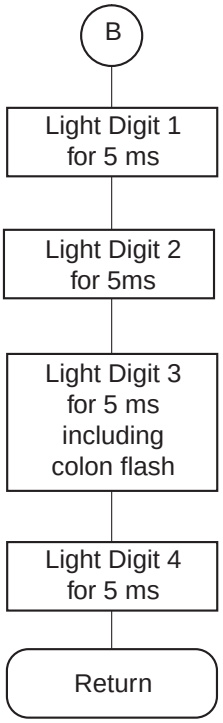
The digits are then multiplexed in turn in 5 ms time slots, timed by the 5 ms flag set by the background process. This gives about a 50 Hz display rate producing a bright, flicker free display (ignoring the short keyscan time).

Each digit drive uses a look-up table stored in EEPROM for the seven segment decoding, taking the index in via the “temp” register and using it to access the byte required to light up that character. Several special characters are used to make keypad input more meaningful. For instance the letter’ E’ is defined for the flashing error display on power up, the letters “o”, “n” and “f” are defined for the load setting ON/OFF inputs. If you are using a larger AVR for your application you may wish to transfer these tables to ROM and access them by indexed addressing.

The colon blinking section then checks for a half second event and changes the “flash” mask used in the previous display process, thus blinking the centre colon to indicate correct clock function.

The function then returns to the calling function with the key value stored in “key”.

Figure 6. .Flowchart for display part of “Display” function



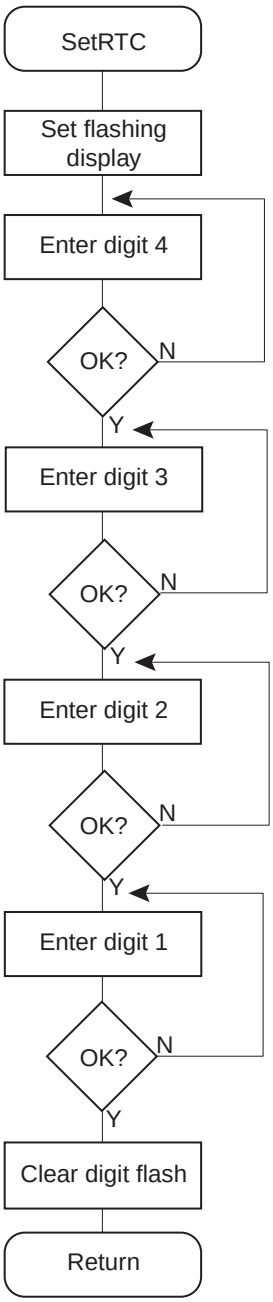
Setrtc Function

The flowchart is shown in Figure 7. This function is called up by all the routines which require keypad input to set up the display. This happens at power up/reset to enter the real time, on pressing the SET key to modify the real time, and on pressing any of the four load setting keys. It calls the display function to find the keypress and display the appropriate digits. It uses a “bounce” counter, incremented every 5 ms by the background interrupt function, to provide a reasonable keypress action.

The function proceeds in four phases, starting from the most significant digit and working to the least significant digit, displays a flashing “n” in each digit until a suitable value has been entered via the keypad. Values that are out of range are trapped and the input requested again until it is in range.

When all four digits have been input correctly the function exits with the hours in the variable “hiset” and the minutes in the varibable “loset”. These are redirected by the calling function into the appropriate variables for use by the house-keeping function.

Figure 7. Flow chart for “setrtc” function

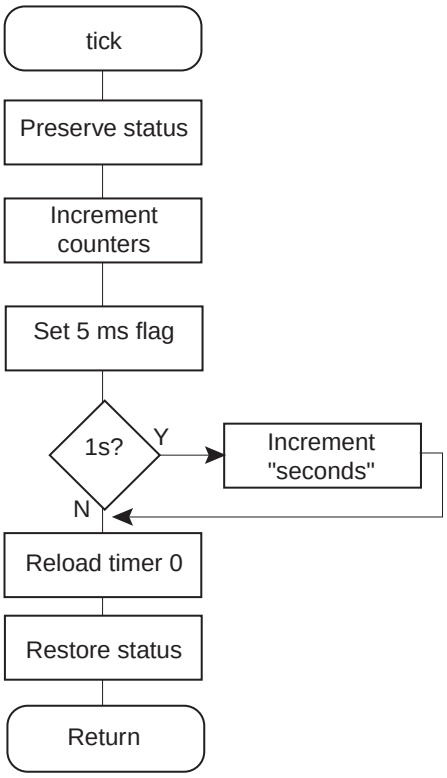


Background Function (tick)

This function is triggered every 5 ms by timer 0 overflow and interrupts the foreground function at any point in the loop. The routine consequently preserves the status register on entry and restores it on exit as a matter of course, to avoid disturbing the foreground processes. The use of the “temp” register is also avoided for the same reason.

The function is very straightforward and merely increments three counting registers on every entry, sets the 5 ms tick flag used by the display routine, reloads timer 0, and increments the RTC second counter if necessary. The flowchart is shown in Figure 8.

Figure 8. Flowchart for “tick” Background Function



Resources

Table 2. CPU and Memory Usage

Function	Code Size	Cycles	Register Usage	Interrupt	Description
Reset	17 words	17 cycles	R16, R31	-	Initiialization
Timesetting	9 words	14 cycles	R1, R2, R18, R19, R24, R25	-	Initial setting of RTC
Housekeeping	97 words	52 typical	R1, R2, R16, R17, R18, R19, R20, R21, R24, R25, R28	-	Main housekeeping loop to maintain real time display, respond to keypad and control loads.
Display	158 words	150 typical	R16, R17, R20, R21, R23, R24, R25, R26, R28	-	Keyscan and Display function
Setrtc	47 words	45 typical	R1, R2, R16, R20, R22, R24, R25, R26, R28	-	Function to handle keypad time and load setting input
tick	15 words	21 cycles	R0, R31	TIMER0	Background interrupt service routine to provide real time 5 ms and 1 s “tick”
TOTAL	343 words	-	R0, R1, R2, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R28, R31	TIMER0	

Table 3. Peripheral Usage

Perpheral	Description	Interrupts
Timer 0	5 ms tick counter	Timer 0 overflow with prescalar set to divide by 256
16 byte EEPROM	Key to value mapping Seven segment decoding	-
8 I/O pins PORT B	4 x 4 keypad connections and LED segment drive(dual function)	-
3 I/O pins PORT D	Load 1 and 2 and piezo-sounder	-
4 I/O pins PORT D	Anoder drive for four digit LED display	-

```
;**** A P P L I C A T I O N   N O T E   A V R 242 ************************
;*
;* Title:        Multiplexing LED drive and 4x4 keypad sampling
;* Version:      1.0
;* Last Updated: 98.07.24
;* Target:       All AVR Devices
;*
;* Support E-mail:avr@atmel.com
;*
;* DESCRIPTION
;* This Application note covers a program to provide a 24 hr Industrial
;* timer or real-time clock using I/O pins for dual functions.
;* With input via a 4 x 4 matrix keypad, output to a multiplexed
;* four digit LED display and two ON/OFF outputs to drive loads via additional
;* interface circuitry.  LED loads are driven in this example but it could drive
;* any load with the addition of suitable components. Tactile feedback is provided
;* on every key press by a piezo sounder which beeps when a key is pressed.
;* Included is a main program that allows clock setting via the keypad
;* and one ON/OFF time setting per 24 hours for each load, functions for the
;* real time clock, key scanning, and adjustment routines. The example runs on
;* the AT90S1200 to demonstrate how limited I/O  can be overcome, but can
;* be any AVR with suitable changes in vectors, EEPROM and stack pointer.
;* The timing assumes a 4.096 MHz crystal is employed (a 4 MHz crystal produces
;* an error of -0.16% if 178 instead of 176 used in the timer load sequence, but this
;* could be adjusted in software at regular intervals). Look up tables are
;* used in EEPROM to decode the display data, with additional characters provided
;* for time and ON/OFF setting displays and a key pad conversion table.
;* If the EEPROM is needed for your application the tables could be moved
;* to ROM in the larger AVR devices.
;***************************************************************************

;***** Registers used by all programs
;******Global variables used by routines

.def    loset      =r1                          ;storage for timeset minutes
.def    hiset      =r2                          ;storage for timeset hours
.def    ld1minon   =r3                          ;storage for load on and off times
.def    ld1hron    =r4                          ;set from keypad entry
.def    ld1minoff  =r5                          ;and tested in the housekeeping function
.def    ld1hroff   =r6                          ;and stores on or off times for the loads
.def    ld2minon   =r7
.def    ld2hron    =r8
.def    ld2minoff  =r9
.def    ld2hroff   =r10
.def    temp       =r16                         ;general scratch space
.def    second     =r17                         ;storage for RTC second count
.def    minute     =r18                         ;storage for RTC minute count
.def    hour       =r19                         ;storage for RTC hour count
.def    mask       =r20                         ;flash mask for  digits flashing
```

```
.def    blink      =r21                         ;colon blink rate counter
.def    bounce     =r22                         ;keypad debounce counter
.def    flash      =r23                         ;flash delay counter
.def    lobyte     =r24                         ;storage for display function minutes digits
.def    hibyte     =r25                         ;storage for display function hours digits
.def    key        =r26                         ;key number from scan

;***'key' values returned by 'keyscan'**************************
;VALUE  0 1 2   3   4 5 6   7     8 9 10  11    12  13  14   15  16
;KEY    1 2 3   F   5 6 E   7     8 9 D   A     0   B    C   NONE
;FUNC   1 2 3 LD1ON 4 5 6 LD1OFF  7 8 9 LD2ON SET  0 CLEAR LD2OFF

.deftock=r27                                    ;5 ms pulse
.defflags=r28                                   ;flag byte for keypad command keys
                                                ;  7     6     5    4      3      2     1    0
                                                ;  5ms  keyok ld2off ld2on ld1off ld1on ld2 ld1
                                                ; tick      0 = off, 1 = on
.equ    ms5     =7                              ;ticks at 5 ms intervals for display time
.equ    keyok   =6                              ;sets when key is debounced, must be cleared again
.equ    ld2off  =5                              ;set by load ON/OFF key press and flags
.equ    ld2on   =4                              ;up the need for action
.equ    ld1off  =3                              ;in the housekeeping routine
.equ    ld1on   =2
.equ    ld2     =1                              ;when set tells the housekeeping routine to
.equ    ld1     =0                              ;check load on/off times.

;***the T flag in the status register is used as a SET flag for time set
.equ    clear   =0                              ;RTC modification demand flag


;Port B pins

.equ    col1    =0                              ;LED a segment/keypad col 1
.equ    col2    =1                              ;LED b segment/keypad col 2
.equ    col3    =2                              ;LED c segment/keypad col 3
.equ    col4    =3                              ;LED d segment/keypad col 4
.equ    row1    =4                              ;LED e segment/keypad row 1
.equ    row2    =5                              ;LED f segment/keypad row 2
.equ    row3    =6                              ;LED g segment/keypad row 3
.equ    row4    =7                              ;LED decimal point/keypad row 4

;Port D pins
.equ    A1      =0                              ;common anode drives (active low)
.equ    A2      =1                              ;
.equ    A3      =2                              ;
.equ    A4      =3                              ;
.equ    LOAD1   =4                              ;Load 1 output (active low)
.equ    LOAD2   =5                              ;Load 2 output (active low)
.equ    PZ      =6                              ;Piezo sounder output (active low)
.include "1200def.inc"
```

```
;***** Registers used by timer overflow interrupt service routine

.def   timer  =r31                              ;scratch space for timer loading
.def   status =r0                               ;low register to preserve status register

;*****Look up table for LED display decoding **********************
.eseg                                           ;EEPROM segment
.org 0

table1:
  .db   0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
  ;digit 0    1    2    3    4    5    6    7    8    9

  .db   0x86,0x8E,0xA3,0xAB,0XFF,0XFF
  ;digit E     f   o    n       BLANK       special characters

;****Look up table for key value conversion into useful numbers****

  ;key1  2  3  F  4  5  6  E  7  8  9   D    A  0   B   C
table2:
  .db    1, 2, 3,15, 4, 5, 6,14, 7, 8, 9,  13, 10, 0, 11, 12
  ;value 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15

;****Source code***************************************************
.cseg                                           ;CODE segment
.org 0
    rjmp   reset                                ;Reset handler
    nop                                         ;unused ext. interrupt
    rjmp   tick                                 ;timer counter overflow (5 ms)
    nop                                         ;unused analogue interrupt

;*** Reset handler ************************************************
;*** to provide initial port, timer and interrupt setting up

reset:
    ser    temp                                 ;
    out    DDRB,temp                            ;initialize port B as all Outputs
    out    DDRD,temp                            ;initialize port D as all Outputs
    out    PORTB,temp                           ;key columns all high/LEDs off
    out    PORTD,temp                           ;turn off LEDs and loads off
    ldi    temp,0x04                            ;timer prescalar /256
    out    TCCR0,temp
    ldi    timer,176                            ;load timer for 5 ms
    out    TCNT0,timer                          ;(256 - n)*256*0.2441 us
    ldi    temp,0x02                            ;enable timer interrupts
    out    TIMSK,temp
    clr    flags                                ;clear control flags
    clr    tock                                 ;clear 5 ms tick
    clr    bounce                               ;clear key bounce counter
    clr    flash
```

```
    clr     blink
    sei                                     ;enable global interrupts

;****Flash EEEE on LEDS as test and power down warning**************
;****repeats until SET key is pressed on keypad

timesetting:
    ldi     hibyte,0xaa                     ;show "EEEE" on LED
    ldi     lobyte,0xaa                     ;display and
    ser     mask                            ;set flashing display
notyet:
    rcall   display                         ;display until time set
    brtc    notyet                          ;repeat until SET key pressed
    rcall   setrtc                          ;and reset time
    mov     hour,hiset                      ;and reload hours
    mov     minute,loset                    ;and minutes
    clt                                     ;clear T flag

;*****Main clock house keeping loop*****************************

do:
    clr     mask                            ;do housekeeping
    cpi     blink,100                       ;is half second up
    brne    nohalf
    clr     blink
    com     flash                           ;invert flash
nohalf:
    cpi     second,60                       ;is one minute up?
    brne    nochange                        ;no
    clr     second                          ;yes clear seconds and
    inc     minute                          ;add one to minutes
    mov     temp,minute
    andi    temp,0x0f                       ;mask high minute
    cpi     temp,10                         ;is it ten minutes?
    brne    nochange                        ;no
    andi    minute,0xf0                     ;clear low minutes
    ldi     temp,0x10
    add     minute,temp                     ;increment high minutes
    cpi     minute,0x60                     ;is it 60 minutes?
    brne    nochange                        ;no
    clr     minute                          ;yes, clear minutes and
    inc     hour                            ;add one to hours
    mov     temp,hour
    andi    temp,0x0f                       ;mask high hour
    cpi     temp,10                         ;is 10 hours up?
    brne    nochange                        ;no
    andi    hour,0xf0                       ;yes, increment
    ldi     temp,0x10
    add     hour,temp                       ;high hours
```

```
nochange:
    cpi     hour,0x24                       ;is it 24 hours?
    brne    sameday                         ;no,
    clr     hour                            ;yes, clear time variables
    clr     minute                          ;to start new day
    clr     second
sameday:                                    ;update times
    mov     lobyte,minute
    mov     hibyte,hour
    rcall   display                         ;show time for 20 ms
    brtc    case1                           ;if not SET
    rcall   setrtc                          ;and reset time
    mov     hour,hiset                      ;and reload hours
    mov     minute,loset                    ;and minutes
    clt                                     ;else, clear T flag
case1:sbrc flags,ld1                        ;is load 1 active?
    rjmp chkload1                           ;yes, check load 1
case2:sbrc flags,ld2                        ;is load 2 active
    rjmp chkload2                           ;yes, check load 2
case3:
    sbrc    flags,ld1on                     ;is load 1 on time reset
    rjmp    setld1on                        ;yes reset on time
case4:
    sbrc    flags,ld1off                    ;is load 1 off time reset
    rjmp    setld1off                       ;yes reset off time
case5:
    sbrc    flags,ld2on                     ;is load 2 on time reset
    rjmp    setld2on                        ;yes reset on time
case6:
    sbrc    flags,ld2off                    ;is load 2 on time reset
    rjmp    setld2off                       ;yes reset on time
case7:
    rjmp    do                              ;repeat housekeeping loop

;****case routines to service load times and key presses********

chkload1:
    cp      hour,ld1hroff                   ;is load 1 off time reached?
    brne    onload1
    cp      minute,ld1minoff
    brne    onload1
    sbi     PORTD,LOAD1                     ;yes, turn load 1 off
onload1:
    cp      hour,ld1hron                    ;is load 1 on time reached?
    brne    case2
    cp      minute,ld1minon
    brne    case2
    cbi     PORTD,LOAD1                     ;yes,turn load 1 on
    rjmp    case2                           ;repeat with load on
```

```
chkload2:
    cp      hour,ld2hroff                   ;is load 2 off time reached?
    brne    onload2
    cp      minute,ld2minoff
    brne    onload2
    sbi     PORTD,LOAD2                     ;yes, turn load 2 off
onload2:
    cp      hour,ld2hron                    ;is load 2 on time reached?
    brne    case3
    cp      minute,ld2minon
    brne    case3
    cbi     PORTD,LOAD2                     ;yes,turn load 2 on
    rjmp    case3                           ;repeat with load on
setld1on:
    sbr     flags,0x01                      ;make load 1 active
    rcall   setrtc                          ;pickup new on time
    mov     ld1hron,hiset                   ;and store
    mov     ld1minon,loset
    cbr     flags,0x04                      ;clear ld1on flag
    rjmp    case4

setld1off:
    rcall   setrtc                          ;pickup new off time
    mov     ld1hroff,hiset                  ;and store
    mov     ld1minoff,loset
    cbr     flags,0x08                      ;clear ld1off flag
    rjmp    case5
setld2on:
    sbr     flags,0x02                      ;make load 2 active
    rcall   setrtc                          ;pickup new on time
    mov     ld2hron,hiset                   ;and store
    mov     ld2minon,loset
    cbr     flags,0x10                      ;clear ld2on flag
    rjmp    case6
setld2off:
    rcall   setrtc                          ;pickup new on time
    mov     ld2hroff,hiset                  ;and store
    mov     ld2minoff,loset
    cbr     flags,0x20                      ;clear ld2off flag
    rjmp    case7

;****Multiplexing routine to display time and scan keypad every*****
;****second pass,used by all routines taking digits from hibyte
;****and lobyte locations with each digit on for 5 ms

display:
    ser     temp                            ;clear display
    out     PORTB,temp
```

```
;****Keypad scanning routine to update key flags*******************

keyscan:
    cbr     flags,0x40                      ;clear keyok flag
    ldi     key,0x10                        ;set no key pressed value
    ser     temp                            ;set keypad port high prior to
    out     PORTB,temp                      ;reinitializing the port
    in      temp,PORTD                      ;turn off LEDs and leave loads
    ori     temp,0x0f                       ;untouched prior to
    out     PORTD,temp                      ;key scan
    ldi     temp,0x0f                       ;set columns output and
    out     DDRB,temp                       ;rows input with pull-ups
    ldi     temp,0xf0                       ;enabled and all columns
    out     PORTB,temp                      ;low ready for scan
    ldi     temp,20                         ;short settling time
tagain1:
    dec     temp
    brne    tagain1
    sbis    PINB,ROW1                       ;find row of keypress
    ldi     key,0                           ;and set ROW pointer
    sbis    PINB,ROW2
    ldi     key,4
    sbis    PINB,ROW3
    ldi     key,8
    sbis    PINB,ROW4
    ldi     key,12
    ldi     temp,0xF0                       ;change port B I/O to
    out     DDRB,temp                       ;find column press
    ldi     temp,0x0F                       ;enable pull ups and
    out     PORTB,temp                      ;write 0s to rows
    ldi     temp,20                         ;short settling time
tagain2:
    dec     temp
    brne    tagain2                         ;allow time for port to settle
    clr     temp
    sbis    PINB,COL1                       ;find column of keypress
    ldi     temp,0                          ;and set COL pointer
    sbis    PINB,COL2
    ldi     temp,1
    sbis    PINB,COL3
    ldi   temp,2
    sbis    PINB,COL4
    ldi     temp,3
    add     key,temp                        ;merge ROW and COL for pointer
    cpi     key,0x10                        ;if no key pressed
    breq    nokey                           ;escape routine, else
    ldi     temp,0x10
    add     key,temp                        ;change to table 2
    out     EEAR,key                        ;send address to EEPROM (0 - 15)
    sbi     EECR,EERE                       ;strobe EEPROM
```

```
    in      key,EEDR                        ;read decoded number for true key
convert:
    cpi     key,10                          ;is it SET key ?
    brne    notset                          ;no check next key
    set                                     ;yes set T flag in status register
notset:
    cpi     key,11                          ;is key CLEAR?
    brne    notclear                        ;no, check next key
    sbi     PORTD,LOAD1                     ;yes, shut down all loads
    sbi     PORTD,LOAD2
    cbr     flags,0x03                      ;deactivate both loads
notclear:
    cpi     key,15                          ;is key LD1ON?
    brne    notld1on                        ;no, check next key
    sbr     flags,0x04                      ;yes, set LD1ON flag
notld1on:
    cpi     key,14                          ;is key LD1OFF?
    brne    notld1off                       ;no, check next key
    sbr     flags,0x08                      ;yes, set LD1OFF flag
notld1off:
    cpi     key,13                          ;is key LD2ON?
    brne    notld2on                        ;no, check next key
    sbr     flags,0x10                      ;yes, set LD2ON flag
notld2on:
    cpi     key,12                          ;is key LD2OFF?
    brne    notld2off                       ;no, check next key
    sbr     flags,0x20                      ;yes, set LD2OFF flag
notld2off:

;***Tactile feedback note generation routine*****************
;***provides a 4 kHz TONE to the piezo sounder for 5 ms*****

tactile:
    cbr     flags,0x80
    cbi     PORTD,PZ                        ;turn on piezo
    ldi     temp,125                        ;for a short time
t1again:
    dec     temp
    brne    t1again
    sbi     PORTD,PZ                        ;turn on piezo
    ldi     temp,125                        ;for a short time
t2again:
    dec     temp
    brne    t2again
    sbrs    flags,ms5                       ;repeat for 5ms
    rjmp    tactile
notok:
    cpi     bounce,40
    brlo    nokey
    sbr     flags,0x40                      ;set bounce flag
```

```
nokey:
    ser     temp
    out     DDRB,temp                       ;reinitialize port B as all Outputs
    out     PORTB,temp                      ;and clear LEDs

;***Display routine to multiplex all four LED digits****************

    cbi     PORTD,A1                        ;turn digit 1 on
    mov     temp,lobyte                     ;find low minute
digit1:
    cbr     flags,0x80                      ;clear 5 ms tick flag
    andi    temp,0x0f                       ;mask high nibble of digit
    out     EEAR,temp                       ;send address to EEPROM (0 - 15)
    sbi     EECR,EERE                       ;strobe EEPROM
    in      temp,EEDR                       ;read decoded number
    sbrs    flash,clear                     ;flash every 1/2 second
    or      temp,mask                       ;flash digit if needed
    out     PORTB,temp                      ;write to LED for 5 ms
led1:
    sbrs    flags,ms5                       ;5 ms finished?
    rjmp    led1                            ;no, check again
    sbi     PORTD,A1                        ;turn digit 1 off
    ser     temp                            ;clear display
    out     PORTB,temp
    cbi     PORTD,A2;
    mov     temp,lobyte                     ;find high minute
    swap    temp
digit2:
    cbr     flags,0x80                      ;clear 5 ms tick flag
    andi    temp,0x0f                       ;mask high nibble of digit
    out     EEAR,temp                       ;send address to EEPROM (0 - 15)
    sbi     EECR,EERE                       ;strobe EEPROM
    in      temp,EEDR                       ;read decoded number
    sbrs    flash,clear                     ;flash every 1/2 second
    or      temp,mask                       ;flash digit if needed
    out     PORTB,temp                      ;write to LED for 5 ms
led2:
    sbrs    flags,ms5                       ;5 ms finished?
    rjmp    led2                            ;no, check again
    sbi     PORTD,A2                        ;
    ser     temp                            ;clear display
    out     PORTB,temp
    cbi     PORTD,A3                        ;
    mov     temp,hibyte
digit3:
    cbr     flags,0x80                      ;clear 5 ms tick flag
    andi    temp,0x0f                       ;mask high nibble of digit
    out     EEAR,temp                       ;send address to EEPROM (0 - 15)
    sbi     EECR,EERE                       ;strobe EEPROM
    in      temp,EEDR                       ;read decoded number
```

```
    sbrs    second,clear                    ;flash colon
    andi    temp,0x7f
    sbrs    flash,clear                     ;flash every 1/2 second
    or      temp,mask                       ;flash digit if needed
    out     PORTB,temp                      ;write to LED for 5 ms
led3:
    sbrs    flags,ms5                       ;5 ms finished?
    rjmp    led3                            ;no, check again
    sbi     PORTD,A3
    ser     temp                            ;clear display
    out     PORTB,temp
    cbi     PORTD,A4;
    mov     temp,hibyte
    swap    temp
    andi    temp,0x0f                       ;is hi hour zero?
    brne    digit4
    ldi     temp,0xff                       ;yes,blank hi hour
digit4:
    cbr     flags,0x80                      ;clear 5 ms tick flag
    andi    temp,0x0f                       ;mask high nibble of digit
    out     EEAR,temp                       ;send address to EEPROM (0 - 15)
    sbi     EECR,EERE                       ;strobe EEPROM
    in      temp,EEDR                       ;read decoded number
    sbrs    flash,clear                     ;flash every 1/2 second
    or      temp,mask                       ;flash digit if needed
    out     PORTB,temp                      ;write to LED for 5 ms
led4:
    sbrs    flags,ms5                       ;5 ms finished?
    rjmp    led4                            ;no, check again
    sbi     PORTD,A4
    ser     temp                            ;clear display
    out     PORTB,temp
    tst     mask                            ;is flash complete?
    breq    outled                          ;yes, exit
    cpi     blink,50                        ;is blink time done?
    brlo    outled                          ;no, exit
    clr     blink                           ;yes, clear blink rate counter
    com     flash                           ;and invert flash byte
outled:
    ret

;****Function to Set RTC/on-off hours and minutes from keypad
;****returns with minutes in 'loset' and hours in'hiset'

setrtc:
    ser     mask                            ;set flashing display
    ldi     hibyte,0xdf                     ;place 'n' in hi hour
    ser     lobyte                          ;and blank in lo hr & minutes
hihrus:
    clr     bounce
```

```
bounce1:
    rcall   display                         ;display and check keypad
    sbrs    flags,keyok
    rjmp    bounce1
    cbr     flags,0x40                      ;clear keyok flag
    cpi     key,0x03                        ;is high hour > 2
    brsh    hihrus                          ;yes, read key again
hihrok:                                     ;no, valid entry
    swap    key                             ;move hihour to hi nibble
    mov     hiset,key                       ;and store in hours
    ldi     hibyte,0x0d                     ;place 'n' in lo hour
    add     hibyte,hiset                    ;merge hihour and 'n'
lohrus:
    clr     bounce
bounce2:
    rcall   display                         ;display and check keypad
    sbrs    flags,keyok                     ;is key stable?
    rjmp    bounce2                         ;no try again
    cbr     flags,0x40                      ;yes, clear keyok flag
    mov     temp,hibyte                     ;check that total hours
    andi    temp,0xf0                       ;are not > 24
    add     temp,key
    cpi     temp,0x24                       ;is hour>24?
    brsh    lohrus                          ;yes, read key again
    add     hiset,key                       ;no, merge hi and lo hours
lohrok:
    mov     hibyte,hiset                    ;display hours as set
    ldi     lobyte,0xdf                     ;place 'n' in hi minutes
himinus:
    clr     bounce
bounce3:
    rcall   display                         ;display and check keypad
    sbrs    flags,keyok
    rjmp    bounce3
    cbr     flags,0x40                      ;clear keyok flag
    cpi     key,6                           ;is hi minutes >5
    brsh    himinus                         ;no, read key again
lominok:
    swap    key                             ;move himin to hi nibble
    mov     loset,key                       ;and store in minutes
    ldi     lobyte,0x0d                     ;place 'n' in lo minutes
    add     lobyte,loset                    ;merge with hi minute
lominus:
    clr     bounce
bounce4:
    rcall   display                         ;display and check keypad
    sbrs    flags,keyok
    rjmp    bounce4
    cbr     flags,0x40                      ;clear keyok flag
    cpi     key,10                          ;is key >9
    brsh    lominus                         ;no, read key again
```

```
    add     loset,key                       ;yes, merge hi and lo minutes
    clr     mask                            ;clear digits flash
    ret                                     ;and return with time set

;****Timer Overflow Interrupt service routine*****************************
;****Updates 5 ms, flash and debounce counter to provide RTC time reference

tick:
    in      status,SREG                     ;preserve status register
    inc     tock                            ;add one to 5 ms 'tock' counter
    inc     blink                           ;and blink rate counter
    inc     bounce                          ;and bounce rate delay
    sbr     flags,0x80                      ;set 5 ms flag for display time
    cpi     tock,200                        ;is one second up?
    breq    onesec                          ;yes, add one to seconds
    nop                                     ;balance interrupt time
    rjmp    nosecond                        ;no, escape
onesec:
    inc     second                          ;add one to seconds
    clr     tock                            ;clear 5 ms counter
nosecond:
    ldi     timer,176                       ;reload timer
    out     TCNT0,timer
    out     SREG,status                     ;restore status register
    reti                                    ;return to main
```