AVR064: A Temperature Monitoring System with LCD Output

8-bit AVR® Microcontrollers

Application Note

Features

- **Presenting data on an LCD-display**
- **Temperature measurement**
- **Real Time Clock (RTC)**
- **UART communication with a PC**
- **PWM implementation**

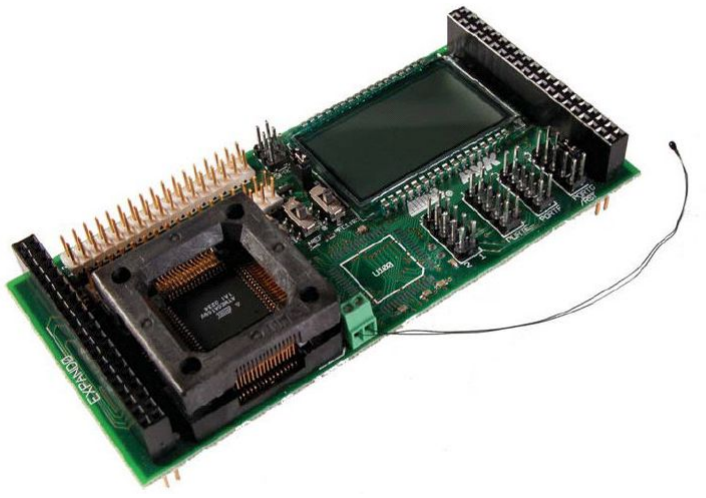
1 Introduction

The STK502 board is a top module designed to add ATmega169(P) support to the STK500 development board from Atmel. STK500 and STK502 provide all hardware needed to get started developing with the ATmega169(P). This application note is meant to be an example of how to use the ATmega169(P) and the STK502.

It includes:

- A complete ATmega169(P) code example written in C-code.
- Flowcharts explaining the code.
- Instruction on how to configure the STK502.
- A pre-programmed ATmega169(P) including the example in this application note is shipped with each STK502 kit.
- The source code is found on the "AVR Technical Library" CD shipped with the STK502 or on the Atmel AVR web site, http://www.atmel.com/products/avr/.

Figure 1-1. The STK502 board

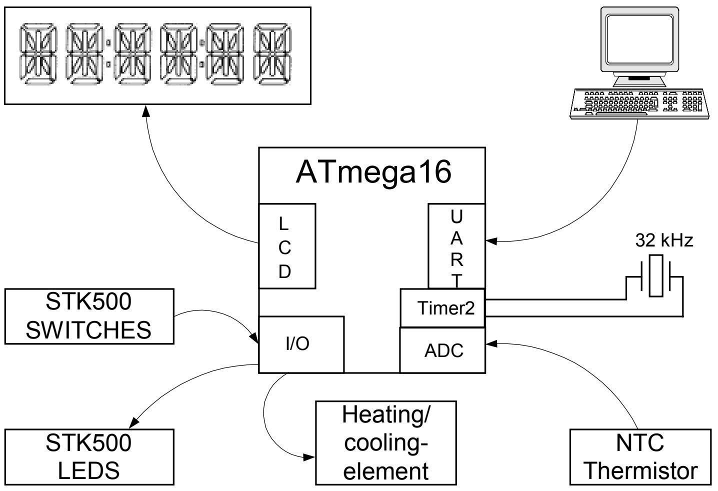


Rev. 2529C–AVR–02/06

2 Application overview

This application note describes how to get started with the ATmega169(P) microcontroller (MCU), the first AVR that has a built in LCD-controller/driver. This application is a temperature control application, including a Real Time Clock (RTC). It will monitor the temperature through a sensor, and regulate the temperature if a heating/cooling element is attached.

Figure 2-1. Application overview



The LCD starts with scrolling the text: "STK502 example application for ATmega169(P)". It is required that the example code is programmed into the ATmega169(P) and the hardware is set up according to the section "Hardware Configuration" on page 6.

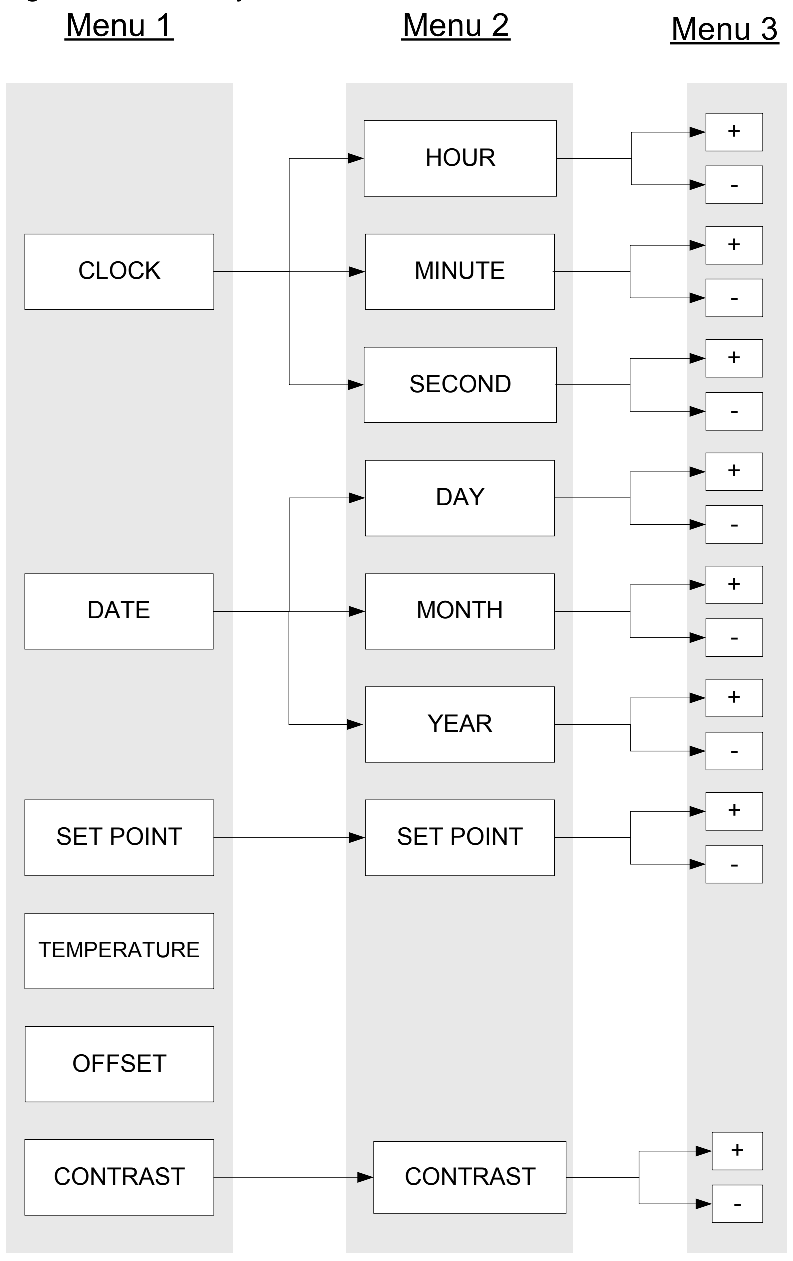
Select a desired temperature set point. When the temperature goes below this set point value, the Heater I/O pin will go high, and a LED on STK500 will flash. When the temperature goes above the set point value, the Cooler I/O pin will go high, and another LED on the STK500 will flash. The duty cycle of the LED flashing will vary with the actual temperature deviation from the set point (the greater the deviation is, the brighter the LEDs will shine) The LCD will display time and temperature information. All data that is presented on the LCD will also be sent through the UART-interface and can be received by etc a standard terminal.

Pressing a button on the STK500 will shift between the different information on the LCD. This information is:

- CLOCK: RTC clock running on the ATmega169(P)
- DATE: Calculated from the RTC-clock
- SET POINT: Selected temperature
- TEMPERATURE: Measured temperature
- OFFSET: Difference between the measured temperature and the set point
- CONTRAST: Shows all the segments available with the default hardware strapping.

Adjusting the CLOCK, DATE, SET POINT or the CONTRAST can be done by using three of the SWITCHES on the STK500. Since these switches are used for different functions, there is a need for a menu system. See Figure 2-2 for an overview of how the menus are arranged in this application.

Figure 2-2. Menu System



Please see section "STK500 switches" on page 17, for more detailed information on how to use the menu system.

The CLOCK, DATE and SET POINT can also be adjusted from the UART interface. See section "Terminal" on page 20.

The implementation is designed to be used with the STK502 and the LCD-display that is included in this starter-kit. For technical specifications and the LCD-bit mapping please refer to the "STK502 User Guide" found in the AVR Studio help system, and for more information on the LCD-driver see Application note "AVR065 - LCD driver for the STK502 LCD".

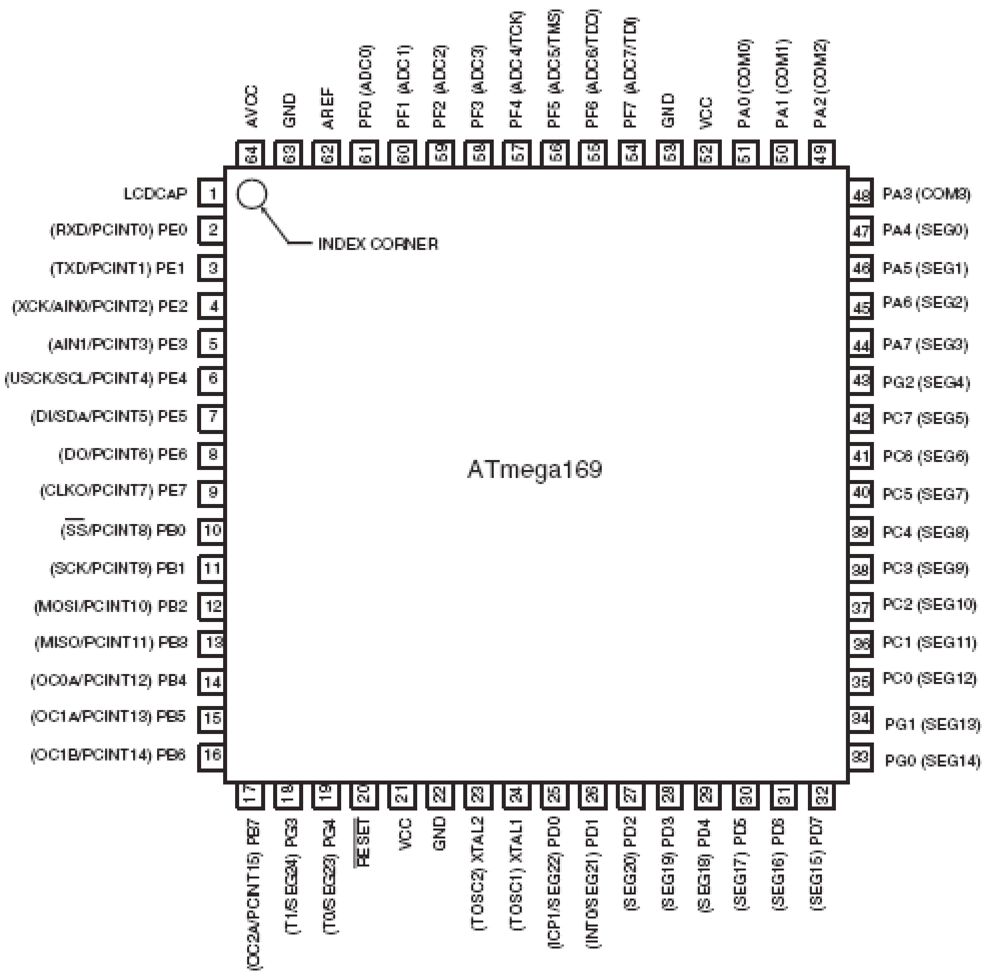
3 Hardware description

3.1 ATmega169(P)

The ATmega169(P) is an ultra low power AVR 8-bit RISC microcontroller. It includes 16K of bytes self-programming Flash Program Memory, 1K bytes SRAM, 512 Byte EEPROM and 8 Channel 10-bit A/D-converter, JTAG interface for on-chip-debugging and 4 X 25 Segment LCD Driver. It can do up to 1 MIPS throughput at 1 MHz for ATmega169(P)V, or 4 MIPS throughput at 4 MHz for the ATmega169(P)L.

The ATmega169(P) is an excellent choice for low power applications that requires user interaction (LCD + keyboard) and the possibility to interface analog sensors etc.

Figure 3-1. ATmega169



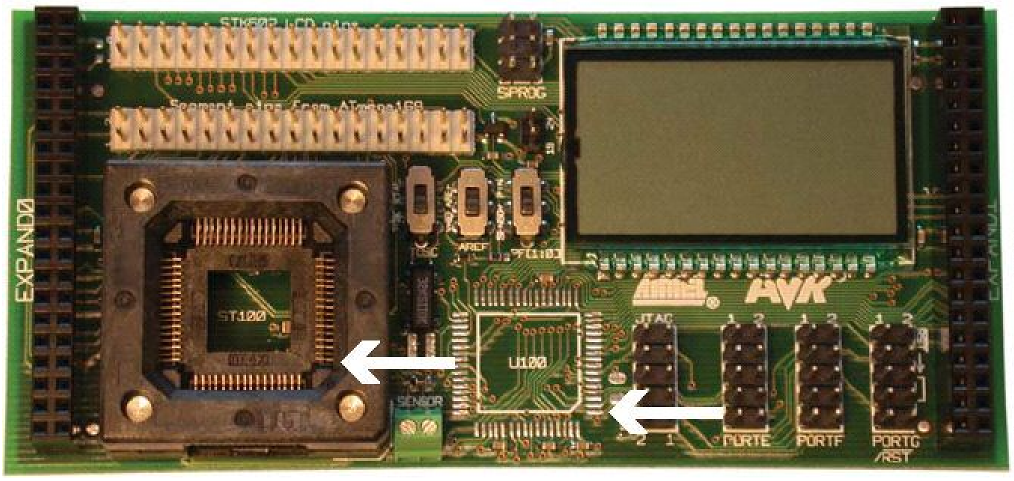
The ATmega169P has lower power consumption in power-down and power-save than ATmega169 revision E, but there are also other differences, e.g. oscillator. Please see application note AVR098 and the ATmega169(P) datasheets for more information.

3.2 STK502

The STK502 board is a top module designed to add ATmega169(P) support to the STK500 development board from Atmel.

The STK502 includes connectors and hardware allowing full utilization of the new features of the ATmega169(P) (including an LCD-display), while the Zero Insertion Force (ZIF) socket allows easy use of TQFP packages for prototyping.

Figure 3-2. STK502 top module for STK500 (pin1 location marked)



See the STK502 User Guide for more information about the STK502

3.2.1 LCD-display

Liquid Crystal Displays (LCDs) are categorized as non-emissive display devices. In that respect, they do not produce any form of light like a Cathode Ray Tube (CRT). LCDs are composed of a polarized liquid crystalline material in between two plates of glass. Typically, one plate is called the common or backplane, and the other is called a segment or frontplane. In a reflective LCD panel (one that has no back light) a voltage difference applied across the two electrodes will result in a polarization that will prevent the light from reflecting back to the observer. This will appear as a dark segment and is, therefore, considered ON. A lack of voltage difference will allow the light to reflect back and is considered OFF.

For more information on the LCD-driver, see application note “AVR065: LCD driver for the STK502 LCD”

3.2.2 NTC-thermistor

Various types of sensors can be used to measure temperature. One of these is the thermistor, or temperature-sensitive resistor. Most thermistors have a negative temperature coefficient (NTC), meaning the resistance goes up as temperature goes down. Of all passive temperature measurement sensors, thermistors have the highest sensitivity (resistance change per degree of temperature change). Thermistors do not have a linear temperature/resistance curve.

The NTC-thermistor used with this application has a resistance of 10kΩ at 25°C ( ), beta-value of 3450 and a tolerance of ±1%. The voltage over the NTC can be found using the A/D converter in the ATmega169(P). See the ATmega169(P) datasheets for how to use the ADC. And by the use of Equation 3-1, the temperature can be calculated.

Equation 3-1. Calculation of temperature from measured ADC values

$$Temperature = \frac{\beta}{\ln\left(\frac{V_{ADC}}{V_{ref} - V_{ADC}}\right) + \frac{\beta}{T_{amb}}} - T_{zero}$$

$$\beta = 3450$$

$$V_{ADC} = \text{Voltage calculated from the A/D conversion}$$

$$V_{ref} = 1.263\text{V}$$

$$T_{zero} = 273°\text{K}$$

$$T_{amb} = 298°\text{K}\ (273°\text{C} + 25°\text{C})$$

3.3 Hardware configuration

In order to make the example code work, it is required to set up the cables and switches in the correct order. Figure 3-3 and Figure 3-4 shows how to set up the cables and switches.

Figure 3-3. Cable settings

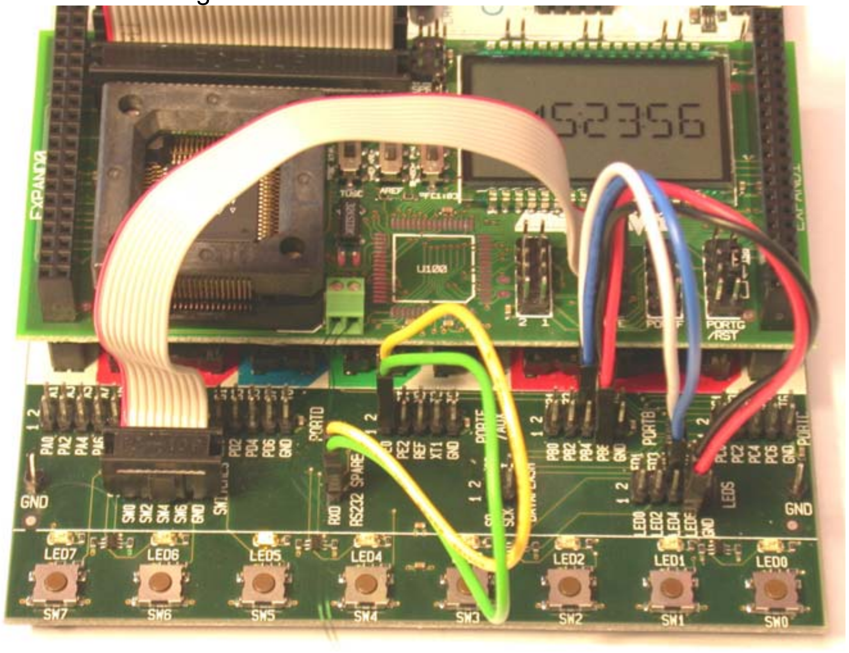
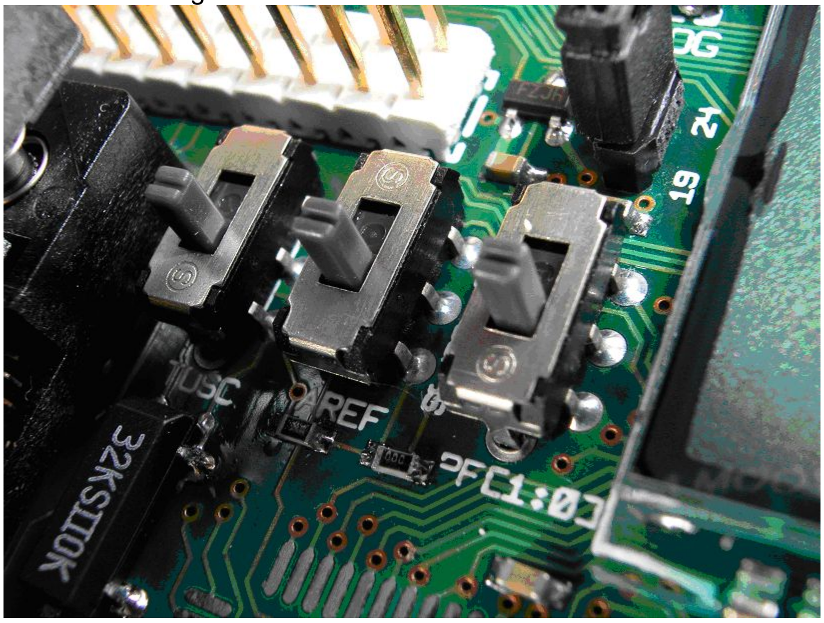


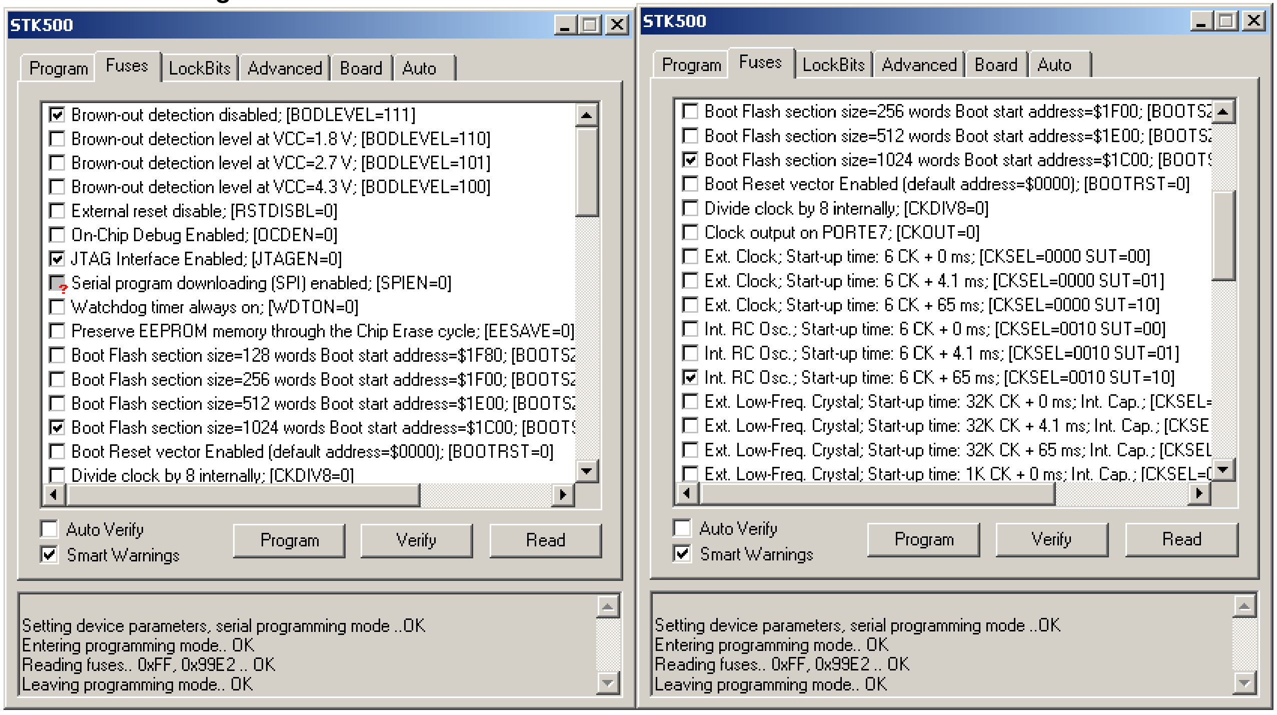
Figure 3-4. Switch configuration



- Connect PORTE on the STK502 to the SWITCHES-header on the STK500 with a 10-pin cable.
- Connect PB5/PB6 to LED5/LED6, PB4/PB7 to respectively Heating/Cooling element. If no heating/cooling element is available, just connect PORTB to the LEDS using a 10pins cable.
- Connect PE0/PE1 on the STK500 to the RXD/TXD.
- Connect the "Segment pins from ATmega169(P)" to the "STK502 LCD pins" with the 34pins cable.
- Place a jumper on the 2pins header "19 24"
- Insert the NTC-thermistor in the screw-terminal.
- All of the three switches on the STK502 should be in the position towards the screw-terminal, i.e. the TOSC switch should be in the TOSC-position, the AREF switch should be in the VREF-position and the PF[1:0] should be in the SENSOR-position.
- Connect PG5 and RST with a jumper, on PORTG/RST.

And most importantly insert the ATmega169(P) in the ZIF-socket. The ATmega169(P) that comes with the STK502 kit, is pre-programmed with the example code. If it is required to reprogram the ATmega169(P), see the STK502 User Guide for help on this topic. The `AVR064.hex` file that should be programmed into the ATmega169(P) can be found on the "AVR Software and Technical Library"-CD that comes with the STK502, and on the ATMEL web site (http://www.atmel.com/products/avr/). If the ATmega169(P) is reprogrammed, make sure the fuses are set up according to Figure 3-5.

Figure 3-5. Fuse settings


STK500

Program | Fuses | LockBits | Advanced | Board | Auto

- ☑ Brown-out detection disabled; [BODLEVEL=111]
- ☐ Brown-out detection level at VCC=1.8 V; [BODLEVEL=110]
- ☐ Brown-out detection level at VCC=2.7 V; [BODLEVEL=101]
- ☐ Brown-out detection level at VCC=4.3 V; [BODLEVEL=100]
- ☐ External reset disable; [RSTDISBL=0]
- ☐ On-Chip Debug Enabled; [OCDEN=0]
- ☑ JTAG Interface Enabled; [JTAGEN=0]
- ☑ Serial program downloading (SPI) enabled; [SPIEN=0]
- ☐ Watchdog timer always on; [WDTON=0]
- ☐ Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=0]
- ☐ Boot Flash section size=128 words Boot start address=$1F80; [BOOTSZ
- ☐ Boot Flash section size=256 words Boot start address=$1F00; [BOOTSZ
- ☐ Boot Flash section size=512 words Boot start address=$1E00; [BOOTSZ
- ☑ Boot Flash section size=1024 words Boot start address=$1C00; [BOOTS
- ☐ Boot Reset vector Enabled (default address=$0000); [BOOTRST=0]
- ☐ Divide clock by 8 internally; [CKDIV8=0]

☐ Auto Verify
☑ Smart Warnings

Program | Verify | Read

```
Setting device parameters, serial programming mode ..OK
Entering programming mode.. OK
Reading fuses.. 0xFF, 0x99E2 .. OK
Leaving programming mode.. OK
```

STK500

Program | Fuses | LockBits | Advanced | Board | Auto

- ☐ Boot Flash section size=256 words Boot start address=$1F00; [BOOTSZ
- ☐ Boot Flash section size=512 words Boot start address=$1E00; [BOOTSZ
- ☑ Boot Flash section size=1024 words Boot start address=$1C00; [BOOTS
- ☐ Boot Reset vector Enabled (default address=$0000); [BOOTRST=0]
- ☐ Divide clock by 8 internally; [CKDIV8=0]
- ☐ Clock output on PORTE7; [CKOUT=0]
- ☐ Ext. Clock; Start-up time: 6 CK + 0 ms; [CKSEL=0000 SUT=00]
- ☐ Ext. Clock; Start-up time: 6 CK + 4.1 ms; [CKSEL=0000 SUT=01]
- ☐ Ext. Clock; Start-up time: 6 CK + 65 ms; [CKSEL=0000 SUT=10]
- ☐ Int. RC Osc.; Start-up time: 6 CK + 0 ms; [CKSEL=0010 SUT=00]
- ☐ Int. RC Osc.; Start-up time: 6 CK + 4.1 ms; [CKSEL=0010 SUT=01]
- ☑ Int. RC Osc.; Start-up time: 6 CK + 65 ms; [CKSEL=0010 SUT=10]
- ☐ Ext. Low-Freq. Crystal; Start-up time: 32K CK + 0 ms; Int. Cap.; [CKSEL=
- ☐ Ext. Low-Freq. Crystal; Start-up time: 32K CK + 4.1 ms; Int. Cap.; [CKSE
- ☐ Ext. Low-Freq. Crystal; Start-up time: 32K CK + 65 ms; Int. Cap.; [CKSEL
- ☐ Ext. Low-Freq. Crystal; Start-up time: 1K CK + 0 ms; Int. Cap.; [CKSEL=

☐ Auto Verify
☑ Smart Warnings

Program | Verify | Read

```
Setting device parameters, serial programming mode ..OK
Entering programming mode.. OK
Reading fuses.. 0xFF, 0x99E2 .. OK
Leaving programming mode.. OK
```

As Figure 3-5 describes, the only fuses that should be programmed is:

- Brown-out detection disabled
- JTAG Interface Enabled
- Boot Flash section size = 1024 words
- Int. RC Osc.; Start-up time 6CK + 65 ms

4 ATmega169(P) firmware

This section contains information about the source code and functions. The firmware can be downloaded from the Atmel website: http://www.atmel.com/products/AVR/. For compiler info and settings, device settings, target setup info and comprehensive source documentation please see the readme.html file included with the source.

The timing related functions are written for an ATmega169(P) running at 1Mhz (except the RTC-clock and the LCD-frame rate that is clocked from an external 32kHz crystal), and the prescaler for the system clock is therefore set to 1/8. Please note that the ATmega169 revision B runs at 4 MHz, so the communication speed of the UART will be reduced from 9,600 to 4,800 baud, but otherwise will be ok.

4.1 Interrupts used

4.1.1 LCD Start of Frame

In this interrupt the data from the LCD_displayData buffer is latched to the LCD Data Registers. The variable LCD_Blink toggles every time this interrupt occurs. The interrupt is dependent of the external 32kHz crystal.

4.1.2 Timer/Counter2 Overflow

This interrupt is used to increment the variable SECOND, which the whole RTC-clock builds on. Timer/Counter2 is clocked asynchronous from the 32kHz and is therefore independent of the clock frequency.

4.1.3 USART0, RX complete

This interrupt takes care of incoming data from the UART interface.

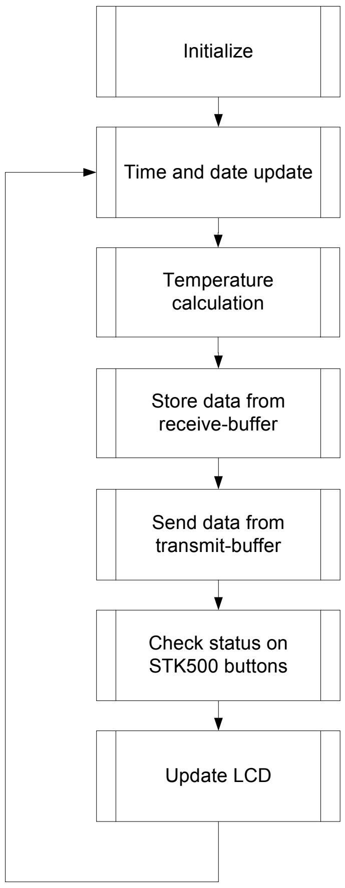
4.1.4 USART0, Data Register Empty

This interrupt transmits data out through the UART interface.

4.2 Main loop

Figure 4-1 shows the main loop.

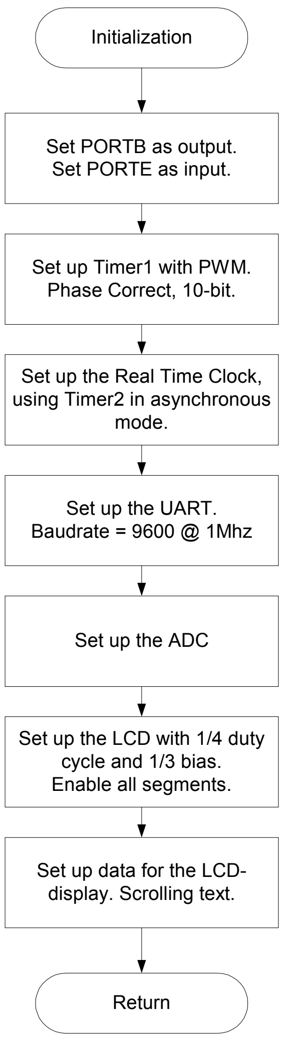
Figure 4-1. Main loop


Initialize
Time and date update
Temperature calculation
Store data from receive-buffer
Send data from transmit-buffer
Check status on STK500 buttons
Update LCD

4.3 Initialize

After a reset the firmware will initialize the ATmega169(P) and its integrated peripherals. The initialization runs only one time after a reset.

Figure 4-2. Initialize


Initialization
Set PORTB as output. Set PORTE as input.
Set up Timer1 with PWM. Phase Correct, 10-bit.
Set up the Real Time Clock, using Timer2 in asynchronous mode.
Set up the UART. Baudrate = 9600 @ 1Mhz
Set up the ADC
Set up the LCD with 1/4 duty cycle and 1/3 bias. Enable all segments.
Set up data for the LCD-display. Scrolling text.
Return

PORTB is set as output and should be connected to the LEDS on STK500. PB5 (OC1A) and PB6 (OC1B) show the offset between measured temperature and selected temperature set point. PB4 and PB7 are heating and cooling pins respectively. Connect a heating and cooling element to these pins.

DDRE is set as input and should be connected to the SWITCHES on the STK500. PE7, PE6 and PE5 are used to select what information should be displayed on the LCD and adjusting time/date, temperature set point and the LCD contrast.

Timer/Counter1 is set up with PWM to use on the OC1A/OC1B (PB5/PB6) pins.

Enable Timer/Counter2 with asynchronous operation, for the RTC. By using an external 32kHz crystal the RTC can run independently of the ATmega169(P) system clock, and will also run during sleep.

Set up the UART with both RX and TX enable, baud rate 9600 @ 1 MHz, asynchronous operation, 8 bit character size, 1 stop bit and disable parity mode.

Set up the ADC in single-ended mode. (Differential mode can be selected by setting ADC_init(Differential) instead of ADC_init(SingleEnded) in the source code). Disable digital input on PORTF and run a dummy ADC conversion.

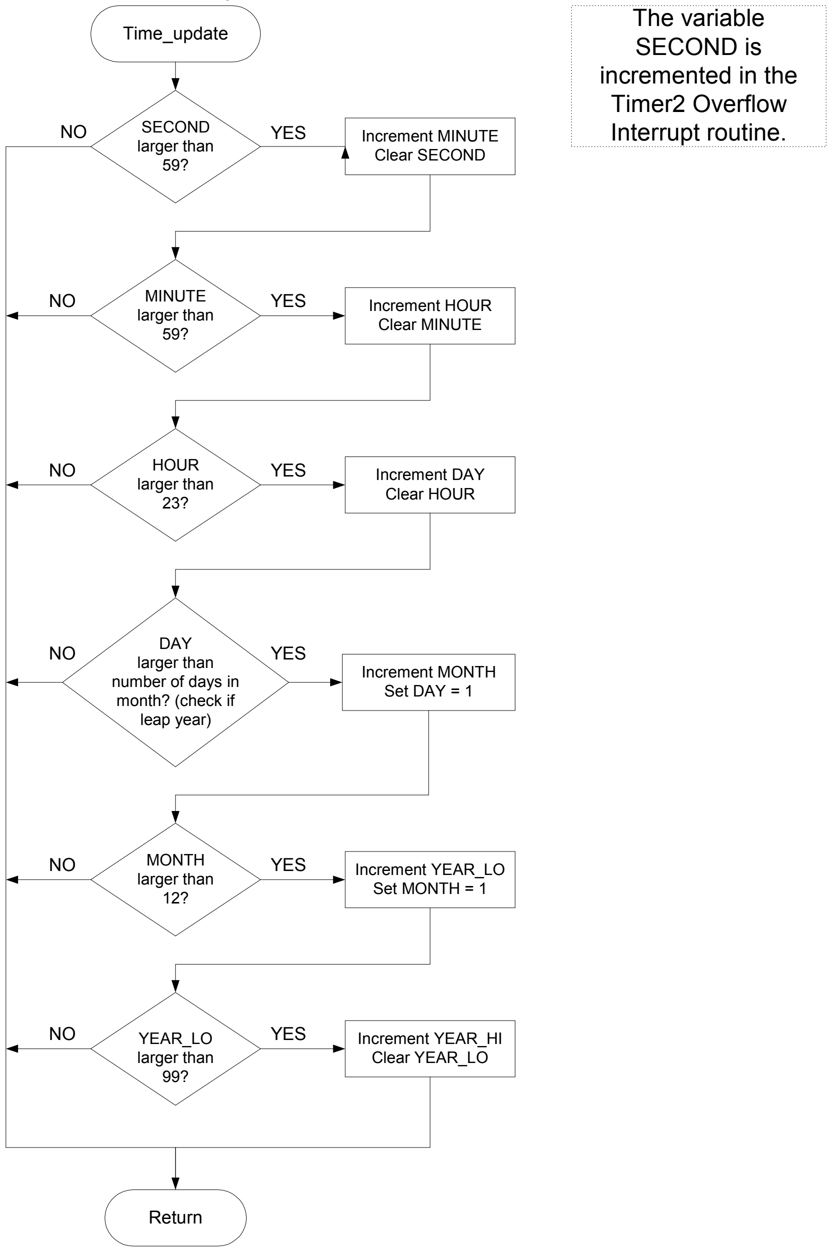
Enable all segment pins on the ATmega169(P). Select the 32kHz as clock source for the LCD, and set the prescaler bits thereafter. Select 1/4 duty cycle and 1/3 bias. Set up Timer/Counter0 Compare Match interrupt to give the required delays for the scrolling and blinking speed of the information on the LCD-display.

Start scrolling the initial string over the LCD-display.

4.4 Time and date update

This routine updates the clock and date according to the variable SECOND that gets incremented every second in the Timer/Counter2 Overflow interrupt routine. The whole update routine is self-explaining from the flow-chart.

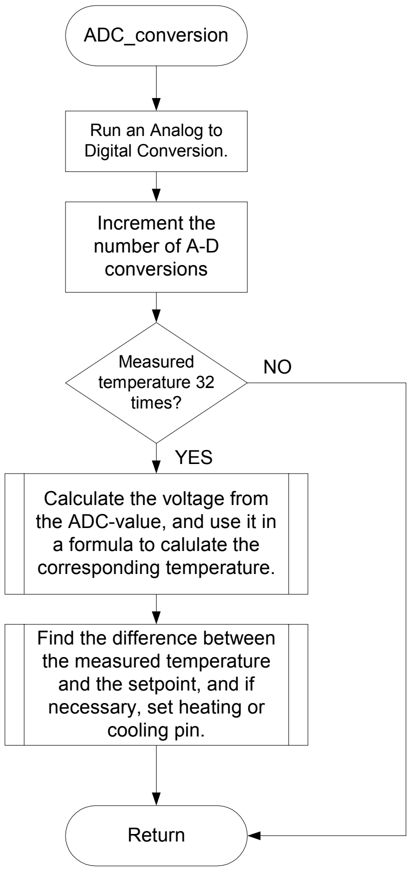
Figure 4-3. Time and date update


Time_update
NO
SECOND larger than 59?
YES
Increment MINUTE Clear SECOND
The variable SECOND is incremented in the Timer2 Overflow Interrupt routine.
NO
MINUTE larger than 59?
YES
Increment HOUR Clear MINUTE
NO
HOUR larger than 23?
YES
Increment DAY Clear HOUR
NO
DAY larger than number of days in month? (check if leap year)
YES
Increment MONTH Set DAY = 1
NO
MONTH larger than 12?
YES
Increment YEAR_LO Set MONTH = 1
NO
YEAR_LO larger than 99?
YES
Increment YEAR_HI Clear YEAR_LO
Return

4.5 Temperature calculation

In this function the voltage over the NTC-thermistor will be measured and the temperature calculated.

Figure 4-4. Temperature calculation

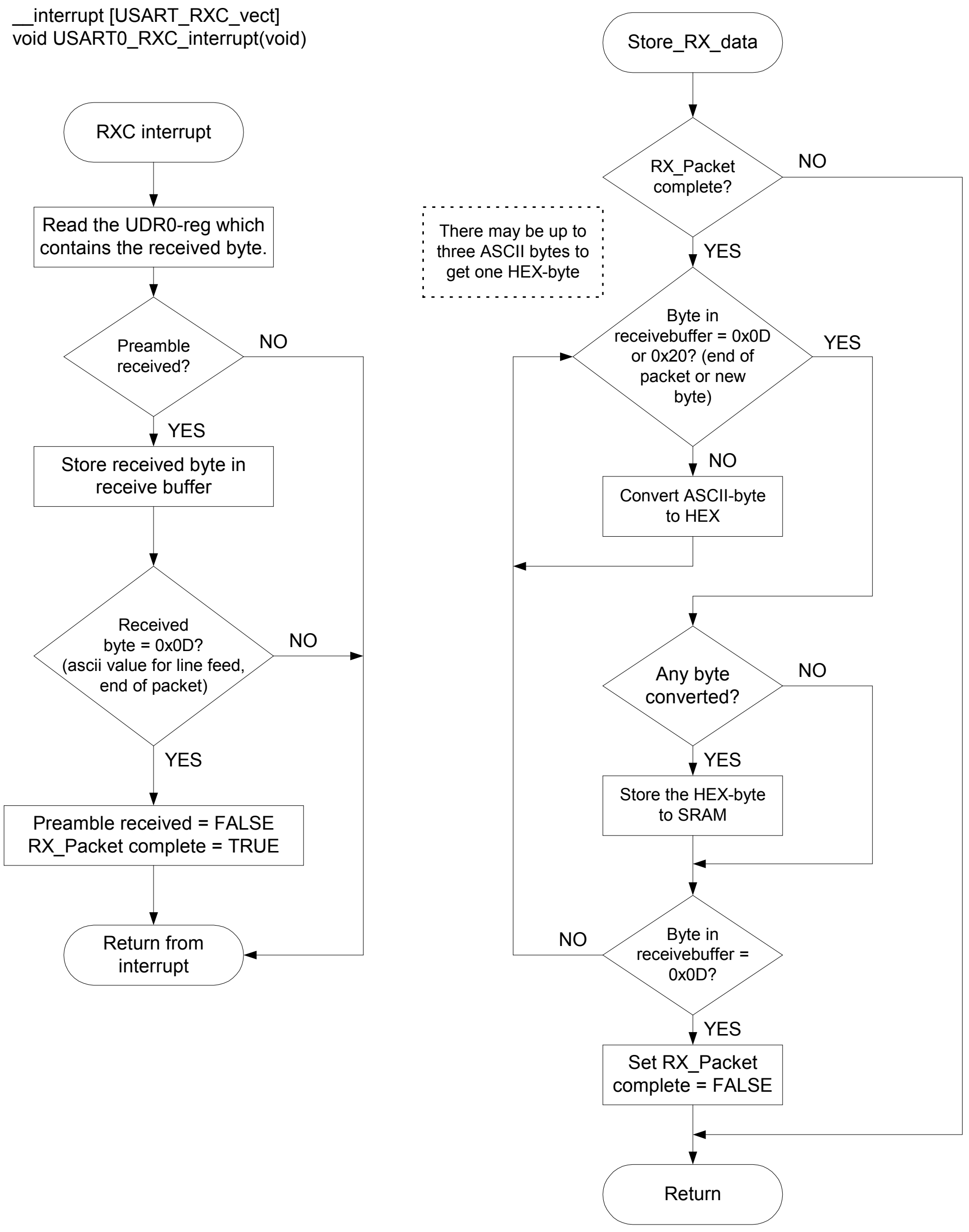

ADC_conversion
Run an Analog to Digital Conversion.
Increment the number of A-D conversions
Measured temperature 32 times?
NO
YES
Calculate the voltage from the ADC-value, and use it in a formula to calulate the corresponding temperature.
Find the difference between the measured temperature and the setpoint, and if necessary, set heating or cooling pin.
Return

Start by doing an A/D conversion. The average of 32 ADC results is used in a formula to calculate the corresponding temperature. The heating or cooling pin is set depending on the difference between the calculated temperature and the temperature set point. The temperature set point is selected by the user. The bigger the difference is, the brighter the heating or cooling LED will shine.

4.6 Receive data from PC

These routines take care of data coming from the PC through the UART interface.

Figure 4-5. Receive packet from PC


__interrupt [USART_RXC_vect]
void USART0_RXC_interrupt(void)
RXC interrupt
Read the UDR0-reg which contains the received byte.
Preamble received?
NO
YES
Store received byte in receive buffer
Received byte = 0x0D? (ascii value for line feed, end of packet)
NO
YES
Preamble received = FALSE
RX_Packet complete = TRUE
Return from interrupt
Store_RX_data
RX_Packet complete?
NO
YES
There may be up to three ASCII bytes to get one HEX-byte
Byte in receivebuffer = 0x0D or 0x20? (end of packet or new byte)
YES
NO
Convert ASCII-byte to HEX
Any byte converted?
NO
YES
Store the HEX-byte to SRAM
Byte in receivebuffer = 0x0D?
NO
YES
Set RX_Packet complete = FALSE
Return

4.6.1 USART_RXC_interrupt

Receiving data from the PC is done in the USART_RXC_interrupt routine. It will discard all data until the correct preamble bytes are received. Then it will store the succeeding bytes in a receive buffer until the byte for Line Feed appears (ASCII value: 0x0D) This indicates the end of the packet and RX_Packet_complete flag will be set to TRUE.

4.6.2 Store_Rx_data:

The packet is then converted from ASCII to hexadecimal. One HEX-byte can contain 1-3 ASCII bytes. ASCII-bytes that belong to different HEX-bytes are separated by an

ASCII-space (0x20). The converted HEX-bytes get continuously stored in the correct place in SRAM until the Line-Feed byte appears, which is the end of the packet.

Table 4-1. Receive Packet from PC

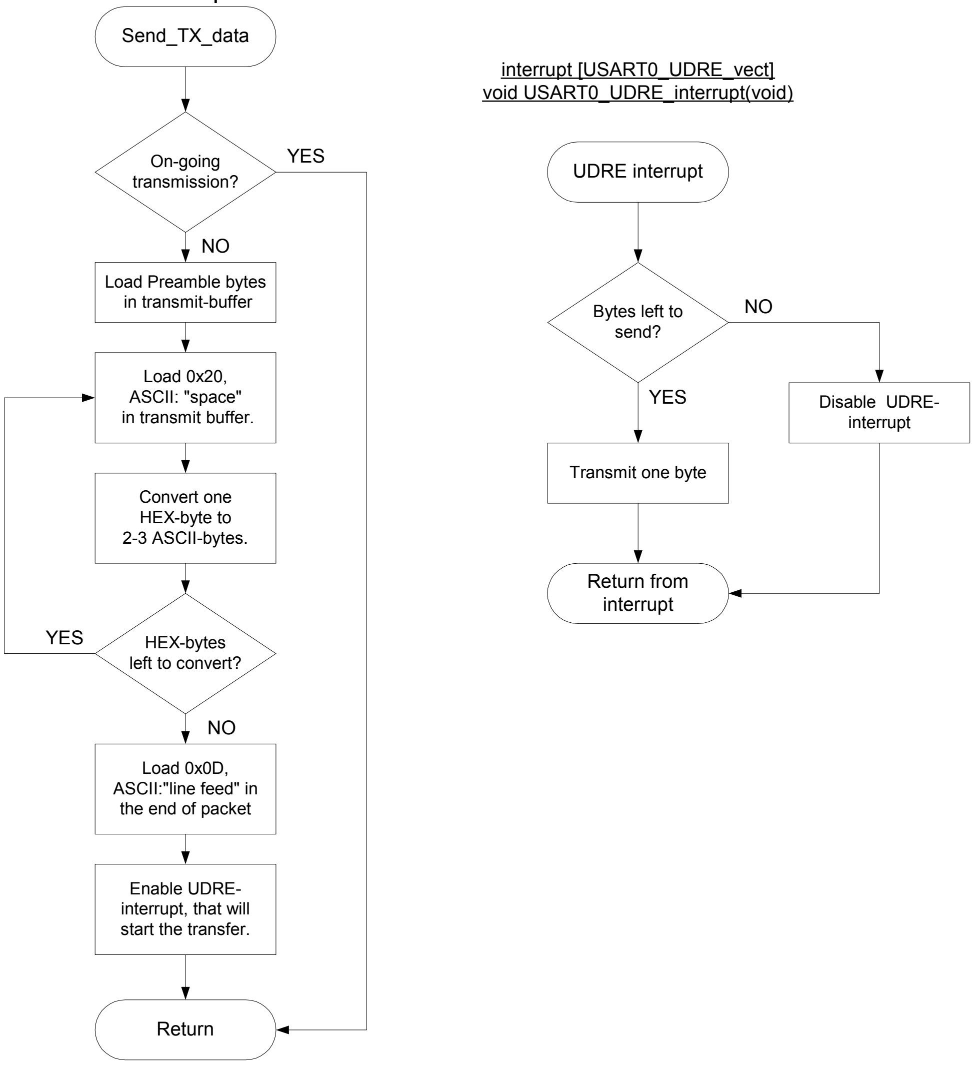
Data	Size
Preamble “STK502”	6 byte
ASCII-space (0x20)	1 byte
HOUR	2 byte
ASCII-space (0x20)	1 byte
MINUTE	2 byte
ASCII-space (0x20)	1 byte
SECOND	2 byte
ASCII-space (0x20)	1 byte
DATE	2 byte
ASCII-space (0x20)	1 byte
MONTH	2 byte
ASCII-space (0x20)	1 byte
YEAR_HI	2 byte
ASCII-space (0x20)	1 byte
YEAR_LO	2 byte
ASCII-space (0x20)	1 byte
SET_POINT	2 byte
ASCII-carriage return (0x0D)	2 byte
ASCII-line feed (0x0A)	2 byte

Transferring the data in ASCII allows a standard terminal to be used on the PC.

4.7 Transmit packet to PC

These routines transmit the data from ATmega169(P) to the PC

Figure 4-6. Transmit packet to PC


Send_TX_data
On-going transmission?
YES
NO
Load Preamble bytes in transmit-buffer
Load 0x20, ASCII: "space" in transmit buffer.
Convert one HEX-byte to 2-3 ASCII-bytes.
YES
HEX-bytes left to convert?
NO
Load 0x0D, ASCII:"line feed" in the end of packet
Enable UDRE-interrupt, that will start the transfer.
Return
interrupt [USART0_UDRE_vect]
void USART0_UDRE_interrupt(void)
UDRE interrupt
Bytes left to send?
NO
YES
Disable UDRE-interrupt
Transmit one byte
Return from interrupt

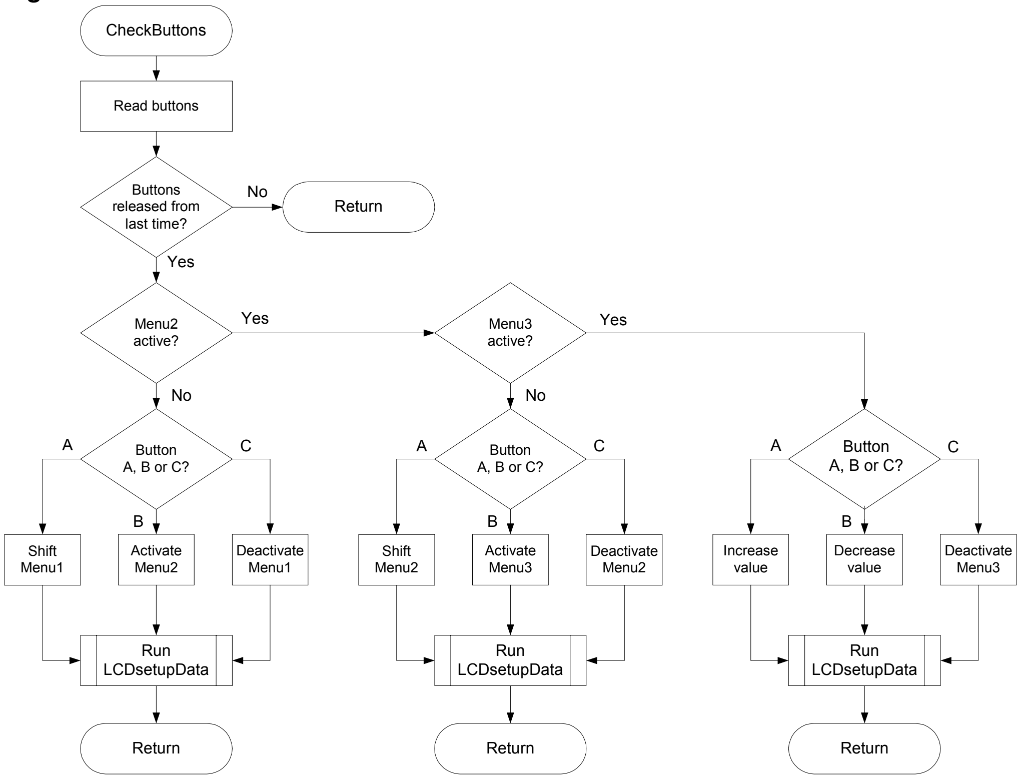
A transmit packet starts with the preamble bytes, and then the HEX-bytes that are to be transmitted get converted to ASCII-bytes and loaded in the packet. Between each HEX-byte that gets converted, an ASCII-byte for space (0x20) is inserted. At the end of the packet, an ASCII-byte for Line Feed is added to indicate the end of frame. The transmission starts by enabling the UDRE interrupt. When all bytes are transmitted the UDRE interrupt gets disabled.

Table 4-2. Transmit packet to PC

Data	**Size**
Preamble “STK502”	6 byte
ASCII-space (0x20)	1 byte
HOUR	2 byte
ASCII-space (0x20)	1 byte
MINUTE	2 byte
ASCII-space (0x20)	1 byte
SECOND	2 byte
ASCII-space (0x20)	1 byte
DATE	2 byte
ASCII-space (0x20)	1 byte
MONTH	2 byte
ASCII-space (0x20)	1 byte
YEAR_HI	2 byte
ASCII-space (0x20)	1 byte
YEAR_LO	2 byte
ASCII-space (0x20)	1 byte
SET_POINT	2 byte
ASCII-space (0x20)	1 byte
TEMP_HIGHBYTE	2 byte
ASCII-space (0x20)	1 byte
TEMP_LOWBYTE	2 byte
ASCII-space (0x20)	1 byte
OFFSET	2 byte
ASCII-space (0x20)	1 byte
Firmware revision	2 byte
ASCII-carriage return (0x0D)	2 byte
ASCII-line feed (0x0A)	2 byte

4.8 STK500 switches

Figure 4-7. CheckButtons



There are three switches that are used as inputs to the application. To do several tasks with only three switches, a menu system is needed. Figure 4-7 shows three menus in a hierarchy, which are used in this code. See Figure 2-1 for an overview of the menus.

Figure 4-7 refers to Button A/B/C, in the application these buttons can be found at:

“ButtonA” is SW7 that is connected to PE7.

“ButtonB” is SW6 that is connected to PE6.

“ButtonC” is SW5 that is connected to PE5.

Example:

After a RESET the LCD is set up to scroll a text. None of the three menus are active. Pressing the SW7 will toggle between the alternatives in Menu1 (Clock, Date, Set point, Temperature, Offset and Contrast)

To adjust the variable MINUTE: Press SW7 until “CLOCK” appears in the LCD-display, and select this by pressing SW6 to activate Menu2 under “CLOCK”. Pressing SW7 will now toggle between the alternatives in Menu2 (Hour, Minute and Second). Press SW7 until the variable MINUTE is blinking in the LCD-display, and select this by pressing SW6. Now Menu3 is activated (the colons should disappear). Pressing SW7 will increase the variable MINUTE and SW6 will decrease. When desired value has been selected, press SW5 to deactivate Menu3, and go back to Menu2. Press SW5 once more to deactivate Menu2 and go back to Menu1.

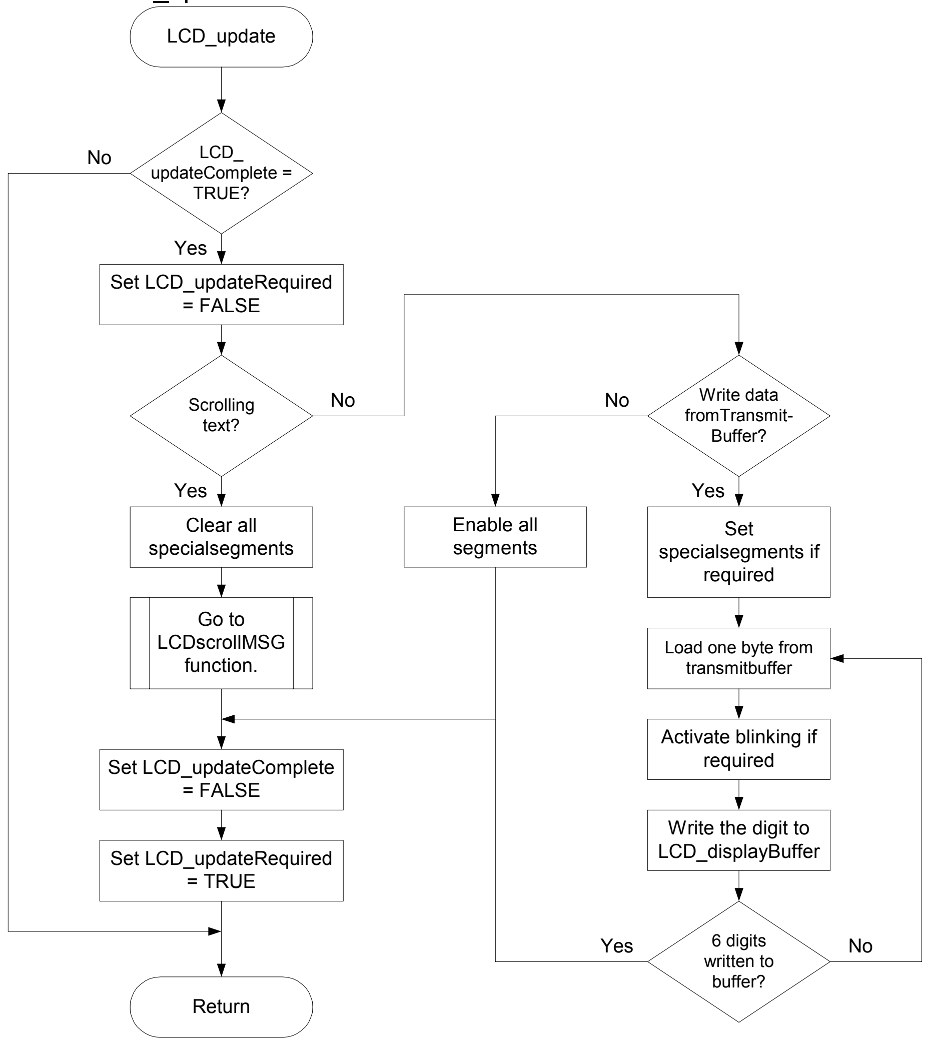
The same procedure can be used to adjust the other variables as well.

4.9 LCD

Writing to the LCD requires an LCD driver. The driver used in this application is described in the application note "AVR065: LCD Driver for the STK502LCD".

4.9.1 LCD update

Figure 4-8. LCD_update



LCD_update
LCD_ updateComplete = TRUE?
No
Yes
Set LCD_updateRequired = FALSE
Scrolling text?
No
Yes
Clear all specialsegments
Go to LCDscrollMSG function.
Write data fromTransmit-Buffer?
No
Yes
Enable all segments
Set specialsegments if required
Load one byte from transmitbuffer
Activate blinking if required
Write the digit to LCD_displayBuffer
6 digits written to buffer?
Yes
No
Set LCD_updateComplete = FALSE
Set LCD_updateRequired = TRUE
Return

This function will load data into the LCD_displayBuffer.

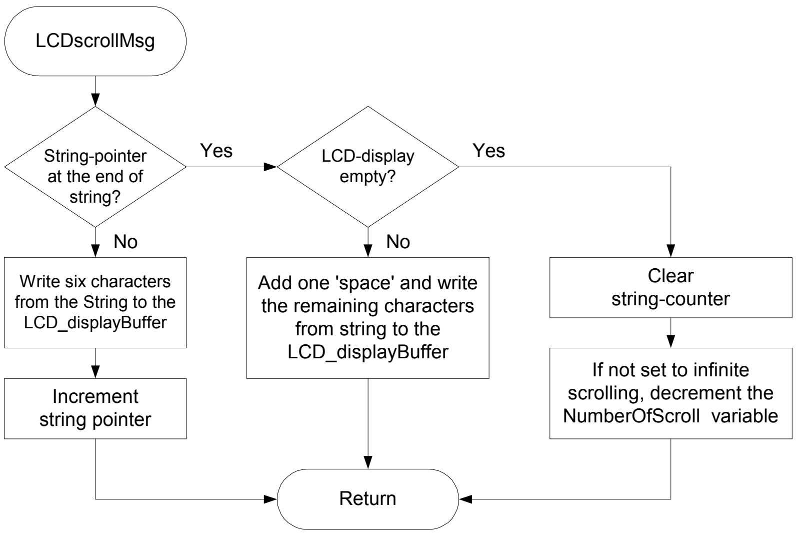
First check if the LCD has been updated with the data already in the LCD_displayBuffer. If so, set the LCD_update required to FALSE. This will prevent the LCD to be updated with incomplete data, if an LCD Start of Frame interrupt should occur during this function.

If a text-string is to be scrolled, clear display and call the LCDscrollMSG function. If no text to scroll, check if there is data to write from the TransmitBuffer, and load the data into the LCD_displayBuffer. Digits can be set to blink on the display. To do this the digit will be loaded with either its data value or a ASCII-space (0x20), depending on the variable LCD_Blink.

After the LCD_displayBuffer has been updated, the LCD_updatedComplete will be set to FALSE and LCD_updateRequired to TRUE. This will cause the LCD_displayBuffer to be written to the LCD in the LCD Start of Frame interrupt.

4.9.2 Scroll function

Figure 4-9. LCDscrollMsg

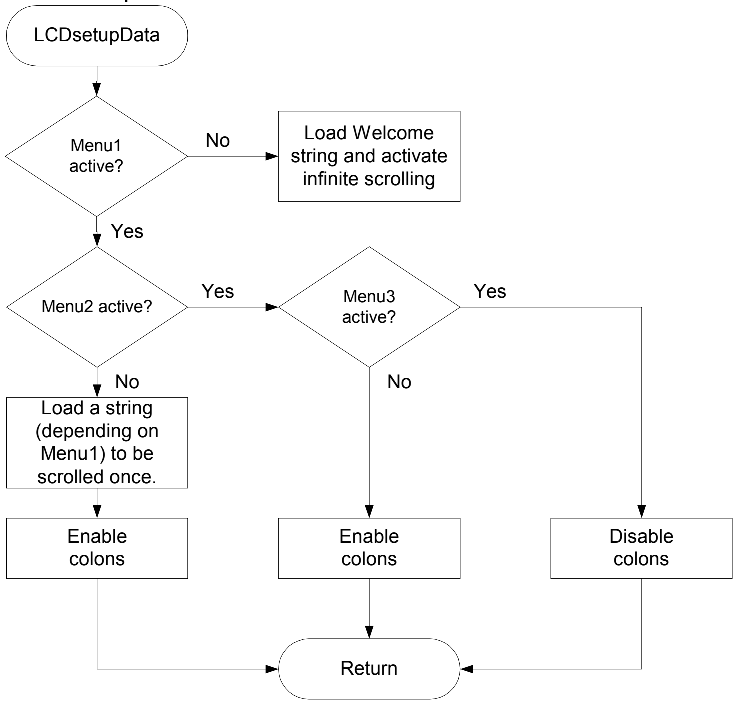

LCDscrollMsg
String-pointer at the end of string?
Yes
LCD-display empty?
Yes
No
No
Write six characters from the String to the LCD_displayBuffer
Add one 'space' and write the remaining characters from string to the LCD_displayBuffer
Clear string-counter
Increment string pointer
If not set to infinite scrolling, decrement the NumberOfScroll variable
Return

This function shifts the six digits on the LCD one step to the left. The scroll function uses a pointer to keep track of what characters to shift in and out of the LCD. When all the six digits have been updated, the pointer gets incremented by one in order to shift the text-string one step the next time this function is called.

If the pointer has reached the end of the string, the LCD has to be filled up with one ASCII-space at the time until all of the six digits are blank. This will “fade” out the text string.

4.9.3 LCD set-up data

Figure 4-10. LCDsetupData

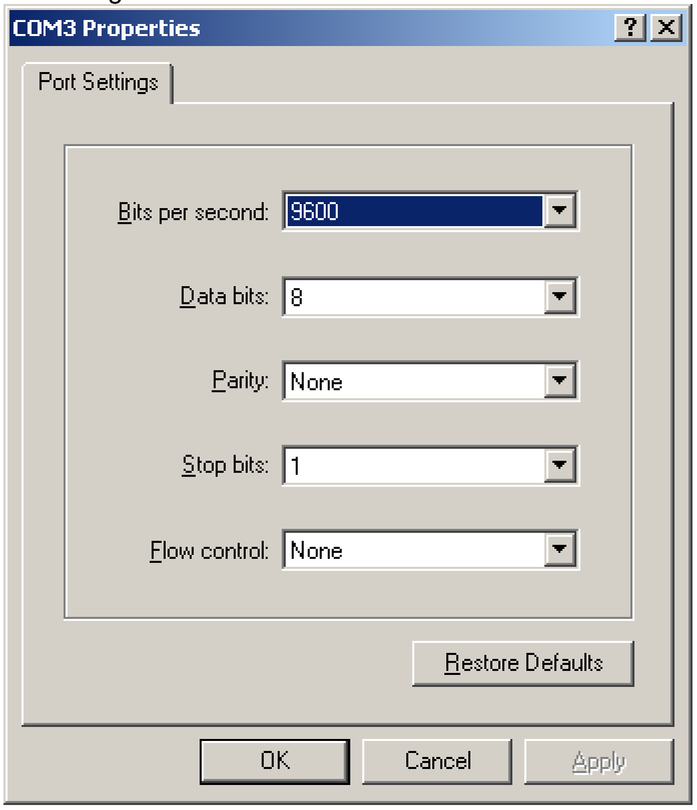


If Menu1 isn't active the welcome will scroll over the LCD. If Menu1 is active but not Menu2, the corresponding string will be scrolled once over the LCD and then the belonging data. If Menu2 is active but not Menu3, just enable the colons. And if Menu3 is active, disable the colons to indicate that the current variable can now be adjusted.

5 Terminal

All temperature and time information is transmitted through the UART-interface. A program on a PC can receive this data by connecting a serial-cable between the "RS232 SPARE" on the STK500 and a com-port on the PC. A standard terminal can be used, e.g. HyperTerminal. Set up the terminal with the settings shown in Figure 5-1.

Figure 5-1. Port Settings


COM3 Properties

Port Settings

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

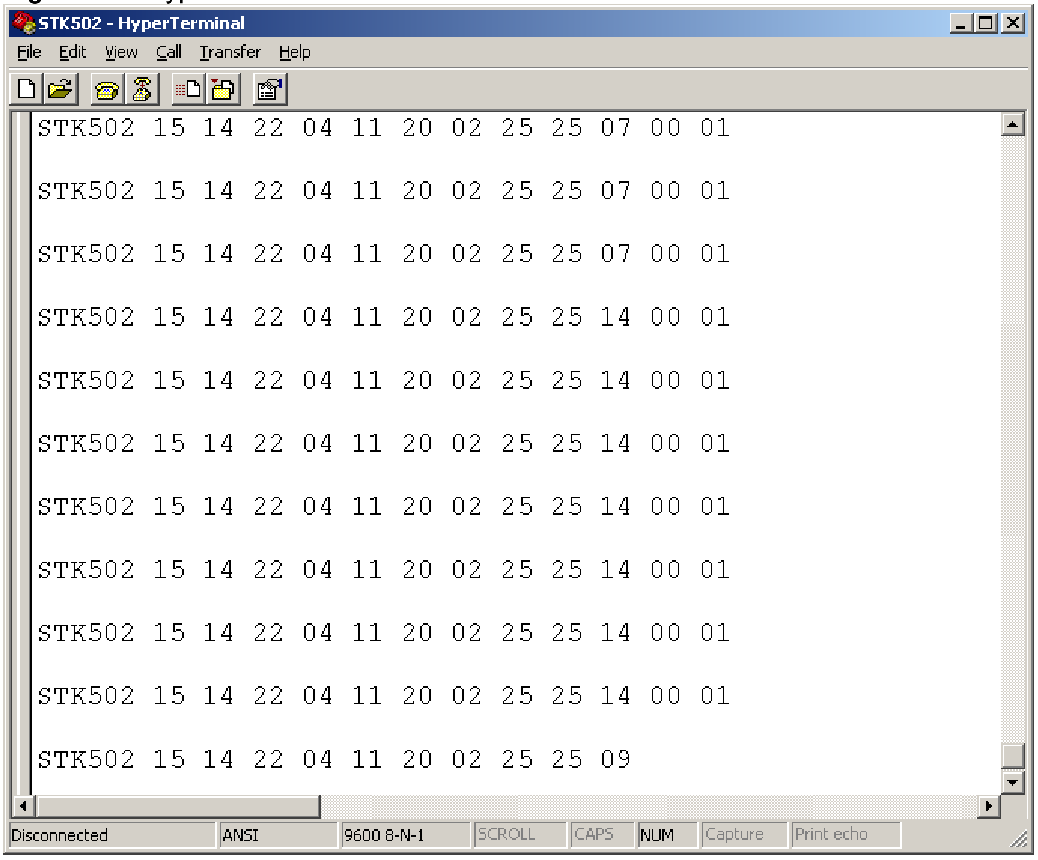
Flow control: None

Press the connect-button and the data from the ATmega169(P) should appear as in Figure 5-2. The data is presented according to Table 5-1.

Table 5-1. Transmit Packet from ATmega169 according to Figure 5-2

Data	**Value**
Preamble	STK502
Hour	15
Minute	14
Second	22
Day	04
Month	11
Year high	20
Year low	02
Set point	25
ºC high byte	25
ºC low byte	14
Offset	00
Versions number	01

Figure 5-2. HyperTerminal



```
STK502 15 14 22 04 11 20 02 25 25 07 00 01

STK502 15 14 22 04 11 20 02 25 25 07 00 01

STK502 15 14 22 04 11 20 02 25 25 07 00 01

STK502 15 14 22 04 11 20 02 25 25 14 00 01

STK502 15 14 22 04 11 20 02 25 25 14 00 01

STK502 15 14 22 04 11 20 02 25 25 14 00 01

STK502 15 14 22 04 11 20 02 25 25 14 00 01

STK502 15 14 22 04 11 20 02 25 25 14 00 01

STK502 15 14 22 04 11 20 02 25 25 14 00 01

STK502 15 14 22 04 11 20 02 25 25 14 00 01

STK502 15 14 22 04 11 20 02 25 25 09
```

Disconnected | ANSI | 9600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

One can also adjust the variables within the ATmega169(P) from the terminal. This has to be done according to Table 4-1. E.g. write: ”STK502 14 37 02 25 11 20 02 24” in the terminal, and press enter to indicate end of frame. This will adjust the clock to 14h37m02s, the date to 25. November 2002, and the temperature set point will be 24°C.

6 Table of Contents

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature