# objective:

* Understand and use UART, I2C, SPI peripherals.
* Understand how to communicate with RTC and EEPROM.

# references:

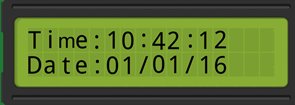
* Lab manual chapter 7, 9, 11
* Atmel-2505-Setup-and-Use-of-AVR-Timers\_ApplicationNote\_AVR130.pdf

# EXPERIMENT 1:

1. Connect the TxD and RxD pins of UART0 to UART\_TxD0 and UART\_RxD0 signals on header J85 in the UART block.
2. Connect a USB-Serial cable to the experimentation kit.
3. Set up the Hercules program with a baud rate of 9600, 8-bit data, no parity, 1 stop bit, and no handshake.
4. Use sample programs from the experiment guide to write a program that initializes UART0 with the specified parameters, waits to receive one byte from UART0, and then sends it back to UART0.
5. Use Hercules to send a character to the kit and observe the received data to check the program's functionality. (Note: The CPU clock frequency on the kit is 8 MHz.)

# EXPERIMENT 2:

1. Connect the SDA and SCL signals of AVR to the corresponding signals on the RTC module. Connect one port pin to the MFP signal. Connect a 16x2 LCD to one port of AVR.
2. Write a subprogram to initialize the RTC with the current time, configure the MFP signal to have a 1Hz frequency, and read the date, month, year, hour, minute, and second values from the RTC. Then, update these values on the LCD.
3. Compile the program and observe the LCD to verify its functionality.



# EXPERIMENT 3:

1. Connect the MOSI and SCK signals from the SPI port of AVR to the SDI and CLK signals of the shift register. Connect two port pins to the nCLR and LATCH signals. Connect the output of the shift register to the Bar LED.
2. Connect the UART signals as in exercise 1.
3. Write a program that receives one value from UART and outputs it to the Bar LED using SPI.

# EXPERIMENT 4:

1. Connect the MOSI, MISO, and SCK signals from the SPI port of AVR to the corresponding signals on header J80. Connect one port pin to the nCS signal.
2. Connect the UART signals as in exercise 1.
3. Connect one port to the Bar LED.
4. Write a program to count the number of characters received from UART and display them on the Bar LED. Every time a byte is received, increment the count and write it to EEPROM. When the microcontroller loses power and then regains it, read the count from EEPROM and use it as the starting value.

# EXPERIMENT 5:

1. Connect the UART signals as in exercise 1.
2. Connect one port to the Bar LED.
3. Write a program to count the number of characters received from UART and display them on the Bar LED. Every time a byte is received, increment the count and write it to the AVR's internal EEPROM. When the microcontroller loses power and then regains it, read the count from the internal EEPROM and use it as the starting value.

# EXPERIMENT 1:

1. Answer the following questions:
   1. **With a clock frequency of 8 MHz, how much will the actual baud rate deviate from the desired 9600 baud?**

Actual deviation from expectation is 0.2%

* 1. **What is the purpose of the UDRE flag?**

The hardware automatically sets the UDREn=1 flag when the UDRn broadcast counter is empty and clears the flag UDREn=0 when writing data to be transmitted to this register

* 1. **Explain the difference between hardware UART and software UART (bit-banging UART).**

The main difference between Hardware UART and Software UART is the method they use to perform UART communication. Hardware UART is a piece of hardware built into the microcontroller, while Software UART is a software method for controlling pins GPIO of the microcontroller to perform UART communication.

* 1. **Which port pins correspond to the TxD0 and RxD0 pins of UART0?**

TxD0: is PD1 pin

RxD0: is PD0 pin

* 1. **How many hardware UARTs does the Atmega324 have?**

There are 2 hardware UARTs: UART0 and UART1.

1. Provide the source code with comments.

| main:  call USART\_Init  loop:  call USART\_ReceiveChar  call USART\_SendChar  rjmp loop  ;init UART0  ;CPU clock is 8Mhz  USART\_Init:  ; Set baud rate to 9600 bps with 8 MHz clock  ldi r16, 103  sts UBRR0L, r16  ;set double speed  ldi r16, (1 << U2X0)  sts UCSR0A, r16  ; Enable transmitter and receiver  ldi r16, (1 << TXEN0) | (1 << RXEN0)  sts UCSR0B, r16  ; Set frame format: 8 data bits, no parity, 1 stop bit  ldi r16, (1 << UCSZ01) | (1 << UCSZ00)  sts UCSR0C, r16  ret  ;receive 1 byte in r16  USART\_ReceiveChar:  push r17  ; Wait for the transmitter to be ready  USART\_ReceiveChar\_Wait:  lds r17, UCSR0A  sbrs r17, RXC0 ;check USART Receive Complete bit  rjmp USART\_ReceiveChar\_Wait  lds r16, UDR0 ;get data  pop r17  ret  ;send out 1 byte in r16  USART\_SendChar:  push r17  ; Wait for the transmitter to be ready  USART\_SendChar\_Wait:  lds r17, UCSR0A  sbrs r17, UDRE0 ;check USART Data Register Empty bit  rjmp USART\_SendChar\_Wait  sts UDR0, r16 ;send out  pop r17  ret |
| --- |

# EXPERIMENT 2:

1. Answer the following questions:
   1. **Which pins are the SCL and SDA pins on the AVR?**

SCL: is PC0 pin

SDA: is PC1 pin

* 1. **Draw a diagram illustrating the connections in the experiment.**

1. Provide the source code with comments.

| ........................................................................................................................ |
| --- |

# EXPERIMENT 3:

1. Answer the following questions:
   1. **According to the datasheet of the 74HC595, what is the highest clock frequency it can accept?**

According to the datasheet of 74HC595, the maximum clock frequency that it can operate at dynamic is 100 MHz. However, the maximum frequency that can be used in one Specific applications may be lower depending on factors such as dosage load analysis and specific circuit design. It is important to consult datasheet and any additional application notes to ensure performance correct operation of equipment in a particular circuit

* 1. **With a clock of 8 MHz, what is the maximum SPI speed for the Atmega328?**

Because the SPI speed cannot be greater than ¼ of the clock speed for the chip SPI speed max = Fosc / 16 = ½ MHz

1. Provide the source code with comments.

| .equ SS = 4  .equ MOSI = 5  .equ SCK = 7  .equ CLEAR = 0  .equ LATCH = 1  ; PA0, PA1, PB5, PB7 are outputs  main:  call USART\_Init  call SPI\_Init  call PORT\_Init  loop:  call USART\_ReceiveChar  cbi PORTA, CLEAR  sbi PORTA, CLEAR  call SPI\_SendChar  sbi PORTA, LATCH  cbi PORTA, LATCH  rjmp loop    ;init UART0  ;CPU clock is 8Mhz  USART\_Init:  ; Set baud rate to 9600 bps with 8 MHz clock  ldi r16, 103  sts UBRR0L, r16  ldi r16, (1 << U2X0)  sts UCSR0A, r16  ldi r16, (1 << TXEN0) | (1 << RXEN0)  sts UCSR0B, r16  ; Set frame format: 8 data bits, no parity, 1 stop bit  ldi r16, (1 << UCSZ01) | (1 << UCSZ00)  sts UCSR0C, r16  ret  USART\_ReceiveChar:  push r17  ; Wait for the transmitter to be ready  USART\_ReceiveChar\_Wait:  lds r17, UCSR0A  sbrs r17, RXC0 ;check USART Receive Complete bit  rjmp USART\_ReceiveChar\_Wait  lds r16, UDR0 ;get data  pop r17  ret  SPI\_Init:  ; Configurate the outputs of SPI  ldi r16, (1 << SS)|(1 << SCK)|(1 << MOSI)  out DDRB, r16  sbi PORTB, SS  ; SPI enable, Master select, SPR10:00 = 0:1 (Fosc/16)  ldi r16, (1 << SPE0)|(1 << MSTR0)|(1 <<SPR00)  out SPCR0, r16  ; Double speed => Fosc/8  ldi r16, (1 << SPI2X0)  out SPSR0, r16  ret  SPI\_SendChar:  out SPDR0, r16 ; Send out data  SPI\_SendChar\_Wait:  in r16, SPSR0  sbrs r16, SPIF0 ; SPIF0 = 1 finish transmitting  rjmp SPI\_SendChar\_Wait ; Wait unitl transmission is complete  ret  PORT\_Init:  sbi DDRA, CLEAR  cbi PORTA, CLEAR  sbi DDRA, LATCH  cbi PORTA, LATCH  sbi DDRB, MOSI  cbi PORTB, MOSI  sbi DDRB, SCK  cbi PORTB, SCK  ret |
| --- |

# EXPERIMENT 4:

1. Answer the following questions:
   1. What is the EEPROM size of the 25AA1024?
   2. According to the datasheet, what is the fastest clock frequency that can be provided to this EEPROM?
2. Provide the source code with comments.

| ........................................................................................................................ |
| --- |

# EXPERIMENT 5:

1. Answer the following questions:
   1. **What is the EEPROM size of the Atmega324PA?**

It have 1KB

* 1. **List the differences between SRAM and EEPROM.**

EEPROM can retain values in memory even when the power is turned off, SRAM does not.

* 1. **List the differences between Flash and EEPROM.**

1. Provide the source code with comments.

| ........................................................................................................................ |
| --- |