AVR Training Kit
Lab Book

Date: 15th March, 2010

Document Revision: 1.02

BiPOM Electronics

16301 Blue Ridge Road, Missouri City, Texas 77489
Telephone: (713) 283-9970 Fax: (281) 416-2806
E-mail: info@bipom.com
Web: www.bipom.com

WARRANTY:

BiPOM Electronics warrants AVR Training Kit for a period of 1 year. If the Kit becomes defective during this period, BiPOM Electronics will at its option, replace or repair the Kit. This warranty is voided if the product is subjected to physical abuse or operated outside stated electrical limits. BiPOM Electronics will not be responsible for damage to any external devices connected to the Kit. BiPOM Electronics disclaims all warranties express or implied warranties of merchantability and fitness for a particular purpose. In no event shall BiPOM Electronics be liable for any indirect, special, incidental or consequential damages in connection with or arising from the use of this product. BiPOM's liability is limited to the purchase price of this product.

TABLE OF CONTENTS

Introduction

The purpose of the AVR Training Lab is to familiarize the student with developing practical applications using the ATMEL ATMEGA2560 single-chip microcontroller. .

The AVR Training Kit consists of the following components:

- MicroTRAK Carrier Board
- MINI-MAX/AVR-C Microcontroller Board
- TB-1 Training Board
- PROTO-1 Prototyping Board
- AVR I/O Module
- LCD242 LCD
- KP1-4X4 Keypad
- Cables
- Adapter
- Training Manuals
- Labbook

The following external items are required for each training kit station:

- IBM Compatible Personal Computer (PC) running Windows 95/98/XP or greater. Minimum 256MB memory and 100 MB of available hard disk space.

- One available RS232 Serial port (COM1 through COM8).

Figure 1 shows all the components connected together.

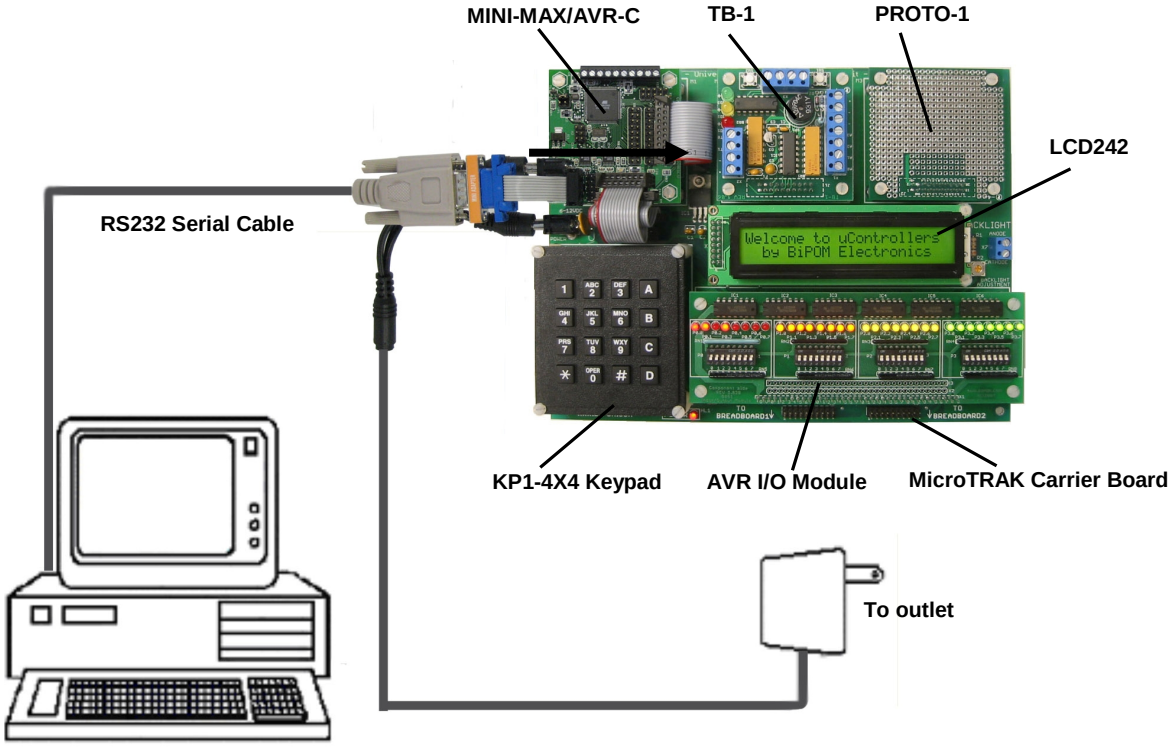


Figure 1

Getting Started

Log on to Windows before using the AVR Training Kit. Enter the user name and password that your instructor has given to you before the Lab. Make sure to log out when you are done with the PC at the end of the Lab. Do not share your username or password with anybody.

The programs are written as C Language Program files (with the extension .). These C files are created using WinAVR.

You can edit and save programs and download to the Training Board using AVR Studio. Creating programs and running them on the Training Board consists of the following steps

1) Download and install WinAVR from http://winavr.sourceforge.net/

2) Download and install AVR Studio 4.16 or later from http://www.atmel.com/avrstudio. Also download any service pack for AVR Studio that may be available on ATMEL website. Service pack should be installed after AVR Studio has been installed.

3) Micro-IDE is a part of BiPOM's ARM Development System.
Download and install the development system from http://www.bipom.com/products/us/319589.html

You can edit and save programs and download to the Training Board using AVR Studio. Creating programs and running them on the Training Board consists of the following steps:

- Edit an existing or create a new project using AVR Studio source file editor.
- Compile the program using WinAVR (*C Compiler*)
- Download the program to the Training Board using Micro-IDE Serial Loader.
- Run and debug the program on the Training Board using Micro-IDE Terminal Window.

LAB1 – Introduction to the AVR Training Kit

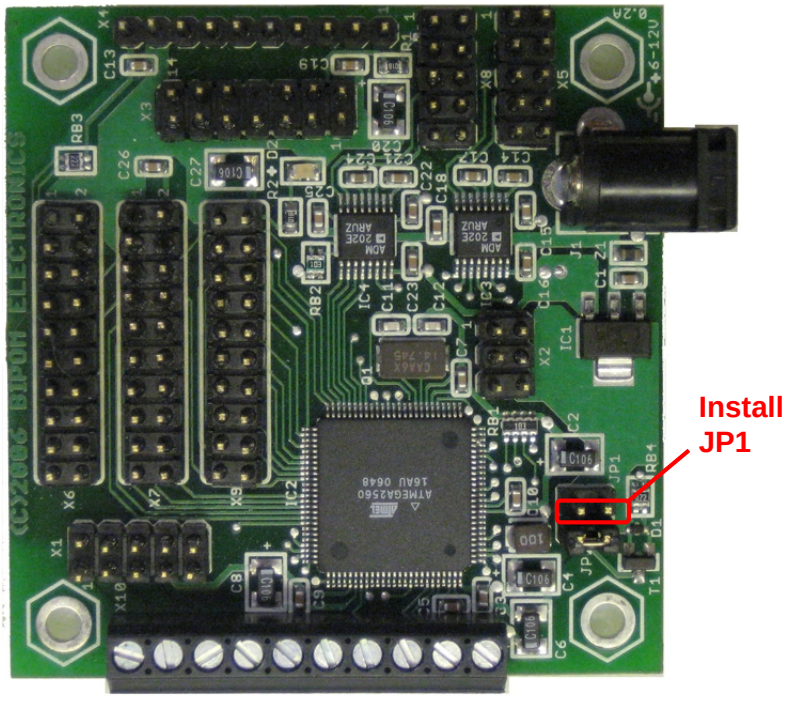
Overview

The purpose of this lab is to familiarize you with the AVR Training Kit and the program development environment. In this lab, you will create a simple program in C language, compile the program to form a hex file, download the program to the MINI-MAX/AVR-C Microcontroller board and execute the program.

The knowledge developed in this lab will be very useful in subsequent labs when working with the ATMEL ATMEGA2560 single-chip flash micro-controller to develop programs.

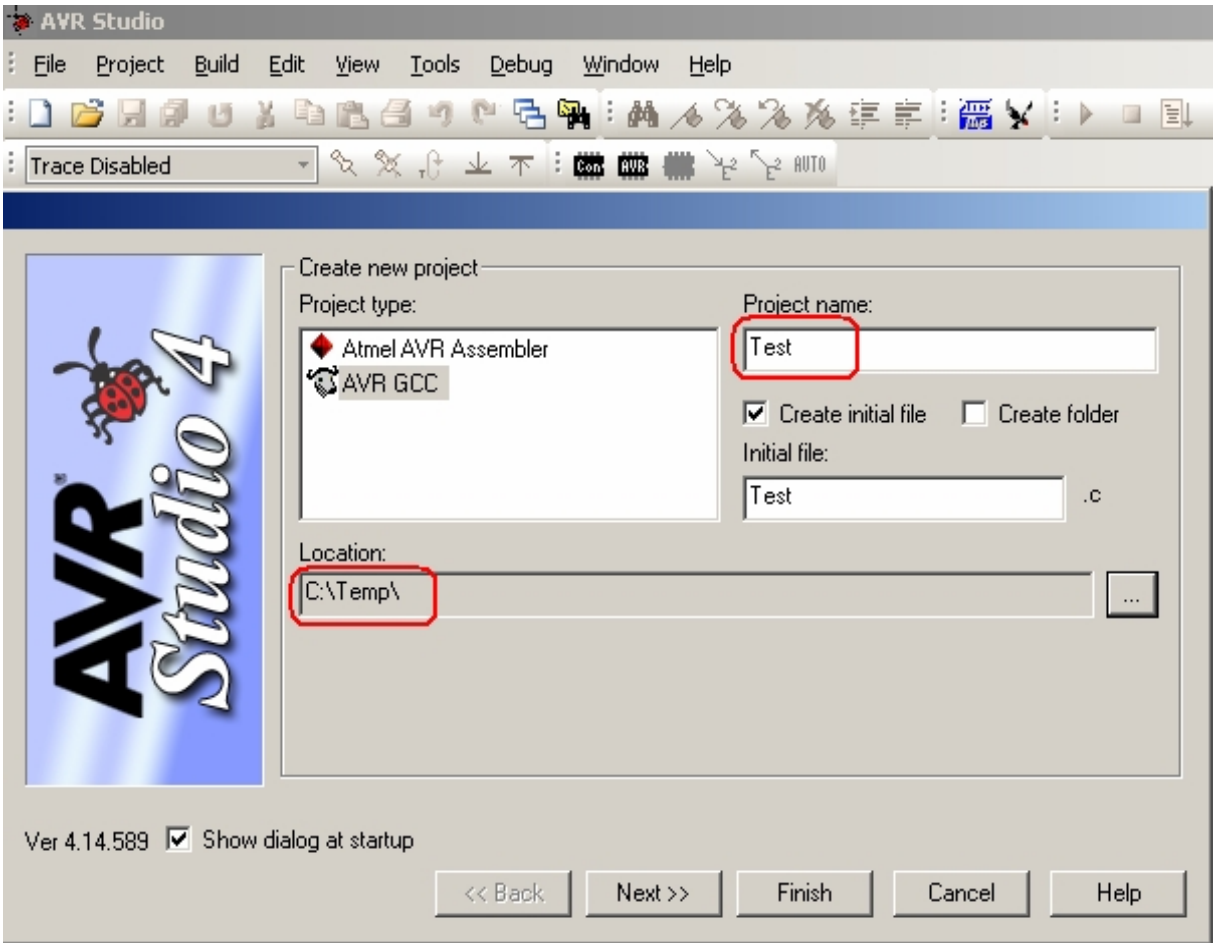
Instructions

1) Place the MINI-MAX/AVR-C Microcontroller board on a clean, non-conductive surface.
2) Connect the provided a 6VDC power supply plug to the power jack on the MINI-MAX/AVR-C. Do not connect the power supply to the outlet yet.
Do not use a power supply other than one that is supplied or approved by BiPOM Electronics. Use of another power supply voids the warranty and may permanently DAMAGE the board or the computer to which the board is connected.
3) Connect the 10-pin header of serial cable to X8 connector of MINI-MAX/AVR board.
Mini-Max/AVR-C board uses UART1 as BOOT serial port.
4) Connect the other end of the serial cable to your PC's COM port.
5) Install JP1 jumper. When this jumper is installed, the board runs in BOOT mode.
6) Connect the 6VDC power supply to a suitable wall outlet. Red LED on MINI-MAX/AVR-C board will turn ON.

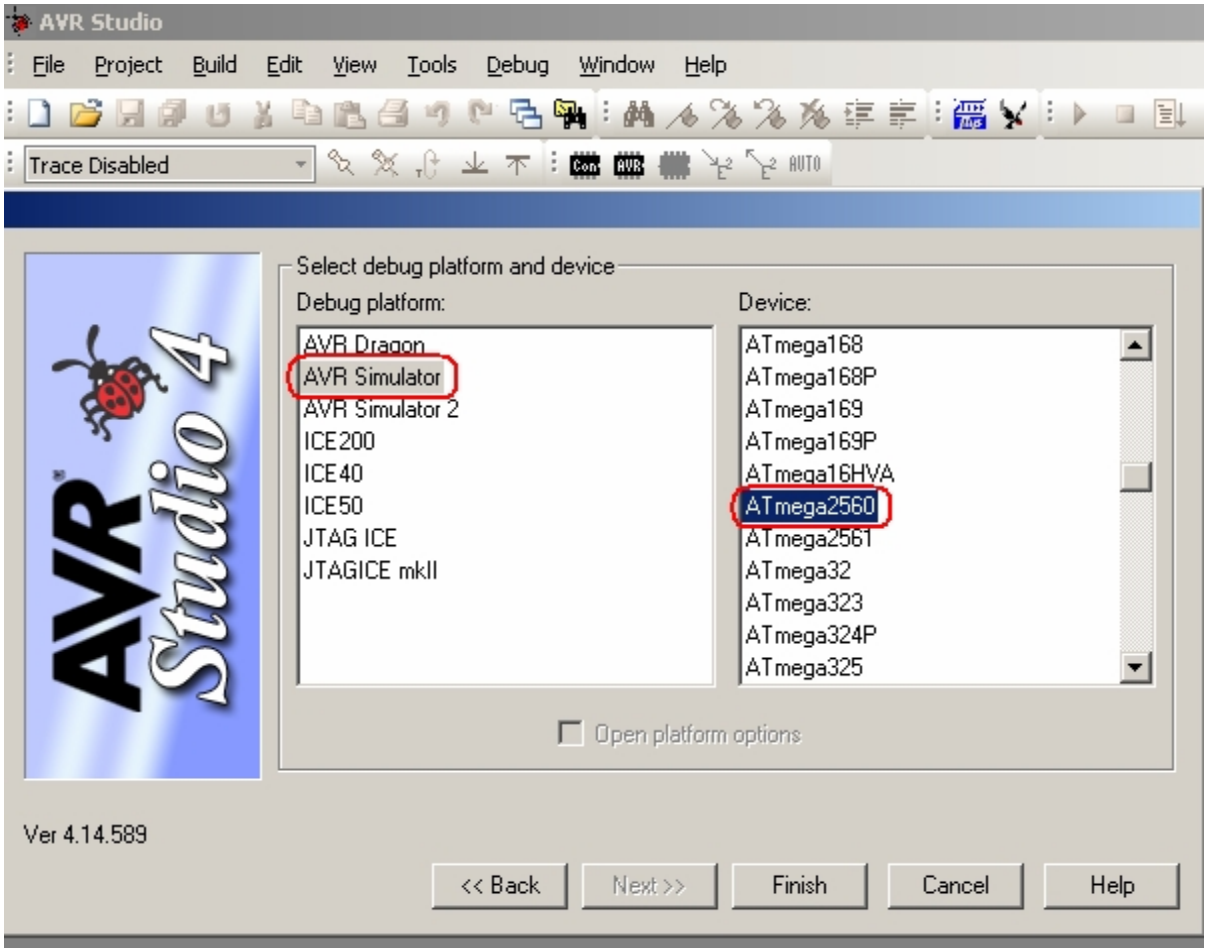


Test example

1.1) To create your own test project please run AVR Studio, select Project menu and select New Project. This will display the New Project dialog (see below). Select AVR GCC, enter the name of the new project and its location and click Next button.

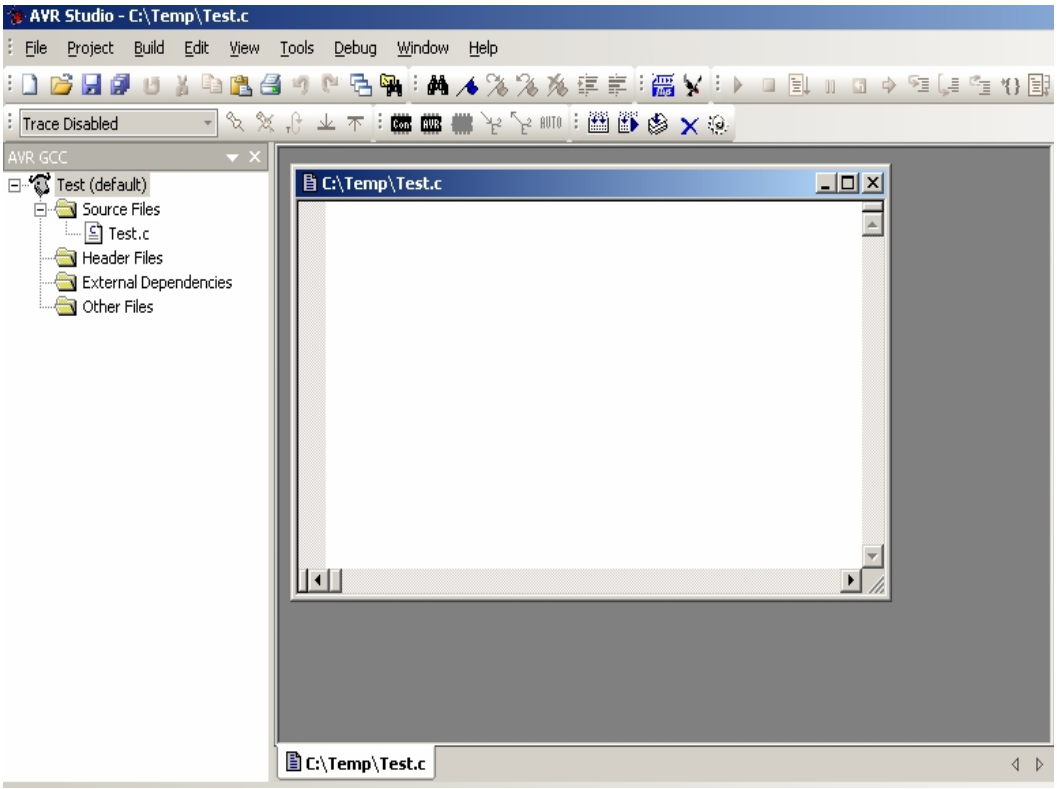


1.2) Select a debug platform and device, then click Finish button.



The new project with 'Test' name will be created under c:\Temp.

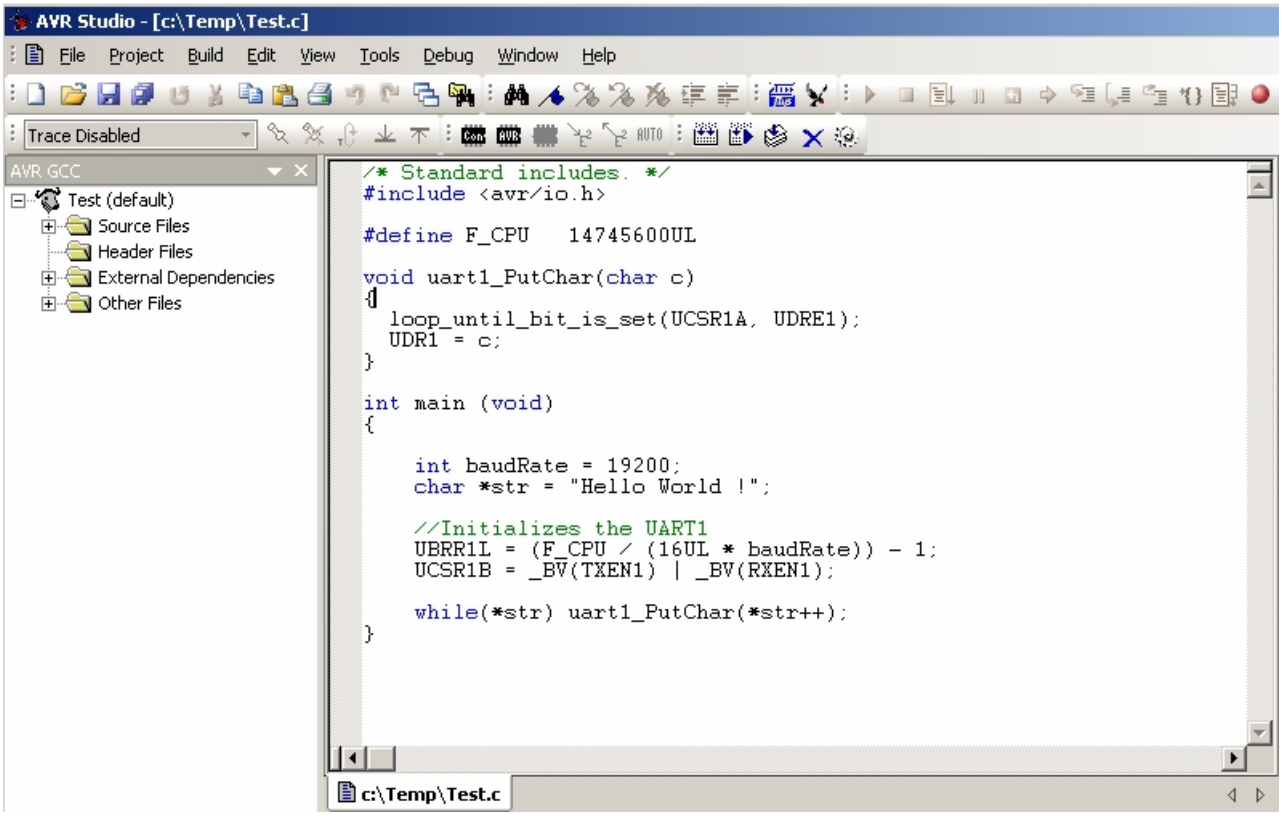
1.3) A blank C file (Test.c) will be created automatically as well.



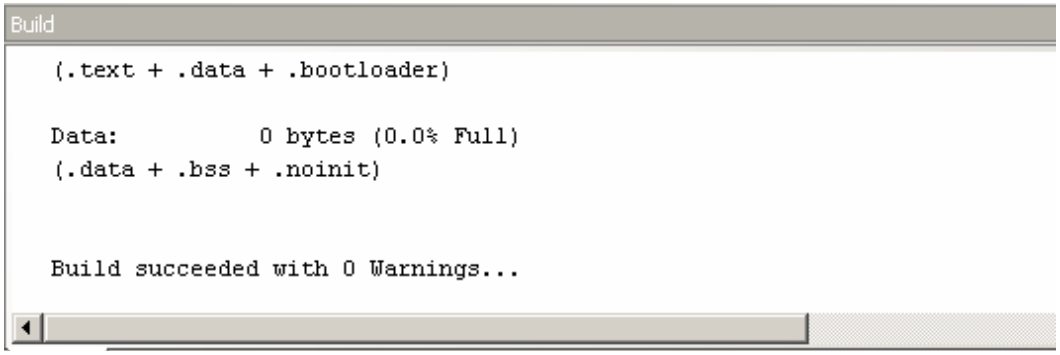
1.4) Type the following C-code:

```
/* Standard includes. */
#include <avrio.h>

#define F_CPU 14745600UL

void uart1_PutChar(char c)
{
	loop_until_bit_is_set(UCSR1A, UDRE1);
	UDR1 = c;
}

int main (void)
{
	int baudRate = 19200;
	char *str = "Hello World !";

	//Initialize the UART1
	UBRR1L = (F_CPU  (16UL * baudRate)) - 1;
	UCSR1B = _BV(TXEN1) | _BV(RXEN1);

	while(*str) uart1_PutChar(*str++);
}
```

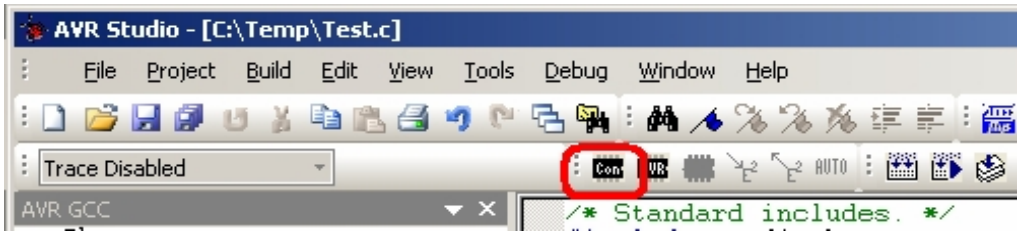
The complete C program should look like this:



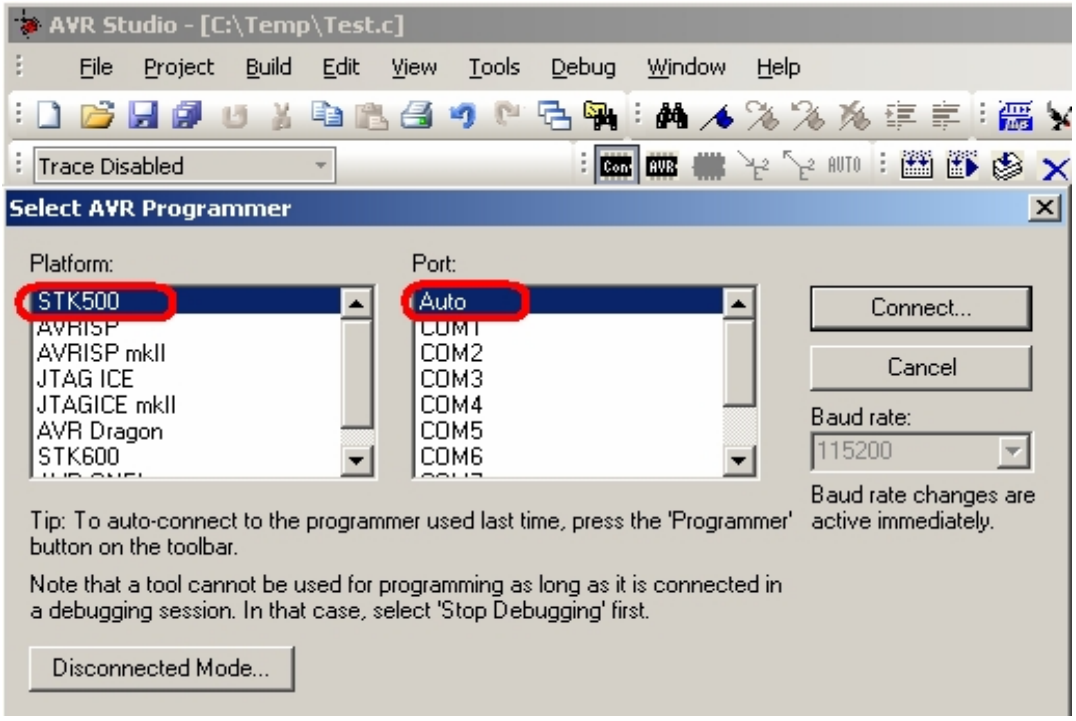
1.5) Build the program by clicking the Build button. If the program builds successfully, you will see the following messages on the Build Window.



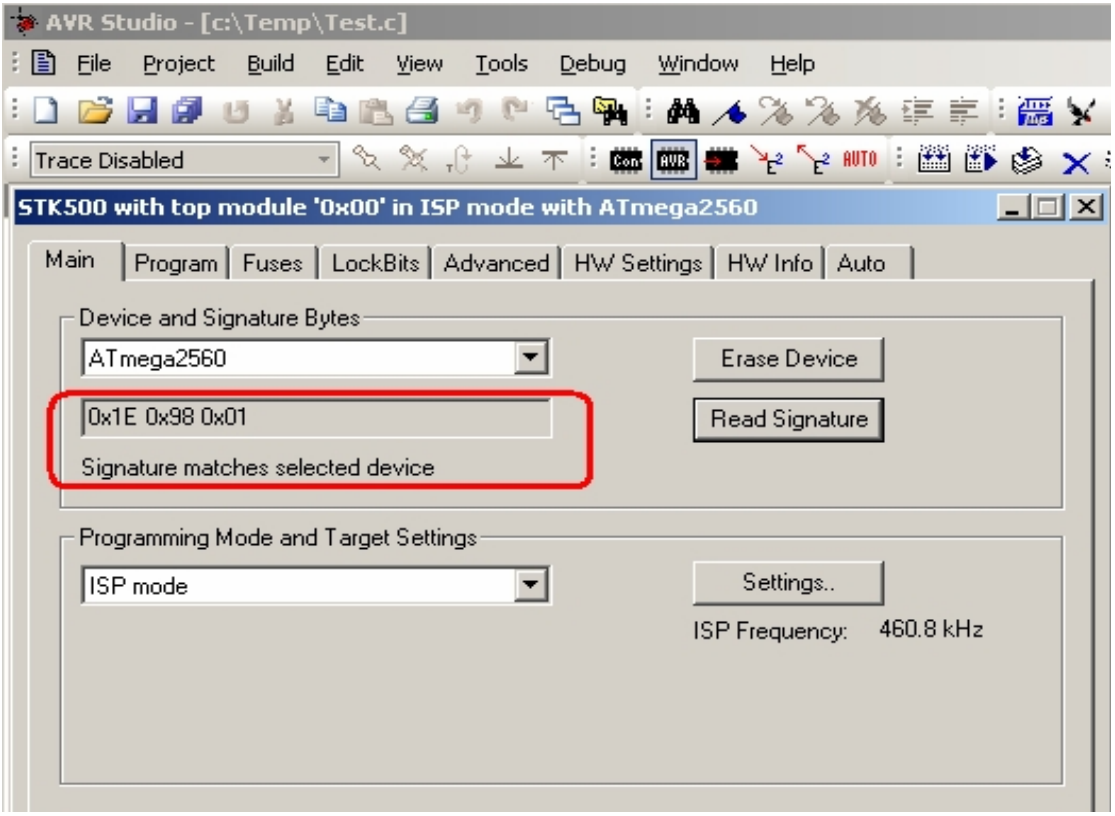
1.6) To download the compiled Test.hex firmware to the board, please click the Con icon button on the tool bar.



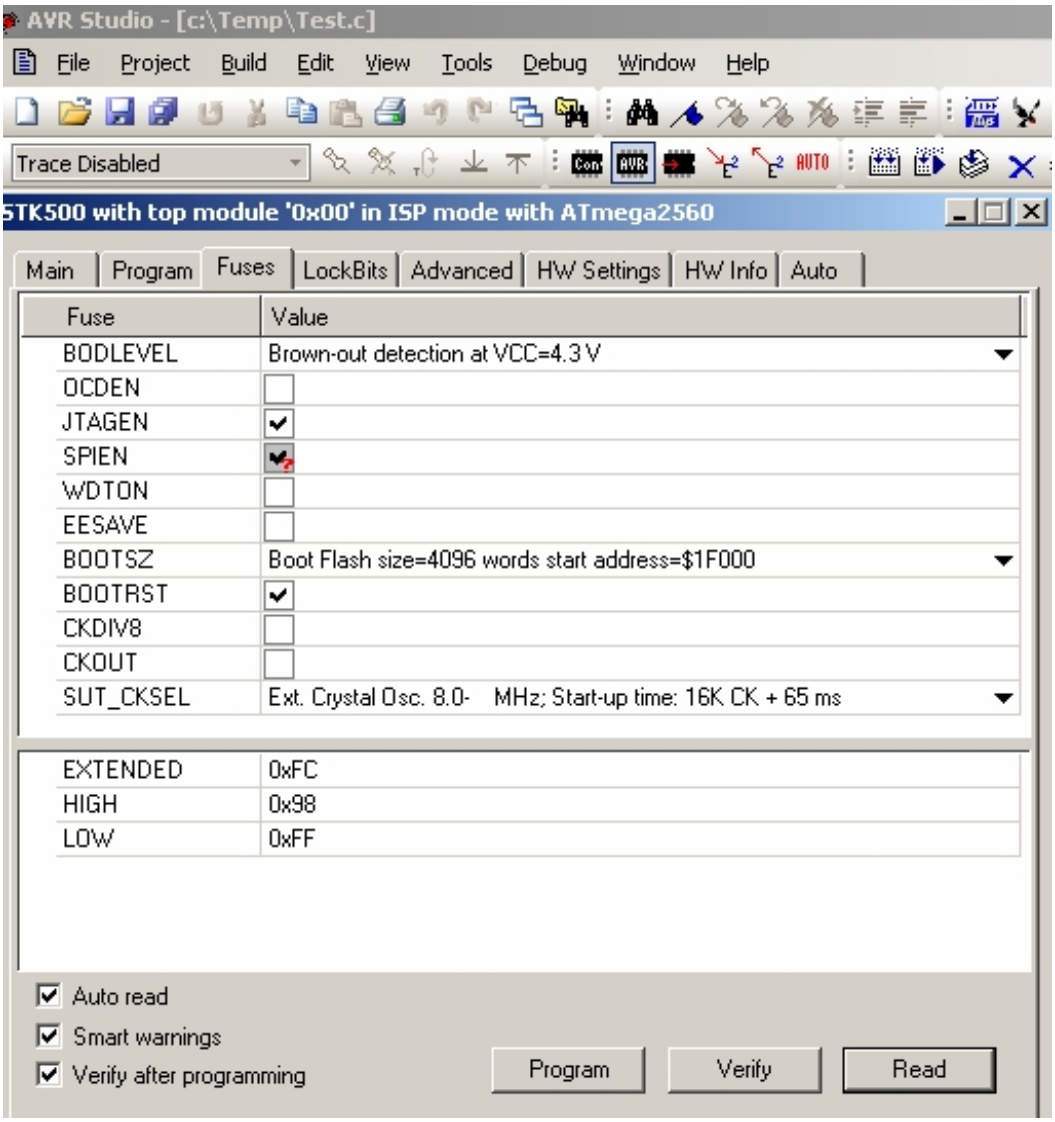
1.7) Select STK500 platform, select Auto, and then click the Connect button.



1.8) Select ATmega2560 device and click the Read Signature button. The signature should match ATMEGA2560.

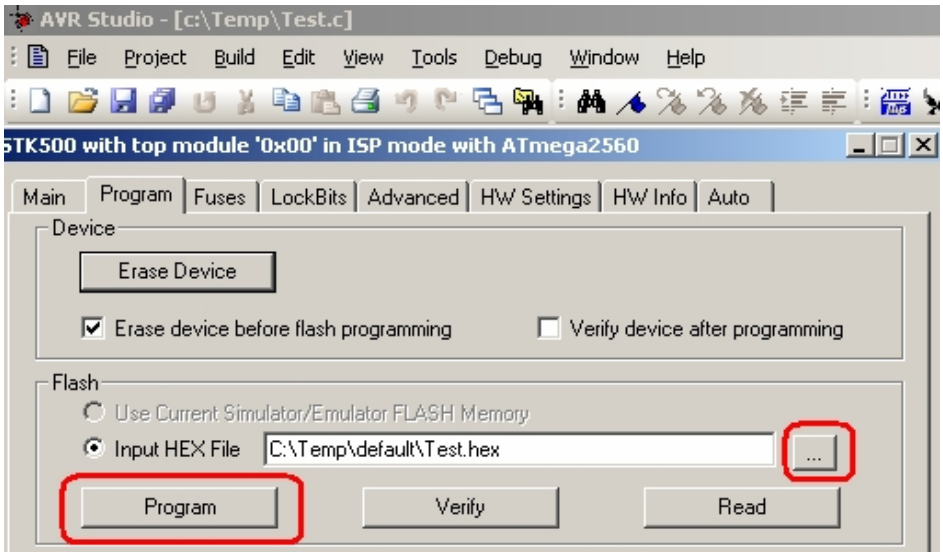


1.9) Check fuses

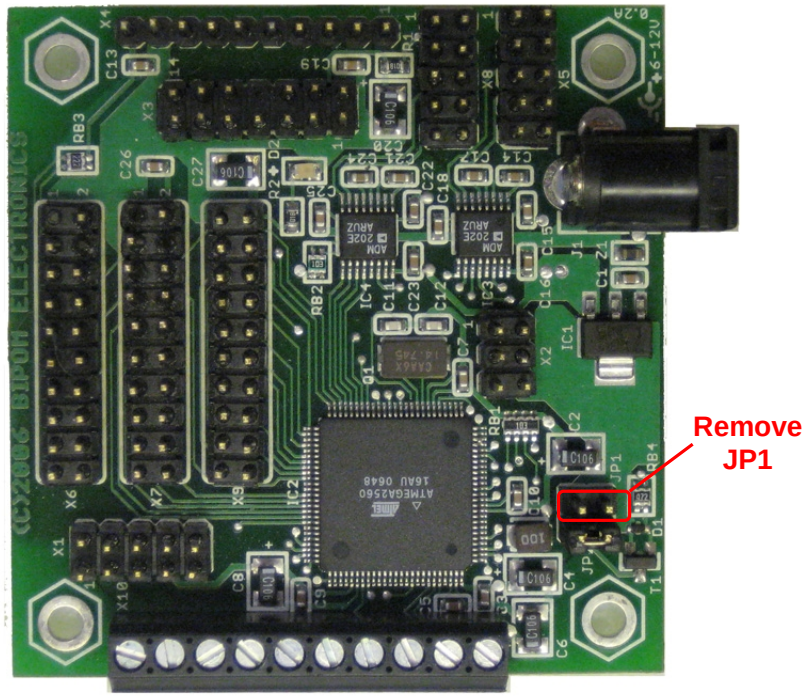


The fuses cannot be changed using the serial boot loader. To do that it is necessary to use a real programmer such as AVR ISP or AVR Dragon.

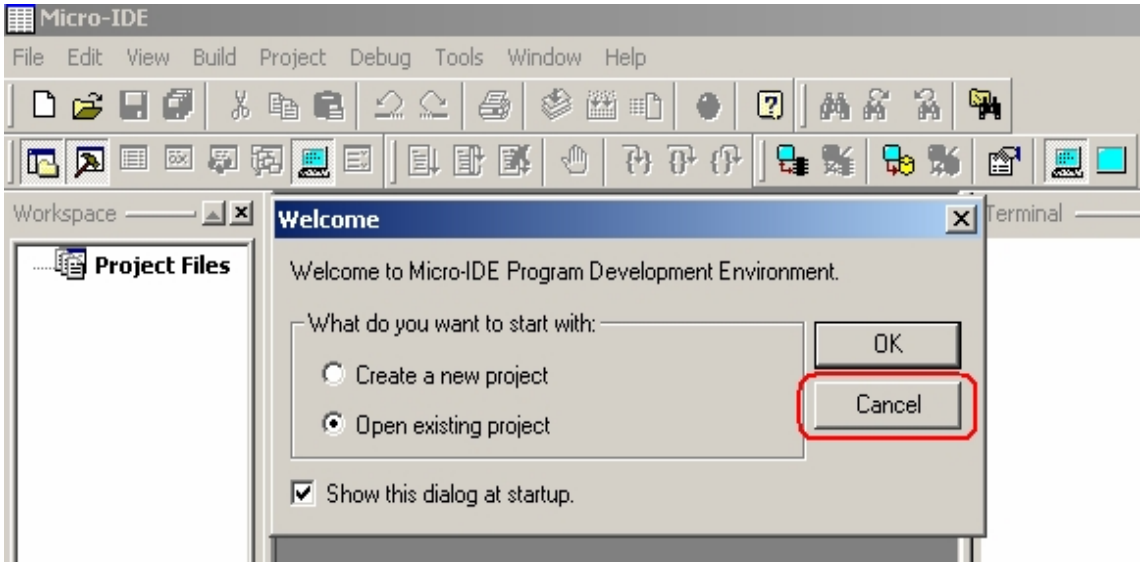
1.10) Select Test.hex file and click the Program button



1.11) Close AVR Studio dialog to release the PC COM port.
1.12) Turn off power to the MINI-MAX/AVR-C. Remove the JP1 jumper.



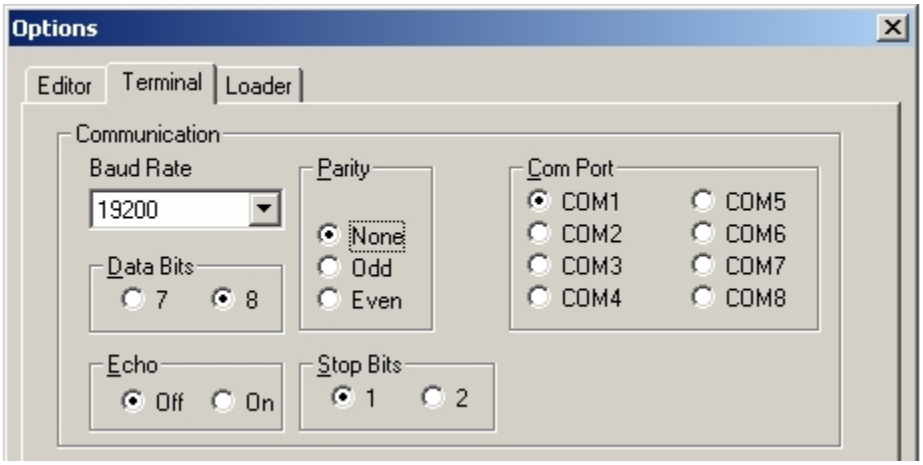
1.13) To see the 'Hello World!' messages that the board sends to the serial port, Micro-DE terminal window is used. This is because the AVR Studio does not have its own terminal window. Run Micro-IDE from Start->Programs->Micro-IDE.



Click the Cancel button.

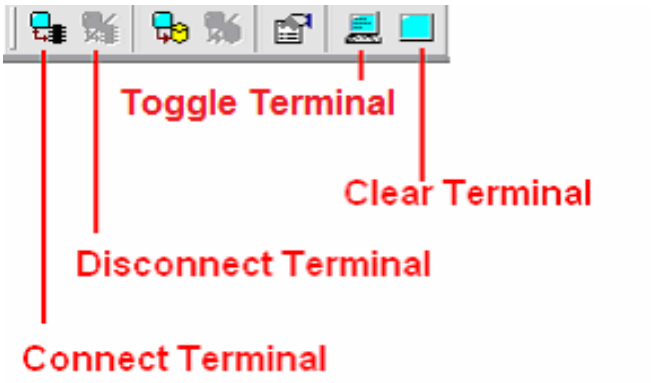
We don't need to create any project. We need only a terminal window. Instead of Micro-IDE, you can also use any other terminal program that can receive messages through a COM port (for example, HyperTerminal).

1.14) To specify the correct terminal settings please select Tools->Options menu:



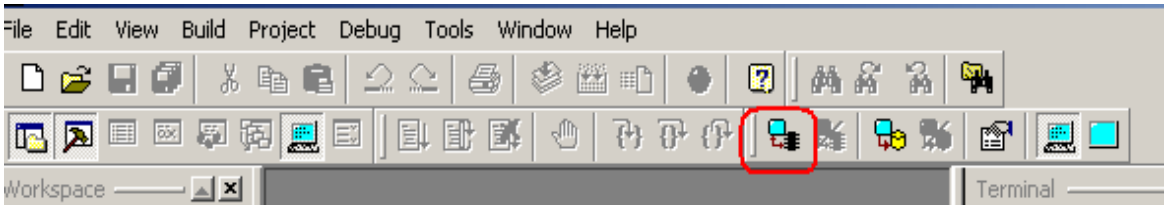
Select the correct PC COM port you have connected the MINI-MAX/AVR-C.
The following settings match the example we run on the Mini-Max/AVR-C board:
Baud rate: 19200
Parity: None
Data Bits: 8
Stop bits: 1
Echo: Off
Click the OK button.

1.15) Open the terminal window using the Toggle Terminal icon button

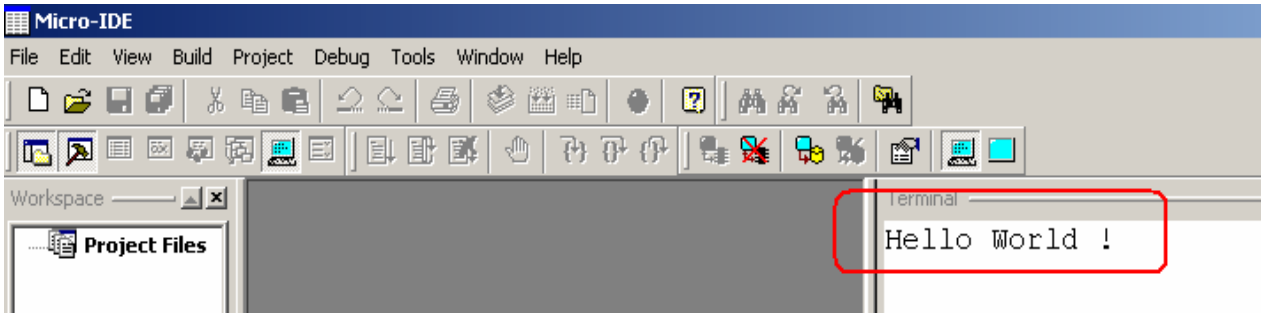


Connect Terminal connects the terminal window to the PC COM port. If a board sends data to the serial port, the messages will appear in Terminal window.
Disconnect Terminal disconnects the terminal window from the PC COM port.
Toggle Terminal shows/hides the terminal window.
Clear Terminal clears all messages in the terminal window.

1.16) Click the Connect icon button to connect the terminal window to the board.



1.17) Power the board. The “Hello World!” message appears in the terminal window.



Congratulations!!! You have created and executed your first program on the MINI-MAX/AVR-C. ☺

LAB2 – Input/Output

Overview

This lab will familiarize you with the AVR I/O Module included in AVR Training Kit for monitoring and control all I/O ports of the microcontroller.

Information

AVR I/O Module allows access to all input/output (I/O) ports of the AVR micro-controller on the MicroTRAK development platform. AVR I/O Module has 32 switches to control the AVR micro-controller inputs and 32 LED's to indicate the port statuses as logic LOW or logic HIGH.

Signal	LED	Switch		Signal	LED	Switch
LD4	HL1	S1-1		IO1	HL17	S2-1
READ	HL3	S1-2		IO0	HL19	S2-2
STROBE	HL5	S1-3		IO3	HL21	S2-3
N/C	HL7	S1-4		IO2	HL23	S2-4
LD0	HL9	S1-5		IO5	HL25	S2-5
LD1	HL11	S1-6		IO4	HL27	S2-6
LD2	HL13	S1-7		I2C SCL	HL29	S2-7
LD3	HL15	S1-8		I2C SDA	HL31	S2-8

Signal	LED	Switch		Signal	LED	Switch
KEY0	HL2	S3-1		/RXD2	HL18	S4-1
KEY1	HL4	S3-2		/TXD2	HL20	S4-2
KEY2	HL6	S3-3		IO6	HL22	S4-3
KEY3	HL8	S3-4		MISO	HL24	S4-4
KEY4	HL10	S3-5		SCK	HL26	S4-5
KEY5	HL12	S3-6		SS	HL28	S4-6
KEY6	HL14	S3-7		IO22	HL30	S4-7
KEY7	HL16	S3-8		MOSI	HL32	S4-8

Table 1.

Exercise

Write a C program that turns on consistently all LED's connected to the corresponding ports and reads status on the corresponding switches.

NOTE: Signal with Switch S1-4 is NOT CONNECTED!

LAB3 – Traffic Light

Overview

The purpose of this lab is to control the outputs of the micro-controller in a given sequence. Green, yellow and red Light Emitting Diodes (LED's) on the TB-1 board are connected to micro-controller outputs.

First, write a program to turn on only one LED and then turn off the same LED, Then, improve the program make the LED blink.

The next part of the lab is to read the input switches. As switches are mechanical objects, some de-bounce time (dead time) will also be placed in the program. You will control an LED with one switch; as long as switch is active the respective LED is On and when switch is inactive, the corresponding LED is Off. Then, the LED will be made to blink as long as switch is On. Finally, you will write a little traffic light controller where the green, red, yellow LED's on the TB-1 board simulate the traffic lights and the switches simulate the car presence sensors at a crossroad.

Information

Most of the micro-controller pins and the power supply are available on the 20-pin connector (X6) for interfacing to external circuitry, prototyping boards and peripheral boards, including the TB-1 Training Board. Table 2 shows the X6 pin-out of the micro-controller pins.

MINI-MAX/AVR-C Expansion (X6)

Signal	Pin	Pin	Signal
/RXD2	20	19	/TXD2
IO6	18	17	MISO
SCK	16	15	SS
IO22	14	13	MOSI
IO1	12	11	IO0
IO3	10	9	IO2
IO5	8	7	IO4
I2C SCL	6	5	I2C SDA
VCC	4	3	GND
VCC	2	1	GND

Table 2.

The ports of the micro-controller are accessible via the X6 connector. TB-1, which is connected to the micro-controller board as explained in Lab1, will be used as it already has input switches and output LED's connected. Table 3 describes the pin-out on the X6 connector and their designation as respective to input (switches) or output (LED's).

Micro-controller Pin	Connector on X6	Input or Output
IO6	18	Input, active Low, SW1
MISO	17	Input, active Low, SW2
IO2	9	Output, active High, Red LED
IO0	11	Output, active High, Yellow LED
IO1	12	Output, active High, Green LED

Table 3.

The term active low indicates that the micro-controller must be programmed in a manner such that the switch is made or activated when IO6 or MISO read low logic. Correspondingly the term active high means that to activate an LED the corresponding micro-controller pin must generate high logic. The port directional register defines the state (input or output) of the micro-controller I/O lines.

Exercise

Initially turn the individual LED's On and Off without interaction with the switches. Write a program to:

- Turn Red LED On only for a brief time interval and than Off
- Turn Yellow LED On only for a brief time interval and than Off
- Turn Green LED On only for a brief time interval and than Off
- Blink Red LED only
- Blink all three LED's

You now have the capability of controlling more than one output line. Next step is to read an input line. When the Switch is not activated the respective line is high and an LED should be turned On. When the Switch is activated, blink an LED:

- Turn Red Led On if Switch 1 is inactive.
- Turn Yellow Led On if Switch 2 is inactive.
- Blink Red Led as long as Switch 1 is active and Red Led is On if Switch 1 is inactive.
- Blink Yellow Led as long as Switch 2 is active and Yellow Led is On if Switch 2 is inactive.

Write a program to generate the results shown in Table 4 for a traffic light controller:

Switch 1	Switch 2	Red LED	Yellow LED	Green LED
Open	Open	OFF	OFF	ON
Open	Closed	ON	ON	OFF
Closed	Open	OFF	OFF	ON
Closed	Closed	OFF	OFF	OFF

Table 4.

Build the program and download to the Training Board. Trace through the program and watch the LED's as you step through each instruction. Open and close Switch 1 and Switch 2 to change the LED's to make sure that your program is handling every case correctly.

LAB4 – Analog To Digital Conversion on TB-1

Overview

This lab is used to familiarize you with the operation of an A/D converter. A varying voltage is applied to the input of the A/D converter by using a potentiometer and its output is read into the micro-controller and displayed on the terminal screen. The A/D and the potentiometer are connected the Analog Input terminal blocks of TB-1. LM35 type temperature sensor will then be connected to the A/D and temperature values will be displayed on the board.

Information

Voltages that continuously vary as a function of time are defined as analog voltages and can have any value within certain range, e.g. 0V to 5V, or –5V to + 5V, or 0V to +12V etc. Digital voltages only have two values i.e. a 1 or a 0. An A/D converter (Analog to Digital converter) writes the value of the analog voltage into digital format or code that is then processed by the micro-controller.

An 8-bit A/D converter, ADC0834 from National Semiconductor as shown in Figure 5 is connected on the TB-1 board.

The pin-outs for this 14 pin IC (ADC0834) is shown in Table 5.

Pin#	Mnemonic	Description
1	V+	
2	CS	Active low
3	CH0	Analog Input Channel 0, 0V to 2.5V DC
4	CH1	Analog Input Channel 1, 0V to 2.5V DC
5	CH2	Analog Input Channel 2, 0V to 2.5V DC
6	CH3	Analog Input Channel 3, 0V to 2.5V DC
7	DGND	Digital Ground.
8	AGND	Analog Ground
9	Vref	Reference voltage for the A/D, 2.5V DC
10	DO	Output Signal generated by the A/D
11	SARS	Output, when high A/D in progress, when low data output
12	CLK	Input, Clock signal to the A/D
13	DI	Input, Data to the A/D
14	VCC	Input, Power to the A/D

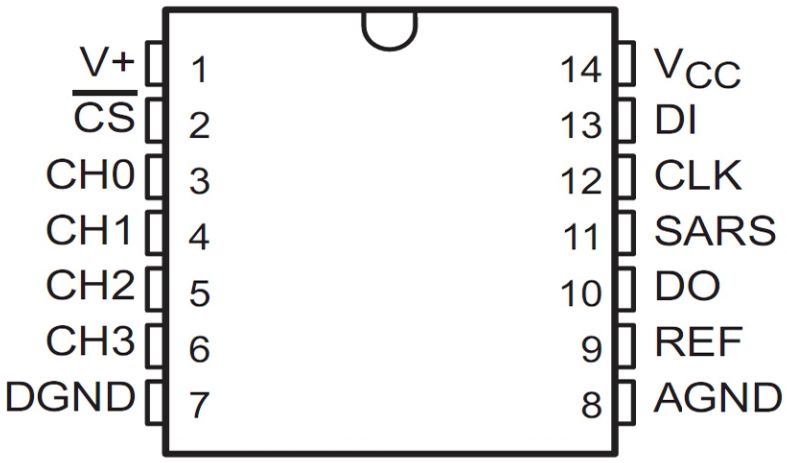
Table 5.

Table 6 shows the connections of the 14 lines of the A/D to the J3 connector.

X6 Pin#	Micro-controller Pin#	A/D Mnemonic	A/D Pin#
14	IO22	CS	2
5	I2C SDA	DO	10
N/C	N/C	SARS	11
6	I2C SCL	CLK	12
13	MOSI	DI	13

Table 6.

ADC0834 4-Channel MUX
Small Outline/Dual-In-Line Package
(WM and N)



COM internally connected to AGND

Top View

Figure 5.

ADC0834 Timing

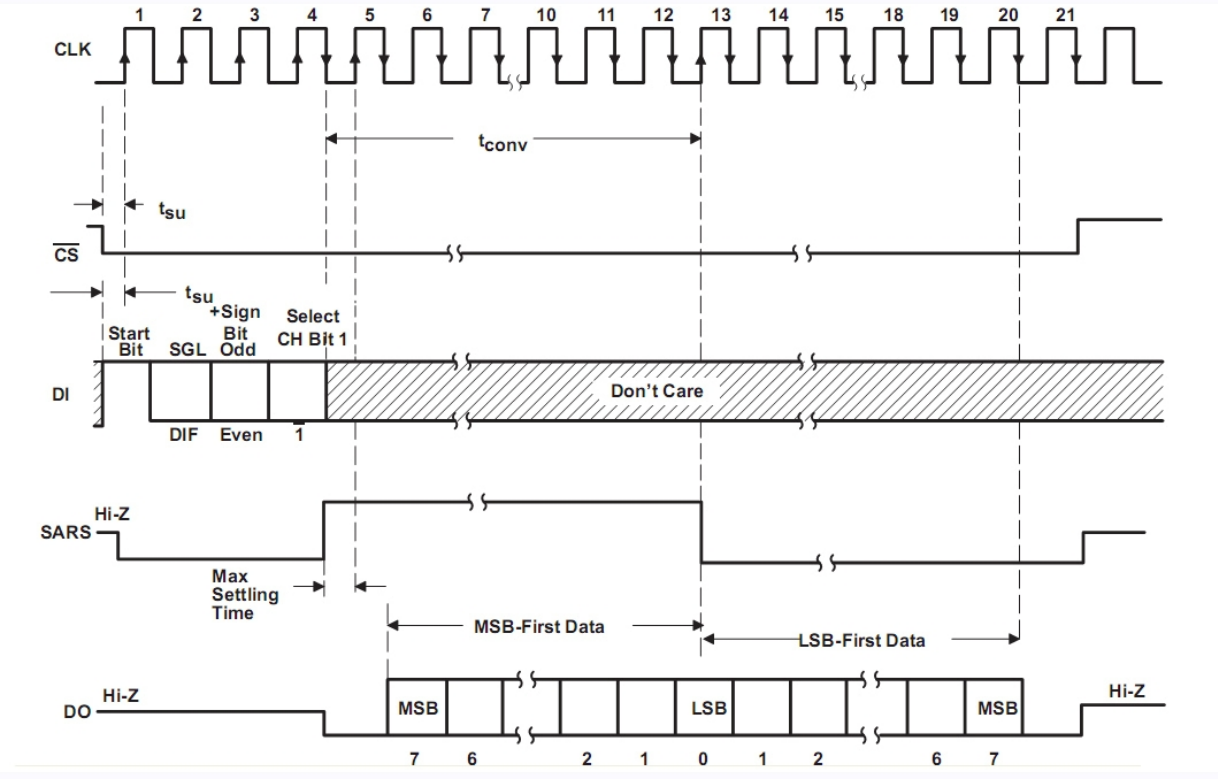


Figure 6

Single-Ended MUX Mode

MUX Address			Channel #			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
1	0	0	+			
1	0	1			+	
1	1	0		+		
1	1	1				+

Differential MUX Mode

MUX Address			Channel #			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
0	0	0	+	−		
0	0	1			+	−
0	1	0	−	+		
0	1	1			−	+

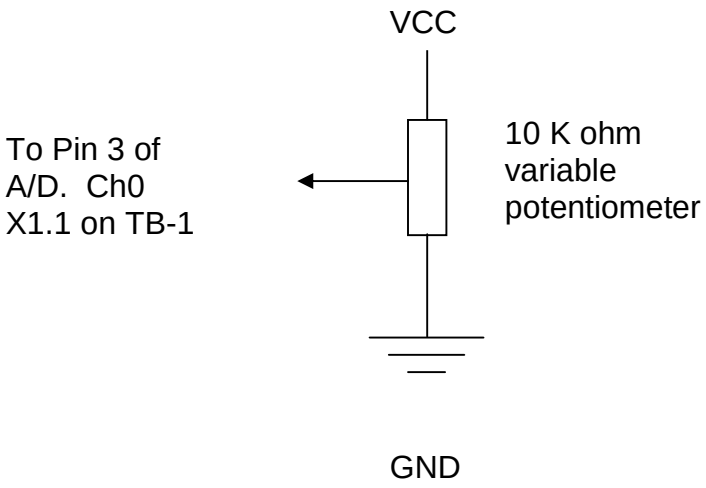


Figure 7.

Pin 3 of the A/D receives the Analog input voltage signal. A DC Voltmeter when placed on Pin 3 should read between VCC and GND as the potentiometer is varied from one extreme to the other as shown in Figure 7. The LM35 temperature sensor will also be connected to this input later.

The A/D converter consists of 4-input multiplexed analog channels, which may be software configured as 4 single-ended channels or 2 differential channels, or a new pseudo differential option. The input format is assigned during MUX addressing sequence prior to start of conversion and this selects the analog input and their mode (single or differential).

Steps for A/D Conversion

- Initially the DI and CS inputs must be high.
- Pull the CS (chip select) line low and must be held low for the entire duration of the conversion. The converter is waiting for the Start bit and the MUX assignment.
- A clock is generated by the processor and output to the A/D clock input.
- The start bit is a logic “1” on the DI input line, after which the MUX assignment word on the DI line is presented. The status of the DI line is clocked on each rising edge of the clock line.
- The SARS status output line from the A/D goes high, indicating that a conversion is now in progress and the DI line is disabled.
- The data out (DO) line comes out of Tri-state and provides a leading zero for one clock period.
- After 8 clock periods the conversion is complete. The SARS line returns low ½ clock cycle later.
- The DO line now generates the binary equivalent of the analog value as 8 output bits with LSB first. The DO line than goes low and remains low until chip select (CS) is made to go high. This clears all the internal registers and another conversion may now be started by making the CS go low again and repeating the process.

Exercise

Write a C program that uses the A/D in single ended mode with the potentiometer connected to Channel 0 or Pin3 on the A/D through the terminal block on the TB-1 board. Vary the analog input voltage and display the value on the screen. Then, connect the potentiometer to Pin4 also and display its value. Connect the temperature sensor to Pin3 and potentiometer to Pin4 and display both the values simultaneously on the terminal screen.

LAB5 – Analog To Digital Conversion on Mini-Max/AVR-C

Overview

The purpose of this lab is to familiarize you with the 10-bit ADC on ATMEGA2560 on MINI-MAX/AVR-C board. The potentiometer (that we used in Lab4) is connected to Analog Port Terminal on MINI-MAX/AVR-C. Measurement results for all 5 channels will be displayed in the terminal screen.

Information

The micro-controller ATMEGA2560 on MINI-MAX/AVR-C board can provide 16-channel 10-bit ADC The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 1.1V or 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

A/D Control and Status Register ADCSRA description

Bit	Description
7	ADEN: ADC Enable Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate the conversion.
6	ADCS: ADC Start Conversion In Single Conversion mode, write this bit to 1 to start each conversion.
5	ADATE: ADC Auto Trigger Enable When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.
4	ADIF: ADC Interrupt Flag This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.
3	ADIE: ADC Interrupt Enable When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.
2:0	ADPS2:0: ADC Prescaler Select Bits These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 7.

The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX5 and MUX2:0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 8. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the Analog Comparator.

ACME	ADEN	MUX5	MUX2:0	Analog Input
1	0	0	000	ADC0
1	0	0	001	ADC1
1	0	0	010	ADC2
1	0	0	011	ADC3
1	0	0	100	ADC4
1	0	0	101	ADC5
1	0	0	110	ADC6
1	0	0	111	ADC7
1	0	1	000	ADC8
1	0	1	001	ADC9
1	0	1	010	ADC10
1	0	1	011	ADC11
1	0	1	100	ADC12
1	0	1	101	ADC13
1	0	1	110	ADC14
1	0	1	111	ADC15

Table 8.

Exercise

Write a C program that uses built-in 10-bit ADC on ATMEGA2560 on MINI-MAX/AVR-C board, reads all 5 channels and displays their values on the terminal screen. Connect the potentiometer consistently to the each of analog inputs, vary the analog input voltage and observe results.

LAB6 – Timers and Interrupts

Overview

This lab is designed to familiarize the student with timers and interrupts. External interrupts on switch closures are generated and back ground timers are made to run for timing certain external events. The switch and the LED's for this experiment are already present on the TB-1 board. Results will be displayed on the terminal screen.

Information

The AVR training kit uses the ATMEGA2560 micro-controller. A brief overview of the interrupts on this micro-controller is given below:

An interrupt is executed when the micro-controller stops its normal code and branches to a predefined section of the memory to execute specific instructions. After this it goes back to its normal operation from where it had left before. A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register(SREG). All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

INT0:

First exercise is to generate an external interrupt INT0 that is tied to SCL (PortD.0) (S2-7 on AVR-IO). A change in the logic level will generate an interrupt in the program. Below defines the registers and the bits that are used for controlling pin change interrupt INT0.

Register	BIT	Description
PCICR	PCIE0	When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7:0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT7:0 pins are enabled individually by the PCMSK0 Register.
PCIFR	INTF2	If the I-bit in SREG and the corresponding interrupt enable bit, INT2 in EIMSK, are set (one), the MCU will jump to the interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Table 9.

Every time S2-7 is pressed 7 on AVR-IO (so that port pin SCL on the micro-controller is logic low), the micro-controller will generate an interrupt and an action will occur, e.g. incrementing a counter and displaying its value on the terminal screen.

Timer0

Register TCCR is the timer counter mode control register and the TCNT register turns the timer On/Off and TIFR indicates an overflow by setting a flag bit.

Register	Bit	Description
TCNT	TR0	Set indicates Timer0 is On, clear indicates Timer0 is Off
TIMSK	ICIE0	When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter Input Capture interrupt is enabled. The corresponding Interrupt is executed when the ICF Flag, located in TIFR, is set.
TIFR	TOV0	The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter0 Overflow interrupt is executed.

Table 10.

The TMOD register enables the user to select different modes of operation for Timer 0 and Timer 1. Write a program to run a background timer of 2 milliseconds. Every time the 2-millisecond period is over, the program will jump to an interrupt routine and toggle the state of an I/O (P1.3 Red LED) line so the Red LED will blink every seconds.

The same can be done with Timer1 where by TR1 and TF1 are used.

IE Register

Bit #	Mnemonic	Description
IE.7 (MSB)	EA	Global Interrupt Enable.
IE.6	------	Not Implemented
IE.5	ET2	Timer2 interrupt enable bit.
IE.4	ES	Serial Port interrupt enable
IE.3	ET1	Timer1 interrupt enable bit
IE.2	EX1	External Interrupt 1 enable bit
IE.1	ET0	Timer0 interrupt enable bit
IE.0 (LSB)	EX0	External Interrupt 0 enable bit

Table 11.

Enable Bit = 1 enables the interrupt
Enable Bit = 0 disables the interrupt

TCON Register

Bit #	Mnemonic	Description
TCON.7 (MSB)	TF1	Timer1 Overflow flag
TCON.6	TR1	Set indicates Timer1 is On and Clear indicates Timer1 is Off
TCON.5	TF0	Timer0 Overflow flag
TCON.4	TR0	Set indicates Timer0 is On and Clear indicates Timer0 is Off
TCON.3	IE1	Set by micro-controller when Timer1 interrupt occurs and cleared by micro-controller when interrupt is processed
TCON.2	IT1	Set than low level triggered, clear than edge triggered
TCON.1	IE0	Set by micro-controller when Timer0 interrupt occurs and cleared by micro-controller when interrupt is processed
TCON.0 (LSB)	IT0	Set than low level triggered, clear than edge triggered

Table 12.

LAB7 - 4x4 Keypad

Overview

Student connects a keypad to the keypad connector on the MicroTRAK Carrier Board and programs the keypad.

Information

The MicroTRAK Carrier Board has a 10-pin keypad connector, which is connected to Port 2 of the micro-controller. The pin-out of the connector is shown below in Table 13.

Micro-controller Port Pin	Keypad Connector Pin#
KEY0	1
KEY1	2
KEY2	3
KEY3	4
KEY4	5
KEY5	6
KEY6	7
KEY7	8
Gnd	9
Vcc	10

Table 13.

Many keypads are wired as a matrix of rows and columns. The internal connections of a 4 -row by 4-column keypad are shown in Figure 8.

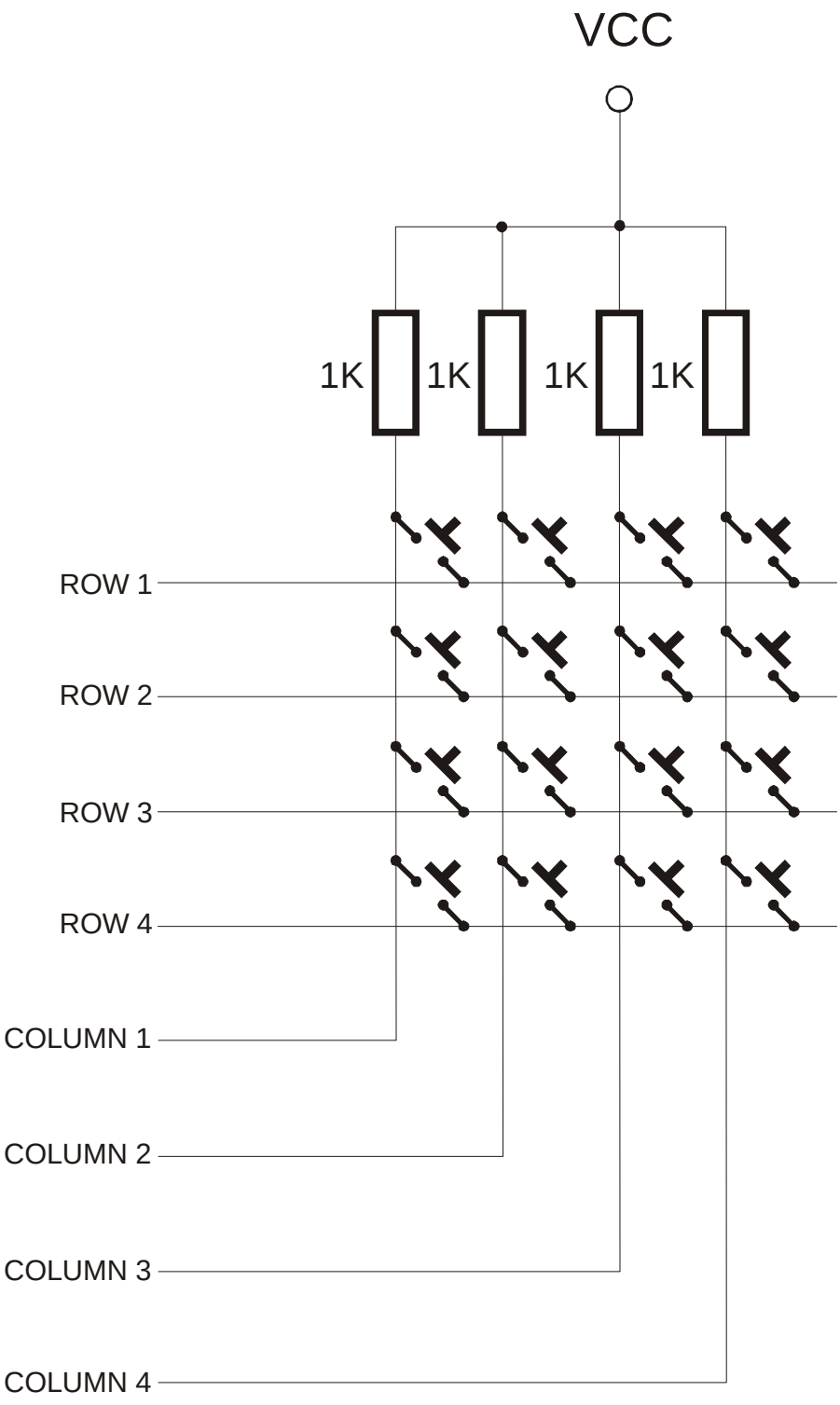

VCC
1K
1K
1K
1K
ROW 1
ROW 2
ROW 3
ROW 4
COLUMN 1
COLUMN 2
COLUMN 3
COLUMN 4

Figure 8.

Matrix connection saves on the number of connections and micro-controller port lines. For example, a 4-row by 4-column keypad would require 17 wires (16 + ground) if each key was individually connected to micro-controller ports. Using the matrix approach and scanning the keypad under software control reduces the number of wires and port pins to 8 (4 rows + 4 columns).

When a key is pressed, the row for that key will be physically connected to the column for that key. Therefore, the port input for the column will be at the same logic level as the port output for the row. Since the columns (inputs) are normally at the HIGH logic level due to pull-up resistors, the only way to make a column LOW will be to press a key and make the row for that key LOW. By periodically strobing each row LOW one row at a time, and reading the column input levels during each strobe, one can determine which key is pressed.

This is illustrated by Table 14 for the 4 by 4 keypad. In the Row Mask, Row 1 is assigned to the Most Significant Bit and Row 4 is assigned to the Least Significant Bit. Similarly in the Column Mask, Column 1 is assigned to the Most Significant Bit and Column 4 is assigned to the Least Significant Bit.

Action	**Row Mask**	**Column Mask**
No keys were pressed	XXXX	1111
Row 1 Column 1 key pressed	0111	0111
Row 1 Column 2 key pressed	0111	1011
Row 1 Column 3 key pressed	0111	1101
Row 1 Column 4 key pressed	0111	1110
Row 2 Column 1 key pressed	1011	0111
Row 2 Column 2 key pressed	1011	1011
Row 2 Column 3 key pressed	1011	1101
Row 2 Column 4 key pressed	1011	1110
Row 3 Column 1 key pressed	1101	0111
Row 3 Column 2 key pressed	1101	1011
Row 3 Column 3 key pressed	1101	1101
Row 3 Column 4 key pressed	1101	1110
Row 4 Column 1 key pressed	1110	0111
Row 4 Column 2 key pressed	1110	1011
Row 4 Column 3 key pressed	1110	1101
Row 4 Column 4 key pressed	1110	1110

Table 14.

Port2 is connected to the Keypad connector with the configuration shown in Table 15.

KEY0	Row1	Output
KEY1	Row2	Output
KEY2	Row3	Output
KEY3	Row4	Output
KEY4	Column 1	Input
KEY5	Column 2	Input
KEY6	Column 3	Input
KEY7	Column 4	Input

Table 15.

In the port direction register, the port pins connected to rows are defined as outputs and the port pins connected to columns are defined as inputs. Each key on the keypad is assigned a given value by the programmer before hand.

The Keypad algorithm can be based on the following rules.

Algorithm

- A Column is generally high (output).
- One row at a time is made to go low (input) and than the columns are read.
- If one or more columns are low than the switches of the corresponding columns are active and their respective values should be displayed on the terminal.

Exercise

Determine the pin-out and the matrix layout of your keypad using the ohmmeter function of your multi-meter. Write a C program that displays – on the terminal screen – the key being pressed on the keypad.

LAB8 – Liquid Crystal Display (LCD)

Overview

This Lab familiarizes the student with industry-standard alphanumeric LCD's by connecting the LCD to the LCD connector of the AVR training kit and writing a program in C to display various characters using 4-bit mode.

Information

Dot matrix LCD displays are readily available from many companies such as Sharp, Hitachi and OPTREX. The LCD's generally come in display formats of 16 X 1, 16 X 2, 24 X 2 and 40 X 4 (column X row). These typically have 8 data lines DB0 – DB7 (Data 0 through Data 7), VCC (Power), GND (Ground), RS (Register Select), R/W (Read/Write), E (Enable).

This exercise will use a 24 X 2 display, to be connected to the LCD connector. The three control lines are explained below.

RS Register Select Control
1 = LCD in data mode
0 = LCD in command mode

E Data / Control state
Rising Edge = Latches control state
Falling Edge = Latches data

R/W Read / Write control
1 = LCD to write data
0 = LCD to read data

In the 4-bit mode, data is transferred either on the lower or upper nibble of the port, this saves in I/O lines but the program occupies more space as two commands are required to display a character.

Table 16. shows the connection between the display and the LCD connector in 4-bit mode with low nibble.

LCD Connector Pin #	Micro-controller Pin	LCD Display Pin	LCD Display Pin #
1	GND	GND	1
2	VCC	VCC	2
3	Vee(V-PWM)	VEE	3
4	LD4 (PL4)	RS	4
5	READ (PL5)	R/W	5
6	STROBE (PL6)	E	6
7	N/C	N/C	7
8	N/C	N/C	8
9	N/C	N/C	9
10	N/C	N/C	10
11	LD0 (PL0)	DB4	11
12	LD1 (PL1)	DB5	12
13	LD2 (PL2)	DB6	13
14	LD3 (PL3)	DB7	14

Table 16.

Exercise

Using the LCD datasheet, write a C program that displays “Hello World” on the first row of the LCD. Then, display the same message on the second row of the LCD.

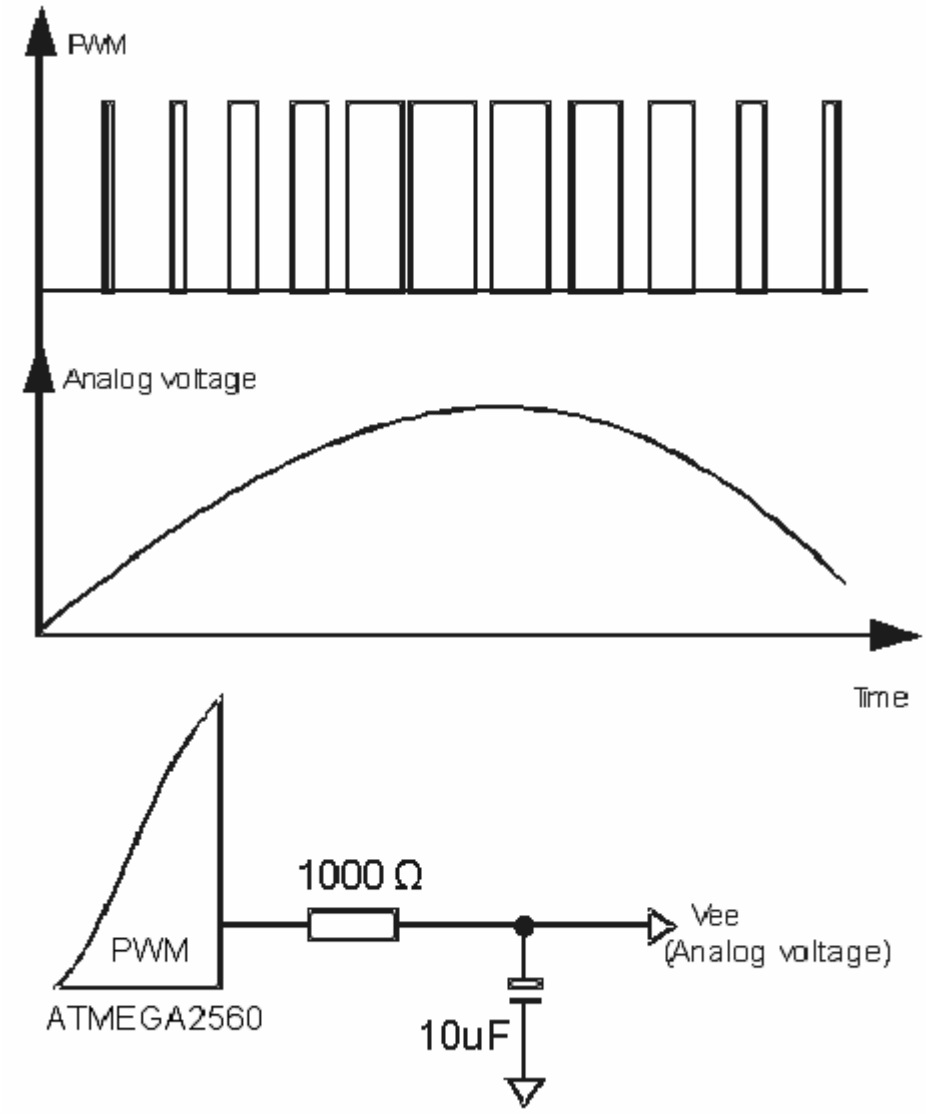
LAB9 - How to adjust LCD contrast

Overview

Student adjusts LCD contrast, using DAC embedded in ATMEGA2560 on MINI-MAX/AVR-C board.

Information

The contrast of the LCD module can be adjusted by means of the Vee Voltage (pin 3). DAC embedded in the ATMEGA2560 on the MINI-MAX/AVR-C is connected to Vee pin of the LCD. This enables software contrast adjusting. PWM (Pulse Width Modulation) output produce digital waveform with programmable duty cycle and to converted to an analog voltage, a low-pass filter is used (see figure below).



For 10 bits PWM you should to send as first byte highest 4 bits from needed 10 bit PWM value and after that you should send lowest 6 bits + 0x40 as second byte.

	10 bits PWM							
first byte	0	0	0	0	PWM9	PWM8	PWM7	PWM6
second byte	0	1	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0

Table 17.

Exercise

Write a C program that uses 4 bits PWM for contrast adjusting and displays on the first row of the LCD the current contrast. Expand the program with using 10 bits PWM.

LAB10 - Buzzer

Overview

Student programs the buzzer on the TB-1 to generate different notes using software. Student then generates a little musical piece using the notes that he/she programmed.

Information

A buzzer or simple speaker will generate music when a series of square waves is applied to its positive input with the negative grounded. The frequency of the square wave should be less than 12KHz to be audible. Varying the frequency of the square wave will generate different musical tones.

The buzzer on the TB-1 is connected to port pin IO3 (which maps to PortB.7) on the micro controller. This pin needs to be programmed as an output pin and square waves of varying frequencies and duty cycle need to be generated.

Exercise

- Program IO3 on the micro-controller as an output pin
- Decide a time period or frequency of square wave between 1KHz to 15KHz.
- Generate a signal of 50% Duty cycle, where
  Duty Cycle = On Time / (On Time + Off Time) of square wave.
- Activate buzzer for approx 5 to 10 seconds
- Change frequency of buzzer and note the different sound
- Change only duty cycle and note the different intensity.