Robotics experiment with PIC microcontroller

based-on Robo-PICA robot kit

3rd Edition

INNOVATIVE EXPERIMENT

Contents

Chapter 1

Part list of Robo-PICA and Introduce software tool

1.1 Robo-PICA part list

There are 2 groups :

1.1.1 Mechanical parts

1.1.2 Electronic parts

1.1.1 Mechanical parts

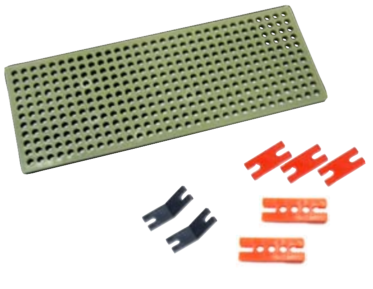
Motor Gearbox – Uses a 4.5V (9V max.) and 180 mA DC motor with a ratio of 48:1; torque 4kg/cm; comes with the mounting.

Track wheel set - includes 3-lengths of Track wheel, many support wheels and sprockets, axels and shaft bases

Many sizes of Screw and Nut (Screw : 3x6mm.,3x10mm., 3x15mm.,3x25mm. and 3x35mm., 3mm. nuts), Flat head screws and Thumb screws.

Set of Plastic Spacers (length : 3mm., 15mm. and 25 mm.)

Hex Standoffs : 3x30mm.

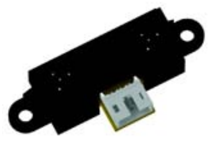


The Plate set and **4-types of the color-mixed Plastic Joiner** (10 of Straight Joiner, 10 of Right-angle Joiner, 10 of Obtuse Joiner and 3/5/12 Holes straight joiners)

1.1.2 Electronic parts

RBX-877V2.0 PIC16F887 Robot Experiment Board

ZX-01
Switch input
(x2)


GP2D120
4 to 30cm. Infrared
Distance sensor

ZX-03
Infrared Reflector
(x2)

ZX-IRM
38kHz Infrared Receiver

ER-4
Infrared
Remote Control

ZX-POTH
Potentiometer (x1)

USB Programmer board
with ICD2 cable

USB cable

4 of AA batteries
(Rechargable battery is
recommended
- not include this kit)

1.2 Tools for making the robot kit

Cutter plier

A sharp-tipped
hobby knife or
Handy Cutter

Philips Screwdriver

Computer
Install Windows98SE or
higher and has both RS-232
serial port and Parallel port

1.3 Software development tools for Robot programming

The RoboPICA kit uses the PIC Micrcontroller PIC16F887. Builders can write the controlled program in assembly, BASIC and C language. Only BASIC and C program language requires the use of a compiler software.

However in this kit all examples are in C language with mikroC compiler from mikroElektronika (mikroE : www.mikroe.com). The Robo-PICA robot kit can use this compiler as well.

The demo version of Mikro C compiler is used for this robot kit. Builders who need to develop the advance program will need to purchase the full version from MikroE at their webiste. The demo version of mikroC can be downloaded from http://www.mikroe.com. However in the Robo-PICA robot kit, this software is in the bundled CD-ROM. You must download the mikroC manual latest version from mikroElektronika website. This building manual does not describe all the instructions.

Another one tools is PIC microcontroller programmer software. The Robo-PICA provides a USB programmer. Its function is compatible Microchip's PICkit2™ programmer. The software can use PICkit2™ programming software. Free downlaod the latest version at www.microchip.com.

1.3.1 mikroC Compiler (Demo version)

1.3.1.1 Overview

mikroC is a powerful, feature rich development tool for PICmicros. It is designed to provide the customer with the easiest possible solution for developing applications for embedded systems, without compromising performance or control.

mikroC provides a successful match featuring highly advanced IDE, ANSI compliant compiler, broad set of hardware libraries, comprehensive documentation, and plenty of ready-to-run examples.

mikroC allows you to quickly develop and deploy complex applications:

- Write your C source code using the highly advanced Code Editor
- Use the included mikroC libraries to dramatically speed up the development: data acquisition, memory, displays, conversions, communications...

Special thanks : All information about mikroC Compiler and PICkit2 Programming software are referenced from owner website and documentation (www.mikroe.com and www.microchip.com). Thanks for all free and open-source developement tools. User who need the full features of mikroC compiler can purchase on-line at www.mikroe.com.

● Monitor your program structure, variables, and functions in the Code Explorer. Generate commented, human-readable assembly, and standard HEX compatible with all programmers.

● Inspect program flow and debug executable logic with the integrated Debugger. Get detailed reports and graphs on code statistics, assembly listing, calling tree...

● mikroE have provided plenty of examples for you to expand, develop, and use as building bricks in your projects.

● In Demo version, hex output is limited to 2k of program words.

1.3.1.2 Installation the mikroC compiler Demo version

Download the latest version from mikroElektronika website; www.mikroe.com. Run the installation file. Addition, you must download the 5 of necessary documentation files about compiler manual, Creating First Project in mikroC for PIC, Quick Reference Guide for C language, Compilers IDE document and Obtaining and Activating the License Key.

You can see all C syntax and all function details from the mikroC manual. In this manual would be describe about the robot activities only.

1.3.2 PICkit2™ Programming Software

The PICkit™ 2 Microcontroller Programming software is capable of programming most of Microchip's Flash microcontrollers. For specific products supported, see the README file or check with Microchip's website.

The full featured Windows programming interface supports baseline (PIC10F, PIC12F5xx, PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, and dsPIC33 families of 8-bit and 16-bit microcontrollers, and many Microchip Serial EEPROM products.

The PICkit™ 2 Microcontroller Programming software works with a PICkit2™ OEM USB programmer. The USB programmer is the in-system programming via ICD2 jack.

1.3.2.1 PICkit2™ Programming Software installation

1.3.2.1.1 Install from PX-200 CD-ROM

The working software of the USB programmer is PICkit2™ Programming Software. The newer version is developed from Microsoft.NET. Thus, user must install the Microsoft.NET Framework first.

(A) Install of the Microsoft .NET Framework

First thing to do is to install the Microsoft.NET Framework. Select from the folder **PICkit 2 Setup v2.01 dotNET → dotnetfx** in the bundled CD-ROM. Double-click at **dotnetfx.exe** file. After complete, install the PICkit2™ Programming Software by double-click at **PICkit2Setup.msi** file. The software installation will start.

(B) Microsoft .NET Framework is installed ready

User can install the PICkit2™ Programming Software by enter to folder **PICkit 2 Setup v2.01x** in the bundled CD-ROM of Robo-PICA kit. Double-click at **PICkit2Setup.msi** file. The software installation will start.

1.3.2.1.2 Install from the internet.

Visit the Microchip website at ***www.microchip.com***. Select ***Development tools*** webpage and enter to ***PICkit 2 Programmer/Debugger*** webpage.

(A) Install of the Microsoft .NET Framework

For user who have not install Microsoft .NET Framework, they will need to install it first via downloading the file from topic ***PICkit2V2.01 Install with .NET Frame-work***. You will get the **PICkit 2 Setup v2.01 dotNET.zip** file *(version number may vary)*. Extract this file and store it in the folder **PICkit 2 Setup v2.01 dotNET**. Enter to this folder and into the **dotnetfx** folder. Double-click at **dotnetfx.exe** file to start Microsoft .NET Framework installation. After this is completed, install the Pickit2™ Programming Software by double-clicking on the **PICkit2Setup.msi** file. THe software installation will start.

(B) Microsoft .NET Framework is installed ready

Users who have Microsoft .NET Framework already installed can download the setup file from ***PICkit2V2.01 Install*** header. You will get file **PICkit 2 Setup v2.01.zip** *(version number may be vary)* Extract this file and store in the folder **PICkit 2 Setup v2.01**. Enter to this folder and double-click on the **PICkit2Setup.msi** file to start the software installation.

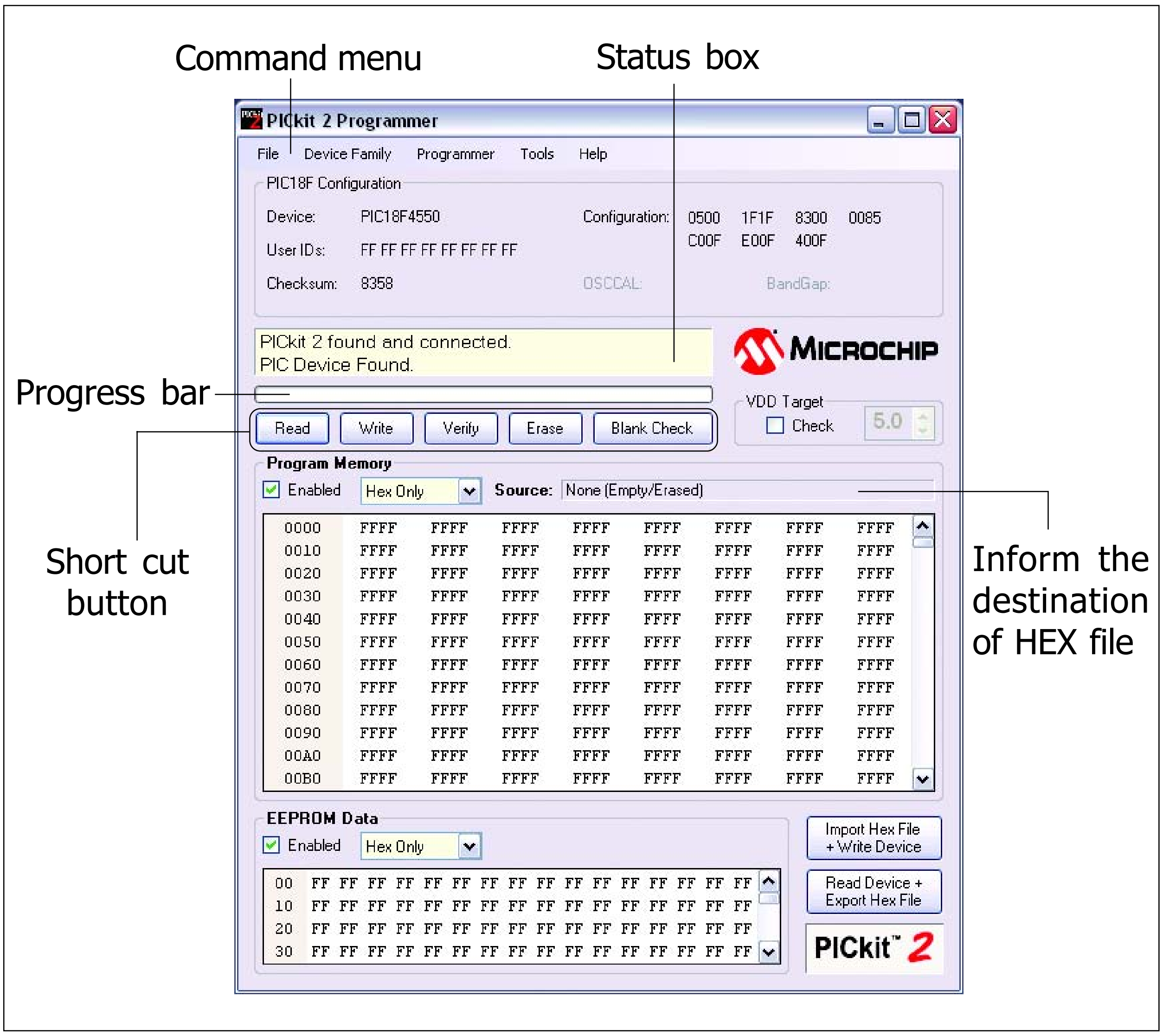
After run the installation setup file ; **PICkit2Setup.msi**. Click on the accept button on each step and follow the installation progress until it is finished.

1.3.2.2 Using PICkit2™ Programming Software

1.3.2.2.1 Testing hardware connection

(1) Connect the USB cable between the programmer and Computer's USB port. Open the software Pickit2™ Programming Software by entering the Start → All programs → Microchip → Pickit 2 V201. *The main window will appear as shown in figure 1-1*.

(2) On successful connnection, the message **PICkit 2 found and connected** will appear in the Status box.



PIC18F Configuration

Device: PIC18F4550

User IDs: FF FF FF FF FF FF FF FF

Checksum: 8358

Configuration: 0500 1F1F 8300 0085 C00F E00F 400F

OSCCAL:

BandGap:

PICkit 2 found and connected.
PIC Device Found.

VDD Target ☐ Check 5.0

Program Memory

☑ Enabled Hex Only Source: None (Empty/Erased)

```
0000  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0010  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0020  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0030  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0040  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0050  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0060  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0070  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0080  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
0090  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
00A0  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
00B0  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
```

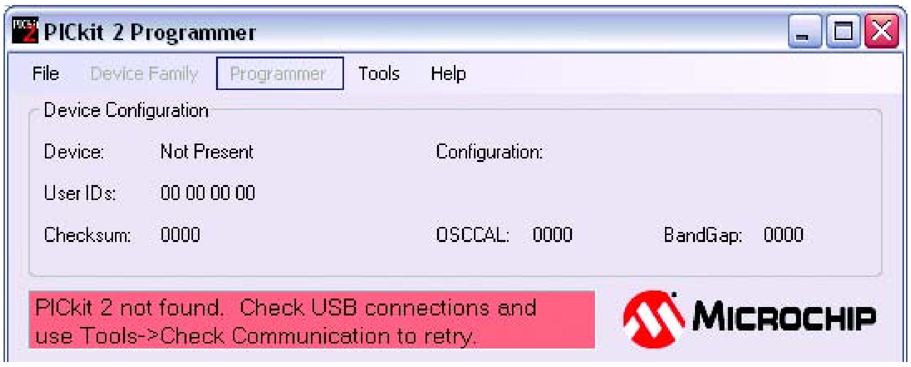
EEPROM Data

☑ Enabled Hex Only

```
00  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
30  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Figure 1-1 : The screen of Pickiit2TM Programming Software

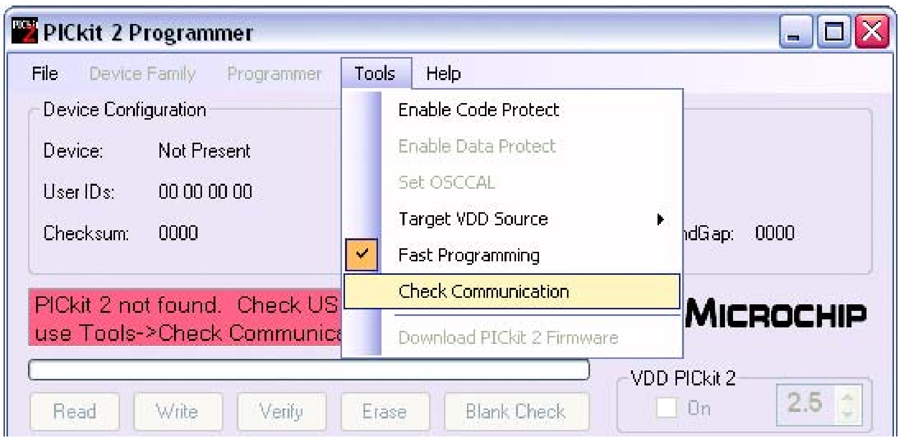
(3) If the connection is incompleted. The message **<u>PICkit 2 not found. Check USB connections and use Tools → Check Communication to retry</u>** will appear in the Status box. Check the cables and connections.



Device Configuration
Device: Not Present
Configuration:
User IDs: 00 00 00 00
Checksum: 0000
OSCCAL: 0000
BandGap: 0000

PICkit 2 not found. Check USB connections and use Tools->Check Communication to retry.

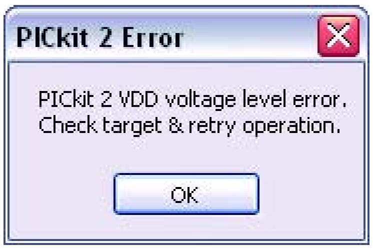
(4) Go to **Tools** menu and select **Check Communication** command. If all's correct, the message **<u>PICkit 2 found and connected</u>** will be show in the Status box.



Device Configuration
Device: Not Present
User IDs: 00 00 00 00
Checksum: 0000
dGap: 0000

PICkit 2 not found. Check US
use Tools->Check Communic

However if everytime during re-connection or checking hardware, it does not connect the target microcontroller at ICD2 jack and ICSP point or any mismatch in number, the warning dialog box will appear. It will warn you about any error supply voltage. You need not worry about this, click on the **OK** button to continue.



PICkit 2 VDD voltage level error.
Check target & retry operation.

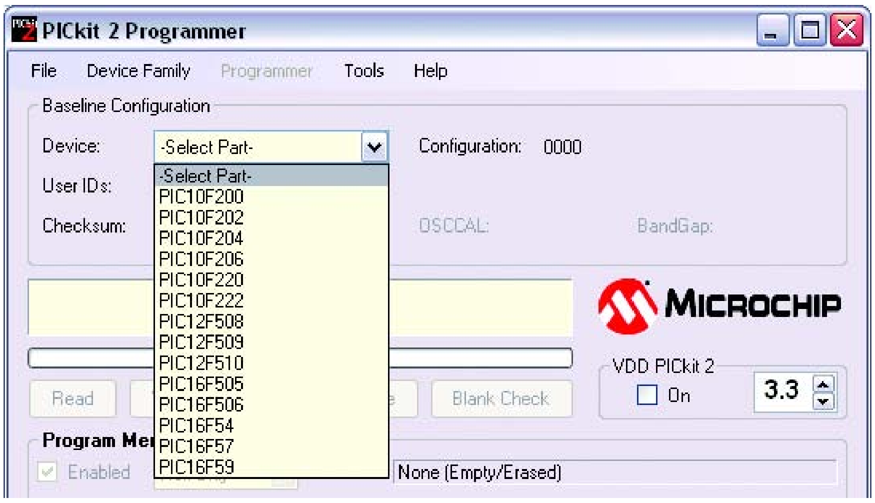
1.3.2.2.2 Command menu description

FILE

- **Import File** – Import a hex file for programming
- **Export File** – Export a hex file read from a device
- **Exit** – Exit the program (duplicated with the **Quit** button)

DEVICE FAMILY

- **Baseline** (12-bit Core) – Configures the programming software for baseline Flash devices



- **Mid-range** - Configures the programming software for 14-bit core flash devices. The devices in this range include *PIC12F6xx and 16F6xx, 7x, 7xx, 8x, 8xx*. When selected, software will check the connection target at ICD2 and ICSP terminal. If found the correct device, device number will appear at **Device** line in **Midrange Configuration** box. Click the **OK** button to continue. For RBX-877V2.00 board would be use this group chip because the controller board provides PIC16F887; it is mid-range PIC microcontroller.

- **PIC18F** - Configures the programming software for PIC18F core flash devices.

- **PIC18F_J_** - Configures the programming software for *PIC18FxxJxx* low voltage devices.

- **PIC24** - Configures the programming software for 16-bit core devices; *PIC24FJxx.*

- **dsPIC30** - Configures the programming software for 16-bit core devices; *dsPIC30Fxx.*

- **dsPIC33** - Configures the programming software for 16-bit core devices; *dsPIC33Fxx.*

PROGRAMMER

• **Read Device** – Reads the program memory, data EEPROM memory, ID locations, and Configuration bits.

• **Write Device** – Writes the program memory, data EEPROM memory, ID locations, and Configuration bits.

• **Verify** – Verifies the program memory, data EEPROM memory, ID locations and Configuration bits read from the target MCU against the code stored in the programming software.

• **Erase** – Performs a bulk erase of the target MCU. OSCCAL and band gap values are preserved (PIC12F629/675 and PIC16F630/676 only).

• **Blank Check** – Performs a blank check of program memory, data EEPROM memory, ID locations and Configuration bits.

• **Verify on Write** - Verifies program memory, data EEPROM memory, ID locations and Configuration bits read from the target MCU against the code stored in the programming software with word per word.

• **Full Erase (OSCCAL and BG erased)** – Performs a bulk erase including the OSCCAL and Band Gap (BG) values (PIC12F629/675 and PIC16F630/676 only).

• **Regenerate OSCCAL** – Regenerates the OSCCAL value (only for PIC12F629/675 and PIC16F630/676). The AUX line must be connected to the RA4/T1G pin.

• **Set Band Gap Calibration Value** – Sets the band gap value.

• **Write on PICkit Button** - Set for supporting of programming the target microcontroller witth PROGRAM switch on the USB programmer board.

TOOLS

• **Enable Code Protect** – Enables code protection for Flash program memory.

• **Enable Data Protect** – Enables code protection for EEPROM data memory.

• **Set OSCCAL** - Sets the OSCCAL value for alignment internal clock frequency.

• **Target VDD Source** – Power target from the USB Programmer.

Auto-Detect : Select to USB programmer turn on or off the supply voltage to target microcontroller automatically (not suggess to use this option).

Forced PICkit2 : Set the programmer to supply the suitable voltage to target microcontroller. After select, LED at Targer position will light and at VDD PICkit2 box on screen will check atr On position. User can adjust the supply voltage from selection box in the right-hand (not suggess to use this option).

Forced Target : Select to inform the software knows about the target has voltage applied. Suggess to use this option for safty operation. Also in this option, user must apply the supply voltage to the target PIC microcontroller.

• **Fast Programming** - Select the PX-200 to programs the Flash device with high speed.

• **Check Communication** – Verifies communication with the USB Programmer and reads the device ID of the target MCU.

• **Download PICkit 2 Firmware** – Performs a download of the USB Programmer firmware operating system. (this USB programmer is compatible PICkit2™ Programmer). Sometime call this function to OS update.

Help

Displays all user manual, technical document and a dialog box indicating the version and date.

1.3.2.2.3 Important things to know in using the PICkit2™ Programming Software

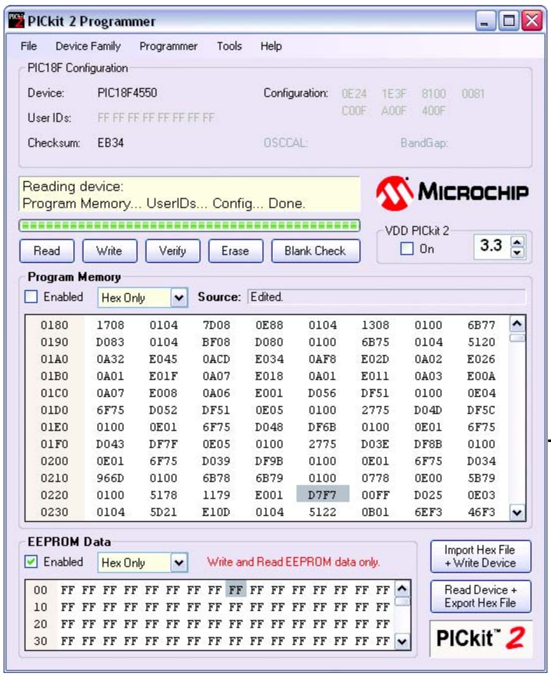
Editing memory value

The PICkit2™ Programming Software supports the editing memory value in each address, both Flash program and data EEPROM memory. User can click at any address that need to change the value and input the new value directly.

Moreover user can select to access both memory types and only one.

(a) Access only EEPROM data memory

Click at Enabled box in Program Memory border to remove the mark. At EEPROM data border, it will show **Write and Read EEPROM data only** in red message. It means user can read and write only EEPROM data memory. See the illlustration below.



PIC18F Configuration

Device: PIC18F4550
Configuration: 0E24 1E3F 8100 0081 C00F A00F 400F
User IDs: FF FF FF FF FF FF FF FF
Checksum: EB34
OSCCAL:
BandGap:

Reading device:
Program Memory... UserIDs... Config... Done.

VDD PICkit 2: [ ] On 3.3

Program Memory: [ ] Enabled Hex Only Source: Edited.

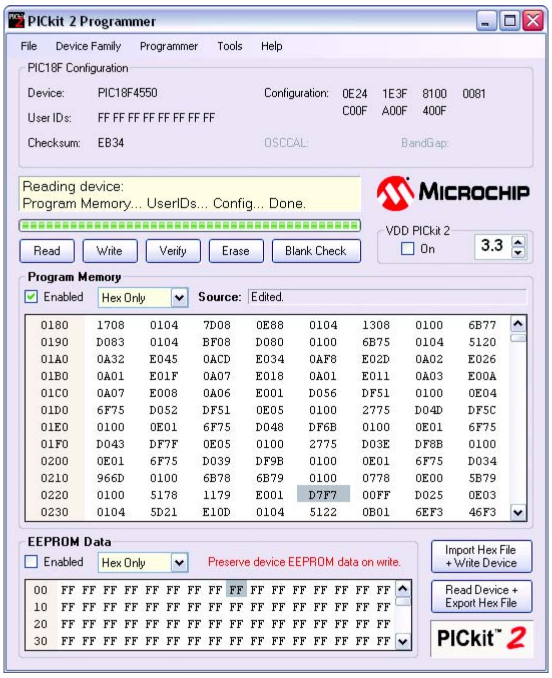
```
0180  1708  0104  7D08  0E88  0104  1308  0100  6B77
0190  D083  0104  BF08  D080  0100  6B75  0104  5120
01A0  0A32  E045  0ACD  E034  0AF8  E02D  0A02  E026
01B0  0A01  E01F  0A07  E018  0A01  E011  0A03  E00A
01C0  0A07  E008  0A06  E001  D056  DF51  0100  0E04
01D0  6F75  D052  DF51  0E05  0100  2775  D04D  DF5C
01E0  0100  0E01  6F75  D048  DF6B  0100  0E01  6F75
01F0  D043  DF7F  0E05  0100  2775  D03E  DF8B  0100
0200  0E01  6F75  D039  DF9B  0100  0E01  6F75  D034
0210  966D  0100  6B78  6B79  0100  0778  0E00  5B79
0220  0100  5178  1179  E001  D7F7  00FF  D025  0E03
0230  0104  5D21  E10D  0104  5122  0B01  6EF3  46F3
```

EEPROM Data: [✓] Enabled Hex Only Write and Read EEPROM data only.

```
00  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
30  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

(b) Access only Flash program memory

Click at Enabled box in EEPROM data border to remove the mark. At EEPROM data border will show **Preserve device EEPROM data on write** in red message. It means the EEPROM data memory will be protected. User can access only Flash program memory. See the illlustration below.



PICkit 2 Programmer

PIC18F Configuration

Device: PIC18F4550
User IDs: FF FF FF FF FF FF FF FF
Checksum: EB34
Configuration: 0E24 1E3F 8100 0081 C00F A00F 400F

Reading device:
Program Memory... UserIDs... Config... Done.

VDD PICkit 2: On [ ] 3.3

Program Memory: Enabled [✓] Hex Only Source: Edited.

```
0180   1708  0104  7D08  0E88  0104  1308  0100  6B77
0190   D083  0104  BF08  D080  0100  6B75  0104  5120
01A0   0A32  E045  0ACD  E034  0AF8  E02D  0A02  E026
01B0   0A01  E01F  0A07  E018  0A01  E011  0A03  E00A
01C0   0A07  E008  0A06  E001  D056  DF51  0100  0E04
01D0   6F75  D052  DF51  0E05  0100  2775  D04D  DF5C
01E0   0100  0E01  6F75  D048  DF6B  0100  0E01  6F75
01F0   D043  DF7F  0E05  0100  2775  D03E  DF8B  0100
0200   0E01  6F75  D039  DF9B  0100  0E01  6F75  D034
0210   966D  0100  6B78  6B79  0100  0778  0E00  5B79
0220   0100  5178  1179  E001  D7F7  00FF  D025  0E03
0230   0104  5D21  E10D  0104  5122  0B01  6EF3  46F3
```

EEPROM Data: Enabled [ ] Hex Only Preserve device EEPROM data on write.

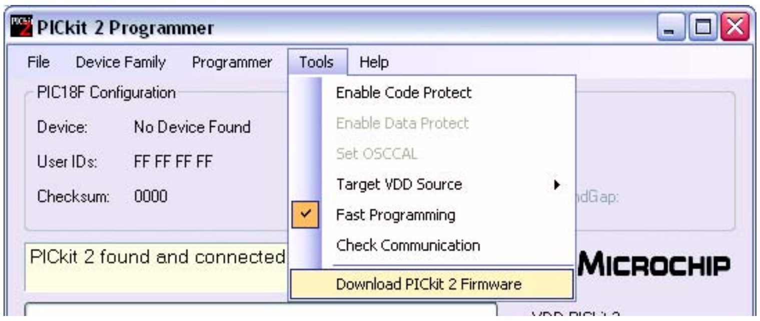
```
00  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
30  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

1.3.2.3 Updating the USB Programmer Firmware

To update the programmer firmware Operating System, complete the following steps.

(1) Download the latest PICkit 2 Operating System from the Microchip web site at www.microchip.com. Because the Robo-PICA's USB programmer is compatible Microchip's PICkit2™ programmer.

(2) From the menu, select **Tools → Download PICKit 2 OS Firmware**, as shown in figure below



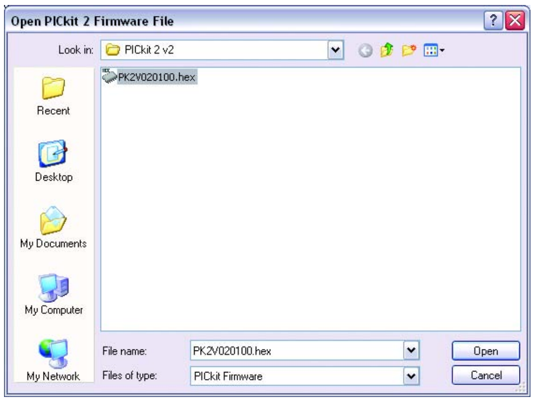
PICkit 2 Programmer

PIC18F Configuration

Device: No Device Found
User IDs: FF FF FF FF
Checksum: 0000

PICkit 2 found and connected

(3) Browse to the directory where the latest Operating System code was saved, Select the **PK2*.hex** file and click on the **Open** button as shown in figure below.



Look in: PICkit 2 v2
File name: PK2V020100.hex
Files of type: PICkit Firmware

(4) The progress of the OS update will be displayed in the status bar of the programming software and the Busy LED on the USB Microcontroller Programmer will flash. When the update completes successfully, the status bar will display **"Operating System Verified"** and the Busy LED will go out. The operating system update is then complete.

1.3.2.4 Short cut button

The PICkit2™ Programming Software has 7 short cut buttons as follows :

(1) **Read** : Read data from target MCU.

(2) **Write** : Write or program the code into target MCU.

(3) **Verify** : Verify programming.

(4) **Erase** : Erase data in target MCU.

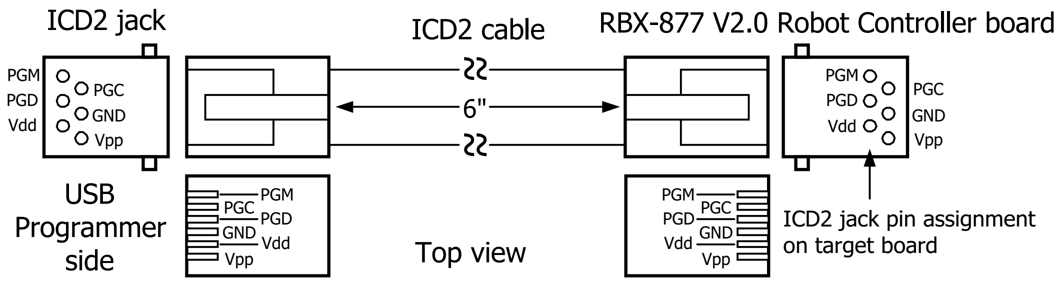
(5) **Blank Check** : Check blank data in target MCU.

(6) **Import Hex File + Write Device** : Open the HEX file and program into target MCU automatically

(7) **Read Device + Export Hex File** : Read device and save as the HEX file automatically.

1.3.2.5 ICD2 cable assignment

The USB Programmer comes with an ICD2 cable for connecting between the programmer and the target board. The wire assignment of this cable is shown below.


ICD2 jack
ICD2 cable
RBX-877 V2.0 Robot Controller board
PGM
PGD
Vdd
PGC
GND
Vpp
6"
USB Programmer side
Top view
ICD2 jack pin assignment on target board

1.4 Programming devleopment for Robo-PICA

The summary of steps of program the Robo-PICA robot kit are as follows :

1. Create the C project file with mikroC IDE software.

2. Compile the project file.

3. If any error occurs, edit the C program to fix the error and compile the project file until all are correct.

4. The HEX file would be created after the compilation is completed.

5. Open the PICkit2™ Programming software. Connect the USB programmer with USB port and connect the ICD2 cable between the USB Programmer and the RBX-877 V2.0 Controller board at ICD2 jack.

6. Download the HEX file to the RBX-877 V2.0 Controller board of Robot-PICA.

7. Run the program and check the hardware operation. If it is not correct, go back to edit the C program, compile and download again. Do these steps unitl the operation are completed.

1.5 Getting Start

From here, we will describe about the getting start of programming development for the Robo-PICA. This robot kit is controlled by the RBX-877 V2.0 Robot Controller board. The heart of this controller board is PIC16F887 chip. The programming development includes 2 main steps as C programming development and Download the HEX file to microcontroller.

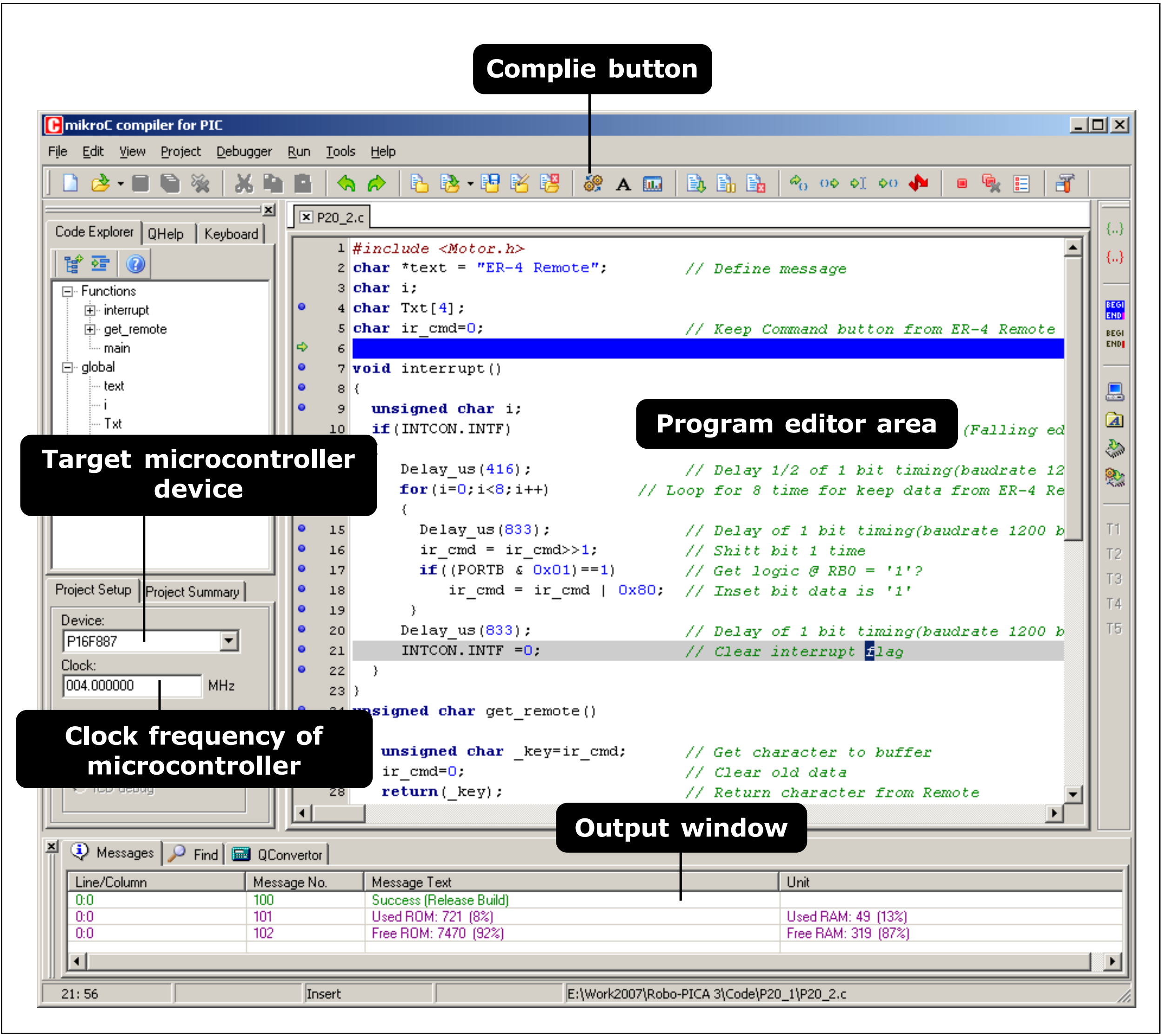
The C programming development will be using mikroC IDE included C compiler and the other provides support tools and libraries. However this kit will work with the demo version. You can purchase the full version for more programming at www.mikroe.com.

You can develop the C project file and test the operation of the Robo-PICA's hardware from these procedures below.

1.5.1 Install the mikroC software tools following the instruction manual. See this document in Robo-PICA's CD-ROM or download the update document from www.mikroe.com.

1.5.2 Install the PICkit2 Programming software for USB programmer.

1.5.3 Open the mikroC IDE by clicking at Start → Programs → Mikroelktronika → mikroC → mikroC. The main window will appear. The Figure 1-1 shows the main window of mikroC IDE and the important components.


Complie button
mikroC compiler for PIC
File Edit View Project Debugger Run Tools Help
Code Explorer QHelp Keyboard
Functions
interrupt
get_remote
main
global
text
i
Txt
Target microcontroller device
Project Setup Project Summary
Program editor area
Clock frequency of microcontroller
Output window
Messages Find QConvertor
P20_2.c

```
#include <Motor.h>
char *text = "ER-4 Remote";            // Define message
char i;
char Txt[4];
char ir_cmd=0;                         // Keep Command button from ER-4 Remote

void interrupt()
{
  unsigned char i;
  if(INTCON.INTF)                                                         (Falling ed

    Delay_us(416);                     // Delay 1/2 of 1 bit timing(baudrate 12
    for(i=0;i<8;i++)               // Loop for 8 time for keep data from ER-4 Re
    {
      Delay_us(833);                   // Delay of 1 bit timing(baudrate 1200 b
      ir_cmd = ir_cmd>>1;              // Shitt bit 1 time
      if((PORTB & 0x01)==1)            // Get logic @ RB0 = '1'?
         ir_cmd = ir_cmd | 0x80;       // Inset bit data is '1'
     }
    Delay_us(833);                     // Delay of 1 bit timing(baudrate 1200 b
    INTCON.INTF =0;                    // Clear interrupt flag
  }
}
unsigned char get_remote()

   unsigned char _key=ir_cmd;          // Get character to buffer
   ir_cmd=0;                           // Clear old data
   return(_key);                       // Return character from Remote
```

Device: P16F887

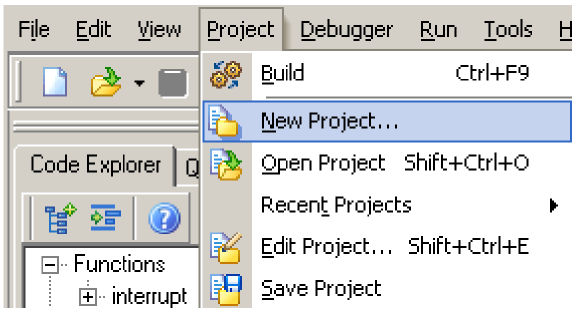
Clock: 004.000000 MHz

Line/Column	Message No.	Message Text	Unit
0:0	100	Success (Release Build)	
0:0	101	Used ROM: 721 (8%)	Used RAM: 49 (13%)
0:0	102	Free ROM: 7470 (92%)	Free RAM: 319 (87%)

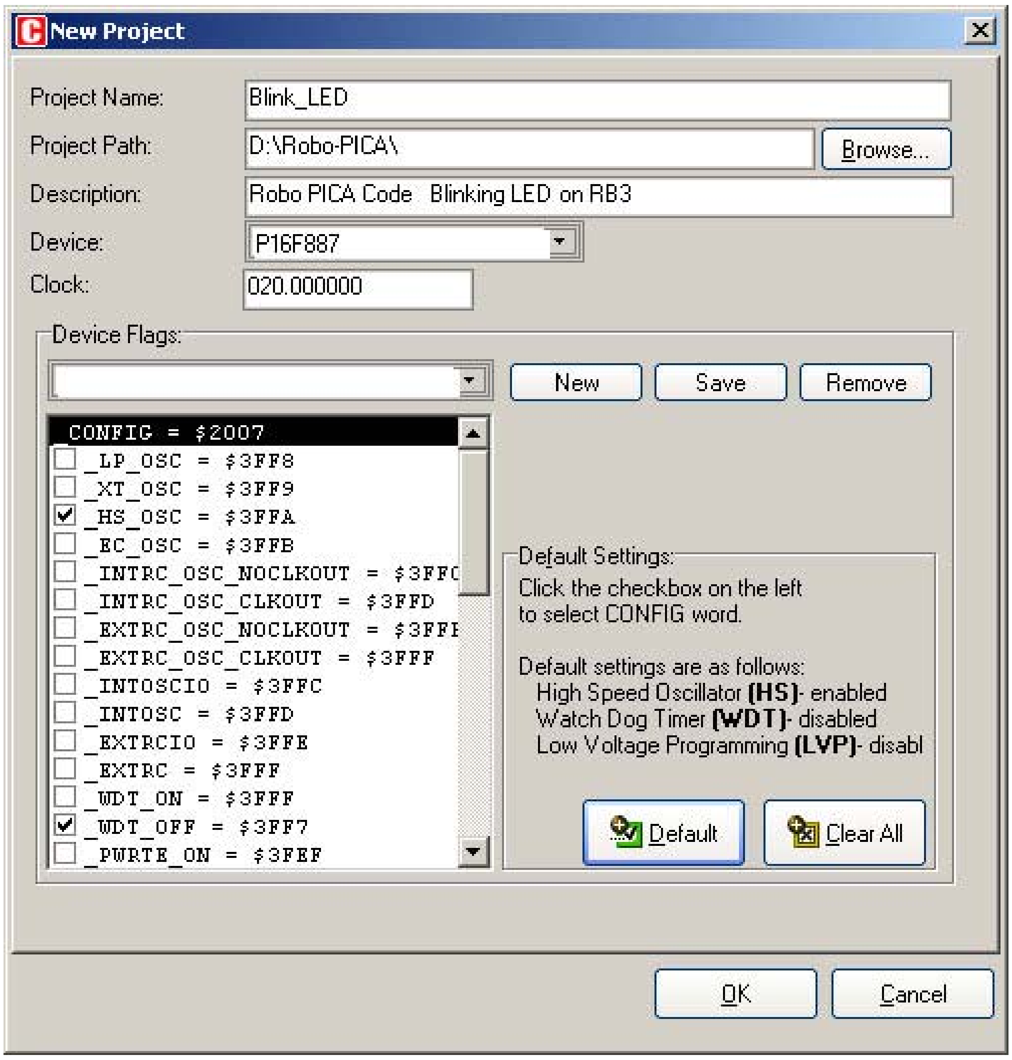
21: 56 Insert E:\Work2007\Robo-PICA 3\Code\P20_1\P20_2.c

Figure 1-1 : The main window of mikroC IDE and the important components need to know.

1.5.4 Create the new project file by entering to menu **Project** and select **New Project...**



1.5.5 The **New Project** window will appear. You must set the important parameter as follows :



New Project

Project Name: Blink_LED

Project Path: D:\Robo-PICA\

Description: Robo PICA Code Blinking LED on RB3

Device: P16F887

Clock: 020.000000

Device Flags:

```
CONFIG = $2007
[ ] _LP_OSC = $3FF8
[ ] _XT_OSC = $3FF9
[✓] _HS_OSC = $3FFA
[ ] _EC_OSC = $3FFB
[ ] _INTRC_OSC_NOCLKOUT = $3FFC
[ ] _INTRC_OSC_CLKOUT = $3FFD
[ ] _EXTRC_OSC_NOCLKOUT = $3FFE
[ ] _EXTRC_OSC_CLKOUT = $3FFF
[ ] _INTOSCIO = $3FFC
[ ] _INTOSC = $3FFD
[ ] _EXTRCIO = $3FFE
[ ] _EXTRC = $3FFF
[ ] _WDT_ON = $3FFF
[✓] _WDT_OFF = $3FF7
[ ] _PWRTE_ON = $3FEF
```

Default Settings:
Click the checkbox on the left to select CONFIG word.

Default settings are as follows:
High Speed Oscillator **(HS)**- enabled
Watch Dog Timer **(WDT)**- disabled
Low Voltage Programming **(LVP)**- disabl

(a) **Project Name** : Put the project name. mikroC IDE will create the folder to support your project file which includes the C sourcecode. For example is ***Blink_LED*** project file.

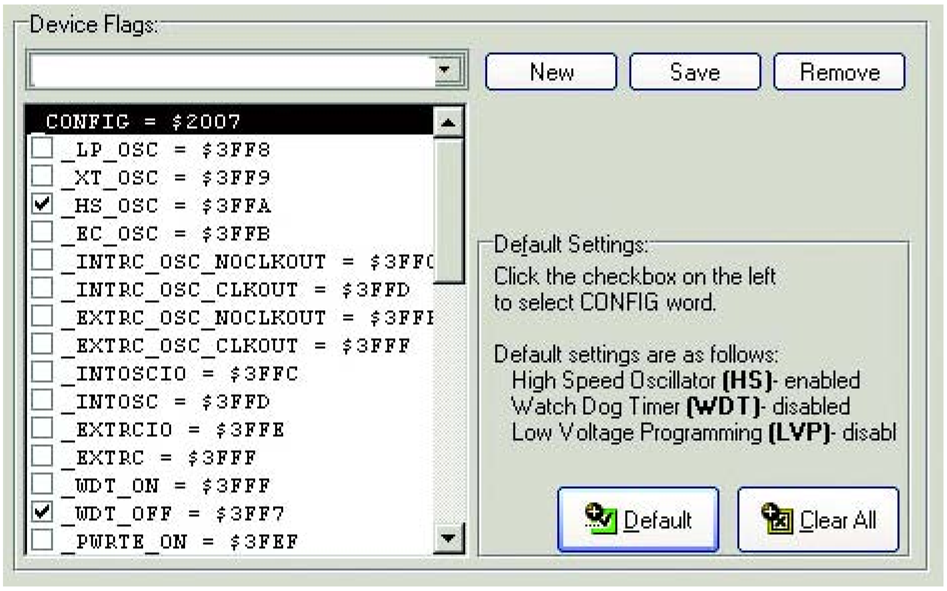
(b) **Project Path** : Select the location of your project file. Click the **Browse** button to select the location. For example is ***D:\ROBO-PICA***

(c) **Description** : Put the information of your porject file. For example is *"Robo PICA Code Blinking LED on RB3"*

(d) **Device** : Select the target microcontroller. For the Robo-PICA kit and RBX-877V2.0 Controller board must select to **PIC16F887**

(e) **Clock** : Select the clock frequency for the target microcontroller. For the Robo-PICA kit and RBX-877V2.0 Controller board use 20MHz clock. Put the value **020.000000.**

(f) **Device Flags** : Set the configuration for the target microcontroller. Developer can set very easy by Default button Default. The Default will set the 3 main configurations as follows :



```
CONFIG = $2007
[ ] _LP_OSC = $3FF8
[ ] _XT_OSC = $3FF9
[✓] _HS_OSC = $3FFA
[ ] _EC_OSC = $3FFB
[ ] _INTRC_OSC_NOCLKOUT = $3FF
[ ] _INTRC_OSC_CLKOUT = $3FFD
[ ] _EXTRC_OSC_NOCLKOUT = $3FF
[ ] _EXTRC_OSC_CLKOUT = $3FFF
[ ] _INTOSCIO = $3FFC
[ ] _INTOSC = $3FFD
[ ] _EXTRCIO = $3FFE
[ ] _EXTRC = $3FFF
[ ] _WDT_ON = $3FFF
[✓] _WDT_OFF = $3FF7
[ ] _PWRTE_ON = $3FEF
```

Default Settings:
Click the checkbox on the left to select CONFIG word.

Default settings are as follows:
High Speed Oscillator **(HS)**- enabled
Watch Dog Timer **(WDT)**- disabled
Low Voltage Programming **(LVP)**- disabl

*High Speed Oscillator enabled **(HS_OSC)** for 10MHz and above clock frequency.*

*Watchdog timer disabled **(WDT_OFF)***

*Low Voltage Programming disabled **(LVP_OFF)***

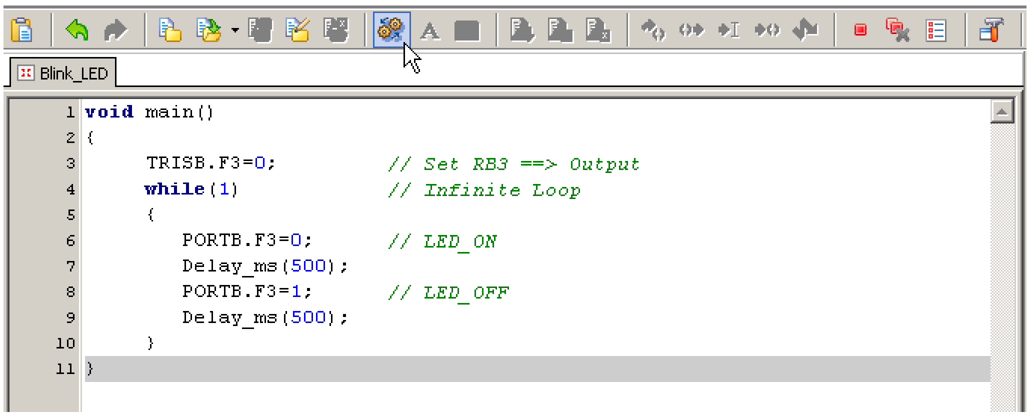
After the configuration is being set, click on the **OK** button. mikroC IDE would close the **New Project** window and create the ***Blink_LED.C*** file with the blank editor area for writing the C program.

1.5.6 Type the C program following the Listing 1-1.

```
void main()
{
      TRISB.F3=0;            // Set RB3 as Output
      while(1)               // Infinite Loop
      {
         PORTB.F3=0;         // LED_ON
         Delay_ms(500);
         PORTB.F3=1;         // LED_OFF
         Delay_ms(500);
      }
}
```

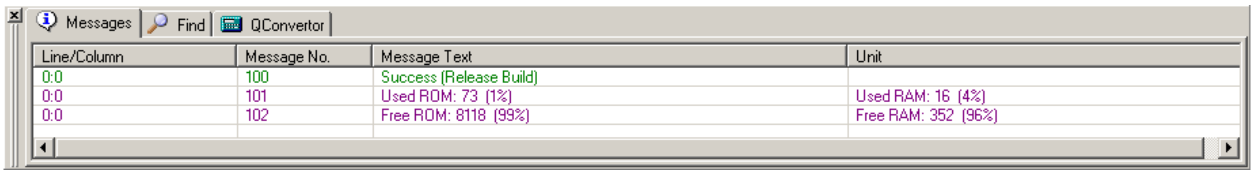
Listing 1-1 : Blinking LED test code

1.5.7 Click on the **Build Project** button or **Ctrl+F9** for compiling the project file.



```
void main()
{
     TRISB.F3=0;          // Set RB3 ==> Output
     while(1)             // Infinite Loop
     {
        PORTB.F3=0;       // LED_ON
        Delay_ms(500);
        PORTB.F3=1;       // LED_OFF
        Delay_ms(500);
     }
}
```

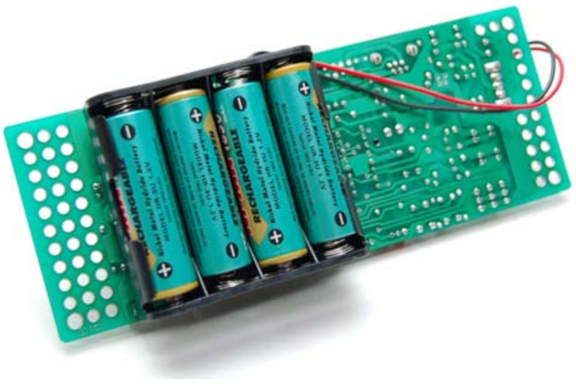
1.5.8 Observe the error message at the **Output window**. If all is correct, it would show the size of usage program memory of this file and Success message.



Line/Column	Message No.	Message Text	Unit
0:0	100	Success (Release Build)	
0:0	101	Used ROM: 73 (1%)	Used RAM: 16 (4%)
0:0	102	Free ROM: 8118 (99%)	Free RAM: 352 (96%)

After that, you will get the HEX file; ***Blink_LED.HEX*** for downloading to the Robo-PICA's controller board; RBX-877V2.0.

1.5.9 Put 4 of AA batteries into battery holder of the RBX-877V2.0 Controller board.



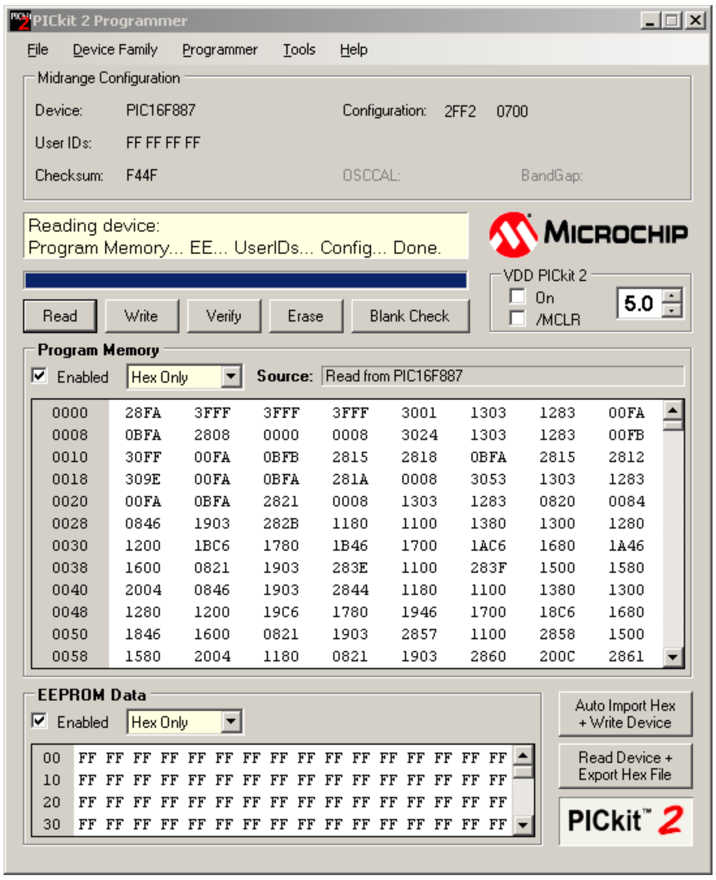
1.5.10 Connect the USB programer with PC's USB port

1.5.11 Connect the ICD2 cable between the USB programmer and the RBX-877V2.0 Controller board.

1.5.12 Turn-on the power to the RBX-877 V2.0 Controlller board.

1.5.13 Open the PICkit2™ Programming software.

1.5.14 If all connections are correct, the PICkit2™ Programming software will check the target microontroller automaticcaly and show **PIC16F887 is found.**


PICkit 2 Programmer
File Device Family Programmer Tools Help
Read Write Verify Erase Blank Check
Auto Import Hex + Write Device
Read Device + Export Hex File
PICkit™ 2
MICROCHIP

Midrange Configuration

Device: PIC16F887 Configuration: 2FF2 0700

User IDs: FF FF FF FF

Checksum: F44F OSCCAL: BandGap:

Reading device:
Program Memory... EE... UserIDs... Config... Done.

VDD PICkit 2: On ☐ /MCLR ☐ 5.0

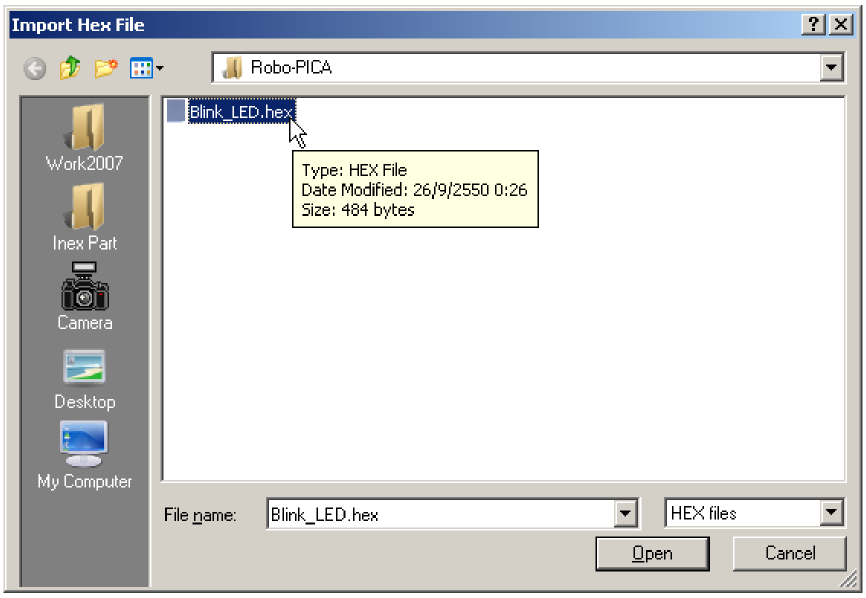
Program Memory: ✓ Enabled Hex Only Source: Read from PIC16F887

```
0000  28FA  3FFF  3FFF  3FFF  3001  1303  1283  00FA
0008  0BFA  2808  0000  0008  3024  1303  1283  00FB
0010  30FF  00FA  0BFB  2815  2818  0BFA  2815  2812
0018  309E  00FA  0BFA  281A  0008  3053  1303  1283
0020  00FA  0BFA  2821  0008  1303  1283  0820  0084
0028  0846  1903  282B  1180  1100  1380  1300  1280
0030  1200  1BC6  1780  1B46  1700  1AC6  1680  1A46
0038  1600  0821  1903  283E  1100  283F  1500  1580
0040  2004  0846  1903  2844  1180  1100  1380  1300
0048  1280  1200  19C6  1780  1946  1700  18C6  1680
0050  1846  1600  0821  1903  2857  1100  2858  1500
0058  1580  2004  1180  0821  1903  2860  200C  2861
```

EEPROM Data: ✓ Enabled Hex Only

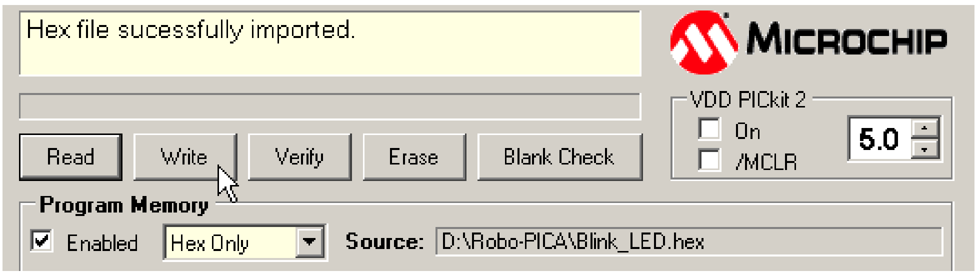
```
00  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
30  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

1.5.15 Select the HEX fle which require program to microcontroller by entering menu **File → Import Hex**. The open HEX file window will appear. Select to ***D:\ROBO-PICA\Blink_LED*** for selecting the ***Blink_LED.hex***



Type: HEX File
Date Modified: 26/9/2550 0:26
Size: 484 bytes

1.5.16 Click on the **Write** button to download HEX file to the RBX-877 V2.0 Controlller board.



Hex file sucessfully imported.

VDD PICkit 2
☐ On
☐ /MCLR
5.0

☑ Enabled
Hex Only
Source: D:\Robo-PICA\Blink_LED.hex

1.5.17 Observe the result at RB3 LED on the RBX-877 V2.0 Controlller board.

RB3 LED of the RBX-877V2.0 Controller board blinks always.

Chapter 2

RBX-877V2.0 Robot Controller Board

Robo-PICA robotic kit is controlled by the RBX-877 V2.0 (PIC16F887 Robot Experiment board). The main microcontroller is the PIC16F887. Figure 2-1 shows operating diagram of RBX-877 board. In this chapter will present the operation of RBX-877 board and some example experiment. Builders must read and test all experiments for building and programming the robot in next chapter.

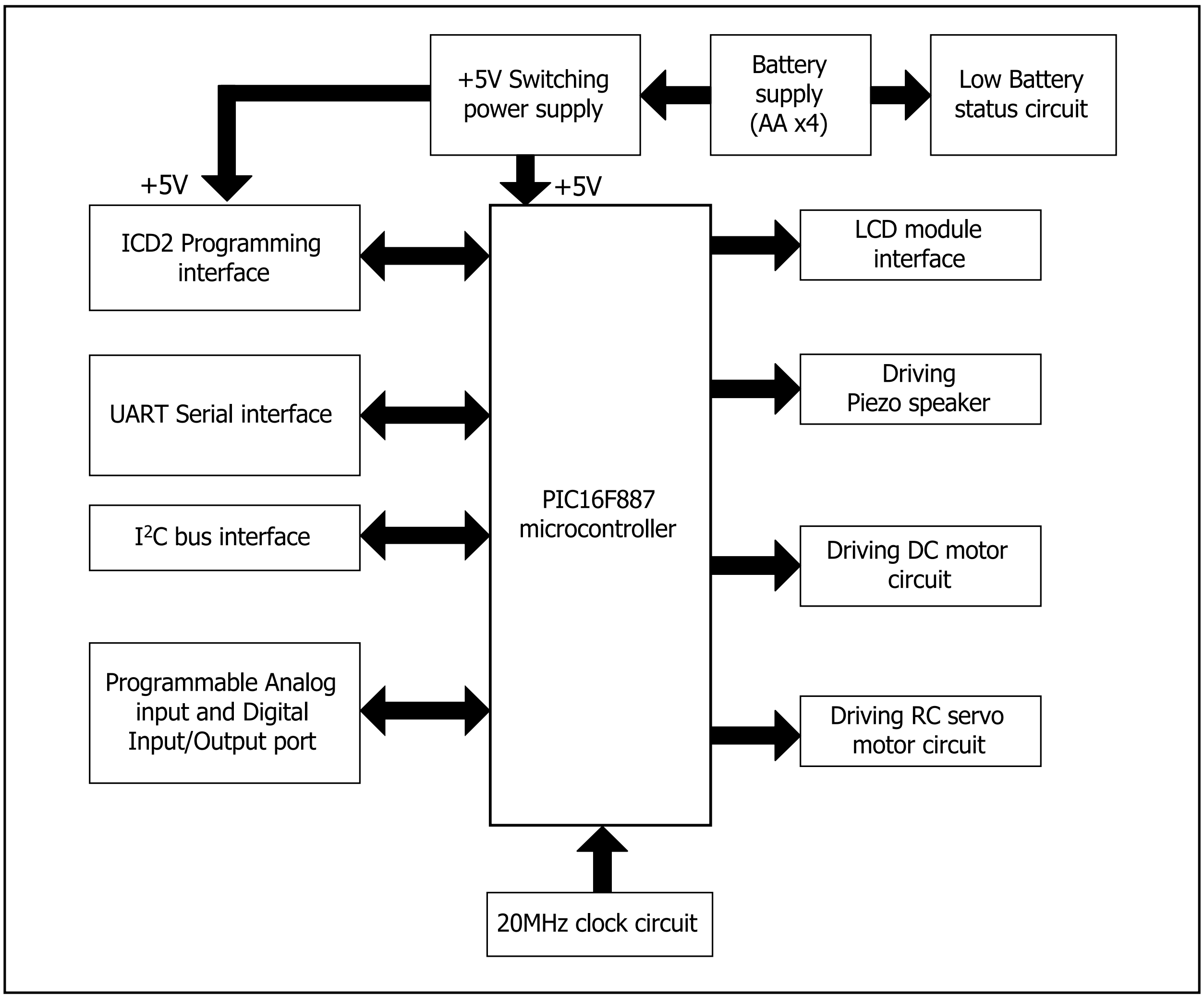

+5V Switching power supply
Battery supply (AA x4)
Low Battery status circuit
+5V
+5V
ICD2 Programming interface
UART Serial interface
I²C bus interface
Programmable Analog input and Digital Input/Output port
PIC16F887 microcontroller
LCD module interface
Driving Piezo speaker
Driving DC motor circuit
Driving RC servo motor circuit
20MHz clock circuit

Figure 2-1 : RBX-877 V2.0 Robot controller board's block diagram

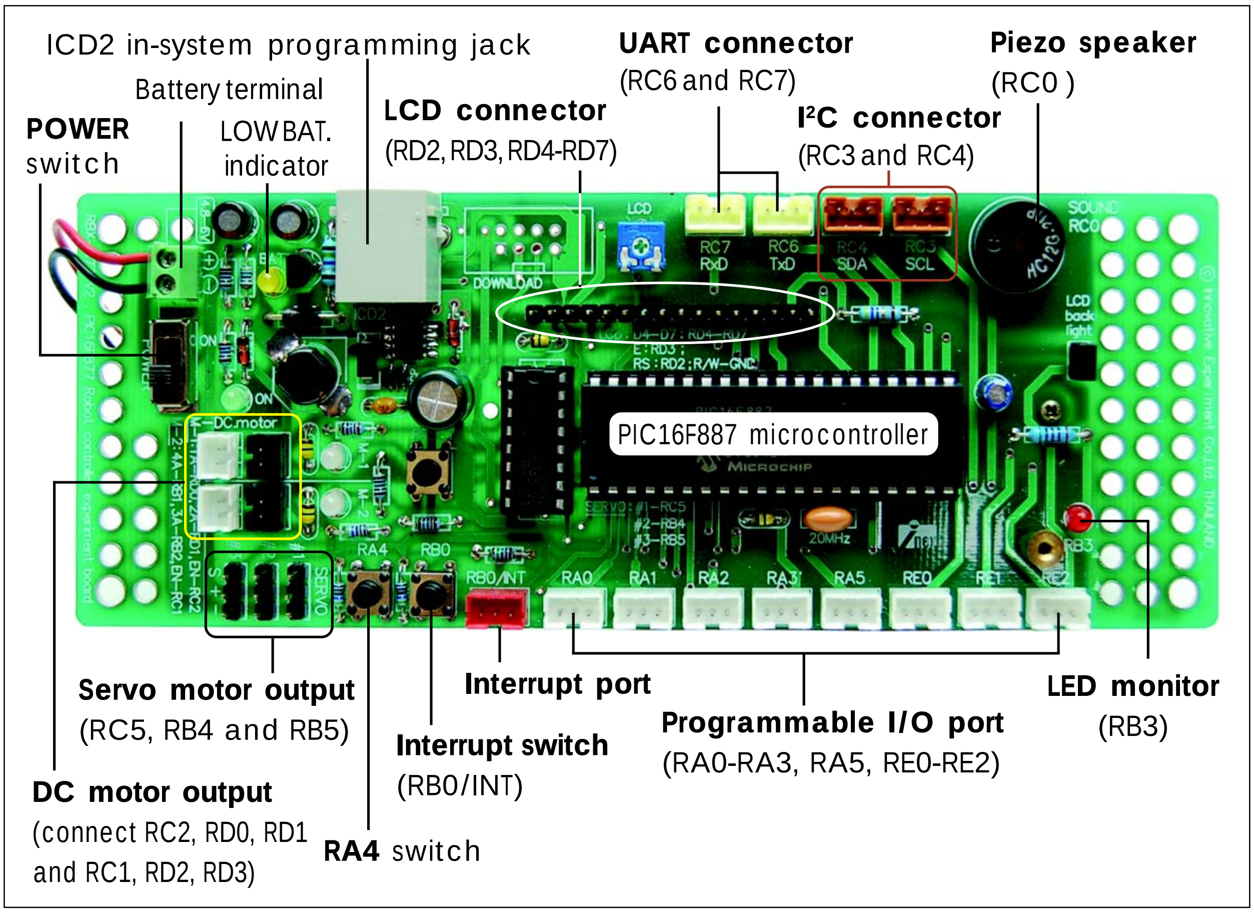

ICD2 in-system programming jack
Battery terminal
POWER switch
LOW BAT. indicator
LCD connector (RD2, RD3, RD4-RD7)
UART connector (RC6 and RC7)
I²C connector (RC3 and RC4)
Piezo speaker (RC0)
PIC16F887 microcontroller
Servo motor output (RC5, RB4 and RB5)
DC motor output (connect RC2, RD0, RD1 and RC1, RD2, RD3)
RA4 switch
Interrupt port
Interrupt switch (RB0/INT)
Programmable I/O port (RA0-RA3, RA5, RE0-RE2)
LED monitor (RB3)

Figure 2-2 : RBX-877 V2.0 Robot controller board layout

2.1 Technical features

- Controlled by PIC16F887 Microcontroller with 8Kword memory. Run with 20MHz clock
- Download the program via ICD2 jack.
- LCD16x2 display with LED back light and jumper to on/off control
- Piezo speaker
- a LED monitor
- Drive 2-DC motors 4.5V to 6V and 3-RC Servo motors (in range 4.8 to 6V)
- 9-Programmable ports support all analog inout and digtial input/output
- I²C bus port
- UART port for interfacing the serial device such as RS-232 transceiver, XBee module and Bluetooth.
- Supply voltage from 4 of AA batteries (Rechargable battery is recommended)
- 2.375 x 6.25 Inches size

2.2 RBX-877 V2.0 board circuit description

2.2.1 Microcontroller circuit

The heart of this board is the PIC16F887 microcontroller. The 20MHz ceramic resonator, CR1 is used to make the 20MHz clock for PIC16F887.

2.2.2 Power supply

The RBX-877 V2.0 board contains a step-up switching power supply to supply +5V regulated for PIC16F887. Although the level of battery will decrease when driving the motor. This switching power supply circuit will maintain the +5V for microcontroller until battery voltage level down to 1.5V

S1 is on-off switch to supply the voltage from batteries to RBX-877 V2.0 board. R3, D1 and ZD1 ard used to limit the input voltage to IC2 not over 5.1V

IC1 is a switching power supply IC, NCP1450-5.0. It can support input voltage 1.5 to 4.2V range for regulating +5V supply voltage. ZD2 is used to limit output voltage of NCP1450-5.0 not over +5V.

2.2.3 In-System Programming circuit

The RBx-877 V2.0 board require In-system programming via ICD2 or ISP connector. The USB programmer which is bundled in the Robo-PICA kit will connect to ICD2 jack of the RBX-877 V2.0 controller board. It use the supply voltage from USB port of computer.

The programming signal will send to RB6 and RB7 pin of PIC16F887. The high voltage programming is sent to MCLR pin. All programming status would be show on the PICkit2 Programming software on computer’s monitor. After programming complete, this controller can work suddenly.

2.2.4 Display circuit

Character display : The RBX-877 V2.0 board provides LCD module connector. It supports 16 characters 2 lines LCD. PIC16F877’s RD4 to RD7 pin are assigned to D4 to D7 data pins, RD3 to E pin and RD2 to RS pin for selection data mode. VR1 is used to contrast adjustment of LCD screen. In case using Back-light LCD, it provides a jumper to control the LED back-light of LCD.

LED monitor : RBX-877 V2.0 board has a general purpose LED. They are connected to RB3 of PIC16F887 microcontroller via a current limited resistor.

Sound output : RBX-877 V2.0 board has a sound driver circuit. Connect RC0 pin to a piezo speaker via a capacitor 10µF. This circuit can drive audio frequency signal. However the piezo speaker has the resonance frequency of range 1kHz to 3kHz.

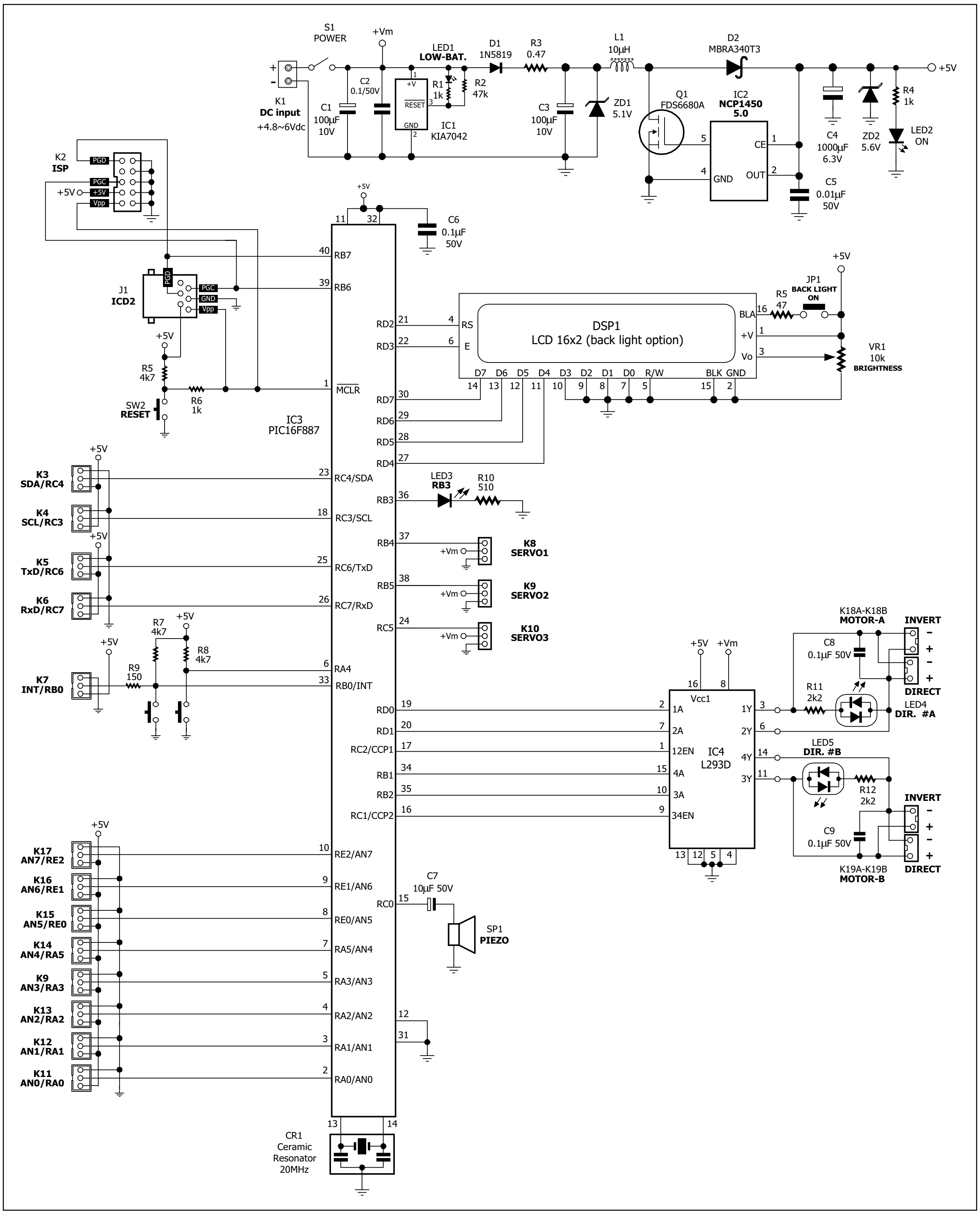


Figure 2-3 : Schematic diagram of RBX-877 V2.0 Robot Controller board

2.2.5 Programmable port

The RBX-877 V2.0 board provides 9-programmable multipurpose ports. It includes RA0-RA3, RA5, RE0-RE2 and RB0 pin. All port pin can program to 3 functions as

(1) Analog input - to get analog signal to A/D converter circuit inside microcontroller. Input voltage range is 0 to 5V. Converter resolution is 10-bit.

(2) Digital input - to get digital signal from digital device and switch.

(3) Digital output - to drive digital signal logic “0” and “1” to external device.

In default all port will be set to analog input port.

On RBX-877 V2.0 board provides all ports in 3-pin JST connector. Each connector includes +5V and GND.

2.2.6 UART port for serial data wired/wireless communication

Builders can make the serial data communication from RBX-877 V2.0 board to computer’s RS-232 serial port and many wireless serial device such as XBEE module and Bluetooth. PIC16F887 microcontroller provides RC6 and RC7 pin UART module port pin for this purpose. Serial signal from PIC16F887’s are connected to 2 free JST connectors for support all serial device.

2.2.7 DC motor driver circuit

The RBX-877 V2.0 board use IC4, L293D H-bridge motor driver IC are used for driving 2 channel DC motors. The suitable motor is 4.5-6V 100 to 200mA or up to 400mA.

Motor A speed is controlled by RD0 with RD1 pin and enable by RC2 port. Motor B speed is controlled by RB1 with RB2 pin and enable by RC1. LED4 and LED5 are bi-color LED. They are used for showing the motor output status.

Voltage is supplied to L293D includes +5V supply voltage and Motor supply voltage (+Vm). The +Vm is concentrated direct from batteries for powerful driving.

2.2.8 RC servo motor driver circuit

The RBX-877 V2.0 Controller board provides 3 port pins for RC servo motors. It includes RB4, RB5 and RC5 . RC servo motor supply comes from system battery. This driver cannot support high-current and high power RC servo motor. The suitable RC servo motor is 4.8 to 6V motor and need current consumption about 100-200mA.

2.2.9 I^{2}C connector

A way to expansion of RBX-877 V2.0 board is using a I^{2}C bus connector. Many external device need I^{2}C bus protocol such as Real-time clock, memory, A/D and D/A converter, Port expansion device and etc. RC3/SCL and RC4/SDA of PIC16F887 are connected to I^{2}C bus connector includes +5V supply and GND. No any pull-up resistor are connected to theses port. User must provides them at the external devices.

Activity 1

Write programs for testing RBX-877 V2.0 Controller board

Procedure

For all activities of the programming development for Robo-PICA robot kit have the summary of steps are as follows :

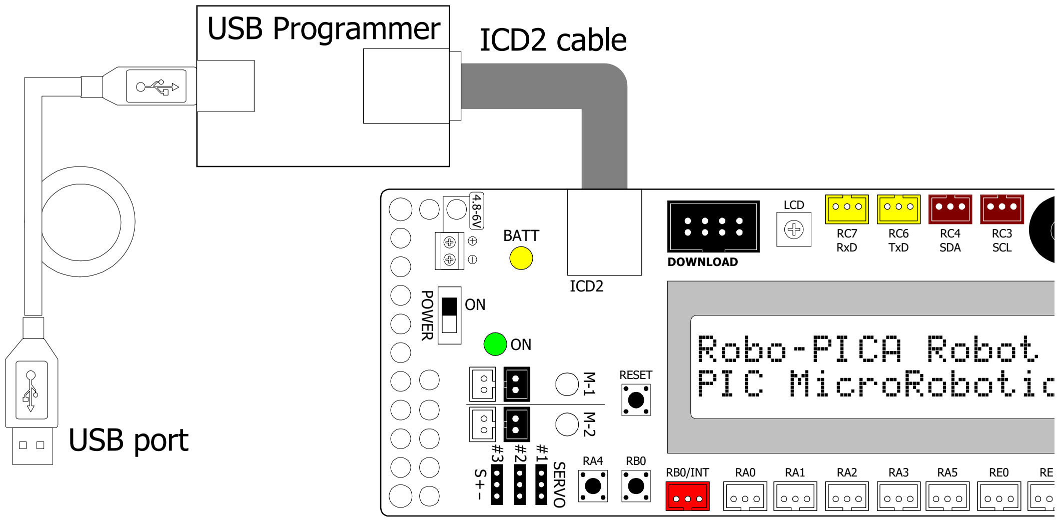
1. Create the C project file with mikroC IDE software.

2. Compile the project file.

3. If any error happens, edit the C program to fix the error and compile the project file until all are correct.

4. The HEX file would be created after the compilation is completed.

5. Connect the USB programmer with USB port and connect the ICD2 cable between the USB Programmer and the RBX-877 V2.0 Controller board at ICD2 jack.



6. Open the PICkit2™ Programming software and checking the connection.

7. Download the HEX file to the PIC16F887 on the RBX-877 V2.0 Controller board of Robot-PICA.

8. Run the program and check the hardware operation. If it is not correct, go back to edit the C program, compile and download again. Do these steps unitl the operation are complete.

Activity 1-1 LED testing

See the Figure A1-1, RB3 of PIC16F887 is connected to LED via current limited resistor 510Ω. For turning-on this LED must send logic “1” to this port. and send logic “0” for turning-off.

A1.1.1 Write program following the Listing A1-1 then compile and download to RBX-877 V2.0 Robot Controller board. See the operation.

LED at RB3 on.

A1.1.2 Write program following the Listing A1-2 then compile and download to RBX-877 V2.0 Robot Controller board. See the operation.

LED at RB3 will blink with 0.5 second duration.

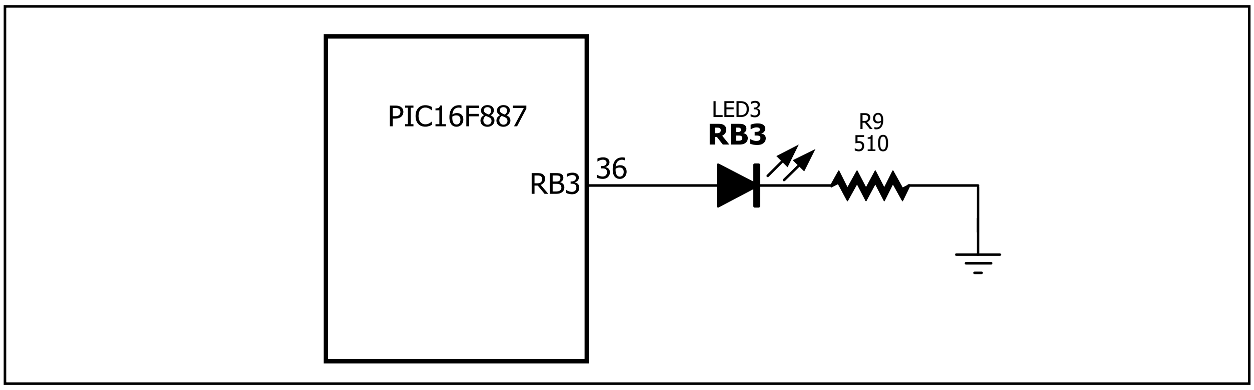


Figure A1-1 : LED connection on RBX-877 V2.0 Robot Controller board

```
void main()
{
  TRISB.F3=0;      // Set RB3 ==> Output
  PORTB.F3=1;      // Turn on RB3
}
```

Listing A1-1 : The C program for Turning on the RB3-LED of the RBX-877 V2.0 Robot Controller board

```
void main()
{
  TRISB.F3=0;      // Set RB3 ==> Output
  while(1)
  {
    PORTB.F3=1;      // Turn on RB3
    Delay_ms(500);
    PORTB.F3=0;      // Turn off RB3
    Delay_ms(500);
  }
}
```

Listing A1-2 : The C program for Blinking the RB3-LED of the RBX-877 V2.0 Robot Controller board

Activity 1-2 Reading digital data via switch and driving sound

See the figue A1-2, it shows the schematic of the switch input of the RBX-877 V2.0 Controller board. The switch tested in this activity is the RA4-switch. If switch is not pressed, DATA point as logic “1” from pull-up resistor 10kΩ. If switch is pressed, DATA point will connect to ground. It causes DATA point is logic “0”. PIC16F887 will drive a sound following the activated swtich at RA4 pin.

Reading switch input programming

The easiest way to check this switch being pressed in C program of mikroC compiler is looping and check with **IF** command. If switch is being pressed, the program will jump to the following condition. In writing the program, you must select the port that interface the switch first.

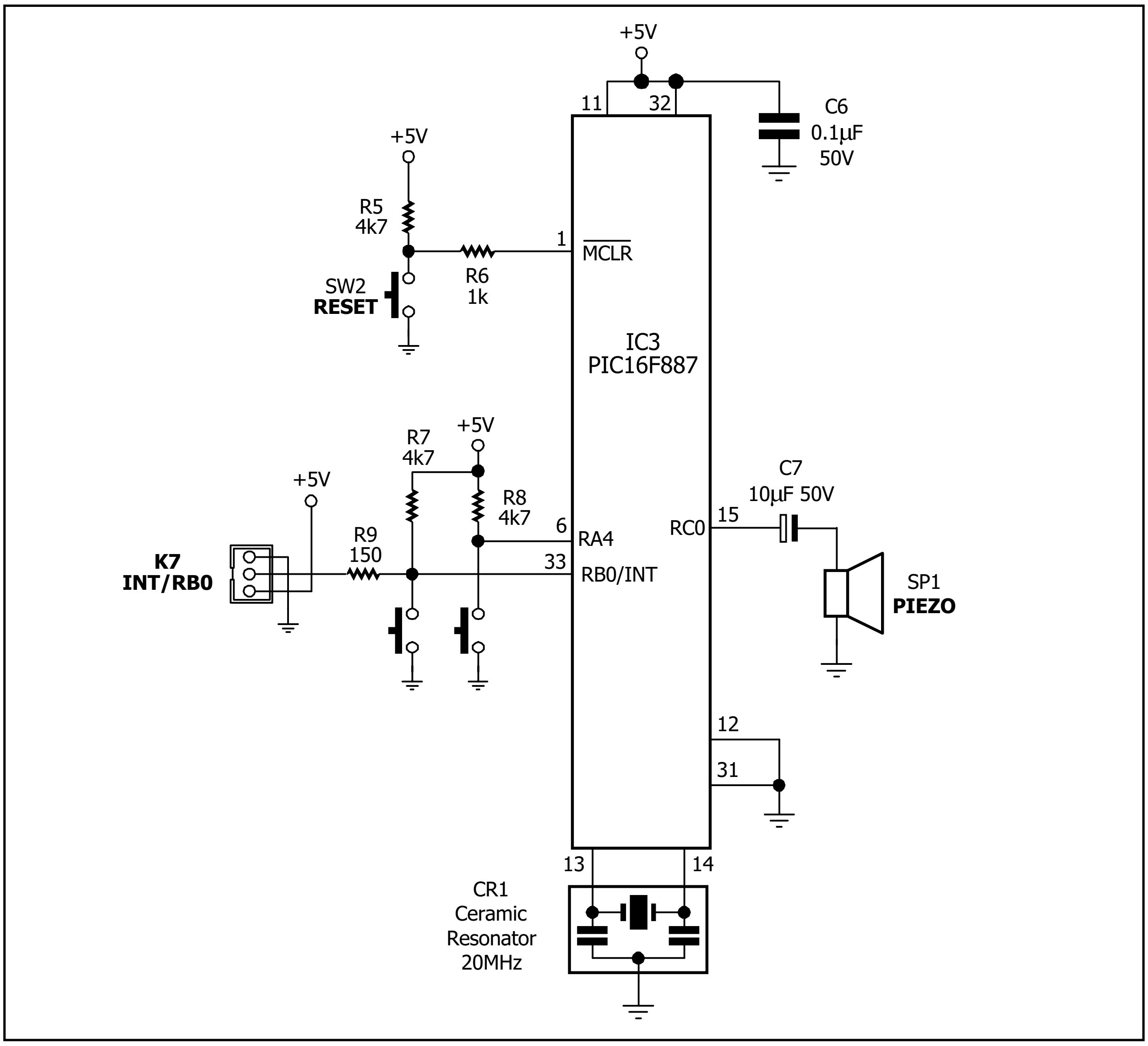


Figure A1-2 : RA4 Switch input schematic of the RBX-877 V2.0 Controller board

Testing

A1.2.1 Write the Listing A1-3. Compile and download the code to RBX-877 board.

A1.2.2 Press the switch at RA4 and observe the operation of the Piezo speaker on the RBX-877 V2.0 Robot Controller board.

Listen sound from the piezo speaker following the switch pressing.

```
void main()
{
 Sound_Init(&PORTC, 0);      // Init Sound
 while(1)
 {
  if (!PORTA.F4)             // Test RA4 keypress
  sound_play(250,50);        // 2kHz sound ON RC0
 }
}
```

Listing A1-3 : The C program of reading digital value from the Switch input at RA4 pin to control the sound generation of Piezo speaker at RC0 pin. The operation is similar the door chime.

Activity 1-3 Show message on LCD module

The RBX-877 V2.0 Robot Controller board provides the connector to interface LCD moudule. The schematic diagram is shown in the Figure A1-3. User must use this information to define in the C program for mikroC compiler knows the port pin that use in this interface.

When interfacing, you wil require 6 port pins which includes the RD2 for RS pin of LCD module, RD3 for E pin and RD4 to RD7 for data pin D4 to D7 in 4-bit interface mode. The R/W pin of LCD is connected to ground for only writing all data to LCD. With this connection, help developers to make the C code for interfacing the LCD module easier. Because you can use the LCD built-in function of mikroC compiler; **Lcd_Init(&PORTD)**.

Testing

A1.3.1 Write the Listing A1-4. Compile and download the code to RBX-877 V2.0 Robot Controller board.

A1.3.2 Observe the operation.

At LCD module show message **Innovative** *on the upper line and* **Experiment** *on the lower line. If need to use the back-light LED, put jumper at LCD backlight position.*

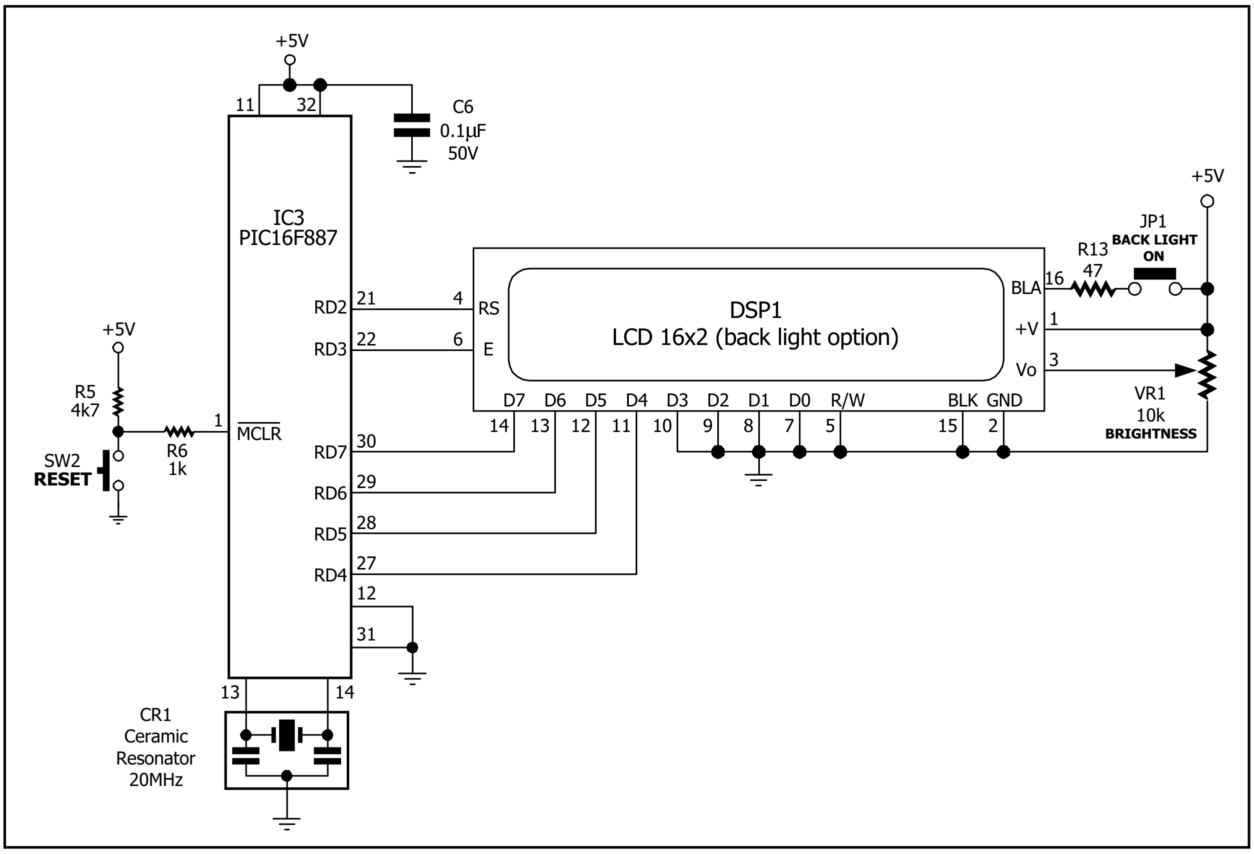


Figure A1-3 : LCD interface schematic of RBX-877 board

```
char *text1 = "Innovative";
char *text2 = "Experiment";

void main()
{
   Lcd_Init(&PORTD);
   Lcd_Cmd(LCD_CURSOR_OFF);
   while(1)
   {
      Lcd_Out(1,1,text1);
      Lcd_Out(2,1,text2);
      Delay_ms(5000);
      Lcd_Cmd(LCD_CLEAR);
      Delay_ms(500);
   }
}
```

Listing A1-4 : The C program for displaying message on LCD module of RBX-877 V2.0 Robot Controller board

Chapter 3
Building Robo-PICA kit

This chapter describes about how to building the Robo-PICA robot kit. The features of Robo-PICA robot kit are as follows :

● Driving with DC motor gearboxes and Track wheel

● Controlled by PIC16F887 microcontroller

● 8KWords program memory

● Re-programmable at least 10,000 times for flash program memory

● Support many types of sensor and detector such as

ZX-01 Switch input board for attacking detection,

ZX-03 Infrared Reflector for line tracking and area,

ZX-IRM Infrared receiver module for remote controlling,

GP2D120 Infrared distance sensor,

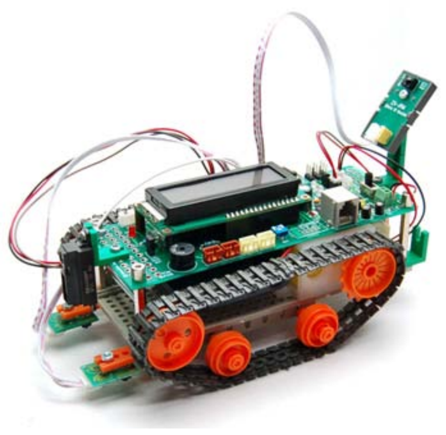
SRF05 Ultrasonic sensor,

CMPS03 Digital compass,

Memsic2125 Accelerometer sensor

and more...

● Provides Character LCD moduel 16x2 and LED status for displaying the robot operation.



Activity 2

Make the Robo-PICA

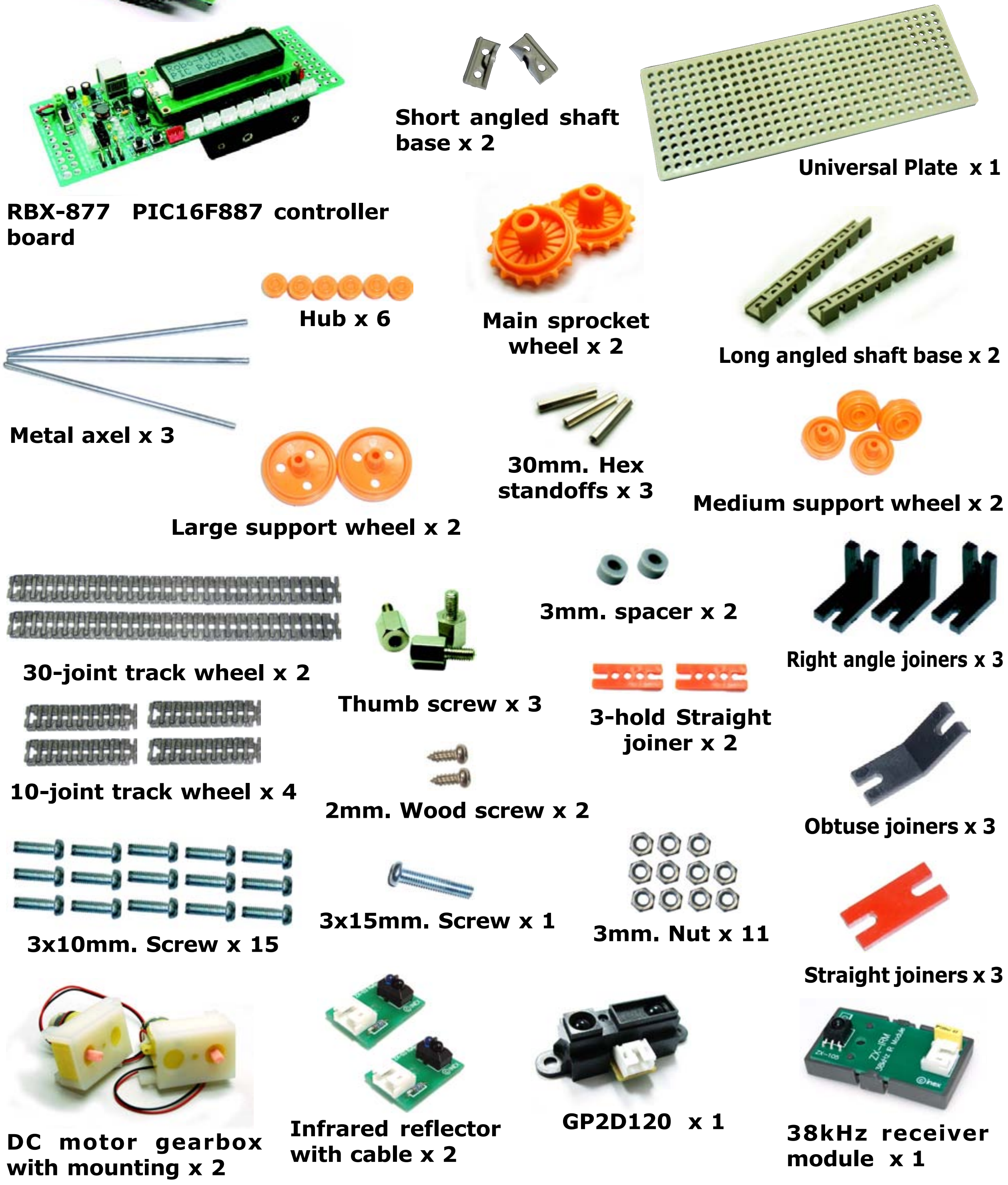
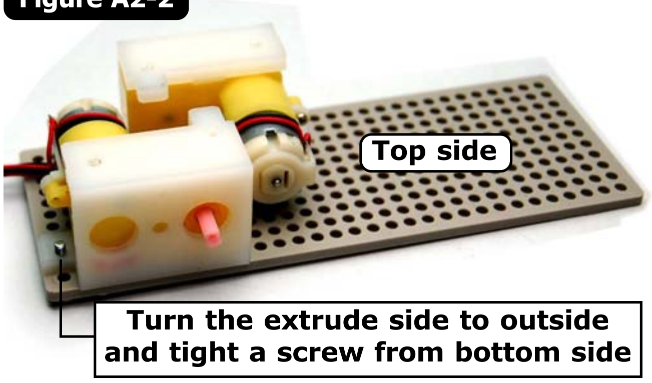


Figure A2-1 : Shows the parts for making a Robo-PICA.

A2.1 Fix 2 of DC motor gearboxes at the base. Turn the extrude side of the right gearbox out side shown in Figure A2-2. Tighten the 3x10mm. screws from bottom side to fix this gearbox. Leave the inside hole of the left gearbox. Do not tighten the screw.

Figure A2-2


Top side
Turn the extrude side to outside and tight a screw from bottom side

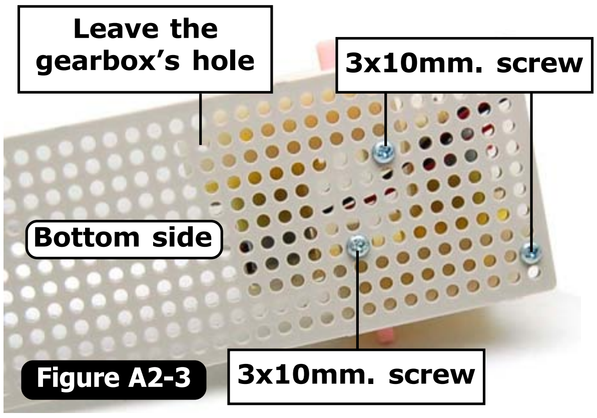
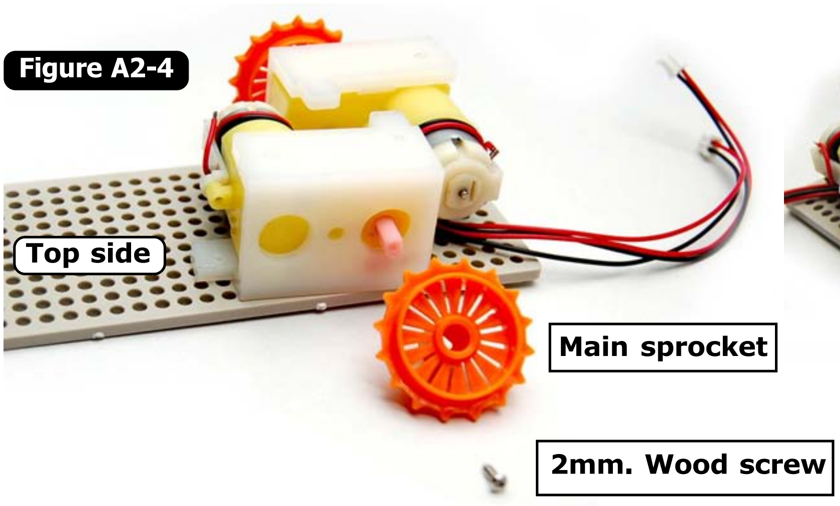


Figure A2-3

A2.2 Insert the main sprocket to the gearbox's shaft and fix with 2mm. Wood screw. Do both DC gearboxes.

Figure A2-4



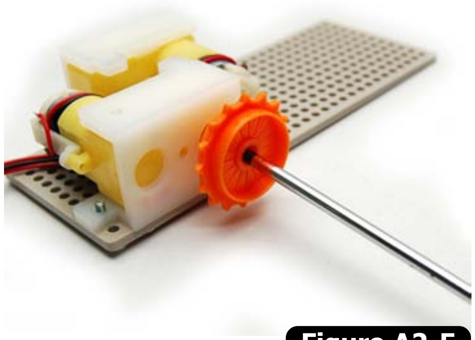


Figure A2-5

A2.3 Put up side down. Attach the Long angled shaft base with the base at the specific position as shown in the figure A2-6. Tighten the 3x10mm. screw to a leave hole from step A2.1.Next, tight a 3x10mm. screw and 3mm. nut to fix the second hole of the Long angled shaft base as shown in the figure A2-6.

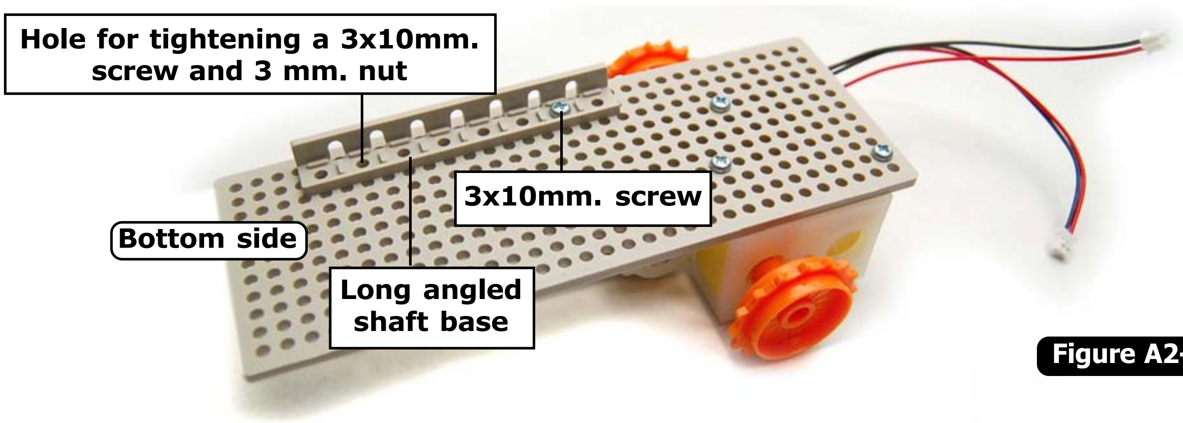


Figure A2-6

A2.4 Attache the rest of Long angled shaft base with a base by inserted the 3x10mm. screws from top side through the hole and tighten with 3mm. nuts following the Figure A2-7.

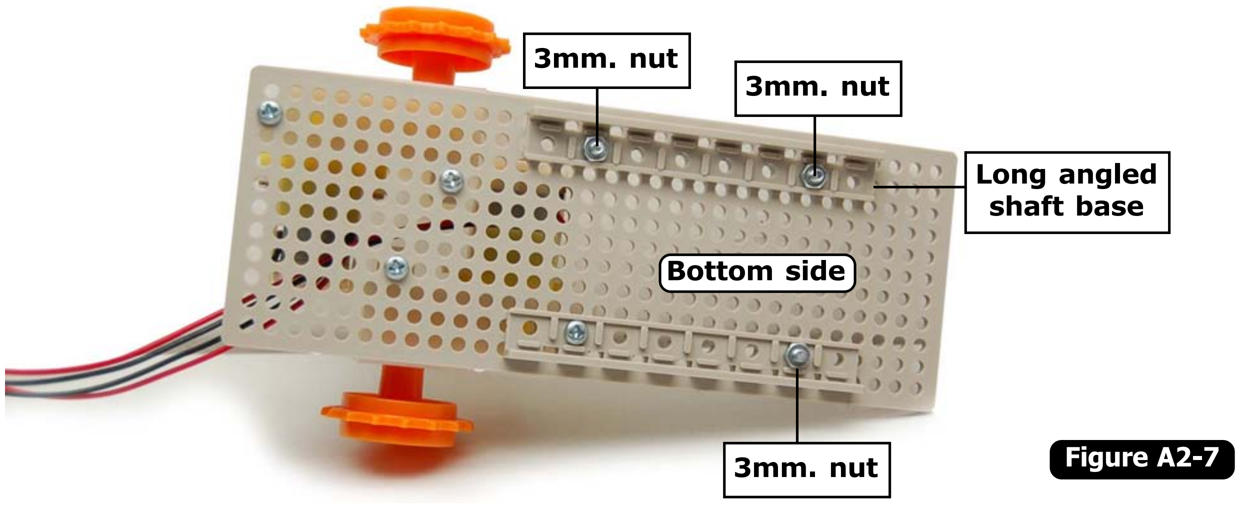


Figure A2-7

A2.5 Turn the base over. Attach 2 of the Short angled shaft bases at the front of the robot's base as shown in the Figure A2-8 by inserted the 3x10mm. screws from bottom side through the shaft bases' holes and tighten with 3mm. nuts. Tighten the screw on the inside hole. Leave the outside holes.

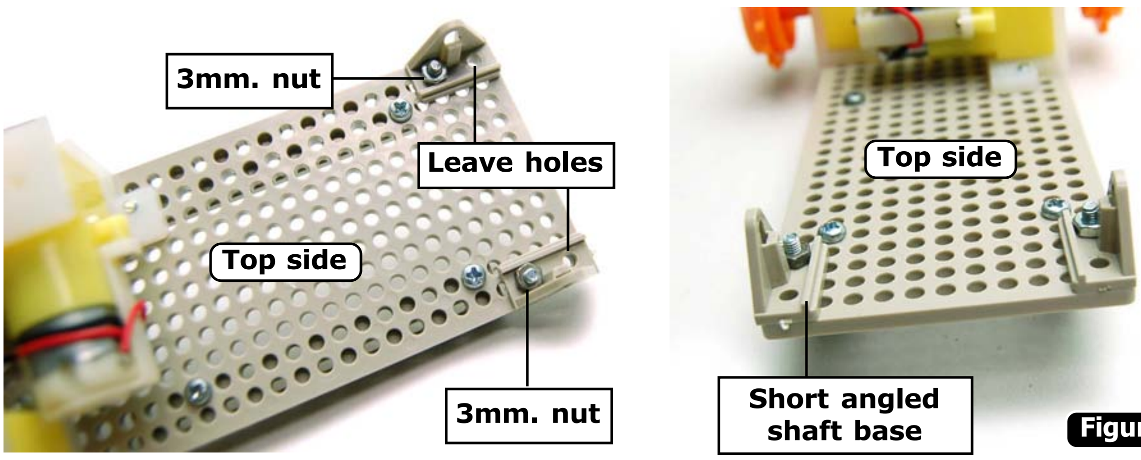
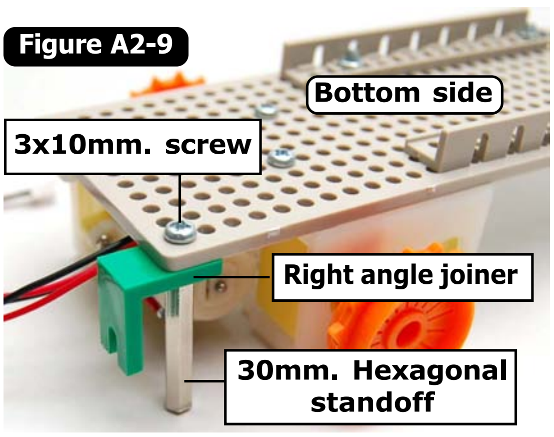
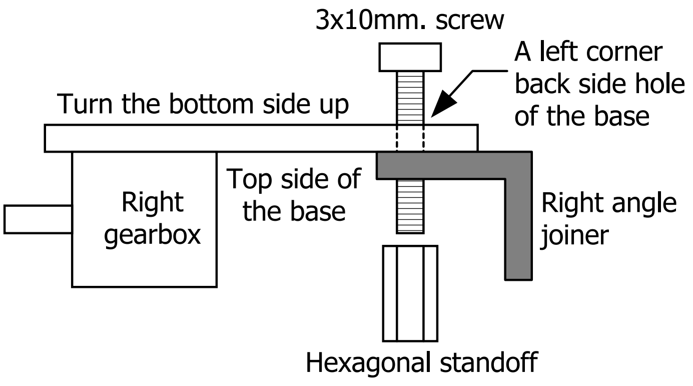


Figure A2-8

A2.6 Fix a Hexagonal standofff at bottom side of base by put upside down and tight a 3x10mm. screw through a left corner hole and the Right angle joiner.

Figure A2-9





A2.7 At front side, attach 2 of the Hexagonal standoffs. Insert the 3x10mm. screw through the 3-hole straight joiner and the leave hole of the short angled shaft base from step A2.5 to fix with the 30mm. Hexagonal standoffs.

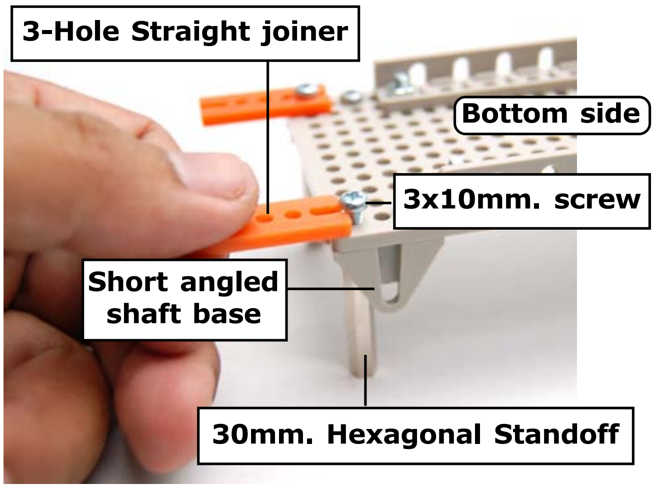


Figure A2-10

A2.8 With the board still upside down, Insert the metal axel into the holes of the long angled shaft in the hole positions of 2 and 6 as shown in the Figure A2-11. Place the Medium track support wheels over the metal axel. Insert the hubs over the wheels so that the wheels and the axels are connected tightly. Turn up the base. Insert the 3rd metal axel into the holes of the short angled shaft. Place the Large support wheel over the axel. Insert the hubs over the wheels so that the wheels and the axels are connected tightly.

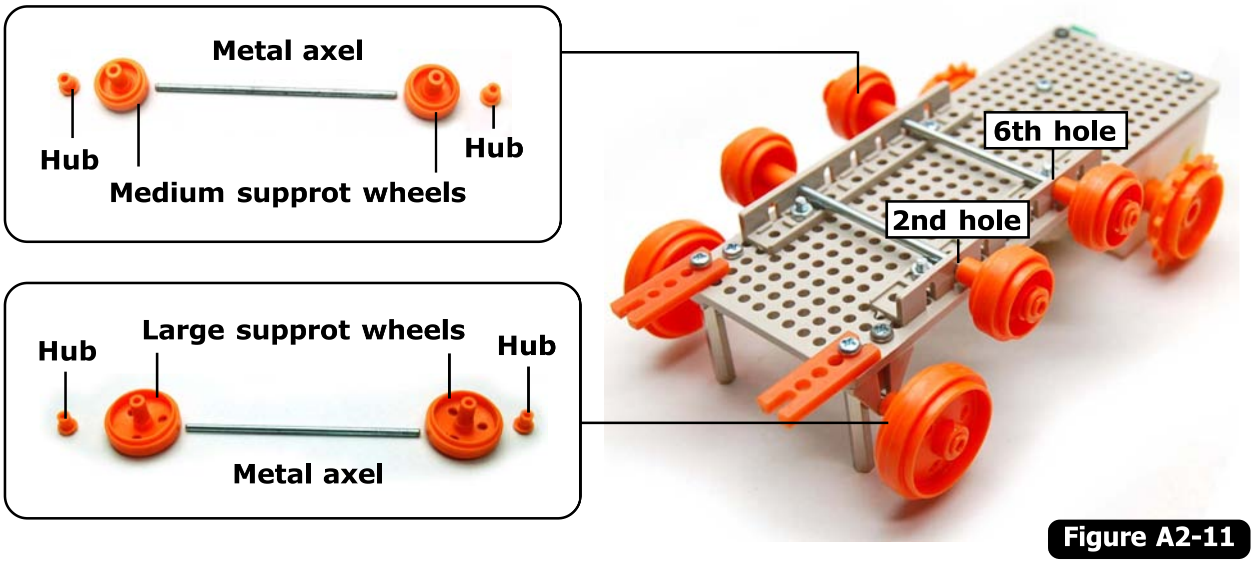


Figure A2-11

A2.9 Create two track belts by putting the different size tracks together. One track would consist of the following: One 30-joint track and two 10-joint tracks. Connect all tracks together. Take one end and connect it to the other end of the track to form one complete loop. Repeat the steps to make two track sets. If the track is too tight or loose, you can either adjust the length of the track or adjust the position of the short angled shaft base until the track has a good fit.

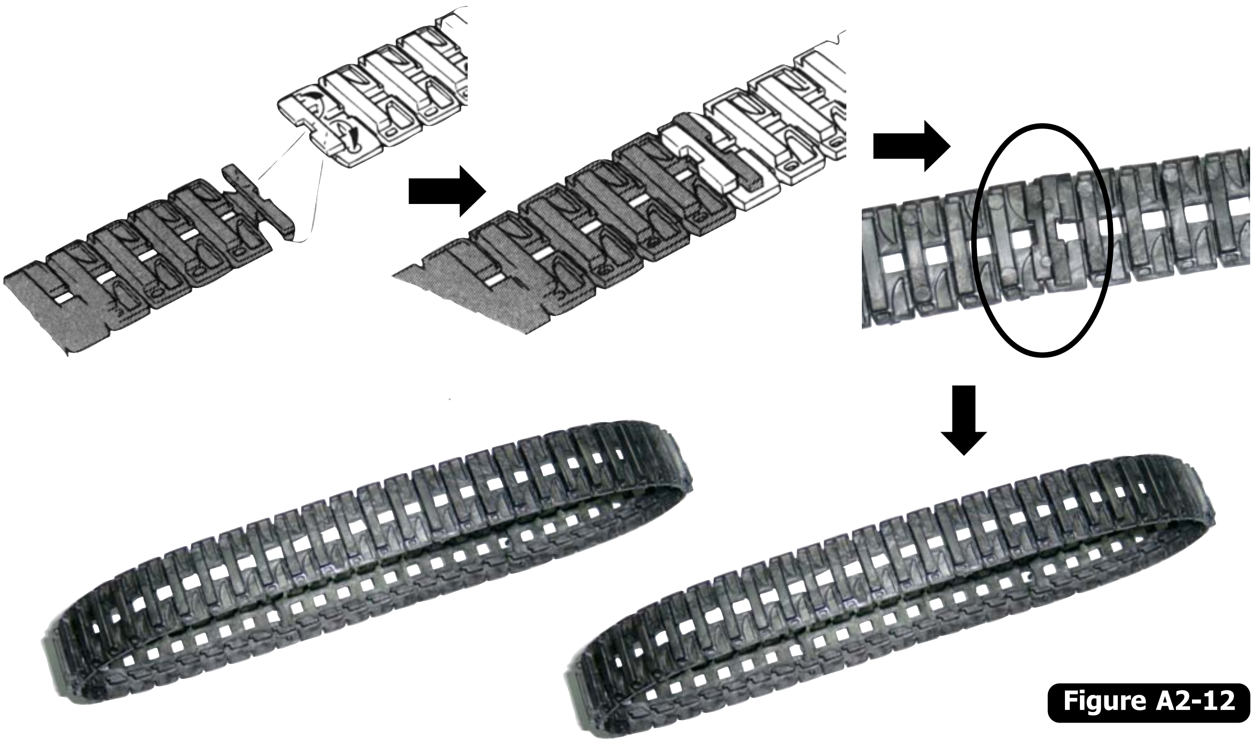


Figure A2-12

The example shown above is only a sample to show you the standard type of track width used. You can of course assemble your own track length based on your own requirements for your robot.

A2.10 Attach the tracks to the supporting wheels of the robot.

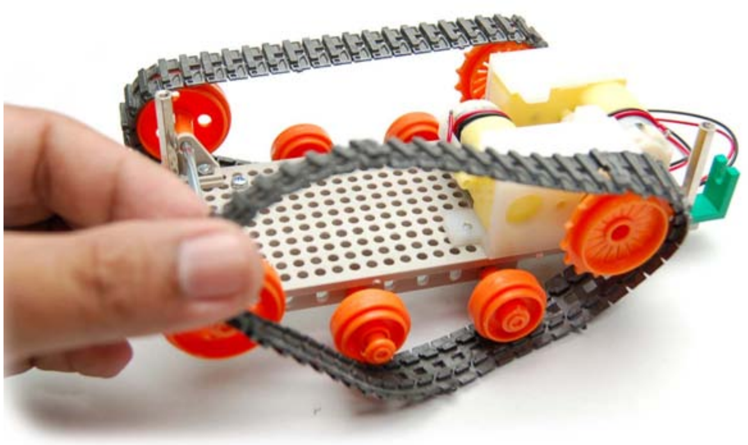


Figure A2-13

A2.11 Attach the RBX-877 V2.0 controller board on top of robot's chasis. Please fix the board with the Power swtch at the side where the DC motor gearboxes are. Secure with 3 Thumb screws at the ends.

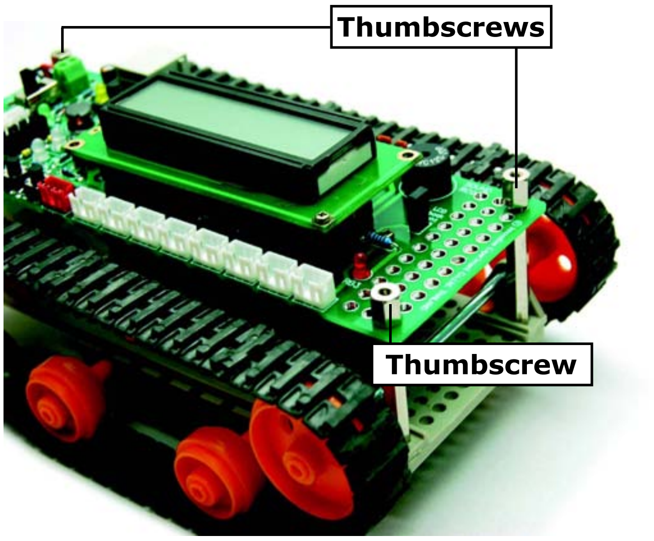


Figure A2-14

A2.12 Attach a ZX-IRM 38kHz Receiver module sensor board with the Obtuse joiner using 3x15mm. screw and 3mm. nut. Insert a Straight joiner at another end of the Obtuse joiner.

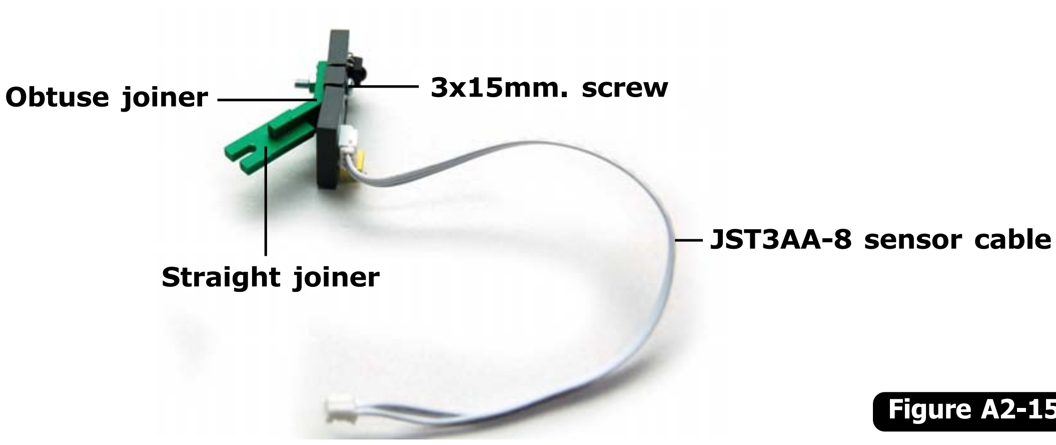


Figure A2-15

A2.13 Attach a Right angle joiner at the center hole of the back side (Power switch side) of the RBX-877 V2.0 controller board by a 3x10mm. screw and 3mm. nut for attaching the ZX-IRM sensor board.

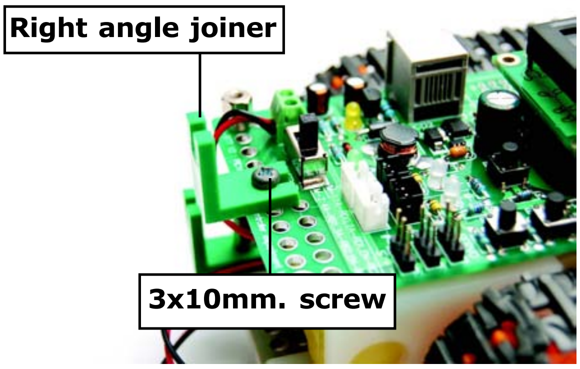


Figure A2-16

A2.14 Connect the ZX-IRM structure from step A2.12 to the Right angle joiner on the RBX-877 V2.0 controller board from step A2.13. Plug in the Zx-IRM sensor cable to RB0/INT connector.

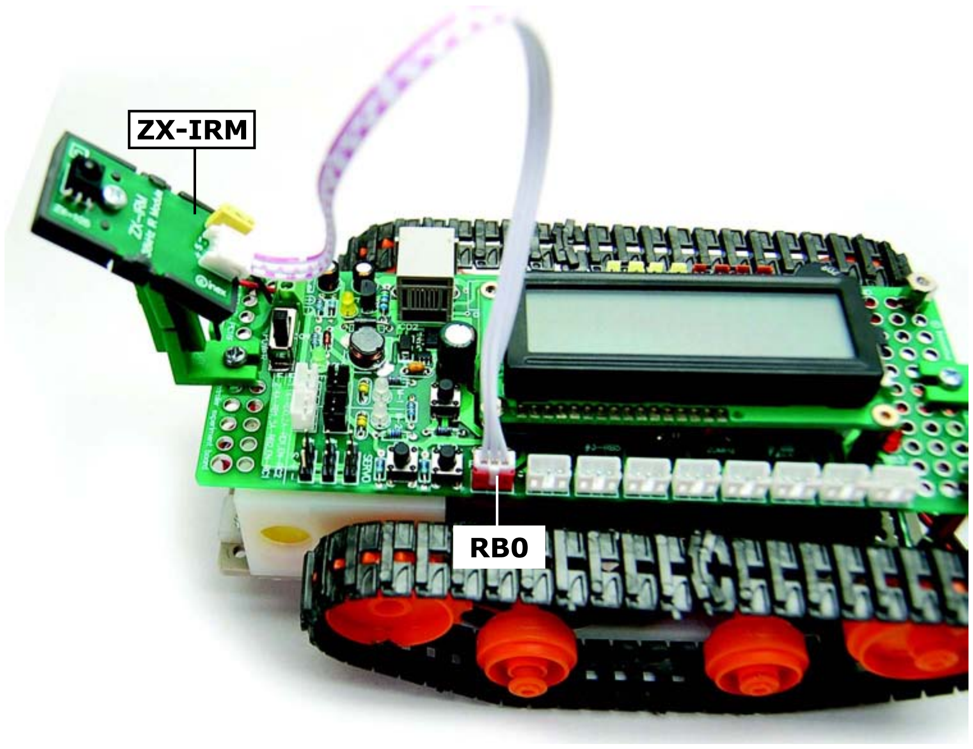


Figure A2-17

A2.15 Plug the DC motor gearboxes cable to Motor connectors. The right motor is connected to the white M-2 output and left motor is connected to the black M-1 output. However the motor's pole (white or black connector) can be changed depending on the programming and mission. Normally, refer from the motor output's indicator, if both light green, it means the forward movement and both light red mean the backward movement. You can change later if the operation incorrect.

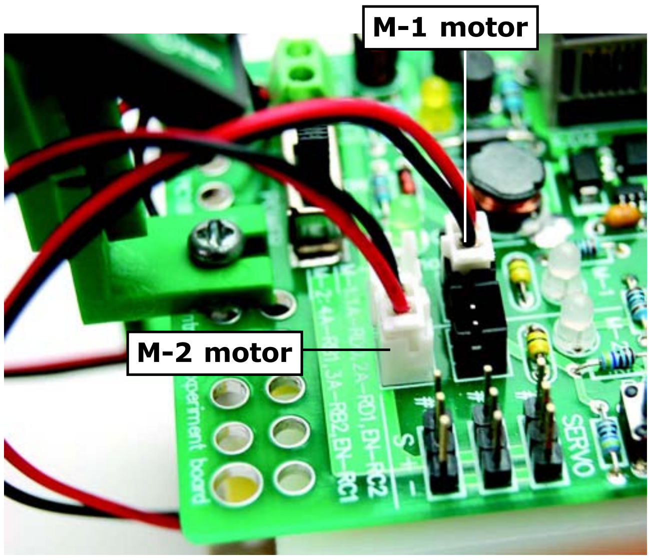


Figure A2-18

A2.16 Install the ZX-03 Infrared reflector sensor board at the bottom of the robot's chasis. Attach the sensor with the end hole of the 3-hole Straight joiner by inserted the 3x10mm. screw through the sensor board, 3mm. plastic spacer, joiner and tighten with 3mm. nut. Install both side; left and right.

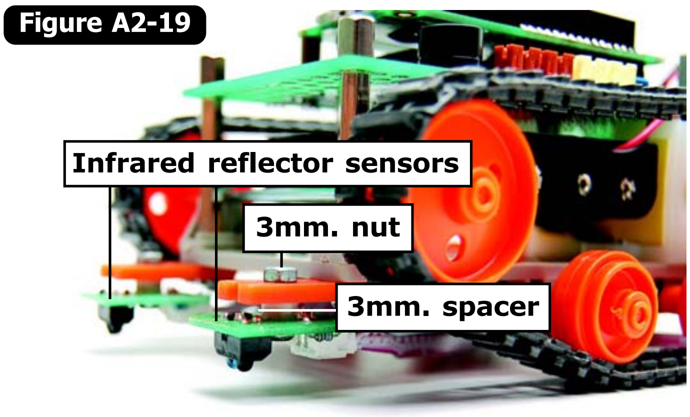


Figure A2-19

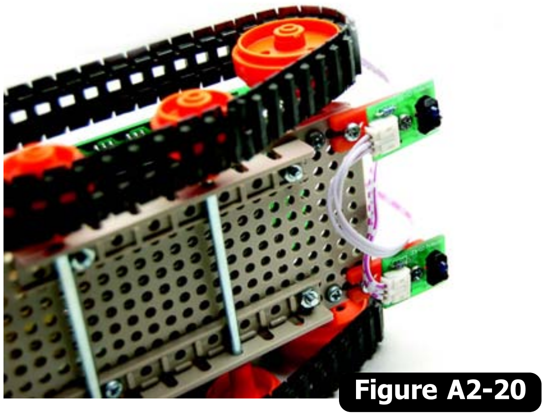


Figure A2-20

A2.17 Attach a GP2D120 module with a Right angle joiner as shown in the Figure A2-21 by 3x10mm. screw and 3mm. nut.

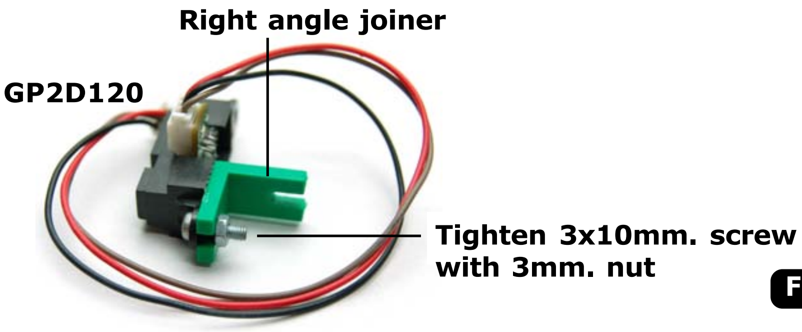


Figure A2-21

A2.18 At the front of robot, insert a 3x10mm. screw through a center hole position of the RBX-877V2.0 board and 3mm. nut from top side as shown in the Figure A2-22. Do not tighten. Next, Insert the GP2D120 structure from step A2.17 between a screw and controller board (see the Figure A2-23). Tighten the screw to fix all together.

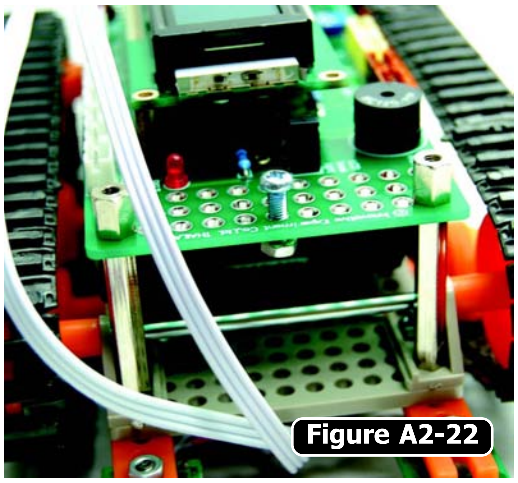
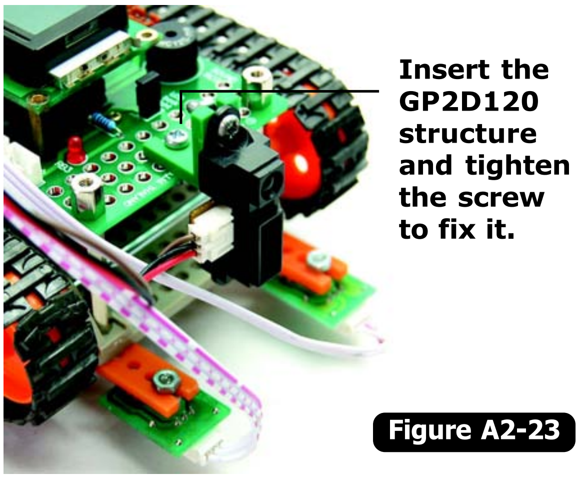
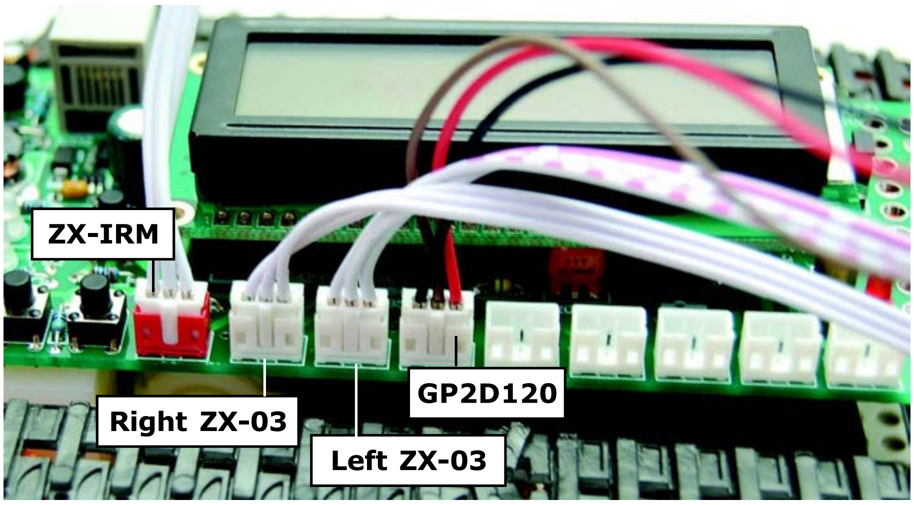


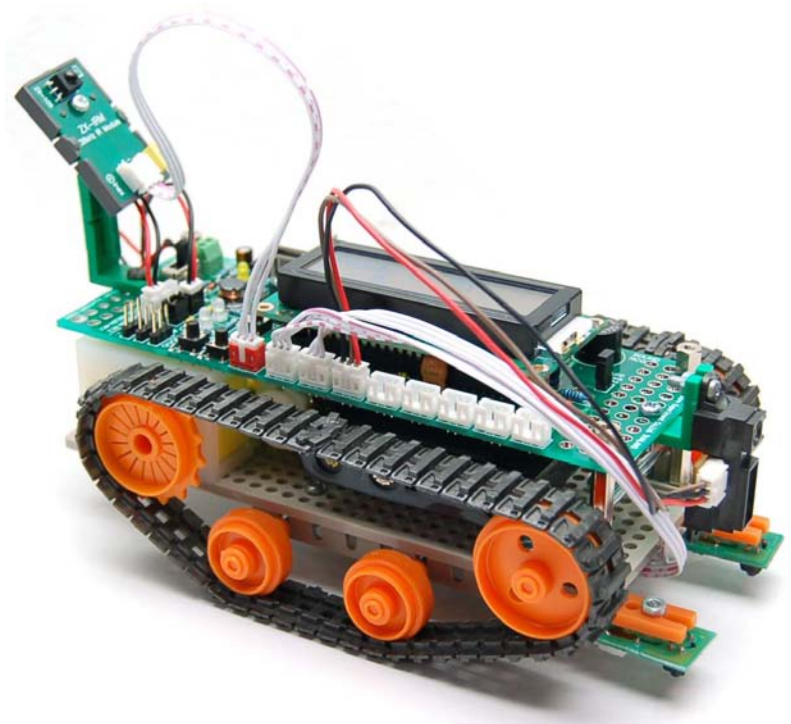
Figure A2-22


Insert the GP2D120 structure and tighten the screw to fix it.
Figure A2-23

A2.19 Plug the GP2D120 cable to RA2 port, the left ZX-03 sensor's cable to RA0 port and the right ZX-03 sensor's cable to RA1 port.



A2.20 Arrange all cables and check all connection carefully. **Your Robo-PICA is now ready for programing.**



Chapter 4

Simple robot 's programming control

The first thing is to control robot Movement. The heart of this movement is DC motor circuit. In Robo-PICA has DC motor gearbox in driving. The Figure 4-1 shows the DC motor circuit. PIC16F887 assigns 6 port pins to connect the DC motor driver circuit for driving 2 motors.

The motor driving mechanism are divided into 4 types as follows :

(1) Clockwise motor driving

(2) Anti-clockwies motor driving

(3) Motor's shaft is free

(4) Motor's shaft is locked or Braked

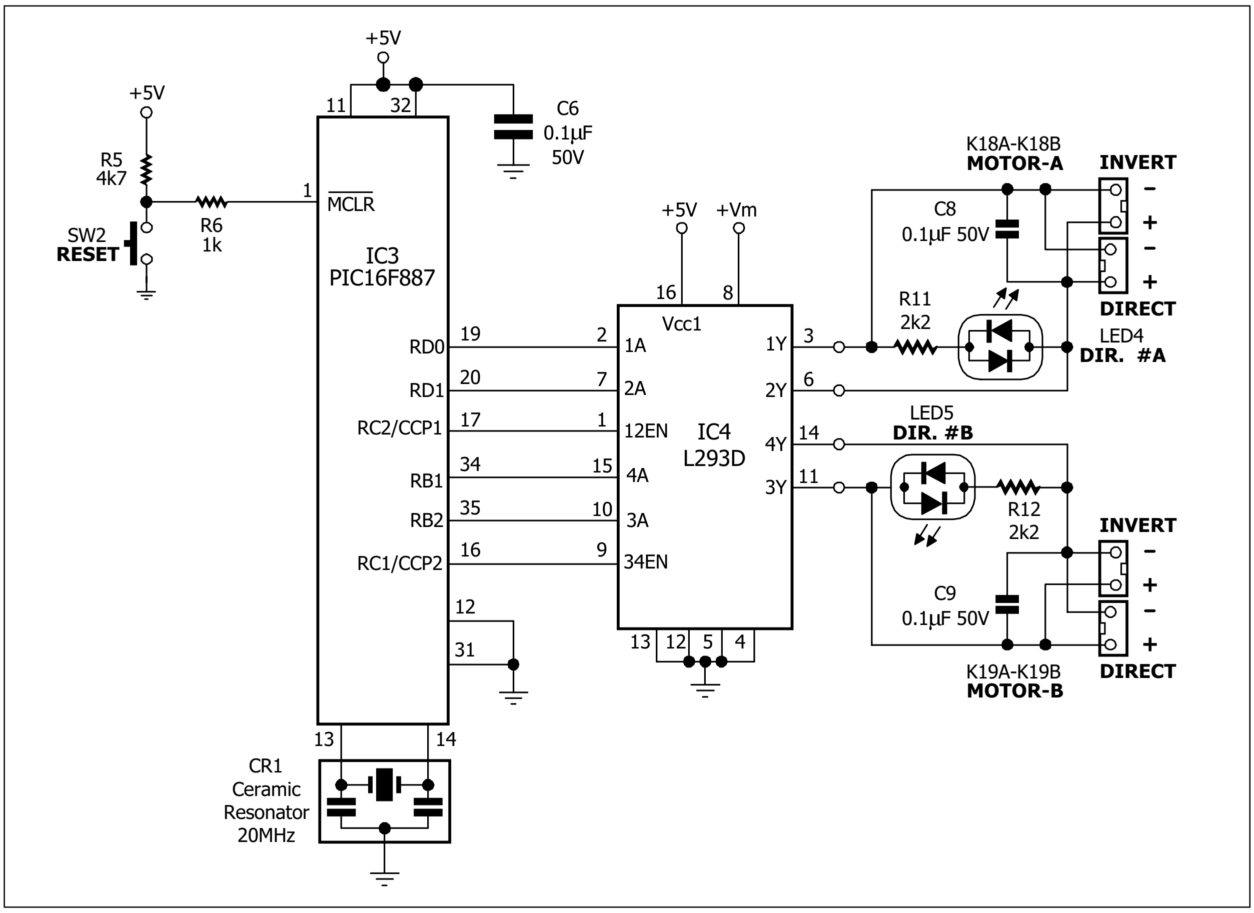


Figure 4-1 : The DC motor driver schematic of RBX-877 V2.0 board

12EN/34EN pin	1A/3A pin	2A/4A pin	Motor operation
0	X	X	Shaft free
1	0	0	Shaft locked or Brake
1	0	1	Clockwise turning
1	1	0	anti-clockwise turning
1	1	1	Shaft locked or Brake

X means logic "0" or "1"

Table 4-1 : Shows logic signal to control motor direction

The heart of DC motor driver circuit is the L293D H-Bridge driver (may be replaced by SN754410). In the Table 4-1 shows all the required signals to control the DC motor driver circuit.

L293D outputs connects to DC motor gearbox and provides LED status for motor supply voltage. If power is supplied DIRECTLY, the LED will light up in Green. When it is opposite, if red LED lights up, it means the supply voltage is INVERTED. Builders can use the different color for defining direction. In other words, if red LED are turned on, the robot will be moving backwards. If the green LED are turned on, the robot will be moving forwards.

4.1 Motor library file

For better performance and ease of programming, we make the library for driving and movement controls for the DC motors. It is the ***motor.h*** library file. The souce code of this library is shown in Listing 4-1.

You can use simple text editor t ocreate this library and save as .h file or open mikroC IDE to create this file. After that copy this library file to the library folder of mikroC software. The location is ***C:\Program Files\Mikroelektronika\mikroC\include***. You must copy the ***motor.h*** file to this folder. Because the complier will link to this folder for including any library.

motor.h library file consists of many functions of movement control. Include :

`Motor_Init` : Initial the micrococontroller port pin for interfacing the DC motor driver circuit.

`Change_Duty` : Control the motor's speed.

`Motor_A_FWD` : Drive motor A (M-1 output) to forward direction (LED indicates of M-1 lights in green).

```
char motor_duty_= 127;              // Defalt PWM 50%
char motor_init_=0;                 // Status initial

//   *** Motor A *****
//    PD0 ====>  1A
//    PD1 ====>  1B
//    PC2 ====>  1E (PWM1)

//   *** Motor B *****
//    PB1 ====>  2A
//    PB2 ====>  2B
//    PC1 ====>  2E (PWM2)

//*****************************************************
//********** Initial Motor Function *****************
//*****************************************************
void Motor_Init()
{
  if (motor_init_==0)              // First time ?
  {
    motor_init_=1;                 // Status
    ANSELH.F0=0;                   // RB1 ==> Digital IO
    ANSELH.F2=0;                   // RB2 ==> Digital IO
    TRISB.F1=0;                    // Motor B 2A
    TRISB.F2=0;                    // Motor B 2B
    TRISD.F0=0;                    // Motor A 1A
    TRISD.F1=0;                    // MOtor A 1B
    Pwm1_Init(5000);               // Initail PWM 1E
    Pwm2_Init(5000);               // Initail PWM 2E
  }
}
//*****************************************************

//*****************************************************
//********** Control Duty Cycle  ********************
//*****************************************************
void Change_Duty(char speed)
{
 if (speed != motor_duty_)        // Check Same old speed
 {
   motor_duty_=speed;             // Save for old speed
   Pwm1_Change_Duty(speed);       // Motor A
   Pwm2_Change_Duty(speed);       // Motor B
 }
}
//*****************************************************

/********** Motor A Forward  ********/
void Motor_A_FWD()
{
  Pwm1_Start();
  PORTD.F0 =0;
  PORTD.F1 =1;
}
/*************************************/
```

Listing 4-1 : The source code of motor.h library file for driving the DC motor (continue)

```
/********** Motor B Forward  ********/
void Motor_B_FWD()
{
  Pwm2_Start();
  PORTB.F1 =0;
  PORTB.F2 =1;
}
/************************************/

/********** Motor A Backward  *******/
void Motor_A_BWD()
{
  Pwm1_Start();
  PORTD.F0 =1;
  PORTD.F1 =0;
}
/************************************/

/********** Motor B Backward  *******/
void Motor_B_BWD()
{
  Pwm2_Start();
  PORTB.F1 =1;
  PORTB.F2 =0;
}
/************************************/

/********** Motor A Off  ************/
void Motor_A_Off()
{
  Pwm1_Stop();
  PORTD.F0 =0;
  PORTD.F1 =0;
}
/************************************/

/********** Motor B Off  ************/
void Motor_B_Off()
{
  Pwm2_Stop();
  PORTB.F1 =0;
  PORTB.F2 =0;
}
/************************************/

/********** Go Forward   ************/
void Forward(char speed)
{
    Motor_Init();
    Change_Duty(speed);
    Motor_A_FWD();
    Motor_B_FWD();
}
/************************************/
```

Listing 4-1 : The source code of motor.h library file for driving the DC motor (continue)

```
/********** Go Backward  ************/
void Backward(char speed)
{
    Motor_Init();
    Change_Duty(speed);
    Motor_A_BWD();
    Motor_B_BWD();
}
/************************************/

/********** Spin Left   *************/
void S_Right(char speed)
{
    Motor_Init();
    Change_Duty(speed);
    Motor_A_FWD();
    Motor_B_BWD();
}
/************************************/

/********** Spin Right   ************/
void S_Left(char speed)
{
    Motor_Init();
    Change_Duty(speed);
    Motor_A_BWD();
    Motor_B_FWD();
}
/************************************/

/********** Stop Motor   ************/
void Motor_Stop()
{
    Motor_Init();
    Change_Duty(0);
    Motor_A_Off();
    Motor_B_Off();
}
/************************************/
```

Listing 4-1 : The source code of motor.h library file for driving the DC motor (final)

Motor_B_FWD : Drive motor B (M-2 output) to forward direction (LED indicates of M-2 lights in green).

Motor_A_BWD : Drive motor A (M-1 output) to backward direction (LED indicates of M-1 lights in red).

Motor_B_BWD : Drive motor B (M-2 output) to backward direction (LED indicates of M-2 lights in red).

Motor_A_off : Turn off or Stop motor A (M-1 output).

Motor_B_off : Turn off or Stop motor B (M-2 output).

foward : Drives both DC motror to move the Robo-PICA forward.

backward : Drives both DC motror to move the Robo-PICA backward.

S_right : Drives both DC motror to spin the Robo-PICA in right direction.

S_left : Drives both DC motror to spin the Robo-PICA in left direction.

Motor_stop : Stop both DC motror.

Activity 3
Simple movement control

Robo-PICA moves forward or backward by driving both DC motor gearboxes in same direction and at the same time. If need to turn or rotate, below shows the method :

1. **Stop one motor and Drive another one** If stop left motor and drive right motor, the robot will turn left. In the opposite direction, stop right motor and drive left motor. The robot will turn right. The speed of movement is similar. The pivot turning point of this is at the stationary track. See the Figure A3-1.

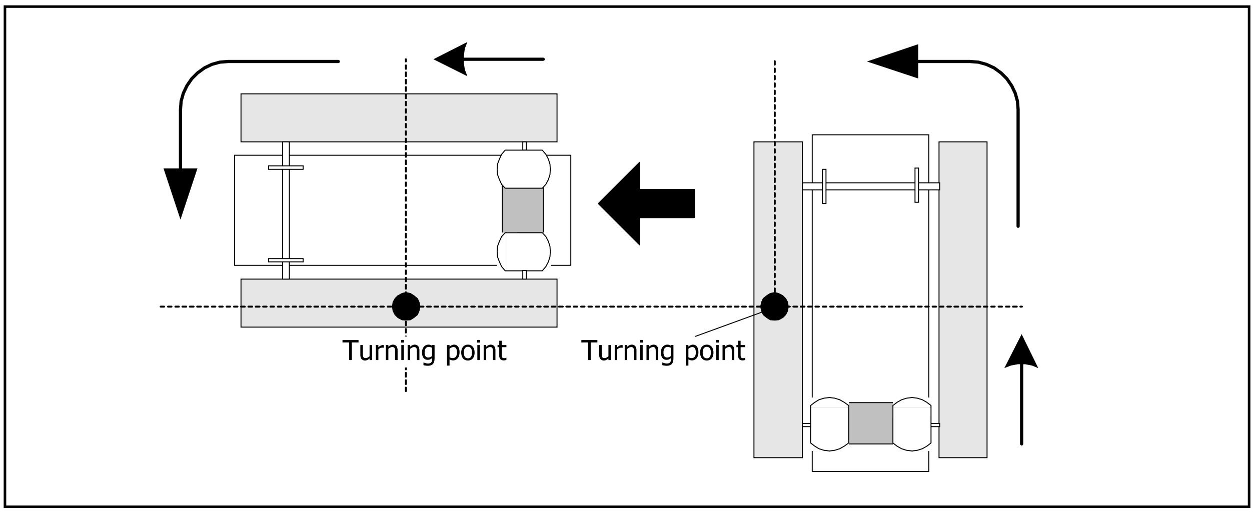


Figure A3-1 Turning method by stopping a motor and fix a wheel.

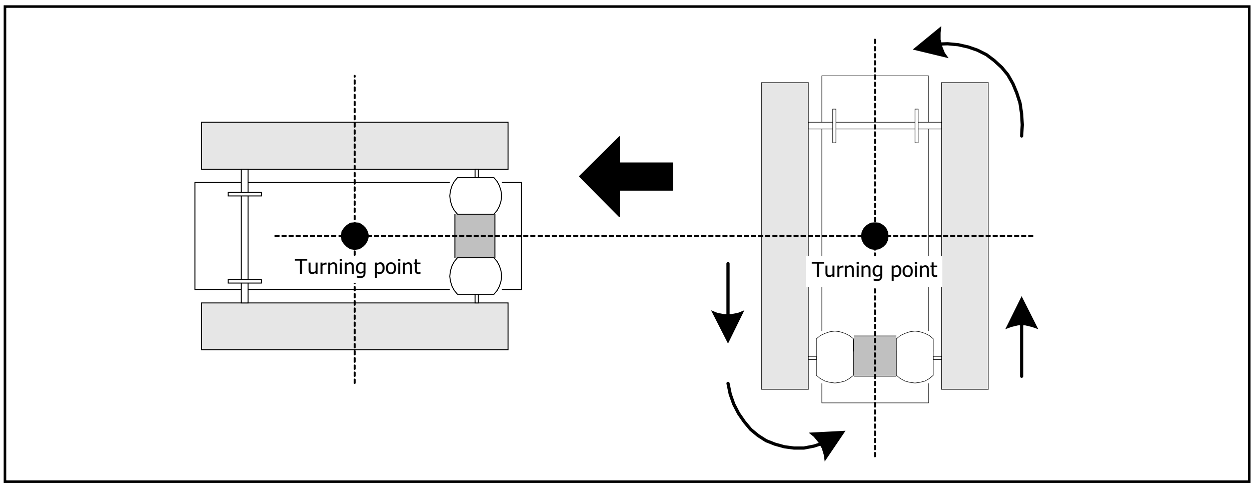


Figure A3-2 Turning method by driving both motors in opposite direction.

2. **Drive both motors in opposite direction** If the left motor drives forward and right motor drives backward, the robot will rotate right direction. If its in the opposite direction, the left motor drives backward and right motor drives forward. The robot will rotate left direction instead. In this method the speed of rotation will be increase 2 times and less friction. The turning point is center of robot body. See Figure A3-2.

A3.1 Write program following the Listing A3-1 then compile and download to RBX-877 V2.0 Robot Controller board. Turn-off power switch.

```
#include <motor.h>
void main()
{
 Sound_Init(&PORTC, 0);      // Init Sound
 while(1)
 {
  Forward(255);              // Call Forward
  Delay_ms(2000);
  sound_play(100,50);        // 1 kHz sound ON RC0

  S_Left(255);               // Call Spin Left
  Delay_ms(800);
  sound_play(100,50);        // 1 kHz sound ON RC0

  Forward(255);              // Call Forward
  Delay_ms(2000);
  sound_play(100,50);        // 1 kHz sound ON RC0

  S_Right(255);              // Call Spin Right
  Delay_ms(800);
  sound_play(100,50);        // 1 kHz sound ON RC0

  Forward(255);              // Call Forward
  Delay_ms(2000);
  sound_play(100,50);        // 1 kHz sound ON RC0

  Backward(255);             // Call Backward
  Delay_ms(1000);
  sound_play(100,50);        // 1 kHz sound ON RC0
  Motor_Stop;                // Stop all
 }
}
```

Listing A3-1 The movement program demonstration of Robo-PICA

A3.2 Remove the downlaod cable from Robo-PICA. Place the robot on the floor. Turn-on power to run the program. See the operation.

The robot will move forward 2 seconds and spin left 0.8 second and foward 2 seconds again. Next, it will spin right 0.8 second to change the direction and forward 2 seconds, moves backward 1 second and stop movement finally. In each changing movement, thr robot will beep a sound to report the operation.

However it is possible the robot moves in an incorrect direction. If this happens, pleae check the motor cable connection. You can change the motor connection from black to white connector and white to black connector in each motor output.

You can see the LED indicator of DC motor output. During forward movement, both LEDs must light Green color. In backward movement, both LEDs light Red color. Must change until the movement direction is corrected and remember or fix the correct connection for all activities onwards.

This is the limitation of malfunctioning, We do not know about the correct pole of DC motor. But we can control and fix with hardware and software via DC motor control circuit. This problem can be easily fixed and it is important to know and understading this.

Because the robot use battery to power source. In during the battery level is full power and not full, the speed of movement is not equal. It cause the distance from movement may be not equal. It is limitation of all robot that use open loop movement control.

Activity 4
Speed control of Robo-PICA

Robo-PICA can control the speed movement by send the signal to the enable pin (EN) of motor driver IC, L293D. Refer the figure 4-1 (in this chapter), EN pin of L293D is connected to RC2/CCP1 and RC1/CCP2 port pins of PIC16F887. Both port pins are PWM output port. Builders can write the program to control the PWM output signal for adjustment motor speed.

PWM operation

Normal driving motor technique is apply the voltage to motor directly. The motor works in full speed. Sometime this speed faster. Then the simple method to control motor speed is control the voltage applied to motor. The populate technique is PWM (pulse-width modulation). This technique will control the width of the positive pulse. The voltage is applied to motor as average value. Ratio of positive pulse width and totally pulse width is called Duty cycle. Its unit is percentage (%)

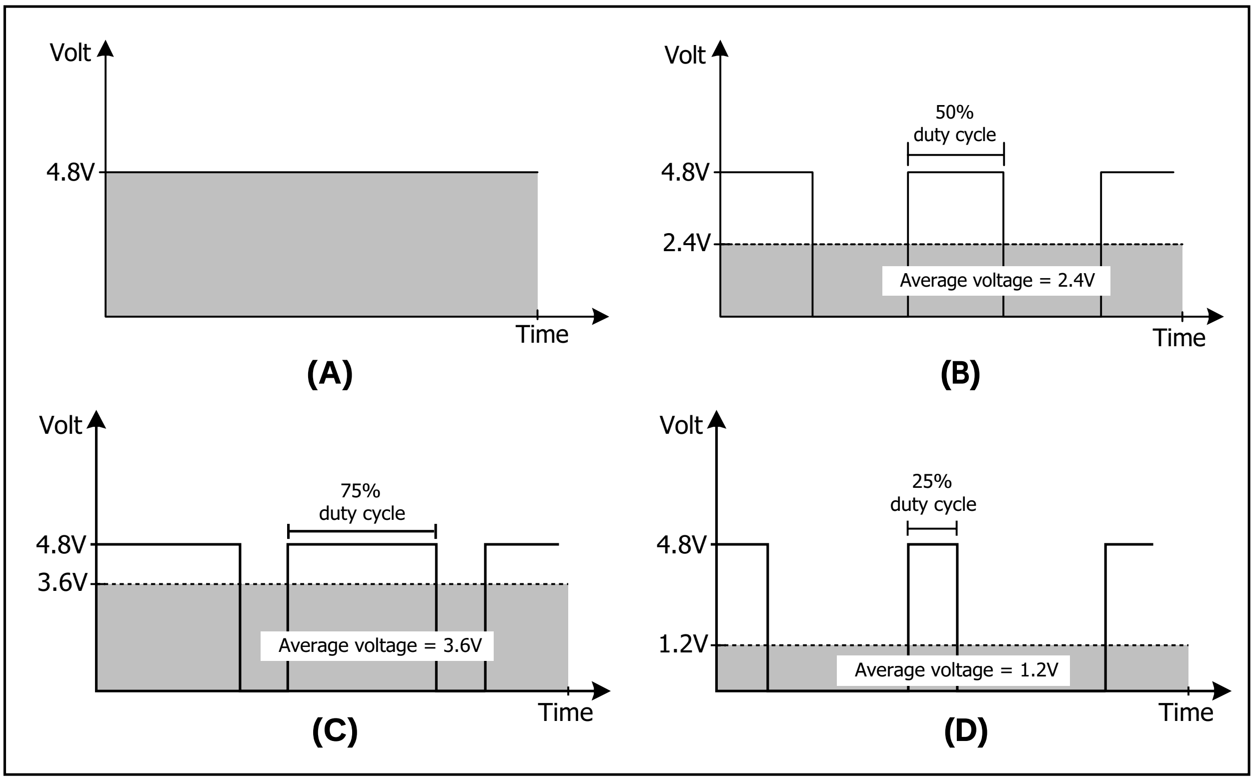


Figure A4-1 : Shows average voltage output of PWM
(A) Full volatge apply. (B) 50% duty cycle PWM
(C) 75% duty cycle PWM (D) 25% duty cycle PWM

```
#include <motor.h>
char i;
void main()
{
     Forward(255);                    // Motor Forward
     while(1)
     {
          Delay_ms(2000);
          Pwm1_Change_Duty(220);     // Motor A  85% Duty
          Pwm2_Change_Duty(255);     // Motor B  100% Duty
          Delay_ms(5000);
          Pwm1_Change_Duty(255);     // Motor A 100% Duty
     }
}
```

Listing A4-1 : The speed control program for Robo-PICA by using PWM.

For Robo-PICA, we prepare the speed control with PWM technique via software by the ***motor.h*** library file. You can see detail in motor.h sourcecode in Listing 4-1 (in this chapter). ***motor.h*** library has PWM function for supporting 2 PWM modules of PIC16F887 as follows :

```
Pwm1_Change_Duty(speed);    // Motor A  Duty
Pwm2_Change_Duty(speed);    // Motor B  Duty
```

You can put the required duty cycle value in **(speed)**. The range is 0 to 255 for 0 to 100% duty cycle.

A4.1 Write the Listing A4-1. Compile and download the code to Robo-PICA. Turn-off power switch.

A4.2 Remove the downlaod cable from Robo-PICA.

A4.3 Place the robot on the floor. Turn-on power to run the program. See the operation.

The Robo-PICA robot will move forward fastest in 2 seconds and spin left 5 second. After that the robot will move foraward with fastest speed again. The robot will move this routine all times.

i

The suitable PWM duty cycle value for driving the robot is more than 70%. *If select the less value, the robot has not torque more enough for turning or rotation.*

Chapter 5
Contactless object detection

The one of most important function of mobile robot is interfacing the sensors. Robo-PICA can interface with many type of sensors. Because it has both digital and analog inputs. PIC16F887 the main microcontroller of Robo-PICA has many ports. We assign 9 programmable port pins for supporting the analog and digital sensors. In addtion 2 types of serial coomunication ports; UART and I^{2}C bus.

In this chapter, we will concentrate to interfacing with angalog sensors. The Robo-PICA kit provides 2 kinds of analog sensors ; GP2D120 the infrared distance sensor and ZX-03 Infrared Reflector sensors for line tracking activities.

5.1 PIC16F887's A/D converter

PIC16F887 microconttroller contains 14-channel 10-bit analog to digital converter module (ADC). All analog input ports can be configured to digital input and output. They include RA0 to RA3, RA5, RB0 to RB5 and RE0 to RE2.

The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESL and ADRESH).

The ADC voltage reference is software selectable to either VDD or a voltage applied to the external reference pins.

5.2 ADC register

The important register of this module are **ADCON0** and **ADCON1** register**.** The ADCON0 is used to select the analog pin fucntion and ADCON1 is used to select the result data format and voltage reference.

5.2.1 ADCON0 : A/D Control register 0

Detail of each bit in ADCON0 register is shown below.

bit	7	6	5	4	3	2	1	0
	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

bit 7 and 6 - ADCS1, ADCS0 : A/D Conversion Clock Select bits

00 = FOSC/2

01 = FOSC/8

10 = FOSC/32

11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)

bit 5 to 2 - CHS3 to CHS0 : Analog Channel Select bits

0000 = AN0 (RA0 pin)

0001 = AN1 (RA1 pin)

0010 = AN2 (RA2 pin)

0011 = AN3 (RA3 pin)

0100 = AN4 (RA5 pin)

0101 = AN5 (RE0 pin)

0110 = AN6 (RE1 pin)

0111 = AN7 (RE2 pin)

1000 = AN8 (RB2 pin - reserve for DC motor circuit of the RBX-877V2.0 Robot Controller board)

1001 = AN9 (RB3 pin - reserve for LED monitor of the RBX-877V2.0 Robot Controller board)

1010 = AN10 (RB1 pin - reserve for DC motor circuit of the RBX-877V2.0 Robot Controller board)

1011 = AN11 (RB4 pin - reserve for servo motor output of the RBX-877V2.0 Robot Controller board)

1100 = AN12 (RB0 pin - alternative function wih External Interrput and Swtich of the RBX-877V2.0 Robot Controller board)

1101 = AN13 (RB5 pin - reserve for servo motor output of the RBX-877V2.0 Robot Controller board)

1110 = CVREF

1111 = Fixed Ref (0.6 volt fixed reference)

bit 1- GO/DONE: A/D Conversion Status bit

1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle. This bit is automatically cleared by hardware when the A/D conversion has completed.

"0" = A/D conversion completed/not in progress

bit 0 - ADON: ADC Enable bit

1 = ADC is enabled

0 = ADC is disabled and consumes no operating current

5.2.2 ADCON1 : A/D Control register 1

Detail of each bit in ADCON1 register is shown below.

bit	7	6	5	4	3	2	1	0
	ADFM	-	VCFG1	VCFG0	-	-	-	-
	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0

bit 7 - ADFM: A/D Conversion Result Format Select bit

1 = Right justified

0 = Left justified

bit 6 - Unimplemented: Read as "0"

bit 5 - VCFG1: Voltage Reference bit

1 = VREF- pin

0 = V_{SS}

bit 4 - VCFG0: Voltage Reference bit

1 = VREF+ pin

0 = V_{DD}

bit 3 to 0 - Unimplemented: Read as '0'

5.2.3 ANSEL : Analog Select register

The ANSEL register is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSEL bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

Detail of each bit in ANSEL register is shown below.

bit	7	6	5	4	3	2	1	0
	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

bit 7 to 0 - ANS7 to ANS0 : Analog Select bits

Analog select between analog or digital function on pins AN<7:0> or RE2, RE1, RE0, RA5, RA3, RA2, RA1 and RA0 respectively.

1 = Analog input. Pin is assigned as analog input (default).

0 = Digital I/O. Pin is assigned to port or special function.

5.2.4 ANSELH : Analog Select High register

The ANSELH register is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELH bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly. The port pins which are controlled by this register consists of AN8 to AN13 (RB2, RB3, RB1, RB4, RB0 and RB5).

Detail of each bit in ANSELH register is shown below.

bit	7	6	5	4	3	2	1	0
	-	-	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

bit 7 and 6 - Unimplemented: Read as '0'

bit 5 to 0 - ANS13 to ANS8 : Analog Select bits

Analog select between analog or digital function on pins AN<13:8> or RB5, RB0, RB4, RB1, RB3 and RB2 respectively.

1 = Analog input. Pin is assigned as analog input.

0 = Digital I/O. Pin is assigned to port or special function.

5.3 ADC configuration

For using the ADC module of PIC16F887 microcontroller the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Results formatting

5.4.1 Port configuration

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits.

5.4.2 Channel selection

The CHS bits of the ADCON0 register determine which channel is connected to the sample and hold circuit. When changing channels, a delay is required before starting the next conversion.

5.4.3 ADC Voltage reference

The VCFG bits of the ADCON0 register provide independent control of the positive and negative voltage references. The positive voltage reference can be either Vdd or an external voltage source. Likewise, the negative voltage reference can be either Vss or an external voltage source.

For the RBX-877V2.0 Robot Controller board will select the positive reference to +5V and negative reference at ground or Vss.

5.4.4 Conversion Clock

The source of the conversion clock is software selectable via the ADCS bits of the ADCON0 register. There are four possible clock options:

- $F_{OSC}/2$: for 20MHz clock, T_{AD} = 100ns
- $F_{OSC}/8$: for 20MHz clock, T_{AD} = 400ns
- $F_{OSC}/32$: for 20MHz clock, T_{AD} = 1.6µs
- F_{RC} (dedicated internal oscillator) : T_{AD} = 2 to 6µs

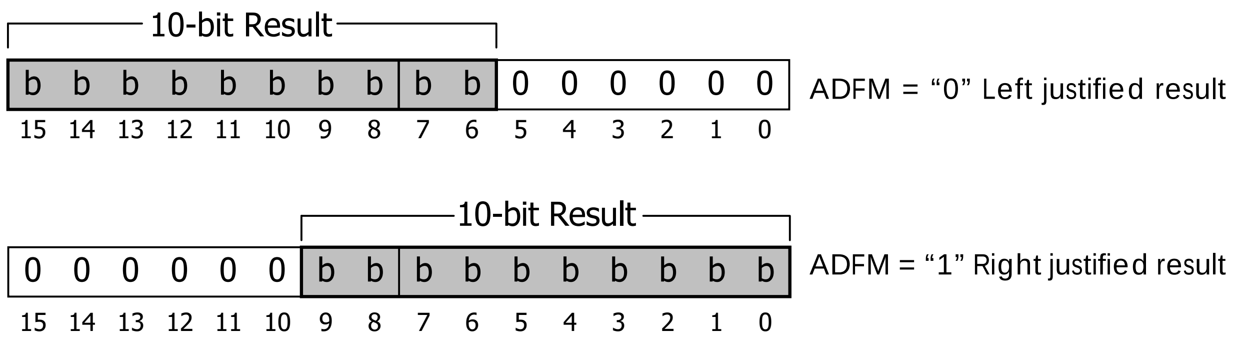
The time to complete one bit conversion is defined as T_{AD}. One full 10-bit conversion requires 11 T_{AD} periods.

5.4.5 Result formatting

The A/D converter result will store in a pair of regeter; ADRESH:ADRESL. They will keep the data format following the selection of ADFM bit.

If Left justified is selected (ADFM = '0'), ADRESH register keeps 8 upper bits and ADRESL register keeps 2 lower bits.

If Right justified is selected (ADFM = '1'), ADRESH register keeps 2 upper bits and ADRESL register keeps 8 lower bits.


10-bit Result
b b b b b b b b b b 0 0 0 0 0 0
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ADFM = "0" Left justified result
10-bit Result
0 0 0 0 0 0 b b b b b b b b b b
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ADFM = "1" Right justified result

The result data format from ADFM bit selection of ADCON1 register

5.5 A/D Conversion procedure

This is an example procedure for using the ADC to perform an Analog-to-Digital conversion pf PIC16F887 microcontroller:

1. Configure Port:
   - Disable pin output driver
   - Configure pin as analog by setting ANSEL or ANSELH register
2. Configure the ADC module (by setting ADCON0 register) :
   - Select ADC conversion clock
   - Configure voltage reference
   - Select ADC input channel
   - Select result format by setting ADCON1 register
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt
   - Enable global interrupt.

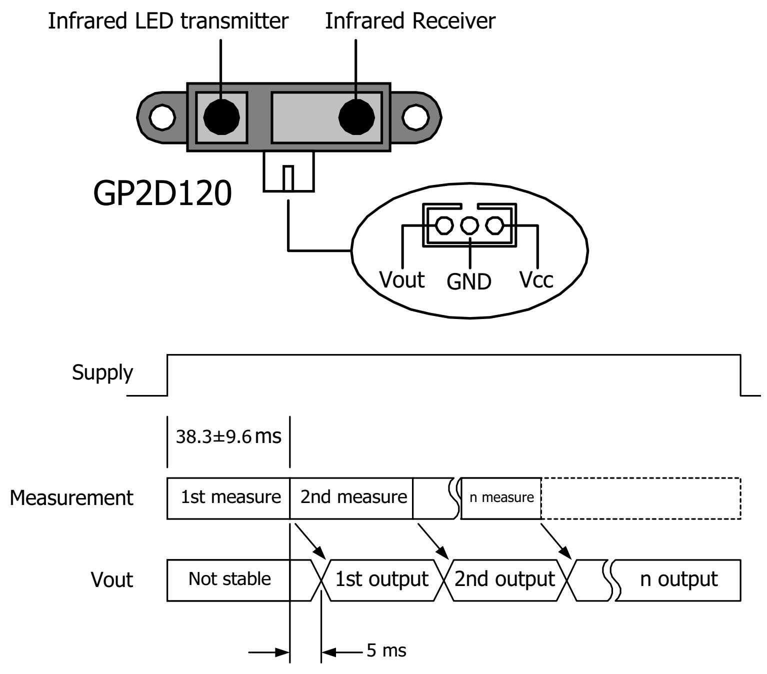
4. Wait the required acquisition time.
5. Start conversion by setting the GO/DONE bit in ADCON0 register.
6. Wait for ADC conversion to complete by one of the following:
    - Polling the GO/DONE bit
    - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result. The result data will store in ADRESH and ADRESL register.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

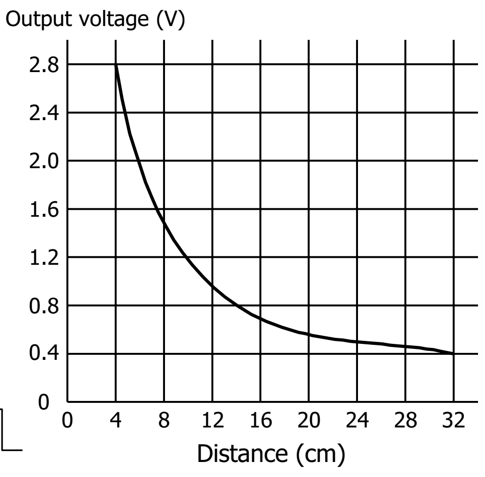
5.6 GP2D120 : 4 to 30cm. Infrared distance sensor

One of the special sensors in robotics is the Infrared Distance sensor. Some people call it the IR Ranger. With the GP2D120 module, it adds the distance measuring and Obstacle detection using infrared light feature to your robot. Your Robo-PICA robot can avoid obstacles without having to make any physical contact.

5.6.1 GP2D120 features

- Uses Infrared light reflection to measure range
- Can measure a range from 4 to 30 cm.
- 4. 5 to 5 V power supply and 33mA electric current
- The output voltage range is 0.4 to 2.4V when supplied by +5V





* Use Kodak R-27 gray-white paper. The white side has a 90% reflection rate, made from a material that reflects light for range measurement.

Figure 5-1 : GP2D120 pin assignment, operation and characteristic curve

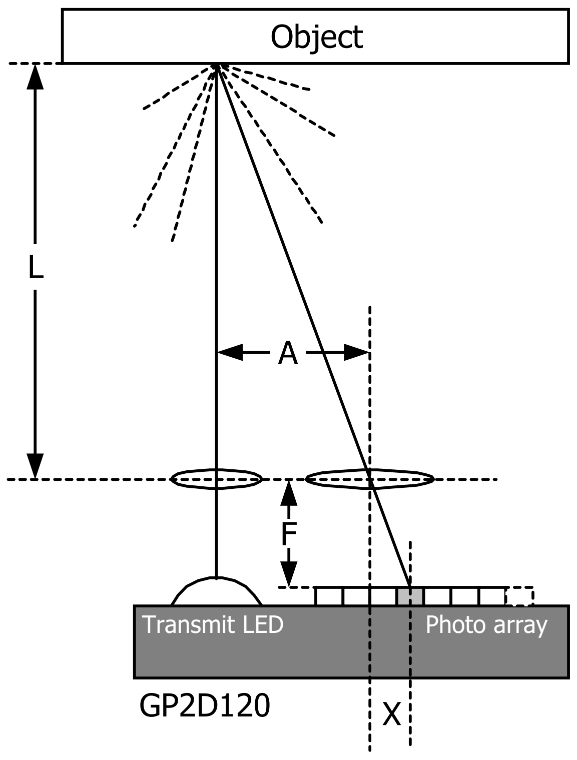
GP2D120 Infrared Ranger module has 3 terminals: Power input (Vcc), Ground (GND) and Voltage output (Vout). To read the voltage values from the GP2D120, you must wait till after the acknowledgement period which is around 32 to 52.9 ms.

The output voltage of GP2D120 at a range of 30 cm and +5V power supply is between 0.25 to 0.55V, with the mean being 0.4V. At the range of 4 cm., the output voltage will change at 2.25V± 0.3V.

5.6.2 How the IR Ranger Module works

Measuring range can be done in many ways. The easiest to understand is through ultra sonic where sound waves are sent to the object and the time it takes to reflect back is measured. This is because sounds waves do not travel fast, and can be measured by present day equipment. However, in the case of infrared light, the time it takes to hit an obstacle and reflect back can not be measured because infrared light travels fast. No measurement equipment is available yet. Therefore, the following theory must be used.

The infrared light is sent out from a transmitter to the object in front, by passing through a condense lens so that the light intensity is focused on a certain point. Refraction occurs once the light hits the surface of the object. Part of the refracted light will be sent back to the receiver end, in which another lens will combine these lights and. determine the point of impact. The light will then be passed on to an array of photo-transistors. The position in which the light falls can be used to calculate the distance (L) from the transmitter to the obstacle using the following formula:



$$\frac{L}{A} = \frac{F}{X}$$

Therefore, L equals

$$L = \frac{F \times A}{X}$$

Thus, the distance value from the phototransistors will be sent to the Signal Evaluation Module before it is changed to voltage, resulting in a change of voltage according to the measured distance.

5.6.3 Reading GP2D120 with A/D converter

The GP2D120's output voltage will change acoording to the detection distance. For example, Vout 0.5V is equal 26cm. distance and Vout 2V is equal 6cm. distance. The table 5-1 shows the summary of GP2D120's Vout and Distance relation.

For interfacing with A/D converter module within microcontroller, the result is raw data from the A/D conversion. The user will need to use the software to convert the raw data to the exact distance. You can calculate the approximate distance from the formular below.

$$R = \frac{2914}{V+5} - 1$$

Thus, R as Distance in Centimetre unit

V as Digital data from A/D conversion

For example, see the Table 5-1. The raw data from conversion is 307. It is equal 8cm. distance.

Warning for the signal cable of the GP2D120

The GP2D120 module has a different pin arrangement then that of the MicroCamp board, even though it looks similar. Therefore, a special signal cable has already been connected to the GP2D120 module. The user just needs to connect the other end of the cable to the connection points of the MicroCamp board. DO NOT remove the cable from the module, and do not replace it with signal cables from other sensor modules.

GP2D120 output voltage (V)	**10-bit A/D converter result**	**Distance (cm.)**
0.4	82	32
0.5	102	26
0.6	123	22
0.7	143	19
0.8	164	16
0.9	184	14
1.0	205	13
1.1	225	12
1.2	246	11
1.3	266	10
1.4	287	9
1.5	307	8
1.6	328	8
1.7	348	7
1.8	369	7
1.9	389	6
2.0	410	6
2.1	430	6
2.2	451	5
2.3	471	5
2.4	492	5
2.5	512	5
2.6	532	4

Table 5-1 : The relation of GP2D120 output voltage, A/D converter result and Measured distance.

Activity 5
Reading the analog signal

This activity introduces the simple experiment about reading the analog signal. The simple Variable resistor or Potentiometer is used to analog voltage source and plug into the analog input port of PIC16F887. You make the simple C code to read the result from ADC module inside PIC16F887 to display on LCD module.

A5.1 Write the Listing A5-1. Compile and download the code to RBX-877 V2.0 Robot Controller board.

A5.2 Plug the ZX-POTH; potentiometer sensor to RA3 port pin of RBX-877 V2.0 Robot Controller board.

A5.3 Run the program. Adjust the potentiometer and see the result at LCD screen on the RBX-877 V2.0 Robot Controller board.

LCD module shows message **SENSOR1 = xxx** *(***xxx** *as 0 to 1023).*

```
/*************************************/
/*****  Show ADC from RA1 to LCD ***/
/*************************************/
char data_[6];
int  x;
void main()
{
   Delay_ms(1000);
   Lcd_Init(&PORTD);
   ANSEL = 0xFF;                      // PORTA ==> Analog
   TRISA  = 0xFF;                     // PORTA ==> input
   Lcd_Cmd(LCD_CURSOR_OFF);           // LCD cursor off
   Lcd_Out(1,1,"SENSOR1 = ");         // Show Text
   ADCON0=0b11001101;                 // Select Analog1 RC_Mode and ADON
   while(1)
   {
       ADCON0.GO=1;
       while(ADCON0.GO);
       x= (ADRESH*4)+(ADRESL/64);
       WordToStr(x,data_);
       Lcd_Out(1,10,data_);
       Delay_ms(100);
   }
}
```

Listing A5-1 : The ADC demonstration program for reading the sensor value

ROBO PiCA
PIC MICRO.ROBOT KIT

Activity 6
Testing GP2D120

A6.1 Write the Listing A6-1. Compile and download the code to Robo-PICA.

A6.2 Plug the GP2D120 sensor to RA2 port pin of Robo-PICA. (ready from building activty 3).

A6.3 Run the program. Place some object at the front of GP2D120 module. Observe the LCD operation.

A6.4 Adjust the distance of the object from GP2D120 sensor and observe the result.

From testing, you will found the GP2D120 can detect the object in range 4 to 30cm.

```
int Adc;
char txt[6];

void Read_Adc()
{
  ADCON0=0b11001001;                          // Select Analog2 RC_Mode and ADON
  ADCON0.GO=1;                                // Start Convert
  while(ADCON0.GO);                           // Wait Until Convert Complete
  Adc=(ADRESH*4)+(ADRESL/64);                 // 10 bit Data ==> Adc
}

void main()
{
  Delay_ms(1000);
  Lcd_Init(&PORTD);                           // Initial LCD
  Lcd_Cmd(LCD_CURSOR_OFF);                    // LCD Cursor OFF
  Lcd_Out(1,1,"Raw Data= ");                  // Show Fisrt Line Text
  while(1)
  {
    Read_Adc();
    WordToStr(Adc,txt);                       // Convert To Show on LCD
    Lcd_Out(1,10,txt);
    if (Adc<90)                               // If Data < 90 It Out of Range
    {
      Lcd_Out(2,1,"Out of Range");
    }
    else
    {
      Adc = (2914/(Adc+5))-1;                 // Convert Data to Centimeter
      WordToStr(Adc,txt);                     // Convert Data to String
      Lcd_Out(2,1,"In CM=       ");           // Show on LCD
      Lcd_Out(2,6,txt);
    }
    Delay_ms(1000);
  }
}
```

Listing A6-1 : The GP2D120 demonstration program of Robo-PICA

ROBO PiCA PIC MICRO.ROBOT KIT

Activity 7

Contactless object detection robot

A7.1 Write the Listing A7-1. Compile and download the code to Robo-PICA.

A7.2 Turn-off power and unplug the download cable from Robo-PICA.

```
/*******************************************************/
/***** Robot with Object Detector ***********************/
/*******************************************************/
#include <motor.h>
int Adc;                                // Save analog Data
char Txt[6];                            // Save String

void Read_Adc()
{
 ADCON0=0b11011101;                     // Select Analog2 RC_Mode and ADON
 ADCON0.GO=1;                           // Start Convert
 while(ADCON0.GO);                      // Wait Until Convert Complete
 Adc=(ADRESH*4)+(ADRESL/64);            // 10 bit Data ==> Adc
 }

void main()
{
 Delay_ms(1000);                        // Start up Delay
 ANSELH.F4=0;                           // RBO ==> Digital IO
 ANSEL=0xFF;
 TRISA=0xFF;
 Lcd_Init(&PORTD);                      // Initial LCD
 Lcd_Cmd(LCD_CURSOR_OFF);               // LCD Cursor OFF
 while(PORTB.F0);                       // Wait Key Press
 while(1)
 {
  Read_Adc();                           // Read Analog 2
  WordToStr(Adc,Txt);                   // Convert Data to string
  Lcd_Out(1,1,Txt);                     // Show on LCD
  if (Adc>300)                          // if Detect object in range
  {
    Backward(255);Delay_ms(500);        // Backword and turn left
    S_left(255);Delay_ms(400);
  }
  else
  {
    Forward(255);                       // Object out of range FORWARD
  }
 }
}
/*******************************************************/
```

Listing A7-1 : The C program of the contactless object detection robot

A7.3 Place the robot on the floor. Turn-on the power and observe its operation.

A7.4 Try to place any object at the front of the robot and see its operation.

The robot will check the distance of the object in 8cm. range. If not any obstacle, robot will move forward continue. If found the object, it will move backward, turn left and move forward again.

Chapter 6

Line following mission

Line tracking or following is a popular activity in any Robotics activity. The purpose of this activity is to learn about how to interface analog sensor. In Robo-PICA robot kit, it comes with a pair of Infrared reflector sensor for this activity. Add senses to the Robo-PICA so that it can detect and move following the line, by using the IR Reflector Sensor. Two IR Reflector Sensors will be installed at the bottom of the Robo-PICA so that it can detect both white and black lines.

6.1 ZX-03 Infrared Reflector

The heart of this sensor is TCRT5000 reflective object sensor. It is designed for close proximity infrared (IR) detection. There's an infrared diode behind its transparent blue window and an infrared transistor behind its black window. When the infrared emitted by the diode reflects off a surface and returns to the black window, it strikes the infrared transistor's base, causing it to conduct current. The more infrared incident on the transistor's base, the more current it conducts.

When used as an analog sensor, the ZX-03 can detect shades of gray on paper and distances over a short range if the light in the room remains constant.

The suitable distance from sensor to line or floor is during 3 to 8 mm. The output voltage is during 0.1 to 4.8V and digital value from10-bit A/D converter is 20 to 1,000. Thus, ZX-03 will suitable to apply to line tracking sensor.

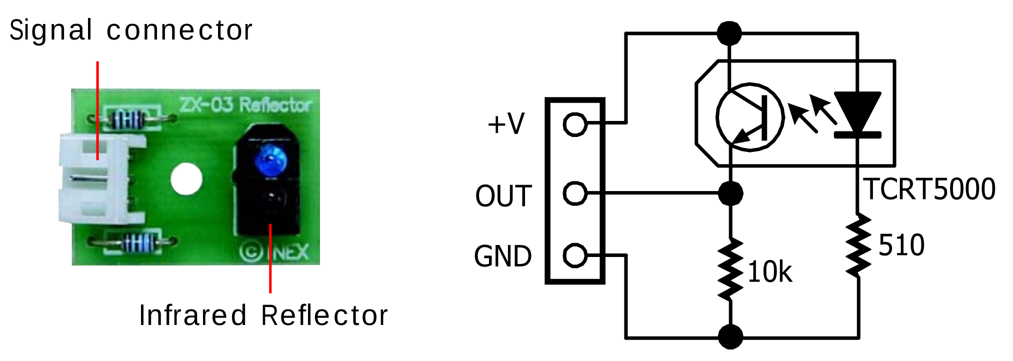


Figure 6-1 : ZX-03 Infrared Reflector information

Activity 8
Reading the Line tracking sensor

In building Robo-PICA from activity 2, both of ZX-03 Infrared reflectors are installed at the bottom of the robot body and connected to the sensor's cable to RA0 and RA1 port pin. This activity is to write the program to read sensor's value to display at the LCD module of Robo-PICA.

A8.1 Write the Listing A8-1. Compile and download the code to Robo-PICA.

A8.2 Turn-off power and unplug the download cable from Robo-PICA.

A8.3 Place Robo-PICA on the white surface of the paper demonstration field which is bundled with Robo-PICA kit. The position of both ZX-03 sensors must above the white surface. Turn-on POWER switch. See the result at LCD screen and record it.

A8.4 Place Robo-PICA on the black line of the paper demonstration field . The position of both ZX-03 sensors must above the black line. Turn-on POWER switch. See the result at LCD screen and record it.

From testing can conclude the result as :

The white surface reading is during 400 to 900

The black surface reading is during 0 to 150

Thus, the reference value for making decision is during 150 to 400. You can select the suitable value and make decision as :

If the sensor reading value more than the reference value, sensor detect ***"WHITE surface"***

If the sensor reading value less than the reference value, sensor detect ***"BLACK surface or line"***

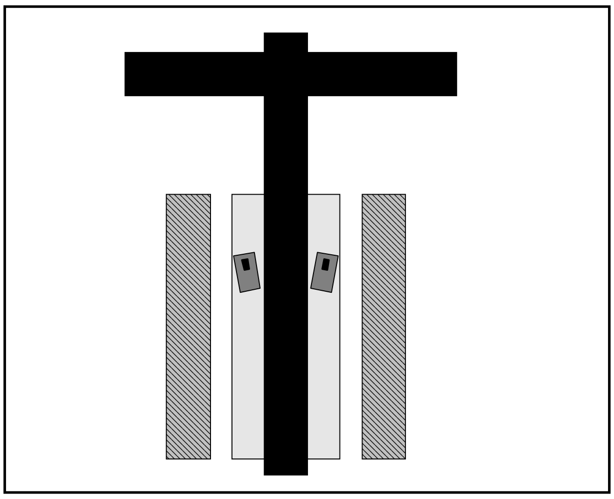
```
/*******************************************************/
/************ Reading line tracking sensor *************/
/*******************************************************/
int Sensor0,Sensor1;                   // Save Analog
char Txt[6];                           // Save convert to string

void Read_Adc()
 {
    ADCON0=0b11000001;                     // Select Analog0 RC_Mode and ADON
    ADCON0.GO=1;                           // Start Convert
    while(ADCON0.GO);                      // Wait Until Convert Complete
    Sensor0=(ADRESH*4)+(ADRESL/64);        // 10 bit Data ==> sensor0
    ADCON0=0b11000101;                     // Select Analog1 RC_Mode and ADON
    ADCON0.GO=1;                           // Start Convert
    while(ADCON0.GO);                      // Wait Until Convert Complete
    Sensor1=(ADRESH*4)+(ADRESL/64);        // 10 bit Data ==> sensor1
 }

void main()
{
  Delay_ms(1000);                        // Start up Delay
  Lcd_Init(&PORTD);                      // Initial LCD
  ANSEL = 0xFF;                          // PORTA ==> Analog
  TRISA  = 0xFF;                         // PORTA ==> input
  Lcd_Cmd(LCD_CURSOR_OFF);               // LCD cursor off

  while(1)
  {
    Read_Adc();
    WordToStr(Sensor0,Txt);            // Convert Sensor0 to string
    Lcd_Out(1,1,Txt);                  // and show on LCD

    WordToStr(Sensor1,Txt);            // Convert Sensor1 to string
    Lcd_Out(2,1,Txt);                  // and show on LCD
  }
}
```

Listing A8-1 : The C program for reading ZX-03 line tracking sensor of Robo-PICA

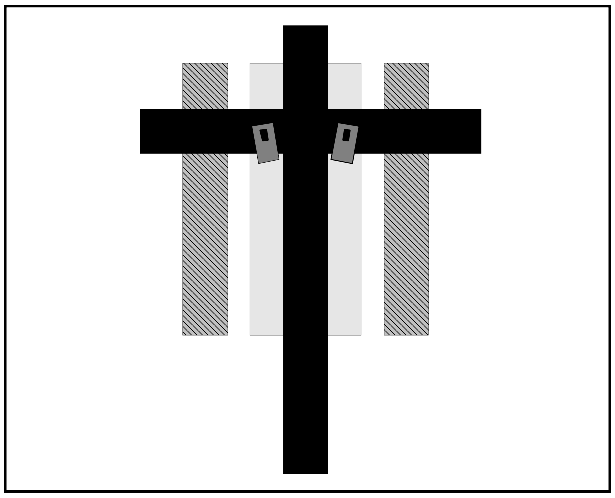
If different value between white and black surface reflection is less. Builder must adjust the distance from sensor to surface decreasing. If the sensor far from surface more, value will near zero.

Activity 9
Moves follow the black line

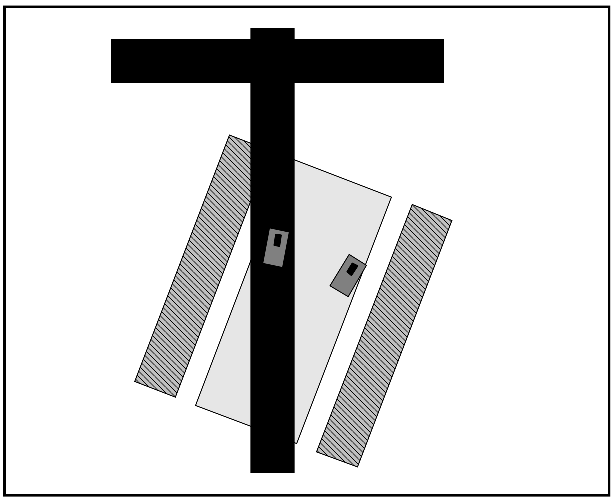
From installation ZX-03 Infrared reflector in Robo-PICA can set the line following behavior to 4 scenarios. See the figure below.



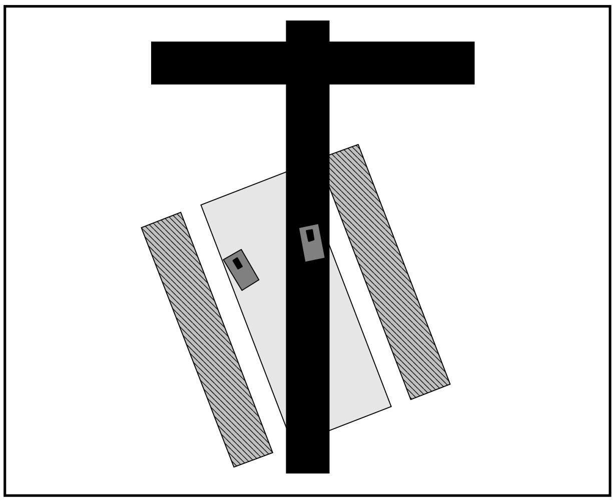
1. Both sensor above the white surface
This shows that the robot is moving along the black line.



2. Both sensor above the black surface
This shows that the 2 sensors are on the black line surface. The robot can choose to do a few options, to turn right, left, stop, reverse, etc...



3. The left sensor is above the black floor.
This means that the robot must turn left back into place.



4. The right sensor is above the black floor. This means that the robot must turn right back into place.

From the black line following behavior, you can apply to make the program to control the Robo-PICA. See the Listing A9-1. The sequence of program operations are

1. Set the default value to interface A/D Converter module in PIC16F887 microcontroller.

2. Enable both DC motor driver circuits.

3. Get data from both sensors that connected at RA0 and RA1 port pins. Load data to SENSOR0 (store data from RA0) and SENSOR1 (store data from RA1) variable

4. Compare the data read with the reference data. *If SENSOR0 and SENSOR1 data are more than the reference, control the robot to move forward.*

5. *If SENSOR0 and SENSOR1 data are less than the reference, control the robot to move forward.* This condition is the robot meets an intersection and decide to move forward continue.

6. *If SENSOR0 data is less than the reference only, the robot turns left.*

7. *If SENSOR1 data is less than the reference only, the robot turns right.*

A9.1 Write the Listing A9-1. Compile and download the code to Robo-PICA.

A9.2 Turn-off power and unplug the download cable from Robo-PICA.

A9.3 Place Robo-PICA cross the black line in the demonstration paper filed (bundled in Robo-PICA robot kit).

A9.4 Turn-on POWER switch. Observe the robot operation.

The Robo-PICA will move following the black line and not detect the across line or intersection line.

How to find the reference value ?

If the sensor reading value less than the reference value, can interpret the deteced surface as **“Black”**

If the sensor reading value more than the reference value, can interpret the detected surface as **“White”**

The reference value can calulate from :

(The sensor’s minimum reading value from white surface + The sensor’s maximum reading value from black surface) / 2

<u>Example</u>

If The sensor’s minimum reading value from white surface is equal 300 and T h e sensor’s maximum reading value from black surface is equal 100

The reference value will be equal (300+100) / 2 = **200**

```
/*****************************************************/
/***** Track Black line Forward if cross line ***********/
/*****************************************************/
#include <motor.h>
int Sensor0,Sensor1;                    // Save Analog
char Txt[6];                            // Save convert to string

void Read_Adc()
 {
    ADCON0=0b11000001;                      // Select Analog0 RC_Mode and ADON
    ADCON0.GO=1;                            // Start Convert
    while(ADCON0.GO);                       // Wait Until Convert Complete
    Sensor0=(ADRESH*4)+(ADRESL/64);        // 10 bit Data ==> sensor0
    ADCON0=0b11000101;                      // Select Analog1 RC_Mode and ADON
    ADCON0.GO=1;                            // Start Convert
    while(ADCON0.GO);                       // Wait Until Convert Complete
    Sensor1=(ADRESH*4)+(ADRESL/64);        // 10 bit Data ==> sensor1
 }

void main()
{
  Delay_ms(1000);                         // Start up Delay
  Lcd_Init(&PORTD);                       // Initial LCD
  ANSEL = 0xFF;                           // PORTA ==> Analog
  TRISA  = 0xFF;                          // PORTA ==> input
  Lcd_Cmd(LCD_CURSOR_OFF);                // LCD cursor off

  while(1)
  {
    Read_Adc();
    WordToStr(Sensor0,Txt);             // Convert Sensor0 to string
    Lcd_Out(1,1,Txt);                   // and show on LCD

    WordToStr(Sensor1,Txt);             // Convert Sensor1 to string
    Lcd_Out(2,1,Txt);                   // and show on LCD

    if ((Sensor0>500)&(Sensor1>500))    // Check All Sensor are white
       Forward(255);                    // Forward
    if ((Sensor0<500)&(Sensor1<500))    // All Sensor are black (on cross)
    {
      Forward(255);                     // Forward 0.3 Second
      Delay_ms(300);
    }
    if (Sensor0<500)                    // Sensor0 Only  detect line
    {
      S_Left(255);                      // spin Left
    }
    if (Sensor1<500)                    // Sensor1 Only detect line
    {
      S_Right(255);                     // Spin Right
    }
  }
}
```

Listing A9-1 : The C program of Line tracking robot

Summary

The heart of line following robot operation are 3 factors. First, the sensor's performance. Second, sensor's installation and Third, the control software. Sometimes the robot need more sensors to detect the complex line such as 3 cross line, intersection line, etc. See the Figure A9-1.

About the installation Infrared reflector to detect the line, the distance between both sensors is important. The suitable space improves the detection. If the sensor is near the line more, the robot will detect the line many times. It causes the robot movement to swing from left to right similar to a snake movement. See the Figure A9-2. It means the speed of movement reduce.

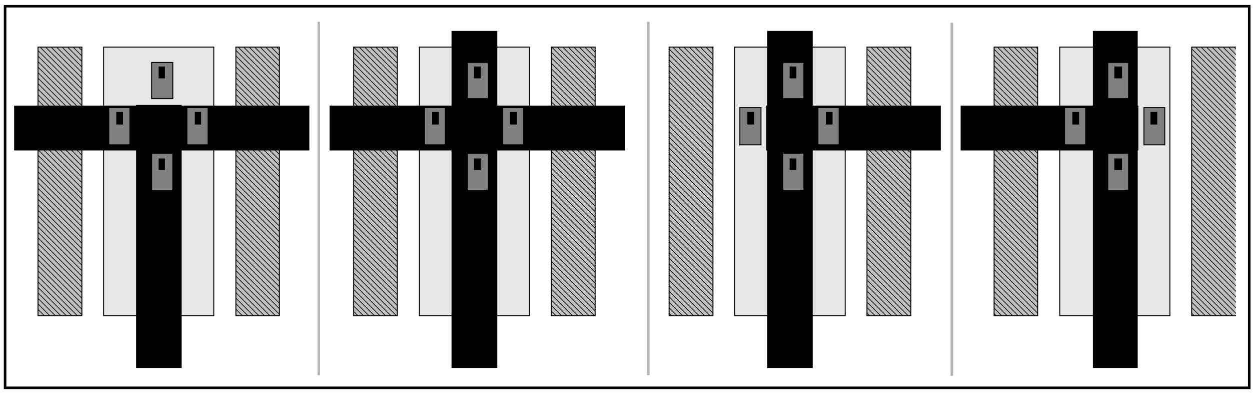
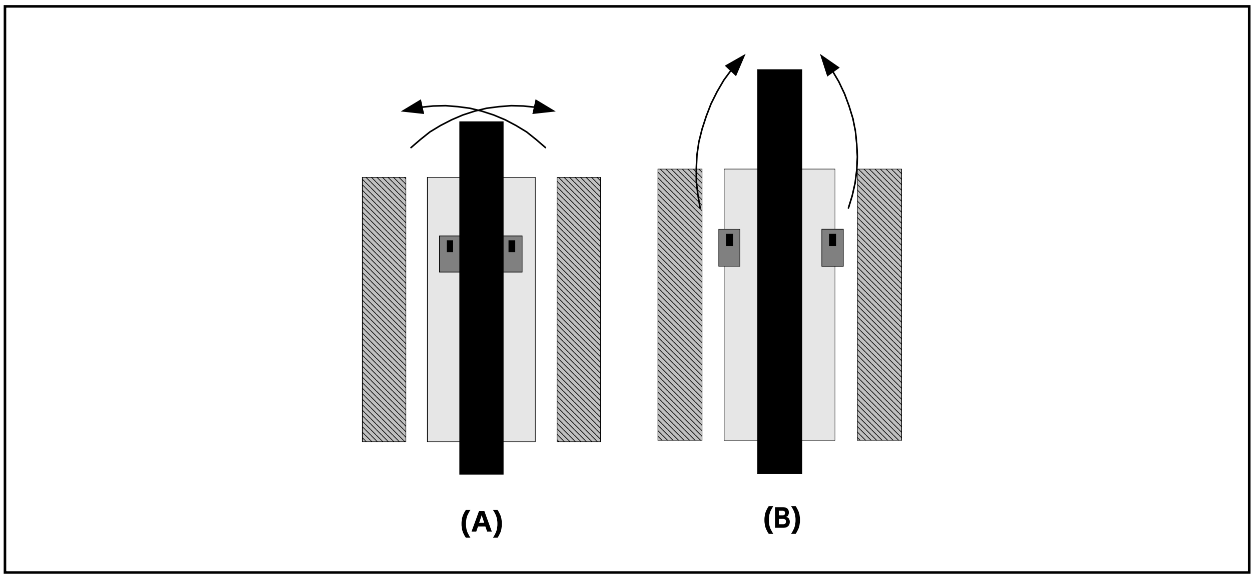


Figure A9-1 : Add more line tracking sensors to detect the complex line.



Figue A9-2 : The result of installation line detector.
(A) Install the sensor near over. It cause the robot moves swing.
(B) Installation both sensors far from them and line. If good enough, they help the robot to move following the line better.

Chapter 7

Robot with Remote control

Another feature of automatic robots is that it can receive commands from a far distance by using infrared light. This is similar to a remote-controlled robot except that the commands received are through serial communication. Robo-PICA provides an Infrared Remote control, called the ER-4. The ER-4 remote control will modulate serial data with infrared light. Robo-PICA must first be installed with a 38kHz Infrared receiver module for receiving.

7.1 ZX-IRM : Infrared Receiver module

The data transferedcwith Infrared Light can reach up to distnce of 5 to 10 meters which is similar to TV remote control. The carrier frequency is 38kHz.The receiver must demodulate 38kHz carrier frequency. After this, it is then transferred as serial data to microcontroller.

If the sensor does not detect the 38kHz frequency with the infrared light, the output will be logic “1”. Otherwise, if it detects the 38kHz frequency, the output logic is “0”.

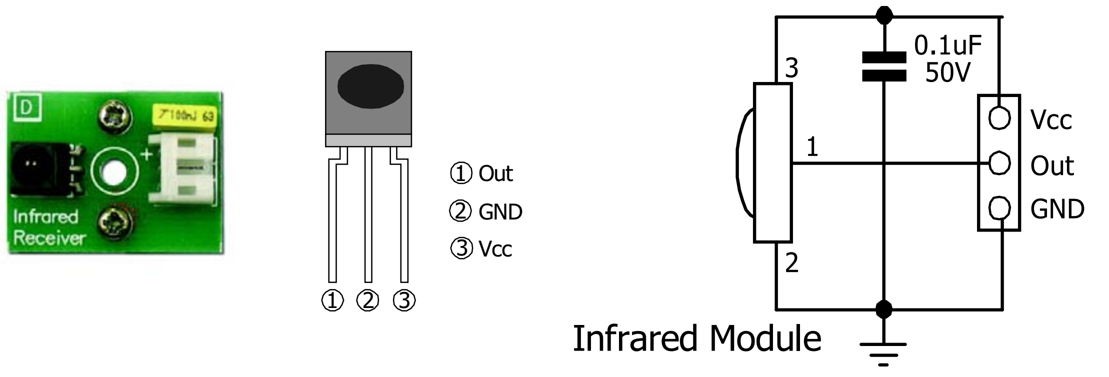


Figure 7-1 : Shows the photo of 38kHz Infrared Receiver module, pin assignment and schematic diagram

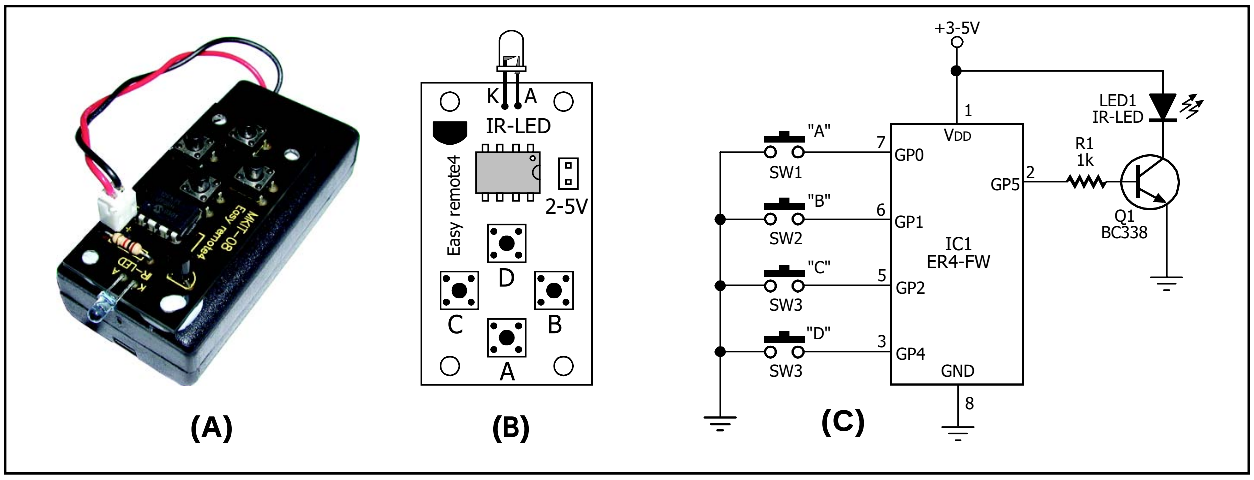


Figure 7-2 : Shows the photo, board layout and Schematic of ER-4 Easy remote control

7.2 ER-4 Infrared Remote control

- Operational distance is 4 to 8 meters in any open space.
- The 4-channel switch operates in an on/off mode
- Uses low power; Automatically resumes power-save mode once data is sent
- Uses only 2.4 to 3V from two AA batteries - both regular and rechargeable.
- Transmits serial data using the RS-232 standard with 1200 bps baud rate and 8N1 data format (8 data bit, no parity, 1 stop bit)

7.2.1Format of data sent by ER-4 Remote control

To make it easier for the receiver to read the switch value from the remote control, the ER-4 transmit serial data according to the RS-232 standard, with a baud rate of 1,200 bps and 8N1 format. Characters are transmitted according to what switch is pressed on the remote. The switch positions are displayed in Figure 7-2

Press switch A, the large cap A , followed by small cap A (a) is sent.

Press switch B, the large cap B, followed by small cap B (b) is sent.

Press switch C, the large cap C, followed by small cap C (c) is sent

Press switch D, the large cap D, followed by small cap D (d) is sent.

The reason that we have to alternate large cap and small cap letters is so that the receiver can differentiate if a user presses continuously or if the user represses. If a user represses, the large cap character will be sent the first time. If the user represses the same button again, the small cap character will be sent the second time If the user presses continually, the last character will be sent repeatedly.

Activity 10

Reading Remote control data

In the Listing A10-1 is C program for reading data from ER-4 remote control and shows on LCD screen. In addition use this data to compare the reference data for driving sound to piezo speaker by **Sound_Play** function of mikroC compiler.

A10.1 Write the Listing A10-1. Compile and download the code to Robo-PICA.

```
char *text = "ER-4 Remote";          // Define message
unsigned char ir_cmd=0;              // Keep Command button from ER-4 Remote

//------------------ Interrupt service routine INT -----------------//
void interrupt()
{
 unsigned char i;                    // Keep counter
 if(INTCON.INTF)                     // Check interrupt flag RB0  (Falling edge)
 {
  Delay_us(416);                     // Delay 1/2 of 1 bit timing
                                     // (baudrate 1200 bps)
  for(i=0;i<8;i++)                   // Loop 8 times for keep data from ER-4
  {
    Delay_us(833);                   // Delay of 1 bit timing
                                     // (baudrate 1200 bps)
    ir_cmd = ir_cmd>>1;              // Shitt bit 1 time
    if((PORTB & 0x01)==1)            // Get logic @ RB0 = '1'?
       ir_cmd = ir_cmd | 0x80;       // Inset bit data is '1'
  }
  Delay_us(833);                     // Delay of 1 bit timing
                                     // (baudrate 1200 bps)
  INTCON.INTF =0;                    // Clear interrupt flag
 }
}

//------------------ Function for get character from Remote ---------//
unsigned char get_remote()
{
 unsigned char _key=ir_cmd;          // Get character to buffer
 ir_cmd=0;                           // Clear old data
 return(_key);                       // Return character from Remote
}

//------------------ Main Program ---------------------------------//
void main()
{
 unsigned char key;                  // Save Remote Key Press
 ANSELH.F4=0;                        // RB0 ==> Digital IO
 OPTION_REG.INTEDG = 0;              // INT falling edge
 INTCON.INTE =1;                     // Enable INT/PB0
 INTCON.GIE =1;                      // Enable Global interrupt
```

Listing A10-1 : Reading data from ER-4 remote control program (continue)

```
  Lcd_Init(&PORTD);                         // Initialize LCD connected to PORTD
  Lcd_Cmd(Lcd_CLEAR);                       // Clear display
  Lcd_Cmd(Lcd_CURSOR_OFF);                  // Turn cursor off
  Lcd_Out(1, 1, text);                      // Print text to LCD,2nd row,1st column
  Sound_Init(&PORTC,0);
  while(1)                                  // Infinite loop
  {
   key = get_remote();                      // Get Remote
   if(key=='a' || key=='A')                 // Button A press?
   {
     Lcd_Out(2, 1, "Button A Press  ");     // Display message Button A press
     Sound_Play(100,500);
   }
   else if(key=='b' || key=='B')            // Button B press?
   {
     Lcd_Out(2, 1, "Button B Press  ");     // Display message Button B press
     Sound_Play(110,500);
   }
   else if(key=='c' || key=='C')            // Button C press?
   {
     Lcd_Out(2, 1, "Button C Press  ");     // Display message Button C press
     Sound_Play(120,500);
   }
   else if(key=='d' || key=='D')            // Button D press?
   {
     Lcd_Out(2, 1, "Button D Press  ");     // Display message Button D press
     Sound_Play(130,500);
   }
  }
}
```

Listing A10-1 : Reading data from ER-4 remote control program (final)

A10.2 Put the 2 of AA batteries in the Batter holder of ER-4 Remote control.

A10.3 Turn-on power. Press the button switch on ER-4 Remote control to send data to ZX-IRM at the Robo-PICA robot. Observe the operation at LCD module and piezo spekaer.

LCD will show letter A, B, C, D or a,b,c,d following press switch on ER-4 Remote control and listen the diffrent beep sound frequency.

Activity 11

IR control Robo-PICA’s movement

This activity shows how to control Robo-PICA’s movement via ER-4 Infrared remote control. The switch or button on ER-4 remote control will function move forward-backward-left-right.

A11.1 Write the Listing A11-1. Compile and download the code to Robo-PICA.

A11.2 Turn-off power and unplug the download cable from Robo-PICA.

A11.3 Place Robo-PICA on the floor.

A11.4 Turn-on POWER switch. Use ER-4 remote control and observe the robot operation.

Robo-PICA will not move until the ER-4 button is being pressed. Pressing button on ER-4 Remote control. The direction of sending light must straigth. The communication will be complete.

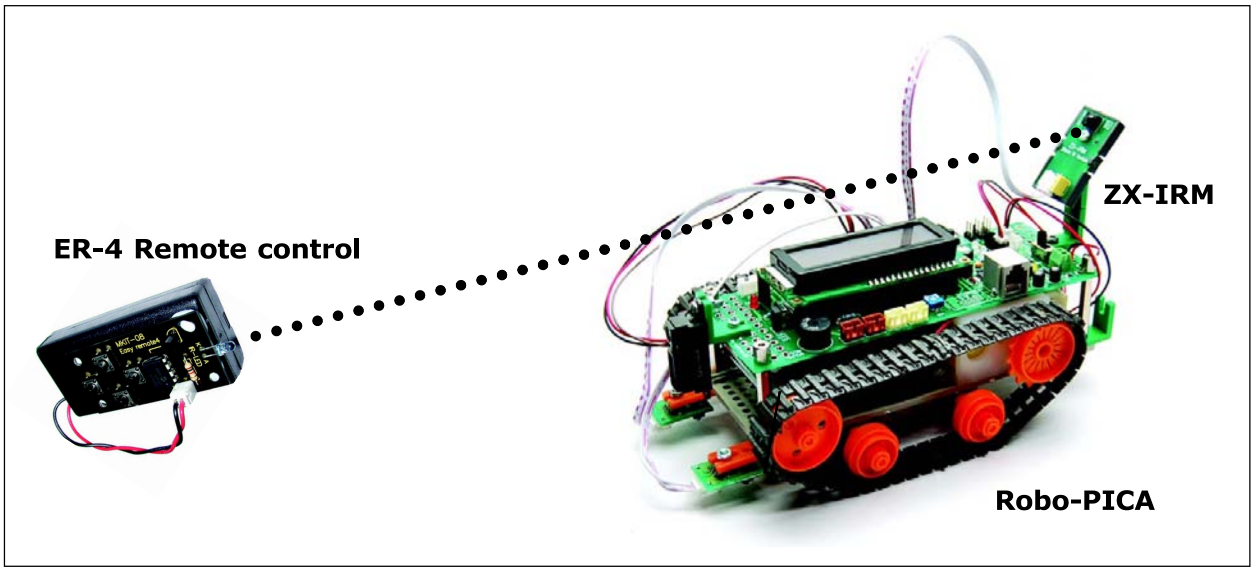


Figure A11-1 : Shows the Robo-PICA controlling with ER-4 Infrared remote control.

```
#include <motor.h>
char *text = "ER-4 Remote";  // Define message
unsigned char ir_cmd=0;      // Keep Command button from ER-4 Remote

//------------------ Interrupt service routine INT -----------------//
void interrupt()
{
 unsigned char i;            // Keep counter
 if(INTCON.INTF)             // Check interrupt flag RB0  (Falling edge)
 {
  Delay_us(416);             // Delay 1/2 of 1 bit timing(baudrate 1200 bps)
  for(i=0;i<8;i++)           // Loop for 8 time for keep data from ER-4 Remote
  {
    Delay_us(833);           // Delay of 1 bit timing(baudrate 1200 bps)
    ir_cmd = ir_cmd>>1;      // Shitt bit 1 time
    if((PORTB & 0x01)==1)    // Get logic @ RB0 = '1'?
       ir_cmd = ir_cmd | 0x80; // Inset bit data is '1'
  }
  Delay_us(833);             // Delay of 1 bit timing(baudrate 1200 bps)
  INTCON.INTF =0;            // Clear interrupt flag
 }
}

//------------------ Function for get character from Remote ---------//
unsigned char get_remote()
{
 unsigned char _key=ir_cmd;  // Get character to buffer
 ir_cmd=0;                   // Clear old data
 return(_key);               // Return character from Remote
}

//------------------ Main Program ----------------------------------//
void main()
{
 unsigned char key;          // Save Remote Key Press
 ANSELH.F4=0;                // RB0 ==> Digital IO
 OPTION_REG.INTEDG = 0;      // INT falling edge
 INTCON.INTE =1;             // Enable INT/PB0
 INTCON.GIE =1;              // Enable Global interrupt

 Lcd_Init(&PORTD);           // Initialize LCD connected to PORTD
 Lcd_Cmd(Lcd_CLEAR);         // Clear display
 Lcd_Cmd(Lcd_CURSOR_OFF);    // Turn cursor off
 Lcd_Out(1, 1, text);        // Print text to LCD, 2nd row, 1st column
 Sound_Init(&PORTC,0);
 while(1)                    // Infinite loop
 {
  if (ir_cmd==0)
  {
    Motor_Stop();
  }
  else
  {
    key = get_remote();        // Get Remote
    if(key=='a' || key=='A')   // Button A press?
```

Listing A11-1 : The C program for Robo-PICA to controlled with ER-4 remote control (continue)

```
      {
         Lcd_Out(2, 1, "Button A Press  ");  // Display message Button A press
         Backward(255);Delay_ms(50);
      }
      else if(key=='b' || key=='B')           // Button B press?
      {
         Lcd_Out(2, 1, "Button B Press  ");  // Display message Button B press
         S_Right(255);Delay_ms(50);
      }
      else if(key=='c' || key=='C')           // Button C press?
      {
         Lcd_Out(2, 1, "Button C Press  ");  // Display message Button C press
         S_Left(255);Delay_ms(50);
      }
      else if(key=='d' || key=='D')           // Button D press?
      {
         Lcd_Out(2, 1, "Button D Press  ");  // Display message Button D press
         Forward(255);Delay_ms(50);
      }
    }
  }
}
```

Listing A11-1 : The C program for Robo-PICA to controlled with ER-4 remote control (final)

Apppendix A

Activating the License Key of mikroC compiler

About Free Version

The latest version of compiler is always available for download from our website. It is a fully functional software with all the libraries, examples, and comprehensive help included. You can also download a separate manual in PDF format, provided for printing.

The only limitation of the free downloaded version is that you cannot generate hex output over 2k of program words. Although it may sound restrictive, this margin allows you to develop practical, working applications without ever thinking of the demo limit. If you intend to develop *really* complex projects in compilers, then you should consider purchasing the license key.

Note: This license key is valid until you reformat your hard disk. In case you have to format your hard disk, you should request new activation key. Still, you can reinstall your Windows operating system without formatting the hard drive - the license key will still be valid.

Support

If you have any questions about the activation of the compiler, visit our support forums at www.mikroe.com/forum/ If you are experiencing problems with any of our products or you just need additional information, please let us know.

Contact mikroE:

mikroElektronika

Voice: + 381 (11) 30 66 377, + 381 (11) 30 66 378

Fax: + 381 (11) 30 66 379

Web: www.mikroe.com

E-mail: office@mikroe.com

Special thanks for **Mikroelektronika** (www.mikroe.com) about the demo version of mikroC software and **Microchip Technology**; the PIC16F887 micrcontroller manufacture about supporting the PICkit2™ Programming software.