EZIO-100

User's Guide

Revision: 200

M030416

Table of Contents

Chapter 1 Introduction

1.1 About EZIO-100

Proprietary keypad and LCD display interfaces are implemented in traditional computing system design, but they are usually different from system to system. The main purpose to roll this module out is to provide an easier man-machine interface for those computing systems regarding application friendly operation as a "must."

The design goals of this interface are:

- A single interface for those applications where both LCD display and keypad are required.
- This interface should be available in every computing system.
- The communication implementation should be OS independent.

Our solution is to use "Serial port" as the interface for both LCD display and keypad. A simple protocol is further defined so that applications can directly communicate with this module no matter what the Operating System is.

WARNING!

THE LCD DRIVER ICS ARE MADE OF CMOS PROCESS, DAMAGED BY STATIC CHARGE VERY EASILY. MAKE SURE THE USER IS GROUNDED WHEN HANDLING THE LCD.

1.2 Features

- Ideal user interface for communication appliance
- No driver required; OS independent
- Alphanumeric characters display support
- Four key pads can be customized for different applications
- Easy system installation and operation
- Clearly display system status
- Single interface to SBC or M/B

1.3 Technical Support Information

For further support, users may also contact Portwell's or your local distributors.

Chapter 2 Specification

2.1 Mechanical Specification

Module Size (mm):	• 101.6(W) x 26.0(H) x 30.6(D) (max.)
Display Format:	• 16 characters x 2 lines
Character Size:	• 3.0 x 5.23 mm

2.2 General Specification

General Specification

Display Resolution:	• 16 characters x 2 lines
Dimensional Outline (mm):	• 101.6(W) x 26.0(H) x 30.6(D) (max.)
Function Key:	• Four operation keys (up, down, enter and ESC)
Display Icon:	• Eight self-defined icons
Interface:	• RS-232

Absolute Maximum Rating

Item	*Normal Temperature*			
	Operating		Storage	
	Max.	Min.	Max.	Min.
Ambient Temperature	0ºC	+50ºC	-20ºC	+70ºC
Humidity (w/o condensation)	Note 2, 4		Note 3, 5	

2.3 Product Outlook

2.4 Interface Pin Assignment

There are only two connectors in this module, as shown in ***Figure 2-1***: power connector and Serial Port connector. The power source into this module is 5 volt only. There are only three pins used in the Serial Port interface (***Figure 2-2***).

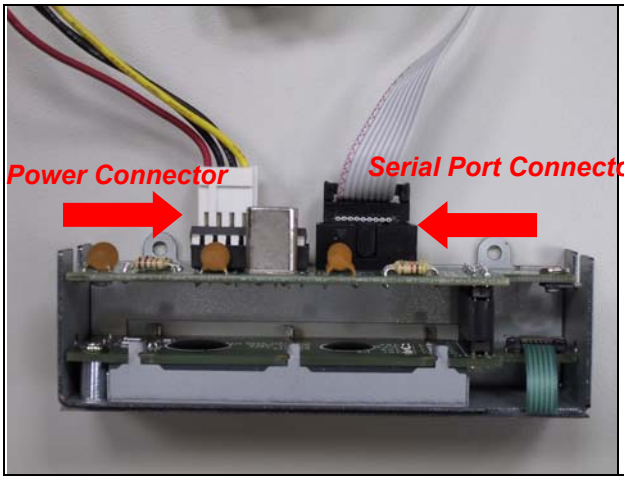
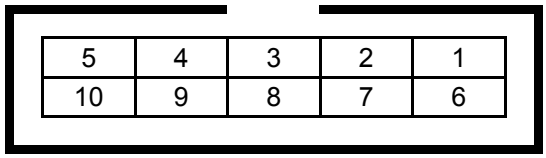


Fig. 2-1 *Power connector and serial port connector of EZIO-100*



Pin 2: TxD Pin 3 : RxD Pin 5 : Ground

Fig. 2-2 *Pin assignment*

In other words, the Serial Port is defined as DCE. Therefore, we can use a straight-through cable to connect it to the Serial Port of most of the computers, defined as DTE.

(1) Interface Pin Assignment

PIN NO.	PIN OUT	Description
1	NC	No connector
2	RXD	RS232 Data
3	TXD	RS232 Data
4	NC	No connector
5	V_{SS}	Ground
6	NC	No connector
7	NC	No connector
8	NC	No connector
9	NC	No connector
9	NC	No connector

(2) Power

PIN NO.	PIN OUT	Description
1	NC	No connector
2	GND	Power GND
3	GND	Power GND
4	+5V	Power VCC (+5V)

Chapter 3 Get Started

3.1 Hardware installation

The installation steps are:

- Connect the power connector to the power connector of this module.
- Connect the straight-through cable between Serial Port of this module and computer

3.2 EZIO Function Command

First, all versions (00A, 01A, 02A) of EZIO can use those commands. Only the 02A version of EZIO firmware that adds “FE 28” & “FE 37” command can control start of HEX & End of HEX.

EZIO is an intelligent device, which will display those data received from RS-232 port and reply key pressing status to polling command from RS-232 port. Both commands and data go thru RS-232 ports. To distinguish between data and commands, the LCD/key-pad module recognizes a command prefix, 254 (Hex 0FE). The byte following “254” will be processed as a command. For example, to clear the screen, send the command prefix (254) followed by the LCD clear-screen code (1). The valid data range is shown as the following table:

Valid data range	*Displayed characters*
0-7	Customized icon 0-7
48-57 (30-39 Hex)	0-9
65-90 (41-5A Hex)	A-Z
97-122 (61-7A Hex)	a-z

To get the key pressing status, a “read key” command can be issued to this module, which will check the key-pressing status and reply accordingly. The following are the commands and corresponding Decimal/Hex values:

	Functions/commands	*Decimal/Hex*	*Comment*
1.	Start Of HEX	40/28	Only for 02A
2.	End Of HEX	55/37	Only for 02A
3.	Clear screen	1/01	
4.	Home cursor	2/02	
5.	Read key	6/06	See note 1
6.	Blank display (retaining data)	8/08	
7.	Hide cursor & display blanked characters	12/0C	
8.	Turn on (blinking block cursor)	13/0D	
9.	Show underline cursor	14/0E	
10.	Move cursor 1 character left	16/10	
11.	Move cursor 1 character right	20/14	
12.	Scroll 1 character left	24/18	
13.	Scroll 1 character right	28/1C	
14.	Set display address (position the cursor) location	128 (Hex080)+ Location	See note 2
15.	Set character-generator address	64 (Hex 040)+ address	See note 3

Note 1: Upon receiving the “read key” command from host computer, the LCD/keypad module will check the status of every key and reply with status command accordingly. The replied message from LCD/key-pad module consists of a header and a status byte. The header byte is 253 (Hex0FD). The high nibble (with the most significant bit) of the status byte is always “4” and the low nibble (with the least significant bit) of the status byte is used to indicate key pressing status of the keypad module. This nibble will be “F” (of four 1s), if no key pressed while the “read key” received. “0” will be used to indicate key pressing status of corresponding key. There are four keys in this module – upper arrow, down arrow, enter (ENT), and escape (ESC). The relationship between the function key, corresponding status bit and status byte is shown as the table below.

Function key	*Corresponding status bit*	*Status byte*
Escape	The fourth bit of lower nibble (the least significant bit) (1110)	4E (H)
Up arrow	The third bit of lower nibble (1101)	4D (H)
Enter	The second bit of lower nibble (1011)	4B (H)
Down arrow	The first bit of lower nibble (0111)	47 (H)

More than one key can be pressed at the same time so that there may be more than one “0”s in the low nibble of status byte. For example, if Up and Down arrow keys are pressed at the same time while “read key” command received, the replied status will be “Hex045”.

Note 2: This command can be used to place the cursor at any location. The corresponding address for each character on the screen is as follows:

For 16×2 Display Address

Character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Location (Address)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

The addresses of characters at the same row are continuous, so moving cursor commands can be applied to shift the cursor position back and forth. However, the addresses of characters between upper and lower row are discontinuous. To change cursor position between upper row and lower row, this command will be applied.

Note 3: This command can be used to create customized icon. The starting address is 64 and every character will take 8 bytes to create a 5(W) x 7(H) resolution picture, as shown below:

CG RAM MAPPING

CG RAM Address							Character Patterns (CG RAM data)								
5	4	3	2	1	0		7	6	5	4	3	2	1	0	
High					Low		High							Low	
			0	0	0					0	1	1	0	0	
			0	0	1					1	0	0	1	0	
			0	1	0					0	0	1	0	0	←Character
			0	1	1					0	1	0	0	0	Pattern
0	0	0	1	0	0		*	*	*	1	1	1	1	0	
			1	0	1					0	0	0	0	0	
			1	1	0					0	0	0	0	0	
			1	1	1					0	0	0	0	0	←Cursor
			0	0	0					1	1	1	1	1	
			0	0	1					1	0	0	0	1	
			0	1	0					1	0	1	0	1	←Character
			0	1	1					1	0	1	1	1	Pattern
0	0	1	1	0	0		*	*	*	1	0	1	0	1	
			1	0	1					1	0	0	0	1	
			1	1	0					1	1	1	1	1	
			1	1	1					0	0	0	0	0	←Cursor
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
			0	0	0					1	1	1	1	1	
			0	0	1					1	0	0	0	1	
			0	1	0					1	1	1	0	1	←Character
			0	1	1					1	0	0	0	1	Pattern
1	1	1	1	0	0		*	*	*	1	0	1	1	1	
			1	0	1					1	0	0	0	1	
			1	1	0					1	1	1	1	1	
			1	1	1					0	0	0	0	0	←Cursor

To show the customized icon, simply send the data between "0" to "7" to this module.

For example, this module will display the customized icon at location 64 to 71 upon receiving data "0"; white it will display the customized icon at location 72 to 79 upon receiving data "1".

Watchdog timer is also built in the module. This module will reset itself and send out "reset packet" (0FDH, 0EH) thereafter.

The input must be a standard RS-232 or inverted TTL signal. The RS-232 setting should be:

- *Baud rate: 2400 bps*
- *Parity: None*
- *Data bits: 8*
- *Stop bit: 1*

What follows is the default setup after LCD module initiated:

- *2-line display mode; every character is 5 x 8 dots.*
- *Display on; cursor off; cursor blink off.*
- *Display will be cleared.*
- *Shift right for entry mode.*
- *Set address counter to "00"(cursor position to 0)*
- *In entry mode.*

3.3 Character Generator ROM (CGROM)

Upper 4 bits / Lower 4 bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)															
0001	CG RAM (2)															
0010	CG RAM (3)															
0011	CG RAM (4)															
0100	CG RAM (5)															
0101	CG RAM (6)															
0110	CG RAM (7)															
0111	CG RAM (8)															
1000	CG RAM (1)															
1001	CG RAM (2)															
1010	CG RAM (3)															
1011	CG RAM (4)															
1100	CG RAM (5)															
1101	CG RAM (6)															
1110	CG RAM (7)															
1111	CG RAM (8)															

3.4 Sample Codes

```
/* ************************************
 * EZIO RS232 LCD Control Sample Program
 * ************************************
* ************************************************************************
 * Company:          Portwell Inc.
 * Date:             4/16/2003
 * Program:          02A.c
 * Version:          1.02
 * Compile:          Linux GNU C
 * Purpose:          Direct access to EZIO LCD, the program will display
 *                           messages according to the control button. The current
 *                           version only has the following function:
 *
 *                           1: display welcome message
 *                           2: display UP message if "scroll up" button is pressed
 *                           3: display ENTER message if "ENTER" button is pressed
 *                           4: display ESC message if "ESC" button is pressed
 *                           5: display DOWN message if "scroll down" button is pressed
 *
 * Program Overview:
 *
 *              - Parameters:
 *                        fd              : a file name for open() method, here represents the com port
 *                        Cmd             : command prefix
 *                        cls             : clear command
 *                        init            : initialize command
 *                        blank           : display blank screen
 *                        stopsend        : stop input/output
 *                        home            : move cursor to initial position
 *                        readkey         : set to read from EZIO
 *                        hide            : hide cursor & display blanked characters
 *                        movel           : move cursor one character left
 *                        mover           : move cursor one character right
 *                        turn            : turn on blinking-block cursor
 *                        show            : turn on underline cursor
 *                        scl             : scroll cursor one character left
 *                        scr             : scroll cursor one character right
 *                        setdis          : set character-generator address
 *
 *              - Procedure:
 *                        1. The program sets up the environment, i.e. com port settings.
 *                        2. The main function MUST call init() twice to initialize EZIO
 *                           before any communication.
 *                        3. For executing any command, the command prefix, Cmd, MUST be
 *                           called be command. So all command contains two parts, eg.
 *                           to initialize the sequence of HEX number is 0xFE, 0x25.
 *                        4. After clear screen and display welcome message, ReadKey()
 *                           method must be call to advise EZIO for reading data.
 *                        5. A pooling method is implemented to get input from EZIO while
 *                           any button is pressed.
 *
 *              - NOTE: This program is a sample program provided " AS IS" with NO
 *                           warranty.
 *
 * Copyright (c) Portwell, Inc. All Rights Reserved.
 *
 * ************************************************************************/
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

static int fd;

void SetEnvironment () {
  system("stty ispeed 2400 < /dev/ttyS1");
  system("stty raw < /dev/ttyS1");
}

int Cmd = 254;  /* EZIO Command */
int cls = 1;    /* Clear screen */
void Cls () {
  write(fd,&Cmd,1);
  write(fd,&cls,1);
}

int init = 0x28;
void Init () {
  write(fd,&Cmd,1);
  write(fd,&init,1);
}

int stopsend = 0x37;
void StopSend () {
  write(fd,&Cmd,1);
  write(fd,&init,1);
}

int home = 2    ;   /* Home cursor */
void Home () {
  write(fd,&Cmd,1);
  write(fd,&home,1);
}

int readkey = 6           ;   /* Read key */
void ReadKey () {
  write(fd,&Cmd,1);
  write(fd,&readkey,1);
}

int blank = 8    ;   /* Blank display */
void Blank () {
  write(fd,&Cmd,1);
  write(fd,&blank,1);
}

int hide = 12    ;   /* Hide cursor & display blanked characters */
void Hide () {
  write(fd,&Cmd,1);
  write(fd,&hide,1);
}

int turn = 13    ;   /* Turn On (blinking block cursor) */
void TurnOn () {
  write(fd,&Cmd,1);
  write(fd,&turn,1);
}
```

```
int show = 14  ;   /* Show underline cursor */
void Show () {
  write(fd,&Cmd,1);
  write(fd,&show,1);
}

int movel = 16 ;   /* Move cursor 1 character left */
void MoveL () {
  write(fd,&Cmd,1);
  write(fd,&movel,1);
}

int mover = 20 ;   /* Move cursor 1 character right */
void MoveR () {
  write(fd,&Cmd,1);
  write(fd,&mover,1);
}

int scl = 24;         /* Scroll cursor 1 character left */
void ScrollL(){
  write(fd,&Cmd,1);
  write(fd,&scl,1);
}

int scr = 28;       /* Scroll cursor 1 character right */
void ScrollR(){
  write(fd,&Cmd,1);
  write(fd,&scr,1);
}

int setdis =  64;/* Command */
void SetDis(){
  write(fd,&Cmd,1);
  write(fd,&setdis,1);

 }


/* Add or Change Show Message here */
char mes1[] = "Portwell EZIO";
char mes2[] = "**************";
char mes3[] = "Up is selected";
char mes4[] = "Down is selected";
char mes5[] = "Enter is selected";
char mes6[] = "ESC is selected";
char nul[] = "                                    ";

int a,b;
void ShowMessage (char *str1 , char *str2) {
   a = strlen(str1);
   b = 40 - a;
   write(fd,str1,a);
   write(fd,nul,b);
   write(fd,str2,strlen(str2));
}

int main () {

  SetEnvironment();    /* Set RAW mode */
```

```
  fd = open("/dev/ttyS1" ,O_RDWR);/** Open Serial port (COM2) */

          Init(); /* Initialize EZIO twice */
        Init();

          Cls();   /* Clear screen */
          ShowMessage(mes1,mes2);

   while (1) {
     int res;
     char buf[255];

     SetDis();
     ReadKey(); /* sub-routine to send "read key" command */
     res = read(fd,buf,255); /* read response from EZIO */

     switch(buf[1]) {       /* Switch the Read command */

          case 0x4D :    /* Up Botton was received */
                     Cls();
                ShowMessage(mes1,mes3); /** display "Portwell EZIO" */
                break;                /** display "Up is selected */


       case 0x47 :       /** Down Botton was received */
                Cls();
                ShowMessage(mes1,mes4);  /** display "Portwell EZIO" */
                break;                /** display "Down is selected" */

       case 0x4B :       /** Enter Botton was received */
                Cls();
                ShowMessage(mes1,mes5);  /** display "Portwell EZIO" */
                break;                /** display "Enter is selected" */

       case 0x4E :       /** Escape Botton was received */
                Cls();
                ShowMessage(mes1,mes6);  /** display "Portwell EZIO" */
                break;                /** display "Escape is selected */
      }

   }

}
```