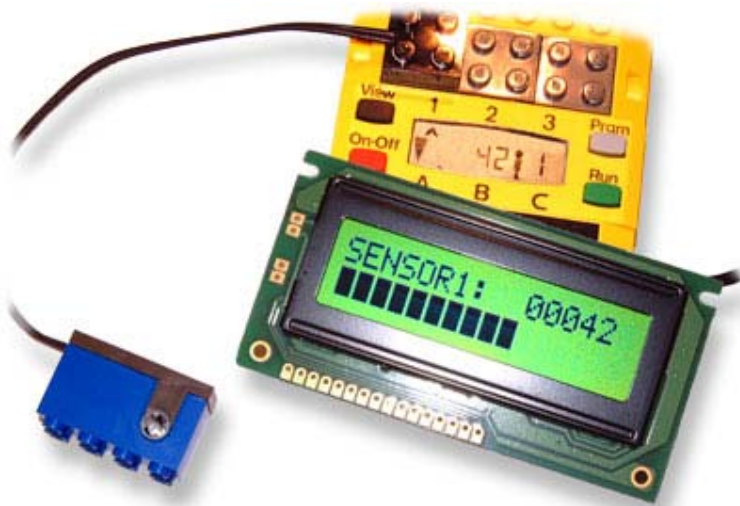


# PromaSoft

---

## PS100 v1.00 User's Manual RCX 16x2 LCD Interface

Hardware version: 1.00 – Document version: 1.02 – Release date: March 16th, 2004



## 1 Table of contents

|      |  |   |
|------|--|---|
| 1    | Table of contents.....                 | 2 |
| 2    | Introduction.....                      | 2 |
| 3    | Support.....                           | 2 |
| 4    | Package contents.....                  | 2 |
| 5    | Main Features.....                     | 2 |
| 6    | Modes of operation.....                | 3 |
| 6.1  | Fast mode.....                         | 3 |
| 6.2  | Slow mode.....                         | 3 |
| 7    | Cursor positions.....                  | 3 |
| 8    | Commands.....                          | 3 |
| 8.1  | Fast command syntax.....               | 3 |
| 8.2  | Slow command syntax.....               | 3 |
| 8.3  | Fast Commands list.....                | 3 |
| 8.4  | Slow Commands list.....                | 4 |
| 8.5  | NQC Commands list.....                 | 4 |
| 9    | Sending commands to the PS100.....     | 4 |
| 10   | Connecting the PS100.....              | 4 |
| 11   | PS100.NQH Include file.....            | 5 |
| 12   | NQC Code examples.....                 | 5 |
| 12.1 | Clear LCD (without PS100.NQH).....     | 5 |
| 12.2 | Clear LCD (with PS100.NQH).....        | 5 |
| 12.3 | Send Text (without PS100.NQH).....     | 5 |
| 12.4 | Send Text (with PS100.NQH).....        | 5 |
| 12.5 | Display a VU Bar (with PS100.NQH)..... | 6 |
| 12.6 | Complex example.....                   | 6 |
| 13   | PS100.NQH Commands.....                | 6 |
| 14   | Notes.....                             | 7 |
| 14.1 | Random IR Reception.....               | 7 |
| 14.2 | Direct commands.....                   | 7 |

## 2 Introduction

The PS100 makes it possible for the RCX (not included) to display *alphanumeric* text on a *2x16 Liquid Crystal Display* (LCD). The LCD is included in the PS100 package contents. The PS100 is controlled by infrared signals sent from the RCX. The advantage of this is that the PS100 will not occupy a sensor port on the RCX. Next to that the PS100 can (when powered by an external power block) be located meters away from the RCX.

Before continuing to read all details you might want to take a look at the examples section in this manual. This will demonstrate that it is "NOT" difficult to control the PS100. From the examples you should be able to start using the PS100.

The PS100 can be controlled by any of the following devices:

- RCX running the standard Lego firmware (2.x).
- RCX running other firmwares such as LEjos, brickOS.
- Any other micro controller which is capable of sending infrared signals.

## 3 Support

This manual provides examples on how to use the PS100 module. All examples, source files and documentation can be downloaded from the PS100 product page at following URL

<http://www.autoreplying.com/rcxsensors/ps100.htm>

Should you experience problems or should you need help using the PS100 you can contact the support department at following email address

<mailto:rcxsensors@autoreplying.com>

## 4 Package contents

The PS100 contains following components

- 16x2 character LCD Display
- Infrared interface (piggybacked to the LCD)
- Lego electric connector to power the PS100
- User's manual

## 5 Main Features

- 2 X 16 Alphanumeric display (2 lines of 16 characters each)
- Controlled by Infrared Signals from the RCX so that no sensor ports are taken up by the PS100.
- Can be powered by an external power block so that no motor ports are taken up by the PS100. This also allows the PS100 to be located meters away from the RCX itself.
- Can be powered by a motor port on an RCX running on rechargeable (NiMH, NiCad) or normal batteries.
- Two modes of operation. Fast to allow fast text generation on the display and slow to allow error free infrared transmission. The fast mode allows up to 200 characters per second to be displayed. The slow mode allows up to 40 characters per second to be displayed.
- Microprocessor controlled
- Internal 90 character command buffer
- Operating voltage: 7v – 10v
- Operating current: 5ma – 10ma
- Built in conversion routines to display decimal, hexadecimal, binary values or VU meters (progress bars).
- Uses the default RCX IR RS232 protocol (2400 baud, 8 data bits, 1 parity bit, 1 stop bit)
- Can be controlled by any IR capable device (not only the RCX)

## 6 Modes of operation

The PS100 has two modes of operation.

### 6.1 Fast mode

The fast mode allows the RCX to send character data to the PS100 at a quick pace (about 200 characters per second) so that real-time dynamic data can be displayed (e.g. real-time sensor values).

The fast mode uses only one byte to display a character.

In almost all cases you can use the fast mode.

### 6.2 Slow mode

This mode is the default mode selected when you first power up the PS100.

The slow mode works slower (about 40 characters per second) than the fast mode but does provide error free infrared transmission.

The slow mode uses five bytes to display a character. This is the overhead needed to guarantee error free transmission.

It is obvious then when the IR signal is not able to reach the PS100, nothing will happen.

When you position the PS100 close enough to the RCX IR Port (within 1 meter) no transmission errors will occur and you can use the fast mode.

## 7 Cursor positions

The PS100 can display two rows of 16 characters each. The position of the first character on the first line is 0 while the position of the first character on the second line is 16.

For example, to position the cursor on the second line you need to move the cursor to position 16 using the

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Line 1 ▶ | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| Line 2 ▶ | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

POS command.

## 8 Commands

### 8.1 Fast command syntax

A fast command consists of a stream of one or two bytes depending on the command. The first byte always is the operand (OP) and the second byte is the argument (AR). The argument is left out for certain commands.

|       |
|-------|
| OP AR |
|-------|

### 8.2 Slow command syntax

A slow command consists of a stream of 5 bytes. The first two bytes are the header bytes and are always 0xEE

and 0xAA. The third byte is the operand (OP). The fourth byte is the argument (AR) and the last byte is the checksum (CS).

|                    |
|--------------------|
| 0xEE 0xAA OP AR CS |
|--------------------|

The checksum needs to be calculated for each command separately and is the sum of the first 4 bytes.

|                                       |
|---------------------------------------|
| Checksum (CS) = 0xEE + 0xAA + OP + AR |
|---------------------------------------|

Below tables list the low level command byte streams, the action the command performs and the equivalent NQC command.

To use the special NQC commands you need to include PS100.NQH in your NQC program. This is discussed later.

All types of commands can be used together.

Refer to the examples section for a clear understand on how to use these commands.

### 8.3 Fast Commands list

| Command     | Action  |
|-------------|---|
| 00 to EF    | Send a character to the display where xx is the ASCII code  |
| F0 xx       | Sets the cursor to the specified position.  |
| F3          | Clear the screen (LCD)  |
| F7          | Clear line 1 (0-15) of the screen   |
| F8          | Clear line 2 (16-31) of the screen  |
| F9 xx FA yy | Display a two byte decimal number (range 0-65536). Xx is the high byte and yy is the low byte. Cursor position remains unchanged. |
| FB FC       | Set SLOW mode. Used to change to slow mode when in FAST mode  |
| FC xx       | Display a VU bar with length equal to xx. A VU bar can be from 0 to 16 characters long. The cursor position remains unchanged.    |
| FD xx       | Display a decimal number with range from 0 to 255. The cursor position remains unchanged.   |
| FE xx       | Display a binary number   |
| FF xx       | Display a hexadecimal number  |

## 8.4 Slow Commands list

| Command                          | Action  |
|----------------------------------|---|
| EE AA 01 xx CS                   | Send a character to the display where xx is the ASCII code  |
| EE AA 00 xx CS                   | Sets the cursor to the specified position.  |
| EE AA 02 00 CS                   | Clear the screen (LCD)  |
| EE AA 02 01 CS                   | Clear line 1 (0-15) of the screen   |
| EE AA 02 02 CS                   | Clear line 2 (16-31) of the screen  |
| EE AA 08 xx CS<br>EE AA 09 yy CS | Display a two byte decimal number (range 0-65536). Xx is the high byte and yy is the low byte. Cursor position remains unchanged. |
| EE AA 02 04 CS                   | Set FAST mode. Used to change to fast mode when in SLOW mode  |
| EE AA 03 xx CS                   | Display a VU bar with length equal to xx. A VU bar can be from 0 to 16 characters long. The cursor position remains unchanged.    |
| EE AA 04 xx CS                   | Display a decimal number with range from 0 to 255. The cursor position remains unchanged.   |
| EE AA 05 xx CS                   | Display a binary number   |
| EE AA 06 xx CS                   | Send an LCD command directly to the LCD. Check the datasheet of the LCD for all available commands                                |
| EE AA 07 xx CS                   | Display a hexadecimal number  |

## 8.5 NQC Commands list

| Command         | Action  |
|-----------------|---|
| _A, _B, _a, ... | Send a character to the display where xx is the ASCII code  |
| _POS(x)         | Sets the cursor to the specified position.  |
| _CLS            | Clear the screen (LCD)  |
| _CLL1           | Clear line 1 (0-15) of the screen   |
| _CLL2           | Clear line 2 (16-31) of the screen  |
| _DEC5(x)        | Display a two byte decimal number (range 0-65536). Xx is the high byte and yy is the low byte. Cursor position remains unchanged. |
| _SLOW           | Set SLOW mode. Used to change to slow mode when in FAST mode  |
| _FAST           | Set FAST mode. Used to change to fast mode when in SLOW mode  |

|          |  |
|----------|--|
| _VU(x)   | Display a VU bar with length equal to xx. A VU bar can be from 0 to 16 characters long. The cursor position remains unchanged. |
| _DEC3(x) | Display a decimal number with range from 0 to 255. The cursor position remains unchanged.                                      |
| _BIN(x)  | Display a binary number  |
| _HEX(x)  | Display a hexadecimal number   |

## 9 Sending commands to the PS100

The way to send a command to the PS100 is over the RCX Infrared port. For specific details on how to send Infrared data from for example RoboLab check your user's manual.

In this manual we will concentrate on using NQC to send commands to the PS100.

What is described here is the low level communication to the PS100 allowing you to control it also from an RCX running other firmware than the original Lego firmware or even other robotics microcontrollers.

The communication protocol is RS232 with 1 start bit, 8 data bits, 1 parity bit and 1 stop bit at 2400 baud which in fact is the default communication protocol used by the RCX. This means that you do not have to change any IR settings in the RCX.

## 10 Connecting the PS100

The easiest way to connect the PS100 is it to plug it into one of the motor ports of the RCX. Then when your program starts make sure to provide power to that motor port; For the examples in this manual, port A is used. It does not matter how you connect the connector to the motor port.

You can also connect the PS100 to an external power block. This can be a Lego power block but can for instance also be a 9V battery block. In this case you of course don't need to send power to a motor port (except to drive motors of course).

When power is applied to the PS100 following text will appear on the display. This indicates that the PS100 is working ok and ready to receive infrared data.



## 11 PS100.NQH Include file

Although it is perfectly possible to only use the SetSerialData and SendSerial NQC commands, it would lead to hard to read code.

To solve this, the PS100 comes with a NQC include file called PS100.NQH. Using the include file has following advantages

- Compact code
- Easy to read code

It is advised to always use PS100.NQH as demonstrated in following examples.

Note that when using PS100.NQH you may put several commands on a single line (separated with a space). For example the command to display the word "Hello" on the second line is written as follows

```
_POS(16) _H _e _l _l _o
```

Check the support website to download the latest available PS100.NQH file.

## 12 NQC Code examples

Following examples are written with or without the use of the PS100.NQH include file. Not using the PS100.NQH include file makes the source code harder to read but demonstrates perfectly how things are done on a lower level.

The reason for this way of working is because NQC unfortunately does not support strings.

### 12.1 Clear LCD (without PS100.NQH)

This example demonstrates an NQC program to clear the display without the use of PS100.NQH and using the SLOW mode.

```
task main()
{
  OnFwd(OUT_A);           // POWER TO LCD
  wait(50);               // WAIT FOR POWER
                          // ABOVE 2 LINES
```

```
                          // NOT NEEDED WHEN
                          // USING EXT POWER
SetSerialData(0,0xEE);   // HEADER BYTE 1
SetSerialData(1,0xAA);   // HEADER BYTE 2
SetSerialData(2,02);     // OPCODE
SetSerialData(3,00);     // ARGUMENT
SetSerialData(4,0x198+2+0); // CS = 0x9A
SendSerial(0,5);         // SEND COMMAND
}
while(true){}           // KEEP RUNNING
```

### 12.2 Clear LCD (with PS100.NQH)

This example demonstrates an NQC program to clear the display with the use of PS100.NQH and using the SLOW mode. This does exactly the same as above example.

```
#include "ps100.NQH"     // INCLUDE ROUTINES
task main()
{
  OnFwd(OUT_A);           // POWER TO LCD
  wait(50);               // WAIT FOR POWER

  _CLS                    // CLEAR THE DISPLAY
}
while(true){}           // KEEP RUNNING
```

### 12.3 Send Text (without PS100.NQH)

This example demonstrates how to send some text to the display without using the PS100.NQH include file

```
task main()
{
  OnFwd(OUT_A);           // POWER TO LCD
  wait(50);               // WAIT FOR POWER

  // --- SEND CHARACTER A ---
  SetSerialData(0,0xEE);   // HEADER BYTE 1
  SetSerialData(1,0xAA);   // HEADER BYTE 2
  SetSerialData(2,01);     // OPCODE
  SetSerialData(3,65);     // ARGUMENT
  SetSerialData(4,0x198+1+65); // CS
  SendSerial(0,5);         // SEND COMMAND

  // --- SEND CHARACTER B ---
  SetSerialData(0,0xEE);   // HEADER BYTE 1
  SetSerialData(1,0xAA);   // HEADER BYTE 2
  SetSerialData(2,01);     // OPCODE
  SetSerialData(3,66);     // ARGUMENT
  SetSerialData(4,0x198+1+66); // CS
  SendSerial(0,5);         // SEND COMMAND
}
while(true){}           // KEEP RUNNING
```

### 12.4 Send Text (with PS100.NQH)

This example demonstrates how to send some text using PS100.NQH in slow mode.

```
#include "ps100.NQH"     // INCLUDE ROUTINES
task main()
{
  OnFwd(OUT_A);           // POWER TO LCD
  wait(50);               // WAIT FOR POWER

  _A _B                    // SEND CHAR A AND B
}
while(true){}           // KEEP RUNNING
```

Line1:100  
Line2:200

## 12.5 Display a VU Bar (with PS100.NQH)

This example demonstrates how to display a VU Bar. The length of the VU Bar corresponds with the value of SENSOR\_1. Connect a rotation sensor to INPUT 1

```
#include "PS100.NQH"          // INCLUDE ROUTINES
task main()
{
    SetSensor(SENSOR_1,SENSOR_ROTATION);

    OnFwd(OUT_A);              // POWER TO LCD
    wait(50);                  // WAIT FOR POWER

    _FAST                       // SELECT FAST MODE

    // CLEAR THE LCD AND DISPLAY LINE 1

    _CLS _S _E _N _S _O _R _1 _COLON

    start dispens;             // START TASK TO
                                // DISPLAY VU BAR

    while(true){}             // KEEP RUNNING
}

task dispens()
{
    while(true)
    {
        _POS(10)
        _DEC5(SENSOR_1)
        _POS(16)                // MOVE CURS TO POS 16
        _VU(SENSOR_1/4)         // DISPLAY VAL OF SENS1
    }
}
```

The result of running above application is as follows



## 12.6 Complex example

To demonstrate the "ease of use" look at below example that performs some actions with the display. Following lines of code will switch to FAST mode, clear line 1, set cursor to position 0, display the word Line1 followed by a colon, display the decimal number 100 and then do the same for line2.

```
_FAST _CLL1 _POS(0) _L _i _n _e _1 _COLON _DEC3(100)
_CLL2 _POS(16) _L _i _n _e _2 _COLON _DEC3(200)
```

The resulting display is as follows

## 13 PS100.NQH Commands

Below is a list of all commands defined in PS100.NQH. We advise to use this include file to allow for the most compact code possible. Note that all commands defined in PS100.NQH are prefixed with an underscore to prevent possible conflict with variable, function and subroutine names out of your own code. You are free to add more commands to this include file if needed.

| Command   | Action  |
|-----------|---|
| _FAST     | Switch to fast mode   |
| _INITIR   | Initialize the RCX IR Interface   |
| _SLOW     | Switch to slow mode   |
| _POS(P)   | Set the cursor to position P where P ranges from 0 to 31  |
| _VU(P)    | Display a VU bar of length P where P ranges from 0 to 16. Cursor position remains unchanged.                      |
| _DEC3(P)  | Display the number P in decimal format and 3 digits. P ranges from 0 to 255. Cursor position remains unchanged.   |
| _DEC5(P)  | Display the number P in decimal format and 5 digits. P ranges from 0 to 65536. Cursor position remains unchanged. |
| _BIN(P)   | Display the number P in binary format. P ranges from 0 to 255. Cursor position remains unchanged.                 |
| _HEX(P)   | Display the number P in hexadecimal format. P ranges from 0 to 255. Cursor position remains unchanged.            |
| _ASCII(P) | Send the number P is character format. P ranges from 0 to 239   |
| _CLS      | Clear the LCD   |
| _CLL1     | Clear line 1 of the LCD   |
| _CLL2     | Clear line 2 of the LCD   |
| _A        | Display "A"   |
| _B        | Display "B"   |
| ...       | ...   |
| _Z        | Display "Z"   |
| _a        | Display "a"   |
| _b        | Display "b"   |
| ...       | ...   |
| _z        | Display "z"   |
| _SPACE    | Display a space   |
| _EXL      | Display "!"   |
| _DQUOTE   | Display double quotes   |
| _HASH     | Display "#"   |

|           |                       |
|-----------|-----------------------|
| _DOLLAR   | Display "\$"          |
| _PERCENT  | Display "%"           |
| _AND      | Display "&"           |
| _QUOTE    | Display single quotes |
| _OBRACKET | Display "("           |
| _CBRACKET | Display ")"           |
| _STAR     | Display "**"          |
| _PLUS     | Display "+"           |
| _COMMA    | Display ","           |
| _MINUS    | Display "-"           |
| _DOT      | Display "."           |
| _SLASH    | Display "\"           |
| _0        | Display "0"           |
| _1        | Display "1"           |
| _2        | Display "2"           |
| _3        | Display "3"           |
| _4        | Display "4"           |
| _5        | Display "5"           |
| _6        | Display "6"           |
| _7        | Display "7"           |
| _8        | Display "8"           |
| _9        | Display "9"           |
| _COLON    | Display ":"           |
| _UND      | Display "_"           |

## 14 Notes

### 14.1 Random IR Reception

When in fast mode it is normal that the PS100 will display seemingly random characters when an application is being uploaded to the RCX and the PS100 is powered. This is because the PS100 will pickup any infrared signal that is being transmitted by the RCX or even the remote control of your TV set, DVD Player and so on.

### 14.2 Direct commands

It is possible to send direct commands to the LCD. This allows you to turn on a blink cursor and more. To send a direct command use following byte stream in slow mode

|                |
|----------------|
| EE AA 06 xx CS |
|----------------|

Examples

- Turn on block cursor: EE AA 06 0D CS
- Turn off block cursor: EE AA 06 0C CS
- Shift display: EE AA 06 1E CS