Technical Manual & Programming Guide

Series PHL-1600

Handheld Laser Data Collection Terminal

Manual No. 25-TRMTK-01

Revision 2.0 / January 1999

Opticon

8 Olympic Drive
Orangeburg, NY 10962
Tel 914.365.0090
Fax 914.365.1251

Technical Manual
Version 2.0 January 1999
Part Number 25-TRMTK-01

Forward

Thank you for purchasing the Opticon PHL-1600 Laser Terminal. This manual is separated into two distinct parts; a USER'S GUIDE section, which is devoted to the basic operation and setup of the PHL-1600, and a PROGRAMMER'S & LIBRARY REFERENCE manual, which addresses the procedures to generate and compile programs for the terminal.

The USER'S GUIDE section will provide detailed procedures for terminal setup, self-testing, and establishing parameters for the various communication options.

The PROGRAMMER'S & LIBRARY REFERENCE manual provides abbreviated instructions for using the IAR Systems Cross Compiler to generate machine-loadable code from original application program source code. It also contains a complete listing of library functions to enable access to, and control of, the hardware functions of the PHL-1600 for use when writing original code, or converting application code from another terminal.

We believe the information contained herein to be correct, however, if any difficulty is encountered, please contact Opticon via the options shown in the HELP appendix.

Table of Contents

Users Guide

Introduction

Please read this section carefully before using the terminal to become familiar with how the unit operates. This section explains the basic operations of the terminal.

Please refer this guide for precautions regarding usage and handling. For programming and downloading information please refer to the enclosed materials.

Important Handling Instructions

REMARKS: *Opticon does not cover damage resulting from accident, misuse, abuse, improper operation, lack of reasonable care, unauthorized modification, the affixing of any attachment not provided by Opticon with the unit and loss of parts.*

1) Read and understand all instructions in this manual.
2) Do not disassemble the unit. Doing so will damage the sensitive electronics inside.
3) When necessary, clean the unit with a dry, soft cloth. Do not clean with water or volatile liquids such as thinner or benzene.
4) Keep the unit away from sources of excessive heat (including direct sunlight), moisture, and dust.
5) Never leave the unit in a car.
6) Do not allow any liquids to spill onto or into the unit.
7) Avoid using or storing the unit in areas where condensation may occur on the unit. If condensation has occurred, do not use the unit until the condensation has evaporated.
8) Do not throw, drop or strike the unit. Treat the unit as you would any fine electronic instrument.
9) Be sure to press the keys on the unit with your finger or with the tip of something soft and round (i.e. a pencil eraser). Using a sharp, pointed object can damage the unit.
10) Do not place anything on the top of the unit.
11) Only install batteries recommended in this manual.

Regulatory Information

☞ **CAUTION**: Changes or modifications not expressly approved by the party responsible for compliance to the FCC rules could void the user's authority to operate this equipment.

Important Safety Instructions

☞ **CAUTION**: To reduce the risk of fire or injury to persons, read and follow these instructions:

1) Use only the following type and size batteries: AA Alkaline 1.5V, Lithium CR2032.
2) Do not dispose of the batteries in a fire. The cells may explode. Check with local codes for possible special disposal instructions. **Do not open or mutilate the batteries. Released electrolyte is corrosive and may cause damage to the eyes or skin. It may be toxic if swallowed.**
3) Exercise care in handling batteries in order not to short the battery with conducting materials such as rings, bracelets, and keys. The battery or conductor may overheat and cause burns.
4) Do not attempt to recharge the alkaline batteries provided with or identified for use with this product. The batteries may leak corrosive electrolyte or explode.
5) Do not attempt to rejuvenate the batteries provided with or identified for use with this product by heating them. Sudden release of the battery electrolyte may occur causing burns or irritation to eyes or skin.
6) When inserting the batteries into this product, the proper polarity or direction must be observed. Reverse insertion of batteries can cause charging, which may result in leakage or explosion.
7) Discard "dead" batteries as soon as possible since they are more likely to leak in a product.
8) Do not store this product, or the batteries provided with or identified for use with this product, in high-temperature areas. Batteries that are stored in a freezer or refrigerator for the purpose of extending cell life should be protected from condensation during storage and defrosting. Batteries should be stabilized at room temperature prior to use after cold storage.

Package Contents

Carefully remove the contents from the shipping carton, confirming receipt of the items described below. If there are any missing or damaged parts, return the unit with packaging to the place of purchase.

The package contains:

- 1 ea. PHL-1600 Hand Held Laser Terminal
- 2 ea. Main AA batteries
- 1 ea. Lithium backup battery
- 1 ea. Wrist strap
- 1 ea. Quick Start Instructions
- 1 set *Rapid* GEN™ Application Generator software (see Appendix 4)

Chapter 1
Setting Up

Main Battery Installation

The PHL-1600 Terminal uses 2 types of batteries; the main batteries (2 AA type alkaline dry cells or an NiMH rechargeable battery pack[1]) and a backup battery (a coin type lithium battery).

Insert the batteries included with the unit before usage. Under normal usage, the main batteries are good for approximately 120 hours of operation (based on 2 readouts every 10 seconds at normal temperature).

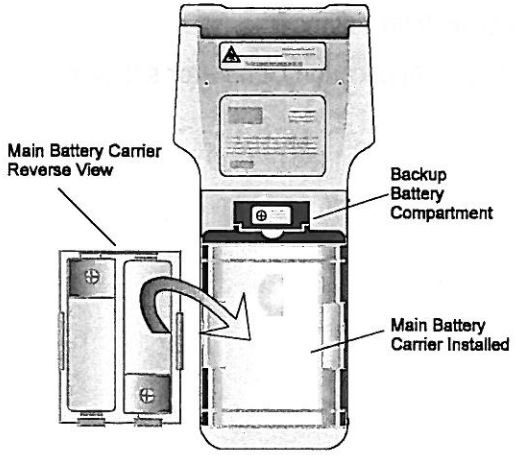


Fig. 1-1

Opticon PHL-1600 Laser Terminal
(Rear View With Battery Cover Removed - Not To Scale)

[1] A charger for the NiMH battery pack is available. Contact your supplier or Opticon for additional information.

Main Batteries

1) Remove Battery Case Cover. Press in the dimple at the bottom of the unit and push outward. The battery case cover will release. Remove cover.
2) Remove Battery Case. Hold unit in palm of one hand. Using the other hand, pinch the sides of the battery case and pull it out of the unit.
3) Insert Batteries. If using AA-type batteries, insert into the battery case, noting the correct polarity. Place battery case back into unit, being certain the case is positioned correctly (battery case terminals should be positioned toward bottom of unit). Case should click into place. If using NiMH battery pack, insert the entire battery pack observing the procedure above.
4) Replace battery case cover.

REMARKS: *When using NiMH rechargeable battery pack, do not attempt to remove battery cells from holder. NiMH cells are not user replaceable.*

Backup Battery Installation

Backup Battery - Lithium Button Cell CR2032

1) Remove Battery Case Cover. Press in the dimple at the bottom of the unit and push outward. The battery case cover will release. Remove cover.
2) Remove Battery Cover. The backup battery is located directly above the main battery case (see illustration). Release the backup battery cover by inserting a flat-blade screwdriver under the center of the cover and lift up.
3) Insert Battery. Insert lithium battery, noting the correct polarity as indicated. Do not use lithium button cells other than the type designated (CR2032) in this manual.
4) Close backup battery cover and replace battery case cover on unit.

Low Battery Indication

Check and replace or recharge batteries, as applicable, when the low main battery mark ( ) or the low backup battery mark ( ) is indicated on the display LCD. Make sure the power is turned off when changing the batteries.

IMPORTANT - To prevent erasing user-installed programs and stored data, observe the following:

Never remove the main batteries and the backup battery (lithium button cell) at the same time.

1) Do not remove the lithium battery until the unit's power is turned off.
2) Insert new AA-type alkaline dry cells in order to prevent the backup battery (lithium cell) from discharging if the unit will not be used for an extended period of time.

Damage to the unit and/or loss of data resulting from improper battery installation and management as described above is the sole responsibility of the user.

Handstrap

Insert the narrow loop of the handstrap through the opening at the base of the unit. Thread the braided handle of the handstrap through the narrow loop and pull. This will secure the handstrap to the unit. Place the handstrap over your wrist when using the PHL-1600 to prevent the unit from falling. Never wrap the handstrap around the unit where it could damage the keys on the keypad.

Chapter 2
Operation

Guide To Controls

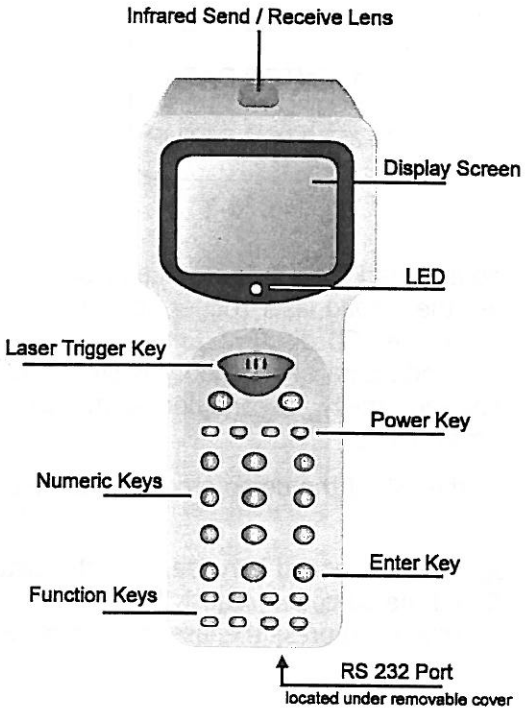


Fig. 2-1

Bar Code Reading

Turn the unit's power on by depressing the red "PW" power key. If the unit has not been pre-programmed with a user's application program, the Opticon default demo program* will activate and the following display will be shown on the LCD screen:

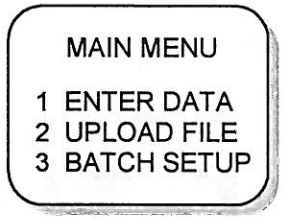


Fig. 2-2

Depress the (1) key and a "Data Entry" screen will appear. Position the scanner lens over the barcode. Press the oblong laser trigger key. The barcode information will appear on the third line of the LCD display. Enter a random quantity and depress the (ENT) key. The bar code data and quantity is stored in memory an the data entry screen is re-displayed, ready for another scan. Depressing the (F1) key returns the unit to the main menu.

*Further information on the use of the demo program can be obtained from Opticon's web site.

If the unit does not register the scanned barcode, alter the distance and angle between the PHL-1600 and the barcode. Adjust the distance so that the laser line strikes wider than the barcode and press the laser trigger key again.

Chapter 3
Setting Functions

Booting up the Utility Screen

1) With the power off, depress the (0) (ENT) (PW) keys simultaneously and the following utility screen will appear:

2) Move the highlight bar to the desired utility function by depressing the (F7▲) (Q2) or (F8▼) (Q1) keys. These keys will be used to move the highlight bar for all following utility functions.

3) When the highlight bar is over the desired selection, depress the (ENT) key.

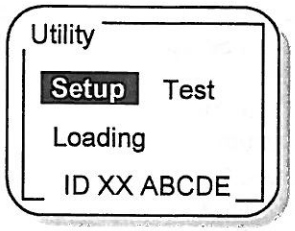


Fig. 3-1

Operation of Setup Menu

1) With the utility screen displayed, position the highlight bar over "Setup" and depress the (ENT) key. The screen at right will be displayed.

REMARKS: *To access these screens and the additional available selections, scroll down the highlight bar using the (F8▼) or (Q1) keys as previously described. Scrolling down past the last displayed selection will advance the cursor to the next screen. When reaching the last available selection, the highlight bar will no longer advance.*

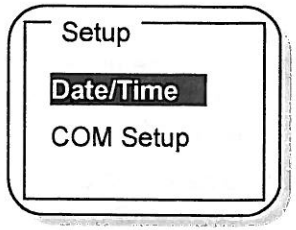


Fig. 3-2

Setting Functions

The following functions can be set from the setting menu:

Selection	Description
Date/Time	Set date and time
COM Setup	Set COM parameters
Backlight	Set delay of auto backlight off
Auto Power	Set delay of auto power off
Resume	Set resume yes/no
Special Key	Enable contrast and backlight keys
ID Setup	Set ID of the terminal
RAM Disk	Format RAM disk

To select function to be set, position the highlight bar over the desired function and depress the (ENT) key. After setting any function as described in this chapter, you may return to the Setup menu by depressing the (F2) key or return to the Utility menu by depressing the (F1) key (except where otherwise noted).

Setting Date/Time

When Date/Time is selected, the screen at right will be displayed.

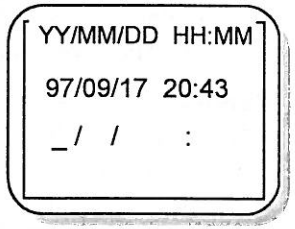


Fig. 3-3

1) Using the numeric keys, enter the desired year, month, day and time (24 hour clock) in the space provided.
2) Once all the digits are entered, the display screen will automatically return to the Setup menu.

REMARKS: *When entering settings, mistakes can be deleted by depressing the (CLR) or (BS) key, once for each incorrect field grouping. (F2) will cancel operation at any point in the data entry sequence, producing no changes and returning to the setup menu screen.*

Setting COM Port

Upon selection, the Com Ports menu will be displayed (Fig. 3-4).

1) Position the highlight bar over the desired COM Port setting and depress the (ENT) key.
2) The COM Setup menu will be displayed.

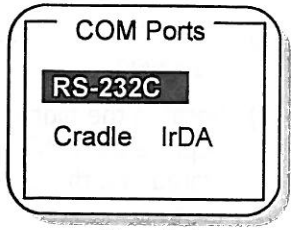


Fig. 3-4

Setting COM Parameters

Following selection of the desired COM port, the Com Setup menu (Fig. 3-5) will be displayed.

1) Position the highlight bar over the desired parameter setting and depress the (ENT) key. Enter the desired settings.

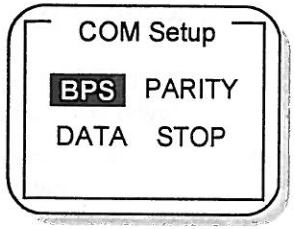


Fig. 3-5

Setting Baud Rate Parameters

1) When "BPS" is selected, the Baudrate menu will be displayed.
2) Position the highlight bar over the desired setting and depress the (ENT) key. The selected baud rate will be stored and the display will return to the COM Setup menu (Fig. 3-5).

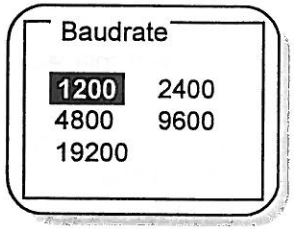


Fig. 3-6

Setting Parity Parameters

1) When "PARITY" is selected, the Parity menu will be displayed.
2) Position the highlight bar over the desired setting and depress the (ENT) key. The selected parity will be stored and the display will return to the COM Setup menu (Fig. 3-5).

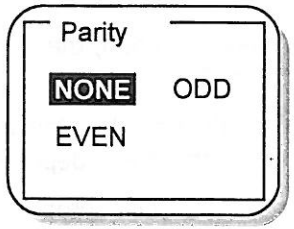


Fig. 3-7

Setting Data Bit Parameters

1) When "DATA" is selected, the Data Bits menu will be displayed (Fig. 3-8).
2) Position the highlight bar over the desired setting and depress the (**ENT**) key. The data bit selection will be stored and the display will return to the COM Setup menu (Fig. 3-5).

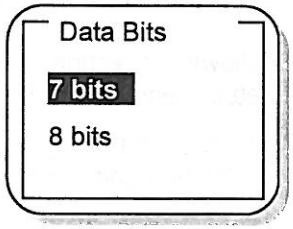


Fig. 3-8

Setting Stop Bit Parameters

1) When "STOP" is selected, the Stop Bits screen (Fig. 3.9) is displayed.
2) Position the highlight bar over the desired setting and depress the (**ENT**) key. The stop bit selection will be stored and the display will return to the COM Setup menu (Fig. 3-5). Depress the (**F2**) key to return to the Setup menu.

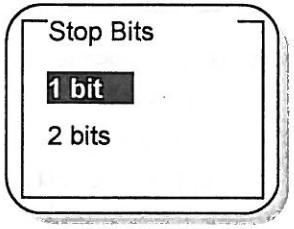


Fig. 3-9

Setting Backlight

1) When "Backlight" is selected, the screen at right will be displayed.
2) Using the numeric keys, enter the amount of delay in minutes. The allowable range is from 0 to 60 minutes. Entering a delay of 0 min will disable the auto-backlight off feature. The (BS) or (CLR) key may be used to clear incorrect entries.
3) Depress the (ENT) key and the Setup menu will be displayed.

Fig. 3-10

Setting Auto Power Off

1) When "Auto Power" is selected, the screen at right will be displayed.
2) Using the numeric keys, enter the amount of delay in minutes. The allowable range is from 0 to 60 minutes. Entering a delay of 0 min will disable the auto-power off feature.
3) Depress the (ENT) key and the Setup menu will be displayed.

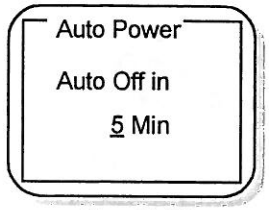


Fig. 3-11

Setting Resume

1) When "Resume" is selected, the screen at right will be displayed.
2) Depress the (F7▲) or (Q1) key to toggle the resume function between On and Off.
3) Depress the (ENT) key and the Setup menu will be displayed.

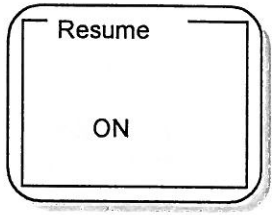


Fig. 3-12

Setting Special Key

1) When "Special Key" is selected, the screen at right will be displayed.
2) Position the highlight bar over the desired function.
3) Depress the (ENT) key to toggle the function settings between On and Off. Contrast will enable the (F7▲) and (F8▼) keys to control the display contrast when in the shift mode. "Backlight" will enable the (F4/BL) key to switch the backlight On or Off when in the shift (S) mode.
4) Depress the (F2) key and the Setup menu will be displayed.

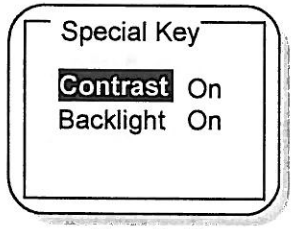


Fig. 3-13

ID Setup

1) When "ID Setting" is selected, the screen at right will be displayed.
2) Using the numeric keys, enter the desired terminal ID number. The allowable range is from 001 to 128.
3) Depress the (ENT) key and the Setup menu will be displayed.

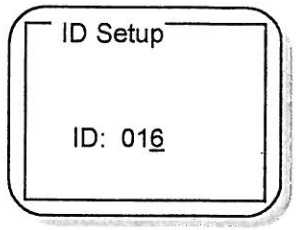


Fig. 3-14

Setting RAM Disk

It is necessary to set the RAM Disk function (one time) after the application creates a file.

1) When "RAM Disk" is selected, the following screen is displayed:
2) Depressing the (ENT) key starts the RAM disk formatting process and depressing the (F2) key stops the process. After a short delay, the process is completed and the screen at right will be displayed. Upon completion, the RAM size is specified as illustrated above.
3) Depress the (F2) key and the Setup menu will be displayed.

☞ **CAUTION**: The RAM Disk contents are erased by formatting.

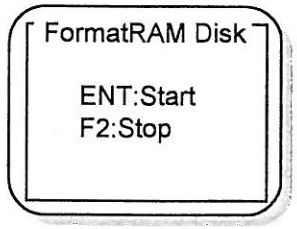


Fig. 3-15

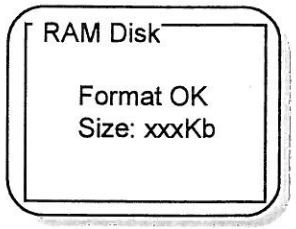


Fig. 3-16

Chapter 4
Testing Functions

Test Menu

1) In order to test the unit's functions, go to the utility screen (Fig. 3-1), position the highlight bar over "Test" and depress the (ENT) key.

2) When "Test" is selected, the screen at right will be displayed. The following functions can be selected and tested in this menu:

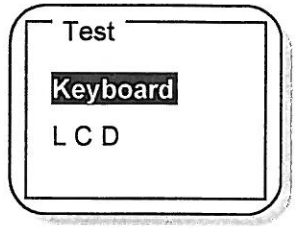


Fig. 4-1

Function	Description
Keyboard	Key depression
LCD	LCD display
COM	COM functions
RAM+ROM	Current status
CGROM	CGROM functions
Buzzer/LED	Buzzer/LED functions

3) Position the highlight bar over the desired function to be tested and depress the (ENT) key.

Keyboard Test

1) When "Keyboard" is selected, the screen at right will be displayed.

2) Upon depressing each key, a corresponding display screen circle will be highlighted. One circle should highlight for each key depressed, indicating a properly functioning keyboard.

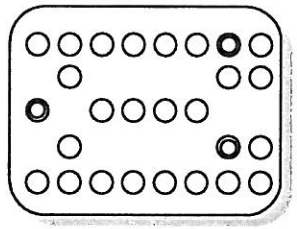


Fig. 4-2

3) Depress the (ENT) (0) (Trigger) keys simultaneously to conclude the test and return to the menu screen.

LCD Test

When "LCD" is selected, the screen at right will be displayed.

1) Depress the (ENT) key and another screen is displayed, as above, displaying a smaller character set (12 Dots).
2) Depress the (ENT) key and a rectangular matrix will appear and highlight itself in an inward manner until the complete rectangular matrix is highlighted.
3) The screen will automatically return to the test menu when the test is completed.

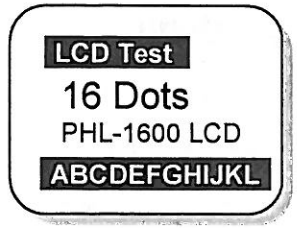


Fig. 4-3

COM Test RS-232C

1) When "COM" is selected, the COM Testing screen (Fig. 4-4) is displayed.
2) Position the highlight bar over "RS-232C" and depress the (ENT) key. The unit will beep and the screen shown in Fig. 4-5 will be displayed.
3) Depress the (ENT) key to return to the COM Testing screen.

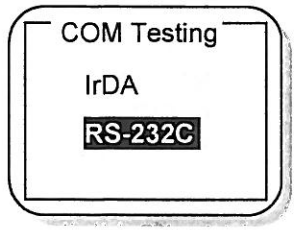


Fig. 4-4

Fig. 4-5

COM Test IrDA

REMARKS: *Two Opticon handheld laser terminals are required to run this test. Both terminals must be set to the identical COM parameters.*

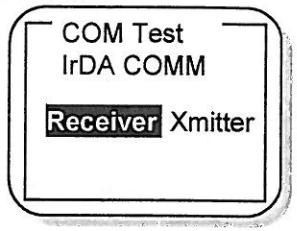


Fig. 4-6

1) When "IrDA COMM" is selected, the COM Test screen (Fig. 4-6) be displayed.
2) Position highlight bar over "Receiver" or "Xmitter" to define the testing mode for each hand held terminal.
3) Position and align the two terminals on a flat surface as illustrated:

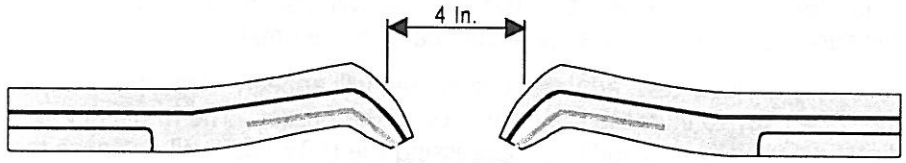


Fig. 4-7

Depress the (ENT) key on the terminal designated as the receiver and immediately depress the (ENT) key on the terminal designated as the transmitter.

REMARKS: *Always start this test from the receiver.*

4) A screen will appear identifying the receiver and transmitter and a normal end message will be displayed. If the test fails, the receiver and/or the transmitter will display a "COM 1 Time Out" message.
5) Depress (F2) key to return to the Test menu screen.

REMARKS: *If the test is accidentally run on a single unit, after a short delay, a "Time Out" message will appear. Depress (F2) to return to the Test screen.*

RAM + ROM Test

This function is disabled.

CGROM Test

This function is disabled.

Buzzer/LED Test

1) When "Buzzer/LED" is selected, the terminal will first run an LED test. The LED will flash Red - Green - Orange sequentially, three times.
2) Following the LED test, a buzzer test screen will appear. Repeatedly depress the (F7▲) key to increase the frequency of the buzzer. The frequency will be displayed visually and audibly. Depressing the (F8▼) key will decrease the frequency during this process. When the desired frequency is achieved, depress the (ENT) key.
3) Depress the (F2) key to return to the Test menu.

Bar Code Test

1) When "Barcode" is selected, the screen at right will be displayed.
2) Position the scanner lens over the bar code. Press the trigger key, and the bar code information will appear between the upper set of brackets on the LCD screen. The identified symbology and number of digits in the bar code will be displayed at the bottom of the LCD screen.
3) If the unit does not register the scanned barcode, alter the distance and angle between the PHL-1600 and the barcode. Adjust the distance so that the laser line strikes wider than the barcode and press the laser trigger key again.
4) Depress the (F2) key to return to the Test menu.

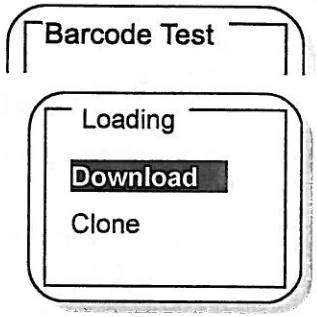


Fig. 5-1

Chapter 5
Loading Applications

Loading Menu

☞ **CAUTION:** *Before proceeding, be sure to enter the Setup menu and select the COM Port that you wish to use for downloading or copying.*

1) Go to the utility menu screen (Fig.3-1) by depressing the (F1) key. Position the highlight bar over "Loading" and depress the (ENT) key. The screen at right will be displayed. The functions in this menu perform downloading of application programs and copying of application programs between terminals.
2) Position the highlight bar over the desired function and depress the (ENT) key.

☞ **CAUTION:** *Do not perform downloading or copying without a host or another terminal in place to either transmit or receive data. If this occurs, an error message will appear. Depress (F1) to return to the Utility menu.*

Downloading

When "Download" is selected, the screen at right will be displayed.

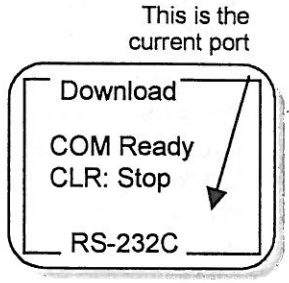


Fig. 5-2

1) When downloading is selected and the above screen is displayed, the terminal is in receive ready mode. The red LED will be illuminated.
2) During downloading communication, the red LED will blink.
3) When downloading is complete, the buzzer will sound and a completion notice will be displayed. The green LED will be illuminated.
4) To stop communication, depress the (CLR) key. A stop notice will be displayed. To resume communication, depress the (ENT) key. To return to the Utility menu, depress the (F1) key.

Clone Applications

REMARKS: *Two Opticon PHL-1600 Laser Terminals are required to perform this function.*

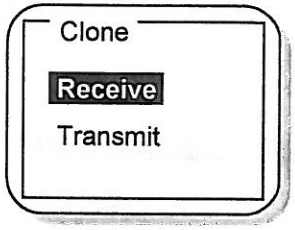


Fig. 5-3

1) When "Clone" is selected, the screen at right will be displayed.
2) Position highlight bar over Receive or Transmit to define testing mode for each hand held terminal.
3) Position and align the two terminals on a flat surface as illustrated below:

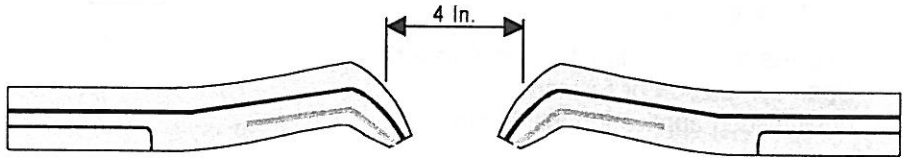


Fig. 5-4

4) Depress the (ENT) key on both terminals to begin the copy process.
5) When the sender identifies the receiver, the red LED on the receiving terminal will blink.
6) Upon completion, the green LED on both terminals will illuminate, a beep will sound, and a completion message will be displayed.
7) Depress (F1) key to return to the utility menu screen.

REMARKS: *If the receiver or transmitter does not connect with another device, it will continuously attempt to communicate. Depress the (CLR) key to terminate this process and then depress (F1) to return to the Utility menu.*

PC Remote

This function is disabled.

Start Applications

1) When an application has been downloaded, the application starts when the terminal power is reset. If the disk format is not complete at this stage, the screen at right will be displayed.

2) Depress the (ENT) key to start formatting. Upon completion of formatting, the downloaded application will begin.

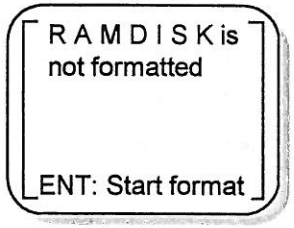


Fig. 5-5

Programmers & Library Reference Manual

Contents

Introduction

The programmer's and library reference guide is provided to offer assistance in creating programs to be run on the PHL-1600 terminal. A program may be "created" in several ways. One path is to write original source code and compile it to be downloaded and run on the terminal. Another path is to use an existing program written for another terminal, modify it for use on the PHL-1600, and compile and download it for subsequent use. The procedures for pursuing either path is contained herein with the exception of creating "new" source code. That is left to the creative talents of the user.

This guide will give the user a working knowledge of the use of the ICC 7700 Cross Compiler but is not meant to replace the detailed information given in the compiler's documentation. The user should read all the available information to familiarize him/herself with all the available features.

It is assumed that the user is reasonably familiar with the operation of a PC and detailed operations involving specific DOS or operating commands are not given.

General Development Procedure

The user will create application software written in the C language in whatever environment is easiest and most convenient. The completed (source) code is then compiled into assembly language code for the target processor. If no errors are Remarked in the compilation process, a binary (executable machine language) HEX file is then created for downloading to the PHL-1600. If errors are encountered, refer to the compiler manual for interpretation of error messages, make corrections as necessary, and recompile. Repeat this process until a successful compilation is achieved and a HEX file can be created.

Upon generation of the HEX file, it is downloaded to the PHL-1600 using the procedures shown here and in the User's Guide section. The application can now be tested.

REMARKS: *A successful compilation does not mean that the application will perform as desired. It only means that no coding errors were generated in creating the source code. The entire process of generating, compiling and downloading is reiterative until the application runs on the PHL-1600 exactly the way expected.*

Hardware/Software Requirements

- System Requirements
  PC with MS DOS 3.0 or higher
  1.4 MB floppy disk drive
  Hard disk with 40 MB available
  RAM Memory; 1 MB minimum, 4 MB recommended
- Compiler
  ICC 7700 IAR Systems C Cross Development Kit, part # : ICC7700PCV110
- Miscellaneous
  Application Development Environment disk (supplied with this manual)
  PHL-T1600 RS232 Communications Cable (Opticon part #: 41-T1600-001)

Installing the Compiler & Application Development Environment

Installing the Compiler

The compiler software is contained on two 1.4 MB floppy disks. For the software to operate properly, a control connector assembly is supplied with the development kit. This connector MUST be plugged into the parallel port of the PC to be used with the compiler. If there is a (printer) cable plugged into this port, remove it, insert the control connector in its place, and re-plug the cable into the connector provided on the control connector.

REMARKS: *The compiler will not operate without the control connector in place.*

With the computer on and operating, exit all programs and Windows. At the DOS prompt in the root (C:/) directory, make a new directory to which the compiler will be installed. For illustration purposes, the directory will be assigned the (arbitrary user-defined) name "ICC."

Go to the ICC directory. Insert Compiler Install Disk #1 into the floppy drive and then type "a:install" to begin the installation process.

REMARKS: *Any screen which displays a message and does not require user input can be exited by depressing the ENTER key to proceed to the next step in the installation process.*

When asked to name the IAR root, change it to ICC as designated above. Follow the prompts to complete the installation. Remove the 2nd floppy disk and change the working directory to ICC.

It is required that changes to the autoexec.bat file be made prior to running the compiler. Detailed information on the changes necessary may be found in the AUTOEXEC.IAR file. To view the contents of this file, display it on the screen by using the TYPE command, i.e., TYPE AUTOEXEC.IAR. Modify AUTOEXEC.BAT by performing the following:

Using the editor, open AUTOEXEC.BAT and create (or change*) the path statement to all IAR executable (exe) and user interface (ui) files to include the ICC directory created above, e.g., the path statement must include the following:

C:\ICC\EXE
C:\ICC\UI

Next, define the environment variables for the IAR tools. While still in the EDIT mode, create (or change*) the following in the AUTOEXEC.BAT file as follows:

```
SET C_INCLUDE=C:\ICC\INC\
SET XLINK_DFLTDIR=C:\ICC\LIB\
```

Save the modified AUTOEXEC.BAT file and exit the editor.

REMARKS: *REBOOT the computer to make the new autoexec.bat effective.*

The compiler is now ready to be used.

- *Changing is required only if the statements are already in the autoexec.bat file but under another user-defined name.*

Installing Application Development Environment

Make a new directory under the directory created for the compiler. For illustration purposes, it shall be named "appl."

e.g., C:\ICC>MD appl

Go to the appl directory and copy all files on the Application Development Environment disk to the appl directory. This completes the installation process.

Modify Example Application Program

Using any text editor, open Example.c and modify the character string that defines the title of the main menu screen from "MAIN MENU" to "DEMO MENU." This will be used to differentiate the demo program shipped with the PHL-1600 from the compiled Example program used to demonstrate the operation of the compiler.

Compiling Source Code & Creating an Executable HEX File

Upon completing the application source code, and if created on a PC other than where the compiler has been installed, the source code must be transferred to a floppy disk to be used on the PC containing the compiler. If created on the PC where the compiler is resident, save the completed application source code in the appl directory.

Preparation

On the PC where the compiler is installed, return to the user (e.g., ICC) directory. If the application source code is stored on a floppy disk, go to the appl directory and copy the source code file from the floppy disk to this directory. For illustrative purposes, this file shall be named "Example.c".

Compilation & Creating an executable HEX file

Type the following:*

PHLMAKE

The source code will be first compiled into object code and then into an executable hex file named Example.hex.

* PHLMAKE is an executable macro that performs the combined function of compiling and linking to a hex file.

The machine language code is now ready to be downloaded to the PHL-1600.

Troubleshooting for the Hardware Lock (Dongle Key)

If the Embedded Workbench reports a missing hardware lock (dongle):

Should you receive an error message "missing hardware lock", there are a series of issues that should be investigated. Please review the following checklist and make the modification that are applicable to your operating system.

1) **Hardware key (dongle) inserted** - Make sure that the hardware key (dongle) is properly inserted in the parallel port of your PC.

2) **If running under a high speed Pentium PC.** For some high speed Pentium computers the hardware lock might have synchronization problems. To solve this, the environment variable SSI_ACT can be set in the AUTOEXEC.BAT (for Windows 3.x or Windows 9x) or in the Control Panel/System Dialogue Box (for Windows NT) according to following: set SSI_ACT=10,10,20. Restart your computer, and try again. If the hardware lock still is not working, try increasing the numbers gradually in steps of 10 or 20 (the maximum limit is 200).

   Further information: These parameters control the delay times in the hardware lock libraries for each stage of the interrogation process of the Activator/M (power up, output to the port, input from the port). On certain computers, the hardware configuration may be such that default times are not appropriate. The default settings in the hardware lock for delay parameters of 2,2,5 are sufficient for 486 and Pentium 75 to 133 computers. On certain computers, the hardware configuration may be such that default delay times are not appropriate. The optimal parameter setting for 166Mhz, 200Mhz, 300Mhz Pentium Pro has shown to be 10,10,20.

   YOU MUST RESTART YOUR COMPUTER

   If the "missing hardware lock" error persists, then try increasing the numbers gradually in steps of 10 or 20 (the maximum limit is 200).

3) **Change the settings for the Parallel port.** Try the "lowest" possible BIOS mode for the LPT port, e.g., normal, standard, AT compatible, Uni-directional, dumb, only output mode, etc. and avoiding "extended modes". After changing the LPT settings, change the SSI_ACT settings as described above and play around with different combinations of both.

   Further information: The hardware lock system requires a basic technical standard from the parallel port. For example, the port must be bi-directional (allow both read and writing). This fact prevents it to be used with some notebooks and high-speed machines. Also, the port needs to be able to provide a certain voltage level on some pins. Connected equipment, like printers, behind the hardware lock could possibly decrease the voltage level so that the hardware lock fails. A quick solution could be to install a second parallel port. Though they are cheap, they normally meet the technical requirements by a wide margin.

   There are some additional, even more rare problems that may appear, e.g., power saving printer ports, incompatible printer port BIOS settings.

4) **Try another machine.** For any version of Windows it may be worthwhile to try installing on another machine to verify that the dongle itself is working properly.

Downloading a program File to the PHL-1600

The serial (RS232) communication parameters of the PC must be set to match the communication parameters of the PHL-1600 in order for the data to be transferred successfully. If you do not know the settings of the PHL-1600 or you need to set them, refer to Section 1 for detailed instructions.

Using the MODE command while in the application development environment directory on the PC, set the desired RS232 port to the PHL-1600 parameters. Example commands are shown below:

```
C:\icc\appl>mode com1:9600,n,8,1
```

This sets the parameters to 9600 baud, no parity, 8 data bits, 1 stop bit.

Connect the RS232 port on the terminal's bottom to the host computer via the RS232 accessory cable. Prepare the terminal for receiving data through the RS232 port. This is accomplished by going to the Setup utilities, COM Setting screen. Refer to the "Users Guide" Section for detailed information.

Copy the HEX file to the terminal using the standard DOS copy command, e.g.,

```
copy Example.hex com1
```

The multicolored LED on the terminal will flash to indicate a data transfer is in process. When completed, the LED will glow a steady green. Depress the F1 key as indicated. This completes the installation process. The PHL-1600 should be turned OFF, then back to ON to start the Example application program.

The Example program differs from the demo program pre-installed on the PHL-1600 only in the title of the menu which appears on the start-up screen. By virtue of the source code change made after installing the Application Development Environment, the menu will be titled "DEMO MENU" instead of "MAIN MENU," indicating a successful download.

PHL-1600 Library Functions

In many instances the user may have source code developed for another terminal and desires to convert it for use on the PHL-1600. This section will discuss the procedure to be followed, relying on the developer's prior knowledge of C programming.

Most of the code generated for the original terminal should be directly re-compilable into PHL-1600 code. However, any code related to the hardware functions (i.e., keyboard definitions, screen maps, communication protocols, etc.) will generally have to be re-coded for operation with the PHL-1600 equivalents. To assist in this effort, **Appendix 1 contains a complete listing of all library functions available to the software developer.**

The developer should examine the original source code and identify all calls to library functions for the original terminal. Each one should be reviewed for compatibility with the equivalent function in the PHL-1600. Noting the parameters required, replace the original library call with the PHL-1600 equivalent. REMARKS: screen maps will probably need a new layout in entirety, so the available screen size and font maps in the PHL-1600 should be examined prior to changing the library calls related to the screen

Appendix 1

Library Function Listing

The following pages contain the PHL-1600 library functions available to the software developer. They are grouped by function. The table immediately following this page provides an overview of the libraries. The actual listings follow in the same order as the table in the pages immediately following the table.

Key Library

keyin	Get key (do not wait for key depression)
keyhit	Check key depression
keytouch	Check the key under depression
kbclr	Key buffer clear
kclk	Set click sound
keygetc	Get key (wait for key depression)

Display Library

dispmode	Set display font mode
dispc	Display one character
disps	Display character string
displine	Display grid or box
dispdot	Display dot image data
dispclr	Clear mark
dispcont	Adjust contrast
dispcur	Cursor ON/OFF
dispcurp	Move cursor
dispcurget	Get cursor coordinates

Serial Com Library

comopen	Open com port
comclose	Close com port
comread	Receive one byte data
comwrite	Send one byte data
comloc	Check number of receive data
comstatus	Receive status check
comrts	Control RTS (Request To Send)
comdtr	Control DTR (Data Terminal Ready)
comclr	Clear receive buffer

Power Control Library

poweroff	Power OFF
dipower	Disable Power OFF
resetoff	Delay Power OFF
stsbattery	Get battery status

Bar Code Reading Library

bcropen	Enable bar code reading
bcrclose	Disable bar code reading
bcrmodeset	Set conditions of bar code reading
bcrmodeget	Get conditions of bar code reading
bcrgets	Laser scanning is enabled

File Library

open	Open file
close	Close file
fwrite	Write file
fread	Read file
lseek	Move pointer
tell	Get pointer position
eof	Check if file end
remove	Delete file
freesize	Get size of unused area
fsize	Get file size
rename	Change name of a file

RTC Library

dateset	Set date
dateget	Get date
timeset	Set time
timeget	Get time

Timer Library

wait_time	Wait
wait_time2	Timer start and detect the time-up

Buzzer Library

buzzer	Buzzer and light LED
buzzer_stop	Stop buzzer

Other Library

ledon	Light LED
back_light	Set backlight ON/OFF
tmid_get	Get terminal ID

Abbreviation List

I	int
UI	unsigned int
C	char
UC	unsigned char
L	long

They are declared in the header file of "LIBEXT.H"

Key Library

keyin

Description Get a key from the keypad on the PHL terminal.

UI keyin (void);

Remarks Get key input from the keyboard. It does not wait for a key input. Under shift keys, only F1 to F8 keys are valid. Make use of keygetc for input of alphabets. Auto-power OFF is delayed if there is a key input.

Return Value Lower byte:Key data (ASCII).
Key input codes are shown below:

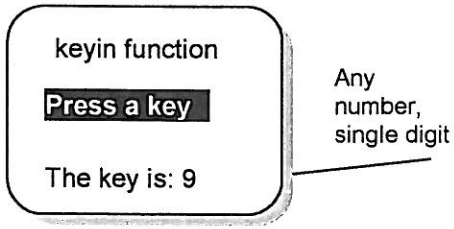
Key	Non-shift	Shift
Trigger	E1H	E1H
Q1	E2H	E2H
Q2	E3H	E3H
BS	O8H	O8H
CLR	18H	18H
0/$/+	30H	30H
1/STU	31H	31H
2/VWX	32H	32H
3/YZ_	33H	33H
4/JKL	34H	34H
5/MNO	35H	35H
6/PQR	36H	36H
7/ABC	37H	37H

Key	Non-shift	Shift
8/DEF	38H	38H
9/GHI	39H	39H
ENT	0DH	0DH
./%:*	2EH	2EH
F1/-	F1H	2DH
F2/DEL	F2H	7FH
F3/SP	F3H	20H
F4/BL	F4H	1BH
F5/←	F5H	1DH
F6/→	F6H	1CH
F7/△	F7H	1EH
F8/▽	F8H	1FH

Example

```
/*********************************************
 Library function:  keyin

 Description:  This program loops until the
 user presses a key.  The key is displayed on
 the screen.
*********************************************/
#include "libext.h"

void main(void)
{
   UC key;

   dispcurp(0,1);
   disps(0,0," keyin function ");
   dispcurp(0,3);
   disps(0,1," Press  a  key  ");

   while(1)
   {
       if(!keyhit()) /* Waits for a key depression */
       continue;
       key = keyin();         /* Gets the key */
       dispcurp(0,7);
       disps(0,0,"The key is:");

       /* Display the key pressed */
       dispcurp(13,7);
       dispc(0,0,key);
   }
}
```

Output



keyhit

Description Checks for a key depression on the PHL terminal.

UI keyhit (void);

Remarks Check if a key is depressed.

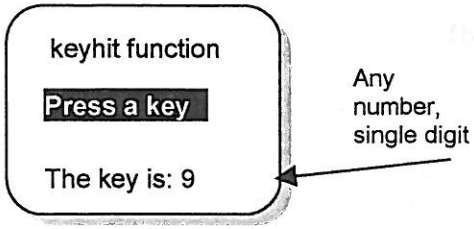
Return Value 0: No key depression
Non 0: Number of data in key buffer

Example

```
/*****************************************
 Library Function:  keyhit

 Description: This program loops until the
 user presses a key.  The key is displayed on
 the screen.
*****************************************/
#include "libext.h"

void main(void)
{
    UC key;

    dispcurp(0,1);
    disps(0,0,"keyhit  function");
    dispcurp(0,3);
    disps(0,1," Press  a  key  ");

    while(1)
    {
       if(!keyhit())      /* Waits for a key depression */
       continue;
       key = keyin();      /* Gets the key */
       dispcurp(0,7);
       disps(0,0,"The key is:");

       /* Display the key pressed */
```

```
            dispcurp(13,7);
            dispc(0,0,key);
        }
    }
```

Output



keytouch

Description Check a key under depression on the PHL terminal.

UL keytouch (void);

Remarks Check if a key is touched (pressed) on the keypad.

Return Value Corresponding bit is ON. If OFF, there is no key under depression.

Bit	Key
31	F5
28	1
25	SFT
22	F2
19	5
16	---
13	CR
10	9
7	F8
4	---
1	Q1

Bit	Key
30	F1
27	4
24	---
21	.
18	8
15	F7
12	3
9	Clr
6	F4
3	---
0	trg

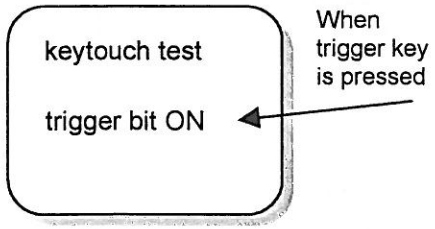
Bit	Key
29	0
26	7
23	F6
20	2
17	BS
14	F3
11	6
8	---
5	---
2	Q2

Example

```
/*********************************************
 Library function:  keytouch

 Description:  This program checks for the key
 under depression.  Display's a message when the
 trigger key(0 Bit) is pressed.  Otherwise it
 will display a message for the other key.
 *********************************************/
#include "libext.h"

void main(void)
{
   UC key;
```

```
        dispcurp(0,1);
        disps(0,0," keytouch  test ");

        while(1)
        {
           if(!keyhit())  /* Waits for a key depression */
           continue;

           key = keytouch();     /* Gets the key */
           dispcurp(0,5);
           if (key==0x0001)      /* Checks the trigger bit */
           {
              disps(0,0,"trigger bit ON");
              break;
           }
           else
           {
              disps(0,0,"Other bit ON");
              break;
           }
        }
     }
```

Output



kbclr

Description Clears the key buffer.

void kbclr (void);

Remarks Key buffer is cleared.

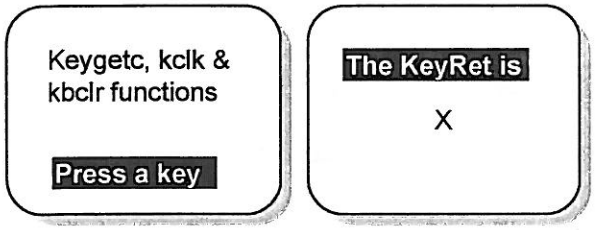
Return Value None.

Example

```
/*********************************************************
 Library Functions:  keygetc, kclk, kbclr

 Description: This program uses LCD_KEY structure defined
 in the libext.h header file.  The kclk function sets
 the click sound.  The keygetc function waits for a key
 inside the function and saves the key entered in the
 buffer for later use.  On completion of that loop kbclr
 clears the key board buffer.
*********************************************************/
#include "libext.h"

void main(void)
{

   UC    KeyRet;              /* Key return char */
   struct    LCD_KEY key;     /* Structure definition */

   key.echo   = 1;             /* initial keygetc() parameters */
   key.fnt    = 0;
   key.page   = 5;
   key.col    = 1;
   key.mincol = 1;
   key.maxcol = 14;

   kclk(1);              /* Set click sound */

   dispcurp(0,1);
```

```
    disps(0,0,"Keygetc, kclk & ");
    dispcurp(0,3);
    disps(0,0," kbclr functions");
    dispcurp(0,7);
    disps(0,1," Press  a   key ");

  while(1)
  {
     KeyRet = (UC)keygetc(&key);
      /* Get a key, it waits for a key inside function */

     if (( KeyRet >= '0' ) && ( KeyRet <= '9' )
            || ( KeyRet >= 'A' )&& ( KeyRet <= 'Z' )
            || ( KeyRet == ' ' ))
     {
         dispclr();
         dispcurp(0,1);
         disps(0,1," The  KeyRet is ");
         dispcurp(7,5);
         wait_time(40);
         dispc(0,0,KeyRet);
         kbclr();              /* Clears the key buffer */
         continue;
     }
  }
}
```

Output



kclk

Description Set key click sound for the keypad on the PHL terminal.

void kclk (UI switch);

UI switch: Switch setting key click
1: Set click sound
0: Release click sound

Remarks Set/release key click sound. If the key click sound is set, a click is heard whenever a key is pressed.

Return Value None.

Example See "kbclr"

Output See "kbclr"

keygetc

Description Get one character of a key depression.

UI keygetc (struct LCD_KEY *lcdkey);

```
Struct  LCD_KEY {
        UC    echo;        /* echo yes/no  */
        UC    fnt;         /* display font  */
        UC    mincol;      /* starting row of display*/
        UC    maxcol;      /* ending row of display  */
        UC    page;
        UC    col;
        UC    max;
};
```

UC echo echo yes/no (1:yes 0:no)
UC fnt Set display font (0:one half size/1:one quarter size)
UC mincol Display start column (Start column of echo back)
UC maxcol Display end column (End column of echo back)

When echo is disabled, all other settings are ignored.

The display cannot be done over two rows on echo back.

Remarks Get a key from keyboard input. It waits for a key input.
Input of alphabet is allowed.
Alphabet input on shifted 0-9 keys are input by depressing ENT key or another key.
Shifted key is toggled upon depression.
Auto-power OFF is delayed on key input.

Return Value Lower byte: key data (ASCII).
Key input codes are shown below.

KEY INPUT CODES

Key	Non-shift	Shift	Key	Non-shift	Shift
Trigger	E1H	E1H	8/DEF	38H	38H,44H,45H,46H
Q1	E2H	E2H	9/GHI	39H	39H,47H,48H,49H
Q2	E3H	E3H	ENT	0DH	0DH
BS	O8H	O8H	./%:*	2EH	2EH,25H,3AH,2AH
CLR	18H	18H	F1/-	F1H	2DH
0/$/+	30H	30H,24H,2FH,2BH	F2/DEL	F2H	7FH
1/STU	31H	31H,53H,54H,55H	F3/SP	F3H	20H
2/VWX	32H	32H,56H,57H,58H	F4/BL	F4H	1BH
3/YZ_	33H	33H,59H,5AH,5CH	F5/←	F5H	1DH
4/JKL	34H	34H,4AH,4BH,4CH	F6/→	F6H	1CH
5/MNO	35H	35H,4DH,4EH,4FH	F7/△	F7H	1EH
6/PQR	36H	36H, 50H,51H,52H	F8/▽	F8H	1FH
7/ABC	37H	37H,41H,42H,43H			

Example See "kbclr"

Output See "kbclr"

Display Library

dispmode

Description Set display font mode.

void dispmode (UI mode);

UI mode :
0:12 dot font mode
1:16 dot font mode

Remarks Set display font mode. There are two font modes, 12 or 16 dots. The mode is set by parameter. Once the mode is set, cursor position is set to 0,0 and display is cleared.

Display font/12 dot mode (default)
1) Full 12x12 dots (4 rows x 8 columns)
2) ½ Alpha-Numeric-Kana 6x12 dots (4 rows x 16 columns)
3) ¼ ANK 6x6 dots (8 Rows x 16 columns)

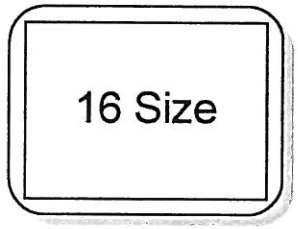
Display font/16 dot mode
1) Full 16x16 dots (3 rows x 6 columns)
2) ½ ANK 8x16 dots (3 rows x 12 columns)
3) ¼ ANK 8x8 dots (6 Rows x 12 columns)

Return Value None.

Example

```
/***********************************************
 Library Function:  dispmode

 Description: This program sets the display font
 mode to 16 size.
***********************************************/
#include "libext.h"

void main(void)
{
    /* Set the display font mode */
    dispmode(1);
    displine(3,6,93,45);    /* Grid co-ordinates */
    dispcurp(2,3);
    disps(0,0,"16 Size");
}
```

Output



dispc

Description Display one character on the screen.

void dispc (ankmode, UI attr, UI C);

UI ankmode : ANK display character
0: ½ size (default)
1: ¼ size
UI attr: Display property
0: default
1: Reverse
UI C :Display character

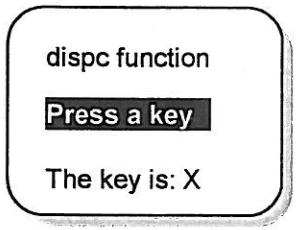
Remarks Display one character at the cursor position on the current font mode. Cursor forwards after display.
BS (08H) / CR (0DH) performs respectively;
BS = Move cursor backward by one character.
CR = Move cursor to the start of the next line.

Return Value None.

Example

```
/*****************************************
 Library Function:  dispc

 Description: This program shows how to make
 use of dispc function to display a single
 character.
*****************************************/
#include "libext.h"

void main(void)
{
   UC key;

   dispcurp(0,1);
   disps(0,0," dispc function ");
   dispcurp(0,4);
   disps(0,1," Press  a  key  ");

   while(1)
   {
      if(!keyhit())
      continue;
      key = keyin();
      dispcurp(0,7);
      disps(0,0,"The char is:");
      dispcurp(13,7);
      dispc(0,0,key);     /* Display only one character */
   }
}
```

Output



disps

Description Display character string on the screen

void disps (UI ankmode, UI attr, char *str);

UI ankmode : ANK display character
0: ½ size (default)
1: ¼ size
UI attr: Display property
0: default
1: Reverse
UI C :Display character string

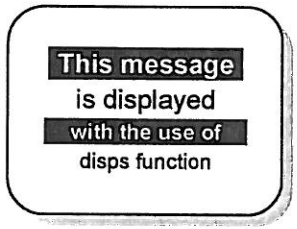
Remarks Display character string at the cursor position on the current font mode. The Cursor forwards.
BS (08H) / CR (0DH) performs respectively;
BS = Move cursor backward by one character.
CR= Move cursor to the start of the next line.

Return Value None.

Example

```
/***********************************************
 Library Function:  disps

 Description: This program displays the string
 with the use of disps function.  Alternatively
 it displays the string with the reverse and the
 default properties.
***********************************************/
#include "libext.h"

void main(void)
{
    /* Display with one half size and reverse mode */
    dispcurp(0,1);
    disps(0,1," This  message  ");

    /* Display with one half size and default mode */
    dispcurp(0,3);
    disps(0,0,"  is displayed  ");

     /* Display with one quarter size and reverse mode */
    dispcurp(0,5);
    disps(1,1,"with  the use of");

     /* Display with one quarter size and default mode */
    dispcurp(0,7);
    disps(1,0," disps function.");
}
```

Output



displine

Description Display grids and boxes on the screen.

void displine (UIX1, UIY1, UIX2, UIY2);

UIX1, Y1 :starting coordinate
UIX2, Y2 :ending coordinate

Remarks Display grids or boxes with starting and ending X and Y coordinates on the display screen of the PHL terminal.
Range: X1, X2 1-96 / Y1, Y2 1-48
starting coordinate (UIX1, Y1)
ending coordinate (UIX2, Y2)

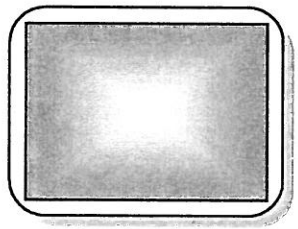
Return Value None.

Example

```
/*****************************************************
 Library Function:  displine

 Description: This function displays a grid box with
 the X and Y co-ordinates which keeps on changing in
 every loop.  You will see a black grid which is
 moving towards the center of the screen.  At the end
 of the loop you will see a screen which is totally
 blacked out due to a large number of grid box's.
*****************************************************/
#include "libext.h"

void main(void)
{
   int x1,x2,y1,y2,loop;
   x1 = 1;
   x2 = 96;
   y1 = 1;
   y2 = 48;

   while(1)
   {
```

```
        for(loop=0;loop<24;loop++)
        {
           /* Grid box co-ordinates */
           displine(x1,y1,x2,y2);
           wait_time(10);
           x1++;
           x2--;
           y1++;
           y2--;
         }
      }
   }
```

Output



dispdot

Description Display dot image

void dispdot (UI start, UI count, char *buffer);

UI start :Display start position
UI count :Display byte number
char *buffer :Display dot image

Remarks By setting the start position and the number of the byte, dot image is displayed.
Range start : 0-575
Count: 1-576

Return Value None.

dispclr

Description Display clear

void dispclr (void);

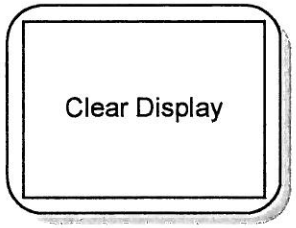
Remarks Clear the display of the PHL screen. It will erase all the existing data on the screen.

Return Value None.

Example

```
/*******************************************
 Library Function:  dispclr

 Description: The program loops by displaying
 the message and then it clears the screen
 after some wait time.
*******************************************/
#include "libext.h"

void main(void)
{
   while(1)
   {
      displine(3,6,93,45);
      dispcurp(2,4);
      disps(0,0,"Clear Display");
      wait_time(200);
      dispclr(); /* Clear the display screen */
      wait_time(200);
   }
}
```

Output



Note: The screen will change continuously, first by displaying the message and then clearing it.

dispcur

Description ON or OFF cursor specification.

void dispcur (UI type);

UI type: 0: No display
1: Under bar cursor

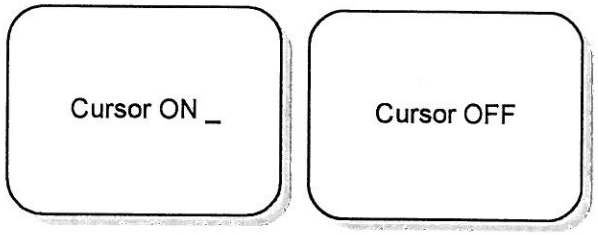
Remarks Put specified cursor ON or OFF. An Underscore bar will appear on the screen at the current cursor position when the cursor is ON. Cursor does not blink.

Return Value None.

Example

```
/***********************************************
 Library Function: dispcur

 Description: This program displays the cursor
 at the current position and then it turns off
 the display of the cursor.  The position of the
 cursor will remain the same, its just that the
 under score bar is not displayed on the screen.
***********************************************/
#include "libext.h"

void main(void)
{
   while(1)
   {
      dispcur(1);            /* Show the cursor */
      dispcurp(2,3);
      disps(0,0,"Cursor ON  ");
      wait_time(200);
      dispcur(0);            /* Hide the cursor */
      dispcurp(2,3);
      disps(0,0,"Cursor OFF");
      wait_time(200);
   }
}
```

Output



Note: These two screens will be displayed alternately.

dispcurp

Description Move cursor to the specified position.

void dispcurp (UI column, UI line);

UI Column :Set Column
12 dot mode 0-15
16 dot mode 0-11

UI line :Set row
12 dot mode 0-7
16 dot mode 0-5

Remarks Move cursor to the specified position.
Cursor is positioned under the displaying character.

¼ Size Full Size

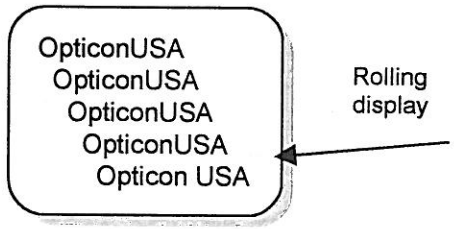
0th row 1st row

Return Value None.

Example

```
/************************************************
 Library function:  dispcurp

 Description: In this program the position of the
 cursor is constantly changing in every loop.  On
 the display screen you will see a rolling display
 of "OpticonUSA".
 ************************************************/
#include "libext.h"

void main(void)
{
   int i;

   while(1)
   {
      for(i=0;i<7;i++)
      {
          dispcurp(i,i+1);   /* Cursor position */
          disps(0,0,"OpticonUSA");
          wait_time(30);
          dispclr();
      }
   }
}
```

Output



dispcurget

Description Get current cursor position.

UI dispcurget (void);

Remarks Get the position of the cursor where ever it is currently pointing at or was placed at that position due to the last operation.

Return Value Lower 8 bits X coordinate
12 dot mode 0-15
16 dot mode 0-11

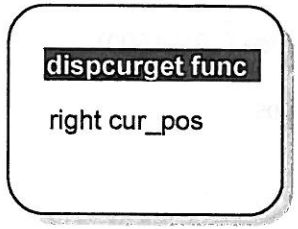
Upper 8 bits Y coordinate
12 dot mode 0-7
16 dot mode 0-5

Example

```
/************************************************
  Library Function:  dispcurget

  Description: This program checks for the current
  cursor position.
************************************************/
#include "libext.h"

void main(void)
{
   dispcurp(0,1);
   disps(0,1," dispcurget func");

   /* Move the cursor to a new position */
   dispcurp(2,3);
   wait_time(200);

  /* Check if the cursor has really moved by comparing
     the lower 8 bits for X and the upper 8 bits for Y */

   if(dispcurget()==0x0302)
```

```
    {
       dispcurp(0,5);
       disps(0,0,"right cur_pos"); /* No error */
    }
    else
    {
       dispcurp(0,5);
       disps(0,0,"not right_pos");  /* Error message */
    }
}
```

Output



Serial Com Library

comopen

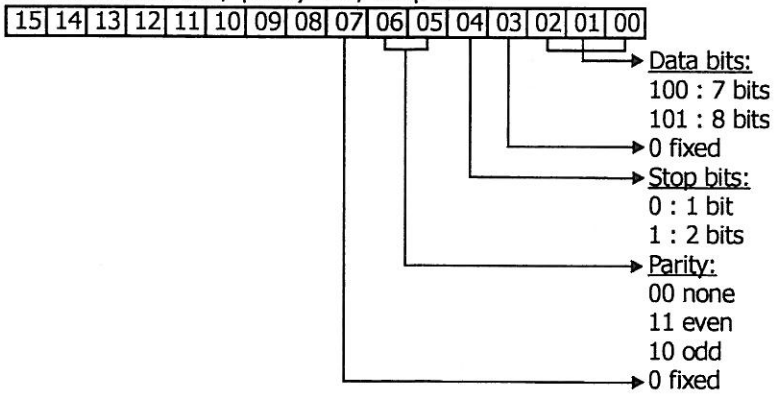
Description Open the com port.

I comopen (UI com, UI rate, UI mode, UI flow);

UI com: com port0: RS232C 1: Irda
2: Optical interface (1RU-1600)

UI rate: baud rate 1: 19200 bps
2: 9600 bps
3: 4800 bps
4: 2400 bps
5: 1200 bps

UI mode: data bit / parity bit / stop bit



UI flow: flow control 0: no control
1: RS.CS control

Remarks Open the communication port on the defined conditions.
Put DTR ON when opened on RS-232C.
Put RTS ON when opened on RS-232C, and controlled by RS-CS.

Return Value 0: No error
-1: Parameter error
-2: Double open
-3: IRU open error

☞ **CAUTION**: Optical interface (IRU-1600) can be opened when the unit is on the cradle. IRU open error occurs (-3) when opened without cradle.

Example

```
/***********************************************
 Library Function:  comopen

 Description: This program shows how do use the
 communication parameters to open a com port.
***********************************************/
#include "libext.h"

/* Communication definition */
#define COM_232C  0       /* RS 232C port */
#define COM_B96   2       /* Baud rate */
#define COM_PNON  0x0000  /* Parity bit */
#define COM_D8    0x0005  /* Data bit */
#define COM_S1    0x0000  /* Stop bit */
#define COM_FNON  0       /* Flow control */

void main(void)
{
   dispcurp(0,1);
   disps(0,0,"Opening RS-232C");
   dispcurp(0,4);
   wait_time(200);

 /* Open the RS-232C port with 9600,N,8,1 parameters */

if((comopen(COM_232C,COM_B96,COM_PNON|COM_D8|COM_S1,COM_FNON))!=0)
   {
      disps(0,0,"Port open error");
   }
   else
```

```
    {
        disps(0,0,"Port open is OK");
    }
}
```

Output

comclose

Description Close the com port.

I comclose (UI com);

UI com : com port

Remarks Close the specified communication port.
Put RTS, DTR off when COM 0 is specified.

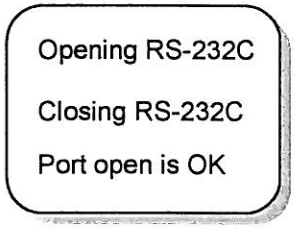
Return Value 0: No error
-1: Parameter error
-2: Not opened

Example

```
/*********************************************************
  Library Function:  comclose

  Description: This program opens the com port and then
  after some wait time it closes it.  On closing the port
  you will see a message for the operation of comclose.
 *********************************************************/
#include "libext.h"

#define COM_232C  0       /* RS 232C port */
#define COM_B96   2       /* Baud rate */
#define COM_PNON  0x0000 /* Parity bit */
#define COM_D8    0x0005 /* Data bit */
#define COM_S1    0x0000 /* Stop bit */
#define COM_FNON  0       /* Flow control */

void main(void)
{
   dispcurp(0,1);
   disps(0,0,"Opening RS-232C");
   comopen(COM_232C, COM_B96,COM_PNON|COM_D8|COM_S1,COM_FNON);
   wait_time(200);
   dispcurp(0,4);
```

```
    disps(0,0,"Closing RS-232C");
    wait_time(200);
    dispcurp(0,7);
    if((comclose(COM_232C))!=0)        /* Close the port */
    {
       disps(0,0,"Port close error");
    }
    else
    {
       disps(0,0,"Port close is OK");
    }
}
```

Output



comread

Description Receive data (1 character).

I comread (UI com)

UI com : com port

Remarks Receive one character from the specified COM port.

Return Value Upper bytes receive status
07
06 - parity error 1 : error
05 - framing error 1 : error
04 - overrun error 1 : error
03 - receive buffer overflow
02
01
00

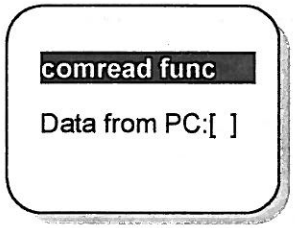
Lower bytes receive status
0 : No error
-1 : Parameter error
-2 : Not opened
-3 : No receive data

Example

```
/************************************************************
 Library Functions:  comread

 Description: This program reads a character send from the PC
 (key board) through the hyperterminal via the com port.

 Caution: You need to setup the HyperTerminal with proper
 communication parameters in order to make this program work.
************************************************************/
#include "libext.h"

/* Communication definition */
#define  COM_232C 0              /* RS-232C port */
```

```
#define  COM_B96  2              /* Baud rate */
#define  COM_D8   0x0005         /* Data bit */
#define  COM_PNON  0x0000        /* Parity bit */
#define  COM_S1    0x0000        /* Stop bit */
#define  COM_FNON  0             /* Flow control */

void main(void)
{
   UI    RcData;

   dispcurp(0,1);
   disps(0,1," comread  func  ");
   dispcurp(0,4);
   disps(0,0,"Data from PC:[ ]");      /* Char from PC */

   comopen(COM_232C,COM_B96,COM_D8|COM_PNON|COM_S1,COM_FNON);
   while(1)
   {
      /* Number of bytes received in data buffer */
      RcData = comread(COM_232C);
      if (!(RcData & 0x7100))     /* Get a byte If no error */
      {
         dispcurp(14,4);          /* Display received data */
         dispc(0,0,RcData);
      }
   }
}
```

Output



comwrite

Description Send data through the com port.

I comwrite (UI com,char c);

UI com : com port
char c : send character

Remarks Send one character from the specified COM port.

Return Value 0: No error
-1: Parameter error
-2: Not specified

Example

```
/*****************************************************
 Library Function:  comwrite

 Description: This program writes a character to the
 HyperTerminal on the PC, send by the keypad on the
 PHL terminal.

 Caution: You need to setup the HyperTerminal with
 proper communication parameters in order to make
 this program work.
*****************************************************/
#include "libext.h"

/* Communication definition */
#define COM_232C  0          /* RS 232C port */
#define COM_B96   2          /* Baud rate */
#define COM_PNON  0x0000     /* Parity bit */
#define COM_D8    0x0005     /* Data bit */
#define COM_S1    0x0000     /* Stop bit */
#define COM_FNON  0          /* Flow control */

void main(void)
{
```

```
    UC    key;

    kclk(1);
    dispcurp(0,1);
    disps(0,1," comwrite  func ");
    dispcurp(0,4);
    disps(0,0," Send to PC:[ ]");          /* Char to PC */

    comopen(COM_232C,COM_B96,COM_D8|COM_PNON|COM_S1,COM_FNON);
    while(1)
    {
       if(!keyhit())
          continue;
       key = keyin();

       /* If it is numeric key, display on screen & send to PC */
       if ((key >='0')&&(key <= '9'))
       {
          dispcurp(13,4);
          dispc(0,0,key);         /* Display the key */
          comwrite(COM_232C, key);
          continue;
       }
    }
}
```

Output

comloc

Description Count the received data through the com port.

I comloc (UI com);

UI com : com port

Remarks Count bytes received from the specified com port.

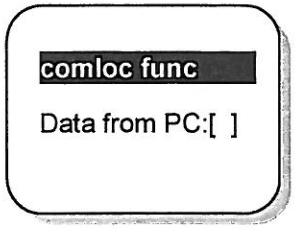
Return Value Number of received bytes
-1: Parameter error
-2: Not opened

Example

```
/*******************************************************
 Library Function:  comloc

 Description: This program opens the com port and reads
 data from the com port.  If the count of receive data
 is successful then it will display the character or
 else it will show an error message.

 Caution: You need to setup the HyperTerminal with
 proper communication parameters in order to make
 this program work.
*******************************************************/
#include "libext.h"

/* Communication definition */
#define COM_232C  0      /* RS 232C port */
#define COM_B96   2      /* Baud rate */
#define COM_PNON  0x0000 /* Parity bit */
#define COM_D8    0x0005 /* Data bit */
#define COM_S1    0x0000 /* Stop bit */
#define COM_FNON  0      /* Flow control */

void main(void)
{
```

```
    UI    RcData;

    dispcurp(0,1);
    disps(0,1," comloc   func  ");
    dispcurp(0,4);
    disps(0,0,"Data from PC:[ ]");

    comopen(COM_232C,COM_B96,COM_D8|COM_PNON|COM_S1,COM_FNON);
    while(1)
    {
       RcData = comread(COM_232C);
      /* Get a byte If no error in receive data */
       if (!(RcData & 0x7100)||comloc(COM_232C)!=0)
       {
          dispcurp(14,4);
          dispc(0,0,RcData);
       }
       else
       {
          disps(0,0,"Error");
       }
    }
}
```

Output



comstatus

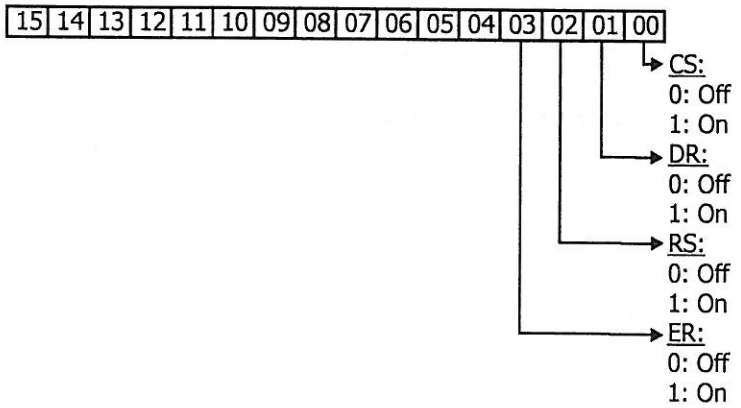
Description Signal check of receive status of the com port.

UI comstatus (UI com);

UI com : com port

Remarks Check status of the signals on the specified com port.

Return Value



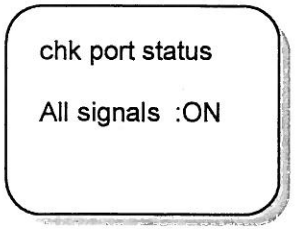
-1: Parameter error -2: Not opened

Example

```
/*************************************************
 Library Function:  comstatus

 Description: This program opens the com port and
 then it checks if all the signals(ER,RS,DR,CS) are
 ON at the same time or not.  It then displays the
 message accordingly.
*************************************************/
#include "libext.h"

/* Communication definition */
#define COM_232C  0       /* RS 232C port */
#define COM_B96   2       /* Baud rate */
#define COM_PNON  0x0000 /* Parity bit */
```

```
#define COM_D8    0x0005 /* Data bit */
#define COM_S1    0x0000 /* Stop bit */
#define COM_FNON  0      /* Flow control */

void main(void)
{
   dispcurp(0,1);
   disps(0,0,"chk port status");
   wait_time(200);
   comopen(COM_232C,COM_B96,COM_D8|COM_PNON|COM_S1,COM_FNON);
   dispcurp(0,5);

   /* Signal check of the receive status */
   if(comstatus(COM_232C)==0x000F)
   {
      disps(0,0,"All signals :ON");
   }
   else
   {
      disps(0,0,"All signals are");
      dispcurp(0,7);
      disps(0,0,"notON @ sametime");
   }
}
```

Output



comrts

Description Control RTS (Request to send) signal.

void comrts (UI com, UI switch);

UI com : com port
UI switch: 0: OFF 1: ON

Remarks Control RTS signal of the specified com port.

Return Value 0: No error
-1: Parameter error
-3 : Not opened

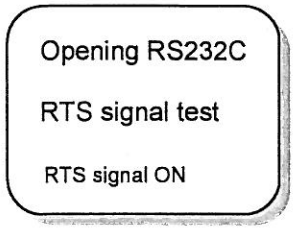
CAUTION: The function is only applicable to COMO (RS-232C). The function should not be used when RS-CS control is used.

Example

```
/*********************************************
 Library Functions:  comrts

 Description: This program shows how to control
 the RTS signal.  It displays an error message
 if the control of RTS signal fails.
*********************************************/
#include "libext.h"

/* Communication definition */
#define COM_232C  0          /* RS 232C port */
#define COM_B96   2          /* Baud rate */
#define COM_PNON  0x0000     /* Parity bit */
#define COM_D8    0x0005     /* Data bit */
#define COM_S1    0x0000     /* Stop bit */
#define COM_FNON  0          /* Flow control */

void main(void)
{
   dispcurp(0,1);
```

```
    disps(0,0,"Opening RS-232C");
    dispcurp(0,4);
    disps(0,0,"RTS signal test");


   while(1)
   {
      comopen(COM_232C,COM_B96,COM_PNON|COM_D8|COM_S1,COM_FNON);
      wait_time(200);
      dispcurp(1,7);

      /* Checks for the return value */
      if((comrts(COM_232C,1))==0)
      {
          disps(1,0,"RTS signal ON");
      }
      else
      {
          disps(1,0,"ERROR IN RTS");
      }
    }
}
```

Output



comdtr

Description Control DTR (Data Terminal Ready) signal.

void comdtr (UI com, UI switch);

UI com : com
UI switch : 0: OFF / 1: ON

Remarks Control DTR signal of the specified com port.

Return Value 0: No error
-1: Parameter error
-3 : Not opened

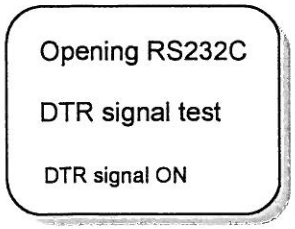
 **CAUTION**: The function is only applicable to COMO (RS-232C).

Example

```
/*********************************************
 Library Functions:  comdtr

 Description: This program shows how to control
 the DTR signal.  It displays an error message
 if the control of DTR signal fails.
*********************************************/
#include "libext.h"

/* Communication definition */
#define COM_232C  0          /* RS 232C port */
#define COM_B96   2          /* Baud rate */
#define COM_PNON  0x0000     /* Parity bit */
#define COM_D8    0x0005     /* Data bit */
#define COM_S1    0x0000     /* Stop bit */
#define COM_FNON  0          /* Flow control */

void main(void)
{
   dispcurp(0,1);
   disps(0,0,"Opening RS-232C");
```

```
    dispcurp(0,4);
    disps(0,0,"DTR signal test");



    while(1)
    {
       comopen(COM_232C,COM_B96,COM_PNON|COM_D8|COM_S1,COM_FNON);
       dispcurp(1,7);
       wait_time(200);

       /* Checks for the return value */
       if((comdtr(COM_232C,1))==0)
       {
          disps(1,0,"DTR signal ON");
       }
       else
       {
          disps(1,0,"ERROR in DTR");
       }
    }
}
```

Output



comclr

Description Clear the receive buffer of the com port.

void comclr (UI com);

UI com : com port

Remarks Clear the contents of the buffer of the specified com port.

Return Value 0: No error
-1: Parameter error
-3 : Not opened

Example

```
/*********************************************
 Library Function:  comclr

 Description: This program clears the receive
 buffer of the specified com port.  It displays
 an error message if the operation fails.
*********************************************/
#include "libext.h"

/* Communication definition */
#define COM_232C  0      /* RS 232C port */
#define COM_B96   2      /* Baud rate */
#define COM_PNON  0x0000 /* Parity bit */
#define COM_D8    0x0005 /* Data bit */
#define COM_S1    0x0000 /* Stop bit */
#define COM_FNON  0      /* Flow control */

void main(void)
{
   dispcurp(0,1);
   disps(0,0,"chk receive buff");
   wait_time(200);
   comopen(COM_232C,COM_B96,COM_D8|COM_PNON|COM_S1,COM_FNON);
   dispcurp(0,5);
```

```
    /* Clear the receive buffer and check the return value */
    if(comclr(COM_232C)==0)
    {
       disps(0,0,"buffer cleared");
    }
    else
    {
        disps(0,0,"recieve buff err");
    }
}
```

Output

Power Control Library

poweroff

Description Switch the Power OFF.

void poweroff (UI switch);

UI switch:
0: Power OFF with the specified boot up preparation
1: Power OFF with the initial boot up preparation.

Remarks Define power OFF with the specified boot up preparation.

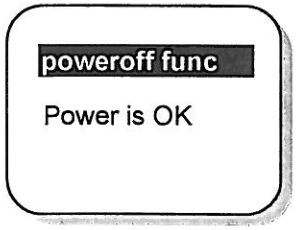
Return Value None.

Example

```
/*****************************************************
 Library Function:  poweroff

 Description: This program checks the status of the
 battery.  If the main and the sub batteries are both
 low, then it will switch the power OFF of the PHL.
*****************************************************/
#include "libext.h"

void main(void)
{
   int bat;

   dispcurp(0,1);
   disps(0,1," poweroff  func ");
   dispcurp(0,4);
   bat=stsbattery();  /* Get the battery status */

   if((bat==0x0003)||(bat==0x0103))
   {
      disps(0,0,"Low Power");
```

```
        dispcurp(0,6);
        disps(0,0,"Power shutdown.");
        wait_time(300);
        /* Power off with the initial bootup preparation */
        poweroff(1);
    }
    else
    {
        disps(0,0,"   Power is OK");
    }
}
```

Output



dipower

Description Power OFF

void poweroff (UI switch);

UI switch:
0: Power OFF with the specified boot up preparation
1: Power OFF with the initial boot up preparation.

Remarks Define power OFF with the specified boot up preparation.

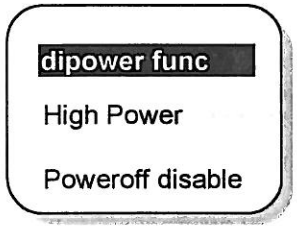
Return Value None.

Example

```
/*************************************************************
  Library Function:  dipower

  Description: This program checks the status of the
  battery.  If the batteries are high then it will disable
  the power switch.

  Caution: If the High Power message is displayed, then the
  only way to turn the power off is to open the battery case
  and take off the battery for a while and place it back again.
*************************************************************/
#include "libext.h"

void main(void)
{
   int bat;

   dispcurp(0,1);
   disps(0,1," dipower  func  ");
   dispcurp(0,4);
   bat=stsbattery();  /* Get the battery status */
   if((bat==0x0100)||(bat==0x0000))
   {
      disps(0,0,"High Power");
```

```
        dispcurp(0,7);
        disps(0,0,"Poweroff disable");
        /* Power off is disabled */
        dipower(1);
    }
    else if((bat==0x0003)||(bat==0x0103))
    {
        disps(0,0,"Low Power");
        dispcurp(0,7);
        disps(0,0,"Power off enable");
         /* Power off is enabled */
        dipower(0);
    }
}
```

Output



resetoff

Description Set delay time for auto-power OFF.

void resetoff (UI time);

UI time: duration (seconds)
10 seconds to 3600 seconds.

Remarks Set delay time of auto-power OFF. The setting becomes effective without reset. Auto-power OFF is disabled when zero (0).

Return Value None.

Example

```
/*******************************************
 Library Function:  resetoff

 Description: This program checks the status
 of the battery.  If the power is LOW, then
 it will turn off the PHL terminal in 10 sec.
*******************************************/
#include "libext.h"

void main(void)
{
   int bat;
   dispcurp(0,1);
   disps(0,1," resetoff  func ");
   dispcurp(0,4);
   bat=stsbattery();  /* Get the battery status */
   if((bat==0x0100)||(bat==0x0000))
   {
      disps(0,0,"High Power");
      dispcurp(0,7);
      disps(0,0,"NO resetoff ");
   }
   else
   {
```

```
        disps(0,0,"Low Power");
        dispcurp(0,7);
        disps(0,0,"resetoff in10sec");
        /* Power shuts off automatically in 10 sec */
        resetoff(10);
    }
}
```

Output

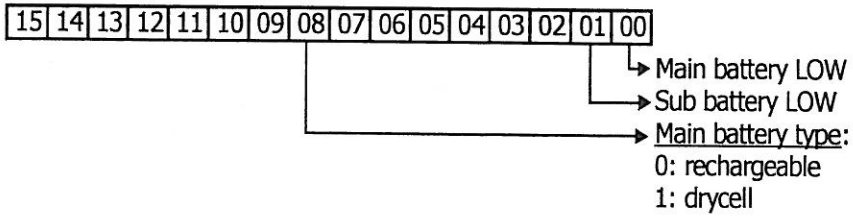
stsbattery

Description Get the battery status.

UI stsbattery (void);

Remarks Get the status of the Main battery and the Sub battery. It also finds out which type of battery is being used (Dry Cell or Rechargeable).

Return Value

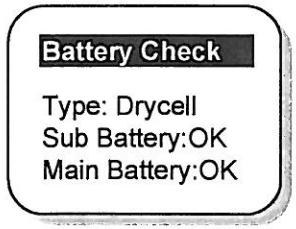


Example

```
/*************************************************
 Library Function:  stsbattery

 Description:  This program checks the detailed
 status of the Battery.  It checks for the type of
 battery used and then it checks the status of
 the main battery and the sub battery.
*************************************************/
#include "libext.h"

void main(void)
{
   int bat;
   dispcurp(0,1);
   disps(0,1," Battery  Check ");
   dispcurp(0,3);
   while(1)
   {
      bat=stsbattery();  /* Get the battery status */
      if((bat==0x0000)||(bat==0x0001)
```

```
            ||(bat==0x0002)||(bat==0x0003))
        {
            disps(0,0,"Type:Rechargable");
        }



        else
        {
            disps(0,0,"Type:Drycell     ");
        }
        dispcurp(0,5);
        if((bat==0x0000)||(bat==0x0001)||(bat==0x0100)||(bat==0x0101))
        {
            disps(0,0,"Sub Battery:OK   ");
        }
        else
        {
            disps(0,0,"Sub Battery:LOW  ");
        }
        dispcurp(0,7);
        if((bat==0x0000)||(bat==0x0002)||(bat==0x0100)||(bat==0x1002))
        {
            disps(0,0,"Main Battery:OK  ");
        }
        else
        {
            disps(0,0,"MainBattery:LOW  ");
        }
    }
}
```

Output



Bar Code Reading Library

bcropen

Description Enable bar code reading.

UI bcropen (void);

Remarks Enable bar code reading through the trigger key on the scanner.
Set to default.

Return Value None.

Example See "bcrgets"

Output See "bcrgets"

bcrclose

Description Disable bar code reading.

UI bcrclose (void);

Remarks Bar code reading is disabled.

Return Value None.

Example See "bcrgets"

Output See "bcrgets"

bcrmodeset

Description Set bar code reading parameters

UI bcrmodeset (BMD_TBL*bmdtable);

BMD_TBL *bmdtable : Area pointer to setting table.

Construction

```
typedef  struct  BMD_TBL{
   /* Bar code reading setting table */
   UI    setm1;   /* setting 1 available codes  */
   UI    setm2;   /* setting 2    */
   UI    setm3;   /* setting 3    */
   UI    setm4;   /* setting 4    */
   UI    setm5;   /* setting 5    */
   UI    keta[5];  /* reading bar code digits     */
   UI    buzzfc;   /* buzzer frequency on correct reading      */
   UI    buzztime;  /* buzzer duration on correct reading*/
   UI    timeout;  /* laser timeout     */
   UI    yobi[10]; /* reserved     */
};
```

Remarks Set bar code reading parameters.

Return Value 0: No error
1: Bar code reading is not opened

Example See "bcrgets"

Output See "bcrgets"

bcrmodeget

Description Get bar code reading parameters.

UI bcrmodeget (BMD_TBL*bmdtable);

BMD_TBL*bmdtable : Pointer to the parameter table.

Remarks Get parameter values of bar code reading from the different parameter tables.

Return Value 0: No error
1: bar code reading is not opened

Example See "bcrgets"

Output See "bcrgets"

Parameter Tables

1) Parameter 1 / UI setm1

15 – reserved
14 – reserved
13 – reserved
12 – reserved
11 – reserved
10 – reserved
09 – reserved

08 - MSI	0: disable	**1: enable**
07 - CODE-128	0: disable	**1: enable**
06 - CODE-93	0: disable	**1: enable**
05 - CODE-11	0: disable	**1: enable**
04 – 2OF5 interlvd	0: disable	**1: enable**
03 - 2OF5 standard	**0: disable**	1: enable
02 - WPC	**0: disable**	1: enable
01 - Codabar	**0: disable**	1: enable
00 - CODE39	**0: disable**	1: enable

* The default is **bold**

2) Parameter 2 / UI setm2

15 – reserved
14 – reserved
13 – reserved
12 – reserved
11 – reserved
10 – reserved
09 – reserved
08 – Reserved

07 - CODE-39 full ASCII	**0: disable**	1: enable
06 - Codabar	**0: 7 check**	1: MOD16
05 - WPC add on code	**0: disable**	1: enable
04 - Fixed digits	**0: disable**	1: enable
03 - Codabar/Code 39 ICG Spec.	**0: within 1 char.**	1: within 8 char
02 - Codabar start/stop up/low case	0: lower case	**1: upper case**
01 - Codabar start/stop digits	0: disable	**1: enable**
00 - CODE39 start/stop digits	**0: disable**	1: enable

* The default is **bold**

3) Parameter 3 / UI setm3

15 – reserved
14 – reserved
13 – reserved
12 – reserved

11 - CODE-128 check digit calculation	0: disable	**1: enable**
10 - MSI check digit calculation	**0: disable**	1: enable
09 - MSI check digit attached	**0: disable**	1: enable
08 - MSI check digit calculation method	**0: 1 check**	1: 2 check
07 - CODE-11 check digit attached	**0: disable**	1: enable
06 - CODE-11 check digit calculation	**0: 1 check**	1: 2 check
05 - 2OF5 check digit attached	**0: disable**	1: enable
04 - 2OF5 check digit calculation	**0: disable**	1: enable
03 - Codabar check digit attached	**0: disable**	1: enable
02 - Codabar check digit calculation	**0: disable**	1: enable
01 - CODE-39 check digit attached	**0: disable**	1: enable
00 - CODE-39 check digit calculation	**0: disable**	1: enable

* The default is **bold**

4) Parameter 4 / UI setm4

15 – reserved		
14 – reserved		
13 – reserved		
12 – reserved		
11 – reserved		
10 – reserved		
09, 08 - WPC CODE-39	00: 0	**01: 1**
Codabar allowed number	10 : 2	11 : 3
of noise bar in the margin		
07 – reserved		
06, 05 - UPC-E output format	00: 6 digits	**01: 7-1 digits**
	10: 7-2 digits	11: 8 digits
04, 03 - UPC-A output format	00: 10 digits	01: 11 digits
	10: 12 digits	**11: 13 digits**
02 - UPC-E output format	**0: 0 suppressed**	1: 0 insert
01 - Codabar ABC code	**0: disable**	1: enable
00 - WPC add on code forced read	**0: disable**	1: enable

* The default is **bold**

5) Reading Digits

UI keta [5]
from 1 to 42
It is set up to 5 codes.
Number of digits is not checked if set to zero (0).
The default value is zero (0)
*JAN . EAN . UPC codes are not considered.
*Start/Stop and C/D are not included.

6) Buzzer frequency on the correct reading

UI buzzfc
The default is 3200 (3.2kHz).

7) Buzzer duration on the correct reading

UI buzztime
10mS interval
The default is 5 (50mS).

8) Laser Timeout

UI timeout
Interval in a second.
The default is 5 (5 seconds).

9) Redundant Reading

quality of reading is assured by referring multiple reading.
UI equctb
The default is twice reading and comparing once.

bcrgets

Description Bar code reading through the scanner.

UI bcrgets(UI mode, UI * readcnt, UC * buf);

UI mode: Set mode of bar code reading
UI * readcnt: Reading digits
UC * buf: Bar code data buffer

Details of the mode
15 thru 04 – reserved

03 - Multiple/Single scan
0: Multiple 1: Single
Multiple scan mode reads the same bar code in succession. Single scan mode does not read the same bar code in succession.

02 - Indication of completion of reading
1: Buzzer and turn on LED

01 - Completion of reading 2
0: Ignore key depression
1: Sense key depression and stop

00 - Completion of reading 1
0: Read, Time-out, Trigger OFF (Scan while trigger is depressed)
1: Read or Time-out

Remarks Laser scanning is enabled.

Return Value 15 - reading completion
1: Complete
0: Incomplete
14 – reserved
13 - 1: Bar code reader is disabled
12 - 1: Error in laser
11 – reserved
10 - 1: Finish by time-out
09 - 1: Finish by key depression
08 - 1: Finish by trigger off
07 – reserved
06 - reserved
05 – reserved
04 – reserved

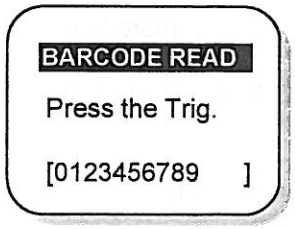
03	02	01	00	Read Code
0	0	0	1	Code 39
0	0	1	0	Codabar
0	0	1	1	WPC
0	1	0	0	2 of 5 Standard
0	1	0	1	2 of 5 Interleaved
0	1	1	0	Code 11
0	1	1	1	Code 93
1	0	0	0	Code 128
1	0	0	1	MSI

Example

```
/*******************************************************
 Library Function :  bcropen, bcrclose, bcrmodeget,
                     bcrmodeset, bcrgets.

 Description: This program shows the use of bar code
 functions.  Bar code reading is enabled and gets the
 parameters for barcode reading.  A specific table is
 set for bar code reading.  On pressing the trigger the
 barcode is scanned and displayed on the screen and then
 barcode reading is disabled.
*******************************************************/
#include "libext.h"

void main(void)
{
   UC    BarData[50];        /* Bar code data reading buffer */
   UI    BarLength, BarRet; /* Bar code data character number */
   BMD_TBL tbl;

   bcropen();
   bcrmodeget(&tbl);     /* Gets the barcode reading parameters */
   tbl.setm4 = 0x10; /* Enables specific table for code reading */
   bcrmodeset(&tbl);     /* Sets the barcode reading parameters */
   dispcurp(0,1);
   disps(0,1," BARCODE   READ ");
   dispcurp(0,3);
   disps(0,0,"Press the Trig.");

   while(1)
   {
      if(!keyhit())
      continue;

      /* Read the barcode */
      BarRet = bcrgets( 4, &BarLength, BarData );
      if( BarRet&0x8000 )    /* Successful read */
      {
         dispcurp(0,6);
```

```
            disps(0,0,"[              ]");
            dispcurp(1,6);
            if (BarLength >14)
            {                          /* Only first 14 char displayed */
                BarLength = 14;
             }
             BarData[BarLength] = '\0'; /* End the string    */
             disps(0,0,BarData);          /* Display the barcode */

             bcrclose();       /* Disable bar code reading */
        }
    }
}
```

Output



File Library

Power OFF should be disabled (dipower) during the execution of file library.

open

Description Opens a file.

I open (C far *fname, UI oflag);

Remarks A file specified by fname is opened on the definition of oflag. The following constant should accompany with oflag.

oflag can accompany one constant. If more than 2 values are set, they should be connected by using the OR operator. Among the O_RDONLY, O_WRONLY, O_RDWR, one value should be specified to 0flag.

Simultaneously, 16 files can be opened.
The files are registered up to 128.

Constant	Description
O_RDONLY	read only
O_WRONLY	write only
O_RDWR	read and write
O_APPEND	file pointer to the end of file at open
O_CREAT	create a file for writing, the existing file is ignored.

Return Value A file handle of an opened file is returned. When error occurs, - 1 is returned, and the following error messages are returned to the global variable:

EEXIST O_CREAT is specified while flag file exists.
EINVAL Invalid oflag is specified.
EMFILE There are too many open files.
ENOENT The specified file is not found.

Example

```
/***********************************************
 Library Function:  open

 Description: This program shows how to open and
 create a file for specific file operations.  It
 creates a file with "demo.dat" name for write
 only in the append mode.
***********************************************/
#include "libext.h"

void main(void)
{
   int handle;
   dispcurp(0,1);
   disps(0,1," FILE OPEN TEST ");
   wait_time(100);
   dispcurp(0,4);

   /* Open a file */
   if ((handle=(open("demo.dat",O_CREAT|O_WRONLY|O_APPEND))==-1))
   {
      disps(0,0,"File open error");
   }
   else              /* Error in opening the file */
   {
      disps(0,0,"File open OK");
   }
}
```

Output

close

Description Closes a file.

I close(I hdl);

Remarks A file specified by the hdl is closed.

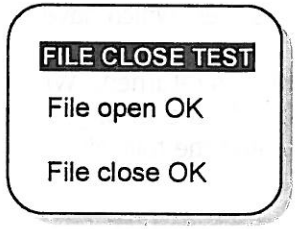
Return Value When a file is closed, zero (0) is returned. When an error occurs, -1 is returned, and EBADF is set to the global variable, Errno showing the file handle is invalid.

Example

```
/*****************************************************
 Library Function:  close

 Description: This program opens a file for specific
 file operations and then it closes the file with
 the handle.
*****************************************************/
#include "libext.h"

void main(void)
{
   int handle;
   dispcurp(0,1);
   disps(0,1," FILE CLOSE TEST");
   wait_time(100);
   dispcurp(0,4);

   if ((handle=(open("demo.dat",O_CREAT|O_WRONLY|O_APPEND))==-1))
   {
      disps(0,0,"File open error");
   }
   else
   {
      disps(0,0,"File open OK");
   }
   wait_time(100);
```

```
    dispcurp(0,7);
    if (close(handle)==0) /* Close the file with the handle */
    {
       disps(0,0,"File close OK");
    }
    else
    {
       disps(0,0,"File not closed");
    }
}
```

Output



fwrite

Description Write data into a file

I fwrite(I hdl,C far *buff, UI count);

Remarks Write counted bytes into a file connected with hdl from buff. Writing starts from the current file pointer. When a file is opened as append mode, the writing starts from the end of the file. File pointer proceeds by the number of bytes, which have been written.

Return Value The number of bytes actually written is returned. When the memory space is smaller than the buff, the writing is not done. When error occurs, -1 is returned, and the following value is set into the global value, erron.

EBADF File handle is invalid.
ENOSPC There is no memory space.

☞ **CAUTION**: Set a size less than 32K in the count.

Example

```
/***********************************************
 Library Function:  fwrite

 Description: This program opens a file for write
 operation.  It writes some data to the file and
 checks if the file write operation is OK or not.
***********************************************/
#include "libext.h"

void main(void)
{
   UC  FileBuffer[20]="OPTICON-USA";
   UI  FileLength;
   int handle, numwritten;

   dispclr();
   dispcurp(0,1);
```

```
    disps(0,1,"FILE WRITE TEST ");
    wait_time(100);
    dispcurp(0,5);

  /* Opens and creates demo.dat file for write in append mode */
  if (-1 !=(handle = open("demo.dat",O_CREAT|O_WRONLY|O_APPEND)))
  {
      /* Write to the file */
      numwritten = fwrite(handle,FileBuffer,FileLength);
      if(numwritten != -1)        /* Check the return value */
      {
          disps(0,0,"File write OK");
          close(handle);
      }
      else
      {
          disps(0,0,"File write error");
          close(handle);
      }
  }
  else
  {
       disps(0,0,"File open error");
  }
}
```

Output

fread

Description Data is read from a file.

I fread(I hdl,C far *buff,UI count);

Remarks A counted number of bytes of an hdl connected file is read out to buff.

Return Value The number of bytes is returned. When there is less bytes in a file than count, there may be less bytes read out than the one in count. When a returned value is zero(0), the readout has been done at the end of a file. When a returned value is - 1, EBADF is set to the global variable, Errno, showing that the file handle is invalid.

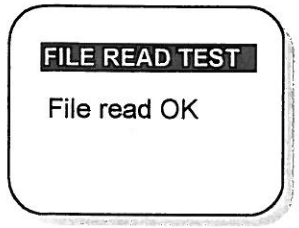
☞ **CAUTION**: Set value less than 32K (max size of int type) in count.

Example

```
/*********************************************
 Library Function:  fread

 Description: This program opens a file for read
 operation.  It reads some data from the file and
 checks if the file read operation is OK or not.
*********************************************/
#include "libext.h"

void main(void)
{
   UC  FileBuffer[50]="OPTICON-USA";
   UI  FileLength;
   int handle, numread;

   dispclr();
   dispcurp(0,1);
   disps(0,1,"FILE  READ  TEST");
   wait_time(100);
   dispcurp(0,5);

   if (-1 !=(handle = open("demo.dat",O_CREAT|O_RDWR|O_APPEND)))
```

```
{
      /* Read from the file */
   numread = fread(handle,FileBuffer,FileLength);
    if(numread != -1)          /* Check the return value */
    {
       disps(0,0," File read OK");
       close(handle);
     }
     else
     {
        disps(0,0,"File read error");
        close(handle);
      }
   }
   else
   {
      disps(0,0,"File open error");
   }
}
```

Output



lseek

Description Move file pointer to the specified location.

L lseek(I hdl,L offset, I origin);

Remarks Move file pointer of a specified file by offset bytes from the position specified by origin.

SEEK_SET Start of file
SEEK_CUR Current position of file pointer.
SEEK_END End of file

Return Value The position of file pointer in the number of bytes from the start of file.

EBADF Invalid file handle
EINVAL The value in origin is invalid.

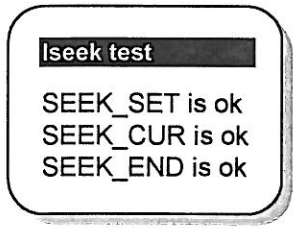
☞ **CAUTION**: Disable power OFF (dipower) to protect operation. File pointer cannot go beyond the end of file.

Example

```
/*************************************************************
 Library Function: lseek

 Description: This program shows how to move the file pointer
 to a specific position.  It opens a file for read only mode.
 It then moves the file pointer at the start of the file.  The
File pointer is moved a little bit and then that position is
checked.  Then it moves the file pointer to the end of the file.

 Caution: Create "demo.dat" file if its not there.
*************************************************************/
#include "libext.h"

void main(void)
{
   int handle;
   L  pos;
   UC SampleB[10];

   dispcurp(0,1);
   disps(0,1,"  lseek   test  ");
   dispcurp(1,3);
   if((handle=open("demo.dat",O_RDONLY))!=-1)
   {
     /* Move the file pointer to start of file */
      pos=lseek(handle,0,SEEK_SET);
      if(pos==-1)
      {
         disps(0,0,"SEEK_SET failed");
      }
      else if(pos==0)
      {
         disps(0,0,"SEEK_SET is ok");
      }
      /* Move the file pointer a little */
      fread(handle,SampleB,10);
      pos=lseek(handle,0,SEEK_CUR);
      dispcurp(1,5);
```

```
        if(pos==-1)
        {
           disps(0,0,"SEEK_CUR failed");
        }
        /* Check if it has moved the specified positions */
        else if(pos==10)
        {
           disps(0,0,"SEEK_CUR is ok");
        }
        /* Move the file pointer to the end of file */
        pos=lseek(handle,0,SEEK_END);
        dispcurp(1,7);
        if(pos==-1)
        {
           disps(0,0,"SEEK_END failed");
        }
        else
        {
           disps(0,0,"SEEK_END is ok");
        }
     }
     else
     {
        disps(0,0,"file open error");
     }
     close(handle);
}
```

Output



tell

Description Get position of file pointer.

L tell(I hdl);

Remarks Get a position of file pointer for a file specified by hdl. (number of byte from the beginning.)

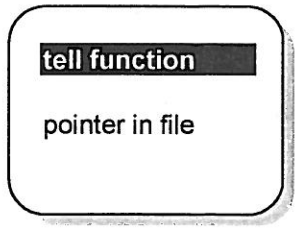
Return Value Get current position of file pointer. (number of byte from the beginning.) When the return value is -1, it indicates error, and EBADF is set in global variable, Errno, indicating file handle is invalid.

Example

```
/************************************************
 Library Function:  tell

 Description: This program gets the position
 of the file pointer.  If there is an error
 in the operation then it displays an error
 message.

 Caution: Create "demo.dat" file if its not there.
************************************************/
# include "libext.h"

void main(void)
{
   int handle;
   L  pos;

   dispcurp(0,1);
   disps(0,1," tell  function ");
   dispcurp(0,4);
   if (-1!=(handle=open("demo.dat",O_RDONLY)))
   {
      if(eof(handle)==0)
      {
```

```
            pos=tell(handle);  /* Get position of file pointer */
            if (pos!=-1)
            {
                disps(0,0,"pointer in file");
            }
        }
        else if (eof(handle)==1)
        {
            disps(0,0,"file end reached");
        }
    }
    else
    {
        disps(0,0,"file open error");
    }
    close(handle);
}
```

Output



eof

Description Check if file pointer has reached the end of file.

I eof(I hdl);

Remarks Check a file specified by hdl if the end of file is reached.

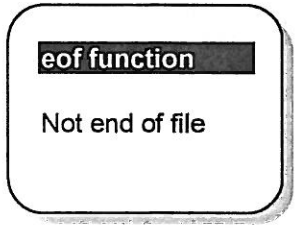
Return Value When the end of file is reached, 1 is returned, otherwise 0 is returned. In case of error, EBADF is set in the global variable Errno, indicating file handle is invalid.

Example

```
/************************************************
 Library Function: eof

 Description: This programs checks if the end
 of the file is reached or not.

 Caution: Create "demo.dat" file if its not there.
************************************************/
# include "libext.h"

void main(void)
{
   int handle;

   dispcurp(0,1);
   disps(0,1,"  eof  function ");
   dispcurp(0,4);
   if (-1!=(handle=open("demo.dat",O_RDONLY)))
   {
      /* Check for the end of file */
      if(eof(handle)==0)
      {
          disps(0,0,"Not end of file");
      }
      else if (eof(handle)==1)
      {
```

```
            disps(0,0,"file end reached");
        }
    }
    else
    {
        disps(0,0,"file open error");
    }
    close(handle);
}
```

Output



remove

Description Delete a file.

I remove(C far *fname);

Remarks Delete a file specified by fname.

Return Value When deleted 0 is returned. Otherwise, -1 is returned and the following value is set to the global variable, Errno.

EBADF There exists a file handle.
ENOENT The file is not found.

Example

```
/*********************************************************
 Library Function:  remove

 Description: This program shows how to delete a file.  It
 creates a file and opens it in the append mode.  It closes
 the file and then deletes the file and checks if the
 operation was successful or not.
*********************************************************/
#include "libext.h"

void main(void)
{
   int handle;

   dispcurp(0,1);
   disps(0,1," REMOVE  A  FILE");
   dispcurp(0,5);
   wait_time(100);
   if ((handle=open("demo.dat",O_CREAT|O_RDONLY|O_APPEND))!=-1)
   {
      close(handle);
      if(remove("demo.dat")==0)    /* delete the file */
      {
         disps(0,0," File deleted");
```

```
        }
        else
        {
            disps(0,0,"File not deleted");
        }
    }
    else
    {
        disps(0,0,"File open error");
    }
}
```

Output

freesize

Description Get free memory space.

L freesize(void);

Remarks Get free memory space.
The maximum usable bytes of a file is: 448K bytes for a model with 512KB; and 972K bytes for a model with 1MB.

Return Value Return unused memory space in byte.

Example See "fsize"

Output See "fsize"

fsize

Description Get file size.

L fsize(C far *fname);

Remarks Get file size specified by fname.

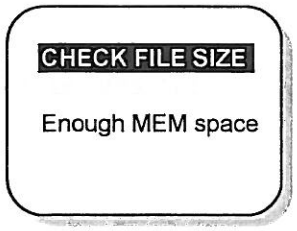
Return Value Return file size. When error occurs, - 1 is returned, and ENOENT is set to Errno.

Example

```
/*******************************************************
 Library Function:  fsize,freesize

 Description: This program opens a file and checks if the
 size of the file is smaller than the free memory space.
*******************************************************/
#include "libext.h"

void main(void)
```

```
{
 int handle;

 dispcurp(0,1);
 disps(0,1," CHECK FILE SIZE");
 dispcurp(0,5);
 wait_time(100);
 if ((handle=open("demo.dat",O_CREAT|O_RDONLY|O_APPEND))!=-1)
 {
     close(handle);
 /* Check if there is enough space in memory compare to file size*/
     if((fsize("demo.dat"))>(freesize()))
     {
        disps(0,0,"Not enough space");
     }
     else
     {
        disps(0,0,"Enough MEM space");
     }
  }
  else
  {
     disps(0,0,"File Error");
  }
}
```

Output



rename

Description Change the file name.

L rename(C far *oldname, C far *newname);

Remarks Change file name of a specified file.

Return Value When the change has been done, 0 is returned, otherwise - 1 is returned for error, and ENOENT is set in the global variable, Errno.

Example

```
/*********************************************************
 Library Function:  rename

 Description:  This program creates a file.  It then
 renames the file and checks if the operation is OK or not.
*********************************************************/
#include "libext.h"

void main(void)
{
   int handle;

   dispcurp(0,1);
   disps(0,1," RENAME  A  FILE");
   dispcurp(0,5);
   wait_time(100);
   if ((handle=open("old.dat",O_CREAT|O_RDWR|O_APPEND))!=-1)
   {
      close(handle);
      /* Rename the file */
      if(rename("old.dat", "new.dat")==0)
      {
         disps(0,0,"Filename changed");
      }
      else if(rename("old.dat","new.dat")==-1)
      {
         disps(0,0,"Name not changed");
```

```
        }
      }
      else
      {
         disps(0,0,"File open error");
      }
   }
```

Output

RTC Library

dateset

Description Set the date.

void dateset(UC *date);

UC date[7] yymmddw
(yy:year mm:month dd:day w:day of the week) specified by string.

Remarks Set the current date.

Return Value None.

dateget

Description Get date.

void dateget(UC *date);

UC date[8] yymmddw
(yy:year mm:month dd:date w:day of the week NULL) specified by string.

Remarks Get the date set in the PHL system.

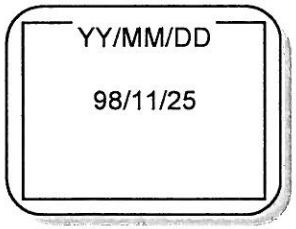
Return Value None.

Example

```
/*******************************************
 Library Function: dateget

 Description: This program gets the date from
 the PHL system and then displays it.
```

```
*******************************************/
#include <string.h>
#include "libext.h"

void main(void)
{
   UC    dt[10],wk[3];       /* Buffers to store date */

   kbclr();                  /* Clear the key buffer */
   displine(3,6,93,45);      /* Co-ordinates for grid box */
   dispcurp(4,1);
   disps(0,0,"YY/MM/DD");
   dispcurp(4,4);
   disps(0,0,"  /  /  ");

   dateget(dt);          /* Get the date from the system */
   wk[2] = 0;            /* Set the initial value to 0 */

   dispcurp(4,4);
   memcpy(wk,dt,2);      /* Memcpy dt to wk */
   disps(0,0,wk);
   dispcurp(7,4);
   memcpy(wk,&dt[2],2);  /* Memcpy the next item in dt to wk */
   disps(0,0,wk);
   dispcurp(10,4);
   memcpy(wk,&dt[4],2);  /* Memcpy the next item in dt to wk */
   disps(0,0,wk);
}
```

Output



timeset

Description Set time.

void timeset(UC *time);

UC time[6] hhmmss (hh:hour mm:minute ss:second) specified by string.

Remarks Set time for the PHL handset.

Return Value None.

timeget

Description Get time.

void timeget (UC * time);

UC time[7]
hhmmss (hh:hour mm:minute ss:second NULL) specified in string.

Remarks Get the time from the PHL system.

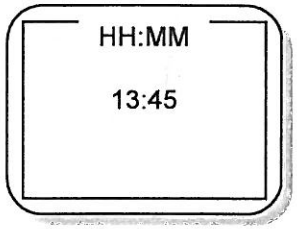
Return Value None.

Example

```
/*********************************************
 Library Function: timeget

 Description: This program gets the time from
 the PHL system and then displays it.
*********************************************/
#include <string.h>
```

```
#include "libext.h"

void main(void)
{
   UC    tm[10],wk[3];        /* Buffers to store time */

   kbclr();                   /* Clear the key buffer */
   dispclr();                 /* Clear the screen */
   displine(3,6,93,45);       /* Co-ordinates for grid box */
   dispcurp(4,1);
   disps(0,0," HH:MM ");
   dispcurp(5,4);
   disps(0,0,"   :   ");

   timeget(tm);           /* Get the time from the system */
   wk[2] = 0;             /* Set the initial value to 0    */

   dispcurp(5,4);
   memcpy(wk,tm,2);       /* Memcpy tm to wk */
   disps(0,0,wk);
   dispcurp(8,4);
   memcpy(wk,&tm[2],2);   /* Memcpy the next item in tm to wk */
   disps(0,0,wk);
}
```

Output



wait_time

Description Time delay.

void wait_time (UI time);

UI time : Wait time in 10msec

Remarks Simply wait for the specified duration.

Return Value None.

wait_time2

Description Specify the duration of wait, start, and detect timeup by the return value.

void wait_time2 (UI time, UI switch);

UI time: Duration in 10msec
UI switch: 0: start 1: stop 2: read

Remarks Specify wait time, start, and detect timeup by return value.

Return Value 0 : equals end of count
1 : does not equal end of countbuzzer

Buzzer Library

buzzer

Description **void buzzer(UI hz, UI time, UC led);**

UI hz : Frequency
UI time Duration
(Unit : msec)
UC led: LED (0:OFF 1:Red 2:Green 3:Orange)

Remarks Turn on buzzer for the specified duration. (higher priority than key click). When LED is turned on by using ledon function, LED OFF and the change of color are not performed by this function.

Return Value None.

Example See "buzzer_stop"

Output See "buzzer_stop"

buzzer_stop

Description Stop buzzer.

void buzzer_stop(void);

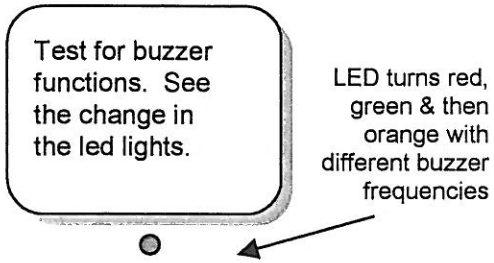
Remarks Stops buzzer (higher priority than key click).

Return Value None.

Example

```
/************************************************
 Library Functions:  buzzer,buzzer_stop

 Description: This program sets the buzzer ON
 and OFF with different led lights and different
 buzzer frequencies.
************************************************/
#include "libext.h"

void main(void)
{
   dispcurp(0,1);
   disps(0,0,"Test for buzzer");
   dispcurp(0,3);
   disps(0,0,"functions.  See");
   dispcurp(0,5);
   disps(0,0,"the change in");
   dispcurp(0,7);
   disps(0,0,"the led lights.");

   while(1)
   {
      /* Buzzer ON:1000 hz freq,50msec time,red led ON */
      buzzer(1000,50,1);
      wait_time(100);
      buzzer_stop();     /* Stop the buzzer */
      buzzer(1500,50,2); /* Buzzer ON with different values */
      wait_time(100);
      buzzer_stop();     /* Stop the buzzer */
      buzzer(2000,50,3); /* Buzzer ON with different values */
      wait_time(100);
      buzzer_stop();     /* Stop the buzzer */
      /* Buzzer ON with lesser freq and led is OFF*/
      buzzer(1000,50,0);
   }
}
```

Output


Test for buzzer functions. See the change in the led lights.
LED turns red, green & then orange with different buzzer frequencies

ledon

Description Turn on LED.

void ledon(UI led,UI switch);

UI led: 0: Red 1: Green 2: Orange
UI switch: 0: OFF 1: ON

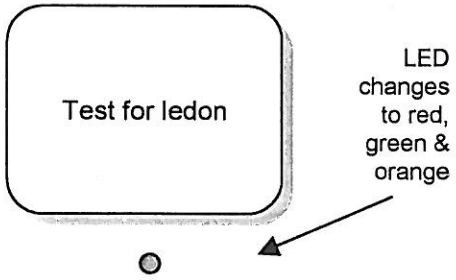
Remarks Turn on the specified LED.
When the LED is turned on by this function, it is not turned off by the buzzer function.

Return Value None.

Example

```
/*********************************************
 Library Function:  ledon

 Description: This program turns the different
 types of LED ON.
*********************************************/
#include "libext.h"

void main(void)
{
    while(1)
    {
       dispcurp(0,4);
       disps(0,0,"Test  for ledon");
       ledon(0,1);            /* Red led is ON */
       wait_time(200);
       ledon(1,1);            /* Green led is ON*/
       wait_time(200);
       ledon(2,1);            /* Orange led is ON */
       wait_time(200);
       ledon(0,2);        /* Orange led is turned OFF*/
    }
}
```

Output



back_light

Description Back light ON/OFF

void ledon(UI typ);

UI typ : 1:ON 0:OFF

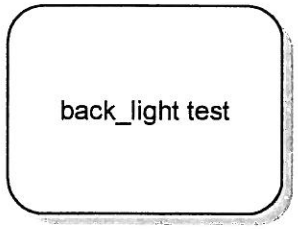
Remarks Turn the back light ON/OFF.

Return Value None.

Example

```
/*********************************************
 Library Functions used: back_light

 Description: This program turns the back light
 ON and OFF.

 Caution: Viewed better if performed in a darker
 environment.
*********************************************/
#include "libext.h"

void main(void)
{
    while(1)
    {
       dispcurp(0,4);
       disps(0,0," back_light test");
       wait_time(100);
       back_light(1);         /* Backlight is ON  */
       wait_time(200);
       back_light(0);         /* Backlight is turned OFF */
    }
}
```

Output



Note: The back light will glow ON and OFF.

tmid_get

Description Get ID of terminal.

void tmid_get(UC *id);

UC id[2] : ID area

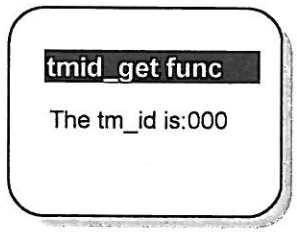
Remarks Get the terminal ID set in the PHL system.
I D consists of 1-16.

Return Value None.

Example

```
/*****************************************
 Library Function: tm_id

 Description: This program gets the terminal
 id set in the PHL system and displays it.
*****************************************/
#include "libext.h"

void main(void)
{
   UC id[5];

   dispcurp(0,1);
   disps(0,1," tmid_get  func ");
   tmid_get(id);          /* Get the terminal id */
   dispcurp(0,5);
   disps(0,0,"The tm_id is:");
   dispcurp(13,5);
   disps(0,0,id);
}
```

Output



Appendix 2

PHL-1600 Specifications

Physical

Case Material	ABS Plastic
Dimensions	6.75" x 2.25" x 1.5"
Weight	7.8 oz., including batteries

Environmental

Temperature
- Operating +32° to +122° F
- Storage -4° to +140° F

Humidity (non-condensing)
- Operating 20% - 90%

Drop Test 4' to concrete, 10 drops, all sides

ESD 15KV static discharge

Optical

Laser	670 nm (visible red)
Output power	<1 mw
Scan rate	100 scans/second
Focal distance	0 - 15"
Min. bar/space width	5 mils (minimum)
Depth-of-field:	10" (10 mil, 0.9 PCS)
Field Width	(@1" & 15" inches from window) @ 1": 2.3" / @ 15": 9.7"

Decodable Symbologies

UPC-E & -A Standard 2 of 5
EAN 13 & 8 Code-11
Code 39 Code-93
Codabar MSI
Code 128Interleaved 2 of 5

Communications

Serial RS232 2400 - 19200 b/s

Optical IrDA (ver1) 2400 - 115200 b/s

Display

Backlit 96 x 48 pixel graphic LCD
Font: Alphanumeric
Number of characters (programmable): 16 char x 8 lines
12 char x 6 lines
16 char x 4 lines
12 char x 3 lines
8 char x 4 lines
6 char x 3 lines

Power

Operating voltage	3 Volts
Battery life @ 1 scan/ 5 sec	AA Alkaline cells (x2):120 hours NiMH (Rechargeable): 60 hours Built-in recharger circuit
Low battery indicators	Main & Backup batteries
Backup power	Lithium battery capable of maintaining data in RAM for four months w/o main battery
Automatic power down	Activates after 0 - 60 minutes of inactivity, user programmable

CPU & Memory

CPU 16-bit CMOS Mitsubishi

Memory 512 kb Data RAM standard; PHL-1600-05
1 mb Data RAM optional; PHL-1600-10
256K Flash memory
64K RAM working memory

Programming

Host	PC running MS-DOS ver 3.0+; Windows V3.1 or Windows 95
Language	C
Cross Compiler	ICC7700 (IAR Systems)

Other features

Real time clock
Programmable LED (tri-color) indicator
Programmable Tone generator

Accessories

- RS232 cable - direct connection to PHL-1600 RS232C port
- NiMH battery Pack

Warranty

One year from date of purchase

Ordering Information

Description	Part Number
PHL-1600 Terminal /1 Mb Memory	PHL-TRM10B-00
PHL-1600 Terminal /½ Mb Memory	PHL-TRM05B-00
RS232 Communications Cable	41-PHL232-00
Battery / Lithium Back-up Battery	02-BATLTH-00
Battery / NiMH Battery Pack	02-BATNMH-00
In-Terminal Direct Battery Charger	PHL-TRMCH00
IAR Cross Compiler	42-SFTCMP-00
Holster	PHL-HOLSTER-01
IR Transceiver	PHLIR-01
RF Transceiver	36-RF-01
QuickStart Manual	25-TRMQS-02
Technical Manual	25-TRMTK-01

Opticon continually introduces new products in order to better serve our customers. Since this list of products available may be updated or revised from time to time, please contact Opticon or your Opticon representative toll free at (800) 636-0090 should you have any questions or require further assistance in any way.

Appendix 3

Help

Opticon may be reached at:

Opticon, Inc.
8 Olympic Drive
Orangeburg, NY 10962

Tel (914) 365-0090

Toll free (800) 636-0090

Fax (914) 365-1251

www.opticonUSA.com

E-Mail techsupport@opticonUSA.com

Ask for PHL-1600 technical support.

Appendix 4

Rapid GEN™ Application Generator

Rapid GEN™ is a PC-based utility that provides a simple means to create simple terminal-resident applications for the PHL-1600. It also contains several "standard" programs that may be used directly as written for many common applications, avoiding the necessity to create and compile separate programs.

To use, follow the instructions for loading the program into a PC. The README file contains detailed instructions for using Rapid GEN and accessing the standard programs.

Procedures for transferring files to the PHL-1600 are contained in this user's manual.

Index

A

B

C

D

E

F

G

H

I

K

L

M

O

NOTE: See **bolded** page number for function designation