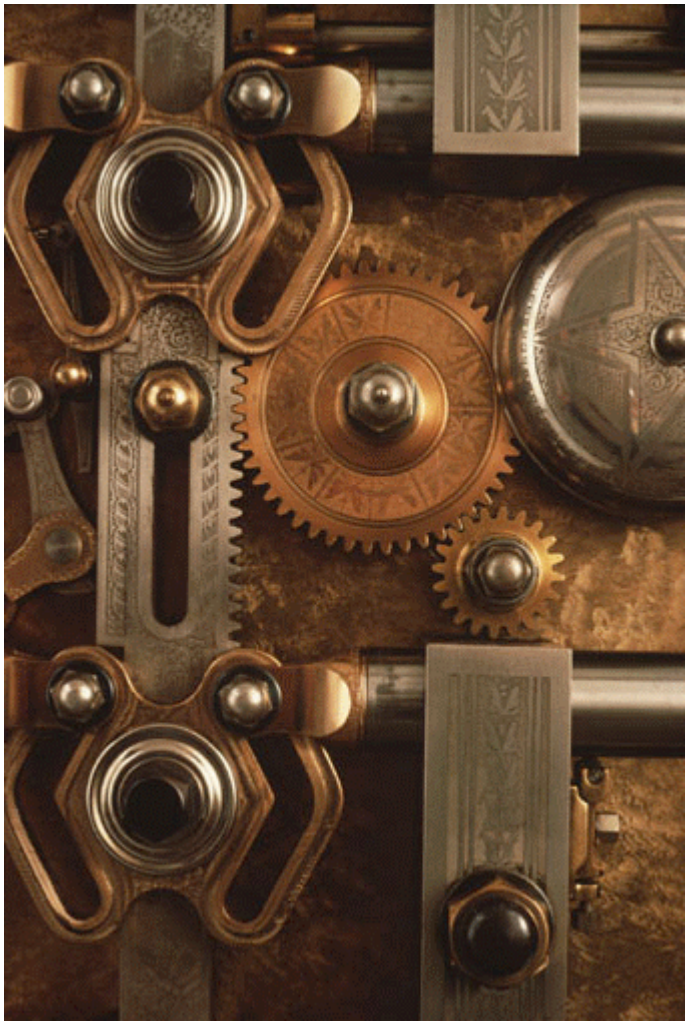


User's Guide

pico

Viewer®

6502

v.1.01

Copyright Notice

www.picodoc.com
Oakville, Ontario
Canada

PV-RE-012104

End User License Agreement (EULA)

Use in part or in whole of this software implies entire acceptance of the following terms and conditions. The numbered headings used in this agreement are provided for convenience only and shall not be used to construe meaning or intent.

Picodoc Inc. (the "Owner"), Oakville, Ontario, Canada and the user (the "Licensee") of picoViewer(tm) Standard Edition (the "Software"), hereby agree to the following:

1. Evaluation License Agreement:
The Owner grants to the Licensee a non-exclusive, royalty-free license to use the "evaluation" and/or "beta" versions of the Software for the purposes of assessing its suitability for subsequent license purchase. All other terms and conditions that follow remain in effect for both evaluation and licensed versions.

2. License
The Owner grants to the Licensee a non-exclusive, royalty-free, non-transferable license to use the Software and its documentation in accordance with this agreement. This license allows the use of the Software by a single individual only. Licensee shall not copy, modify, duplicate, reproduce, distribute, license or sublicense the Software or transfer or convey any right in the Software except to make a single backup copy for the Licensee's own personal use. Picodoc Incorporated retains sole title, all rights, and full ownership of the Software and associated documentation and promotional materials. U.S. and Canadian copyright laws and international treaty provisions protect this ownership.

The Owner grants the Licensee the right to develop and distribute knowledge bases created using the Software. Such knowledge bases may be distributed on a royalty-free basis, electronically or in machine-readable form.

3. Restrictions
Modification, de-compilation, reverse engineering, or obfuscation of the Software, or inclusion of the Software in other products is strictly prohibited. The Software is the intellectual property solely of the Owner at all times.

No right to receive enhancements or updates to the Software or associated documentation is granted to the Licensee. The Licensee at the Owner's then-current prices and conditions may obtain such enhancements or updates, if and when they become available. No right to receive technical support from the Owner is granted to the Licensee.

4. Fee
In consideration for the granting of the license and the use of the Software, the Licensee agrees to pay the Owner the then-current license fee for the Software as specified by the Owner. Price and availability are subject to change without notice.

5. Warranty of Functionality
For a period of twelve (12) months following delivery of the Software to the Licensee (the "Warranty Period"), the Owner warrants that the Software shall perform in all material respects according to the Owner's specifications. In the event of any alleged breach of this warranty, the Licensee shall promptly notify the Owner. The Licensee's sole remedy shall be that the Owner shall correct the Software to bring it into accordance with the Owner's specifications. The Owner makes no warranty or representation that the Software will meet the Licensee's requirements or those of any third parties. The Owner makes no warranty or representation to the Licensee that the Software will be free of defects or errors.

6. Warranty Disclaimer
THE OWNER WARRANTIES SET FORTH IN THIS AGREEMENT ARE EXCLUSIVE AND ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

7. Limitation of Liability
The Owner shall not be responsible for any amount of incidental, consequential or other indirect damages, whether based on lost revenue or otherwise, regardless of whether the Owner was advised of the possibility of such losses in advance. In no event shall the Owner's total liability hereunder exceed the amount of license fees paid by the Licensee, regardless of whether the Licensee's claim is based on contract, tort, strict liability, product liability or otherwise.

8. Governing Law
This agreement shall be construed and enforced in accordance with the laws of the province of Ontario, Canada.

9. Severability
If any term or condition of this agreement is held by a court of competent jurisdiction to be invalid or unenforceable, then this agreement, including all remaining provisions, will remain in full effect as if such unenforceable provision had never been included in the agreement.

10. No Assignment
The Licensee may assign neither this agreement nor any interest in this agreement.

11. Final Agreement
This agreement terminates and supercedes all prior understandings or agreements on the subject matter hereof. Only a further writing that is duly executed by both parties may modify this agreement.

www.picodoc.com
Oakville, Ontario
Canada

Contents:

1 Introduction……. 1-1
1.1 Installation …… 1-2
1.1.1 System Requirements ……..... 1-2
1.1.2 Installation Procedure ..……...……...… 1-2
1.1.3 Running picoViewer®……. 1-3

2 Object Files ……. 2-1
2.1 Intel Hex Files …… 2-2
2.2 Motorola S-Records …… 2-2
2.3 Raw Files …… 2-3
2.4 Creating Object Files ……... 2-4
2.5 Object File Errors ……... 2-5

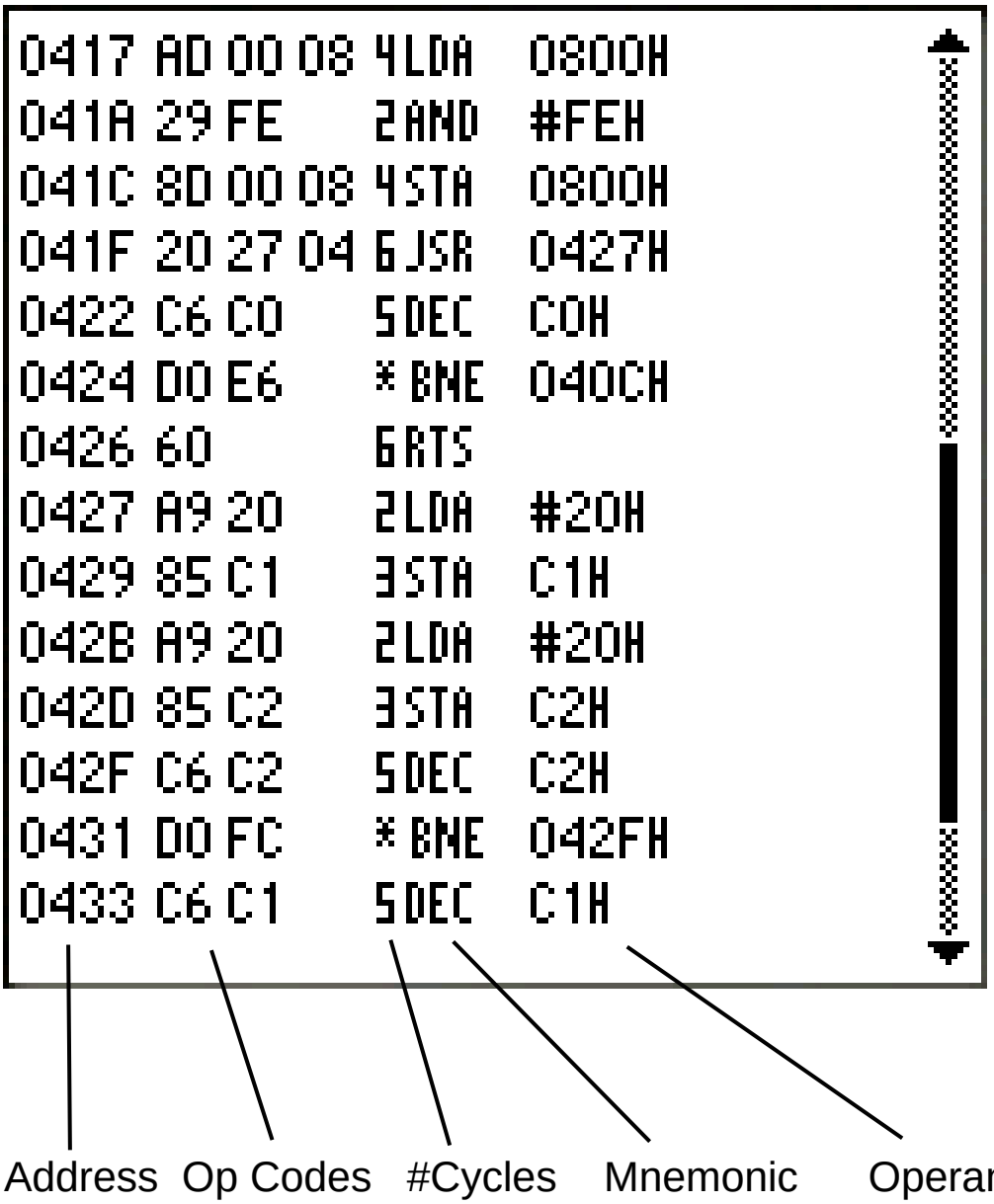
3 The User Interface ……. 3-1
Preference Settings ……..…….. 3-3
‘From’ …… 3-4
‘Jump’ …… 3-5
‘Goto’ …… 3-6

Architecture Notes …… A-1

Assembly Listings for Sample Files ..…… B-1

ASCII Character Set …….. C-1

Disassembly Window



1. Introduction

Almost all computer systems store programs and data as object code. Object code is also known as machine-readable code because it is the form that the microprocessor understands. Object code is not very human-readable.

Almost all computer software is written by humans, even robotic and embedded systems software. A range of languages can be used to translate a programmer's instructions into machine-readable form. The lowest level and common root of all of these is assembly language. A tool that translates assembly language to object code is called an assembler. Conversely, a disassembler is a tool that allows the reconstitution of assembly language listings from object code.

This guide applies to the picoViewer® series of disassemblers in general and the MOS 6502 microprocessor version in particular.

Why use a handheld?

Traditional disassemblers typically run on a Personal Computer (PC), not on a handheld Personal Digital Assistant (PDA). That's great if you have a PC with you and lots of space to set it up, and if you need a fully detailed, printed disassembly for your records. In fact, for a 500K Windows program that uses hundreds of pre-defined symbols and dozens of associated libraries that may be the only way to go.

A PDA, in contrast, allows you to put your disassembler right in the midst of the embedded system you're currently working on, just like any other small piece of test equipment. Even more advantageous is the ability to use a PDA in environments where a PC is far too cumbersome such as while working on installed equipment, outdoors, in remote locations, or while commuting.

Since picoViewer® is designed for a PDA, the user interface is greatly simplified over more elaborate tools. All file operations are accomplished via the PDA's own Memopad application. Object files can be loaded via HotSync, infrared, serial connection, or even directly in a memo. Disassembly is simple, automatic, and fast, allowing you to stay close to your work instead of reaching for a PC software manual.

We hope that after trying picoViewer® on your embedded projects, you'll agree that it's a valuable addition to your workbench.

1.1 Installation

1.1.1 System Requirements:

- Palm OS™ version 3.0 or later capable device
- at least 150K of free memory space
- Palm Desktop™ software, which includes the HotSync™ manager
- HotSync™ connection (serial/USB/Infrared/BlueTooth™/etc)

1.1.2 Installation Procedure:

- On your PC, start the Palm Desktop and choose the Install Tool. Alternatively, if your file types have been set to automatically install .PRC files, you only need to open any .PRC file to install it, and then you can skip to step 4.
- If you use more than one handheld with this PC, select the specific device to install picoViewer® on.
- Click Add on the dialog. Navigate to the PV6502.PRC file and click on it.
- Click "Done" on the selection dialog, then "OK" on the Install dialog. The PRC file will be installed to your handheld device the next time you HotSync with this PC.
- You can move picoViewer® to any category you like your PDA. If you have multiple versions of picoViewer® installed, you may wish to gather them together in the same category.

Sample Object Code Files:

The sample files included picoViewer® are distributed in plain text format. In order to install them as memos, you can use the Palm Desktop application. Simply open a sample file in Notepad (or other editor) on your PC, and select and copy the text to the clipboard. Then, create a new memo in Palm Desktop and paste the copied text into it. The memo will be installed to your PDA by the next HotSync, making it available to picoViewer®.

1.1.3 Running picoViewer®

To run picoViewer®, select its icon from the program launcher:

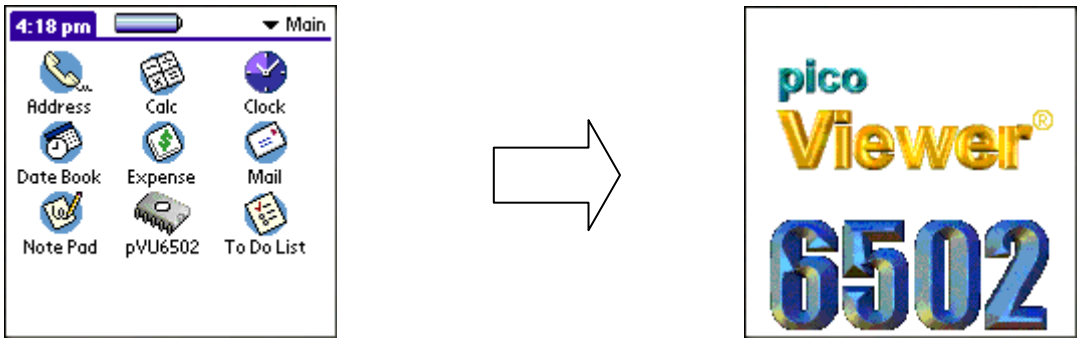


Figure 1.1 PDA's Launcher

The splash screen identifies which version of picoViewer® this is. You can disable the splash screen via user preferences, which will be discussed later.

Tap on the splash screen to begin a picoViewer® session. If an object file was successfully loaded during your last session, it will be automatically loaded and disassembled. Otherwise, you will be prompted to specify an object file to load.

2. Object Files

Object files are the lists of machine language instructions to be disassembled back into source code mnemonics. In picoViewer®, these files are stored as Memopad notes (memos). This allows for all creation, editing, backup, and beaming operations to be done in a familiar and efficient application that you already know and use. That's it – no proprietary file interface to learn. The only real drawback with this is that Memopad notes are limited to about 4K of storage each, so you will have to break up large object files into 4K blocks if they go beyond this limit. However, if you're dealing with very large object files, you may prefer to work with a desktop disassembler anyway.

Note: Some late model PDAs, the Palm® Tungsten|T3 for one, allow for memos up to 32K. Although picoViewer® can read such memos, it does not read more than 4K of object code from them.

Three types of object file formats are supported: Intel Hex, Motorola S-Records and Raw data. Each format represents one byte of hexadecimal object code as two ASCII characters. For example, the 8-bit binary value 00111110 is 3E in hexadecimal notation. This value will be stored as the ASCII characters '3' and 'E' (33H and 45H).

The first line of an object file is used to identify it as a picoViewer® file, and to provide a file name. The first character on this line must be a '%' (percent) identifier. This identifies all object files to picoViewer®. It also allows all object files to appear together in any given Memopad category for convenience. The next character specifies the format of the object file. An 'I' specifies Intel Hex format, an 'S' specifies Motorola S-Record format, and an 'R' specifies Raw format. A space is required as the third character. The rest of the line is used for the filename you specify. The text on this line is not case-sensitive. Here are some examples:

```
%I Laser Calibrator
%S Stepper Controller (1 of 3)
%R Lighting Director for Stage


These are invalid:

Laser Calibrator
%RStepper Controller (1 of 3)
I Lighting Director for Stage
```

2.1 Intel Hex Files

Intel Hex is a standard format widely used in embedded systems. In picoViewer®, a subset of the full Intel standard is implemented. Only Data records and End of File records are processed. Other records types will be ignored. The Data record type allows segment information to be included which is important when your object file exists as two or more non-contiguous blocks of code. Segments of code will be loaded in the order they appear in the hex file.

Here is the general format of an Intel Hex record:

:|nn|aaaa|tt|bb|bb|bb|…|cs

where:

: = record marker (':' which is ASCII 3AH)
nn = number of data bytes in record
aaaa = address
tt = record type (00-Data 01-End of File)
bb = data bytes (variable length field)
cs = checksum (for error checking during load)

} included in 8-bit sum checksum = 2's complement

For example, here is a Data record that contains the three bytes 3FH, 60H, 22H to be loaded at address 1234H, followed by an End of File record:

```
:031234003F6022F6
:00000001FF
```

2.2 Motorola S-Records

This standard is widely used with Motorola microprocessors and microcontrollers. In picoViewer®, a subset of the full S-Record standard is implemented. Only Data records and End of File records are processed. Other records types will be ignored. This includes S0 records which are often used to name the input file. You must still provide a '%S name' title line for your input file as described above. The Data record type allows segment information to be included which is important when your object file exists as two or more non-contiguous blocks of code. Segments of code will be loaded in the order they appear in the S-Records file.

Here is the general format of an S-Record:

Sn|nn|aaaa|bb|bb|bb|…|cs

where:
Sn = record marker (S1-Data record, S9-End of File record)
nn = number of data bytes in record + 3
aaaa = address
bb = data bytes (variable length field)
cs = checksum (for error checking during load)

} included in 8-bit sum checksum = 1's complement

For example, here is a Data record that contains the three bytes 3FH, 60H, 22H to be loaded at address 1234H, followed by an End of File record:

```
S10612343F6022F2
S9030000FC
```

2.3 Raw Files

A Raw file simply contains a list of bytes stored as two hexadecimal digits per byte. Spaces and line breaks are optional, however no spaces are allowed between the two nibbles (4-bit halves) of one byte. Bear in mind that while spaces and line breaks will enhance the readability of your object file (if that's important to you) they will also reduce the overall storage capacity of each individual memo file. Raw object files have no provision for segmentation or checksum verification, and will always load at address 0000. Here are two examples of Raw object files:

```
%R Transmit
020100900118120108 80FEE493
A3B4000122308C99FDC299F599
21084558414D504C45002B


%R Power Up
020060
32
32
32
32
30980BC298E599F5993099FDC299
32
```

2.4 Creating Object Files

Object files of course must be loaded onto your PDA before they can be disassembled. There are several ways to do this.

The first way is simply to enter them into a memo by hand. This is often convenient for short sequences of code that would otherwise require connection and setup of special equipment. Just start a new memo, enter the first line as an object file descriptor as described above, and enter the object code as raw bytes. This is often the quickest method for entering object code from a printed page.

The second method is to HotSync with a PC. Any text file (up to 4K in length) can be added to your PDA's Memopad using the Palm Desktop application. You can copy/paste/edit text to your user directory using the Palm Desktop and it will be installed to your PDA on the next HotSync. This is a good way to load a lot of object code if you already have it available as text files.

The third method is to directly upload code from an EPROM or microcontroller in the embedded system you're working on. In order to do this, your embedded system must support the serial transfer of object code (most do), preferably in Intel Hex format. You will also need a terminal program running on your PDA. There are several of these available; **Online** from www.markspace.com is one you may wish to take a look at. It has the convenient feature of splitting large object files into sets of consecutive memos.

Which terminal program to choose depends on your preferences, and whether the transfer of data should be accomplished by serial cable, USB, Infrared, BlueTooth™, etc. Bear in mind that all PDAs and most embedded systems' serial ports are wired to be connected to a PC, not to each other, so you'll need a null-modem adapter to connect a PDA directly to an embedded system in most cases when using a serial cable. Check with the manufacturer of your embedded system and/or PDA for specific connection information.

Recently, more advanced embedded systems have even begun to incorporate removable flash memory in formats compatible with palm computers. This is the most elegant method of uploading object code and is definitely the wave of the future.

2.5 Object File Errors

Whenever you specify a new object file to open within picoViewer®, that file is read into an internal image of the target processor's memory. Any error detected during loading will abort the loading of the object file and display a message similar to the following:

This message indicates that there is an invalid or missing character in the object file at the location indicated.

Should such an error occur, you must locate and correct it in the object code file using Memopad. Most often, the cause is a space or line break in the wrong place.

Should a checksum error occur in an Intel Hex object file, a message similar to the following will be displayed:

This message indicates that the checksum verification failed for the eight-byte record beginning at address 0028.

If an error occurs in a Raw object file, the displayed message will show the preceding few bytes before the error to aid you in locating its position.

Note that there is no target processor type check done when an object file is specified and loaded. That is, nothing will prevent you from incorrectly disassembling Z8 object code with the 6800 version of picoViewer®. For this reason, it is advisable to include the target processor type in the object file name to avoid confusion when working with multiple target processors.

3. The User Interface

The controls for picoViewer® are very straight forward. The main display is the disassembly window, which is the scrollable source listing. The menu (activated with the menu button on your PDA) allows for the handful of operations needed such as loading object files, moving around within the source listing, and setting preferences.

To load an object file for disassembly, use the Open command in the menu, or the O shortcut. This will open the following window:

Figure 3.1 Open Object File

The '%' identifier is already entered for you. You need only specify, 'I', 'S', or 'R' indicating the file format, followed by a space. Then enter as many letters of the desired file name as needed to distinguish that file from all other object files in the Memopad. The desired file will then be loaded and disassembled.

On a PC, you have a very large screen size, capable of displaying thousands of characters of text, with well over 100 characters per line. You may also have a printer attached, allowing similar spacious display text.

On a standard PDA, the environment is more modest. The screen is only 160 x 160 pixels, about 1/30th that of a PC! The only input is a pen for tapping buttons and entering graffiti characters. Processor power, memory and I/O are similarly restricted. For these reasons, picoViewer® employs a few tricks. Let's have a look at them one by one.

First, picoViewer® uses a custom screen font. This font has compressed text, allowing one line of disassembly (address/object code/mnemonic/operands) to be displayed on a single line of the PDA display. Numbers, however, are not compressed. This makes for a very comfortable and usable output format:

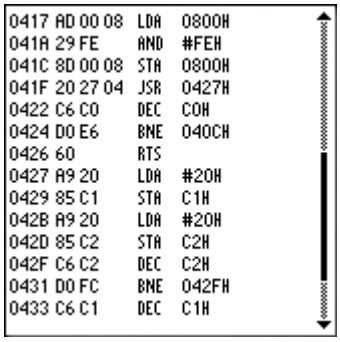


Figure 3.2 Screen Font

Only text is compressed; addresses, op-codes, and numeric operands are all shown at full size, and in fixed width spacing for clarity.

You can copy a block of this source listing into another application (such as Memopad). This is useful if you need to edit the disassembly listing manually, perhaps to add comments or labels.

Use the Copy command from the options menu or the C shortcut. Select a section of the listing using the pen or with the Select All command from the options menu, and then copy it to the clipboard. The clipboard within picoViewer® has been expanded to about 4K in size, allowing you to copy a block of text up to the size of a full memo. If you have selected more text than will fit within this clipboard, you will see an error message when you attempt to Copy:

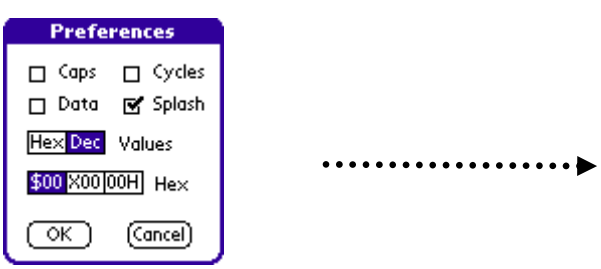
In the Standard Edition, the source listing is limited to 30 lines. If that limit is reached, a warning message to that effect will be displayed at the end of the disassembly listing.

The source listing in the disassembly window can be quite long. There is 64K of memory allocated for displaying it. So, just as very long object files must be divided into several smaller memos, very long source listings can only be copied to another application with repeated copy/paste operations.

picoViewer® uses the following default display preferences (see Figure 3.2 for an example of the default display):

Caps: ON
Cycles: OFF
Data: OFF
Splash: ON
Numeric Base: Hex
Hex Identifier: H suffix

You can alter these defaults using the Preferences dialog. To open the preferences window, select Preferences from the options menu or use the P shortcut:



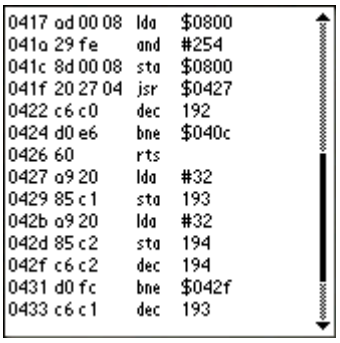


Figure 3.3 Preferences

Preference Settings:

Caps	Capitalization (ON or OFF)
Cycles	Show #cycles per instruction (ON or OFF)
Data	Show data only, do not disassemble instructions (ON or OFF)
Splash	Show splash screen at startup (ON or OFF)
Values	Format for 8-bit operand values (Hex or Decimal)
Hex	Format for hexadecimal operand values ($ prefix, X prefix or H suffix)

When dealing with unfamiliar object code, it is often necessary to explore for regions of data embedded within the code. The Data Only setting in preferences is useful here. With this setting turned on, only data will be shown (in the format specified by the other settings):

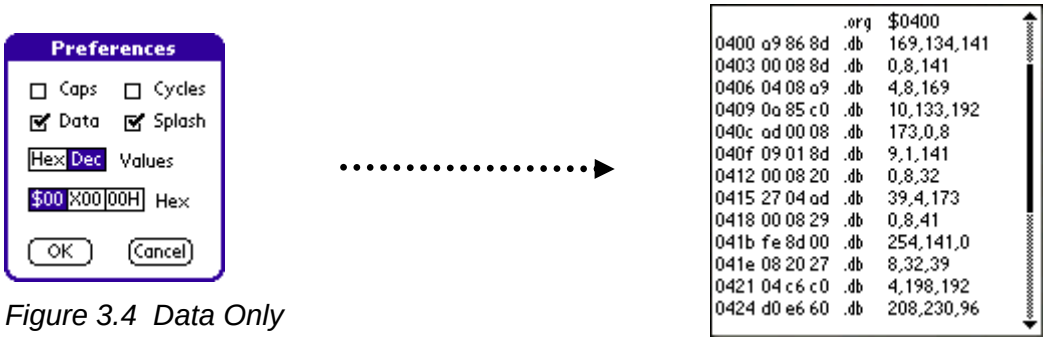


Figure 3.4 Data Only

Whenever you alter preferences, the disassembly listing will be re-drawn to reflect any changes. Also, your new preferences will be saved as the default settings to be used the next time you use picoViewer®, and will remain in effect until you change them again.

You can begin disassembly from any address within the object file using the From command (either by menu selection or the F shortcut):

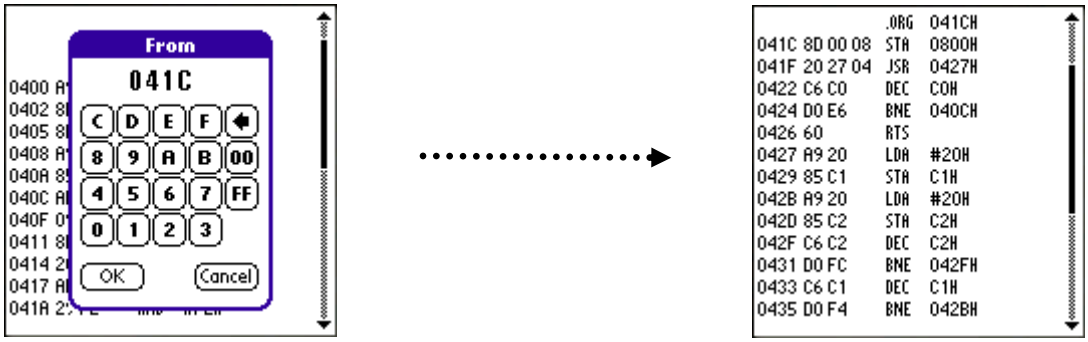


Figure 3.5 Disassemble 'From' address

You can disassemble from the start again by choosing From 0000.

Next, picoViewer® does not use labels. Labels generated by most disassemblers are notoriously vague and cryptic (usually L_01, L_02, etc). Also, they consume precious screen width without contributing significant information, especially when only a few lines of code are visible at once.

Instead of labels, picoViewer® implements a feature we call *active addresses*. To view the code at an address within the program, simply highlight that address and execute a **Jump** (either by menu selection or the J shortcut). This will move the view to that address:

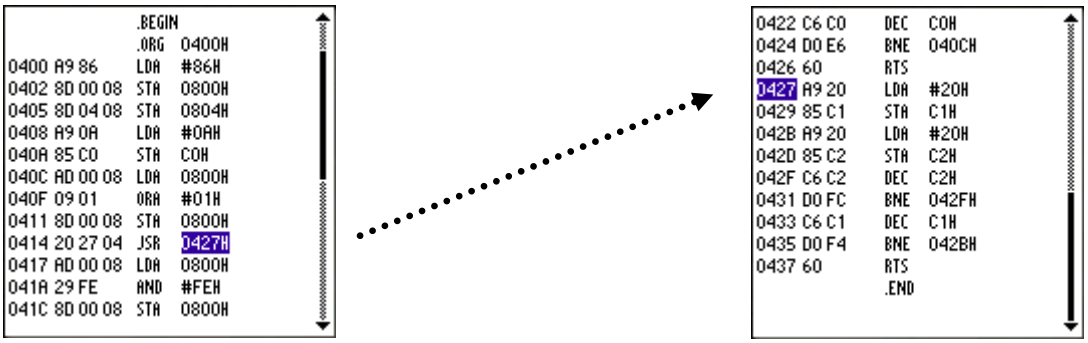


Figure 3.6 Active Address Jump

You must select a four-digit address with the pen before executing a Jump. Extraneous punctuation around the address will be ignored. However, should the target address be less than four digits long, no jump will occur. Should the target address not appear your defined program space, the following warning will be displayed:

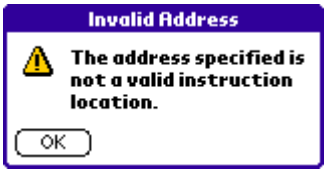


Figure 3.7 Invalid Address

In addition, you can **Goto** any address within your code (either by menu selection or the G shortcut). A Goto is initiated by entering the target address with the address entry pad:

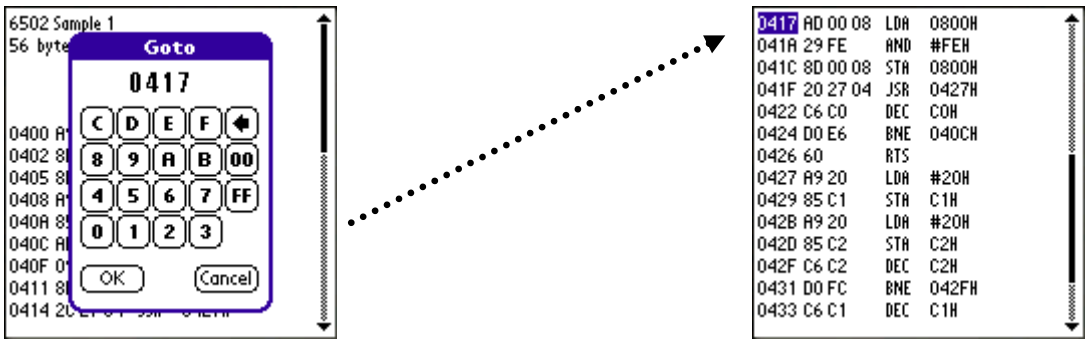


Figure 3.8 Goto Active Address

Use this keypad to enter any hexadecimal address within your code. The backspace key deletes the last digit entered, or the entire address if it has not been edited yet. The '00' and 'FF' buttons fill the remainder of the address with 0's or F's. The Cancel key will abort editing and return to the previous view in the disassembler.

This interactive nature and quick, in-your-hand disassembly makes picoViewer® a valuable addition to your bench instrument set. Testing and analysis can be done without the intervention of a PC and/or printer, making it useful as an exploration tool, especially when searching for code vs. data blocks and other subtle structures within the object file. This makes life easier when you work with object code about which little is known, such as older embedded systems for which the original source is missing or incomplete.

Appendix A: 6502 Architecture Notes

While a full description of MOS 6502 architecture is beyond the scope of this user's guide, here are a few architecture and naming notes:

CPU Registers

Accumulator A

7	6	5	4	3	2	1	0

Index Register X

7	6	5	4	3	2	1	0

Index Register Y

7	6	5	4	3	2	1	0

Program Counter PC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Stack Pointer S

8	7	6	5	4	3	2	1	0

Processor Status P

N	V	-	B	D	I	Z	C

N – Negative
V – Overflow
B – BRK Command
D – Decimal Mode
I – IRQ Disable
Z – Zero
C – Carry

Interrupt Vectors

HEX	Name	picoViewer®	Description
FFFA	NMIL	FFFA	NMI Vector (Low)
FFFB	NMIH	FFFB	NMI Vector (High)
FFFC	RSTL	FFFC	Reset Vector (Low)
FFFD	RSTH	FFFD	Reset Vector (High)
FFFE	IRQL	FFFE	Interrupt Vector (Low)
FFFF	IRQH	FFFF	Interrupt Vector (High)

Notes:

- cycle times for 6502 branch instructions are displayed as *
  This is because branch instructions require 2 cycles if the branch is not taken, 3 cycles if the branch is taken within same page, and 4 cycles if the branch is taken to another page of memory. This information is only determinate at run time.

Appendix B: Assembly Listings for Sample Files

Listing for 6502_test.txt

```
;Test all 6502 instructions and addressing modes

                        .BEGIN
0000
0000            b:      .EQU    $12             ;an 8-bit value
0100                    .ORG    $0100
0100            aaaa:                           ;an absolute address
0100
0100                                            ;add to A with carry
0100 69 12              ADC     #b              ;immediate
0102 61 12              ADC     (b,X)           ;indexed indirect (X)
0104 71 12              ADC     (b),Y           ;indirect indexed (Y)
0106 75 12              ADC     b,X             ;zero page indexed (X)
0108 7D 00 01           ADC     aaaa,X          ;absolute indexed (X)
010B 79 00 01           ADC     aaaa,Y          ;absolute indexed (Y)
010E 65 12              ADC     b               ;zero page
0110 6D 00 01           ADC     aaaa            ;absolute
0113
0113                                            ;and with A
0113 29 12              AND     #b              ;immediate
0115 21 12              AND     (b,X)           ;indexed indirect (X)
0117 31 12              AND     (b),Y           ;indirect indexed (Y)
0119 35 12              AND     b,X             ;zero page indexed (X)
011B 3D 00 01           AND     aaaa,X          ;absolute indexed (X)
011E 39 00 01           AND     aaaa,Y          ;absolute indexed (Y)
0121 25 12              AND     b               ;zero page
0123 2D 00 01           AND     aaaa            ;absolute
0126
0126                                            ;arithmetic shift left
0126 0A                 ASL     A               ;A
0127 16 12              ASL     b,X             ;zero page indexed (X)
0129 1E 00 01           ASL     aaaa,X          ;absolute indexed (X)
012C 06 12              ASL     b               ;zero page
012E 0E 00 01           ASL     aaaa            ;absolute
0131            lab1:
0131 90 FE              BCC     lab1            ;branch on carry clear
0133 B0 FC              BCS     lab1            ;branch on carry set
0135 F0 FA              BEQ     lab1            ;branch on equal
0137 D0 F8              BNE     lab1            ;branch on not equal
0139 30 F6              BMI     lab1            ;branch on minus
013B 10 F4              BPL     lab1            ;branch on plus
013D 50 F2              BVC     lab1            ;branch on overflow clear
013F 70 F0              BVS     lab1            ;branch on overflow set
```

```
0141
0141 24 12           BIT     b           ;bit test zero page with A
0143 2C 00 01        BIT     aaaa        ;bit test absolute with A
0146
0146 00              BRK                 ;force interrupt (break)
0147
0147 18              CLC                 ;clear carry
0148 D8              CLD                 ;clear decimal mode
0149 58              CLI                 ;clear interrupt disable
014A B8              CLV                 ;clear overflow
014B
014B                                     ;compare with A
014B C9 12           CMP     #b          ;immediate
014D C1 12           CMP     (b,X)       ;indexed indirect (X)
014F D1 12           CMP     (b),Y       ;indirect indexed (Y)
0151 D5 12           CMP     b,X         ;zero page indexed (X)
0153 DD 00 01        CMP     aaaa,X      ;absolute indexed (X)
0156 D9 00 01        CMP     aaaa,Y      ;absolute indexed (Y)
0159 C5 12           CMP     b           ;zero page
015B CD 00 01        CMP     aaaa        ;absolute
015E
015E                                     ;compare with X
015E E0 12           CPX     #b          ;immediate
0160 E4 12           CPX     b           ;zero page
0162 EC 00 01        CPX     aaaa        ;absolute
0165
0165                                     ;compare with Y
0165 C0 12           CPY     #b          ;immediate
0167 C4 12           CPY     b           ;zero page
0169 CC 00 01        CPY     aaaa        ;absolute
016C
016C                                     ;decrement
016C D6 12           DEC     b,X         ;zero page indexed (X)
016E DE 00 01        DEC     aaaa,X      ;absolute indexed (X)
0171 C6 12           DEC     b           ;zero page
0173 CE 00 01        DEC     aaaa        ;absolute
0176 CA              DEX                 ;X
0177 88              DEY                 ;Y
0178
0178                                     ;exclusive or with A
0178 49 12           EOR     #b          ;immediate
017A 41 12           EOR     (b,X)       ;indexed indirect (X)
017C 51 12           EOR     (b),Y       ;indirect indexed (Y)
017E 55 12           EOR     b,X         ;zero page indexed (X)
0180 5D 00 01        EOR     aaaa,X      ;absolute indexed (X)
0183 59 00 01        EOR     aaaa,Y      ;absolute indexed (Y)
0186 45 12           EOR     b           ;zero page
0188 4D 00 01        EOR     aaaa        ;absolute
018B
018B                                     ;increment
018B F6 12           INC     b,X         ;zero page indexed (X)
018D FE 00 01        INC     aaaa,X      ;absolute indexed (X)
0190 E6 12           INC     b           ;zero page
0192 EE 00 01        INC     aaaa        ;absolute
```

```
0195 E8              INX                ;increment X
0196 C8              INY                ;increment Y
0197
0197 6C 00 01        JMP     (aaaa)     ;jump absolute indirect
019A 4C 00 01        JMP     aaaa       ;jump absolute
019D
019D 20 00 01        JSR     aaaa       ;jump to subroutine absolute
01A0
01A0                                    ;load A
01A0 A9 12           LDA     #b         ;immediate
01A2 A1 12           LDA     (b,X)      ;indexed indirect (X)
01A4 B1 12           LDA     (b),Y      ;indirect indexed (Y)
01A6 B5 12           LDA     b,X        ;zero page indexed (X)
01A8 BD 00 01        LDA     aaaa,X     ;absolute indexed (X)
01AB B9 00 01        LDA     aaaa,Y     ;absolute indexed (Y)
01AE A5 12           LDA     b          ;zero page
01B0 AD 00 01        LDA     aaaa       ;absolute
01B3
01B3                                    ;load X
01B3 A2 12           LDX     #b         ;immediate
01B5 B6 12           LDX     b,Y        ;zero page indexed (Y)
01B7 BE 00 01        LDX     aaaa,Y     ;absolute indexed (Y)
01BA A6 12           LDX     b          ;zero page
01BC AE 00 01        LDX     aaaa       ;absolute
01BF
01BF                                    ;load Y
01BF A0 12           LDY     #b         ;immediate
01C1 B4 12           LDY     b,X        ;zero page indexed (X)
01C3 BC 00 01        LDY     aaaa,X     ;absolute indexed (X)
01C6 A4 12           LDY     b          ;zero page
01C8 AC 00 01        LDY     aaaa       ;absolute
01CB
01CB                                    ;logical shift right
01CB 4A              LSR     A          ;A
01CC 56 12           LSR     b,X        ;zero page indexed (X)
01CE 5E 00 01        LSR     aaaa,X     ;absolute indexed (X)
01D1 46 12           LSR     b          ;zero page
01D3 4E 00 01        LSR     aaaa       ;absolute
01D6
01D6 EA              NOP                ;no operation
01D7
01D7                                    ;or with A
01D7 09 12           ORA     #b         ;immediate
01D9 01 12           ORA     (b,X)      ;indexed indirect (X)
01DB 11 12           ORA     (b),Y      ;indirect indexed (Y)
01DD 15 12           ORA     b,X        ;zero page indexed (X)
01DF 1D 00 01        ORA     aaaa,X     ;absolute indexed (X)
01E2 19 00 01        ORA     aaaa,Y     ;absolute indexed (Y)
01E5 05 12           ORA     b          ;zero page
01E7 0D 00 01        ORA     aaaa       ;absolute
01EA
```

```
01EA 48              PHA                ;push A onto stack
01EB 08              PHP                ;push status onto stack
01EC 68              PLA                ;pull A from stack
01ED 28              PLP                ;pull status from stack
01EE
01EE                                    ;rotate left
01EE 2A              ROL     A          ;A
01EF 36 12           ROL     b,X        ;zero page indexed (X)
01F1 3E 00 01        ROL     aaaa,X     ;absolute indexed (X)
01F4 26 12           ROL     b          ;zero page
01F6 2E 00 01        ROL     aaaa       ;absolute
01F9
01F9                                    ;rotate right
01F9 6A              ROR     A          ;A
01FA 76 12           ROR     b,X        ;zero page indexed (X)
01FC 7E 00 01        ROR     aaaa,X     ;absolute indexed (X)
01FF 66 12           ROR     b          ;zero page
0201 6E 00 01        ROR     aaaa       ;absolute
0204
0204 40              RTI                ;return from interrupt
0205 60              RTS                ;return from subroutine
0206
0206                                    ;subtract from A with carry
0206 E9 12           SBC     #b         ;immediate
0208 E1 12           SBC     (b,X)      ;indexed indirect (X)
020A F1 12           SBC     (b),Y      ;indirect indexed (Y)
020C F5 12           SBC     b,X        ;zero page indexed (X)
020E FD 00 01        SBC     aaaa,X     ;absolute indexed (X)
0211 F9 00 01        SBC     aaaa,Y     ;absolute indexed (Y)
0214 E5 12           SBC     b          ;zero page
0216 ED 00 01        SBC     aaaa       ;absolute
0219
0219 38              SEC                ;set carry
021A F8              SED                ;set decimal mode
021B 78              SEI                ;set interrupt disable
021C
021C                                    ;store A
021C 81 12           STA     (b,X)      ;indexed indirect (X)
021E 91 12           STA     (b),Y      ;indirect indexed (Y)
0220 95 12           STA     b,X        ;zero page indexed (X)
0222 9D 00 01        STA     aaaa,X     ;absolute indexed (X)
0225 99 00 01        ST      aaaa,Y     ;absolute indexed (Y)
0228 85 12           STA     b          ;zero page
022A 8D 00 01        STA     aaaa       ;absolute
022D
022D                                    ;store X
022D 96 12           STX     b,Y        ;zero page indexed (Y)
022F 86 12           STX     b          ;zero page
0231 8E 00 01        STX     aaaa       ;absolute
0234
0234                                    ;store Y
0234 94 12           STY     b,X        ;zero page indexed (X)
0236 84 12           STY     b          ;zero page
0238 8C 00 01        STY     aaaa       ;absolute
```

```
023B
023B AA               TAX                   ;transfer A to X
023C A8               TAY                   ;transfer A to Y
023D BA               TSX                   ;transfer stack to X
023E 8A               TXA                   ;transfer X to A
023F 9A               TXS                   ;transfer X to stack
0240 98               TYA                   ;transfer Y to A
0241
0241                  .END



 (end of 6502_test.txt)
```

Note: These sample programs are not intended to be functional, complete, or even correct. Their purpose is merely to provide some sample object files for learning to use picoViewer®.

Listing for 6502_sample1.txt

```
                        ;LED blinker

                porta   .EQU $0800
                ddra    .EQU $0804
                blinks  .EQU $C0              ;#blinks
                outer   .EQU $C1              ;outer delay loop
                inner   .EQU $C2              ;inner delay loop

0400                    .ORG $400
0400 A9 86      start:  LDA     #$86          ;bits 7,2,1 of port A
0402 8D 00 08           STA     porta         ; = output
0405 8D 04 08           STA     ddra          ;
0408 A9 0A              LDA     #$0A          ;10 blinks
040A 85 C0              STA     blinks        ;
040C AD 00 08 blink:    LDA     porta         ;LED ON
040F 09 01              ORA     #$01          ;
0411 8D 00 08           STA     porta         ;
0414 20 27 04           JSR     delay         ;delay
0417 AD 00 08           LDA     porta         ;
041A 29 FE              AND     #$FE          ;LED OFF
041C 8D 00 08           STA     porta         ;
041F 20 27 04           JSR     delay         ;delay
0422 C6 C0              DEC     blinks        ;dec #blinks
0424 D0 E6              BNE     blink         ;continue
0426 60                 RTS                   ;
0427                                          ;delay routine
0427 A9 20      delay:  LDA     #$20          ;set #outer loops
0429 85 C1              STA     outer         ;
042B A9 20      loop1:  LDA     #$20          ;set #inner loops
042D 85 C2              STA     inner         ;
042F C6 C2      loop2:  DEC     inner         ;dec inner
0431 D0 FC              BNE     loop2         ;continue inner
0433 C6 C1              DEC     outer         ;dec outer
0435 D0 F4              BNE     loop1         ;continue outer
0437 60                 RTS                   ;
0438                    .END
```

Listing for 6502_sample2.txt

```
                        ;16-bit Multiply

                X       .EQU    $2000           ;first
                Y       .EQU    $2002           ;second, lower product
                Z       .EQU    $2004           ;upper product

0600                    .ORG    $0600
0600 A9 00      MUL16:  LDA     #0
0602 8D 04 20           STA     Z               ;clear product
0605 8D 05 20           STA     Z+1             ;
0608 A2 10              LDX     #$10            ;set #bits
060A AD 03 20   LOOP1:  LDA     Y+1             ;get Y(low)
060D 29 01              AND     #$01            ;look at Y(0)
060F C9 01              CMP     #$01            ;
0611 F0 12              BEQ     NEXT            ;skip if 0
0613 AD 01 20           LDA     X+1             ;Z = X + Z
0616 6D 05 20           ADC     Z+1             ;
0619 8D 05 20           STA     Z+1             ;
061C AD 00 20           LDA     X               ;
061F 6D 04 20           ADC     Z               ;
0622 8D 04 20           STA     Z               ;
0625 6E 04 20   NEXT:   ROR     Z               ;rotate product
0628 6E 05 20           ROR     Z+1             ;
062B 6E 02 20           ROR     Y               ;rotate multiplier
062E 6E 03 20           ROR     Y+1             ;next bit to carry
0631 CA                 DEX                     ;do all bits
0632 D0 D6              BNE     LOOP1           ;
0634 60                 RTS                     ;back to caller
0635                    .END
```

Appendix C: ASCII Character Set

Chr	Dec	Oct	Hex	Chr	Dec	Oct	Hex	Chr	Dec	Oct	Hex	Chr	Dec	Oct	Hex
nul	0	000	00	sp	32	040	20	@	64	080	40	`	96	130	60
soh	1	001	01	!	33	041	21	A	65	081	41	a	97	131	61
stx	2	002	02	“	34	042	22	B	66	082	42	b	98	132	62
etx	3	003	03	#	35	043	23	C	67	083	43	c	99	133	63
eot	4	004	04	$	36	044	24	D	68	084	44	d	100	134	64
enq	5	005	05	%	37	045	25	E	69	085	45	e	101	135	65
ack	6	006	06	&	38	046	26	F	70	086	46	f	102	136	66
bel	7	007	07	‘	39	047	27	G	71	087	47	g	103	137	67
bs	8	010	08	(	40	050	28	H	72	100	48	h	104	140	68
ht	9	011	09	)	41	051	29	I	73	101	49	i	105	141	69
nl	10	012	0A	*	42	052	2A	J	74	102	4A	j	106	142	6A
vt	11	013	0B	+	43	053	2B	K	75	103	4B	k	107	143	6B
np	12	014	0C	,	44	054	2C	L	76	104	4C	l	108	144	6C
cr	13	015	0D	-	45	055	2D	M	77	105	4D	m	109	145	6D
so	14	016	0E	.	46	056	2E	N	78	106	4E	n	110	146	6E
si	15	017	0F	/	47	057	2F	O	79	107	4F	o	111	147	6F
dle	16	020	10	0	48	060	30	P	80	110	50	p	112	150	70
dc1	17	021	11	1	49	061	31	Q	81	111	51	q	113	151	71
dc2	18	022	12	2	50	062	32	R	82	112	52	r	114	152	72
dc3	19	023	13	3	51	063	33	S	83	113	53	s	115	153	73
dc5	20	024	14	4	52	064	34	T	84	114	54	t	116	154	74
nak	21	025	15	5	53	065	35	U	85	115	55	u	117	155	75
syn	22	026	16	6	54	066	36	V	86	116	56	v	118	156	76
etb	23	027	17	7	55	067	37	W	87	117	57	w	119	157	77
can	24	030	18	8	56	070	38	X	88	120	58	x	120	160	78
em	25	031	19	9	57	071	39	Y	89	121	59	y	121	161	79
sub	26	032	1A	:	58	072	3A	Z	90	122	5A	z	122	162	7A
esc	27	033	1B	;	59	073	3B	[	91	123	5B	{	123	163	7B
fs	28	034	1C	<	60	074	3C	\	92	124	5C	|	124	164	7C
gs	29	035	1D	=	61	075	3D	]	93	125	5D	}	125	165	7D
rs	30	036	1E	>	62	076	3E	^	94	126	5E	~	126	166	7E
us	31	037	1F	?	63	077	3F	_	95	127	5F	del	127	167	7F

Picodoc

CORPORATION

Enterprise Knowledge Engineering

www.picodoc.com
Oakville, Ontario
Canada

PV-RE-012104