PICDIS-LITE

Disassembler
for
Microchip PIC
Microcontrollers

Joe's Cat Software
Richmond, BC, Canada

Table of Contents

What Can PICDIS-LITE Do For You?

PICDIS-LITE is a disassembler for Microchip PIC10, 12 and 16 micro controllers. It is free for Student or Hobbyist use as a command line DOS or Linux tool to help diagnose Microchip's HEX format files or HEX files of other formats.

Differences from the full version are: there are no Hexadecimal file Tool options, it is limited to 2048 words, a few PIC10, 12 and 16 chips, and no 14 or 17 chips.

What Is Included In This PICDIS-LITE Package

Inside the ***PICDISL.ZIP*** file, you should find these files:

FILE	DESCRIPTION
PICDISL.PDF	"**This file**" explains how to install and use **PICDIS-LITE**.
PICDISL.EXE	Executable command line program for DOS.
picdisl	Executable binary program for Linux.
TEST.HEX	Sample HEX format file to test **PICDIS-LITE**.

Operating System And Memory Requirements

o **IBM Personal Computer or Compatible**

o **DOS**
DOS version 3.3 or later, 8086 or better CPU, tested on 640K RAM.

o **Windows 3.x, or Windows**
Runs as a DOS program in a DOS box/shell.

o **Linux Kernel 2.4.4 (Intel 386 Compatible) and glib5 (or higher)**
Tested on Mandrake 8.0, expect to work correctly on RedHat 7.2, SuSE 7.2 and Slackware 8.0 due to common Linux Kernel 2.4

Conditions Of Use For PICDIS-LITE

You are free to use, copy or distribute PICDIS-LITE if:
1) NO FEE IS CHARGED FOR USE, COPYING OR DISTRIBUTION.
2) IT IS NOT MODIFIED IN ANY WAY.
3) PICDIS-LITE IS USED FOR HOBBYIST OR STUDENT USE ONLY.

If you are a business or professional user:
PICDIS-LITE is provided as a sample to see if PICDIS is a program you want to purchase. After a reasonable period of 10 days or so, it is expected you go purchase PICDIS at ***http://www.JoesCat.com/micro/picchip.htm***

PICDIS-LITE is freeware. No support or warranty of any kind is given or implied.

DOS, Installing Files Onto Your Harddrive

Installing the Linux version of PICDIS-LITE is described on the next page.
These steps help you install the DOS version of PICDIS-LITE on your harddrive.

1) Make a floppy disk backup copy of your ***PICDISL.ZIP*** file.
Label the floppy disk and put it in a safe place.

2) UNZIP ***PICDISL.ZIP*** file into your MPLAB directory.
If you are not sure where to look, MPLAB is usually installed by default at *C:\MPLAB* or *C:\Program Files\MPLAB*.
Make sure you install these files:
PICDISL.EXE, TEST.HEX

Note: If you prefer not to install <PIDISL.EXE> in your MPLAB directory you can choose another directory. Good choices are directories in your computer's PATH. Type PATH at the command line to display possible directories you may want to use to install <PICDISL.EXE> in. Some suggestions are your *C:\WINDOWS* or *C:\DOS* directory.
Choose or create a suitable directory for your sample <TEST.HEX> file.

Linux, Installing PICDIS-LITE On Your Harddrive

1) Make a floppy disk backup copy of your ***PICDISL.ZIP*** file.
Label the floppy disk and put it in a safe place.

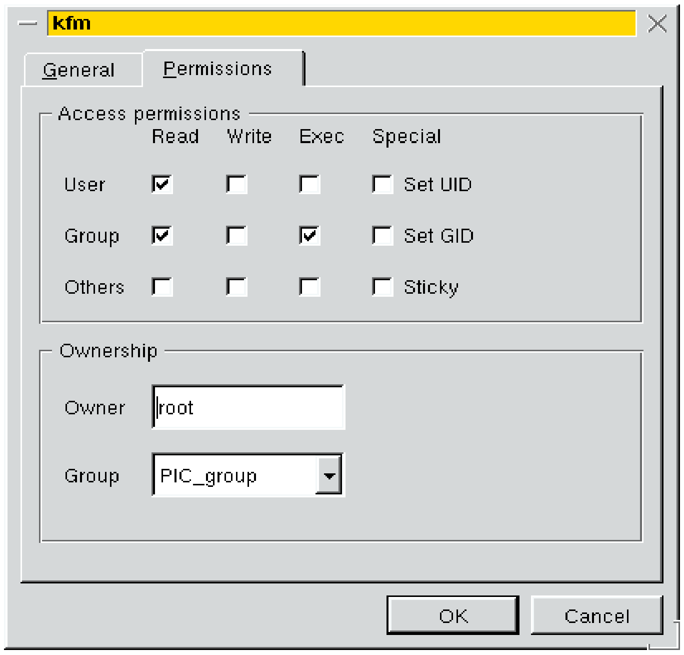
2) If you already have PIC Chip tools located elsewhere on your harddrive such as *gpasm* (GNU PIC Assembler), then you may want to install *picdisl* there too. If you have no special location for <picdisl> then we suggest you install it in */usr/bin* for group users or */usr/local/bin* for single users.

You will need *root* Administrator access to place it in */usr/....* If you do not have permission, then you can only save it and run it from your */home/...*
If you can only install <picdisl> in a */home/...* directory, then you will only be able to run it as *./picdisl* instead of *picdisl* (note the extra ./ in front).

3) If you are not a *root Administrator* then ignore or skip step 3.

If you are an *Administrator* then set the user or group permissions for PICDIS-LITE.

For an office with a networked server shared between users, if you are an *Administrator*, then you may want to set the *Owner* or *Group* for PICDIS-LITE to the correct engineering or user group.



Access permissions

	Read	Write	Exec	Special
User	✓			Set UID
Group	✓		✓	Set GID
Others				Sticky

Ownership

Owner: root

Group: PIC_group

4) Now install <TEST.HEX> and <PICDISL.PDF> in your */home/...* directory.

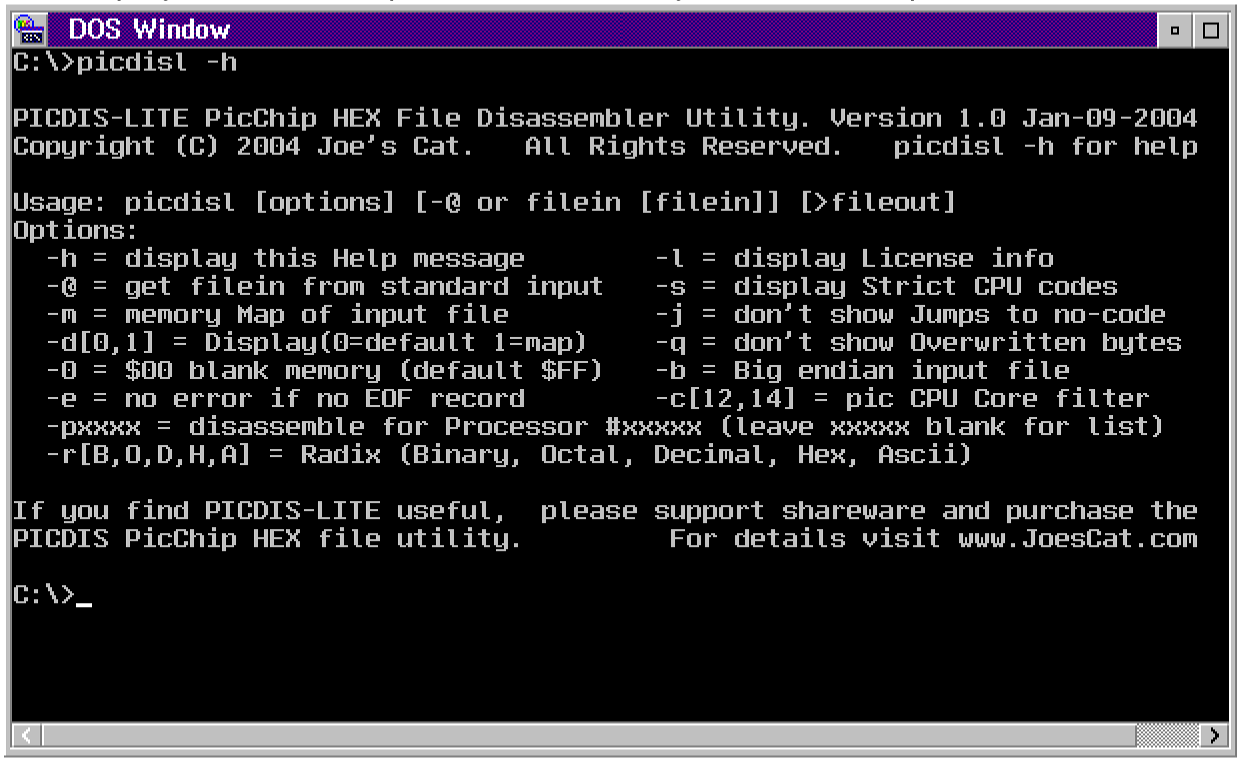
Note: If you placed <picdisl> in your */home/...* directory, then you need to use ***./picdisl*** for all commands.

Using PICDIS-LITE From The Command Line

If you are running Windows, please open up a DOS shell.
If you are running Linux from a Graphical User Interface (GUI, such as GNOME or KDE), please open up a terminal window.

For DOS, please change directory to where <PICDISL.EXE> is located, or type the <drive:\path\PICDISL.EXE> to run PICDIS-LITE. For Linux, if you did not install *picdisl* in */usr/bin*, or your local directory, then use </path/picdisl> to run it.

Now display PICDIS-LITE help by typing *picdisl -h* on the command line.
This displays a list of all options available to you. See Example below:



```
C:\>picdisl -h

PICDIS-LITE PicChip HEX File Disassembler Utility. Version 1.0 Jan-09-2004
Copyright (C) 2004 Joe's Cat.   All Rights Reserved.   picdisl -h for help

Usage: picdisl [options] [-@ or filein [filein]] [>fileout]
Options:
  -h = display this Help message          -l = display License info
  -@ = get filein from standard input     -s = display Strict CPU codes
  -m = memory Map of input file           -j = don't show Jumps to no-code
  -d[0,1] = Display(0=default 1=map)      -q = don't show Overwritten bytes
  -0 = $00 blank memory (default $FF)     -b = Big endian input file
  -e = no error if no EOF record          -c[12,14] = pic CPU Core filter
  -pxxxx = disassemble for Processor #xxxxx (leave xxxxx blank for list)
  -r[B,O,D,H,A] = Radix (Binary, Octal, Decimal, Hex, Ascii)

If you find PICDIS-LITE useful,  please support shareware and purchase the
PICDIS PicChip HEX file utility.         For details visit www.JoesCat.com

C:\>_
```

The screen above should look similar for DOS, Linux or DOS-Windows screens.

PICDIS-LITE can accept one or several options followed by one or several input files, and then everything is figured out, and then sent out to the standard output, or by redirection using "***>out_file.lst***" to an output file.

For a better explanation of each option shown above, here is a list of the command line options in better detail (see next pages):

OPTION	DESCRIPTION OF COMMAND LINE OPTION
H ?	***PICDISL -h*** This option shows a summary help on how to use PICDIS-LITE. To use PICDIS-LITE, you insert options first, then a list of input files, and then an output file (if you are not printing to display). Here are some examples: Merge in_file1 and in_file2 and then display it to screen. ***picdisl in_file1.hxl in_file2.hxh*** Get a file as output from another program, then save results. ***cat in_file1.hex | picdisl -@ >out_file.lst*** Map a data file (as 8 bit). ***picdisl -d1 -peeprom8 in_file.hex >out_file.map*** Printing your latest PIC HEX file to verify that it is correct. ***picdisl -m in_file.s19 >prn:***
L	***PICDISL -l*** PICDIS-LITE is free for Students, Hobbyists, or free distribution. If you are a professional, it is expected you will use PICDIS-LITE for evaluation (approximately 10 days) before buying PICDIS.
@	***[..... |] PICDISL -@ [<in_file.hex] [>out_file.lst]*** PICDIS-LITE will get the HEX input file from standard input and not from a file on disk. This is useful if you use other programs to translate or filter the data before giving it to PICDIS-LITE. If you use this option, you can only have 1 input file at a time and it cannot be intellec *.HXH or *.HXL since PICDIS-LITE will not know if it is the high or low byte file since the file names are not included in this method of input. Here are several examples: normal ***picdisl in_file.hex >out_file.lst*** from std input: ***picdisl -@ <in_file.hex >out_file.lst*** piped DOS: ***type in_file.hex | picdisl -@ >out_file.lst*** piped Linux: ***cat in_file.hex | picdisl -@ >out_file.lst***

OPTION	DESCRIPTION OF COMMAND LINE OPTION
S	***PICDISL -s in_file.hex [>out_file.lst]*** If you use this option, PICDIS-LITE will display the CPU codes as per data sheets. By default, PICDIS-LITE displays the easier to remember Mnemonics for specific CPU codes, therefore making the flow of assembler code easier to read. Sample with Mnemonic — Same Sample using Strict Code ***0743 skpz*** — ***0743 btfss status,z*** Read Chapter "*Strict Code Rules, Mnemonics, Option [-s]*". "*Appendix B: Special Mnemonics*" has a list of codes affected.
M	***PICDISL -m in_file.hex [>out_file.lst]*** This option makes PICDIS-LITE display a memory map. See Chapter "*Show Me A Map, The Memory Map Option [-m]*".
J	***PICDISL -j in_file.hex [>out_file.lst]*** This option is only useful for options *[-d0]* and *[-m]*. Many PIC Chips contain paged program space. This information is recognized in the assembler code, but is lost in the HEX format code. PICDIS-LITE will put all jumps pointing to program page 0. This option only removes references to empty value jump locations to make the code appear cleaner with less clutter. It is up to you to identify if a jump points to the correct program page and then modify the jump labels accordingly. Read more in chapter "*Jump Labels, Pages, And The [-j] Option*"
D0 D1	***PICDISL -d0 in_file.hex [>out_file.lst]*** ***PICDISL -d1 in_file.hex [>out_file.map]*** By default PICDIS-LITE uses *[-d0]* to display a disassembled listing of the HEX code specified. Option *[-d1]* may be useful if you wish to see output in a data format (useful for ROMs). "*Appendix D: Display Formats*" shows example outputs.

OPTION	DESCRIPTION OF COMMAND LINE OPTION
Q	***PICDISL -q in_file.hex [>out_file.lst]*** This option is useful if you are merging 2 or more files together. Sometimes you have bytes overwritten, therefore this removes all overwritten bytes from the resulting output file. Note: no preference is given to first or last byte since one or the other byte may be wrong, therefore, the data byte is erased. See example at "*Why Show e or o Attached? And Option [-q]*"
0 (zero)	***PICDISL -0 in_file.hex [>out_file.lst]*** By default, PICDIS-LITE expects memory to be erased as FFh. If you use the *[-0]* option, then PICDIS-LITE treats memory as erased to 00h instead of FFh. There is an example in "*Erasing Memory To 0bits, Option [-0]*".
B	***PICDISL -b in_file.hex [>out_file.lst]*** PICDIS-LITE is designed to work with 16 bit word data by default but sometimes the information is Big Endian or Little Endian. This option swaps the high and low bytes before processing. This option has no effect if you use the *[-peeprom8]* option. "*Big Endian, Little Endian, And The [-b] Option*" has examples.
E	***PICDISL -e in_file.hex [>out_file.lst]*** PICDIS-LITE verifies the HEX file contains an End-Of-File (EOF) record for the HEX formats that are supposed to contain one, otherwise the file is considered truncated and incomplete. Using this option tells PICDIS-LITE to ignore the fact that the EOF record is missing, therefore process what it has left. "*Why Show p Attached? And Option [-e]*" has an example.
O	***PICDISL -oout_file.lst in_file.hex*** This allows you to enter the output file name on command line for programs that save to file and do not use standard output.

OPTION	DESCRIPTION OF COMMAND LINE OPTION
C12 C14	***PICDISL -c12 in_file.hex [>out_file.lst]*** ***PICDISL -c14 in_file.hex [>out_file.lst]*** PICDIS-LITE is designed to be a PIC Chip tool so this option verifies that the input HEX file(s) are 12 or 14 bits wide only by making sure the upper 4 or 2 bits are always 0 bits. Data words larger than 12 or 14 bits trigger an error. The default for this option is auto-detect since PICDIS-LITE is expected to read HEX files for 8 bit or 16 bit ROMS too. Chapter "*Core Filter [-c]*" explains this in more detail.
Pxxxx	***PICDISL -pxxxx in_file.hex [>out_file.lst]*** If you choose a processor, then PICDIS-LITE knows which processor rules to apply. Memory boundaries are set according to *[-pxxxx]*, and for disassember mode, the correct register labels are used. ***PICDISL -p*** If *[-p]* is used by itself, then PICDIS-LITE displays it's list of known processor models which you can choose from. Read "*How Is PIC Code Displayed & Options [-p][-d0][-d1]*". "*Appendix F: Processor Filter [-p]*" lists known processors.
RB RO RD RH RA	***PICDISL -rB in_file.hex [>out_file.lst]*** ***PICDISL -rO in_file.hex [>out_file.lst]*** ***PICDISL -rD in_file.hex [>out_file.lst]*** ***PICDISL -rH in_file.hex [>out_file.lst]*** ***PICDISL -rA in_file.hex [>out_file.lst]*** PICDIS-LITE shows data values in hexadecimal by default. It can be directed to display data values in *[-rB]* Binary, *[-rO]* Octal, *[-rD]* Decimal, *[-rH]* Hexadecimals, or *[-rA]* ASCII. This is only useful for HEX file disassembly mode. Note: "*Appendix C: ANSI 'C' Escape Sequences*" shows special codes which may be displayed for certain ASCII values.

How Are PIC Ports And Registers Displayed?

Due to the banked register design of the PIC Chip, there is a certain amount of lost information when viewing PIC hexadecimal files.

PICDIS-LITE shows only the 1st possible bank of registers for the processor you choose. PICDIS-LITE does not make assumptions for which port is addressed and will display a list of possible ports. Two reasons for displaying all ports are:

1st is because it makes it easier to debug and/or understand what your code may possibly be doing. For example:

```
1283      bcf    STATUS,5
0185      clrf   PORTA/TRISA  <- PORTA?
1683      bsf    STATUS,5
0185      clrf   PORTA/TRISA  <- TRISA?
```

0185 is the same for both above, yet may seem more apparent if PORTA and TRISA are both shown as PICDIS-LITE output.

2nd and more important, as better/newer PIC Chips are introduced, you may note information on older/abandoned chips may no longer be available from your updated development toolkit or from Microchip's website. PICDIS-LITE is self contained, so this information won't be lost, therefore translating obsolete work into better/current PIC Chips is easier.

Sample PIC16C61 PICDIS-LITE Output

```
0080h     movwf     INDF
0081h     movwf     TMR0/OPTION_REG
0082h     movwf     PCL
0083h     movwf     STATUS
0084h     movwf     FSR
0085h     movwf     PORTA/TRISA
0086h     movwf     PORTB/TRISB
0087h     db        0x00,0x87
0088h     db        0x00,0x88
0089h     db        0x00,0x89
008ah     movwf     PCLATH
008bh     movwf     INTCON
008ch     movwf     0x0c
...       ...       ...
00afh     movwf     0x2f
00b0h     db        0x00,0xb0
```

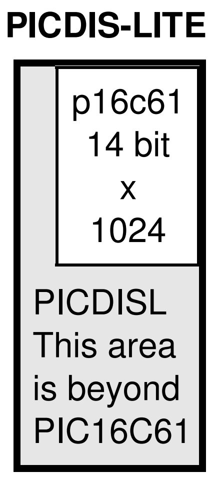
REGISTER MAP

00h	INDF	INDF	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h			87h
08h			88h
09h			89h
0ah	PCLATH	PCLATH	8ah
0bh	INTCON	INTCON	8bh
0ch	register	...	8ch
...	register	...	...
2fh	register	...	afh
30h			b0h

How Is PIC Code Displayed & Options *[-p][-d0][-d1]*

If you select a particular processor using option *[-p]*, then PICDIS-LITE knows what rules to apply to incoming HEX files.

For example, if you select *[-p16c16]*, then PICDIS-LITE knows all code must fit in address range 0000h...03FFh and be 14 bits wide, otherwise it does not fit in a PIC16C61 chip. PICDIS-LITE will not process files outside *[-p16c61]* boundaries. In disassembly *[-d0]* mode, PICDIS-LITE displays correct *[-p16c61]* register names, and knows certain registers do not exist, so it displays instructions affecting non-existing registers as data statements to get your attention (for example: "*Sample PIC16C61 PICDIS-LITE Output*" on previous page has 0087h, 0088h, 0089h and 00b0h shown as db and not movwf instructions).



Substituting code instructions with db statements is not a bug, it is a design feature of display mode *[-d0]* which you may find useful for debugging if you compile code using alpha, beta, or experimental compilers or assemblers but can't explain why some things may not be happening as expected. If you do not like this default disassembly mode and prefer to see instructions in it's pure form, then add option *[-s]* (see chapter "*Strict Code Rules, Mnemonics, Option [-s]*").

Below are example commands to disassemble *[-d0]* input HEX files:

```
picdisl  -d0  -p16c61  in_file.hex  [>out_file.lst]
picdisl  -p16c554  -s  in_file1.hex  in_file2.hex  in_file3.hex
```

Below are example commands to map *[-d1]* input HEX files:

```
picdisl  -d1  -p10f200  in_file1.hex  in_file2.hex  [>out_file.map]
picdisl  -p16c622  -d1  in_file.hex  [>out_file.map]
```

"*Appendix D: Display Formats*" shows detailed example *[-d0]* and *[-d1]* outputs.
"*Appendix F: Processor Filter [-p]*" has a list of PICDIS-LITE known processors.

Erasing Memory To 0bits, Option *[-0]*

Many ROMs and programmable parts (including PIC Chips) are set to 1bits when erased, then bits are "*burned*" to 0 to program the data into the part.

There are some odd ROMs and parts manufactured that are erased to 0bits. PICDIS-LITE has the *[-0]* option to initialize its' memory to 0bits before reading HEX format files for processing. This allows PICDIS-LITE to emulate those types of ROMs when loading HEX files.

Below is an example where PICDIS-LITE loads a byte that contains 0x55 and then merges more data over the same location (merging 0xC3 then 0x99):

OPTION *[-0]* **1 to 0 or 0 to 1**	**INIT MEM**	**LOAD 0x55**	**LOAD 0xC3**	**LOAD 0x99**
1bit burn to 0bit, default	0xFF	0x55	0x41	0x01
0bit burn to 1bit, use *[-0]*	0x00	0x55	0xD7	0xDF

Big Endian, Little Endian, And The *[-b]* Option

PICDIS-LITE is designed mainly to read 16bit hexadecimal files for the PIC Chip, but occasionally you may find non-standard files written for other programmers or other processors. This option allows you to switch the high and low byte of input hexadecimal files.

Example input file to demonstrate option *[-b]*:
<HEXFILE.S19>

```
    S10B00004142434445464748D0
    S9030000FC
```

Resulting outputs without and with the *[-b]* option:

picdisl -ra HEXFILE.S19

```
0000  4241        db   'B','A'
0001  4443        db   'D','C'
0002  4645        db   'F','E'
0003  4847        db   'H','G'
```

picdisl -ra -b HEXFILE.S19

```
0000  4142        db   'A','B'
0001  4344        db   'C','D'
0002  4546        db   'E','F'
0003  4748        db   'G','H'
```

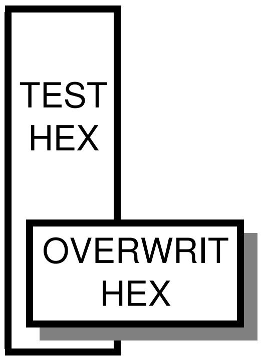
Why Show o or e Attached? And Option *[-q]*

PICDIS-LITE is designed to flag overwritten bytes by adding ***o*** or ***e*** beside the byte values that were overwritten (1 flag for high byte, 1 flag for low byte).

To demonstrate flags ***o*** (overwritten but not changed) and ***e*** (overwritten and value has changed as a result), here is <TEST.HEX> merged with another file:

<TEST.HEX>

```
        :10002000E2010D0844084C0852084F085708200800
        :0A0030004F084C084C08450848082A
        :1002000040000600C0C2700C70010092600E7027A
        :04021000040B000BD0
        :0203FE00000BF2
        :021FFE00FB0FD7
        :00000001FF
```



<OVERWRIT.HEX>

```
        :04021000030C000BD0
        :00000001FF
```

Partial Result of: ***picdisl TEST.HEX OVERWRIT.HEX [>out_file.lst]***

```
etc... 0107   02e7          decfsz  0x07,f
       0108  e0800e         retlw   0x00
       0109  o0b00o         goto    j100
```

o The value o0b shows the high byte value 0Bh was overwritten with 0Bh but nothing was modified during the overwrite. The value 00o shows the low byte value of 00h was overwritten with the same value of 00h again, therefore, nothing changed again. The o flag signals overwritten bytes.

e The e value in e08 shows the high byte was first loaded with one value (0Bh first) and then it got overwritten with another value (0Ch next) that changed the result to 08h. The e shows the low byte (04h first) was then overwritten and changed to 00h by another value (03h).
Read "*Erasing Memory To 0bits, Option [-0]*" to explain result of 0800h.

Overwritten bytes are a warning that there is something wrong with your hexadecimal file(s) unless you are merging files/bytes on purpose.
If you want to erase overwritten bytes, use the *[-q]* option to remove those bytes.

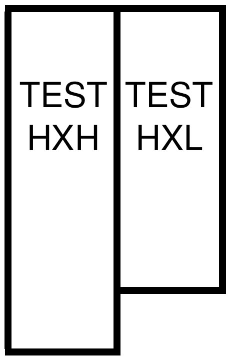
Why Show p Attached? And Option *[-e]*

One of PICDIS-LITE's duties is to recover/merge what you can of abandoned, damaged or corrupted files. All that may be left for you to work with is a partial hexadecimal file. Normally, PICDIS-LITE complains about an incomplete file if it is missing the End-Of-File marker (EOF).

To rescue what you have left (if there is no EOF), then you add option *[-e]* to the command. Below is a sample where <TEST.HXH> is complete but <TEST.HXL> is damaged (Intellec 8 bit format is made of 2 files, 1 = high bytes, 1 = low bytes).

<TEST.HXH> (high bytes of 16 bit words)

```
    :0D0010000108080808080808080808080882
    :0A01000000000C00000900020B0BC8
    :0101FF000BF4
    :010FFF000FE2
    :00000001FF
```



<TEST.HXL> (low bytes of 16 bit words)

```
    :0D001000E20D444C524F57204F4C4C4548D8
    :0A01000040060C27C71026E7040094
```

Partial Result of: ***picdisl -e TEST.HXH TEST.HXL [>out_file.lst]***

```
etc...  0108   0b04              goto      j104
        0109   0b00               goto     j100

        01ff                     org       0x01ff
        01ff   0bffp             db        0x0b,?

        0fff   0fffp             db        0x0f,?
```

As seen above, the *[-e]* option allows PICDIS-LITE to ignore the missing EOF in file <TEST.HXL> and output results of whatever is left of your file(s).

If you look at the results, you may see that words at locations 01FFh and 0FFFh are incomplete. Location 01FFh holds the high byte 0bh while the low byte is still blank, so PICDIS-LITE places a p there. Location 0FFFh is similar in results.

If it is possible to program this into a PIC Chip, then it would be seen as 0b(blank) or 0bffh while location 0FFFh would be seen as 0f(blank) or 0fffh.

Show Me A Map, The Memory Map Option *[-m]*

This option is similar to the memory map option used for assembly or compiling of your programs using your development tools, however, if you have to work with debugging, corrupted files, splicing information, reconstruction, recovering and/or rescue of legacy or abandoned data, then you may find this option helpful.

If you read previous chapters "*Why Show o or e Attached? And Option [-q]*" or "*Why Show p Attached? And Option [-e]*", you will note PICDIS-LITE shows the same useful information within the memory map.

If PICDIS-LITE is used as a disassembler, the memory map option *[-m]* shows possible jump faults as per chapter "*Jump Labels, Pages, And The [-j] Option*".

The default is for PICDIS-LITE to display memory maps in 16 bit word format, but if you use the *[-m]* option with the *[-peeprom8]* option, PICDIS-LITE displays a memory map in 8 bit format (useful for ROMs or 8 bit processors).

Below is a sample map displaying several possible errors you may encounter if you are repairing, merging, splicing, recovering, reformatting hexadecimal files:

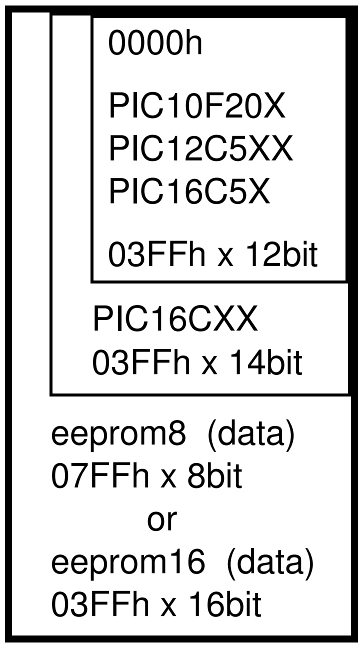
```
Memory map (X=Used, -=Unused)
0000 : XX--j-XXXoepXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
```

(j) Jump label error. This error only shows if you are using the disassembler or specify a processor [-pxxxx], otherwise the file is considered a data HEX file. All jumps will point to the 1st program bank due to the limited nature of the hexadecimal code. You may take this to be a true error or simply a warning. Please read chapter "*Jump Labels, Pages, And The [-j] Option*" for more info.

(o e) Over-burn error. This error occurs if more than one byte or word writes over the same address. This may happen if manipulating or merging files. Chapter "*Why Show o or e Attached? And Option [-q]*" has more information.

(p) Partial word error. Read chapter "*Why Show p Attached? And Option [-e]*".

Core Filter Option *[-c]*



PICDIS-LITE auto-detects between 12, 14 and 16 bit code or data by testing bits 15,14,13,12 for 1 bits while loading the input hexadecimal file (or files).

Using the *[-c]* option allows you to force PICDIS-LITE to only accept 12 bit or 14 bit hexadecimal code by testing the upper bits and rejecting the input file if 1 bits are found in bits 15 & 14 and/or bits 13 & 12.

Notes: PICDIS-LITE does PIC16C5X & PIC16CXX cores only. All input files are limited to 0000h..03FFh x 16bit words or 0000h..07FFh x 8bit bytes in size.

Examples below:

This only allows 2K x 12bit hex code that would fit within a 12 bit PIC16C5X type micro controller. If 14 or 16 bit code is found, then PICDIS-LITE stops and returns an error code because the input file is not 12 bit wide code. If the input file is not within 0000h..03FFh, then it returns an error since the file does not fit within PICDIS-LITE's 2K memory.

```
picdisl  -c12  in_file.hex  [>out_file.lst]
```

This example allows 2K x 14bit hex code that would fit within a 14 bit PIC16CXX type micro controller. 12 bit code will be translated incorrectly as 14 bit code. If the input file is not within 0000h..03FFh, then it returns an error since the file does not fit within PICDIS-LITE's 2K memory limit.

```
picdisl  -c14  in_file.hex  [>out_file.lst]
```

Here, the *[-c]* option is ignored since the *[-p]* option describes the CPU core.

```
picdisl  -p16c54  -c12  in_file.hex  [>out_file.lst]
```

Strict Code Rules, Mnemonics, Option *[-s]*

By default, PICDIS-LITE relaxes the rules of decoding hexadecimal files by using easier to understand mnemonics as shown in Microchip's first PicStart Kits.

Use the *[-s]* option if you prefer absolute PIC instructions, or your development tools do not know the mnemonics ("*Appendix B: Special Mnemonics*").

Here is an example to show the differences when using option *[-s]*:

<IN_FILE.HEX>

```
:020000040000FA
:0C0000000508013E8500031DBF0A002812
:00000001FF
```

Results Using: ***picdisl -p16c61 IN_FILE.HEX***

```
0000   0805      j000:  movfw   PORTA/TRISA
0001   3e01             addlw   0x01
0002   0085             movwf   PORTA/TRISA
0003   1d03             skpz
0004   0abf             db      0x0a,0xbf  (no register here)
0005   2800             goto    j000
```

Results Using: ***picdisl -p16c61 -s IN_FILE.HEX***

```
0000   0805      j000:  movf    PORTA/TRISA,w
0001   3e01             addlw   0x01
0002   0085             movwf   PORTA/TRISA
0003   1d03             btfss   STATUS,2
0004   0abf             incf    0x3f,f     (no register here)
0005   2800             goto    j000
```

Default is to display mnemonics like **movfw** or **skpz** as described in Appendix B.

If you prefer to see the instructions as **movf** or **btfss** exactly as described in the Microchip PIC instruction set, then use the *[-s]* option to disable the optional mnemonics.

The instruction is **incf 0x3f,f** but you are warned that the **pic16c61** processor has no register 0x3f by showing **db 0x0a,0xbf** to catch your attention.

If you use *[-pxxxx]* and prefer to see the instruction even if there is no valid register, then use the *[-s]* option to disable this feature.

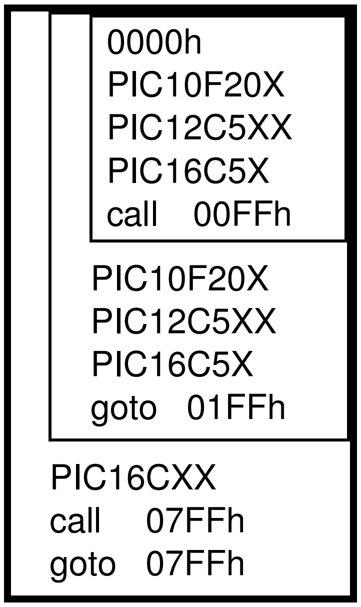
Jump Labels, Pages, And The *[-j]* Option

PICDIS-LITE puts Jump labels only in the 1st program memory page and then leaves it for the user to decide which page is correct for a particular label.

A PIC16C5X ***call*** label can span a range of 00h..0FFh, so all labels are placed in memory at 0000h..00FFh.

A PIC16C5X ***goto*** instruction can jump anywhere from 000h to 1FFh, so labels are placed at 0000h..01FFh.

14bit PIC16CXXs are capable of a ***call*** or ***goto*** from 000h to 7FFh within a program memory page, therefore all labels are placed within 0000h..07FFh.



PICDIS-LITE does **not** use "*smart*" algorithms to guess the correct program memory page to place labels since it only takes a more complicated hexadecimal file to outsmart the algorithms and then give out wrong label data. The following two pages shows examples why.

The 1st example shows what could easily be a good file for PICDIS-LITE putting the labels in the *correct* locations of j200 and j202, however, the 2nd page shows an example why it would be difficult for PICDIS-LITE to label a listing correctly using smart algorithms.

To avoid possible assumptions or errors, no exceptions are made, and all labels are put in the 1st program pages whether the program is simple or complicated and whether there is code there or not.

Now, in terms of the *[-j]* option, sometimes a program may be disassembled and contain jump labels to nowhere. This likely can happen where little of a program exists in the 1st page (such as the example on the next page). The *[-j]* option should clear up a listing by removing non-existent program jump label locations from the disassembly.

Note: The *[-j]* option is only useful for disassembly *[-d0]*, or the map *[-m]* option.

PICDIS-LITE could generate "*smart*" labels j000 & j002 as j200 & j202, but this is not done to stop false assumptions later (see next page for reason).

Example PIC16C56 Program Listing:

```
03FF                org   0x3FF
03FF 0BFF           goto  start
01FF                org   0x1FF
01FF 05A3 start bsf   STATUS,PA0
0200 0902 loop  call  xyz
0201 0A00       goto  loop
0202 0800 xyz   retlw 0
```

Resulting HEX File:

```
<IN_FILE.HEX>
:0203FE00A30555
:06040000020900A0008D9
:0207FE00FF0BEF
:021FFE00FF0FD3
:00000001FF
```

Resulting Disassembly Using: ***picdisl -p16c56 IN_FILE.HEX***

```
0000 pffffp       j000:    db       ?,?

0002                       org      0x0002
0002 pffffp       j002:    db       ?,?

01ff                       org      0x01ff
01ff   05a3       j1ff:    bsf      STATUS,5
0200   0902                call     j002
0201   0a00                goto     j000
0202   0800                retlw    0x00

03ff                       org      0x03ff
03ff   0bff                goto     j1ff

0fff   0fff                __fuses  0x0fff
```

o Insufficient data means ***call*** and ***goto*** labels are placed on the 1st program memory page (even if it seems obvious to go elsewhere).

o The "*p*" means this is a partial byte. The HEX file is missing bytes here to complete word value. This word is blank since there is a "*p*" for the high byte and one for the low byte. The only reason you see this address is because there is a jump label here due to program code pointing here (see ***call j002*** and ***goto j000*** in disassembly).

o Using the *[-j]* option, PICDIS-LITE can hide the j000 and j002 labels since there is NO program code at 0000h or at 0002h.
Example: ***picdisl -j -p16c56 IN_FILE.HEX***

This example is ***very*** complicated for a smart labeling routine, so to keep consistent, PICDIS-LITE, only puts labels in the 1st program page.

Example PIC16C57 Program Listing:

```
07FF                  org    0x7FF
07FF  0A0E            goto   start
000E                  org    0x0E
000E  05A3  start  bsf    STATUS,PA0
000F  0910            call   xyz1
0010  0910            call   xyz2
0011  0910            call   xyz3
0012  0A0E            goto   start
0210                  org    0x210
0210  05C3  xyz1   bsf    STATUS,PA1
0211  0800            retlw  0
0610                  org    0x610
0610  04A3  xyz2   bcf    STATUS,PA0
0611  0800            retlw  0
0410                  org    0x410
0410  04C3  xyz3   bcf    STATUS,PA1
0411  0800            retlw  0
```

Resulting HEX File:

```
<IN_FILE.HEX>
:04001C00A30510091F
:06002000100910090E0A90
:04042000C305000808
:04082000C304000805
:040C2000A304000821
:020FFE000E0AD9
:021FFE00FF0FD3
:00000001FF
```

Resulting Disassembly Using: ***picdisl -p16c57 IN_FILE.HEX***

```
000e              org    0x000e
000e  05a3  j00e:  bsf    STATUS,5
000f  0910         call   j010
0010  0910  j010:  call   j010
0011  0910         call   j010
0012  0a0e         goto   j00e

0210              org    0x0210
0210  05c3         bsf    STATUS,6
0211  0800         retlw  0x00

0410              org    0x0410
0410  04c3         bcf    STATUS,6
0411  0800         retlw  0x00

0610              org    0x0610
0610  04a3         bcf    STATUS,5
0611  0800         retlw  0x00

07ff              org    0x07ff
07ff  0a0e         goto   j00e
```

This program is too complicated to have the labels placed in the correct locations.

Not enough data in the HEX file means ***call*** and ***goto*** labels are placed on the 1st program memory page (even if it seems obvious to go elsewhere).

Correct label locations if HEX files contained **more** information.

Error Messages

PICDIS-LITE recognizes and reports errors when found.
"*Appendix E: Error Messages*" contains a description of possible error codes.

The DOS version of PICDIS-LITE reports text errors to standard output and sends an error value back to the parent program. If the error value returned is zero, then no errors were found while a non-zero value means an error was found.

The Linux version of PICDIS-LITE reports text errors to standard **error** output which is typical of Linux type programs.
Error values are returned to the parent program to aid in pipes and script programs, zero means okay, while non-zero indicates a fault was found.

Examples:

```
DOS:      PICDISL  in_file.hex  >out_file.lst
Linux:    picdisl  in_file.hex  >out_file.lst  2>err_file.txt
```

Bug Reports And Technical Support

Much work was done into making PICDIS-LITE trouble free but errors and bugs may still be encountered. If you have a bug or error to report, please check the PICDIS-LITE web page at ***http://www.JoesCat.com/micro/picchip.htm*** for the latest email address to report to. This email address may change from time to time due to the unfortunate reality of what is considered SPAM or JUNK email on the internet.

If you own a copy of **PICDIS**, please send in your registration along with any question(s) or request for support to the email address mentioned on the website.

Appendix A: HEX File Input Formats

PICDIS-LITE automatically recognizes several HEX format files as input files. These are the list of file types automatically recognized:

- Intel HEX 8-bit Merged Format (INHX8M).
  This is the most common format expected.

- Intellec 8-bit High (.HXH). This is the high byte part of 2 files.
  Intellec 8-bit High (.HXL). This is the low byte part of 2 files.
  This is the only file type where you need to have .HXH or .HXL as the last 4 characters in the file name, otherwise it is seen as Intel 8-bit (above).

- Intel HEX 16-bit Format.
  This allows for the 20-bit segmented address space of 16-bit processors.

- Intel HEX 32-bit Format.
  This allows for the 32-bit linear address space of 32-bit processors.

- Motorola S-record Formats.
  These are the Motorola HEX Formats (S19, S28, S37).

- EMON52, Elektor Monitor Format.

- FPC, Four Packed Code Format.

- Tektronix HEX, and Tektronix Extended HEX Formats.

- Parallax/Tech-tools SPASM.

- Atmel Generic.

Appendix B: Special Mnemonics

PICDIS-LITE shows mnemonic assembler codes by default unless the *[-s]* option for *Strict CPU Codes* is enabled. The status register is register 0x03 for 12-bit and 14-bit PIC Chips, and register 0x04 for the PIC17 family.

MNE-MONIC	STRICT CPU CODE	DESCRIPTION
clrc	bcf status,c	Clear Carry Flag
setc	bsf status,c	Set Carry Flag
clrdc	bcf status,dc	Clear Digit Carry Flag
setdc	bsf status,dc	Set Digit Carry Flag
clrz	bcf status,z	Clear Zero Flag
setz	bsf status,z	Set Zero Flag
clrov	bcf status,ov	Clear Overflow Flag
setov	bsf status,ov	Set Overflow Flag
skpnc	btfsc status,c	Skip On No Carry Flag
skpc	btfss status,c	Skip On Carry Flag
skpndc	btfsc status,dc	Skip On No Digit Carry Flag
skpdc	btfss status,dc	Skip On Digit Carry Flag
skpnz	btfsc status,z	Skip On No Zero Flag
skpz	btfss status,z	Skip On Zero Flag
skpnov	btfsc status,ov	Skip On No Overflow Flag
skpov	btfss status,ov	Skip On Overflow Flag
tstf f	movf f,1	Test File Register
movfw f	movf f,0	Move File Register To W

Note: PICDIS-LITE shows instructions to non-existing registers as DB values, unless you use option *[-s]*. For example a *PIC16C83* has **NO** register 0x30, so a "*movf 0x30,w*" will be shown as "*db 0x08,0x30*" unless you use option *[-s]*.

Appendix C: ANSI 'C' Escape Character Sequences

PICDIS-LITE shows hexadecimal values by default.
PICDIS-LITE shows ASCII values if option *[-rA]* is enabled.

Example: DOS ***PICDISL -rA in_file.hex >out_file.lst***
Linux ***picdisl -rA in_file.hex >out_file.lst 2>err_file.txt***

These are the special ANSI 'C' Escape Codes displayed for option *[-rA]*:

ESCAPE CHAR	HEX VALUE	DEC VALUE	MNE-MONIC	DESCRIPTION
\a	0x07	7	BEL	Bell (Alert) Character
\b	0x08	8	BS	Backspace Character
\t	0x09	9	HT	Horizontal Tab Character
\n	0x0A	10	LF	New-Line Character (Line Feed)
\v	0x0B	11	VT	Vertical Tab Character
\f	0x0C	12	FF	Form-Feed Character
\r	0x0D	13	CR	Carriage Return Character
\\	0x5C	92		Backslash Character
\"	0x22	34		Double Quote Character
\x__	0x__	__		Non-Printable HEX Character

Appendix D: Display Formats

PICDIS-LITE will display output in *[-d0]* or *[-d1]* formats.
The *[-d0]* option is default, you do not need to insert *[-d0]* on the command line.

Example input file to demonstrate options *[-d0]* and *[-d1]*:
<TEST.HEX>

```
     :10002000E2010D0844084C0852084F085708200800
     :0A0030004F084C084C08450848082A
     :100200000400006000C0C2700C70010092600E7027A
     :04021000040B000BD0
     :0203FE00000BF2
     :021FFE00FB0FD7
     :00000001FF
```

Results using *[-d0]* as output style: ***picdisl -d0 TEST.HEX [>out_file.lst]***

```
     0010                 org       0x0010
     0010   01e2   j010:  addwf     PCL,f
     0011   080d          retlw     0x0d
     0012   0844          retlw     0x44
     0013   084c          retlw     0x4c
     0014   0852          retlw     0x52
     0015   084f          retlw     0x4f
     0016   0857          retlw     0x57
     0017   0820          retlw     0x20
     0018   084f          retlw     0x4f
     0019   084c          retlw     0x4c
     001a   084c          retlw     0x4c
     001b   0845          retlw     0x45
     001c   0848          retlw     0x48
```

etc....

Results using *[-d1]* as output style: ***picdisl -d1 TEST.HEX [>out_file.map]***

```
0010 : 01e2 080d 0844 084c 0852 084f 0857 0820  .....D.L.R.O.W.
0018 : 084f 084c 084c 0845 0848 ____ ____ ____   .O.L.L.E.H
0100 : 0040 0006 0c0c 0027 00c7 0910 0026 02e7  .@.....'.....&..
0108 : 0b04 0b00 ____ ____ ____ ____ ____ ____  ....
01f8 : ____ ____ ____ ____ ____ ____ ____ 0b00                ..
0ff8 : ____ ____ ____ ____ ____ ____ ____ 0ffb                ..
```

Note: <"."> refers to a non-ASCII printable value, for example 0844h has 08h as a non-ASCII printable <"."> and 44h as a printable ASCII value of <"D">.

Appendix E: Error Messages

DOS PICDIS-LITE displays error descriptions to standard output.
Linux PICDIS-LITE displays error descriptions to standard error output.
The Error Code is returned to the parent program (useful for script programming).

Example: DOS ***PICDISL in_file.hex >out_file.lst***
Linux ***picdisl in_file.hex >out_file.lst 2>err_file.txt***

Note: There are references to both PICDIS-LITE and PICDIS in this list.

ERROR CODE	DESCRIPTION
101	***Error, not enough memory*** DOS PICDIS-LITE expects more RAM memory. You may have to turn off some TSR programs in DOS. Create more swap file disk space for Linux, OS/2 or Windows.
102	***Can't open file*** Cannot open or find input file named on command line. DOS version expects file names in DOS 8.3 format. Linux version is case sensitive to file names.
103	***This is not a HEX file*** PICDIS-LITE understands several HEX file formats, but this file is not a known format. See "*Appendix A: HEX File Input Formats*".
104	***Byte count does not match data length*** Record length does not match the expected byte count. The record may be corrupted or a different format.
105	***Error, address out of range*** The address for this byte is beyond the range for this type of PIC Chip. PICDIS-LITE is limited to *0000h-03FFh* words.
106	***Error, unknown record type*** This input file appears to change from one HEX format to a new HEX format. Example, changing from Intel HEX to Motorola S.

ERROR CODE	DESCRIPTION
107	***Error, expected 0000 for this record type*** Some HEX files have an End-Of-File (EOF) marker or other special markers and 0000h is expected for this record address.
108	***Error, wrong byte count for this record type*** The byte count for this record does not match the record length.
109	***Error, unknown data value*** Most HEX files use char[2] ASCII values forming 00h..FFh but this record does not appear to have valid data.
110	***Error, this CPU code bigger than 12 bits*** (option *[-c12]*) ***Error, this CPU code bigger than 14 bits*** (option *[-c14]*) PICDIS-LITE is trying to filter only 12-bit or 14-bit values but the value found is larger than expected.
111	***Bad checksum calculated*** Checksum for this record does not match the computed value. This record is corrupted or modified.
112	***Error, more code after EOF record*** The last record in this HEX format must be End-Of-File (EOF) record, but this file has more records merged after the EOF. Possibly this file was merged from 2 or more files incorrectly.
113	***Error, missing an EOF record*** The last record in this HEX format must be End-Of-File (EOF) record but this file has none. This file is possibly truncated.
114	***Wrong commandline option*** An unrecognized command line option or out-of-range value was given. Type *PICDISL -h* at the command line to list help.

Appendix F: Processor Filter *[-p]*

PICDIS-LITE filters displayed results according to the *[-pxxxx]* option by showing appropriate registers and verifying HEX data does not go out of limits.

PROCESSOR:
PICDIS-LITE verifies the HEX file fits the Processor shown. For example, ***picdisl -ppic16f84 in_file.hex [>out_file.lst]***, checks that the program fits within {0000h..03FFh}, is only 14 bits wide, and if the hex file also includes initial flash memory data, then the data fits in the 64 x 8 of flash space.
If the HEX format file does not fit this form, then it is considered an error.

PINS: Has no meaning in a HEX format file. Shown only for your information.

I/O: PICDIS-LITE will display correct I/O PORTS unless you use *[-s]* option.

PROGRAM MEMORY: This is the expected program memory for this Processor.
The HEX format file should fit within it, otherwise it is an error.

USER RAM: This is the expected RAM for this Processor.
Note: Some micro controllers do not have a full set of registers to fill a RAM bank. PICDIS-LITE displays these instructions as data statements unless you use the *[-s]* option to show the true instruction.
Note: Showing the correct RAM bank is a complicated task due to how a user programs code. PICDIS-LITE will display the 1st RAM bank if it is a RAM register, or a possible set of registers if it is an I/O register.

EEPROM/FLASH: This is the non-voltile rewrite memory size for this Processor.

(a) 12-bit processor structure suggests 2048 x 12 is allowed as maximum.
14-bit processor structure suggests 8096 x 14 is allowed as maximum.
PICDIS-LITE is limited to 2048 words.

(b) CE parts do not appear to have a way to program EEPROM via HEX files.
This is for your information. HEX files expected not to contain CE data.

(c) PICDIS-LITE expects a maximum of 64 x 8 bytes of EEPROM and will verify that the EEPROM data fits within limits.

PROCESSOR	PINS	I/O	PROGRAM MEMORY	USER RAM	EEPROM FLASH
pic10f200	6	4	256 x 12	16	
pic10f202	6	4	512 x 12	24	
pic10f204	6	4	256 x 12	16	
pic10f206	6	4	512 x 12	24	
pic10f20x	6	4	512 x 12	n/a	
pic12c508	8	6	512 x 12	25 x 8	
pic12c509	8	6	1024 x 12	41 x 8	
pic12c5xx	8	6	2048 x 12 (a)	n/a	
pic12ce518	8	6	512 x 12	25 x 8	16 x 8 (b)
pic12ce519	8	6	1024 x 12	41 x 8	16 x 8 (b)
pic12ce5xx	8	6	2048 x 12 (a)	n/a	16 x 8 (b)
pic16c505	14	11	1024 x 12	72 x 8	
pic16c52	18	12	384 x 12	25 x 8	
pic16c54	18	12	512 x 12	25 x 8	
pic16c55	28	20	512 x 12	24 x 8	
pic16c56	18	12	1024 x 12	25 x 8	
pic16c57	28	20	2048 x 12	72 x 8	
pic16c58	18	12	2048 x 12	73 x 8	
pic16c5x	n/a	n/a	2048 x 12 (a)	n/a	
pic16cxx	n/a	n/a	2048 x 14 (a)	n/a	
pic16c554	18	13	512 x 14	80 x 8	
pic16c556	18	13	1024 x 14	80 x 8	
pic16c558	18	13	2048 x 14	128 x 8	
pic16c55x	18	13	2048 x 14 (a)	n/a	
pic16c620	18	13	512 x 14	80 x 8	

PROCESSOR	PINS	I/O	PROGRAM MEMORY	USER RAM	EEPROM FLASH
pic16c621	18	13	1024 x 14	80 x 8	
pic16c622	18	13	2048 x 14	128 x 8	
pic16c62x	18	13	2048 x 14 (a)	n/a	
pic16c61	18	13	1024 x 14	36 x 8	
pic16c62	28	22	2048 x 14	128 x 8	
pic16c63	28	22	4096 x 14 (a)	192 x 8	
pic16c64	40	33	2048 x 14	128 x 8	
pic16c65	40	33	4096 x 14 (a)	192 x 8	
pic16c6x	n/a	n/a	2048 x 14 (a)	n/a	
pic16c83	18	13	512 x 14	36 x 8	64 x 8
pic16c84	18	13	1024 x 14	36 x 8	64 x 8
pic16f83	18	13	512 x 14	36 x 8	64 x 8
pic16f84	18	13	1024 x 14	68 x 8	64 x 8
pic16f8x	18	13	2048 x 14 (a)	68 x 8	64 x 8 (c)
eeprom8	n/a	n/a	4096 x 8 (a)	n/a	n/a
eeprom16	n/a	n/a	2048 x 16 (a)	n/a	n/a

Note: Normally the last *[-p]* command takes effect for processor selection, but *[-peeprom8]* and *[-peeprom16]* have priority over the other processor values to keep PICDIS-LITE compatible with MPLAB 4x and 5x.

Joe's Cat Software
11280 Westminster Hwy
Richmond, BC, Canada, V6X-1B3
http://www.JoesCat.com/micro/picchip.htm