```
No Error
OS error
Disk is full
Disk read error

File %s is not a virtual array file
WRAP
Not a virtual array file


The most probable error is the you tried to use incompatible version
of IDA. Starting from IDA 2.0 beta2 the format of virtual files was changed.

Virtual Array %
: Accessing illegal address %
WRAP
Virtual Array: Accessing illegal address


Normally this is an internal error of IDA. Please inform the author.

Virtual Array %
: Maximum number of chunks is reached (press F1)
WRAP
Virtual Array: Maximum number of chunks is reached


A virtual array can have up to ~ 400 chunks. You have reached this limit.

To increase number of allowed chunks you may increase page size for
virtual array. For this, set
:1482[configuration] file variable VPAGESIZE.
For example:

        VPAGESIZE
8192

The page size should be power of 2. The default page size is 4096.
This setting will affect new database only. Old databases will continue to
use old value of VPAGESIZE.

If you were loading New Executable Format file, you may use
'Fill segment gaps' feature (it is activated in the
Load File
 dialog
appearing at the very start) to avoid this error message.
't close file
't position in file
file read error
file write error
page is not in memory
attempt to write but declared read only
fatal error
? unknown error ?
Virtual Memory
Lexical analyser: bad digit
Lexical analyser: bad floating point constant
Lexical analyser: floating point constant conversion error
Lexical analyser: no Separator
Lexical analyser: non-terminated string
Lexical analyser: too long string
Lexical analyser: non-terminated char constant
Lexical analyser: bad character
Lexical analyser: too long ident
Lexical analyser: bad or ill-formed preprocessor command
Lexical analyser: bad #include syntax
Lexical analyser: max depth of includes is 8
Lexical analyser: can't open include file
Lexical analyser: getstr
 function is not defined
Lexical analyser: end of file reached
Lexical analyser: misplaced #else
Lexical analyser: misplaced #endif
Lexical analyser: missing #endif
Lexical analyser: bad macro usage
Lexical analyser: bad # of macro arguments
Lexical analyser: too complex macro
Compiling file
Missing while statement for a do
Missing semicolon
Missing braket
Missing colon
Syntax error near:
Missing brace
Function
' is too large (max compiled size is 64k)
've reached the maximum number of functions
Misplaced 'break'
Misplaced 'continue'
Variable
' is already defined
Variable
' is undefined
Too complex program
Internal parser error
Bad number of arguments for the function
Too many local variables
Illegal postfix/prefix operation
Function declaration is expected
Variable name is expected
Sorry, you can't initialize variables in declarations
Executing function
Zero divide
F1 for Help
Cancel
Error
Information
Enter - select default, Esc - cancel

No memory for module %

Disk Positioning error:
%s
Handler:
```

```
. Position:
Command cancelled.

Disk Write error:

You've entered a bad identifier:
Warning

Can't open for read file

Can't create file

Can't open file
' for modification:

Not Enough Disk Space. Needed:
%lu Available:
There is a possibility that IDA will run out of disk space.
Needed %
, Available %
Continue ?

Disk Space is low
Press F1 for explanations.

Can't close file:

Disk Read error:
(file position %

You've entered an invalid address.

You've entered invalid segment:
Fatal Internal Error


       This is an internal error.
   Please, inform the author about it.

Look at 'readme' file to find the address.
Messages window
```

```
File
' not found
Please press a key for macro
Macro definition cancelled.

You can't redefine the keys Alt
 and Alt-
Defining macro for the key

Already recording a macro for
Macro for
' is defined.

Not enough memory available to complete operation.

Enter string to search (case-insensitive)
WRAP
How to use the help subsystem


You can use the following keys:
  F1           this screen

[Shift-
]      help index

:1385[Ctrl-
]       IDC index
  Tab          next highlighted phrase
  Shift-Tab    prev highlighted phrase
  Enter        follow a link
  Backspace    previous help screen
  F5           zoom help window
  Esc          close help window

  arrows       one step movement
  PgUp         Page Up
  PgDn         Page Down
  Ctrl-PgUp    top of the window
  Ctrl-PgDn    bottom of the window
  Home         start of the line
  End          logical end of the line
      WRAP
```

Main Help Screen

F1 is context sensitive help. Use it when you want to know
more about IDA. If you want to know more about

[help subsystem
 press now Enter.

When you do not know which command to use, try to find out it
through menus. For
[menus
 press F10.
When you are in the menu,
F1 will explain what the current command does.
Exception:
[keyboard macros] are not available through menus.

To
[exit] IDA, press
<Quit

General Notes
. With IDA you can achieve 3 goals:
     A. get a compilable source code
     B. learn a programming tricks of an interesting program
     C. patch a file when you have no access to the source code.

 The goal A is the toughest and hardest job to do. Most of the
people try to achive goals B and C.
With IDA you can perform all 3 kinds of the job.
. IDA is not an automatical analyser of programs.
Normally it is expected that you and IDA work together.
IDA will hint you of suspicious instructions, unsolved

problems etc. It is your job to inform IDA how to proceed.
. If your goal is to get a fully compilable source code
of the file (goal A) then do not forget:
    - to use correct
[segment register values]
    - to get rid of
[unexplored bytes]
    - to get rid of
[void] marks
[converting]
      their operands to numbers or offsets.
. Keep in mind that usually it is enough to learn a program
and patch it (if you are authorized to do so

IDA does these jobs perfectly.

Have fun !

You can move around using:

arrows	one step movement
PgUp	Page Up
PgDn	Page Down
Ctrl-Left	word left
Ctrl-Right	word right
Home	start of the line
End	logical end of the line
Ctrl-Home	(or Home Home) top of the window
Ctrl-End	(or End End) bottom of the window
Ctrl-PgUp	(or Home Home Home) top of the file
Ctrl-PgDn	(or End End End) bottom of the file
Enter	jump to address/name

You can use mouse to move too.

Use Shift
arrows> or Alt-L to drop
[anchor

If you have a mouse, you can drop anchor with it too.

Double click of the mouse is equvalent to <Enter> key.
WRAP
Keyboard Macros

You can define as many macros as you want.
They are defined in the same manner as in the Borland IDE:
        - Press Alt

        - Press <
- macro name
        - Enter macro body
        - Press Alt-

All macros are lost when you

[exit] IDA.
WRAP
Anchor

Various IDA commands use an anchor. For example to select a portion
of file to output or to specify a segment to move
 you need an anchor.
To drop the anchor, you can use Alt-L command.
, more convinient way is
Shift
arrow> combination. You can drop the anchor with the mouse also:
simply click and drag.

To raise (clear) the anchor:

```
          press Alt-L
       or click the mouse anywhere in the window
```

Also you can use Ctrl-
/Ctrl-KH combinations to drop/raise the anchor.

After you've dropped the anchor, you can move freely using arrows, etc.
Any command that uses the anchor, raises it.

The anchored area is displayed with another color.

When you exit from IDA, anchor value is lost.
WRAP
File Not Found

IDA can't find the specified file.
May be you've mistyped its name?
WRAP
File Viewer

Here you can see text file.

You can move around using:

```
        arrows          one step movement
        PgUp            Page Up
        PgDn            Page Down
        Ctrl-Left       word left
        Ctrl-Right      word right
        Home            start of the line
        End             logical end of the line
        Ctrl-Home       (or Home Home) top of the window
        Ctrl-End        (or End End) bottom of the window
        Ctrl-PgUp       (or Home Home Home) top of the file
        Ctrl-PgDn       (or End End End) bottom of the file

<AskNextText>           search for a text

<JumpText>          search again
        Alt-G           goto line number
```

You can use mouse to move too.

You can edit small files using
[EditFile]
command.

To close this window, use 'WindowClose' command.
Hot key is
<WindowClose
WRAP
Cannot Create File

Probably this file exists and has
read-only attribute. Try to delete it
or clear this attribute.

Another reason is that you have too few
number in FILES
 statement of your
CONFIG.SYS file. IDA requires about 10
file handlers to be free.
WRAP
Can't close file

Please inform the author about this situation.
WRAP
Can't Open File

Probably the file does not exist. Or,
may be it resides at the other place?

Another reason is that you have too few
number in FILES
 statement of your
CONFIG.SYS file. IDA requires about 10
file handlers to be free.
WRAP
Disk Read Error

This error occurs when IDA tries to read past end of file. If it happened
when you try to disassemble a new file, the new file has bad structure.

Probably your disk contains unmarked BAD
SECTORS.

The database may be corrupted now.
WRAP
Can't Open File

Probably the file does not exist or has

read-only attribute set. Or, may be it
resides at the other place?

Another reason is that you have too few
number in FILES
 statement of your
CONFIG.SYS file. IDA requires about 10
file handlers to be free.

May be, memory is not enough
WRAP
Disk Write Error

Probably your disk contains unmarked BAD
SECTORS.

Another reason is that your DISK IS
FULL. Try to delete some unnessesary
files. It is a good style to have at
least 1MB of free disk space when you
work with IDA.

As you analyse the program, the IDA
database grows. You can compress the database using 'collect garbage'
checkbox
in the
[exit] dialog box.

The database may be corrupted now.
WRAP
Disk Positioning Error

Please inform the author if this happens.
WRAP
Not Enough Disk Space

IDA requires lot of disk memory to
disassemble files. Try to delete some
unnessesary files.

The database may get corrupted if there is not enough disk space.

As you analyse the program, the IDA
database grows. You can compress the database using 'collect garbage'
checkbox
in the
[exit] dialog box.

You should give more space for IDA to continue its work.
WRAP
Disk Space is Low

Normally when you work with IDA you should have at least 1 MB (or more)

```
of free disk space.
Please
[exit] now and provide more disk space.
WRAP
Not enough memory available to complete operation.


Try to close some windows.
If this does not help, try to unload TSRs, etc.
Shortly, make more memory.
WRAP
How to move/resize windows


Keyboard:

Arrows,Home,
,PgUp,PgDn move the window.
All keys with Shift resize the window.
Ctrl key in the combination with other keys
move/resize the window faster.

Mouse:

To move: point to title of the window and drag to the new location.

To resize: point to the right lower corner of the window and drag.
WRAP
Bad Macro


You can't redefine the keys Alt
 and Alt-

because they are used to define other macros.

Please choose another key for the macro.
WRAP
Already recording a macro for


You can't define 2 macros at the same time.
Please finish definition of the first macro
(press Alt-
) and after define the other one.
WRAP
No Memory


There is not enough memory
to initialize IDA components.
Please free some memory unloading TSRs,device drivers, etc.
WRAP
Entering String for Search


You can enter any string to search.
```

Search will be case-insensitive
Search direction: forward

If the substring is not found,
cursor will not move.
WRAP
How to Enter a Segment Value

You must enter segment in hexadecimal format or segment name. If you
enter invalid segment, IDA will warn you.

You may enter a segment register name too.
WRAP
How to Enter a Number

You can enter any '
' expression with constants. Long arithmetic will
be used for calculations.

In these expressions you can use all names which you created in your
program.
WRAP
How to Enter an Identifier

An identifier is a name which starts with a letter and contains
only letters and digits. The list of allowed characters
is specified in config file
:1482[
All names are case-sensitive.

Maximal length of a name is specified in this file too:

```
        MAX
NAME
LENGTH
120
 Maximal length of new names
```

Default is 120.

Some assemblers have lower name length limit, beware!

If you enter an illegal name, IDA will warn you.
WRAP
How to use The Notepad

You can use the following keys in the notepad:

```
 Ctrl-Enter     starts new line
 Enter          finishes the input
 Esc            cancels the input
 F1             gives some help
```

```
                                        Shift
arrow>  Select
 Ctrl-L         Search Again            Shift-Ins       Paste
 Ctrl-O         Indent Mode             Shift-Del       Cut
 Ctrl-T         Delete Word             Ctrl-Ins        Copy
 Ctrl-U         Undo                    Ctrl-Del        Clear
 Ctrl-Y         Delete Line             Ctrl-K B        Start Select
 Ctrl-Left      Word Left               Ctrl-K H        Hide Select
 Ctrl-Right     Word Right
 Ctrl-PgUp      Text Start              Ctrl-Q A        Replace
 Ctrl-PgDn      Text End                Ctrl-Q F        Search
                                        Ctrl-Q H        Delete Line Start
                                        Ctrl-Q Y        Delete Line End
```

Input containing only whitespaces is equal to an empty input.

Do not forget that you can use the clipboard.
WRAP
How to Enter an Address

You must enter address in the hexadecimal format or a location name. When you enter address in the hexadecimal (depends on the current processor, some processors use octal representation as default) format, you can omit the segment part of the address - the current segment will be used. If you enter an
invalid address, IDA will warn you. Addresses beyond the program
limits are invalid.

Also, you can enter a location name with a displacement:

```
        name+5
```

And finally you can specify a relative address:

```
        +10             10 bytes further
        -5              5 bytes backwards
```

Special addresses:

- current location   (depends on the current
[assembler

Examples:

```
        456             current segment, offset 0x456
        5E              current segment, offset 0x5E
        3000:34         segment 0x3000,  offset 0x34
        ds:67           segment pointed by ds, offset 0x67
        start           a location with name 'start'
```

WRAP
Messages Window

In this window you see various IDA messages. You can use arrow keys,
their combinations with Ctrl key, PgUp,PgDn keys to walk through this

```
window. IDA will keep the most recent messages only.

You can't close this window.

        arrows    - one step movements
        PgUp      - Page Up
        PgDn      - Page Down
        Ctrl-PgUp - top (the most ancient message)
        Ctrl-PgDn - bottom (the last message)

You can write all messages appearing in this window to a file.
For this, you should define an environment variable:

        set IDALOG
logfile

WRAP
How To Use List Viewer
. You can use all movement keys: PgUp,PgDn,Home,End and arrows.
. You can position to a line simply by typing in the desired line number.
. You can find a line typing in the beginning of it. In this case you
   can use
        Backspace key to erase the last character typed in.
        Ctrl-Enter to find another line with the same prefix.
   Please note that list viewer ignores prefix of a line up to last
   backslash
 or slash
 if it exists.
. Alt-T         search for a substring (case-insensitive)
   Ctrl-T        repeat last search

5. If the list is not in dialog mode you can use the following keys:
   Enter         jump to the selected item in the last IDA View window
   Ctrl-E        edit the current item
   Del           delete the current item
   Ins           insert a new item
   Ctrl-U        refresh information in the window

Esc or Enter closes the window.

Manual operand

Enter alternate string for the %D operand

  Original operand:
%A

~perand:



~heck operand:
~llow not matched operand:
't read file
't write file
't create file
Save the current window as
```

```
The current window
' is modified. Do you want to save its contents?
The current window is modified. Do you want to save its contents?
Search failed.
Enter file name to save to
Replace a string

<Pattern    :
<Replace to :
Find a string

<Pattern    :
Replace occurence at
Name
' at %08lX is deleted
Unexplored
Code
Data
Tail
Function
Ret
Extra lines
Flowed
arg
Byte
Word
Dword
Qword
Float
Double
Tbyte
PackedReal
String(
Struct
Alignment
InvSign
08lX)
OuterOff
08lX)
Seg
Var
Void
Help Index



[Main Idea]
:1399[Demonstration Version Constraints]

Indexes


[Menu system]
:1385[Index of IDC functions]

Basics
```

[IDA command line switches]
:1482[Configuration file]
[Keyboard Macros]
[Calculator]
[Segment Register Change Points]
[Problems List]
[Anchor]
[Background Analysis]

Windows

[Main IDA window]
[Messages window]
[Segments window]
[Names window]
[Functions window]
[Structures window]
[Stack variables window]
[Enums window]
[Signatures window]
[Cross-references window]
[Selectors window]
[Segment registers window]
[File viewer]
[File Editor]

How to

[How To Use List Viewer]
[How to use the Notepad]
[How to Enter an Identifier]
[How to Enter a Segment Value]
[How to Enter an Address]
[How to Enter a Number]
[IDC language]
[Expressions]
[Statements]
[Variables]
[Functions]
0 MS DOS EXE File
1 MS DOS COM File
2 Binary File
3 MS DOS Driver
4 New Executable (
)
5 Intel Hex Object File
6 MOS Technology Hex Object File
7 Linear Executable (
)
8 Linear Executable (
)
9 Netware Loadable Module (
)

```
10 Common Object File Format (COFF)
11 Portable Executable (
)
12 Processor-specific format
13 Object Module Format (
)
14 S-record format
15 ZIP archive
16 OMF library
17 ar library
18 User-defined format
19 Executable and Linkable Format (
)
20 Watcom DOS32 Extender (W32RUN)
21 Linux a.out (AOUT)
Unknown
Ctrl-Break detected, exiting

CORE INIT: Unknown file type
Loading file
' into database

Detected file format:
Loading registers
Creating segments
File
' is successfully loaded into IDA database.
Stack segment at 0x%04lX

Checking that segments for all addresses are created
Creating additional segment for %08lX


The input file seems to be packed, continue?
Reading relocation table

Bad relocation table

The input file has extra information at the end (tail %
, loaded %
 continue?

File has overlays. Load them?
Marking typical code sequences
File Name   :
Format      :
Base Address:
%04lXh Range:
%04lXh -
%04lXh Loaded length:
%04lXh
Entry point :
%04lX
04lX
't read input file (file structure error
 only part of file will be loaded
Using IDS file for module
```

```
: internal error (bad call of open
file)
: can't open file
-
: read error
-
: bad input file
: no memory
-
: write error or no disk space
-
: uncompression error
-
: bad input file
: temporary file read error
%
: unknown error code %
Possible file format:
: uncompatible version!

This type of output files is not supported.

Select file for module
Add Cross Reference

  <From :
<To   :
WRAP
Packed Files


Sometimes, executable files are shipped in a packed form. It means
that to disassemble these files you need to unpack them. You surely have
a good collection of packers/unpackers, haven't you?

If you do not have them, try to hunt for programs named: UNP, CUP, TSUP or
TRON.
And we can't miss the opportunity to mention the very first universal EXE
unpacker - our program UUP. It was the originator of all other
universal unpackers.

IDA says that the file is packed when Relocation Table of exe-file is
empty.

FLOW: NextHead %08lX of %08lX in data base !

Can't use BIOS comments base.

BIOS comments internal error: database is corrupt

Input string:

Syntax error:

Register
' is not valid
WRAP
```

```
Packed NLM Files


Sometimes, NLM files are shipped in a packed form.
Unfortunately we do not have information about the packing method used and
therefore we can't decompress such files.
 New Line
Choose line to edit

Choose mark number

Enter mark description

Choose marked location
't find offset base (hint: delete offset)
't find name (hint: use forced arg)
't find alternative string for an operand (hint: delete alt. string)
't find comment (hint: delete comment)
't find references (hint: redo analisys)
Indirect execution flow
't disassemble
Already data or code (hint: make 'unexplored
Execution flows beyond limits
Too many lines

SEGMENT: Can't create 'Segments' structure.

SEGMENT: Can't find file segmentation
. Creating a new segment
08lX
08lX)

     Additional segment
08lX
08lX)
       Undefining bytes
08lX
08lX)
     Request for reanalysing
08lX
08lX)
 OK

 FAILED

Deleting segment
08lX
08lX)
    Name    Start     End   Align Base Type Cls  32
    Name   Start   End  Align  Base  Type Cls  32

Instructions/data can't cross segment boundaries.

The segment would have bytes with negative offset

Addresses
08lX
```

```
08lX) contain instructions/data.
Do you want to discard them?

Can't move segment start address
08lX
08lX
 would overlap another segment

Delete Segment

CAUTION: ALL INFORMATION ABOUT THE SEGMENT WILL BE LOST!
Segment %
~ONFIRM DELETION:
~isable addresses:

Change segment attributes


<Segment ~
~ame    :
<Segment ~
~lass   :
~tart address   :
~nd   address   :
-bit segment:
>
~ombination:
-bit segment:


~lignment:
~ove adjacent segments:
~isable addresses:

Fatal: Can't create database.

Database for file
' isn't closed. Do you want IDA to repair it ?
IDA can't repair the database because the database is badly damaged. Try to
restore the packed database.

bTree error:

IDA has found old database files for
 Do you want IDA to rename old database files into new names?
't rename file
' into
File
' is not IDA database.
Unpacking the database %
Packing the database %
File is packed using unknown method.
Not enough disk space to unpack database

(wanted:
, available:
Not enough disk space to pack database
```

```
(wanted:
, available:

Fatal: Database has obsolete format. Use old version of IDA.

You database has obsolete format
 Now IDA will upgrade it into format %
. Continue ?

Fatal: Database format is newer than expected. Use new version of IDA.
[Main Menu Bar]
[File]

[Open window]

[Load additional file]

[IDA command]

[Produce output file]

[Generate MAP file]

[Generate ASM file]

[Generate LST file]

[Generate EXE file]

[Generate DIF file]

[Dump database to IDC file]

[Save database]

[Save database as


[Abort]

[Quit]
[Edit]

[instruction]

[data]

[ascii]

[array]

[undefine]

[Rename]

[Comments]
```

[Regular comments]

[Repeatable comments]

[Predefined comments]

[Additional comment lines]

[Operand types]

[Convert to number]

[Convert to hex number]

[Convert to decimal number]

[Convert to binary number]

[Convert to octal number]

[Convert to character]

[Convert to segment]

[Convert to offset (

[Convert to offset (

[Convert to offset (any segment

[Convert to offset (user-specified

[Convert to struct offset]

[Convert to enum member]

[Convert to stack variable]

[Change sign of operand]

[User-defined operand]

[Functions]

[Make function

[Edit function

[Delete function

[Set function end]

[Define stack variables

[Change stack pointer

[Structures]

[Add struct type]

[Del struct type]

[Move struct type]

[Declare struct var]

:1222[Enums]

:1379[Add enum type]

:1333[Del enum type]

:1379[Edit enum type]

:1384[Define a enum member]

:1725[Edit a enum member]

:1729[Delete a enum member]

[Segments]

[How to create a new segment]

[How to delete a segment]

[How to change segment attributes]

[How to move a segment start]

[How to change segment translation]

[Set default segment register value]

[Change segment register value]

[Patch core]

[Patch core]

[Other]

[Rename any address]

[Mark as variable]

[Jump table

[Alignment
[Navigate]

[Jump to

[Jump to the Specified Address]

[Jump to the Named Location]

[Jump to the Specified Segment Start]

[Jump to the Previously Marked Position]

[Jump to the Specified Segment Register Change Point]

[Jump to the Next Address from the Problems List]

[Jump to the Cross Reference]

[Jump to the Cross References to Operand]

[Jump to the Function]

[Jump to the Entry Point]

[How to Mark a Location]

[Search for

[void operands]

[instruction bytes]

[data bytes]

[unexplored bytes]

[explored bytes]

[immediate operand values]

[substring in the text representation]

[substring in the binary image of the file]

[bytes not belonging to any function]

[Jump]

[Return]

[Undo Return]

[Empty stack]
[View]

[View Functions]

[Calculator]

[View general registers]

[Open segment registers window]

[Open segments window]

[Open selectors window]

[Open names window]

[Open xrefs window]

[Open structures window]

[Open enums window]

[Open signatures window]

[View internal flags]

[View File]

[Edit file]
[Options]

[Text representation

[Representation of cross references

[Assembler directives

[Names representation

[Demangled C
 names

:1207[Colors

[Dump/Normal View]

```
[Setup data types


:1199[String styles


:1200[ASCII options


[Processor type


[Target assembler


[Auto analysis


[Search direction]
[Windows]
WRAP
Main Idea


Russian:
[Enter]
. IDA is not an automatic analyser of programs.
Normally it is expected that you and IDA work together.
IDA will hint you of suspicious instructions, unsolved
problems etc. It is your job to inform IDA how to proceed.

 2. There is no documentation on IDA. I think that built-
[help] is sufficient in the most of cases.
, use frequently
help system and you will learn IDA quickly.

 3. If you started IDA the very first time, there are commands
you will find useful:
- convert to
[instruction]
: the hotkey is
<MakeCode>
- convert to
[data]
: the hotkey is
<MakeData>

 4. All changes made by you are saved to disk. When you start IDA again
all the information about the file being disassembled is read from disk
and  you can continue your work.

For other commands please refer to the
[menu] system and the help.
Renumbering the generated names.
Rebuilding list of names.

Can't rename byte at %08lX as
```

```
' because the name
' is a register name.

Can't rename byte at %08lX as
' because it contains a bad character
; Title
; Format                           New Executable Format
; Target operating system          unknown
; Target operating system          OS/
; Target operating system          MS Windows
; Target operating system          DOS 4.
; Target operating system          MS Windows 386
; Target operating system          Borland Operating System Services (BOSS)
; Target operating system          unknown
; File Load CRC                    0%08lXh
; Program Entry Point   (
; Initial Stack Pointer (
; Auto Data Segment Index          %04Xh  (
; Initial Local Heap Size          %04Xh  (
; Initial Stack Size               %04Xh  (
; Linker Version                   %
; Minimum code swap area size      %
; Expected Windows Version         %
; Program Flags
 Multiple data
 Single data
 No data
 Global initialization
 Protected mode
 Self loading
 Errors in image
 Uses windowing API
 Incompatible with PM
 Compatible with PM
 DLL
 Application
; Other EXE Flags
 Long file names
.X protected mode
.X proportional font
 Fastload area (start:
%08lX, size:
%08lX)
; Segment Number    :
; Alloc Size        :
%04lXh
; Offset in the file:
%04lXh  Length:
%04lXh
; Attributes
 Use32
 Allocated
 Loaded
 Moveable
 Iterated
 Pure
 Preloaded
 Readonly
 Executeonly
 Relocations
 Conforming
 Discardable
 DPL:
; Resource, type
; File offset      :
%04lXh  Length:
%04lXh
; Attributes
; Resource ID      :
; Resource ID      :
 Cursor
 Bitmap
 Icon
 Menu
 Dialog box
 String table
 Font directory
 Font component
 Accelerator table
 Resource data
 Cursor directory
 Icon directory
 Name table
 Version info
 Unknown
 User defined '
 Movable
 Pure
 Preloaded
; External Entry
d into the Module
; Attributes
 Exported
 Shared dataseg
 Ring transactions %
 Fixed
 Moveable
Relocation type OS FIXUP is ignored by IDA
Unknown relocation type %
(segment %
,rel
Unknown relocation address type %
(segment %
,rel
,target:
Target of relocation entry is illegal (segment %
, target %
Bad target of relocation for segment %
(entry %
Bad target of relocation for segment %
(segment %
, offset %
Bad target of relocation for segment %
(module %
, function %
Bad target of relocation for segment %
(module %
, function name offset 0x%
File is self loading, relocation table is skipped
(negative value added)
Loading resources
Renaming entry points
Fixing relocations
File read error %d bytes at %08lX (may be bad NE structure
 continue?
NE File Manual Loading

        Segment :
%A
        Size    :
%N

 <Start ~
~ddress :
~elector      :
(0 means that selector
                                  will be chosen by IDA)
<STOP LOADING ~


  Note that IDA won't warn you if segments overlap!
(this is a feature, not a bug
Function frame size is incorrect, please add at least 0x%lX bytes

Please enter a structure name

Can't create a structure named
 Probably the name is invalid or is used in the program.

Unknown LX format level %
WRAP
Bad Relocation Table


Relocation table has references beyond program limits.
WRAP
Additional information at the end of file


The file being loaded is not completely loaded to memory by the
operating system. This may be because:
        - the file is overlaid; IDA does not know this type of overlays
        - the file has debugging information attached to its end
        - the file has other type of information at the end

Anyway, IDA will not load these bytes.
WRAP
Overlaid files


Some EXE files are built with overlays.
```

This means that not the whole file is
loaded into the memory at the start of
the program, but only a part of the file. Other
parts are loaded by the program itself
into the dynamic memory or over some
subroutines of the program. This fact
leads to many difficulties when you
disassemble such program.

For the moment IDA knows about overlays created by Borland C compiler.
WRAP
Error loading overlays

One of the following occured:

- overlay stub is not found
- overlay relocation data is incorrect

i.
. the input file structure is bad. If you are sure that the input
file structure is ok, please inform the author about it.
WRAP
Maximal number of segments is reached

When IDA tried to delete bytes outside of any segment, the maximal number
of contigious chunks is reached. This is NOT a fatal error.

Some bytes outside of any segment will be present
> the output text
will be incorrect because of these bytes. But you can delete them in the
output text using a text editor.
WRAP
Can't generate executable file

IDA produces executable files only for:
- MS DOS .exe
  - MS DOS .com
  - MS DOS .drv
  - MS DOS .sys
  - general binary
  - Intel Hex Object Format
  - MOS Technology Hex Object Format

Also, external loaders may or may not support the creation of user-defined
input file formats.
WRAP
Bad input file format

The input file does not conform to the following definitions:

Intel Hex Object Format

This is the default format. This format is
line oriented and uses only printable ASCII characters except for
the carriage return/line feed at the end of each line. Each line in
the file assumes the following format:
:NNAAAARRHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHCCTT

Where:

All fields marked '
' consist of two or four ASCII hexadecimal
digits (
A maximum of 24 data bytes will be represented
on each line.
:
Record Start Character
NN
Byte Count (
)
AAAA
Address of first byte (
)
RR
Record Type (
, 00 except for last record which is 01)
HH
Data Bytes (
)
CC
Check Sum (
)
TT
Line Terminator (carriage return, line feed)

The last line of the file will be a record conforming to the above
format with a byte count of zero
00000001FF

The checksum is defined as:

sum
byte
count + address
+ address
+
record
type +
(sum of all data bytes)
checksum
ffh)

MOS Technology Hex Object Format

This format is line oriented and
uses only printable ASCII characters except for the carriage
return/line feed at the end of each line. Each line in the file

assumes the following format:

```
;NNAAAAHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHCCCCTT
```

Where:

All fields marked '
' consist of two or four ASCII hexadecimal
digits (
A maximum of 24 data bytes will be represented
on each line.
;
Record Start Character
NN
Byte Count (
)
AAAA
Address of first byte (
)
HH
Data Bytes (
)
CCCC
Check Sum (
)
TT
Line Terminator (carriage return, line feed)

The last line of the file will be a record with a byte count of zero

The checksum is defined as:

sum
byte
count + address
+ address
+
(sum of all data bytes)
checksum
(sum
ffffh)
WRAP
FLOW: Bad Next Flow

This means that IDA database is corrupt.
Try to re-analyse the program.
(you can
force IDA to re-analyse the program by
changing processor type to the current
processor type

If you have previously saved your database into a text file,
you can load it. See
[Dump database] command for explanations.

WRAP
Bad BIOS
 DOS Comments Base

IDA can't use this base due to:
        - it does not exist or
        - it is corrupt

IDA will not generate new automatic
comments on DOS and BIOS interrupts and
port I/O instructions.
WRAP
Internal Error

Comments database is corrupt. IDA will not
generate comment(
WRAP
Bad Syntax for Comments Retrieval

The correct syntax is:

  Reg
Number

where reg is register name (
)
          or Op1 - for first operand
          or Op2 - for second operand
      number is its value (in C-notation)

For example:

  Op1
0x21 AX
0x4C01
WRAP
Choose mark number

This command allows you to mark a location so that afterwards you can
jump to the marked location immediately. Select any line from the list.
The selected line will be used for mark description.
Afterwards you will be able to jump to the marked location using <
> key.

You can use <
<Down
<PgUp
<PgDn
<Home
> keys. If you
select the first line of the list, nothing will be selected.

```
Press <Enter> to select line,
> to cancel.

See also:
[How to jump to the marked location]
WRAP
Enter mark description


Mark description is any text line. The description is
solely for your information.
WRAP
Choose marked location


This command allows you to jump to the previously marked location.
Select any line. You will jump to the selected location.

You can use <
<Down
<PgUp
<PgDn
<Home
> keys. If you
select the first line of the list, nothing will be selected.

Press <Enter> to select line,
> to cancel.

See also:
[How to mark a location]
WRAP
Can't Rename a Byte
. The name is ill-formed:
        - it is a reserved word
          NOTE: IDA does not allow to use register names as byte names.
        - it contains bad characters.
          The list of allowed characters is specified in IDA.CFG
        - it starts with a reserved prefix. Some prefixes are used
          by IDA to generate names. See
[Names Representation]
          dialog for the list of prefixes.
. The name is already used in the program. Try to use another
[name

   To learn where the name is used you can try to jump to it using

[Jump to the Named Location]
. The address can't have a name:
     - IDA refuses to rename tail bytes (
. the second, third

       bytes of instruction/data

     - the address does not belong to the program

WRAP
```

Can't find file segmentation

The database is empty or corrupt. All information is lost.

If you have a packed version of your database (fie .
 tell IDA to
override the unpacked version with packed version of database.

If you have previously saved your database into a text file,
you can load it. See
[Dump database] command for explanations.
WRAP
Segment Boundaries Fault

A attempt to change/add a segment failed.
This happened because a segment can't contain a part of
instruction or data. It can contian only a whole
instruction or data.
WRAP
Negative Offsets

A segment can't have bytes with negative offset from the segment base.
Example:

```
        if segment base is 0x3000, this segment can have
        start address above or equal to 0x30000, but it can't
        have start address 0x2FFFF.
```

WRAP
Can't move segment start address

IDA can't change the segment start address if some instructions/data would
occur outside
of the segment.
[Undefine] this instructions/data first.
WRAP
Can't move segment start address

IDA can't change the segment start address because this segment with the
new start address would overlap another adjacent segment. For example,
the following situation:

```
  seg
a  starts at 0x0000 ends at 0x1000
  seg
b  starts at 0x1000 ends at 0x2000
  seg
c  starts at 0x2000 ends at 0x3000
```

If you ask to change seg
c so that it starts at 0x500, you'll see this
error message, because the new seg
c would overlap seg

WRAP
Can't Create Database

Most probably your disk is full or some
disk I/O error occured.
WRAP
DataBase is not closed

The database was not closed after last IDA session. Most probably
this happened due to power fault, unexpected reboot of the computer,
generally, any abnormal session termination. Try to repair database.

No guarantee that the repairing will restore the database.

If you have previously saved your database into a text file,
may be the best solution is to load it now.
See
[Dump database] command for explanations.
WRAP
bTree Error

If memory is low, you may get the error 'no memory for

Try to free some memory and start again.

Other errors should not happen. If you have got another error message,
please inform the author about this error.
Do not forget to keep all IDA files and to describe the events occured
immediately before the error.

If you have previously saved your database into a text file,
you can load it. See
[Dump database] command for explanations.
WRAP
Obsolete Database Format

Please use old version of IDA. This
version of IDA does not support this
format.

If you have previously saved your database into a text file,
now you can load it. See
[Dump database] command for explanations.
WRAP
The Name List is Empty

This command allows you to select a name from USER DEFINED names.
No such names are defined now or all user-defined names are hidden.
To give a name to the location,
use
[Rename] or
[Rename any] commands.

WRAP
Upgrading IDA database

If IDA finds out that your database has old format, it will try to upgrade the database to new format.
The upgrade process is fully automatic, no user intervention is needed. However, after upgrading your database you will not be able to work with it using old versions of IDA. That is why IDA asks your confirmation before upgrading the database.

This feature works only for databases from IDA version 2.
WRAP
Unexpected Database Format

Database format is newer than expected.
That is because you use old version of IDA. The only thing you can do is to get somewhere a new version of IDA (may be from the same source where you got the database
WRAP
Unknown Linear Executable Header Format

IDA handles only LX header format level 0. The file you attempted to disassemble has newer format and IDA does not know about it.
WRAP
Unsupported Byte Order

```
IDA does not support byte
 word orderings other that Intel order:
        Byte Ordering: Little Endian Byte Ordering
        Word Ordering: Little Endian Word Ordering
```

Sorry.

If you send this file to <
estar.
'll try to make a new version
of IDA which supports the new format.
WRAP
Zero length segments

A segment in IDA can't have length equal to zero.
WRAP
Relocating objects

Some other object has been loaded at the specified address, so IDA will try to
relocate the current object to another address. If the object has not internal
fixup records, the result may be incorrect.
WRAP

Imported module is not found

  IDA did not find the specified module in:
    - the current directory
    - the operating system directory (see switch
]
      and configuration file parameters
:1482[WINDIR,OS2DIR]

Entries imported by ordinal will not be commented. If IDA finds a module, all imported by ordinal entries are commented like this:

```
  KERNEL
:
                retf            ; INITTASK


                                  comment
```

This comment will be propagated (
[repeated
 to all
locations which call this
entry:

```
                call    far ptr KERNEL
; INITTASK
```

IDA searches all files named
modulename
 for the module.
If you know that the imported module resides in another directory,
copy it the your current directory. And if the module file name is different
from
modulename
,
rename it. After the database is loaded, you can
delete the copied module.

Also IDA looks for file
modulename.

  in the current directory
  in the IDS subdirectory of the directory where IDA.EXE resides
  in the PATHed directories

You can yourself create such a file. For an example look at DOSCALLS.IDS
in the IDS subdirectory.
WRAP
Renaming old database files

IDA has found its old database in the current directory. It asks your
permission to rename them so that IDA can use the old database.

Starting from version 3.01 IDA can disassemble several files in one directory.
In order to allow this the names of the IDA database files are changed.
WRAP
Load file dialog

Below is description of dialog box fields:

```
 Load address - the paragraph where the file will be loaded.
                Meaningfil only for EXE and binary files. For new exe
                files please use 'manual load' feature.
 Load offset  - meaningful only for binary files.
                specifies offset of the first byte from the start of
                the first segment. For example, if load offset
0x2700 and
                load address
0x1000, the first byte of the file will
                be at 1000:2700.
 DLL directory - path where IDA will look up referenced DLL files.
                Note that if IDA finds .IDS file, it does not look for
                .DLL file.
 Create segments        - meaningful for binary files.
                          If not checked, IDA does not create segments.
 Load resources         - if not checked, IDA does not load resources from
                          NE files.
 Rename DLL entries     - If not checked, IDA makes repeatable comments
                          for entries imported by ordinal.
 Manual load            - meaningful only for NE,
,LX files.
                          If checked, IDA will ask loading addresses
                          and selectors for each object of the file.
                          For experienced users only!
 Fill segment gaps      - meaningful only for NE files.
                          If checked, IDA will fill gaps between
                          segments, creating one big chunk.
                          Use it if loading a new file you
                          get a message
Maximum number of chunks reached

                          Also you could defined environment variable
                          called IDA
PAGESIZE to a bigger value (default
                          is 4096

 Create imports section - meaningful only for PE files.
                          If checked, IDA will convert .idata section
                          definitions to
extrn
 directives and truncate it.
                          Unfortunately sometimes there is some additional
                          data in .idata section so you'll need to disable
                          this feature if some information is not loaded
                          into the database.
```

WRAP
PE .idata section has additional data

If
create imports section
 in the file loading dialog is checked,
IDA will convert .idata section definitions to
extrn

directives and truncate it so it will not contain empty lines.

Unfortunately sometimes there is some additional data in .idata
section so you'll need to disable this feature if some information is
not loaded into the database.

IDA tries to detect additional data in .idata section automatically.

If you disable conversion of .idata section to a segment with
extrn

directives, you will see

```
somename        dd      ?
```

instead of

```
        extrn   somename
```

directives.

Another impact is that the .idata segment will not be truncated in any way.
WRAP

 1. IDA
 Sourcer'
. IDA
.

 2.
 IDA
 F1
 IDA.
.

 3.
 IDA
<MakeCode>
<MakeData>

 4.
 IDA

Change segment combination

~ Private:
~ Group:

```
~ Public:
~ Reserved:
~ Public:
~ Stack:
~ Common:
~ Public:

IDA supports Intel byte ordering only
Object Iterated Pages Offset
08lX) is not equal to Data Pages Offset
08lX)
Bad segment base value
; Title
; Format                           Linear Executable Format (
; Format                           Linear Executable Format (
; Target operating system          unknown
; Target operating system          OS/
; Target operating system          MS Windows
; Target operating system          DOS 4.
; Target operating system          MS Windows 386
; Target operating system          Borland Operating System Services (BOSS)
; Target operating system          unknown
; Module Version                   %08lXh (
; Program Entry Point   (
08lXh
; Initial Stack Pointer (
08lXh
; DS Object                        %
; Heap Size                        %08lXh (
; Program Flags
08lXh
 Library init
 Library term
 No internal fixups
 No external fixups
 Uses PM API
 Incompatible with PM
 Compatible with PM
 Not loadable
 Application
 Protected memory library
 Phys. Dev. Driver
 Virt. Dev. Driver

Object %lu has zero length, object length is set to 1

Object %lu has virtual address %lXh which is already occupied, relocating it
to %
; Object Number     :
; Virtual Size      :
%08lXh
; Number of Pages   :
(present in the file)
; Attributes
08lX
 Readable
 Writable
 Executable
 Resource
 Discardable
 Shared
 PreloadPages
 InvalidPages
 Resident
 Contiguous
 ZeroFilledPages
 Resident
 Resident
 LongLockable
:16 Alias Required
 Privileged
 Big
 Conforms CODE
; External Entry
lu into the Module %
; Attributes:
-bit entry
 286-gate
-bit entry
 forwarder entry
 Exported
  Args:
't forward entry %lu for (module %
, proc off %
't forward entry %lu for (module %
Unknown relocation source type %Xh at %08lX
't apply 16-bit offset fixup at %08lX, target offset %08lX, truncating to 16-
Bad fixup target (source %
,target %
,fixup fileoff %
Fixup target is illegal!
<additive relocation:
Applying fixup records
File read error (may be bad LX structure at 0x%
 continue?
I do not know what to do with pages type %
't unpack page type ITERATE 2, the input file is bad.
/LE File Manual Loading

        Segment :
%A
        Size    :
%N
        Use32   :
%D

 <Start ~
~ddress :
~elector      :
(0 means that selector
                                  will be chosen by IDA)
<Segment ~
~ase  :
(in paragraphs)
```

```
<STOP LOADING ~


  IDA doesn't check values specified by you!
(this is a feature, not a bug
Loading resource table
?
Mouse pointer shape
Bitmap
Menu template
Dialog template
String tables
Font directory
Font
Accelerator tables
Binary data
Error message tables
Dialog include file name
Key to vkey tables
Key to UGL tables
Glyph to character tables
Screen display information
Function key area short form
Function key area long form
Help table for Cary Help manager
Help subtable for Cary Help manager
DBCS uniq/font driver directory
DBCS uniq/font driver
; Resource       ID:
; Resource     Type:
%04X
; Resource Location: obj
. offset
%08lXh size
%08lXh

Please enter a new structure number (

Enter STRUCT name

Enter struct member name

Enter stack variable name

Text representation

<Line ~
~refixes:
<Number of opco~
~e bytes:

1000:0FE4

1000:0FE4

1000:0FE4               loc
:
```

```
1000:0FE4 90                         nop     ;
~se segment names:
>
       ;
void
~egment addresses:
>

(length of

~unction offsets :



 arguments of
<
~structions indention :


 data directives)
~ments indention     :

<Display
' marks:
>

<Display ~
~mpty lines:
>
<Display borders bet~
~een data/code:
<Display bad ~
~nstructions <
> marks:
<void'
~ow  limit:
:1157
<Use ~
~abulations in output:
>            <void'
~igh limit:
:1157
<Display ~
~omments           :
<Display ~
~epeatable comments:
>                 <Display s~
~urce lines:
<Display ~
~uto comments      :
                <Display stac~
~ pointer:

Cross-reference representation

<Display se~
```

```
~ments in xrefs:
<Display xref ~
~ype mark   :

<Display ~
~unction offsets :
>

<Display xref ~
~alues      :





1000:0FE4 90                         nop     ; CREF: 1000:0FE0     |

<Right ~
~argin             :
(max length of line with xrefs)
<Cross reference ~
~epth    :
:1178
~umber of xrefs to display :

Assembler directives
    Generate

~SSUME directives:
~RG   directives:

Background analysis


~nalysis enabled:
>

~ndicator enabled:




~ernel analyser options:

Background analysis


~nalysis enabled:
>

~ndicator enabled:


 The analysis priority:
   <
~rrent :
>
   <
```

```
~e    :
>
~elta:


~egular :
>

~ritical:




~ernel analysis options:
IDA - The Interactive Disassembler Pro

%sVersion 3.
Copyright (
) 1997 by DataRescue sprl.
-mail: ida
datarescue.com

Please see README.TXT for license information.

Fatal error during loading overlays.
; Overlays: base
%08lX, size
%08lX, EXEinfo
%08lX
<EXTRA BYTE>
(the segment was empty)
IDA doesn't allow empty segments
so this byte appeared
Overlay manager interrupt
Runtime memory swap address
Offset in the file to the code
Code size
Relocation area size
Number of overlay entries
Previous stub
Deleting bytes at %08lX
08lX (they do not belong to any segment

Maximal number of segments is reached, some bytes are without a segment.

Enter DIF file name for %

Enter MAP file name for %

Enter ASM file name for %

Enter LST file name for %

Enter binary file name for %
Binary file %s is created.

Enter unload file name
Enter IDC statement(

Enter IDC file name

Enter file name to edit

Enter a new database file name

Patch Bytes

 File offset   :
%N
 Original value:
~alues         :

Patch Word

        Original value:
%N
        File offset   :
%N


~alue (word)

Enter new value for the current word

Auto analysis is not completed, the result may be inaccurate. Do you want to
continue?
 BEWARE
COFF format is not supported fully yet.
(relocation table is not analysed)
File read error (may be bad COFF structure
 continue?
0000 Unknown COFF machine
0210 Motorola 68000 w/o TV
0211 Motorola 68000 with TV
0222 TMS320C5
0401 pdp11 UNIX-rt ldp
0405 pdp11 overlay
0407 pdp11/pre System V vax executable
0410 pdp11/pre System V vax pure executable
0411 pdp11 seperate I
D
0437 pdp11 kernel overlay
0502 Basic-16 w/o TV
0503 Basic-16 with TV
0504 Intel iAPX 16 w/o TV
0505 Intel iAPX 16 with TV
0506 Intel iAPX 20 w/o TV
0507 Intel iAPX 20 with TV
0510 x86 w/o TV
0511 x86 with TV
0512 Intel 286
0514 X386MAGIC: reserved for Intel (0514)
0515 Intel 860
0516 reserved for Intel (0516)
```

```
0517 reserved for Intel (0517)
0520 Motorola 68000 writeable text segments
0521 Motorola 68000 readonly shareable text segments
0522 Motorola 68000 demand paged text segments
0524 I386PTXMAGIC (0524)
0525 reserved for NSC (0525)
0530 IBM 370 writeble text segments
0531 Amdahl 470/580 writable text segments
0532 reserved for u370 (0532)
0533 reserved for u370 (0533)
0534 Amdahl 470/580 readonly sharable text segments
0535 IBM 370 readonly sharable text segments
0540 xl
0544 reserved for Zilog (0544)
0545 reserved for Zilog (0545)
0550 3B20 executable, no TV
0551 3B20 executable with TV
0560 WE 32000, no TV
0561 WE 32000 with TV
0562 reserved for WE 32000
0565 I386AIXMAGIC
0570 VAX 11/780 and VAX 11/750 writeable text segments
0575 VAX 11/780 and VAX 11/750 readonly sharable text segments
0630 IBM RT writeable text segments
0635 IBM RT readonly sharable text segments
0637 IBM RT readonly text segments and TOC
0730 IBM R2 writeable text segments
0735 IBM R2 readonly sharable text segments
0737 IBM R2 readonly text segments and TOC
Machine (
%s
Number of sections      :
%d
Time
 date stamp       :
%s
File pointer to symtab  :
%08lX
Number of symtab entries:
%ld
Sizeof(optional hdr)    :
Flags %
0x0001 Relocation info stripped
0x0002 Executable
0x0004 Line numbers stripped
0x0008 Local symbols stripped
0x0010 Minimal obj file
0x0020 Fully bound update file
0x0040 Bytes swapped
0x0080 16-
, LSB first (PDP 11/
)
0x0100 32-
, LSB first (Intel)
0x0200 32-
, MSB first (Motorola)
0x1000 Dynamically loadable and executable
0x2000 Shared object
Contains
patch
 list
No decision funcs for replaced funcs
0410 RO text segment, data segment follows
0404 object files, eg as output
0407 text and data squashed together
0370 public library created by ldp
0371 data library created by ldp
0401 ldp created kernel process
0402 ldp created supervisor process
0403 ldp created user process
0400 sgen created boot process
0413 segments are aligned and can be paged
0443 host shlib
any  Unknown execution magic
AOUT Header:
(this info is not used by IDA)
Exec flags 0%
%s
      Version stamp:
%d
      .text size   :
%08lX
)
      .data size   :
%08lX
)
      .bss  size   :
%08lX
)
      entry point  :
%08lX
      .text base   :
%08lX
      .data base   :
%08lX

 Section %
.8s
             physical address             :
%08lX
             virtual address              :
%08lX
             section size                 :
%08lX
)
             offset to raw data for section:
%08lX
             offset to relocation         :
%08lX
             offset to line numbers       :
%08lX
             number of relocation entries  :
%ld
             number of line number entries :
```

```
Section flags %08lX:
0x0000 Regular
0x0001 Dummy
0x0002 Noload
0x0004 Grouped
0x0008 Padding
0x0010 Copy
0x0020 Code
0x0040 Data
0x0080 Bss
0x0100 Exception
0x0200 Comment
0x0400 Overlay
0x0800 Lib section
0x1000 COMDAT
0x2000 Debug
0x4000 TypeCheck
0x8000 Overflow
0x01000000 ExtendedRelocs
0x04000000 NotCacheable
0x08000000 NotPageable
0x10000000 Shared
0x20000000 Executable
0x40000000 Readable
0x80000000 Writable
1  function argument
2  character
3  short integer
4  integer
5  long integer
6  floating point
7  double word
8  structure
9  union
10 enumeration
11 member of enumeration
12 unsigned character
13 unsigned short
14 unsigned integer
15 unsigned long
/compression error!
section %
(file %
) length %08lX
 nreloc %u nlinno %


~rocessor specific analysis options:
:1730

Kernel analysis options


~reate offsets and segments using fixup info  :
~ark typical code sequences as code           :
~elete instructions with no xrefs             :
~race execution flow                          :
```

```
<Create ~
~unctions if call is present           :
~nalyse and create all xrefs                  :
~se flirt signatures                          :
<Create function ~
~f data xref data
code32 exists:
~ename jump functions as j
                :
<Rename ~
~mpty functions as nullsub
         :
<Create ~
~tack variables                        :
<Trace stack ~
~ointer                           :
<Create ascii string if data ~
~ref exists       :
<Convert ~
~2bit instruction operand to offset   :
<Create ~
~ffset if data xref to seg32 exists    :
~e final analysis pass                      :
~ocate and create jump tables                   :
~oagulate data segments in the final pass       :

Names
Dummy name representation:
1234    :
> segbase relative to prog base
 offset from segbase
<
1000
1234 :
> segment base address
 offset from the segment base
<
dseg
1234 :
 segment name
 offset from the segment base
<
11234   :
> segment relative to base address
 full address
<
1000
11234:
> segment base address
 full address
<
dseg
11234:
> segment name
 full address
<
12        :
```

```
> full address (no leading zeroes)
0012      :
> full address (at least 4 digits)
00000012  :
> full address (at least 8 digits)
<dseg
1234     :
> the same as
 without data type specifier
<
1         :
 enumerated names (
~enumber:

Types of names included in the list of names:
~ormal:
~ublic:
~utogenerated:

Demangled C
 names
Show demangled C
 names as:
~omments:
>

~ames:
't demangle:
<Setup ~
~hort names:



 <Setup ~
~ong  names:

Demangled C
 names

~nhibit everything except the main name:
<No und~
~rscores in
ccall,
pascal, etc:
<No calling conventions for p~
~rameteres and
based
~eturn type of functions:
~ased
 specifier:
~alling conventions:
<No postfi~
~ const in member function declarations:
~ublic/private/protected keywords:
<No thro~
~ descriptions:
<No static and ~
```

```
~irtual keywords:
<No class/struct/union/
~ keywords:
<No const and vo~
~atile keywords:

Default memory model (if meaningful
~isabled:
>     (or flat)

>
~igned int is displayed as sint  :
>
~nsigned int is displayed as uint:
             <inser~
~ spaces after commas:
:1199

ASCII string style

 Create a string now:                   | Setup default string type:
style (0 terminated)     :
-style (0 terminated
~OS style (
 terminated)
~ DOS style (
 terminated
~ascal style (length byte)
~ Pascal style (length byte
~ide pascal (length 2bytes
~ Wide pascal (length 2bytes
~nicode                    :
~ Unicode:
~haracter terminated       :
~ Character terminated:



~irst  termination character:


~econd termination character:
:1200

ASCII string options


~enerate names:


<Names pre~
~ix  :





~ark as autogenerated:
```

```
<Generate ~
~erial names:




    <Serial ~
~umber :


<Serial ~
~idth  :
<ASCII next ~
~ine char:
(forces start of next line)

Enter a file name
Search direction: up

Search direction: down


Evaluate expression

  <Expression :
%A  Hex      :
%A
  Decimal  :
%D
  Octal    :
%O
  Binary   :
%A
  Character:

Evaluate expression

  <Expression :



  %
WRAP
```

Background Analysis

IDA can analyse the program when it is not occupied performing
an action asked by you. Thus IDA and you disassemble the program
together, but your requests have priority.

The state of background analysis is shown on the right upper corner
of the screen. See
[details] of this indicator for
further explanations.

You can

[disable] autoanalysis, but in this case
some functions of IDA will produce strange results (
. if you try to
convert data to instructions, IDA will NOT trace all threads of
control flow and the data will be converted to instructions only on
the screen

See also
:1730[kernel] analysis options.
WRAP
No Segment for the current byte

Some commands can't be applied to the addresses without a segment.

Create a segment first. You can do this using

[CreateSegment] command.
Default DS register:
%04X

Change segment alignment

~ouble word (4 bytes
~uadro word (8 bytes
~aragraph (16 bytes
~2 bytes:
~4 bytes:
(256 bytes
~096 bytes:
~roup:

Can't rename byte at %08lX as
' because the name is already used in the program.
~avigate
~indows
~ptions
~unctions
~egisters
~egments
~ructs
~nums
~dd struct type
Delete struct type
~ove struct type
~eclare struct var
Add e~
~ete enum
~t enum
't initialize IDP module.
IDA kernel and IDP module are not compatible.

Unknown switch

```
Do you want to disassemble file

Select file to disassemble

Unknown processor type
' Please check if IDA.CFG file is correct and the corresponding processor
module exists.

Database is empty

Can't use these switches with the old file
Database for file
' is loaded.
IDA is analysing the input file

You may start to explore the input file right now.

Can't create segment registers area at %08lX

Can't assign to Segment Register at %08lX
 Trying to recover (please repeat your command
Looking for

Can't create segment register areas

Segment Registers
WRAP
Can't create segment registers area

Happened due to the database. The
database is corrupt.  Try to finish
your work as soon as possible and get
the text source file.

If you have previously saved your database into a text file,
you can load it. See
[Unload] command for explanations.
WRAP
Can't assign to Segment Register

The database is corrupt.
Result of disassembling may be surprising,
nothing can be guaranteed.

If you have previously saved your database into a text file,
you can load it. See
[Unload] command for explanations.
.EXE is corrupted.
You can't disassemble file with such an extension:

Internal error: Attempt to use UNDEFINED register

Internal error: output operand type %

Internal error: loading void operand at %08lX, work:

Internal error: saving void operand at %08lX, work:
```

```
Internal error (
%s at %08lX

Internal error (
 Illegal addressing at %08lX

Internal error (
 Illegal operand size %
. at %08lX

Internal error (
 Illegal immediate type %x at %08lX

Functions window
~dd enum member
Loading IDP module %s for processor %
WRAP
Table Driven Assembler (TASM) by Speech Technology Inc.

Z80 mode: Beware! This assembler produces incorrect results for commands like
        RL (
)

It doesn't recognize
        RST 0
        RST 8

and accepts
        RST 00h
        RST 08h

IDA will substitute them with '
' directives.
WRAP
ASxxxx by Alan R. Baldwin v1.5

Beware! This assembler produces incorrect results for

        SUB


  instruction.

IDA will substitute them it '
' directives.

You are up to change the segment
' bounds
08lX
08lX) to
08lX
08lX
 Continue?

The segment
' addressing mode will be changed to %
```

```
. ALL INFORMATION about instructions/data WILL BE LOST! Are you sure?

Segment end address is invalid or less than start address

Bad segment base: segment would have bytes with negative offset

Invalid segment start address
Good bye
ANALYSIS
ANALYSIS2
ASCII
GENNAMES
ASCII
LINEBREAK
ASCII
STYLE
ASCII
PREFIX
ASCII
SERIAL
ASCII
SERNUM
ASCII
TYPE
AUTO
ASCII
ZEROES
AUTOSAVE
OPCODE
BYTES
CHECK
MANUAL
ARGS
COMMENTS
INDENTION
DEFAULT
PROCESSOR
DUMMY
NAMES
TYPE
ENABLE
ANALYSIS
INDENTION
LIST
NAMES
LOOKBACK
MACRO
MAX
NAMES
LENGTH
MAX
TAIL
MAX
XREF
LENGTH
MAX
DATALINE
```

LENGTH
NO
NOVICE
OFF
ON
OS2DIR
PACK
DATABASE
PRIORITY
CLASS
PRIORITY
DELTA
SCREEN
CURSOR
SCREEN
MODE
SCREEN
PALETTE
SHOW
SEGMENTS
SHOW
ASSUMES
SHOW
AUTOCOMMENTS
SHOW
INSTRUCTIONS
SHOW
BORDERS
SHOW
EMPTYLINES
SHOW
INDICATOR
SHOW
LINEPREFIXES
SHOW
ORIGINS
SHOW
REPEATABLE
COMMENTS
SHOW
SEGXREFS
SHOW
SOURCE
LINNUM
SHOW
VOIDS
SHOW
XREFS
SHOW
XREF
VALUES
SWAP
EXPANDED
SWAP
EXTENDED
USE
FPP

```
USE
SEGMENT
NAMES
USE
TABULATION
YES
WINDIR
XlatAsciiName
AsciiStringChars
NameChars
DATABASE
MEMORY
VPAGESIZE
VPAGES
NPAGESIZE
NPAGES
MangleChars
SubstChar
ShortNameForm
LongNameForm
~dit enum member

Choose segment
Sorry, IDA can't check operands for the current assembler.
~el enum member
~esize/move
~revious
~lose
~scade
~ump to
~earch for
~pen window
~oad additional file
IDC ~
~ommand
~DC file
~roduce output file
Produce ~
~AP file
Produce ~
~SM file
Produce ~
~ST file
Produce ~
~XE file
Produce ~
~IF file
~mp database to IDC file
OS s~
~ave database
~e database as
~bort - do not save changes
~uit - save changes
~ddress
~egment
~ment register change point
~roblem
```

```
~arked position
~ross reference
Cross ~
~eference to operand
~unction
~ntry point
Jump immediate
~eturn
~ndo last return
~mpty stack
Search ~
~irection
Mark ~
~osition
next
next ~
next ~
next ~
~nexplored
next ~
~xplored
~mmediate
next i~
~mediate
next te~
text in c~
next text in co~
not ~
~unction
~ump table
~lignment
~ariable
~SCII
Arra~
~ndefine
~atch program
~ments
~perand types
~ext representation
~oss references
Assembler d~
~rectives
ASCII strings ~
~tyle
ASCII strin~
~s options
~olors
~ame representation
~angled names
/normal view
Setup ~
~ata types
~rocessor type
Target ~
~ssembler
Analysis ~
~ptions
```

```
~ake function
~dit function
~elete function
~et function end
Stac~
~ variables
~hange stack pointer
~ame any address
Drop/Raise ~
~nchor
Enter comment
Enter ~
~epeatable comment
Retrieve ~
~redefined comment
Edit extra ~
~nterior lines
Edit extra p~
~sterior lines
~inary
Octa~
~ecimal
~acter
~exadecimal
~umber
~egment
~ffset by data segment/
Offset by ~
~urrent segment
Offset by ~
~ny segment

Enter segment translation list (list of segment names)
Segment translation list:
Bad keyboard assignment

Keyboard definition syntax is:


ActionName
 Value

  where value may be:

        a string:
Ctrl-

        a char:         '
'
        a scancode:     0x4900
        zero:           0

  Zero scancode disables the hot key.
' not found

By configuration file syntax here must be '
' sign.
```

Expected a string value
String is too long
Macro body is too long
Expected a number
The specified value isn't within the allowed range
Expected YES/NO or ON/OFF values
Too long string value
Page size should be power of 2
Original operand is unknown, can't check the entered operand.
This keyword should be defined in the first pass
Illegal action name

WRAP
Window is in Dumping Mode

Some commands do not work in the windows in dumping mode.
If you want to apply this command, please turn
[dumping] mode off.
Another way is to
[open] another window.
WRAP
Array Size

Action   name: MakeArray
Current hotkey:
<MakeArray>

Enter array size in current array elements (not bytes

The suggested array size is the maximal size so that no
user-defined names would be destroyed.

You can use this command for ASCII strings also.

Items on a line (does not work for ASCII strings

```
        0               place maximal number of items on a line
        other value     number of items on a line
```

Please note that the
[margin] parameter affects number of
items on a line too.

Alignment (does not work for ASCII strings

```
        -1              do not align items
        0               align automatically
        other value     width of each item
```

```
Signed elements:        if checked, IDA treats all elements as signed
numbers.
                        meaningful only for numbers (not for offsets and
                        segments and strings)

Create as array:        if not checked, IDA will create a separate item for
```

each array element. Useful to create huge arrays.
If this checkbox is clear when this command is
applied to ascii string, IDA will create many
ascii strings instead of one big string.

See also

[Edit] submenu

[How to Enter a Number
WRAP
Immediate Operand
 Offset (any segment)

 Action   name: OpAnyOffset
 Current hotkey:
<OpAnyOffset>

This command converts immediate operand(
) type of the
current instruction/data to an offset by specified segment base.

See also:
[offset by data segment/
]

[offset by current segment]

[offset by any user-specified base]

[Edit|Operand types] submenu.

If an area is selected using the
[anchor
 IDA will perform
'en masse' conversion. It will convert
immediate operands of all instructions in the selected area to
offsets. However, first IDA will ask you the lower and upper limits of
immediate operand value. If the an operand value is >
 lower limit and
<
 upper limit then the operand will be converted to offset, otherwise it
will be left unmodified.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

See
[Enter #th operand manually] commands.
WRAP
Immediate Operand

Offset (user-specified base)

Action   name: OpUserOffset
Current hotkey:
<OpUserOffset>

This command converts immediate operand(
) type of the
current instruction/data to an offset by specified base.

See also:
[offset by data segment/
]

[offset by current segment]

[offset by any segment]

[Edit|Operand types] submenu.

IDA will ask you to enter an offset base. The offset base is a linear address.

If an area is selected using the
[anchor
IDA will perform
'en masse' conversion. It will convert
immediate operands of all instructions in the selected area to
offsets. However, first IDA will ask you the lower and upper limits of
immediate operand value. If the an operand value is >
lower limit and
<
upper limit then the operand will be converted to offset, otherwise it
will be left unmodified.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

See
[Enter #th operand manually] commands.
WRAP
Immediate Operand
Struct Offset

Action   name: OpStructOffset
Current hotkey:
<OpStructOffset>

This command converts immediate operand(
) type of the

current instruction/data to an offset within the specified struct.

See also:
[offset by data segment/
]

[offset by current segment]

[offset by any segment]

[offset by any user-specified base]

[Edit|Operand types] submenu.

If an area is selected using the
[anchor
 IDA will perform
'en masse' conversion. It will convert
immediate operands of all instructions in the selected area to
offsets. However, first IDA will ask you the lower and upper limits of
immediate operand value. If the an operand value is >
 lower limit and
<
 upper limit then the operand will be converted to offset, otherwise it
will be left unmodified.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

See
[Enter #th operand manually] commands.
WRAP
Immediate Operand
 Stack Variable

 Action   name: OpStackVariable
 Current hotkey:
<OpStackVariable>

This command converts immediate operand(
) type of the
current instruction to an offset to stack variable,
. a local variable
or function argument in the stack.

You need to
[define] stack variables before using this command.

If the current operand is based on the value of stack pointer

and SP value is traced incorrectly, then you need to correct SP value

using
[change stack pointer] command.

If an area is selected using the
[anchor
 IDA will perform
'en masse' conversion. It will convert
immediate operands of all instructions in the selected area to
stack variables. However, first IDA will ask you the lower and upper limits
of
immediate operand value. If the an operand value is >
 lower limit and
<
 upper limit then the operand will be converted to stack variable, otherwise
it
will be left unmodified.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

See also:
[Edit|Operand types] submenu.

[Enter #th operand manually] commands.

[Define stack variables
WRAP
Immediate Operand
 Symbolic Constant

 Action   name: OpEnum
 Current hotkey:
<OpEnum>

This command converts immediate operand(
) type of the
current instruction/data to an
:1222[enum] member.

If an area is selected using the
[anchor
 IDA will perform
'en masse' conversion. It will convert
immediate operands of all instructions in the selected area to
symbolic constants. However, first IDA will ask you the lower and upper
limits of
immediate operand value. If the an operand value is >
 lower limit and
<
 upper limit then the operand will be converted to offset, otherwise it
will be left unmodified.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are affected.

See also:
[Edit|Operand types] submenu.

[Enter #th operand manually] commands.
WRAP
Change operand sign

Action name: ChangeSign
Current hotkey:
<ChangeSign>

This command change sign of the current operand.
Please note that not all operands can change their sign.

See also:
[Edit|Operand types] submenu.

[Enter #th operand manually] commands.
WRAP
Alternate String For Operand

Action name: ManualOperand
Current hotkey:
<ManualOperand>

You may use User-Specified String for operand when IDA does not represent operand in the needed form. In this case IDA will simply show the specified string instead of the operand. No offset references are generated.

The current operand (near the cursor) will be affected. However, if it is not modifiable (for example, a register
the next operand will be taken.
If no operand is modifiable, the command fails.

You can use this not only with the instructions but with data directives also.

IDA proposes the previous User-Specified String as default.

If you want to delete the User-Specified String, specify empty string.

If
Check operand
checkbox is on, IDA will try to check operand and
if the new operand value differs from its original value, will warn

you.

IDA automatically deletes the manually entered operands when you
change operand representation using operand
[types] submenu.

See also
[Edit|Operand types] submenu.
WRAP
Extra Lines

Action	Name	Hotkey
edit anterior lines	MakeExtraLineA	
<MakeExtraLineA>		
edit posterior lines	MakeExtraLineB	
<MakeExtraLineB>		

If you want to enter multi-line comments or additional instructions,
you can use this feature of IDA.

There are two kinds of Extra Lines: generated before instruction line
and generated after instruction line.

You can add new lines, edit existing lines and delete the last line.

Do not forget that maximal number of lines for an item is

100	16bit version of IDA
500	32bit versions of IDA

See also
[Edit|Text|Comments] submenu.
[How to use the Notepad]
WRAP
Enter a comment

Action   name: MakeComment
Current hotkey:
<MakeComment>

If you stand at the function start and your cursor is
on a function name, IDA will ask you to enter a function comment.

If you stand at the segment start and your cursor is
on a segment name, IDA will ask you to enter a segment comment.

If this command is issued in the

[structures window
it allows you to change comment of a structure or

structure member. If the cursor is on the structure name, it will be changed, otherwise
the member name.

Otherwise this command allows you to enter a normal indented comment for
the current item.

You can show/hide all comments in
[Text Representation Dialog

See also
[Edit|Comments] submenu

[Repeatable comments]
How to use the
[notepad]
WRAP
Enter a repeatable comment

Action   name: MakeRptCmt
Current hotkey:
<MakeRptCmt>

If you stand at the function start,
IDA will ask you to enter a function comment.

If this command is issued in the

[structures window
it allows you to change comment of a structure or
structure member. If the cursor is on the structure name, it will be changed, otherwise
the member name.

Otherwise this command allows you to enter a repeatable comment for
the current item.

You can't enter repeatable segment comments.

All items that refer to the current item will have this comment by default.

Note that if have defined both comment types (
[regular] and repeatable

the regular comment will be displayed for the current item and the repeatable
comment will be displayed for all items that refer to the current item if
they do not have their own comments.

The repeatable comments may be used to describe subroutines, data items etc

because all calls to the subroutine will have the repeatable comment.

You can show/hide all comments in
[Text Representation Dialog

You can show and hide repeatable comments using

:1100[Toggle repeatable comments] command.

Press Ctrl-Enter to start a new line. For a complete list of keys see

How to use the
[notepad]
.

See also
[Edit|Comments] submenu

[Regular comments]
WRAP
Give Name to the Location

 Action   name: MakeName
 Current hotkey:
<MakeName>

This command gives name/renames/deletes
[name] for the current
item.

To delete a name, simply give an empty name.

If the current item is referenced, you can't delete its name. Even if you try, IDA will generate a
[dummy] name.

Here you can also include/remove the name from the
[name list

If the name is hidden you will not see it in
[names window

See also
[Rename any] command.

[Edit] submenu.

[How to Enter an Identifier

[Names representation
WRAP
Give Name to the Any Address

 Action   name: MakeAnyName
 Current hotkey:
<MakeAnyName>

This command gives name/renames/deletes
[name] for the
specified address.

To delete a name, simply give an empty name.

If the specified address is referenced, you can't delete its name. Even if you
try, IDA will generate a
[dummy] name.

Here you can also include/remove the name from the
[name list

If the name is hidden you will not see it in
[names window

If this command is applied to a stack variable in an instruction, then IDA will allow you to rename the stack variable.

See also
[Rename current address] command.

[Edit|Other] submenu.

[How to Enter an Identifier

[Names representation
Binary string format

You can:
- enter sequence of numbers in the selected radix
    (hexdecimal, decimal or octal)
- enter string constants

Sequence must be space or comma separated.

If you enter a number, it will occupy the minimal numbor of bytes it fits in.
Example:

```
  CD 21        - will search for bytes CD 21
  21CD         - will search for bytes CD 21
```

Hello
, 0   - will search for the null terminated string
Hello
Error occured during opening file
't find file
't find file IDA.
WRAP
Search for substring in the file

 Action   name: AskBinaryText

Current hotkey:
<AskBinaryText>

This command searches for the specified substring in the file being disassembled. This command can be used for fast lookups of text strings in the executable file or even to find references to a data. You can interrupt it pressing Ctrl-Break.

If an area is selected using
[anchor
 IDA will search
for the specified substring in the area.

Follow this
[link] to learn about format of the input
string.

For example, if you want to find a reference to the following string:

```
 35F2:106A      db 'Hello
0
```

you could search for the string '0x106A' in the file.

See also
[search for text] command.

[Navigate|Search for
 submenu.
WRAP
Search for next instruction/data with the specified operand

Action   name: AskNextImmediate
Current hotkey:
<AskNextImmediate>

This command searches for the first instruction or data byte that contains the specified immediate operand. This command is relatively slow, because it disassembles each instruction to find operand values.

See also
[Navigate|Search for
 submenu.

[How to Enter a Number
WRAP
Search for substring in the text representation

Action   name: AskNextText
Current hotkey:
<AskNextText>

This command searches for the specified substring in the text representation. This command is a slow command, because it

disassembles each instruction to get the text representation.
IDA will show its progress on the
[indicator

You can interrupt this command pressing Ctrl-Break.

If an area is selected using
[anchor
 IDA will search
for the specified substring in the area.

Note that this command searches the same what you see on your
screen (not in binary image

For binary search look at

[Search for substring in the file]

See also
[Navigate|Search for
 submenu.
WRAP
Open Segment Registers Window

 Action   name: ShowSegmentRegisters
 Current hotkey:
<ShowSegmentRegisters>

This command opens the segment registers window.
The window will contain segment register

[change points]
list.

You can use
[list viewer] commands in this window.

Depending on the current processor type, you will see
DS,
,SS with or without FS,
.

See also
[Edit|Segments] submenu.

[View] submenu.
WRAP
Open Segments Window

 Action   name: ShowSegments
 Current hotkey:
<ShowSegments>

This command opens the segments window.

You can use
[list viewer] commands in this window.

In order to change selector values use
[selectors] window.

See also
[View] submenu.
WRAP
Open Selectors Window

```
 Action    name: ShowSelectors
 Current hotkey:
<ShowSelectors>
```

This command opens the selectors window. Here you can change
selector-base mapping.

You can use
[list viewer] commands in this window:

```
        - jump to the paragraph pointed by the selector
        - add a new selector
        - delete selector (if it is not used by any segment)
        - change selector value (this leads to reanalysis of all program)
```

See also
[View] submenu.
WRAP
Open Names Window

```
 Action    name: ShowNames
 Current hotkey:
<ShowNames>
```

This command opens the
[names] window.

You can use
[list viewer] commands in this window.

See also
[View] submenu.
WRAP
Open Cross References Window

```
 Action    name: OpenXrefs
 Current hotkey:
<OpenXrefs>
```

This command opens the cross-references window. This window contains
all references to the current location.

You can use

[list viewer] commands in this window.

You can add and delete cross references here too.

See also
[View] submenu.
WRAP
Open Structures Window

 Action   name: OpenStructures
 Current hotkey:
<OpenStructures>

This command opens the structure definitions window.

You can modify structure definitions here: add/rename/delete structures,
add/delete/define structure members.

Each structure must have a unique name. A field name must be unique in the
structure. In order to create or delete a field, use data definitions
commands (
[data
[ascii
[array
[undefine
[Rename

Also you may define
[regular] or
[repeatable] comments.

See also
[View] submenu.
WRAP
Open Stack Variables Window

 Action   name: OpenStackVariables
 Current hotkey:
<OpenStackVariables>

This command opens the stack variables window for the current function.

The stack variables are internally represented as a structure. This structure
consists of two parts: local variables and function arguments.

You can modify stack variable definitions here:
add/delete/define stack variables, enter comments for them.

There may be two special fields in this window:
 and
.
They represent size of the function return address and saved registers in
bytes.
You can't modify them directly. To change them, use

[edit function] command.

Offsets at the line prefixes represent offsets from the frame pointer register (

The window indicator at the lower left corner of the window displays offsets from the stack pointer.

In order to create or delete a stack variable, use data definitions commands (
[data
[ascii
[array
[undefine
[Rename

Also you may define
[regular] or
[repeatable] comments.

The defined stack variables may be used in the program by converting operands to
[stack variables

Esc closes this window.

See also
[Edit|Functions] submenu.

[Convert to stack variable]
WRAP
Change Stack Pointer

 Action   name: ChangeStackPointer
 Current hotkey:
<ChangeStackPointer>

This command allows you to specify how the stack pointer (
)
is modified by the current instruction.

You can't use this command if the current function is undefined.

You need to speicfy it only if IDA was not able to trace value of SP register.
Usually IDA can handle it but it some special cases it fails.
An example of such a situation is an indirect call of a function that purges its parameters from the stack. In this case IDA has no imformation about the function and can't properly trace value of SP.

Please note that you need to specify the
difference
 between old and
new values of SP.

The value of SP is used if the current function accesses local variables by [
] notation.

See also
[Edit|Functions] submenu.

[Convert to stack variable]
WRAP
Open Enums Window

 Action   name: OpenEnums
 Current hotkey:
<OpenEnums>

This command opens the enum definitions window.

You can modify enum definitions here: add/rename/delete enums,
add/delete/define enum members (
. user-defined symbolic constants)

Also you can add a comment for the enum and for each enum member.
In order to specify a enum comment, you should stand at the enum name.
Comments are set using regular commands:

[Regular comments]

[Repeatable comments]

See also
[View] submenu.

```
GL internal error: item size
 0 at %08lX, Flags
%08lX
.ERROR 'too many lines (more than 500

 Start  Stop   Length Name               Class



Program entry point at %


  Address         Publics by Value



Can't open for write file
 This file is generated by Interactive Disassembler (
)
 Copyright (
) 1997 by DataRescue sprl,
datarescue.
>
 This file is generated by Interactive Disassembler (
)
 Copyright (
) 1997 by DataRescue sprl,
datarescue.
>
      This file should be used in the following way:
         - reload executable into IDA with using switch -
         - press F2 and enter name of this file.
      NOTE: This file doesn't contain all information from the datbase.
#include <
>

static main(void)
{
        GenInfo

 various settings
        Segments

 segmentation
        Enums

 enumerations
        Structures

 structure types
        Bytes

 individual bytes (code,data)
        Functions

 function definitions
        SegRegs

 segment register values
}
 General information

static GenInfo(void)
{

        DeleteAll

 purge database

 Information about segmentation

static Segments(void)
 Information about bytes

static Bytes
(void)
{
        auto x;
 Information about segment registers

static SegRegs(void)
```

```
 Call all byte feature functions:

static Bytes(void)
 Information about functions

static Functions(void)
 Information about enum types

static Enums(void)
{
        auto id;
 enum id
        auto cid;
 const id


 Information about structure types

static Structures(void)
{
        auto id;
 End of file.
 Partial file is produced.

Write error, may be disk is full?
Writing assembler file %
,
address range %08lX
08lX
Writing listing file %
,
address range %08lX
08lX
Assembler file created, total %lu lines.
Listing file created, total %lu lines.
Map file created, total %lu lines.
Flushing buffers, please wait
 Ok


This command can't be applied to the addresses without a segment. Create a
segment first.

This command doesn't work in dump mode windows. Turn dumping off first.
This command is not allowed for the current processor. Please choose another
one.

You are about to discard all changes and quit to DOS. Continue?
WRAP
Open Signatures Window


 Action    name: OpenSignatures
 Current hotkey:
<OpenSignatures>

This command opens the signatures window.
```

For each signature the following is displayed:
- name of file with the signature
- state of signature:
    - Planned: the signature will be applied
    - Current: the signature is being applied
    - Applied: the signature have been applied
- number of functions found from the signature
- description of the signature

You can modify the planned signatures list here:
add/delete library modules to be used during the disassembling.

You can not delete an applied signature from the list.

To add a signture to the list for the application press <

You will see list of signatures that can be applied to the program being
disassembled. Not all signature files will be displayed (for example,
32 bit signatures will not be shown for a 16 bit program
 If you want
to see full list of signatures, select the first line of the list
saying SWITCH TO FULL LIST OF SIGNATURES.

The library signatures are located in the
 subdirectory of IDA directory.
Also, environment variable
IDASGN
 may point to the directory with
signatures.
The signatures must have extension
.

See also
[View] submenu.

[How To Use List Viewer]
WRAP
File Viewer

 Action   name: ViewFile
 Current hotkey:
<ViewFile>

You can create an additional window to look at the file contents.
Using this command you can look at files of any size.
To close the opened window, press
<WindowClose

You can edit small files using
[EditFile]
command.

See also
[View] submenu.

WRAP
Functions window

 Action   name: OpenFunctions
 Current hotkey:
<OpenFunctions>

A list of all functions in the program is displayed.
You can
[delete
[modify] functions
using
[list] viewer commands.

Listed for each function are:

```
        - function name
        - segment that contains the function
        - offset of the function within the segment
        - function length in bytes
```

The last column of this window has the following format:

```
        R       - function returns to the caller
        F       - far function
        L       - library function
        S       - static function
        B       - BP based frame. IDA will automatically convert
                  all frame pointer [
] operands to stack
                  variables.
        M       - reserved
        S       - reserved
        I       - reserved
        C       - reserved
        D       - reserved
        V       - reserved
```

See also
[View] submenu.
WRAP
Calculator

 Action   name: Calculate
 Current hotkey:
<Calculate>

A simple calculator is provided. You can enter constant C-style
expressions. Syntax of the expressions is the same is the syntax of
IDC
[expressions

Result is displayed in the message window in three
forms: hexadecimal,decimal and character.

All the names created during a disassembly may be used in these expressions. IDA can also pick up the name or number under the cursor and to store it into the input line.

See also
[View] submenu.

[How to Enter a Number
      WRAP
Processor Type

 Action   name: SetProcessor
 Current hotkey:
<SetProcessor>

Changing the processor type leads to reanalysis of all program. Sometimes this is useful.

Valid processor types are:

```
  8086    - Intel 8086

  80286r  - Intel 80286 real mode

  80286p  - Intel 80286 protected mode

  80386r  - Intel 80386 real mode

  80386p  - Intel 80386 protected mode
(IBM PC line)
  80486r  - Intel 80486 real mode

  80486p  - Intel 80486 protected mode

  80586r  - Intel Pentium
 MMX real mode

  80586p  - Intel Pentium
 MMX prot mode

  80686p  - Intel Pentium Pro
 MMX

  8085    - Intel 8085

  z80     - Zilog 80
(Zilog 80 line)
  64180   - Hitachi HD64180

  860xr   - Intel 860 XR
(Intel 860 line)
  860xp   - Intel 860 XP

  8051    - Intel 8051                     (Intel 51 line)
  m6502   - 6502                           (65xx line)
  pdp11   - DEC PDP/11                     (PDP line)
  68000   - Motorola MC68000
  68010   - Motorola MC68010

  68020   - Motorola MC68020

  68030   - Motorola MC68030

  68040   - Motorola MC68040
(Motorola 680x0 line)
  68330   - Motorola CPU32 (68330)

  68882   - Motorola MC68020 with MC68882

  68851   - Motorola MC68020 with MC68851

  68020EX - Motorola MC68020 with both

  6800    - Motorola MC6800

  6801    - Motorola MC6801

  6803    - Motorola MC6803

  6301    - Hitachi HD 6301

  6303    - Hitachi HD 6303

  6805    - Motorola MC6805
(Motorola 8bit line)
  6808    - Motorola MC6808

  6809    - Motorola MC6809

  6811    - Motorola MC6811

  java    - java                           (Java line)
```

You can change processor type only within the current line. If you have selected IBM PC line, you can't select Zilog 80 line
and vice versa.

Please note that when you change the processor type, IDA may change the

[target assembler
 so check it out.

You may get a message telling that IDA does not know the specified processor if IDA fails to load corresponding processor module.

```
        WIN32  IDA uses .W32 modules
        OS/2   IDA uses .DLL modules
        32 bit IDA uses .D32 modules
```

See also
[Options] submenu.
WRAP
Create a New Segment

 Action   name: CreateSegment
 Current hotkey:
<CreateSegment>

This command allows you to create a new segment.

If you select an area using the
[anchor
 IDA will propose the start
address and the end address of the selection as defaults for the segment
bounds.

If another segment already exists at the specified addresses,
the existing segment is truncated and the new segment lasts from the
specified start address to the next segment
(or specified end address, whatever is lower
 If the old and the new segments
have the same base address, instructions/data will not be discarded by IDA,
otherwise IDA will discard all instructions/data of the new segment.

An additional segment
may be created to cover the area after end of the new segment.

Please note that segments have so called
[addressing mode

The addressing mode is effective only for PC processors.

See also
[Edit|Segments] submenu.
WRAP
Change Segment Attributes

 Action   name: EditSegment
 Current hotkey:
<EditSegment>

This command allows you to change segment attributes.
You can change all attributes except the segment base. To change the segment
base you should delete segment and create it again.

[How to change segment name]

[How to change segment class]

[How to change segment addressing mode (

[How to change segment alignment]

[How to change segment combination]

Changing the segment class may change the segment type.

MOVE ADJACENT SEGMENTS: means that previous and next segments will
be shrunk or expanded filling appearing gaps between segments when
you change the current segment.

DISABLE ADDRESSES: when you shrink the segment, completely removes
all information about bytes going out of the segment.
Otherwise IDA will discard information about instruction/data,
comments etc, but retain byte values so that you will be able to create
another segment afterwards. This checkbox is not meaningful you have checked
'move adjacent segments' checkbox.

If IDA creates 2 segments where only
one segment should exist, you may try
the following sequence:

-
[delete] one segment. Choose one with
bad segment base value. Do not disable addresses occupied
by the segment being deleted.
- change bounds of another segment.

Note that
[create] segment command changes bounds of
overlapping segment automatically.

See also
[Edit|Segments] submenu.
WRAP
Delete a Segment

Action   name: KillSegment
Current hotkey:
<KillSegment>

This command allows you to delete a segment.

IDA will ask your permission to disable the addresses occupied by the
segment.
If you allow this operation, information about the segment will
be deleted. In other words, IDA will discard the information about
instruction or data, comments etc.

If you check
disable addresses
checkbox, IDA will and mark the addresses
occupied by the segment as
nonexistent
in the program. You will lose *
*
information, including byte values.

It is impossible to disassemble the content of adresses not located in any
segment, therefore you must create a new segment if you want to resume the
disassembly of that part of the code.

Also, you can edit (see below) an adjacent segment to [expand] it to those addresses.

IDA will ask your permission to disable addresses occupied by the segment. If you give your permission, ALL information about the segment will be deleted, otherwise IDA will discard information about instruction/data, comments etc, but retain byte values so that you will be able to create another segment afterwards.

To disassemble the addresses occupied by the segment you need to create a new segment again (
. you can't disassemble bytes without a segment

Also, you can
[expand] another adjacent segment to
these addresses.

See also
[Edit|Segments] submenu.
WRAP
Moving Segment

If you are sure, please confirm (press Enter

Otherwise, press Esc.

Caution: moving the first segment of the program will
delete all information about the bytes between the old start of the segment and the new start of the segment!

See also another command that changes segment bounds:
[Edit Segment]
and

[Edit|Segments] submenu.
WRAP
Deleting a Segment

If you want to delete the segment, please mark 'CONFIRM DELETION' checkbox

Disable addresses checkbox

CAUTION: ALL INFORMATION ABOUT THE SEGMENT WILL BE LOST!

If you disable the addresses occupied by the segment, all information about these addresses will be lost. You will not see them on the screen anymore.

Otherwise, the segment will be deleted, but its data will rest unchanged.
You shall create another segment(
) for these addresses using

[Create a New Segment] command.

Select additional file to load

```
Please make the current item undefined first.
Please enter floating point constant
:1379

Add/Edit enum type


~ame :
~lace:




~exadecimal:
~ecimal    :
~ctal      :
~inary     :
~haracter  :




~igned:
:1379

Bad enum name
 Probably the name is invalid or is used in the program.
:1333

Do you really want to delete enum
:1384

Add enum member

  Enum:
%A


~ame :
~alue:
:1725

Edit enum member

  Enum:
%A
  Value:
%D


Bad enum member name
 Probably the name is invalid or is used in the program.
:1384

Bad enum member value
. This value is already used in the enum.
internal error: bad enum id %

Enter address to jump (hex or name)
```

```
Choose segment to jump

Choose address to jump
Choose a name

Choose an entry point

Enter value to search %
Searching %s for value %

Binary search

Enter search
) string:
~tring:



-sensitive:

>

~ecimal:
>

~ctal:

Bad binary string format
Search completed. Found at %08lX.
Search failed.
Searching %s for binary string %
Enter line number

Choose base for offset

Choose a structure for offset

Choose enum
WRAP
Choose segment


Select any line. By default, the cursor
is on the current segment.

You can use <
<Down
<PgUp
<PgDn
<Home
> keys. If you
select the first line of the list,
nothing will be selected.

Press <Enter> to select line,
> to cancel.
```

See also
[other segment related commands]
WRAP
Change Segment Name

Enter new name for the segment.
Segment name is up to 8 characters long.
IDA does check it for good shape now.
Try to give mnemonic names.

See also other
[SegEdit] form fields.
WRAP
Change Segment Addressing

You can choose between 16-bit and 32-bit segment addressing.
This is meaningful only for 80386 processors.

All previous information about the segment will be lost!

Never do it if you are not sure. It may have irreversible consequences,
all instructions/data will be converted to undefined bytes.

See also other
[SegEdit] form fields.
WRAP
Segment Class Name

The Segment Class Name identifies the segment with a class name
(such as CODE, FAR
DATA, or STACK
The linker places segments with
the same class name into a contiguous area of memory in the run-time
memory map.

Changing the segment class changes only the segment definition
on the screen. There are the following predefined segment class names.
If you change segment class and the segment type is
Regular
,
then the segment type will be changed accordingly:

```
        CODE    -       Pure code
        DATA    -       Pure data
        CONST   -       Pure data
        BSS     -       Uninitialized data
        STACK   -       Uninitialized data
        XTRN    -       Extern definitions segment
```

Attention
Segment class names are not deleted. Once you
define a segment class name, you can't reuse it as a name of another object.

See also other
[SegEdit] form fields.
WRAP
Change Segment Alignment

Alignment: select between abs,byte,word,dword,para,page

You can specify the segment alignment for the selected
segment. By default, IDA assumes 'byte' alignment.
Changing the alignment changes only the segment definition
on the screen. Nothing else will happen.

See also other
[SegEdit] form fields.
WRAP
Change Segment Combination

Combination

A field that describes how the linker can combine the
segment with other segments. Under MS-
, segments with
the same name and class can be combined in two ways: they
can be concatenated to form one logical segment, or they
can be overlapped. In the latter case, they have either the
same starting address or the same ending address, and they
describe a common area in memory. Values for the field
are:

Private.	Do not combine with any other program segment.
Public.	Combine by appending at an offset that meets the alignment requirement.
Stack.	Combine as for Public. This combine type forces byte alignment.
Common.	Combine by overlay using maximum size.

Changing segment combination changes only the segment definition
on the screen. Nothing else will happen.

See also other
[SegEdit] form fields.
WRAP
Change Segment Translation

Action   name: SegmentTranslation
Current hotkey:

<SegmentTranslation>

A segment translation is a sequence of another segments to use
when resolving the references to instructions from the current segment.
This command is useful only for 16-bit segmented processors.
I hope it is easier to give an example than to give a formal definition.
Suppose we have 3 segments:

```
        start   end
A       0000    1000
B       1000    2000
C       3000    4000
```

The instruction

```
call    1000
```

in the segment C obviously refers to the segment B while the instruction

```
call    500
```

refers to the segment A.

But IDA does not try to link these references unless you tell to do so:
include the segments A and B into a translation list of the segment C.
That is, you should create a translation list

```
A B
```

for the segment C.
Below is a more complicated example:

```
        start   end
A       0000    1000
B       1000    2000
C       1000    2000
D       3000    4000
E       3000    4000
```

translations

```
B:      A
C:      A
D:      A B
E:      A C
```

allow you to emulate overlays (the first set is A B D, the second A C E)

See also
[Edit|Segments] submenu.
WRAP
Jump to the Specified Segment

Action   name: JumpSegment
Current hotkey:
<JumpSegment>

This command jumps to the start of the selected segment.
IDA will ask you to select a target segment.
After:
  The current address is saved in the 'Jumps Stack

  Cursor is positioned to the specified segment start.

The
[Return] command (usually Esc) will return you back.

See also:
[How to choose a segment]
[Other segment related commands]
[Navigate|Jump to
 submenu.
WRAP
Jump to the Specified address

 Action   name: JumpAsk
 Current hotkey:
<JumpAsk>

This command jumps to the specified address in the program.
IDA will ask you a target address.
You can enter name or address in hexadecimal format
with or without segment.
If you enter a valid address then:
  The current address is saved in the 'Jumps Stack

  Cursor is positioned to the specified address.

The
[Return] command (usually Esc) will return you back.

See also
[Navigate|Jump to
 submenu.

[How to Enter an Address
WRAP
Jump to the Specified Segment Register Change Point

 Action   name: JumpSegmentRegister
 Current hotkey:
<JumpSegmentRegister>

This command jumps to the selected

[Segment Register Change Point

IDA will ask you to select a target change point.
And after:
  The current address is saved in the 'Jumps Stack

Cursor is positioned to the specified change point.

The
[Return] command (usually Esc) will return you back.

See also
[Navigate|Jump to
submenu.
WRAP
Segment Register Change Points

When IDA encounters an instruction which changes a segment register, it creates a change point.
, mostly change points
are maintained by IDA itself. IDA assumes that between change points the segment registers do not change their values. If you find out that IDA failed to locate a segment register changing, or you want to change a register value,
you can create a change point using
[Change Segment Register]
command. You can change value of a segment register
using
[Set default segment register value] command.

IDA classifies the change points. In the change points list you can see the following postfixes after register values:

```
 a (auto)     - Created by IDA. May be
                changed by IDA afterwards.
 u (by user)
- Created by user. IDA will
                not change it.
```

IDA generates the appropriate 'assume' instructions for the change points.
WRAP
Jump to the Next Address from the Problems List

Action   name: JumpQ
Current hotkey:
<JumpQ>

This command jumps to the next address from the

[Problems List

If the Problems List is empty, you will hear a beep.
Otherwise, IDA will display a brief description of the problem and will jump to it.
The
[Return] command (usually Esc) will return you back.

See also
[Navigate|Jump to
submenu.

```
Operands do not match, old:
, new:
Operands do not match, old:
, new:
, difference:

Enter additional anterior lines

Enter additional posterior lines

Can't rename byte at %08lX as
' because this byte can't have a name (it is a tail byte

Can't rename byte at %08lX as
' because the name has a reserved prefix.
              Name                Address Pub
WRAP
Jump to Cross Reference


 Action    name: JumpXref
 Current hotkey:
<JumpXref>

This command shows you a list of cross-references to the current location
and you can jump to the selected one by pressing Enter.

See also
[Navigate|Jump to
 submenu.

[Jump to Cross References to Operand]
WRAP
Jump to Cross References to Operand


 Action    name: JumpOpXref
 Current hotkey:
<JumpOpXref>

This command shows you a list of cross-references to the current operand
and you can jump to the selected one by pressing Enter.

See also
[Navigate|Jump to
 submenu.

[Jump to Cross References]
WRAP
Jump to Function


 Action    name: JumpFunction
 Current hotkey:
<JumpFunction>

This command shows you a list of functions
```

```
and you can jump to the selected one by pressing Enter.

See also
[Navigate|Jump to
 submenu.

Can't create Indirect Jumps Table at %08lX size %
startitem 1


Segment Register Value

<Segment Register :
<Value            :

Segment Default Register Value

<Segment Register :
<Default Value    :


This value will be used when the
register value is not known.

Jump Table


~ddress :
~ize    :


  Size of table element:

>

WRAP
Jump to Entry Point


 Action    name: JumpEntryPoint
 Current hotkey:
<JumpEntryPoint>

This command shows you a list of entry points
and you can jump to the selected one by pressing Enter.

See also
[Navigate|Jump to
 submenu.
-defined offset
Struc~
~ offset
WRAP
Problems List


The following problems exist:
```

```
't find offset base]
't find name]
't find alternative string for an operand]
't find comment]
't find references]
[Indirect execution flow]
't disassemble]
[Already data or code]
[Execution flows beyond limits]
[Too many lines]
[Attention! Probably erroneous situation]
[Decision to convert to instruction/data is made by IDA]
WRAP
Problem: Can't find offset base


 Description:
        The current item has an operand marked as an offset,
        but IDA can't find the offset base in the database.

 Possible reason(

        Probably, IDA base is corrupt.
        This may occur after database repairing.

 What to do:
        Mark the operand again as an offset. Use one of the following
        commands:

[Convert to offset (


[Convert to offset (


[Convert to offset by any segment]

[Convert to offset by any user-specified base]
WRAP
Problem: Can't find name


 Description:
        Two reasons can cause this problem:
          1.Reference to an illegal address is made in the program being
            disassembled;
          2.IDA couldn't find a name for the address but
            it should exist.
 What to do:
        1. If this problem is caused by a reference to an illegal address
            - try to enter the operand
[manually]
            - or make the illegal address legal
[creating] a new segment.
        2. Otherwise, database is corrupt, the best thing to do is
           to
[reload] the database.
```

WRAP
Problem: Can't find alternative string for an operand

Description:
The current item has an operand marked as entered manually,
but IDA can't find the manually entered string in the database.

Possible reason(

IDA base is corrupt. This may occur after database repairing.

What to do:
Enter the operand manually again. Use one of the following
commands:

[User-defined operand]
WRAP
Problem: Can't find comment

Should not happen!
Please inform the author if you encounter this problem.
WRAP
Problem: Can't find references

Description:
The current item is marked as referenced from other place(
) in the
program, but IDA can't find any reference to it.

Possible reason(

IDA base is corrupt. This may occur after database repairing.

What to do:
Database is corrupt, the best thing to do is
to
[reload] the database.
WRAP
Problem: Indirect execution flow

Description:
Actually, this is not a problem. IDA warns you,
that here it encountered an indirect jump and
can't follow the execution.

What to do:
Nothing, that is only for your information
WRAP
Problem: Can't disassemble

Description:
IDA can't disassemble the specified bytes as an instruction.

        Probably, you should choose another processor type.

 Possible reason(

        1. The specified bytes do not contain instruction.
        2. The current
[processor] type is not correct.

 What to do:
        If you are sure that the specified bytes contain an instruction,
        you can try to change
[processor] type and
        mark this bytes as instruction using the following command:

[Convert to instruction]
WRAP
Problem: Already data or code

 Description:
        IDA can't convert this byte(
) to an
[instruction]
        or
[data] because it would overlap another instruction.

 What to do:
        Make following instruction or data 'unexplored'
        using
[undefine] command.
WRAP
Problem: Execution flows beyond limits

 Description:
        IDA encountered a jump or call instruction to an illegal address.
        Namely:
          - jump/call beyond program segments
          - near jump/call beyond the current segment
 What to do:
        1. Enter the operand
[manually]
        2. or Create a new
[segment] making the illegal address legal
        3. or Change the current segment bounds using one of the following:

[How to change segment attributes]

[How to move a segment start]
WRAP
Problem: Too many lines

 Description:
        The current item (instruction or data) can't be
        represented in the text form using 500 lines.

 What to do:
        1. If the current item is an
[array] or
[ASCII] string, try to divide it.
     or 2. Delete
[Additional comment lines]
     or 3. Disable
[cross-references] display
WRAP
Problem: Attention! Probably erroneous situation

 Description:
        The value of the stack pointer at the end of a function is different
        from the value at the start of the function. IDA checks for the
        difference only if the function is ended by
return
 instruction.
        The most probable cause is that stack tracing has failed.

 What to do:
        1. Examine value of
[stack pointer] at various locations of the
           function and try to find out why the stack tracing has failed.
           Usually it fails because some called function changed the
           stack pointer (by purging the input parameters, for example)
        2. If you have found the offending function,
[change] its attributes
           (namely, number of bytes purged upon return

        3. Another way is to specify manually how the stack pointer is
           modified. See
[Change stack pointer] command

WRAP
Problem: Decision to convert to instruction/data is made by IDA

 Description:
        Really this is not a problem. Only for your convenience IDA
        collects all the locations where it has made decision to convert
        undefined bytes to instructions or data even if they don't have
        any references to them. We consider this decision as dangerous
        and therefore we provide you with a way to examine all such places.

 What to do:
        Examine the result of conversion and modify the instructions
        or data if IDA has made a wrong conversion.
WRAP
Indirect Jumps Table

 Action   name: MakeJumpTable
 Current hotkey:
<MakeJumpTable>

When you find indirect jumps by table you can inform IDA

about address and size of this table. In this case IDA will
continue analysing of the program using the table. For example,
it will convert to code all addresses referenced by the table.

You should stand on the instruction which uses the jump
table and you should specify address of the table, its size and
size of table element.

See also
[Edit|Other] submenu

[How to Enter a Number]

[How to Enter an Address
WRAP
Create alignment directive

Action   name: MakeAlignment
Current hotkey:
<MakeAlignment>

This command allows you to create an alignment directive.
The alignment directive will replace a number of useless bytes
inserted by the linker to align code and data to paragraph boundary
or any other address which is equal to a power of two.

You can select an area to be converted to an alignment directive.
If you have selected an area, IDA will try to determine a
correct alignment automatically.

See also
[Edit|Other] submenu

[How to Enter an Address
WRAP
Bad Indirect Jumps Table

The specified table is overlapped with some other object
(any data,code instructions) or table address is invalid.
WRAP
Change Segment Register Value

Action   name: SetSegmentRegister
Current hotkey:
<SetSegmentRegister>

Meaningful only for Intel 80x86 processors.

This command creates or updates a segment register

[change point

See

[jump to segment register change point] for more info.

 See also
[Edit|Segments] submenu.
 See also
[How to enter segment value

 See also
[How to Enter a Number
WRAP
Set Default Segment Register Value

 Action   name: SetSegmentRegisterDefault
 Current hotkey:
<SetSegmentRegisterDefault>

Meaningful only for Intel 80x86 processors.

You can specify a default value of a segment register for the current
segment.
When you change the default value, IDA will reanalyze the segment,
taking the default value when it can't determine the actual value
of the register. This takes time, so do not be surprised if references
are not corrected immediately.

To specify other than default value of a segment register you can
use
[change segment register value] command.

 See also
[Edit|Segments] submenu.
 See also
[How to enter segment value
WRAP
IDC: Built-in functions

The following conventions are used in this list:
' is a linear address
  'success' is 0 if a function fails, 1 otherwise
  'void' means that function returns no meaningful value (always 0)

All function parameter conversions are made automatically.
:1385[Index of IDC functions]
WRAP
IDC language

IDC language is a C-like language. It has the same lexical tokens as C does:
character set,constants,identifiers,keywords, etc.
A program in IDC consists of function declarations. By default, execution
starts from a function named 'main

Select a topic to read:

```
[Variables]
[Functions]
[Statements]
[Expressions]
[Built-in functions]
WRAP
IDC: Expressions

                                        See also
[Built-in functions]

In expressions you can use almost all C operations except:
  complex assigment operations as
(comma operation)

You can use the following construct in the expressions:

  [
]

This means to calculate linear (effective) address for segment '
' offset '

The calculation is made using the following formula:

  (
+ o

If a string constant is specified as '
 it denotes a segment by its name.

There are 3 type conversion operations:

  long( expr )          float number is truncated during conversion
  char( expr )
  float( expr )

However, all type conversions are made automatically:
- addition:
        if both operands are strings,
          string addition is performed (strings are concatenated

        if floating point operand exists,
          both operands are converted to floats;
        otherwise
          both operands are converted to longs;
- subtraction/multiplication/division:
        if floating point operand exists,
          both operands are converted to floats;
        otherwise
          both operands are converted to longs;
- comparisions (
, etc

        if both operands are strings, string comparision is performed;
        if floating point operand exists,
          both operands are converted to floats;
        otherwise
          both operands are converted to longs;
- all other operations:
        operand(
) are converted to longs;
WRAP
IDC: Statements


In IDC there are the following statements:
[expression
        (expression-statement)
  if (expression) statement
  if (expression) statement else statement
  for ( expr1; expr2; expr3 ) statement
  while (expression) statement
  do statement while (expression

  break;
  continue;
  return <expr

  return;              the same as 'return 0
{ statements
;                    (empty statement)
WRAP
IDC: Functions


A function in IDC returns a value. There are 2 kinds of functions:
[built-
] functions
  - user-defined functions

A user-defined function is declared in this way:

  static func(arg1,arg2,arg3)
{


[statements]


  }

where arg1,arg2,arg3 are the function parameters
func' is the function name.
It is not nesessary to specify the types of the parameters
because any
[variable] can contain a string or a number.
WRAP
IDC: variables


All variables in IDC are
automatic local variables (
 A variable can contain:
```

- a 32-bit signed long integer
  - a character string (max 255 characters long)
- a floating point number (extra precision, up to 25 decimal digits)

A variable is declared in this way:

  auto var;

This declaration introduces a variable named '
 It can contain a string
or a number. All C and C
 keywords are reserved and cannot be used as
a variable name. The variable is defined up
to the end of the
[function
WRAP
Create MAP File

 Action   name: ProduceMap
 Current hotkey:
<ProduceMap>

Enter a file name for the map. IDA will write the following
information to this file:
  current segmentation
  list of names sorted by values
    (automatically generated names are not listed)

You can use this map file for your information and you can
use it for debugging (for example, Periscope from Periscope
Company or Borland's Turbo Debugger can read this file

See also other
[Produce output file] commands.
WRAP
Dump database to IDC file

 Action   name: DumpDatabase
 Current hotkey:
<DumpDatabase>

This command saves current IDA database into a text file.

You can use it as a safety command:
- to protect your work from disasters
  - to migrate information
    into new database formats of IDA.

This command is used when you want to switch to a new version
of IDA. Usually each new version of IDA has its own database
format. To create a new format database, you need:
  1. to issue the 'Dump
 command for the old
     database (using old version of IDA

You will
     get an IDC file containing all information
     from your old database.
  2. to reload your database using new IDA with switch -
.
  3. to compile and execute the IDC file with command
    'Execute IDC file'
(usually F2)

Please note that this command doesn't save everything to text file.
Information about the local variables will be lost!

See also other

[Produce output file] commands.
WRAP
Create ASM File

Action   name: ProduceAsm
Current hotkey:
<ProduceAsm>

Enter a file name for the assembler text file. IDA will write
the disassembled text to this file.

If you've selected an area on the screen using
[Drop Anchor]
command, IDA will write only the selected area (from the
current address to the anchor

If some I/O problem (
. disk full) occurs during writing
to this file, IDA will stop - a partial file will be created.

Please note that
:1399[demo] version can't produce assembler
files.

See also other
[Produce output file] commands.
WRAP
Create LST File

Action   name: ProduceLst
Current hotkey:
<ProduceLst>

Enter a file name for the assembler listing file. IDA will write
the disassembled text to this file.

If you've selected an area on the screen using
[Drop Anchor]
command, IDA will write only the selected area (from the
current address to the anchor

If some I/O problem (
. disk full) occurs during writing
to this file, IDA will stop - a partial file will be created.

Please note that
:1399[demo] version can't produce assembler
listing files.

See also other
[Produce output file] commands.
WRAP
Create Executable File

 Action   name: ProduceExe
 Current hotkey:
<ProduceExe>

Enter a file name for the new executable file. Usually this
command is used after patching (see commands
[Patch byte] and

[Patch word
 to obtain a patched version of the file.

IDA produces executable files only for:
- MS DOS .exe
  - MS DOS .com
  - MS DOS .drv
  - MS DOS .sys
  - general binary
  - Intel Hex Object Format
  - MOS Technology Hex Object Format

For other file formats please create a
[difference] file.

NOTE: only
[Patch byte/word] commands affect the executable
file contents, other commands
(including
[User-Specified String for the #th Operand

will not affect the content of the disassembled file.

EXE files: Output files will have the same EXE-header and relocation table as
the input file. IDA PRO will fill unused areas of the EXE file (
.
between relocation table and loadable pages) with zeroes.

See also
[File|Produce output file] submenu,

[Edit|Patch core] submenu.
Create Difference File

Action   name: ProduceDiff
 Current hotkey:
<ProduceDiff>

This command will prompt you for a filename and then will create a plain text difference file of the following format:

```
comment

filename
offset: oldval  newval
```

See also
[File|Produce output file] submenu,

[Edit|Patch core] submenu.
WRAP
IDC Command

 Action   name: ExecuteLine
 Current hotkey:
<ExecuteLine>

You can enter and execute a small script written
in the built-in language named IDC.

Here is the list of
:1385[built-
] functions.

See also
] language overview
         Execute
[IDC file] command

[other File
 submenu] commands
         How to use
[notepad
WRAP
IDC File

 Action   name: Execute
 Current hotkey:
<Execute>

    You can select and run an
] script file.

See also
[Immediate execution] of IDC commands

[other File

submenu] commands.
WRAP
Patching the Image

Action name: PatchByte, PatchWord
Current hotkey:
<PatchByte
<PatchWord>

You can modify executable file and eventually
[generate a new] file.
You may modify
[unexplored] items only.

If you patch bytes, then you may enter multiple bytes.
Follow this
[link] to learn about format of the input
string.

You can create a
[difference] file too.

See also
[Edit|Patch core] submenu.

[How to Enter a Number
WRAP
Auto analysis is not completed

Till auto analysis is not completed, the IDA database is not consistent:
- not all
[cross-references] are found
- not all
[instructions] are disassembled
- not all
[data] items are explored

See also:
[auto analysis]
[setup auto analysis]
[wait until end of analysis]
WRAP
Cannot Create File

Probably this file exists and has
read-only attribute. Try to delete it
or clear this attribute.

Another reason is that you have too few
number in FILES
statement of your
CONFIG.SYS file. IDA requires about 10
file handlers to be free.
WRAP

Disk Write Error

May be your DISK IS FULL? Try to
delete some unnecessary files.

Probably your disk contains unmarked BAD
SECTORS. This is very dangerous. I
recommend you to re-format your disk or
to mark somehow these sectors.

Partial file is created.
WRAP
File submenu

In this submenu you can:

[Open]	Open a window
[Load file]	Load additional binary file
[IDA command]	Execute an IDC command
[Produce output file]	Generate output file
[OS shell]	Execute OS commands
[Save database]	Save database in packed form
[Save database as Save database in packed form	in another file
[Abort]	Abort - do not save changes
[Quit]	Quit to DOS - save changes

See also
[Menu Bar] submenus.
WRAP
Main Menu Bar

All IDA commands are available from the followings menus:
[File]
[Edit]
[Navigate]
[View]
[Options]
[Windows]
WRAP
Load additional file

Action   name: LoadFile

Current hotkey:
<LoadFile>

This command loads an additional file into the IDA database.
The new file is appended to the database so that all information in the database is retained.

This command allows you to load binary files only.

See also
[other File
submenu] commands.
WRAP
Open a window

Action name: WindowOpen
Current hotkey:
<WindowOpen>

Opens a new window with disassembled text.
IDA automatically opens one big window
at the start. If you want, you can open other windows.

In IDA there are the following types of windows:

IDAview window	The [main] window type in IDA. is opened using this command.
Messages window	is opened automatically at the start of work, displays various [information]
Segments window	Displays program [segmentation]
Selectors window	Displays program [selectors]
Segment registers	Displays [segment registers]
Names window	Displays all [names] in the program
Functions window	Displays all [functions]
Structures window	Displays all [structures]
Stack variables window	Displays [stack variables for the current function
Enums window	Displays all [enums]
Signatures window	Displays planned [signatures]
Xrefs window	Displays all [cross-references] to the current item
File Viewer	allows you to [view] files of any size
File Editor	allows you to [edit] small files (in memory)

There are dialog boxes of various types too

When you are in the IDAview type of window, you can switch between normal mode and
[dumping] mode.

See also
[other File
 submenu] commands.
WRAP
Jump to the Previously Marked Position

 Action   name: JumpPosition
 Current hotkey:
<JumpPosition>

This command jumps to the selected position.
IDA will ask you to select a target position.
After:
  The current address is saved
  in the
[Jumps Stack

  Cursor is positioned to the
  specified segment start.

The
[Return] command (usually Esc) will return you back.

You can mark position using
[Mark Position] command.
WRAP
Jumps Stack

Each IDA Window has its own jumps stack. This stack keeps cursor locations.
Many IDA commands use jumps stack,
. they save old cursor position to
the stack. For example, when you are at the address 3000:0100 and press key Ctrl-
(find instruction
 the 3000:0100 is saved into the jumps stack
and the search is started. Afterwards, you can return to the old position using
[Return] command.

You can make the jumps stack empty using the
[Navigate|Empty stack]
menu command.
WRAP
Jump Immediate

 Action   name: JumpEnter

Current hotkey:
<JumpEnter>

By pressing
<JumpEnter>
 you nagivate in the program in the same
way as in a hypertext (the way web browers and help screens use

This is the easiest way to explore the program: just position the cursor
at the desried name and press
<JumpEnter>
.

Your current address is saved in the
[jumps stack

The
[Return] command (usually Esc) will return you back.

If the cursor is at a stack variable, a window with

[stack variables] is opened and the definition of the
stack variable is displayed.

See also
[Navigate] submenu.

[Empty Stack] command.
WRAP
'Return' Command

 Action   name: Return
 Current hotkey:
<Return>

This command returns you back to the previous position.
It takes positions from
[Jumps Stack

See also
[Navigate] submenu.

[Undo Return] command.

[Empty Stack] command.
WRAP
Undo the last 'Return' Command

 Action   name: UndoReturn
 Current hotkey:
<UndoReturn>

This command undoes the last
[Return] command.

See also
[Navigate] submenu.

[Empty Stack] command.
WRAP
Empty Stack Command

Action   name: EmptyStack
Current hotkey:
<EmptyStack>

This command makes the
[jumps stack] empty.

See also
[Navigate] submenu.

[Return] command.
WRAP
Navigate|Jump to   submenu

In this menu you can select a command to jump to the specified location in the file. Jumps are very fast and your previous position is saved. This submenu contains the following items:
[Jump to the Specified Address]
[Jump to the Named Location]
[Jump to the Specified Segment Start]
[Jump to the Previously Marked Position]
[Jump to the Specified Segment Register Change Point]
[Jump to the Next Address from the Problems List]
[Jump to the Cross Reference]
[Jump to the Function]
[Jump to the Entry Point]
[How to Mark a Location]

Look also
[Search for
menu for fast navigating.
See
[Jumps Stack] concept.
See
[Navigate] submenu.
WRAP
Navigate|Search for   submenu

In this menu you can select a command to search for something in the file. Searches are relatively slow and your previous position is saved in
[Jumps Stack

You can search for:
[instructions that need your ATTENTION]
(voids)

[instruction bytes]

[data bytes]

[unexplored bytes]

[explored bytes]

[immediate operand values]

[substring in the text representation]

[substring in the binary image of the file]

[bytes not belonging to any function]

in both
[directions]
(up and down

Look also
[Jump to
menu for fast navigating.

See also
[Navigate] submenu.
WRAP
Set Direction for Searches

Action   name: SetDirection
Current hotkey:
<SetDirection>

The current direction for searches is displayed in the right
upper corner of the screen. Using this command you can toggle
it.

See also
[Navigate|Search for
submenu,

[Navigate] submenu,

[Options] submenu.
WRAP
Mark Position

Action   name: MarkPosition
Current hotkey:
<MarkPosition>

You can mark certain locations of the file to be able to
[jump] to
them quickly. Text description of the location may help to find
a desired location easily.

First you select a slot for the mark, after you enter a
description for the location.

If you enter an empty description line, the slot is freed.

See also
[Navigate] submenu,

[Navigate|Jump to
submenu.
WRAP
Search for next void

Action   name: JumpVoid
Current hotkey:
<JumpVoid>
'void' instructions are instructions that need your attention
because they contain an immediate operand that could be a
number or offset. IDA does not know about it, so it marks these
instructions as 'void
You can change 'void' operands
definition using
:1157[set lower limit of voids] and

:1157[set upper limit of voids] commands.

Data arrays are considered to be void if the first element of the
data array is within the lower and upper void limits. Values of other
elements are not examined.

You can also disable display of the 'void' marks using

:1006[toggle voids display] command.

NOTE

I strongly recommend that before producing an ASM file you go
through 'void' marks and get rid of them. After this you have
a certain level of confidence that the file have been
disassembled correctly.

See also
[Navigate|Search for
submenu.
WRAP
Search for next code

Action   name: JumpCode

Current hotkey:
<JumpCode>

This command searches for the first instruction in the

[current direction

See also
[Navigate|Search for
submenu.
WRAP
Search for next data

Action name: JumpData
Current hotkey:
<JumpData>

This command searches for the first data in the

[current direction

See also
[Navigate|Search for
submenu.
WRAP
Search for next unexplored byte

Action name: JumpUnknown
Current hotkey:
<JumpUnknown>

This command searches for the first unexplored byte in the

[current direction

See also
[Navigate|Search for
submenu.
WRAP
Search for next explored byte

Action name: JumpExplored
Current hotkey:
<JumpExplored>

This command searches for the first defined byte
(instruction or data) in the

[current direction

See also
[Navigate|Search for
 submenu.
WRAP
Search for bytes not belonging to any function

 Action   name: JumpNotFunction
 Current hotkey:
<JumpNotFunction>

This command searches for the first byte not beloning to any function in the

[current direction

See also
[Navigate|Search for
 submenu.
WRAP
Repeat search for instruction/data with the specified operand

 Action   name: JumpImmediate
 Current hotkey:
<JumpImmediate>

This command repeats
[search for immediate] command.

See also
[Navigate|Search for
 submenu.
WRAP
Repeat search for substring in the text representation

 Action   name: JumpText
 Current hotkey:
<JumpText>

This command repeats
[search for text] command.

See also
[Navigate|Search for
 submenu.
WRAP
Repeat search for substring in the file

 Action   name: JumpBinaryText
 Current hotkey:
<JumpBinaryText>

This command repeats

[search for text in core] command.

See also
[Navigate|Search for
 submenu.
WRAP
Undefine a byte

 Action   name: MakeUnknown
 Current hotkey:
<MakeUnknown>

This command deletes the current instruction or data converting
it to 'unexplored' bytes. IDA will delete subsequent instructions if
there are no more references to them (functions are never deleted

If you've selected an area using the
[anchor
 all bytes in the area
will be converted to 'unexplored' bytes. Also, IDA won't delete any other
instructions if an area was selected.

See also

[Edit] submenu
WRAP
Wait for end of auto analysis

 Action   name: Wait
 Current hotkey:
<Wait>

This command suspends execution of
[macro] until auto
analysis in ended,
. the auto analysis
[indicator] is 'Ready

See also
[other File
 submenu] commands.
WRAP
Convert to instruction

 Action   name: MakeCode
 Current hotkey:
<MakeCode>

This command converts the current unexplored bytes to instruction(

If it is not possible, IDA will warn you.

If you've selected an area using the
[anchor
 all bytes in the area
will be converted to instructions.

If you apply this command to an instruction, it will be reanalyzed.

See also

[Edit] submenu
WRAP
Convert to data

 Action   name: MakeData
 Current hotkey:
<MakeData>

This command converts the current unexplored bytes to data.
If it is not possible, IDA will warn you.
Multiple using of this command will change data type:

```
        db
 dw
 dd
 float
 dq
 double
 dt
 packreal
    ;
        ^                                                      |    ;

    ;
```

You may remove some items from this list using
[setup data] command.

If the
[target assembler] does not support double words or another
data type, it will be skipped.
To convert back, use
[Undefine] command.
To create an array, use
[Array] command.

See also

[Edit] submenu
WRAP
Convert to ASCII string

 Action   name: MakeAscii
 Current hotkey:
<MakeAscii>

This command converts the current unexplored bytes to a string.

Set of allowed characters is specified in the
:1482[configuration]
file, parameter AsciiStringChars. Character
' is not allowed in any case.
If the current
[assembler] does not allow ASCII characters
with high bit set, characters with high bit set are not allowed.

If the
[anchor] was dropped then IDA takes for the string
all characters between the current cursor position and the anchor.

Use the
[anchor] if the string starts a disallowed character.

This command also generates a
[name] for the string.
In the
:1482[configuration] file you can specify character
translation (XlatAsciiName) and characters allowed in names (NameChars

You can change ASCII string length using
[Array] command.

Pascal Strings

To create Pascal style strings (with first byte indicating string
length) use
:1199[Set Ascii Style] command.

See also

[Edit] submenu
WRAP
Specify Target Assembler

Action   name: SetAssembler
Current hotkey:
<SetAssembler>

This command allows you to change the target assembler,
. the assembler
for which the output is generated. You select the target assembler from
a menu. The menu items depend on the current
[processor type

NOTE: For the moment IDA supports only a generic assembler
for 80x86 processors. We recommend the use Borland's TASM to compile the
output assembler files.
WRAP
Setup Auto Analysis

Action   name: SetAuto
 Current hotkey:
<SetAuto>

 IDA displays a form and asks you to enable or disable auto analysis
and its
[indicator

Usually, auto analysis is enabled.
Disable it if you are sure that this will help you.

Here you may setup various kernel analysis
:1730[options

If the current
[processor] has options, there will be a button
to change processor-specific options too.

Under OS/2 you can specify priority for IDA. It has effect only for the
current
session. You can change the default priority in the
:1482[Configuration file

Please do not set
time critical
 priority unless you know what you are doing.
Your system will not react to any external events.

See also
[Options] submenu.

[auto analysis explanation
WRAP
Edit|Comments submenu

Here you can manipulate with different kinds of comments.
Use them to make disassembled text more understandable.
[Regular comments]
[Repeatable comments]
[Additional comment lines]
[Predefined comments]

See also
[Edit] submenu.
WRAP
Edit|Operand types submenu

Here you can change operand types to offset,number,chars, etc.
Use them to make disassembled text more understandable.

Use this commands to delete
[void] marks.
[Convert to number]

[Convert to hex number]
[Convert to decimal number]
[Convert to binary number]
[Convert to octal number]
[Convert to character]
[Mark as variable]
[Convert to segment]
[Convert to offset (
[Convert to offset (
[Convert offset by any segment]
[Convert offset by any user-specified base]
[Convert to struct offset]
[Convert to enum]
[Convert to stack variable]
[Change sign of operand]
[User-defined operand]

See also
[Edit] submenu.
WRAP
Functions submenu

[Make a function

[Edit a function

[Delete a function

[Set function end]

[Define stack variables

[Change stack pointer

See also
[Edit] submenu.
WRAP
Structures submenu

Commands of this submenu are available in the

[structures window

[Add a struct]

[Delete a struct]

[Move a struct]

[Declare struct var]

Please use regular commands to specify struct members, their types, comments etc.

See also
[Edit] submenu.
WRAP
Define a new structure

 Action   name: AddStruct
 Current hotkey:
<AddStruct>

This command defines a new structure.
The new structure is created zero length. You should add structure members using
[structure] manupulation commands.

These command is available when you open a structures
[window

You can add a new members to the structure using the following commands:

	command	hotkey
	[make data]	<MakeData>
	[make ascii]	<MakeAscii>
	[make array]	<MakeArray>
	[rename]	<Rename>

See also
[Edit|Structs] submenu.

[How to Enter an Identifier]
WRAP
Delete a structure

 Action   name: DelStruct
 Current hotkey:
<DelStruct>

This command deletes the current structure.

Beware, when you delete a structure all references to it will be destroyed,
Even if you recreate it later, you'll have to respecify all references
to it.

These command is available when you open a structures
[window

See also
[Edit|Structs] submenu.
WRAP
Rename a structure/member

IDA maintains separate namespaces for each structure. For example, you can
define
something like this:

```
 xxx    struc
 xxx    db ?
 xxx    struc
```

Beware, usually
assemblers have one common namespace and do not allow the mentioned above
example.

You can't specify an empty name.

These command is available when you open a structures
[window

See also
[Edit|Structs] submenu.

[How to Enter an Identifier]
WRAP
Move a structure

 Action   name: MoveStruct
 Current hotkey:
<MoveStruct>

This command moves the current structure type to another place.

Each structure has its ID and a serial number. The ID is a number used to
refer to the structure, while a serial number is used to order structures
during output. Changing the serial number moves the structure to another
place.

The serial number of a structure is displayed at the lower left corner of the
window.

You can specify any number, IDA will move the structure to the specified
place.
        1 - the current structure becomes the first structure

        2 - to the second place

These command is available when you open a structures
[window

See also
[Edit|Structs] submenu.

[How to Enter a Number
WRAP
Declare a structure variable

 Action   name: MakeStructVar
 Current hotkey:
<MakeStructVar>

This command declares a variable of the specified structure type.

IDA will ask you to choose a structure type. You should have some structure types
[defined] in order to use this command.

See also
[Edit|Structs] submenu.
WRAP
Delete a structure member

 Please remember that deleting a member deletes also all information
about this member, including comments, member name etc.

See also
[Edit|Structs] submenu.
WRAP
Make Function

 Action   name: MakeFunction
 Current hotkey:
<MakeFunction>

This command defines a new function.

You can specify function boundaries using the
[anchor

If you do not, IDA will try to find the boundaries automatically:
   - function start point is equal to the current cursor position;
   - function end point is calculated by IDA.

A function can't contain references to undefined instructions.
If a function has already been defined at the specified addresses,
IDA will jump to its start address, showing you a warning message.

A function must start with an instruction.

See also
[Edit|Functions] submenu.

[Edit a function]

[Delete a function]
WRAP
Edit a Function

Action name: EditFunction
Current hotkey:
<EditFunction>

Here you can change function bounds, its name and flags.
In order to change function end address you could use
[FunctionEnd]
command.

If the current address doesn't belong to any function, IDA beeps.

Also this command allows you to change function frame parameters.
You can change size of some parts of frame structure.

IDA considers the stack as the following structure:

```
      | function arguments             |

      | return address                 |

      | saved registers (
)     |

 BP
      | local variables                |

 SP
```

You may specify number of bytes in each part of stack frame. Size of the
return address is calculated by IDA itself.
BP based frame
allows IDA to automatically convert [
] operands
to
[stack variables

If you press <Enter> even without changing any parameter
then IDA will reanalyse the function.

See also
[Edit|Functions] submenu.

[Make a function]

[Delete a function]
WRAP
Delete a Function

 Action   name: DelFunction
 Current hotkey:
<DelFunction>

Deleting a function deletes only information about a function,
nothing else.

See also
[Edit|Functions] submenu.

[Make a function]

[Edit a function]
WRAP
Set Function End

 Action   name: FunctionEnd
 Current hotkey:
<FunctionEnd>

Changes the current or previous function bounds so that its end
will be set at the cursor. If it is not possible, IDA beeps.

See also
[Edit|Functions] submenu.

[Make a function]

[Edit a function]
WRAP
Retrieve predefined comment

 Action   name: MakePredefinedComment
 Current hotkey:
<MakePredefinedComment>

You can retrieve predefined comment for instructions.

For this, you should specify the instruction mnemonics
and so-called
comment index

.

Comment Index for Instruction

The correct syntax is:

```
  Reg
Value


where '
' can be

        - register name (
)
        -
- for first operand
        -
- for second operand

and 'value' is a valid C number.

For example:

  int
  Op1
0x21 AX
0x4C01

means

        int     21h     where AX
4C01h

if the program being disassembled is a MS DOS executable,
the predefined comment will be:

        DOS -
- QUIT WITH EXIT CODE (EXIT)
        AL
 exit code
```

See also
[Edit|Comments] submenu.
WRAP
Immediate Operand
 Decimal

 Action   name: OpDecimal
 Current hotkey:
<OpDecimal>

This command converts immediate operand(
) type of the
current instruction/data to decimal. Therefore, it

becomes a 'number

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are affected.

If an area was selected using the
[anchor] then all
[void]
operands will be converted to decimal numbers.
Other (
-void) operands will not be affected.

See also

[Edit|Operand types] submenu.
WRAP
Immediate Operand
 Octal

 Action   name: OpOctal
 Current hotkey:
<OpOctal>

This command makes the current instruction or data operand type octal. IDA always uses 123o notation for octal numbers even if the current assembler does not support octal numbers.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are affected.

If an area was selected using the
[anchor] then all
[void]
operands will be converted to octal numbers.
Other (
-void) operands will not be affected.

See also

[Edit|Operand types] submenu.
WRAP
Immediate Operand
 Binary

Action name: OpBinary
Current hotkey:
<Opbinary>

This command makes the current instruction or data operand type binary.
IDA always uses 123b notation for binary numbers even if the current
assembler does not support binary numbers.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

If an area was selected using the
[anchor] then all
[void]
operands will be converted to binary numbers.
Other (
-void) operands will not be affected.

See also

[Edit|Operand types] submenu.
WRAP
Immediate Operand
Character

Action name: OpChar
Current hotkey:
<OpChar>

This command converts immediate operand(
) type of the
current instruction/data to character.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

If an area was selected using the
[anchor] then all
[void]
operands will be converted to character constants.
Other (
-void) operands will not be affected.

See also

[Edit|Operand types] submenu.

WRAP
Immediate Operand
 Number

 Action   name: OpNumber
 Current hotkey:
<OpNumber>

This command converts immediate operand(
) type of the
current instruction/data to a number. So you can
delete
[void] mark of the item.

The number is represented in the default radix for the current processor
(usually hex, but octal for PDP-
, for example)

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

If an area was selected using the
[anchor] then all
[void]
operands will be converted to numbers.
Other (
-void) operands will not be affected.

See also

[Edit|Operand types] submenu.
WRAP
Immediate Operand
 Hex Number

 Action   name: OpHex
 Current hotkey:
<OpHex>

This command converts immediate operand(
) type of the
current instruction/data to hex number. So you can
delete
[void] mark of the item.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before then the first operand will be affected; otherwise all other operands are affected.

If an area was selected using the
[anchor] then all
[void]
operands will be converted to hexadecimal numbers.
Other (
-void) operands will not be affected.

See also

[Edit|Operand types] submenu.
WRAP
Make Current Byte as a Variable

Action name: MakeVariable
Current hotkey:
<MakeVariable>

This command mark/unmarks the current byte as variable.
Variable bytes have asterisk
in the beginning
of the line.

Use it only for your purposes. IDA ignores
these marks, but creates them for your convenience.

See also

[Edit|Other] submenu.
WRAP
Immediate Operand
Segment

Action name: OpSegment
Current hotkey:
<OpSegment>

This command converts immediate operand(
) type of the
current instruction/data to segment base.

When you use this command, IDA deletes the
[manually] entered operand.

If IDA can't find a segment with the immediate operand's
value, it simply displays it as hex number.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are affected.

If an area was selected using the

[anchor] then all
[void]
operands will be converted to segments.
Other (
-void) operands will not be affected.

See also

[Edit|Operand types] submenu.
WRAP
Immediate Operand
Offset (base DS)

Action name: OpOffset
Current hotkey:
<OpOffset>

This command converts immediate operand(
) type of the
current instruction/data to offset by DS.

If current DS value is unknown (or equal 0xFFFF) IDA
will warn you
it will beep. In this case you should
define DS register value for the current byte. The
best way to do it:
[jump to segment register change point]
[change value of DS]
[return]

or you can
[change default value] of DS for the current
segment.

If you want to delete offset definition, you can use
this command again - it works as trigger.

See also:
[offset by current segment]

[offset by any segment]

[offset by any user-specified base]

[Edit|Operand types] submenu.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

If an area is selected using the
[anchor

IDA will perform
'en masse' conversion. It will convert
immediate operands of all instructions in the selected area to
offsets. However, first IDA will ask you the lower and upper limits of
immediate operand value. If the an operand value is >
lower limit and
<
upper limit then the operand will be converted to offset, otherwise it
will be left unmodified.

See
[Enter #th operand manually] commands.
WRAP
Immediate Operand
Offset (base CS)

Action name: OpOffsetCs
Current hotkey:
<OpOffsetCs>

This command converts immediate operand(
) type of the
current instruction/data to offset by CS.

See also:
[offset by data segment/no offset]

[offset by any segment]

[offset by any user-specified base]

[Edit|Operand types] submenu.

When you use this command, IDA deletes the
[manually] entered operand.

If the cursor is on the first operand (the cursor is before

then the first operand will be affected; otherwise all other operands are
affected.

If an area is selected using the
[anchor
IDA will perform
'en masse' conversion. It will convert
immediate operands of all instructions in the selected area to
offsets. However, first IDA will ask you the lower and upper limits of
immediate operand value. If the an operand value is >
lower limit and
<
upper limit then the operand will be converted to offset, otherwise it
will be left unmodified.

See
[Enter #th operand manually] commands.
WRAP

Move a Segment

 Action   name: MoveSegment
 Current hotkey:
<MoveSegment>

There are 2 commands to change segment boundaries:
        - move a segment (this command)
        -
[edit segment] command

The difference between them is simple:
changing
 segment bounds
does not affect other segments while
moving
 a segment may change
adjacent segments too.

Moving a segment means moving its beginning.
, the proper name
for this command would be 'Expand/Shrink a Segment'
(due to
historical reasons the name is 'move segment
 First you
choose a segment by dropping the
[anchor] anywhere in the segment.
After you go to the desired new beginning of the segment and
call this command. IDA will try to shrink/expand previous
segment in order to move the beginning of the selected segment.
Of course you can't move start of the segment 'too far
- the segment should have at least 1 byte
  - start address of the segment should be less than end of segment
  - no other segments can be killed during moving
  - segment can't have bytes with negative offset

You cannot shrink a segment to 0 bytes. A segment must have
at least 1 byte.

Mainly this command is used when IDA does not detect boundary
between segments correctly.

Sometimes
IDA creates 2 segments where only
one segment should exist. In this case you should not use this command.
Use the following sequence instead:
        -
[delete] one segment. Choose one with bad segment base value.
          Do not disable addresses occupied by the segment being deleted.
        - change bounds of another segment.

See also
[Edit|Segments] submenu.
WRAP
Show Registers

Action   name: ShowRegisters
Current hotkey:
<ShowRegisters>

This command displays segment register contents in the messages window. If the messages window is hidden beyond other windows, the screen will not change.

It is recommended to use this command for window refreshes.

See also
[Edit|Segments] submenu.

[View] submenu.
WRAP
Disable/Enable AutoAnalysis Display

Action   name: ToggleAutoShow
Current hotkey:
<ToggleAutoShow>

This command enables or disables auto
[analysis] display.
The display
is located in the upper right corner of the screen, near the timer.

AU
AutoAnalysis indicator. This indicator represents
the state of AutoAnalysis. Possible values of
this indicator are presented below:

the indicator is turned off. This command turns
on/off this indicator.
AU:
°idle°
AutoAnalysis is finished.
AU:disable   AutoAnalysis is disabled.
FL
address> execution flow is being traced
PR
address> function is being created
AC
address> the code analysis goes through the noted address
LL
number> a signature file is being loaded
L1
address> the first pass of FLIRT
L2
address> the second pass of FLIRT
L3
address> the third pass of FLIRT
FI
address> the final pass of autoanalysis

address> the noted address becomes unexplored number> indication of various activity

See also

[Options] submenu.
WRAP
Dump/normal View

 Action   name: ToggleDump
 Current hotkey:
<ToggleDump>

This command switches the current window mode:
dump view and normal view. At the start, all windows are in the normal mode,
so you can see instructions and data. When you are in the dump mode, no
instructions or data is displayed. All bytes of the program will be dumped
on the screen as plain bytes and in ASCII. This command may be useful
when you want to find something in the program visually.
Do not forget that many commands continue to work. For example, you
can use binary
[search] command.

A mode is a feature of a window, so you can open many windows
and in each of them set its own mode.

See also

[Options] submenu.
WRAP
Setup data types

 Action   name: SetupData
 Current hotkey:
<SetupData>

This command allows you to select the data types used in the round-robin
carousel in
[MakeData] command.

Valid data types are:
```
        - byte
        - word          (2 bytes)
        - double word   (4 bytes)
        - float         (4 bytes)
        - quadro word   (8 bytes)
        - double        (8 bytes)
        - long double   (10 or 12 bytes)
        - packed real   (10 or 12 bytes)
```

Naturally, not all data types are usable for all processors. For example,
Intel 8051 processor doesn't have 'double word' type.

Also, this command allows you to select a data type for the current undefined
item and convert it to a data.

Please note that if the current processor doesn't support a data
type, you can not assign it even if you have selected it.
If you unselect all data types, IDA will use 'byte' type.

See also

[Options] submenu.
WRAP
Show Internal Flags

Action name: ShowFlags
Current hotkey:
<ShowFlags>

This command displays internal flag values for the current byte.

See also
[View] submenu.
WRAP
Navigate submenu

This submenu allows fast access to desired address
or item in the file. You can search for things,
jump to specified addresses etc.
[Jump to
submenu

[Search for
submenu

[Return] command

[Set direction for searches]
[Mark position]

See also
[Menu Bar] submenus.
WRAP
Edit submenu

This submenu allows you to modify text representation
and to patch the file.
Also, here you'll find commands to control program
analysis. There are the following commands:
[instruction]
[data]
[ascii]
[array]
[undefine]
[Rename]
[Operand types] submenu

[Comments] submenu

[Functions]
[Structures]
:1222[Enums]
[Segments]
[Patch program]
[Other]

See also
[Menu Bar] submenus.
WRAP
Windows submenu

Here you can manipulate with windows on the screen.
I guess you do not need help on this topic?

See also
[Menu Bar] submenus.
WRAP
View submenu

Here are commands to open various windows, display information etc.

[View Functions]
[Calculator]
[View general registers]
[Open segment registers window]
[Open segments window]
[Open selectors window]
[Open names window]
[Open xrefs window]
[Open structures window]
[Open enums window]
[Open signatures window]
[Open stack variables window]
[View internal flags]

[View File]
[Edit file]

See also
[Menu Bar] submenus.
WRAP
Options Submenu
[Text representation]
[Cross references representation
[Assembler directives
[Names representation
[Demangled C
 names
[Setup data types

:1199[String styles
:1200[ASCII options
:1207[Colors
[Dump/normal View]
[Processor type
[Target assembler
[Auto analysis]

See also
[Menu Bar] submenus.
WRAP
Edit|Other Submenu

[Rename any address]

[Mark as variable]

[Jump table

[Alignment

See also
[Edit] submenu.
WRAP
Edit|Segments submenu
[How to create a new segment]
[How to delete a segment]
[How to change segment attributes]
[How to move a segment start]
[How to change segment translation]
[Set default segment register value]
[Change segment register value]

See also:
[How to choose a segment]
[How to jump to a segment]
[Edit] submenu.
WRAP
Edit|Patch core submenu

This submenu gives the way of quick patching of the file.
You can easily modify executable file:

        - change a byte
        - change a word

IDA will display the original value, the current value and file offset.
If file offset
 0xFFFFFFFF then the current item comes from a compressed
page (
/NE iterated pages, for example) and/or it is not possible to
tell the file position.

You can create a
[difference] file afterwards.
[Patch core]
[Produce EXE file]
[Produce DIF file]

See also
[Edit] submenu.
WRAP
Jump to the Chosen Name

 Action   name: JumpName
 Current hotkey:
<JumpName>

You can choose a target name.
IDA will display a list of names (sorted by addresses)
and you can choose a name.
[Dummy] names (generated by IDA) are not listed.
Hidden names are not list too.

See also
[Navigate|Jump to
 submenu.

[How to use the lister
WRAP
File|Produce output files submenu

This submenu allows you to produce ASM, MAP and EXE files.
It also allows you to unload the database.
[Generate MAP file]
[Generate ASM file]
[Generate LST file]
[Generate EXE file]
[Dump database to IDC file]

See also
[other File
 submenu] commands.
WRAP
Invoke DOS Shell

 Action   name: Shell
 Current hotkey:
<Shell>

Using this command you can temporarily quit to the operating system

This command is not available in the 16-bit version due to the lack of the memory.

The database is left open whe you use this command, so be careful.

See also
[other File
 submenu] commands.
WRAP
Exit IDA

 Action   name: Quit
 Current hotkey:
<Quit>

This command terminates the current IDA session. IDA will
write all changes to the disk and will close all databases.

You can enable/disable database packing. When the database is packed,
it consists of one file (with IDB extension
 When the database is not
packed, it consists of several files on the disk, but IDA exits quickly.
If packing is disabled, in the next session you can'
[abort] IDA.
We do not recommend to leave the database in the unpacked form because
you will not have a backup copy.

Also you can perform garbage collection on the database before packing it.
The garbage collection removes the unused database pages, making it smaller.
However, IDA needs some free database pages when it works, so it will
allocate them again when you reuse the database. Removing and adding
free pages takes time and what is most important it changes the database
control blocks.

Use garbage collection only when you do not intend to work with the
database in the near future.

IDA will remember all information about the screen, cursor
position, jumps stack, etc.
The following information
will be lost:

[keystroke macros

        the
[anchor] position

To resume a disassembly session simply type:
ida file

See also
[other File
 submenu] commands.

[Abort] command.
WRAP
Unpacked database is dangerous

First of all you can lose all information because you won't have a backup
copy of the database. IDA makes modifications to the unpacked database
and if some unexpected condition occurs during work, the unpacked database
usually is damaged. IDA is able to repair such a database, but some
information
could be irreversibly lost leading to disastrous consequences.

The only advantage of unpacked database is that it is loaded very fast.
Naturally, exiting to DOS is fast too.

If packing is disabled, in the next session you can'
[abort] IDA.

YOU ARE WARNED.

WRAP
Save database

Action   name: SaveBase
Current hotkey:
<SaveBase>

This command saves and packs the current database.

See also
[other File
submenu] commands.

[Save database as
command.

WRAP
Save database as

Action   name: SaveBaseAs
Current hotkey:
<SaveBaseAs>

This command saves and packs the current database.
IDA will prompt you to enter a new name for the database file. Databases
are saved with an .IDB extension.
Use this command if you don't want to overwrite an earlier database.

See also
[other File
submenu] commands.

[Save database] command.

WRAP
Abort IDA

Action   name: Abort
Current hotkey:
<Abort>

This command terminates the current IDA session.

The Abort command is not available if the database was not packed.

```
        IDA will NOT save changes to the disk.
```

See also
[other File
 submenu] commands.

[Quit] command.
WRAP
IDA command line switches

IDA is started with the following command line:

```
        ida file
```

It may specify a file mask instead of file name.

```
  where switches:
-c     disassemble a new file (kill old database)
[processor type]
-a     disable
[auto analysis]
 program entry point (
 loading address (
-R     load MS Windows exe file resources
 -
  specify MS Windows directory
 -x     do not create segmentation
        (used in pair with
[Dump database] command)
        this switch does affect EXE and COM format files only.
-n     load MS DOS EXE portion of the input file
 -d     debug
 -f     disable FPP instructions (IBM PC only)
+    pack database
 -
-    do not pack database (see
[Abort] command)
-M     disable mouse
 -
  execute IDC file. Note that IDA always executes IDA.IDC
        file if it exists.
-A     autonomous (batch) mode. IDA will not display dialog boxes.
        Designed to be used together with -S switch.
-h     this screen

     this screen
 ?      this screen
```

WRAP
Database Is Empty

Although the database exists, it is empty.
Delete it and start again.

If you have previously saved your database into a text file,
you can load it. See
[Dump database] command for explanations.

See also
[IDA usage help]
WRAP
Illegal Usage of the Switch

Not all switches can be used when you
start IDA the second or more times.
Below are valid switches:
-a   disable
[auto analysis]
-d   debug

See also
[IDA usage help]
WRAP
Names Representation

 Action   name: SetNameType
 Current hotkey:
<SetNameType>

Dummy names are automatically generated by IDA. They are used to denote
subroutines, program locations and data.

Dummy names have various prefixes depending on the item type and value:

```
  sub
          instruction, subroutine start
  locret
       'return' instruction
  loc
          instruction
  off
          data, contains offset value
  seg
          data, contains segment address value
  asc
          data, ascii string
  byte
         data, byte (or array of bytes)
  word
         data,
-bit (or array of words)
  dword
        data,
-bit (or array of dwords)
  qword
        data,
-bit (or array of qwords)
  flt
          floating point data,
-bit (or array of floats)
  dbl
          floating point data,
-bit (or array of doubles)
  tbyte
        floating point data,
-bit (or array of tbytes)
  stru
         structure (or array of structures)
  algn
         alignment directive
  unk
          unexplored byte

You can change representation of dummy names.
IDA supports several types of dummy names:

  0     loc
1234     segment base address relative to program base address

 offset from the segment base
  1     loc
1000
1234  segment base address
 offset from the segment base
  2     loc
dseg
1234  segment name
 offset from the segment base
  3     loc
11234    segment relative to base address
 full address
  4     loc
1000
11234 segment base address
 full address
  5     loc
dseg
11234 segment name
 full address
  6     loc
12         full address
  7     loc
0012       full address (at least 4 digits)
  8     loc
00000012   full address (at least 8 digits)
  9     dseg
1234      the same as 2, but without data type specifier
 10     loc
1          enumerated names (


If you have selected names type 10 (enumerated names
 you may
```

renumber them using a checkbox. The process is relatively fast, surprisingly.

The best representation for MS DOS programs is #
,
for 16-bit processors -
, and for 32-bit processors -
.
You can change dummy names type any time when you want.

Also, here you can set up types of names included in the
[names list

IDA knows about the following types of names:
- normal names
- public names
- weak public or extern names
- autogenerated (meaningful) names.
- dummy (meaningless) names.

Dummy names may be public or weak, but they never appear in the list of names.
You can specify type of a name when you create or
[modify]
.

Also, here you can set maximal length of new names. Old names will not be affected by this parameter.

See also:
[Rename] command

[Rename any address] command

[Options] submenu
WRAP
Demangled Names

Action   name: SetDemangledNames
Current hotkey:
<SetDemangledNames>

IDA can demangle mangled C
names of the most popular C
compilers:
from Microsoft, Borland and Watcom. The demangled names are represented in two
forms: short and long form. The short form is used when a name is used as a reference, the long form is used at the declaration.

You can set how demangled C
names should be represented:
- as comments. this representation allows you to obtain recompilable source text
- instead of mangled names. this representation makes the output more readable. the disadvantage is that you can't recompile the

```
          output
        - don't display demangled names.
```

You can setup short and long forms of demangled names. Short form is used
when a reference to the name is made; long form is used at the declaration.

To make demangled names more readable, we instroduce concept of default
memory
model for demangled names. IDA will not display memory model modifier (near/
/huge)
if it is equal to the default memory model.

Also, to make the demangled name more compact, unsigned types may be
displayed
as uchar, uint, ushort, ulong. The same with signed basic types.

See also
[How to customize demangled names]

WRAP
Customize Demangled Names

All your changes will be saved in the current database. If you want to change
form of demangled names for all new databases, then you need to edit
the
:1482[configuration] file,
variables 'ShortNameForm' and 'LongNameForm

Below is the list of all checkboxes with examples (spaces are inserted only
for the ease of understanding

No underscores in
ccall,
pascal, etc

```
        on:     void   pascal func(void

        off:    void
pascal func(void
```

No calling conventions for parameteres and
based

```
        on:  void         func(int
          f2
void

        off: void
cdecl func(int
pascal f2
void
```

No return type of functions

```
        on:         func(void

        off:    int func(void
```

No
based
 specifier

```
        on:   int                            a;
        off:  int
based(
segname(
DATA
;
        on:   void
based(void)
;
        off:  void               *
;
        on:   int
based(
self)
void

        off:  int
void
```

No calling conventions

```
        on:     void          func(void

        off:    void
pascal func(void
```

No postfix const in member function declarations

```
        on:     void aclass
func(void

        off:    void aclass
func(void) const;
```

No near/
/huge modifiers

```
        on:     char     * func(void

        off:    char far * func(void

        on:     char     *      *

        off:    char far * near *
```

```
  note: at the bottom of the dialog box you can specify the default
        memory model.

No public/private/protected keywords

        on:             void func(void

        off:    private void func(void


No throw descriptions

        on:     void func(void

        off:    void func(void) throw(Class
Member


No static and virtual keywords

        on:                    void aclass
func(void

        off:    static virtual void aclass
func(void


No class/struct/union/enum keywords

        on:            A
 func(      A
      B

        off:     class A
 func(class A
 enum B


Inhibit everything except the main name

        on:     func;
        off:    static void
pascal func(void) const;

No const and volatile keywords

        on:     char       * func(void

        off:    char const * func(void



Default memory model:
        Near            'near' keywords never will be displayed
        Far             '
```

```
'  keywords never will be displayed
        Huge            'huge' keywords never will be displayed
        None            all keywords will be displayed unless

No near/
/huge modifiers
 is checked


d     WRAP
Demonstration Version



This is the demo version of IDA Pro.
You may order IDA Pro by faxing the order form below.
(Users in Russia please contact ida
geliosoft.
)

Site licenses, unlimited corporate licenses, education discount,  please
contact

        sales
datarescue.com
        ph  :
-3446510
        fax :
-3446514




                        IDA Pro Order Form


            Name:


         Company:


         Address:


            City:


 State, ZIP Code:


         Country:


         Phone
```

```
           FAX


          e-mail:



           1 computer
  USD199 (BEF6190)
    2 to   9 computers
  USD159 (BEF4929)
# computers

   10 to  24 computers
  USD135 (BEF4185)
# computers



   DELIVERY - please choose one

   EMS, DHL, TNT (depending on the destination) USD30/package



   EEC RESIDENTS ONLY - please choose one

   [
]  my VAT number is


        or

   [
]  I don't have a VAT number, VAT is billed in Belgium
        add 21% VAT
 VAT

PAYMENT METHOD - please choose one


              MasterCard [
]  EuroCard [
]   VISA [
]


              Expiration Date


              Card Number


              Signature



              BANK Transfer
```

               [
] CGER (Belgium) 001-2804935-
)
               [
] CGER (Belgium) 001-2710240-
)
               [
] CGER (Belgium) 005-4343178-
)

               Postal Transfer

               [
] CCP (Belgium)   000-1370690-80

               Company Purchase Order (requires approval)

               [
]

 fax or mail this form to

      DataRescue sprl   PHONE:
-3446510
      45 Quai de la Derivation   FAX:
-3446514
      4020 Liege   BBS:
-3420304
      Belgium
      VAT :
.553

WRAP
Edit File

 Action   name: EditFile
 Current hotkey:
<EditFile>

Here is a built-in text editor in IDA. You may use it to edit small files.
(I never tested it on big files
 All
[keys] are usual

You can view files of arbitrary size using
[ViewFile]
command.

See also
[How to use the file editor]

[View] submenu.
WRAP
How to use The File Editor

You can use the following keys in the file editor:

Ctrl-Q F	Search	Shift arrow>	Select
Ctrl-L	Search Again	Ctrl-K B	Start Select
Ctrl-Q A	Replace (no prompts)	Ctrl-K H	Hide Select
Ctrl-U	Undo	Ctrl-Ins	Copy
		Ctrl-Del	Clear
Ctrl-Left	Word Left	Shift-Ins	Paste
Ctrl-Right	Word Right	Shift-Del	Cut
Ctrl-PgUp	Text Start		
Ctrl-PgDn	Text End	Ctrl-Y	Delete Line
		Ctrl-Q H	Delete Line Start
Ctrl-O	Indent Mode	Ctrl-Q Y	Delete Line End
		Ctrl-T	Delete Word

When you close this window using 'WindowClose' command (hotkey
<WindowClose

IDA will ask you to save the file if it was modified. No backup files will be created.

Do not forget that you can use the clipboard.

You can view files of arbitrary size using

[View File]
command.
WRAP
Cross References Dialog

This command changes the representation of cross references.
Here you can change value of the following checkboxes/input fields:
[Cross references display]
:1003[Segments in cross references display]
:1178[xref depth]

Also, you can change right margin for lines with cross references.

See also

[Options] submenu.
WRAP
Assembler Directives Dialog

This command enables/disables generation of some assembler directives, namely:

        assume directive

        origin directive

Sometimes (when you do not intend to assemble the output file
 you may want
to disable their generation.

You can change these settings any time.

 See also

[Options] submenu.
WRAP
Text Representation Dialog

This command changes the look of disassembled text.
Here you can change value of the following checkboxes/input fields:
[Line prefixes display]
[Opcode bytes]
[Segment names display]
[Segment addresses display]
:1106[Instructions indention]
:1142[Comments indention]
:1006
void' marks display]
:1100[Repeatable comments display]
:1007[Automatic comments display]
:1101[Empty line display]
:1103[Borders display]
:1156[Bad instructions display]
:1155[Use tab stops in the output file]
:1157[void limits]

 See also

[Options] submenu.
Disable/Enable Line Prefixes Display

This checkbox enables or disables line prefixes display.
Line prefix is the address of the current byte:

```
 3000:1000      mov ax, bx
```

See also

[Opcode bytes]
[Text Representation Dialog]
WRAP
Disable/Enable Opcode Bytes

Here you specify number of opcode to display
on one line of the listing.
Opcode bytes are shown below:

```
 3000:1000 55           push    bp
```

The opcode is the operation code of the current instruction.
For data items the opcodes are elements of data directives.
Sometimes there is not enough place to display all bytes of an item (of large array, for example
 In this case IDA will display just a few first bytes
of the item. By default IDA shows 8 bytes per line.
You can specify numbers from 0 to 16.

See also

[Line prefixes display]
[Text Representation Dialog]
WRAP
Disable/Enable Segment Names Display

This checkbox enables or disables segment names display.
If enabled, IDA will use segment name in all addresses.
If disabled, IDA will use segment bases.

Example (codeseg has base 3000

```
   Enabled:     codeseg:0034
   Disabled:    3000:0034
```

See also

[Text Representation Dialog]
The key file is not found or is corrupted.

This feature doesn't work in the demonstration version. Please see README.TXT for information.
Unloading information about segmentation
~ member

The names list is empty.
Disable/Enable Segment Addresses Display

 Using this checkbox you can disable segment addresses in the
 listing. IDA will show only offsets.

Example:

```
   Enabled:     codeseg:0034
   Disabled:    0034
```

See also

[Text Representation Dialog]
Overlay stub at %08lX, code at %08lX
Pascal style overlays file

```
' not found.
Please enter pascal overlays file
Searching for the data segment
:1399

Sorry, the demo version can't load old databases.
Press F1 for details.
WRAP
Cross-references Display


IDA maintains cross-references automatically. Of course, when IDA starts to
disassemble a new file the cross-references will not appear immediately;
they will be collected during background
[analysis


Using Cross References
[Dialog] you can
enable or disable cross-references display.
S u b r o u t i n e

Enter comment

Enter ENUM repeatable comment
WRAP
Segments Cross-references Display


This checkbox enables or disables segments in cross references:

                ; CODE REF: 3000:1025   - enabled
                ; CODE REF: 1025        - disabled

See also

[Text Representation Dialog]

Enter repeatable comment

Create a new segment

 Start address and end address should be valid.
         End address > Start address


~tart address   :
-notation:
~nd   address   :
   hex is 0x
~ase            :
 in paragraphs

~lass           :
(class is any text)
-bit segment  :
'void' Marks Display
```

This command enables or disables
:1157[void] marks display.

See also

[Text Representation Dialog]
WRAP
AutoComments Display

AutoComments are predefined comments for all instructions.

If you forgot meaning of a certain command, you can use
this command to get comments to all lines of the screen.

IDA does not give comments to very simple instructions such as
'
' instruction
and does not override existing comments.

See also

[Text Representation Dialog]
IDA has found unpacked version of database %s on the disk. Please choose:

Enter ENUM regular comment

Program Segmentation

Selectors

Selector %04X is used in the program.
Attempt to call non-existent external function
Attempt to call undefined function
IDC stack overflow!
Undefined code
Runtime error at
, arg:
Load segment

Entry point %08lX is not loaded into the database. Propably you need to
reload the input file manually.
File read error at %08lX (may be bad PE structure
 continue?
't find translation for virtual address %08lX, continue?
.idata
 segment has some additional data (
%08lX
 do you still want to truncate it?
Unknown fixup type
x is ignored

```
; Format           : Portable Executable Format (
; Created at       :
; CPU              :
```

```
) unknown
; CPU               :
) Intel 80386
; CPU               :
) Intel 80486
; CPU               :
) Intel 80586
; CPU               :
) MIPS Mark I (R2000, R3000)
; CPU               :
) MIPS Mark II (R6000)
; CPU               :
) MIPS Mark III (R4000)
; CPU               :
) DEC Alpha
; Subsystem         : Unknown
; Subsystem         : Native
; Subsystem         : Windows GUI
; Subsystem         : Windows Character
; Subsystem         :
/2 Character
; Subsystem         : Posix Character
; Flags
08lXh)
;                    Reversed bytes (
;                    Library image.
;                    System file.
;                    Debugging info stripped.
;                    32-bit word machine.
;                    Reversed bytes (
;                    16-bit word machine.
;                    (Update Object
;                    (Minimal Object
;                    Local symbols stripped.
;                    Line numbers stripped.
;                    Image is executable.
;                    Relocation info stripped.
; DLL flags
04Xh)
 Per-Process Library Initialization.
 Per-Process Library Termination.
 Per-Thread Library Initialization.
 Per-Thread Library Termination.
; OS version:
d User:
d Subsystem:
d Linker:
; Object alignment:
%08lX
 File algn:
%08lX
; Stack reserved  :
%08lX
 commited :
%08lX
; Heap  reserved  :
%08lX
```

```
 commited :
%08lX
; Entry point     :
%08lX
08lX)
; Section %
(virtual address %08lX)
; Virtual size                  :
%08lX
; Section size in file          :
%08lX
; Offset to raw data for section:
%08lX
; Flags %08lX:
 Regular
 Dummy
 Noload
 Grouped
 Padding
 Copy
 Text
 Data
 Bss
 Exception
 Comment
 Overlay
 Lib
 Loader
 Debug
 Type check
 Overflow
 Discardable
 Not cachable
 Not pageable
 Shareable
 Executable
 Readable
 Writable
; Alignment     : 1 byte
; Alignment     : 2 bytes
; Alignment     : 4 bytes
; Alignment     : 8 bytes
; Alignment     : 16 bytes
; Alignment     : 32 bytes
WRAP
```

Repeatable Comments Display

This command enables or disables

[Repeatable comments] display.

The repeatable comment will be displayed by IDA when the current item has not a
[regular] comment and the current item refers to an item
with a repeatable comment or the current item itself has a repeatable comment.

See also

[Text Representation Dialog]
WRAP
Empty Lines Display

This command enables or disables empty lines display.

Useful to decrease number of blank lines on the screen
increasing amount of information on it.

See also

[Text Representation Dialog]
Not IDL file
WRAP
Border Lines Display

This command enables or disables border lines display.

Useful to decrease number of border lines on the screen
increasing amount of information on it.

Border line:

is generated automatically by IDA to separate data and code.

See also

[Text Representation Dialog]

Rename address

```
  Address:
%N

~ame :



~aximum length of new names:



~nclude in names list:
>

~ublic name:
>

~utogenerated name:
>
```

```
~eak name:

Do you really want to delete structure
Change Indention of Text


You can change indention of disassembled instructions:

                mov ax, bx


   indention


This can be a great help when you are going to produce
a large disassembled text.

See also
[Text Representation Dialog]
end of section %
(file %
function %s symidx %ld size %08lX lnnumoff %08lX nxtidx %
startitem 1


Rename any address


~ld name :
(or address)


~ew name :



~aximum length of new names:



~nclude in names list:
>

~ublic name:
>

~utogenerated name:
>

~eak name:

Please enter a valid old address.
Internal error: extra pop
 from operands stack
-formed abstract declaration
Abstract declaration is not allowed here
Too many type qualifiers in declaration
```

```
Internal error: extra top
 from operands stack
Internal error: too many purges in idc interpeter
Storage class is already specified in declaration
Bad storage class is specified in declaration
Pointer type is already specified in declaration
Illegal pointer modifiers
Undefined structure/union/enum tag name
Structure/union length is unknown
Keywords
signed
unsigned
short
long
 may be applied to scalars only
Too complex type declaration

Enter repeatable SEGMENT comment

Enter SEGMENT comment
Define a selector


~elector:
~alue   :
Symbol is already defined
Size of array is unknown
Size of array is negative
't define arrays of functions or voids
Objects of
void
 type can't be declared
Functions can't be
const
 or
volatile
Functions can't be
auto
 or
register
Functions can't return functions or arrays
Internal error: unknown BTS type %04lX
Internal error: NULL ptr to tid (TidAbstractSize)
Internal error: Illegal tid

Enter enum member repeatable comment
Typedefs cannot be initialized
Unloading information about instructions/data
WRAP
Change Indention of Comments


You can change indention of comments:

        mov ax, bx                      ; this is a comment
```

```
               indention


See also
[Text Representation Dialog]
Delete Cross Reference

  <From :
<To   :



~ndefine if no more references:

Enter enum member regular comment
Unloading information about segment registers
Unload information to file
(Ctrl-Break to cancel
No enum contains a constant with value 0x%
      Function name               Segment   Start    Length  RFLSBMICDV
  Enumeration         Comment

Modify function


~ame         :
~tart address:


~nd   address:
't return:
>

~ar function:
>

~ibrary func:
>
  Enter size of (in bytes)      <
~atic func:
<Local ~
~ariables   :
~P based frame:
<Saved ~
~egisters   :
~urged upon return:



Stac~
~ variable
Change si~
~nter operand manually

Enter repeatable FUNCTION comment
WRAP
Enable/Disable Tab Stops
```

You can disable tab stops (0x09) in the
output file if you do not like them.

By default IDA produces output files with tab stops.

See also
[Text Representation Dialog]
WRAP
Bad Instructions Display

Some assemblers do not understand some instructions even if they should to.
For example, processor Z80 has several undocumented instructions and many
assemblers fail to eat them. IDA knows about this fact and tries to produce
an output that can be compiled without errors, so it replaces such
instructions
with data bytes.

The problem is more severe with Intel 80x86 processors: the same instruction
can be coded differently. There are 2 operation codes for ADD instruction e
t.
.
The worst thing is that the different operation codes have different lengths.
If the assembler used to compile a file and your assembler produce different
operation codes, you may obtain completely different output files.

That is why IDA marks such instructions as <
> and replaces them with
data bytes.

However, you can disable this feature of IDA (if you do not intend to
recompile
the file these marks simply annoy you

Example:

```
   Enabled:
                db 0Fh,
> jbe     loc
205
                db 0Fh,
> jb      loc
205
                db 0Fh,
> jnb     loc
205
   Disabled:
                jbe     loc
205
                jb      loc
205
                jnb     loc
205
```

```
The Borland's TASM 4.0 was the assembler which I used to decide whether
the instruction is bad or not.

See also

[Text Representation Dialog]
WRAP
Low
 High Offset Limits


IDA marks some instructions with 'void' marks. When doing so, IDA asks you
to pay attention to these instructions.

Two values control 'void' marks generation.  An item is 'void' if
it has an immediate value as an operand or part of an operand and this
immediate value is between low and high 'voids' limits. The comparison
is always unsigned,
. in the instruction

        mov ax
]

  the immediate operand is 0xFFFE, not -
.

See also
[Text Representation Dialog]

[How to Enter a Number
Attributes:
 noreturn
 far
 varargs
 member
 static
 virtual
 constructor
 destructor
 library function

Enter FUNCTION comment
't open the database: access denied. Probably the database is read-only or in
use
~ew segment
~elete segment
Are you sure you want to delete function
A function can start with an instruction only
't set a function
' start to %
:1100

Repeatable comments are hidden, unhide them.
Autoanalysis subsystem is initialized.
The initial autoanalysis is finished.
WRAP
Maximum Depth For Tail Bytes Lookup
```

```
This value controls
[cross references] displaying.
 It means
 'how many bytes of an object to look at
        to collect cross references


For example we have an array:
        A       db 100 dup(
)

If some instruction refers to the 5-th element of the array:
        mov     al,
+5

 with TD
3      we'll have no xrefs displayed
 with TD
10     we'll have this xref

See also
[Text Representation Dialog]

[How to Enter a Number
~dit segment
~ove current segment

Comments are hidden, unhide them.
:1218

Choose a function
The function has undefined instruction/data at %s
Your request has been put in the autoanalysis queue.
The function is already defined, start address at %
Unknown instruction mnemonics
Change segment ~
~ranslation
internal error in lnar: too many lines!
~et default segment register value
            Name                  Address   #

Enter STRUCT repeatable comment

Enter STRUCT comment

Enter member repeatable comment

Enter member comment
Bad structure field name
 Probably the name is invalid or is used in this structure.
Bad field offset %04lX. Either the field size is too big or another field is
present at the specified offset.
Bad field size.

Do you really want to delete field
```

HEAP IS CORRUPTED!
WRAP
String styles dialog

In this dialog you can setup string styles and also define a new string immediately.

The following string styles are defined:

```
      C-style (0 terminated)
      DOS style (
 terminated)
      Pascal style (length 1 byte)
      Wide pascal (length 2 bytes)
      Unicode
      Character terminated
```

If you select
character terminated
 string style then you may
specify up to 2 termination characters. The string will be terminated by any of these characters. If the second character
is equal to 0, then it is ignored.

See also:
[Options] submenu.
WRAP
Strings options

This dialog deals with string settings.
Here you can setup:

```
        automatically generated
:1201[serial names]

        character to
:1206[force] start of next line
```

Here you can specify prefix for automatically generated names.

If a name is marked as
[autogenerated
 it will be displayed in a
different color and will be included in the
[list] of names depending
on the current setting.

Also you can disable automatic generation of string names.

See also:
[Options] submenu.
WRAP
ASCII String Serial Names

```
IDA can generate serial names for ASCII strings,
.

        pref
,pref
,pref
003 etc


You should enable serial names generation, specify
prefix for names,starting serial number and number of leading zeroes.

Each time you
[create] an ASCII string, IDA generates
a new serial name and assigns it to the string.

See also
:1199[ASCII style Dialog]

Struct array size (in elements)

  Current offset       :

  Next defined item at :


  Array element width  :
%D
  Maximal possible size:
%D
  Current array size   :
%D

  <Array ~
~ize     :
(in elements)
~tems on a line:
~lignment      :
-none,
-auto)

  <Use
 construct:
~ned elements    :

Change SP value

  Current SP value     :
%N


~IFFERENCE between old and new SP:
(the current instruction modifies SP value)
           Structure             Size
Choose a structure (not the current
WRAP
ASCII Break Character
```

This character forces IDA to start a new
line when it displays ASCII string on the
screen. By default, it equals to

See also:
[Convert to ASCII] command

:1199[ASCII style Dialog
[How to Enter a Number
WRAP
Color configuration

This dialog allows you to customize
:1746[color settings

IDA keeps colors in file IDACOLOR.
. This file may reside in the IDA
directory or in the current directory. It is a binary file.
IDA automatically saves all your changes into this file.

If the current item line prefix has
black on black
 color, then
the current item will not be hightlighted. The same is with current line
color - if it is
black on black
, the current line will not be highlighted.

There are four palettes. You can easily switch between them or disable
syntax hightlighting at all.

See also:
[Options] submenu.
:1746[Color codes]
WRAP
Can't Find Input File

IDA have tried to find file with the following extensions:

```
        .com    .bin    .ov4
        .exe    .ovr    .ov5
        .dll    .ov0    .ov6
        .ovl    .ov1    .ov7
        .drv    .ov2    .ov8
        .sys    .ov3    .nlm
```

No such files were found.

See also
[IDA usage help]
IDA found a database for file %
. Do you want to overwrite it?

```
WRAP
Patched Bytes Are Skipped


Some bytes in program memory have no corresponding byte in the executable
file. For example, uninitialized data is not kept in the file.
/2 Warp and Window support compressed pages.

In this case IDA can't create full difference file. It shows the skipped
byte addresses along with their values in the messages window.
: create/delete enumeration types
Ctrl-
: create a new symbolic constant
N      : rename a symbolic constant
U      : delete a symbolic constant
No predefined comment for the specified instruction and values.

Text search (slow
~tring:
~ase sensitive:
           <Search ~
>
                                <Search ~
Too long record (length
) is encountered at file position %08lX
Pattern is not found.
Checksum error at file position %08lX, continue?

Choose a structure type
WRAP
List of Functions


You can use
[list viewer] commands in this window.
Here is
[format] of this window.
WRAP
Perform en masse operation


You can choose range of operands to perform en masse operation.

ALL OPERANDS



The operation will be performed on all
operands as a toggle. For example, if you
ask to convert to a character, then all
non-character operands will become characters,
and all character operands will become non-chars.

VOID OPERANDS
```

The operation will be performed on
[void] operands only.
You may change value of void
:1157[limits] in this dialog box.
 OPERANDS

This selection will convert all operands with the specified type to void operands.
Example: all characters become non-characters.

NOT
 OPERANDS

This selection allows to convert all operands that don't have the specified type
to the specified type. Example: all non-characters to characters.

NOT TYPED OPERANDS

This selection allows to convert all operands without a type to the specified type. Example: all operands with no type to characters.

APPLY ONLY IF POSSIBLE

IDA will check whether an operand can be represented with the specified type
(as a character constant, for example
 and perform type conversion only if
the check is positive.
Struct size is zero
Char
WRAP
Enums submenu

Commands of this submenu are available in the

[enums window

:1379[Add a enum]

:1333[Delete a enum]

:1379[Edit a enum]

:1384[Define a enum member]

:1725[Edit a enum member]

:1729[Delete a enum member]

All enum members should be unique in the program. You can't define more than one enum member with the same value.

Also you can add a comment for the enum and for each enum member.
In order to specify a enum comment, you should stand at the enum name.
Comments are set using regular commands:

[Regular comments]

[Repeatable comments]

See also
[Edit] submenu.

```
; End of subroutine
Unsupported record COMFIX is encountered at file position %08lX, continue?
Unsupported record SELDEF is encountered at file position %08lX, continue?
Unknown record type %02X is encountered at file position %08lX, continue?
: reference to undefined segment number %u at file position %08lX, continue?
Unrecognized OMF COMENT extension %02X is encountered at file position %08lX,
continue?
 Object file has an incremental compilation error record
:1411
: the input file is bad (file position %08lX)
An unexpected condition occured. Please send the input file to <
-tech
datarescue.
 Thank you.
Illegal BAKPAT record type %d at file position %08lX, continue?
Fixup overflow at %08lX (target offset %08lX)
Bad input file: at position %08lX IDA found a self-relative fixup on LIDATA.
This is not permitted.
Alias %
Module name      :
Version          :
Vendor info      :
Translator       :
Intel copyright  :
DOSSEG directive : Yes
includelib
0 Segment type: Regular
1 Segment type: Externs
2 Segment type: Pure code
3 Segment type: Pure data
4 Segment type: Imports
6 Segment type: Group
7 Segment type: Zero-length
9 Segment type: Uninitialized
Segment type: Unknown

Debug info type  : CodeView
Debug info type  : AIX
Debug info type  : IBM PM
Debug info type  : Turbo Debugger
Debug info type  : Unknown
```

```
loaddds used    : Yes, create protected mode library
Big endian       : Yes
Library module   :
EXESTR           :
Pharlap format   : Yes
Comment          :
Compiler         :
Date stamp       :
Phoenix timestamp:
Time stamp       :
User info        :
Dependency       :
02d %
02d %
Language         :
0x00 unspecified
0x01 C
0x02 Pascal
0x03 Basic
0x04 Assembly
0x05 C

unknown
0 tiny
1 small
2 medium
3 compact
4 large
5 huge
6 386small
7 386medium
8 386compact
9 386large
, with underscores
, no underscores
Borland debuginfo: version %
Borland flags    : Optimization %08lX
Compiler options :
MS parameters    :
Watcom parameters:
0x30 8086
0x31 80186
0x32 80286
0x33 80386
0x4F opt
0x73 small
0x6D medium
0x63 compact
0x6C large
0x68 huge
0x41 68000
0x42 68010
0x43 68020
0x44 68030
0x63 fp:call
emulator
0x65 fp:emulated
```

```
inline
0x70 use
0 in AX
2 in DX
3 in BX
6 in SI
7 in DI
in unkreg
Linker directives:
[For new exe]
[Omit
PUBLICS]
[Run MPC]
-code interpreter version:
, CodeView version:
                 This segment won't be padded even if
           /PADDATA or /PADCODE linker directives are used.
Bad input file: illegal GRPDEF record at %08lX
Bad input file: illegal COMDEF record at %08lX
12s
%ld
  public
24s ; Absolute external value
12s
Far data,
%lu elements %u bytes each
Near data,
%lu bytes
VIRDEF, segment %s
NOTE: VIRDEF records can not be represented in assembly!
This segment is used by COMDAT record %
COMDAT, segment %
(COMDAT records can't be represented in assembly
Local COMDAT (effectively LCOMDAT)
Data in code segment. Can't be placed in the overlayed module.
0 Selection : no match (only one instance allowed)
1 Selection : pick any instance
2 Selection : same size, all instances must have the same size
3 Selection : exact match, all instances must have the same checksum
Selection : unknown
Enum
Struct offset
Stack variable
0 Allocation: explicit
1 Allocation: as CODE16
2 Allocation: as DATA16
3 Allocation: as CODE32
4 Allocation: as DATA32
Allocation: unknown
FIARQQ
FICRQQ
FIDRQQ
FISRQQ
FIWRQQ
FIERQQ
FJARQQ
FJCRQQ
FJSRQQ
0 at
1 byte
2 word
3 para
4 page
5 dword
6 page4K
Change segment register ~
~alue

Names window
Do you really want to delete selector %
Forced
0 private
2 public
4 public
5 stack
6 common
7 public
0 BYTE
1 OFF16
2 SEG16
3 PTR32
4 OFF32
5 PTR48
Unknown fixup
40s
40s
Loading signature %
The installed version of IDA doesn't contain signature file
Only the full-featured version of IDA contains FLIRT signature file
Please contact your distributor for more information.

List of library modules to apply
Change ~
 Selfrel
 Unused
Internal error: illegal variable type in idc interpeter
Internal error in idc interpeter: bad local var
Internal error in idc interpeter: bad arg
Floating point operation underflow/overflow
 EXTDEF
Too long built-in function name:
Unloading completed.
array %s symidx %ld lnno %u size %

Enter base address for the offset


~ase address:
-based frame
Output file
' already exists.
%08lX:
(internal error) Bad data type
) is passed to OutValue
```

```
 function.
Reference to undefined symbol
COFF File Manual Loading

        Segment :
%A
        Size    :
%N
        Use32   :
%D

 <Start ~
~ddress :
~elector      :
(0 means that selector
                                  will be chosen by IDA)
<Segment ~
~ase  :
(in paragraphs)
<STOP LOADING ~


  IDA doesn't check values specified by you!
(this is a feature, not a bug
Bad segment base value
Unusual segment alignment is replaced by 'page' in the output
Change ~
  File     State #func             Library name
0 Applied
1 Current
2 Planned
IDC interpeter can't be run recursively.
WRAP
Delete a enum type


 Action    name: DelEnum
 Current hotkey:
<DelEnum>

This command deletes the current enum.
Beware, when you delete a enum all references to it will be destroyed,
Even if you recreate it later, you'll have to respecify all references
to it.

These command is available when you open a enums
[window


See also
:1222[Edit|Enums] submenu.
Changing the comment strings, please wait
static
expr
{ return %
static
internal error: invalid au
```

```
code %
switch %d cases
low halves of switch values
high halves of switch values
value table for switch statement
List of available library modules
Illegal key code

You've mistyped key code. Below are some examples:




Shift-




Ctrl-
Syntax error
Configuration file syntax is:
        KeyWord
 Value
~nctions

Leaving the database in unpacked form is dangerous. You can lose all data. Do
you want to continue?
All previous disassembly will be lost. Are you sure you wish to overwrite it?
IDA can't remove the database file:
IDA can't rename the database. The repaired database is saved as %
. OS error:
Repairing the database, please wait.
The database is successfully repaired.
Collecting garbage in the database.
Garbage collection is successfully completed.
Garbage collection error occured, rolling back to original database.

QUIT IDA

  IDA will save all changes to disk.

  <
t pack database:
<Pack database
~tore
<Pack database
~eflate
IDA can't collect garbage in the database: not enough disk space.
(wanted:
, available:
Attention! Probably erroneous situation.
: no such file
2 %
: not IDA DLL file
3 %
: can't load: bad segments
4 %
```

```
: no linkage info
5 %
: bad relocation type
6 %
: bad imported ordinal
7 %
: bad relocation atype
8 %
: invalid DLL data offset
%
: can't load (code %
f1f33a56f38ecf2346439d5ad729f7ecf9f43ac4f9f03a8ead0c9055258194237a955650279b9
525b419cd218ad0b40ecd21fec8a2c204be80008a0c32ed0bc97503e9890046e89e02e8a20272
0ea0c5042206c6047502eb75e98300817c013a207415817c013a0d740e803ec404007538c606c
40401eb31803ec60400754fc606c604ff803ec404007505c606c404028a04e867022c413a06c2
047606ba7405e90302a2c30446eb9c803ec50400751ec606c504ff8936c80446e831027307803
c207402ebf38936ca044ee977ffba2106b409cd21ba7406b409cd21e9e901e8cf027306ba3005
e9
~alculate
IDA can't rename the database after performing garbage collection. Compressed
file is abadoned.
:1218

Choose function to jump to
: error in zipfile
3 %
: severe error in zipfile
4 %
: insufficient memory
5 %
: insufficient memory
6 %
: insufficient memory
7 %
: insufficient memory
8 %
: insufficient memory
9 %
: zipfile not found
10 %
: bad or illegal parameters specified
11 %
: no files found
50 %
: disk full
51 %
: unexpected EOF
%
: unknown error
1 internal error
2 bad compressed file
3 not enough memory
4 file write error
5 file read error
6 error during 'explode'
unknown zip error
:1399
```

```
Sorry, the demo version can't load files larger than 64Kbytes.
Press F1 for details.
Show segregs
Archive:
%s Type:
't set a function
' end to %
   File Optional              Library name
Not enough memory, some signatures are not displayed.
Standard library:
This version doesn't support signatures.
Optional libraries:
%s OS type         :
%s Application type:
1 MS DOS
2 MS Windows
4 OS/2
8 Novell Netware
WRAP
Add/Edit a enum


 Action    name: AddEnum
 Current hotkey:
<AddEnum>

 Action    name: EditEnum
 Current hotkey:
<EditEnum>

This command allows you to define and to edit a enum type.
You need to specify:
        - name of enum
        - its serial number (

        - representation of enum members

Each enum has its ID and a serial number. The ID is a number used to
refer to the enum, while a serial number is used to order enums
during output. Changing the serial number moves the enum to another place.

The serial number of a enum is displayed at the lower left corner of the
window.

You can specify any number as a serial number,
IDA will move the enum to the specified place.
        1 - the current enum becomes the first enum
        2 - the current enum becomes the second enum



Also,you need to specify representation of enum constants. You may choose
from various number bases (
) and character constants.

These command is available when you open a enums
```

[window

See also
:1222[Edit|Enums] submenu.

[How to Enter a Number

QUIT IDA

  IDA will save all changes to disk.

  <
t pack database:
<Pack database
~tore
<Pack database
~eflate

~ollect garbage:
0x0001 Console
0x0002 GUI
0x0004 Executable
0x0008 DLL
0x0010 Driver
0x0020 Singlethreaded
0x0040 Multithreaded
0x0080 16bit
0x0100 32bit
SWITCH TO FULL LIST OF SIGNATURES

```
                      File Name                     Method         Size
```

WRAP
Define a enum member

This command allows you to define enum member. A enum member
is a symbolic constant. You should specify its name and value.
You can't define several constants with the same value in a enum.

See also
:1222[Edit|Enums] submenu.
Alphabetical list of IDC functions
:1601[AddCodeXref]
:1710[AddConst]
:1639[AddEntryPoint]
:1704[AddEnum]
:1449[AddHotkey]
:1736[AddSourceFile]
:1676[AddStruc]
:1681[AddStrucMember]
:1573[AltOp]
:1463[AnalyseArea]
:1742[Analysis]
:1579[AskAddr]
:1579[AskFile]

```
:1579[AskIdent]
:1579[AskSeg]
:1580[AskSelector]
:1579[AskStr]
:1579[AskYN]
:1543[AutoMark]
:1543[AutoMark2]
:1742[AutoShow]
:1578[Batch]
:1742[BeginEA]
:1546[Byte]
:1626[ChooseFunction]
:1742[CmtIndent]
:1571[Comment]
:1742[Comments]
:1455[Compile]
:1714[CreateArray]
:1721[DelArrayElement]
:1601[DelCodeXref]
:1711[DelConst]
:1705[DelEnum]
:1535[DelExtLnA]
:1536[DelExtLnB]
:1650[DelFixup]
:1619[DelFunction]
:1450[DelHotkey]
:1741[DelLineNumber]
:1583[DelSelector]
:1738[DelSourceFile]
:1677[DelStruc]
:1682[DelStrucMember]
:1459[DeleteAll]
:1717[DeleteArray]
:1601[Dfirst]
:1601[DfirstB]
:1577[Direction]
:1601[Dnext]
:1601[DnextB]
:1548[Dword]
:1456[Exit]
:1533[ExtLinA]
:1534[ExtLinB]
:1579[Fatal]
:1574[FindBinary]
:1574[FindCode]
:1574[FindData]
:1574[FindExplored]
:1628[FindFuncEnd]
:1574[FindImmediate]
:1574[FindProc]
:1581[FindSelector]
:1574[FindText]
:1574[FindUnexplored]
:1574[FindVoid]
:1584[FirstSeg]
:1720[GetArrayElement]
:1715[GetArrayId]
```

```
:1575[GetCharPrm]
:1697[GetConst]
:1694[GetConstByName]
:1703[GetConstCmt]
:1696[GetConstEnum]
:1702[GetConstName]
:1695[GetConstValue]
:1640[GetEntryOrdinal]
:1641[GetEntryPoint]
:1638[GetEntryPointQty]
:1689[GetEnum]
:1691[GetEnumCmt]
:1693[GetEnumFlag]
:1688[GetEnumIdx]
:1690[GetEnumName]
:1686[GetEnumQty]
:1692[GetEnumSize]
:1698[GetFirstConst]
:1723[GetFirstIndex]
:1668[GetFirstMember]
:1655[GetFirstStrucIdx]
:1648[GetFixupTgtDispl]
:1647[GetFixupTgtOff]
:1646[GetFixupTgtSel]
:1645[GetFixupTgtType]
:1545[GetFlags]
:1629[GetFrame]
:1632[GetFrameArgsSize]
:1630[GetFrameLvarSize]
:1631[GetFrameRegsSize]
:1633[GetFrameSize]
:1627[GetFuncOffset]
:1623[GetFunctionFlags]
:1625[GetFunctionName]
:1699[GetLastConst]
:1733[GetLastIndex]
:1669[GetLastMember]
:1656[GetLastStrucIdx]
:1740[GetLineNumber]
:1575[GetLongPrm]
:1653[GetMarkComment]
:1652[GetMarkedPos]
:1672[GetMemberComment]
:1674[GetMemberFlag]
:1671[GetMemberName]
:1670[GetMemberOffset]
:1665[GetMemberQty]
:1673[GetMemberSize]
:1675[GetMemberStrId]
:1565[GetMnem]
:1700[GetNextConst]
:1643[GetNextFixupEA]
:1734[GetNextIndex]
:1657[GetNextStrucIdx]
:1567[GetOpType]
:1568[GetOperandValue]
:1566[GetOpnd]
```

```
:1701[GetPrevConst]
:1644[GetPrevFixupEA]
:1735[GetPrevIndex]
:1658[GetPrevStrucIdx]
:1554[GetReg]
:1600[GetSegmentAttr]
:1575[GetShortPrm]
:1737[GetSourceFile]
:1636[GetSpDiff]
:1635[GetSpd]
:1663[GetStrucComment]
:1660[GetStrucId]
:1661[GetStrucIdByName]
:1659[GetStrucIdx]
:1662[GetStrucName]
:1667[GetStrucNextOff]
:1666[GetStrucPrevOff]
:1654[GetStrucQty]
:1664[GetStrucSize]
:1564[GetTrueName]
:1687[GetnEnum]
:1742[HighVoids]
:1742[Indent]
:1561[ItemEnd]
:1562[ItemSize]
:1537[JmpTable]
:1451[Jump]
:1569[LineA]
:1570[LineB]
:1549[LocByName]
:1742[LowVoids]
:1435[
:1472[MakeArray]
:1475[MakeByte]
:1462[MakeCode]
:1465[MakeComm]
:1508[MakeDouble]
:1486[MakeDword]
:1507[MakeFloat]
:1634[MakeFrame]
:1618[MakeFunction]
:1521[MakeLocal]
:1464[MakeName]
:1509[MakePackReal]
:1488[MakeQword]
:1471[MakeRptCmt]
:1473[MakeStr]
:1513[MakeStruct]
:1511[MakeTbyte]
:1522[MakeUnkn]
:1532[MakeVar]
:1480[MakeWord]
:1651[MarkPosition]
:1742[MaxEA]
:1579[Message]
:1742[MinEA]
:1563[Name]
```

```
:1555[NextAddr]
:1621[NextFunction]
:1557[NextHead]
:1559[NextNotTail]
:1585[NextSeg]
:1527[OpAlt]
:1523[OpBinary]
:1523[OpChr]
:1523[OpDecimal]
:1529[OpEnum]
:1523[OpHex]
:1526[OpNumber]
:1523[OpOctal]
:1524[OpOff]
:1525[OpSeg]
:1528[OpSign]
:1531[OpStkvar]
:1530[OpStroff]
:1538[PatchByte]
:1540[PatchDword]
:1539[PatchWord]
:1556[PrevAddr]
:1622[PrevFunction]
:1558[PrevHead]
:1560[PrevNotTail]
:1716[RenameArray]
:1642[RenameEntryPoint]
:1601[Rfirst]
:1601[Rfirst0]
:1601[RfirstB]
:1601[RfirstB0]
:1601[Rnext]
:1601[Rnext0]
:1601[RnextB]
:1601[RnextB0]
:1572[RptCmt]
:1551[ScreenEA]
:1596[SegAddrng]
:1594[SegAlign]
:1591[SegBounds]
:1550[SegByBase]
:1597[SegByName]
:1593[SegClass]
:1595[SegComb]
:1589[SegCreate]
:1598[SegDefReg]
:1590[SegDelete]
:1587[SegEnd]
:1588[SegName]
:1592[SegRename]
:1586[SegStart]
:1553[SelEnd]
:1552[SelStart]
:1718[SetArrayLong]
:1719[SetArrayString]
:1575[SetCharPrm]
:1713[SetConstCmt]
```

:1712[SetConstName]
:1708[SetEnumCmt]
:1709[SetEnumFlag]
:1706[SetEnumIdx]
:1707[SetEnumName]
:1649[SetFixup]
:1541[SetFlags]
:1620[SetFunctionEnd]
:1624[SetFunctionFlags]
:1739[SetLineNumber]
:1575[SetLongPrm]
:1685[SetMemberComment]
:1683[SetMemberName]
:1684[SetMemberType]
:1576[SetPrcsr]
:1542[SetReg]
:1599[SetSegmentType]
:1582[SetSelector]
:1575[SetShortPrm]
:1637[SetSpDiff]
:1680[SetStrucComment]
:1678[SetStrucIdx]
:1679[SetStrucName]
:1742[StringStp]
:1742[Tabs]
:1742[TailDepth]
:1742[Voids]
:1452[Wait]
:1579[Warning]
:1547[Word]
:1544[WriteExe]
:1544[WriteMap]
:1544[WriteTxt]
:1601[XrefShow]
:1601[XrefType]
:1601[
dref]
:1446[atoa]
:1448[atol]
:1415[byteValue]
:1601[
dref]
:1603[fclose]
:1609[fgetc]
:1604[filelength]
:1602[fopen]
:1436[form]
:1611[fprintf]
:1610[fputc]
:1605[fseek]
:1606[ftell]
:1430[hasName]
:1406[hasValue]
:1434[isBin0]
:1434[isBin1]
:1434[isChar0]
:1434[isChar1]

```
:1428[isCode]
:1428[isData]
:1434[isDec0]
:1434[isDec1]
:1434[isDefArg0]
:1434[isDefArg1]
:1434[isEnum0]
:1434[isEnum1]
:1430[isExtra]
:1430[isFlow]
:1434[isFop0]
:1434[isFop1]
:1428[isHead]
:1434[isHex0]
:1434[isHex1]
:1417[isLoaded]
:1434[isOct0]
:1434[isOct1]
:1434[isOff0]
:1434[isOff1]
:1430[isRef]
:1434[isSeg0]
:1434[isSeg1]
:1434[isStkvar0]
:1434[isStkvar1]
:1434[isStroff0]
:1434[isStroff1]
:1428[isTail]
:1428[isUnknown]
:1430[isVar]
:1607[loadfile]
:1447[ltoa]
:1613[readlong]
:1612[readshort]
:1616[readstr]
:1608[savefile]
:1742[
start
:1742[
start
:1444[strlen]
:1438[strstr]
:1437[substr]
:1615[writelong]
:1614[writeshort]
:1617[writestr]
:1445[xtol]

File %s can't be accepted as module
This difference file is created by The Interactive Disassembler

%
Difference file is created.
~egments
SWITCH TO ABRIDGED LIST OF SIGNATURES
signature file: main
 hints are incorrect!
```

```
main
 function at %08lX, named
jump table for switch statement
File %s is used for module %
Loading file into the database.
apply
entry %
: can't create function

Load Binary or User-Defined Format file

  File name:
%A


~inary file:




~oading segment:
(in paragraphs)
<Loading  ~
~ffset:




~reate segments:
; Alignment     : 64 bytes
WRAP
Demonstration Version Constraints


The demo version works only with new databases.
However, if you have created a database using the demo version, it can be
loaded by
[full featured] version of IDA.

The demo version is not able to produce .ASM and .LST files.
Full featured version produces .ASM and .LST files.

It can't load files larger than 64 Kbytes. Full featured version can load
files of virtually any size (addressing space is limited only by your free
disk space
The demonstration version has expired. Please visit http
.datarescue.com to get new versions.
:1210

Some patched bytes are skipped, see messages window for details
:1208

Can't find input file
0 STORED
1 SHRUNK
2 REDUCED1
3 REDUCED2
4 REDUCED3
5 REDUCED4
```

```
6 IMPLODED
7 TOKENIZED
8 DEFLATED
UNKNOWN
Total %lu undefined symbols found
Illegal keyword
Below is the list of allowed keywords:
NPAGES      ASCII
STYLE    PRIORITY
DELTA    OPCODE
BYTES
OS2DIR      ASCII
PREFIX   SCREEN
PALETTE    ShortNameForm
VPAGES      ASCII
SERIAL   SHOW
INDICATOR    COMMENTS
INDENTION
WINDIR      ASCII
SERNUM   USE
TABULATION    SHOW
SOURCE
LINNUM
USE
FPP     ASCII
ZEROES   ASCII
LINEBREAK   MAX
DATALINE
LENGTH
AUTOSAVE    SHOW
ASSUMES   DATABASE
MEMORY   SHOW
INSTRUCTIONS
MAX
TAIL    SHOW
BORDERS   ENABLE
ANALYSIS   SHOW
REPEATABLE
COMMENTS
INDENTION   SHOW
ORIGINS   MAX
XREF
LENGTH   LongNameForm
NPAGESIZE   PACK
DATABASE  SHOW
EMPTYLINES   LIST
NAMES
NameChars   SHOW
SEGMENTS  AsciiStringChars  ASCII
TYPE
AUTO
VPAGESIZE   SHOW
SEGXREFS  DUMMY
NAMES
TYPE  ANALYSIS
SubstChar   SWAP
EXPANDED  MAX
```

```
NAMES
LENGTH  NOVICE
MangleChars SWAP
EXTENDED  DEFAULT
PROCESSOR
SHOW
VOIDS  XlatAsciiName  SHOW
AUTOCOMMENTS
SHOW
XREFS  ASCII
GENNAMES SHOW
LINEPREFIXES
SCREEN
MODE PRIORITY
CLASS USE
SEGMENT
NAMES
 Do flags contain byte value?
. has the byte a value
 if not, the byte is uninitialized.
#define hasValue(
)
)
 any defined value?
: can't open library
2 %
: bad library format
%
: unknown error

Load File of New Format

  Load file %A as

~inary file:



~oading segment:
(EXE
 BIN)
  <Loading  ~
~ffset:
)


~reate segments:
>      (
<Load ~
~esources:
<Rename DLL en~
~ries:
~anual load:
~ill segment gaps:
>    (
<Make ~
~mports section:
```

```
~lign segments:
<IBM ob~
~ect file:
      (
)


~LL directory:
: can't open archive
2 %
: archive read error
3 %
: bad archive format
4 %
: not enough memory
%
: unknown error
WRAP
Bad Object File Format
```

The input file format is illegal. This message may occur because of wrong OMF type. There are 2 major OMF types: IBM and Microsoft. In most cases Microsoft format is used. Unfortunately, it is not possible to discern IBM and Microsoft. You can specify OMF type in
load file
 dialog.

The OMF format is too complicated and has many dialects so this message may occur due to a bug in IDA. Please double check that the input file is correct and send it to support
datarescue.com if you are sure that
the input file is ok.

```
; Alignment     : 16 bytes by default
Reading exports directory
Segment ~
~egisters
 Get byte value from flags

 Get value of byte provided that the byte is initialized.
 This macro works ok only for 8-bit byte machines.
#define byteValue(
)    (
)
 quick replacement for Byte
Illegal type of value for the keyword
 Is the byte initialized?
#define isLoaded(
)    hasValue(GetFlags(
 any defined value?
~ectors
~ames
~ss references
Struc~
~ures
~erations
Create segment
08lX
08lX, sel %08lX
segment base or start addresses are invalid. The segment would have negative
offsets.
Create a segment
08lX
08lX, sel %08lX
IDP module disallowed creation of the segment.
Create a segment
08lX
08lX, sel %08lX
't trim the previous segment.

Create a segment
08lX
08lX, sel %08lX
't allocate virtual memory for the segment.

Create a segment
08lX
08lX, sel %08lX
't create a segment.
#define MS
CLS  0x00000600L
 Mask for typing
#define FF
CODE 0x00000600L
 Code ?
#define FF
DATA 0x00000400L
 Data ?
#define FF
TAIL 0x00000200L
 Tail ?
#define FF
UNK  0x00000000L
 Unknown ?
#define isCode(
)
CODE)
 is code byte?
#define isData(
)
DATA)
 is data byte?
#define isTail(
)
TAIL)
 is tail byte?
#define isUnknown(
)
 is unexplored byte?
#define isHead(
)
DATA)
)
 is start of code/data?
```

```
~natures
      Common bits

#define MS
COMM 0x000FF800L
 Mask of common bits
#define FF
COMM 0x00000800L
 Has comment?
#define FF
REF  0x00001000L
 has references?
#define FF
LINE 0x00002000L
 Has next or prev cmt lines ?
#define FF
NAME 0x00004000L
 Has user-defined name ?
#define FF
LABL 0x00008000L
 Has dummy name?
#define FF
FLOW 0x00010000L
 Exec flow from prev instruction?
#define FF
VAR  0x00080000L
 Is byte variable ?
#define isFlow  (
)
FLOW)
#define isVar   (
)
VAR )
#define isExtra (
)
LINE)
#define isRef   (
)
#define hasName (
)
NAME)
Internal ~
~lags
Reading imports directory
:1219

Convert to
 en masse

  Apply to:

~ll operands:
>

~oid operands:
>
```

```
~perands:
>

%s operands:
>
        <Not ~
~yped operands:


        <
~ply only if possible:


 Please enter lower and upper limits
  of VOID OPERAND values (inclusive)


~ower value:


~pper value:
#define MS
0TYPE 0x00F00000L
 Mask for 1st arg typing
#define FF
0VOID 0x00000000L
 Void (unknown
#define FF
0NUMH 0x00100000L
 Hexadecimal number?
#define FF
0NUMD 0x00200000L
 Decimal number?
#define FF
0CHAR 0x00300000L
 Char
#define FF
0SEG  0x00400000L
 Segment?
#define FF
0OFF  0x00500000L
 Offset?
#define FF
0NUMB 0x00600000L
 Binary number?
#define FF
0NUMO 0x00700000L
 Octal number?
#define FF
0ENUM 0x00800000L
 Enumeration?
#define FF
0FOP  0x00900000L
 Forced operand?
#define FF
0STRO 0x00A00000L
 Struct offset?
#define MS
1TYPE 0x0F000000L
 Mask for 2nd arg typing
#define FF
1VOID 0x00000000L
 Void (unknown
#define FF
1NUMH 0x01000000L
 Hexadecimal number?
#define FF
1NUMD 0x02000000L
 Decimal number?
#define FF
1CHAR 0x03000000L
 Char
#define FF
1SEG  0x04000000L
 Segment?
#define FF
1OFF  0x05000000L
 Offset?
#define FF
1NUMB 0x06000000L
 Binary number?
#define FF
1NUMO 0x07000000L
 Octal number?
#define FF
1ENUM 0x08000000L
 Enumeration?
#define FF
1FOP  0x09000000L
 Forced operand?
#define FF
1STRO 0x0A000000L
 Struct offset?
 The following macros answer questions like

'is the 1st (or 2nd) operand of instruction or data of the given type
 Please note that data items use only the 1st operand type (
#define isDefArg0(
)
0TYPE)
0VOID)
#define isDefArg1(
)
1TYPE)
1VOID)
#define isDec0(
)
0TYPE)
0NUMD)
#define isDec1(
)
1TYPE)
1NUMD)
#define isHex0(
)
0TYPE)
0NUMH)
#define isHex1(
)
1TYPE)
1NUMH)
#define isOct0(
)
0TYPE)
0NUMO)
#define isOct1(
)
1TYPE)
1NUMO)
#define isBin0(
)
0TYPE)
0NUMB)
#define isBin1(
)
1TYPE)
1NUMB)
#define isOff0(
)
0TYPE)
0OFF)
#define isOff1(
)
1TYPE)
1OFF)
#define isChar0(
)
0TYPE)
0CHAR)
#define isChar1(
)
1TYPE)
1CHAR)
#define isSeg0(
)
0TYPE)
0SEG)
#define isSeg1(
)
1TYPE)
1SEG)
#define isEnum0(
)
0TYPE)
0ENUM)
#define isEnum1(
)
1TYPE)
1ENUM)
#define isFop0(
)
```

```
0TYPE)
0FOP)
#define isFop1(
)
1TYPE)
1FOP)
#define isStroff0(
)
0TYPE)
0STRO)
#define isStroff1(
)
1TYPE)
1STRO)
#define isStkvar0(
)
0TYPE)
0STK)
#define isStkvar1(
)
1TYPE)
1STK)
      Bits for DATA bytes

#define DT
TYPE 0xF0000000L
 Mask for DATA typing

#define FF
BYTE 0x00000000L
 byte
#define FF
WORD 0x10000000L
 word
#define FF
DWRD 0x20000000L
 dword
#define FF
QWRD 0x30000000L
 qword
#define FF
TBYT 0x40000000L
 tbyte
#define FF
ASCI 0x50000000L
 ASCII ?
#define FF
STRU 0x60000000L
 Struct ?
#define FF
XTRN 0x70000000L
 Extern data, unknown size
#define FF
FLOAT 0x80000000L
 float
#define FF
DOUBLE 0x90000000L
```

```
 double
#define FF
PACKREAL 0xA0000000L
 packed decimal real
#define FF
ALIGN    0xB0000000L
 alignment directive


      Bits for CODE bytes

#define MS
CODE 0xF0000000L
#define FF
FUNC 0x10000000L
 function start?
#define FF
IMMD 0x40000000L
 Has Immediate value ?
#define FF
JUMP 0x80000000L
 Has jump table

 Return value of expression:
+ off)

long    MK
FP           (long seg,long off

 the same as [ seg, off ]

 Return a formatted string.
      format - printf-style format string.
               %
- means address expression.
               floating point values are output only in one format

                regardless of the character specified (
               %p is not supported.
 The resulting string must be less than 255 characters

char    form            (char format

 works as sprintf

 The resulting string should

 be less than 255 characters.
 Return substring of a string

      str - input string

      x1  - starting index (
      x2  - ending index. If x2
, then return substring

            from x1 to the end of string.
```

```
char    substr          (char str,long x1,long x2
 substring [
]

 if x2
, then till end of line

 Search a substring in a string

      str    - input string

      substr - substring to search

 returns:
- index in the '
' where the substring starts

          -1   - if the substring is not found

long    strstr          (char str,char substr
 find a substring,
- not found


~ file
~dit file
No hot key is defined for
Current Processor:
Target Assembler:
 Return length of a string in bytes

      str - input string

 Returns: length (
)

long    strlen          (char str

 calculate length

 Convert ascii string to a binary number.
(this function is the same as hexadecimal 'strtol' from C library)

long    xtol            (char str

 ascii hex
 number

(use long
 for atol)
 Convert address value to a string

char    atoa            (long ea

 returns address in
 the form 'seg000:1234'

(the same as in line prefixes)
 Convert a number to a string.
      n - number

      radix - number base (
)

char    ltoa            (long n,long radix

 convert to ascii string

 Convert ascii string to a number

      str - a decimal representation of a number

 returns: a binary number

long    atol            (char str

 convert ascii decimal to long

 Add hotkey for IDC function

      hotkey  - hotkey name
, etc)
      idcfunc - IDC function name

 returns:
#endif
#define IDCHK
OK        0
 ok
#define IDCHK
ARG       -1
 bad argument(
#define IDCHK
KEY       -2
 bad hotkey name
#define IDCHK
MAX       -3
 too many IDC hotkeys
#ifdef
notdefinedsymbol

long AddHotkey(char hotkey,char idcfunc
 Delete IDC function hotkey

success DelHotkey(char hotkey
 Move cursor to the specifed linear address

      ea - linear address

success Jump            (long ea

 move cursor to ea
 screen is refreshed at

 the end of IDC execution

 Wait for the end of autoanalysis

 This function will suspend execution of IDC program

 till the autoanalysis queue is empty.

void    Wait

 Process all entries in the

 autoanalysis queue

 SEGDEF
Referenced
 Compile an IDC file.
 The file being compiled should not contain functions that are

 currently executing - otherwise the behaviour of the replaced

 functions is undefined.
      filename - name of file to compile

 returns:
, otherwise it returns an error message.

char    Compile         (char filename

 Compile an IDC file.

 returns error message.
 Stop execution of IDC program, close the database and exit to OS

      code - code to exit with.

void    Exit            (long code

 Exit to OS


Do you really want to abort IDA?

Retrive predefined comment


~nstruction mnemonics:
~perand values       :
 Delete all segments, instructions, comments,
. everything

 except values of bytes.

void    DeleteAll
```

```
 delete ALL information

 about the program


Array size (in elements)

  Current address      :

  Next defined item at :

  Next named   item at :
%A
  Array element width  :
%D
  Maximal possible size:
%D
  Current array size   :
%D
  Suggested array size :
%D

  <Array ~
~ize     :
(in elements)
~tems on a line:
~lignment      :
-none,
-auto)

  <Use
 construct:
~ned elements    :
<Create as a~
~ray    :
Purges:
 Create an instruction at the specified address

      ea - linear address

 returns:
- can't create an instruction (no such opcode, the instruction would

              overlap with existing items, etc)
          otherwise returns length of the instruction in bytes

long    MakeCode        (long ea

 convert to instruction

 returns number of bytes

 occupied by the instruction


 Perform full analysis of the area
```

```
      sEA - starting linear address

      eEA - ending linear address (excluded)
 returns:
-Ctrl-Break was pressed.

long    AnalyseArea     (long sEA,long eEA

 analyse area and try to

 convert to code all bytes

 Returns 1-
-CtrlBreak pressed

 Rename a byte

      ea - linear address

      name - new name of address. If name
, then delete old name

 returns:
-failure

success MakeName        (long ea,char name

 assign a name to a location


 Set an indented regular comment of an item

      ea      - linear address

      comment - comment string

success MakeComm        (long ea,char comment
 give a comment

Choose a enum
Comment:
Repeatable Comment:
%9lu %
%4u allocating memory for virtual array
an archive format is detected, exiting.
 Set an indented repeatable comment of an item

      ea      - linear address

      comment - comment string

success MakeRptCmt      (long ea,char comment
 give a repeatable comment

 Create an array.
      ea      - linear address
      nitems  - size of array in items

 This function will create an array of the items with the same type as the

 type of the item at '
 If the byte at '
' is undefined, then this

 function will create an array of bytes.

success MakeArray       (long ea,long nitems
 convert to an array

 Create a string.
 This function creates a string (the style is determinted by the value

 of GetLongPrm(
STRTYPE
 see below
      ea - linear address

      endea - ending address of the string (excluded)
              if endea
 BADADDR, then length of string will be calculated

              the the kernel

 returns:
-failure

success MakeStr         (long ea,long endea
 convert to ASCII string

Please wait
 Convert the current item to a byte

      ea - linear address

 returns:
-failure

success MakeByte        (long ea

 convert to byte


Setup data types

  Immediately convert the |   Use the following types
  current item to:        |   in the data carousel:
~yte       :
         |
~ Byte:
~ord       :
         |
~ Word:
```

```
~ouble word:
         |
~ Double word:
~loat      :
         |
~ Float:
~uadro word:
         |
~ Quadro word:
~ble     :
         |
~ Double:
~byte      :
         |
~ Tbyte:
~acked real:
         |
~ Packed real:
Analysing area %08lX
08lX
:1399

Sorry, the demo version can't produce asm files.
Press F1 for details.

  bytes   pages size description

%9lu %
%4u allocating memory for b-tree
 Convert the current item to a word (2 bytes)
      ea - linear address

 returns:
-failure

success MakeWord        (long ea

 convert to word

Create a segment
08lX
08lX
end address is lower than start address.
WRAP
Configuration file


Configuration file '
' is searched in IDA.EXE directory and PATH.
If it is not found, IDA uses default values.
In the configuration file you can use C,
 style comments and include files.
Configuration file syntax is:

        KeyWord
 Value
```

```
ActionName
 Value

  where value may be:

        a string:
Ctrl-

        a char:         '
'
        a scancode:     0x4900
        zero:           0

  Zero scancode disables the hotkey.

  Also it is possible to define keyboard macros:

     MACRO key { key1 key2 key3
 keyN }

  where key is a string (key name
 char or a scancode.
  Example:

        MACRO
 this sample macro jumps to
start
 label
        {

Ctrl-

                '
'

Enter

        }

Below is the list of allowed keywords:

 ASCII
GENNAMES      MAX
DATALINE
LENGTH        SHOW
LINEPREFIXES
 ASCII
LINEBREAK     MAX
NAMES
LENGTH           SHOW
ORIGINS
 ASCII
STYLE         MAX
TAIL                   SHOW
REPEATABLE
COMMENTS
 ASCII
```

```
PREFIX        MAX
XREF
LENGTH            SHOW
SEGMENTS
 ASCII
SERIAL        PACK
DATABASE              SHOW
SEGXREFS
 ASCII
SERNUM        OS2DIR                     SHOW
SOURCE
LINNUM
 ASCII
ZEROES        SCREEN
MODE                SHOW
VOIDS
 AUTOSAVE            SCREEN
PALETTE             SHOW
XREFS
 OPCODE
BYTES        SHOW
ASSUMES               SWAP
EXPANDED
 COMMENTS
INDENTION  SHOW
AUTOCOMMENTS          SWAP
EXTENDED
 DEFAULT
PROCESSOR   SHOW
INSTRUCTIONS      USE
FPP
 DUMMY
NAMES
TYPE    SHOW
BORDERS               USE
SEGMENT
NAMES
 ENABLE
ANALYSIS     SHOW
EMPTYLINES            USE
TABULATION
 INDENTION           SHOW
INDICATOR             WINDIR
 XlatAsciiName       AsciiStringChars           NameChars
 PRIORITY
CLASS      PRIORITY
DELTA             DATABASE
MEMORY
 VPAGESIZE           VPAGES                     NPAGESIZE
 NPAGES              MangleChars                SubstChar
 ShortNameForm       LongNameForm               LIST
NAMES
 ASCII
TYPE
AUTO     ANALYSIS                  NOVICE
Creating the output file.
%9lu %
```

```
%4u allocating memory for name pointers

Create alignment directive


 Convert the current item to a double word (4 bytes)
      ea - linear address

 returns:
-failure

success MakeDword       (long ea

 convert to double-word

Use data definition commands to create local variables and function
arguments.
Two special fields
 and
 represent return address and saved registers.
Frame size:
; Saved regs:
; Purge:
 Convert the current item to a quadro word (8 bytes)
      ea - linear address

 returns:
-failure

success MakeQword       (long ea

 convert to quadro-word


Upgrading the database.
Segment alignment
' can not be represented in assembly
1 Execute
2 Write
3 Write/Execute
4 Read
5 Read/Execute
6 Read/Write
7 Read/Write/Execute
Unknown
Segment permissions:
%s
press Enter
%lXh %loo %
IDA did not find any occurence of
' in the forced operand(
(bad xref from %
Reading fixups
; Exported entry %
Database for
' already exists. Do you want to overwrite it?
IDA Pro is currently in NOVICE Mode.
```

```
In this mode, IDA Pro's complex functions are disabled.
This message will appear each time a new file is loaded.
See IDA.CFG file to learn how to disable the novice mode
WRAP

The input file is now loaded in the database. IDA Pro is now busy
analysing it. You may either wait until the initial analysis is
over or immediately start exploring the disassembly. Some parts
of the file are probably still displayed as bytes : IDA Pro has
probably not explored them yet. You may manually speed up the
analysis and convert those bytes into instructions. Move the cursor
cursor over the first undefined byte and press C.

When the initial analysis is over, IDA Pro will beep. Use F6 to
toggle between the message and the disassembly window. A few useful
keys to get you started.

At the cursor position

  C converts bytes to code
  D converts bytes to data
  U undefines byte

Press Esc to close this window.
:1207

Color configuration

  <Palette ~
<Palette ~
<Palette ~
<Palette ~
~ustomize:



~nable syntax highlighting:
:1505

IBM PC specific analyser options

 <Convert ~
~mmediate operand of
push
 to offset          :
<Convert db 90h after
 to
                    :
<Convert immediate operand of
mov ~
 to offset   :
<Convert immediate operand of
mov ~
~emory
 to offset:
IBM PC specific analyser options
```

```
Convert immediate operand of
push
 to offset

        In sequence

                push    seg
                push    num

        IDA will try to convert <
> to offset.


Convert db 90h after
 to


        Sequence

                jmp     short label
                db      90h

        will be converted to

                jmp     short label
                nop


Convert immediate operand of
mov reg
 to offset

        In sequence

                mov reg,    num
                mov segreg, immseg

        where
          reg    - any general register
          num    - a number
          segreg - any segment register
          immseg - any form of operand representing a segment paragraph

        <
> will be converted to an offset

Convert immediate operand of
mov memory
 to offset

        In sequence

                mov x1, num
                mov x2, seg

        where
```

```
                x1,
- any references to memory

        <
> will be converted to an offset

't open palette file
' for writing
 Convert the current item to a floating point (4 bytes)
      ea - linear address

 returns:
-failure

success MakeFloat       (long ea

 convert to float

 Convert the current item to a double floating point (8 bytes)
      ea - linear address

 returns:
-failure

success MakeDouble      (long ea

 convert to double


 Convert the current item to a packed real (10 or 12 bytes)
      ea - linear address

 returns:
-failure

success MakePackReal    (long ea

 convert to packed real

%9lu            total memory allocated


 Convert the current item to a tbyte (10 or 12 bytes)
      ea - linear address

 returns:
-failure

success MakeTbyte       (long ea

 convert to 10 bytes (tbyte)
't create alignment directive (
, minalign
, maxalign
 Convert the current item to a structure instance

      ea      - linear address
      strname - name of a structure type

 returns:
-failure

success MakeStruct      (long ea,char strname
 convert to structure instance

Converting operands to
 type.
WRAP

A saved database has been loaded.
You may now resume your disassembly.

An easy way to navigate through your file is to use the ENTER and the
ESC keys. If you position the cursor on a call or a jump address, or
on a data offset and press ENTER, IDA Pro will jump to this location
and push your current location on a stack. You may take several jumps
and then return to previous locations with the ESC key.

Press Esc to continue
:1517

ELF patching (for PIC) and viewing mode


~eplace PIC form of 'Procedure Linkage Table' to non PIC form   :
~irect jumping from PLT (without GOT) irrespective of its form  :
~onvert PIC form of loading '
GLOBAL
OFFSET
TABLE
 of address:
~bliterate auxiliary bytes in PLT
 GOT for 'final autoanalysis
~atural form of PIC GOT address loading in relocatable file     :
~npatched form of PIC GOT references in relocatable file        :


     auto-process options                relocation comments
<Disable ctor/dtor r~
~naming  :
<Disable
' comment       :
<Disable coagulation of da~
<Disable
~TTENTION' comment :
<Disable alternative na~
~e    :
<Disable
~hared' comment    :
ELF patching (for PIC) mode


Replace PIC form of 'Procedure Linkage Table' to non PIC form
```

If the program was compiled in
position independent code (

mode, then the
Procedure Linkage Table (
will contain
instructions like

```
                jmp     [
]
```

where ebx points to
GLOBAL
OFFSET
TABLE

The current version of IDA can not use there instructions to create
cross-refenreces and proper function parameters.

If this option is enabled (default
then the instruction will be replaced
by

```
                jmp     ds:
xxx
```

where off
xxx - label in the Global Offset Table. IDA will append a

PIC mode
comment to the instruction (if enabled

Direct jumping from PLT (without GOT) irrespective of its form

All jump instructions in the Procedure Linkage Table refer to the
Global Offset Table. The Global Offset Table is a simple table of
offsets to enternal names (for PIC programs GOT may contain offsets
to internal names too)

If this option is enabled, all instructions in the Program Linkage Table
are replaced by:

```
                jmp     ds:name
```

where name - the target address of original jump instruction

Convert PIC form of loading '
GLOBAL
OFFSET
TABLE
of address

All position independent programs use EBX register to access to
Global Offset Table. They use the following snippet to load EBX:

```
                call
+5
        temp:
                pop     ebx
                add     ebx, offset
GLOBAL
OFFSET
TABLE
- offset temp
```

If this option is enabled, IDA will replace it by:

```
                nop             ;n times
                mov     ebx, offset
GLOBAL
OFFSET
TABLE
```

and append a
PIC mode
comment to it.

Obliterate the auxiliary bytes in PLT
GOT for 'final autoanalysis'

During the final pass of the analysis (see
:1730[kernel options
all undefined
bytes are converted to data and instructions.

If this option is enabled, IDA will
uninitialize
some bytes in PLT and GOT
because they usually contain garbage. Therefore, IDA will not try to create
instructions or data from garbage.

Natural form of PIC GOT address loading in relocatable file

This option means the same as 'Convert PIC form of
but is applied
only for object (relocatable) PIC files.
ATTENTION! We do not recommend to use this option because the output
assembler file will not be compilable.

Unpatched form of PIC GOT references in relocatable file

In the object (repocatable) files the references to external names
require creation of Global Offset Table. All references are made through
this table. By default, all such referneces are replaced by direct
references by IDA. If you turn this option off, you will get almost the

same instructions as in the input file, but the resulting assembler file will not be compilable.
ATTENTION! We do not recommend to use this option because the output assembler file will not be compilable.

relocation comments

Disable '
' comment

Disable
PIC mode
comment for replaced instructions.

Disable 'ATTENTION' comment

Disable
ATTENTION
comment which is created for uncompilable instructions.
These instructions may apper only if you play with the abovementioned options.

Disable 'Shared' comment

The ELF shared libraries export some data in the following way: the data should be copied to a reserved place in the program when a shared library is loaded. IDA will mark such reserved places with
Copy of Shared Data

comment.
This option allows you to disable this comment.

auto-process options

Disable ctor/dtor renaming

Disable autocreation of names ctor
/dtor
nnn for unnamed
C
constructors and destructors.

Disable coagulation of data

IDA tries to create arrays in array dimensions are known. You can disable this behaviour using this option.
If this option and the previous options are checked, then ctor/dtor tables are not converted to arrays too.

```
Disable alternative name

  Some location in the program may have several names. Unfortunately,
  is it not possible to determine which name is created by the linker and
  which - by the programmer. If a location has several names, IDA displays
  additional names as comments (in demangled form

  This option allows you to disable such comments.
:1519

Java loading options

                  Class File version %
D


<Include ~
~ocal variable declarations (if possible)      :
't create 'import' segment with external-references :
~reate 'import' segment with ordinary references       :
~ield/variable declarations are included to references :
~ethod declarations are included to references         :
Java-VM class file loading options


Include local variable declarations (if possible)


.


Don't create 'import' segment with external-references


.


Create 'import' segment with ordinary references


 XREF'
.


Field/variable declarations are included to references


 XREF'
(Filed)


    Field borland.
.AboutDialog about


 XREF
 class borland.
```

```
.AboutDialog


Method declarations are included to references


 XREF'
 XREF.


Please send the input file to <support
datarescue.
 Create a local variable

      start,end - range of addresses for the local variable

                  The current version doesn't use '
' address

                  and creates a stack variable for the whole function

                  If there is no function at 'start' then this function

                  will fail.
      location  - variable location in the form
 where xx is

                  a hexadecimal offset.
      name      - name of the local variable

 returns:
-failure

success MakeLocal(long start,long end,char location,char name
 Convert the current item to an explored item

      ea     - linear address

      expand -
: just undefine the current item

               1: undefine other instructions if the removal of the

                  current instruction removes all references to them.
                  (note: functions will not be undefined even if they

                         have no references to them)
 returns:
-failure

void    MakeUnkn        (long ea,long expand
 convert to 'unknown'

 expand!
> undefine consequent

 instructions too
 Convert an operand of the item (instruction or data) to a binary number

      ea - linear address
/       n  - number of operand

              0 - the first operand

              1 - the second, third and all other operands

              -
- all operands

 Note: the data items use only the type of the first operand

 Returns:
-failure

success OpBinary        (long ea,int n

 make operand binary

- first operand

- second, third etc. operands

- all operands


 Convert an operand of the item (instruction or data) to an octal number

(see explanation of
:1523[OpBinary] functions)

success OpOctal         (long ea,int n
 Convert operand to decimal,
,char (see
:1523[OpBinary
 for explanations)

success OpDecimal       (long ea,int n

success OpHex           (long ea,int n

success OpChr           (long ea,int n
 Convert operand to an offset

(for the explanations of '
' and '
' please see
:1523[OpBinary
      base - base of the offset as a linear address

             If base
 BADADDR then the current operand becomes non-offset

 Example:
  seg000:2000 dw      1234h
```

and there is a segment at paragraph 0x1000 and there is a data item

within the segment at 0x1234:

```
  seg000:1234 MyString        db 'Hello, world
```

Then you need to specify a linear address of the segment base to

create a proper offset:

```
      OpOffset
seg000
,0x2000
,0x10000
```

and you will have:

```
  seg000:2000 dw      offset MyString
```

Motorola 680x0 processor have a concept of
outer offsets
If you want to create an outer offset, you need to combine number

of the operand with the following bit:

```
#define OPND
OUTER      0x80
```

outer offset base

Please note that the outer offsets are meaningful only for

Motorla 680x0.

```
success OpOff           (long ea,int n,long base
```

Convert operand to a segment expression

(for the explanations of '
' and '
' please see
:1523[OpBinary

```
success OpSeg           (long ea,int n
```

Convert operand to a number (with default number base, radix)
(for the explanations of '
' and '
' please see
:1523[OpBinary

```
success OpNumber        (long ea,int n
```

Specify operand represenation manually.
(for the explanations of '
' and '
' please see
:1523[OpBinary
str - a string represenation of the operand

IDA will not check the specified operand, it will simply display

it instead of the orginal representation of the operand.

```
success OpAlt           (long ea,long n,char str
 manually enter n-th operand

 Change sign of the operand.
(for the explanations of '
' and '
' please see
:1523[OpBinary


success OpSign          (long ea,int n

 change operand sign

 Convert operand to a symbolic constant

(for the explanations of '
' and '
' please see
:1523[OpBinary
      enum - name of enumration type

success OpEnum          (long ea,int n,char enum
 make operand a enum

 Convert operand to an offset in a structure

(for the explanations of '
' and '
' please see
:1523[OpBinary
      strid - id of a structure type

success OpStroff        (long ea,int n,long strid
 make operand a struct offset

 Convert operand to a stack variable

(for the explanations of '
' and '
' please see
:1523[OpBinary


success OpStkvar        (long ea,int n

 make operand a stack variable

 Mark the location as
variable
 Note: All that IDA does is to mark the location as
variable
. Nothing else,
 no additional analysis is performed.
 This function may disappear in the future.
```

```
void    MakeVar         (long ea

 the location is 'variable'
 Specify an additional line to display before the generated ones.
      ea   - linear address

      n    - number of anterior additioal line (
      line - the line to display

 IDA displays additional lines from number 0 up to the first unexisting

 additional line.
, if you specify additional line #150 and there is no

 additional line #
, your line will not be displayed.

void    ExtLinA         (long ea,long n,char line
 insert an additional line before the generated ones

 Specify an additional line to display after the generated ones.
      ea   - linear address

      n    - number of posterior additioal line (
      line - the line to display

 IDA displays additional lines from number 0 up to the first unexisting

 additional line.
, if you specify additional line #150 and there is no

 additional line #
, your line will not be displayed.

void    ExtLinB         (long ea,long n,char line
 insert an additional line after the generated ones

 Delete an additional anterior line

      ea   - linear address

      n    - number of anterior additioal line (
)

void    DelExtLnA       (long ea,long n

 delete an additional line before the generated ones

 Delete an additional posterior line

      ea   - linear address

      n    - number of posterior additioal line (
)

void    DelExtLnB       (long ea,long n
 delete an additional line aftr  the generated ones

 Create a jump table (obsolete)
      jumpea  - address on instruction that uses the jump table

      tableea - address of jump table

      nitems  - number of items in the jump table

      is32bit -
: table entry is 16 bit

                1: table entry is 32 bit

success JmpTable        (long jmpea,long tableea,long nitems,long is32bit

 define a jump table

 Change value of a program byte

      ea    - linear address

      value - new value of the byte

void    PatchByte       (long ea,long value
 change a byte

 Change value of a program word (2 bytes)
      ea    - linear address

      value - new value of the word

void    PatchWord       (long ea,long value
 change a word (2 bytes)
 Change value of a double word

      ea    - linear address

      value - new value of the double word

void    PatchDword      (long ea,long value
 change a dword (4 bytes)
 Set new value of flags

 This function should not used be used directly if possible.
 It changes properties of a program byte and if misused, may lead to

 very-very strange results.

void    SetFlags        (long ea,long flags
 change internal flags for ea

 Set value of a segment register.
      ea - linear address

      reg - name of a register, like
, etc.
```

value - new value of the segment register.
IDA keeps tracks of all the points where segment register change their

values. This function allows you to specify the correct value of a segment

register if IDA is not able to find the corrent value.

```
success SetReg          (long ea,char reg,long value
```
set value of segment register

Plan to perform an action in the future.
This function will put your request to a special autoanalysis queue.
Later IDA will retrieve the request from the queue and process

. There are several autoanalysis queue types. IDA will process all

queries from the first queue and then switch to the second queue, etc.

```
void    AutoMark        (long ea,long queuetype
```
plan address to analyse
```
void    AutoMark2       (long start,long end,long queuetype
```

plan range of addresses

```
#define AU
UNK  10
```
make unknown
```
#define AU
CODE 20
```
convert to instruction
```
#define AU
PROC 30
```
make function
```
#define AU
USED 40
```
reanalyse
```
#define AU
LIBF 60
```
apply a flirt signature (the current signature
```
#define AU
FINAL 200
```
coagulate unexplored items

```
void    WriteMap        (char file
```

produce a .map file
```
void    WriteTxt        (char file,long ea1,long ea2
```
produce an .asm file
```
void    WriteExe        (char file
```

produce an executable file

Get internal flags

ea - linear address

```
 returns:
-bit value of internal flags. See start of IDC.IDC file

 for explanations.

long    GetFlags        (long ea

 get internal flags for ea

 Get value of program byte

      ea - linear address

 returns: value of byte. If byte has not a value then returns 0xFF

long    Byte            (long ea

 get a byte at ea

 Get value of program word (2 bytes)
      ea - linear address

 returns: value of word. If word has not a value then returns 0xFF

long    Word            (long ea

 get a word (2 bytes) at ea

 Get value of program double word (4 bytes)
      ea - linear address

 returns: value of double word. If double word has not a value

 then returns 0xFF

long    Dword           (long ea

 get a double-word (4 bytes) at ea

 Get linear address of a name

      name - name of program byte

 returns: address of the name

          BADADDR - no such name

long    LocByName       (char name

 BADADDR - no such name

 Get segment by segment base

      base - segment base paragraph or selector

 returns: linear address of the start of the segment
```

```
          BADADDR - no such segment

long    SegByBase       (long base

 BADADDR - no such segment

 Get linear address of cursor

long    ScreenEA

 the current screen ea


 Get start address of the selected area

 returns BADADDR - the user has not selected an area

long    SelStart

 the selected area start ea

 BADADDR - no selected area

 Get end address of the selected area

 returns BADADDR - the user has not selected an area

long    SelEnd

 the selected area end ea

 BADADDR - no selected area

 Get value of segment register at the specified address

      ea - linear address

      reg - name of segment register

 returns: value of segment register. The segment registers in 32bit program

 usually contain selectors, so to get paragraph pointed by the segment

 register you need to call AskSelector
 function.

long    GetReg          (long ea,char reg

 get segment register value

 BADADDR - undefined or error

(selector, use AskSelector
 to

  get its mapping)
 Get next addresss in the program

      ea - linear address

 returns: BADADDR - the specified address in the last used address

long    NextAddr        (long ea

 returns next defined address

 BADADDR if no such address exists

 Get previous addresss in the program

      ea - linear address

 returns: BADADDR - the specified address in the first address

long    PrevAddr        (long ea

 returns prev defined address

 BADADDR if no such address exists

 Get next defined item (instruction or data) in the program

      ea - linear address

 returns: BADADDR - no (more) defined items

long    NextHead        (long ea

 returns next defined item address

 BADADDR if no such address exists

 Get previous defined item (instruction or data) in the program

      ea - linear address

 returns: BADADDR - no (more) defined items

long    PrevHead        (long ea

 returns prev defined item address

 BADADDR if no such address exists

 Get next not-tail address in the program

 This function searches for the next displayable address in the program.
 The tail bytes of instructions and data are not displayable.
      ea - linear address

 returns: BADADDR - no (more) not-tail addresses

long    NextNotTail     (long ea
 returns next not tail address

 BADADDR if no such address exists

 Get previous not-tail address in the program

 This function searches for the previous displayable address in the program.
 The tail bytes of instructions and data are not displayable.
      ea - linear address

 returns: BADADDR - no (more) not-tail addresses

long    PrevNotTail     (long ea

 returns prev not tail address

 BADADDR if no such address exists


 Get address of the end of the item (instruction or data)
      ea - linear address

 returns: address past end of the item at '
'

long    ItemEnd         (long ea

 returns address past end of

 the item

 Get size of instruction or data item in bytes

      ea - linear address

 returns:
n

long    ItemSize        (long ea

 returns item size, min answer
 Get visible name of program byte

 This function returns name of byte as it is displayed on the screen.
 If a name contains illegal characters, IDA replaces them by the substitution

 character during displaying. See IDA.CFG for the definition of the

 substitution character.
      ea - linear address

 returns:
- byte has no name

char    Name            (long ea
 get visible name of the byte

 Get true name of program byte

 This function returns name of byte as is without any replacements.
      ea - linear address

 returns:
- byte has no name

char    GetTrueName     (long ea

 get true name of the byte

 Get mnemonics of instruction

      ea - linear address of instruction

 returns:
- no instruction at the specified location

 note: this function may not return exactly the same mnemonics

 as you see on the screen.

char    GetMnem         (long ea

 get instruction name

 Get operand of an instruction

      ea - linear address of instruction

      n  - number of operand:
              0 - the first operand

              1 - the second operand

 returns: the current text representation of operand

char    GetOpnd         (long ea,long n

 get instruction operand

- first operand

 Get type of instruction operand

      ea - linear address of instruction

      n  - number of operand:
              0 - the first operand

              1 - the second operand

 returns:
      -1      bad operand number passed
      0       None

      1       General Register (
      2       Memory Reference

      3       Base + Index

      4       Base + Index + Displacement

      5       Immediate

      6       Immediate Far Address

      7       Immediate Near Address

      8       FPP register

      9       386 control register

      10      386 debug register

      11      386 trace register

      12      Condition (for Z80)
      13      bit (8051)
      14      bitnot (8051)

long    GetOpType       (long ea,long n

 get operand type

 Get number used in the operand

 This function returns an immediate number used in the operand

      ea - linear address of instruction

      n  - number of operand:
              0 - the first operand

              1 - the second operand

 If the operand doesn't contain a number, it returns -
.

long    GetOperandValue (long ea,long n

 get instruction operand value

 Get anterior comment line

      ea - linear address

      num - number of anterior line (
)
```

```
char    LineA           (long ea,long num

 get additional line before generated ones

 Get posterior comment line

      ea - linear address

      num - number of posterior line (
)

char    LineB           (long ea,long num

 get additional line after generated ones

 Get regular indented comment

      ea - linear address

char    Comment         (long ea

 get comment

 Get repeatable indented comment

      ea - linear address

char    RptCmt          (long ea

 get repeatable comment

 Get manually entered operand string

      ea - linear address

      n  - number of operand:
              0 - the first operand

              1 - the second operand

char    AltOp           (long ea,long n

 get manually entered operand

      The following functions search for the specified byte

              ea - address to start from

              flag |
- search forward

              flag |
- search case-sensitive (only for FindText)
      return BADADDR - not found


long    FindVoid        (long ea,long flag
long    FindCode        (long ea,long flag

long    FindData        (long ea,long flag

long    FindProc        (long ea,long flag

long    FindUnexplored  (long ea,long flag

long    FindExplored    (long ea,long flag

long    FindImmediate   (long ea,long flag,long value

long    FindText        (long ea,long flag,long y,long x,char str
```

- number of text line at ea to start from (
)

- x coordinate in this line

```
long    FindBinary      (long ea,long flag,char str
```

str - a string as a user enters it for Search Text in Core

example:
41 42
- find 2 bytes 41h,42h

The default radix depends on the current IDP module

(radix for ibm pc is 16)
The following functions allow you to set/get common parameters.

```
long    GetLongPrm (long offset

long    GetShortPrm(long offset

long    GetCharPrm (long offset

success SetLongPrm (long offset,long value

success SetShortPrm(long offset,long value

success SetCharPrm (long offset,long value
```

'offset' may be one of the following:

INF
VERSION
short; Version of database
INF
PROCNAME
char[
Name of current processor
INF
LFLAGS

```
 char;  IDP-dependent flags
  LFLG
FPP
          decode floating point processor

          instructions?
  LFLG
FLAT
          Flat model?
INF
DEMNAMES
 char;  display demangled names as:
  DEMNAM
CMNT
          comments
  DEMNAM
NAME
          regular names
  DEMNAM
NONE
          don't display
INF
FILETYPE
 short; type of input file (see core.
)
  FT
EXE
          MS DOS EXE File
  FT
COM
          MS DOS COM File
  FT
BIN
          Binary File
  FT
DRV
          MS DOS Driver
  FT
WIN
          New Executable (
)
  FT
HEX
          Intel Hex Object File
  FT
MEX
          MOS Technology Hex Object File
  FT
LX
          Linear Executable (
)
  FT
LE
          Linear Executable (
)
  FT
NLM
```

```
          Netware Loadable Module (
)
  FT
COFF
          Common Object File Format (COFF)
  FT
PE
          Portable Executable (
)
  FT
USER
          file is loaded using IDP loader function
  FT
OMF
          Object Module Format
  FT
SREC
          R-records
  FT
ZIP
          ZIP file
  FT
OMFLIB
          Library of OMF Modules
  FT
AR
          ar library
  FT
LOADER
          file is loaded using LOADER DLL
  FT
ELF
          Executable and Linkable Format (
)
  FT
W32RUN
          Watcom DOS32 Extender (W32RUN)
  FT
AOUT
          Linux a.out (AOUT)
INF
OSTYPE
 short; OS type the program is for
  OSTYPE
MSDOS
  OSTYPE
WIN
  OSTYPE
OS2
  OSTYPE
NETW
INF
APPTYPE
 short; Application type
  APPT
CONSOLE
          console
  APPT
GRAPHIC
          graphics
  APPT
PROGRAM
          EXE
  APPT
LIBRARY
          DLL
  APPT
DRIVER
          DRIVER
  APPT
1THREAD
          Singlethread
  APPT
MTHREAD
          Multithread
  APPT
16BIT
          16 bit application
  APPT
32BIT
          32 bit application
INF
START
SP
 long;  SP register value at the start of

        program execution
INF
START
AF
 short; Analysis flags:
  AF
FIXUP
          Create offsets and segments using fixup info
  AF
MARKCODE
          Mark typical code sequences as code
  AF
UNK
          Delete instructions with no xrefs
  AF
CODE
          Trace execution flow
  AF
PROC
          Create functions if call is present
  AF
USED
          Analyse and create all xrefs
  AF
FLIRT
          Use flirt signatures
  AF
PROCPTR
```

```
          Create function if data xref data
code32 exists
  AF
JFUNC
          Rename jump functions as j

  AF
NULLSUB
          Rename empty functions as nullsub

  AF
LVAR
          Create stack variables
  AF
TRACE
          Trace stack pointer
  AF
ASCII
          Create ascii string if data xref exists
  AF
IMMOFF
          Convert 32bit instruction operand to offset
  AF
DREFOFF
          Create offset if data xref to seg32 exists
  AF
FINAL
          Final pass of analysis
INF
START
IP
 long;  IP register value at the start of

        program execution
INF
BEGIN
EA
 long;  Linear address of program entry point
INF
EA
 long;  The lowest address used

        in the program
INF
EA
 long;  The highest address used

        in the program - 1
INF
OFF
 long;  low limit of voids
INF
HIGH
OFF
 long;  high limit of voids
INF
MAXREF
```

```
 long;  max xref depth
INF
ASCII
BREAK
 char;  ASCII line break symbol
INF
INDENT
 char;  Indention for instructions
INF
COMMENT
 char;  Indention for comments
INF
XREFNUM
 char;  Number of references to generate

        0 - xrefs won't be generated at all
INF
ENTAB
 char;  Use
' chars in the output file?
INF
VOIDS
 char;  Display void marks?
INF
SHOWAUTO
 char;  Display autoanalysis indicator?
INF
AUTO
 char;  Autoanalysis is enabled?
INF
BORDER
 char;  Generate borders?
INF
NULL
 char;  Generate empty lines?
INF
SHOWPREF
 char;  Show line prefixes?
INF
PREFSEG
 char;  line prefixes with segment name?
INF
ASMTYPE
 char;  target assembler number (
)
INF
BASEADDR
 long;  base paragraph of the program
INF
XREFS
 char;  xrefs representation:
  SW
SEGXRF
          show segments in xrefs?
  SW
XRFMRK
          show xref type marks?
  SW
XRFFNC
          show function offsets?
  SW
XRFVAL
          show xref values?
(otherwise-
)
INF
BINPREF
 short;
# of instruction bytes to show

          in line prefix
INF
CMTFLAG
 char;  comments:
  SW
RPTCMT
          show repeatable comments?
  SW
ALLCMT
          comment all lines?
  SW
NOCMT
          no comments at all
  SW
LINNUM
          show source line numbers
INF
NAMETYPE
 char;  dummy names represenation type
  NM
OFF
  NM
OFF
  NM
OFF
  NM
EA
  NM
EA
  NM
EA
  NM
EA
  NM
EA4
  NM
EA8
  NM
SHORT
  NM
SERIAL
INF
SHOWBADS
 char;  show bad instructions?
```

```
        an instruction is bad if it appears

        in the ash.badworks array

INF
PREFFLAG
 char;  line prefix type:
  PREF
SEGADR
          show segment addresses?
  PREF
FNCOFF
          function offsets?

INF
PACKBASE
 char;  pack database?

INF
ASCIIFLAGS
 uchar; ascii flags
  ASCF
GEN
          generate ASCII names?
  ASCF
AUTO
          ASCII names have 'autogenerated' bit?
  ASCF
SERIAL
          generate serial names?

INF
LISTNAMES
 uchar; What names should be included in the list?
  LN
NORMAL
          normal names
  LN
PUBLIC
          public names
  LN
AUTO
          autogenerated names
  LN
WEAK
          weak names

INF
START
SS
 long;
INF
START
CS
 long;
INF
```

```
STRTYPE
 ulong; current ascii string type
  ASCSTR
TERMCHR
          Character-terminated ASCII string

          The termination characters are kept in

          the next bytes of string type
  STRTERM1(strtype)
strtype
0xFF)
  STRTERM2(strtype)
strtype
0xFF)

          if the second termination character is


 then it doesn't exist.
  ASCSTR
PASCAL
          Pascal-style ASCII string (length byte)
  ASCSTR
LEN2
          Pascal-style, length has 2 bytes
  ASCSTR
UNICODE
          Unicode string
INF
AF2
 ushort;Analysis flags 2
  AF2
JUMPTBL
          Locate and create jump tables

 Change current processor

      processor - name of processor in short form.
                  run 'ida
 to get list of allowed processor types

success SetPrcsr        (char processor

 set processor type

 Set current search direction

      direction:
- down

                -
- up

 returns old value of direction flag

long    Direction       (long direction
 Enable/disable batch mode of operation

      batch:  0 - ida will display dialog boxes and wait for the user input

              1 - ida will not display dialog boxes, warnings, etc.
 returns: old balue of batch flag

long    Batch           (long batch

 enable/disable batch mode

 returns old value


char    AskStr          (char defval,char prompt
 ask a string
char    AskFile         (char mask,char prompt
 ask a file name
long    AskAddr         (long defval,char prompt
 BADADDR - no or bad input
long    AskSeg          (long defval,char prompt
 BADADDR - no or bad input
char    AskIdent        (char defval,char prompt

long    AskYN           (long defval,char prompt
:cancel,
-ok
void    Message         (char format
 show a message in msg window
void    Warning         (char format
 show a warning a dialog box
void    Fatal           (char format
 exit IDA immediately

 get a selector value
        arguments:      sel - the selector
        returns:        selector value if found
                        otherwise the input value (
)
        note:           selector values are always in paragraphs

long    AskSelector     (long sel

 returns paragraph

 find a selector which has the specifed value
        arguments:      val - value to search for
        returns:        selector if found
                        otherwise the input value (
)
        note:           selector values are always in paragraphs

long    FindSelector    (long val
 set a selector value
        arguments:      sel - the selector, should be
                        less than 0xFFFF
                        val - new value of selector
        returns:        nothing
        note:           ida supports up to 64 selectors.
                        if '
' then the
                        selector is destroyed because
                        it has no importance

void    SetSelector     (long sel,long value
 delete a selector
        arguments:      sel - the selector to delete
        returns:        nothing
        note:           if the selector is found, it will
                        be deleted

void    DelSelector     (long sel
 Get first segment

 returns: linear address of the start of the first segment

 BADADDR - no segments are defined

long    FirstSeg

 returns start of the first

 segment, BADADDR - no segments

 Get next segment

      ea - linear address

 returns: start of the next segment

          BADADDR - no next segment

long    NextSeg         (long ea

 returns start of the next

 segment, BADADDR - no more segs


 Get start address of a segment

      ea - any address in the segment

 returns: start of segment

          BADADDR - the specified address doesn't belong to any segment

long    SegStart        (long ea

 returns start of the segment

 BADADDR if bad address passed

 Get end address of a segment
```

```
      ea - any address in the segment

 returns: end of segment (an address past end of the segment)
          BADADDR - the specified address doesn't belong to any segment

long    SegEnd          (long ea

 return end of the segment

 this address doesn't belong

 to the segment

 BADADDR if bad address passed

 Get name of a segment

      ea - any address in the segment

 returns:
- no segment at the specified address

char    SegName         (long ea

 returns name of the segment

 if bad address passed


 Create a new segment

      startea  - linear address of the start of the segment

      endea    - linear address of the end of the segment

                 this address will not belong to the segment

                 'endea' should be higher than 'startea'
      base     - base paragraph or selector of the segment.
                 a paragraph is 16byte memory chunk.
                 If a selector value is specified, the selector should be

                 already defined.
      use32    -
: 16bit segment,
: 32bit segment

      align    - segment alignment. see
:1594[below] for alignment values

      comb     - segment combination. see
:1595[below] for combination values.
 returns:
-failed,
-ok
```

```
success SegCreate(long startea,long endea,long base,
                                           long use32,long align,long comb
 Delete a segment

   ea      - any address in the segment

   disable -
: discard all bytes of the segment from the disassembled text

             0: retain byte values

success SegDelete       (long ea,long disable
 Change segment boundaries

   ea      - any address in the segment

   startea - new start address of the segment

   endea   - new end address of the segment

   disable - discard bytes that go out of the segment

success SegBounds       (long ea,long startea,long endea,long disable
 Change name of the segment

   ea      - any address in the segment

   name    - new name of the segment

success SegRename       (long ea,char name
 Change class of the segment

   ea      - any address in the segment

   class   - new class of the segment

success SegClass        (long ea,char class
 Change alignment of the segment

   ea      - any address in the segment

   align   - new alignment of the segment

success SegAlign        (long ea,long alignment
#define saAbs      0
 Absolute segment.
#define saRelByte  1
 Relocatable, byte aligned.
#define saRelWord  2
 Relocatable, word (
-byte,
) aligned.
#define saRelPara  3
 Relocatable, paragraph (
-byte) aligned.
#define saRelPage  4
 Relocatable, aligned on 256-byte boundary (
page


 in the original Intel specification
#define saRelDble  5
 Relocatable, aligned on a double word (
-byte)

 boundary. This value is used by the PharLap OMF for

 the same alignment.
#define saRel4K    6
 This value is used by the PharLap OMF for page (
)

 alignment. It is not supported by LINK.
#define saGroup    7
 Segment group
#define saRel32Bytes 8
 32 bytes
#define saRel64Bytes 9
 64 bytes
#define saRelQword 10
 8 bytes

 Change combination of the segment

   ea      - any address in the segment

   comb    - new combination of the segment

success SegComb         (long segea,long comb
#define scPriv     0
 Private. Do not combine with any other program

 segment.
#define scPub      2
 Public. Combine by appending at an offset that meets

 the alignment requirement.
#define scPub2     4
 As defined by Microsoft, same as C
(public
#define scStack    5
 Stack. Combine as for C
. This combine type forces

 byte alignment.
#define scCommon   6
 Common. Combine by overlay using maximum size.
#define scPub3     7
 As defined by Microsoft, same as C
(public
 Change segment addressing

   ea      - any address in the segment
   use32   -
: 16bit,
: 32bit

success SegAddrng       (long ea,long use32
 Get segment by name

      segname - name of segment

 returns: segment base address or BADADDR

long    SegByName       (char segname

 returns segment base

 Set default segment register value for a segment

   ea      - any address in the segment

   reg     - name of segment register

   value   - default value of segment register.
-undefined.

success SegDefReg       (long ea,char reg,long value
 set segment type
        arguments:      segea - any address within segment
                        type  - new segment type:
#define SEG
NORM        0
#define SEG
XTRN        1
* segment with 'extern' definitions

   no instructions are allowed
#define SEG
CODE        2
 pure code segment
#define SEG
DATA        3
 pure data segment
#define SEG
IMP         4
 implementation segment
#define SEG
GRP         6
* group of segments

   no instructions are allowed
#define SEG
NULL        7
 zero-length segment
#define SEG
UNDF        8
 undefined segment type
#define SEG
BSS         9
```

```
 uninitialized segment
#define SEG
ABSSYM     10
* segment with definitions of absolute symbols

   no instructions are allowed
#define SEG
COMM       11
* segment with communal definitions

   no instructions are allowed

        returns:        !
- ok

success SetSegmentType  (long segea,long type
 get segment attribute
        arguments:      segea - any address within segment

long    GetSegmentAttr  (long segea,long attr
#define  SEGATTR
ALIGN  20
 alignment
#define  SEGATTR
COMB   21
 combination
#define  SEGATTR
PERM   22
 permissions
#define  SEGATTR
USE32  23
 use32 (
-bit segment
#define  SEGATTR
FLAGS  24
 segment flags
#define  SEGATTR
SEL    26
 segment selector
#define  SEGATTR
ES 28
 default ES value
#define  SEGATTR
CS 30
 default CS value
#define  SEGATTR
SS 32
 default SS value
#define  SEGATTR
DS 34
 default DS value
#define  SEGATTR
FS 36
 default FS value
#define  SEGATTR
GS 38
 default GS value
```

```
#define  SEGATTR
TYPE   40
 segment type


      Flow types:
#define fl
CF   16
 Call Far
#define fl
CN   17
 Call Near
#define fl
JF   18
 Jump Far
#define fl
JN   19
 Jump Near
#define fl
US   20
 User specified
#define fl
F    21
 Ordinary flow

 Mark exec flow 'from'
'
void    AddCodeXref(long From,long To,long flowtype

long    DelCodeXref(long From,long To,int undef
 Unmark exec flow 'from'
'

 undef - make '
' undefined if no

        more references to it

 returns 1 - planned to be

 made undefined


 The following functions include the ordinary flows:
long    Rfirst  (long From

 Get first xref from 'From'
long    Rnext   (long From,long current
 Get next xref from
long    RfirstB (long To

 Get first xref to '
'
long    RnextB  (long To,long current
 Get next xref to '
 The following functions don't take into account the ordinary flows:
long    Rfirst0 (long From
long    Rnext0  (long From,long current

long    RfirstB0(long To

long    RnextB0 (long To,long current
      Data reference types:
#define dr
O    1
 Offset
#define dr
W    2
 Write
#define dr
R    3
 Read
#define dr
T    4
 Text (names in manual operands)

void    add
dref(long From,long To,long drefType

 Create Data Ref
void    del
dref(long From,long To

 Unmark Data Ref

long    Dfirst  (long From

 Get first refered address
long    Dnext   (long From,long current

long    DfirstB (long To

 Get first referee address
long    DnextB  (long To,long current


long    XrefType(void

 returns type of the last xref

 obtained by [
]first/next[
]

 functions. Return values

 are fl
 or dr
 set number of displayed xrefs
#define XrefShow(
)            SetCharPrm(
XREFNUM,
 open a file
```

```
        arguments: similiar to C fopen

        returns:        0 -error
                        otherwise a file handle

long    fopen           (char file,char mode
 close a file
        arguments:      file handle
        returns:        nothing

void    fclose          (long handle
 get file length
        arguments:      file handle
        returns:        -
- error
                        otherwise file length in bytes

long    filelength      (long handle
 set cursor position in the file
        arguments:      handle  - file handle
                        offset  - offset from origin
                        origin  -
 from start of file
                                 1
 from current cursor position
                                 2
 from end of file
        returns:        0 - ok
                        otherwise error

long    fseek           (long handle,long offset,long origin
 get cursor position in the file
        arguments:      file handle
        returns:        -
- error
                        otherwise current cursor position

long    ftell           (long handle
 load file into IDA database
        arguments:      handle  - file handle
                        pos     - position in the file
                        ea      - linear address to load
                        size    - number of bytes to load
        returns:        0 - error
                        1 - ok

success loadfile        (long handle,long pos,long ea,long size
 save from IDA database to file
        arguments:      handle  - file handle
                        pos     - position in the file
                        ea      - linear address to save from
                        size    - number of bytes to save
        returns:        0 - error
                        1 - ok

success savefile        (long handle,long pos,long ea,long size
 read one byte from file
```

```
        arguments:      handle  - file handle
        returns:        -
- error
                        otherwise a byte read.

long    fgetc           (long handle
 write one byte to file
        arguments:      handle  - file handle
                        byte    - byte to write
        returns:        0 - ok
                        -
- error

long    fputc           (long byte,long handle
 fprintf
        arguments:      handle  - file handle
                        format  - format string
        returns:        0 - ok
                        -
- error

long    fprintf         (long handle,char format
 read 2 bytes from file
        arguments:      handle  - file hanlde
                        mostfirst 0 - least significant byte is first (intel)
                                  1 - most  significant byte is first
        returns:        -
- error
                        otherwise: a 16-bit value

long    readshort       (long handle,long mostfirst
 read 4 bytes from file
        arguments:      handle  - file hanlde
                        mostfirst 0 - least significant byte is first (intel)
                                  1 - most  significant byte is first
        returns:        a 32-bit value

long    readlong        (long handle,long mostfirst
 write 2 bytes to file
        arguments:      handle  - file hanlde
                        word    - a 16-bit value to write
                        mostfirst 0 - least significant byte is first (intel)
                                  1 - most  significant byte is first
        returns:        0 - ok

long    writeshort      (long handle,long word,long mostfirst
 write 4 bytes to file
        arguments:      handle  - file hanlde
                        dword   - a 32-bit value to write
                        mostfirst 0 - least significant byte is first (intel)
                                  1 - most  significant byte is first
        returns:        0 - ok

long    writelong       (long handle,long dword,long mostfirst
 read a string from file
        arguments:      handle  - file hanlde
        returns:        a string
```

```
                        on EOF, returns -1

char    readstr         (long handle
 write a string to file
        arguments:      handle  - file hanlde
                        str     - string to write
        returns:        0 - ok

long    writestr        (long handle,char str
 create a function
        arguments:      start,end - function bounds
        returns:        !
- ok

success MakeFunction(long start,long end
 delete a function
        arguments:      ea - any address belonging to the function
        returns:        !
- ok

success DelFunction(long ea
 change function end address
        arguments:      ea - any address belonging to the function
                        end - new function end address
        returns:        !
- ok

success SetFunctionEnd(long ea,long end
 find next function
        arguments:      ea - any address belonging to the function
        returns:        -
- no more functions
                        otherwise returns the next function start address

long NextFunction(long ea
 find previous function
        arguments:      ea - any address belonging to the function
        returns:        -
- no more functions
                        otherwise returns the previous function start address

long PrevFunction(long ea)
 retrieve function flags
        arguments:      ea - any address belonging to the function
        returns:        -
- function doesn't exist
                        otherwise returns the flags:
#define FUNC
NORET      0x00000001L
 function doesn't return
#define FUNC
FAR        0x00000002L
 far function
#define FUNC
LIB        0x00000004L
 library function
#define FUNC
```

```
STATIC     0x00000008L
 static function

long GetFunctionFlags(long ea
 change function flags
        arguments:      ea - any address belonging to the function
                        flags - see GetFunctionFlags
 for explanations
        returns:        !
- ok

success SetFunctionFlags(long ea,long flags
 retrieve function name
        arguments:      ea - any address belonging to the function
        returns:        null string - function doesn't exist
                        otherwise returns function name

char GetFunctionName(long ea
 ask the user to select a function
        arguments:      title - title of the dialog box
        returns:        -
- user refused to select a function
                        otherwise returns the selected function start address

long ChooseFunction(char title
 convert address to 'funcname+offset' string
        arguments:      ea - address to convert
        returns:        if the address belongs to a function then
                          return a string formed as 'name+offset'
                          where 'name' is a function name
                          'offset' is offset within the function
                        else
                          return null string

char GetFuncOffset(long ea
 Determine a new function boundaries


        arguments:      ea  - starting address of a new function
        returns:        if a function already exists, then return
                        its end address.
                        if a function end cannot be determined,
                        the return BADADDR
                        otherwise return the end address of the new function

long FindFuncEnd(long ea
 Get ID of function frame structure


        arguments:      ea - any address belonging to the function
        returns:        ID of function frame or -1
                        In order to access stack variables you need to use
                        structure member manipulaion functions with the
                        obtained ID.
                        If the function does't exist, return -1

long GetFrame(long ea
 Get size of local variables in function frame


        arguments:      ea - any address belonging to the function
        returns:        Size of local variables in bytes.
                        If the function doesn't have a frame, return 0
                        If the function does't exist, return -1

long GetFrameLvarSize(long ea
 Get size of saved registers in function frame


        arguments:      ea - any address belonging to the function
        returns:        Size of saved registers in bytes.
                        If the function doesn't have a frame, return 0
                        This value is used as offset for BP
                        (if FUNC
FRAME is set)
                        If the function does't exist, return -1

long GetFrameRegsSize(long ea
 Get size of arguments in function frame


        arguments:      ea - any address belonging to the function
        returns:        Size of function arguments in bytes.
                        If the function doesn't have a frame, return 0
                        If the function does't exist, return -1

long GetFrameArgsSize(long ea
 Get full size of function frame


        arguments:      ea - any address belonging to the function
        returns:        Size of function frame in bytes.
                        This function takes into account size of local
                        variables + size of saved registers + size of
                        return address + size of function arguments
                        If the function doesn't have a frame, return size of
                        function return address in the stack.
                        If the function does't exist, return 0

long GetFrameSize(long ea
 Make function frame


        arguments:      ea      - any address belonging to the function
                        lvsize  - size of function local variables
                        frregs  - size of saved registers
                        argsize - size of function arguments
        returns:        ID of function frame or -1
                        If the function did not have a frame, the frame
                        will be created. Otherwise the frame will be
                        modified

long MakeFrame(long ea,long lvsize,long frregs,long argsize
 Get current delta for the stack pointer
```

```
        arguments:      ea      - address of the instruction
        returns:        The difference between the original SP upon
                        entering the function and SP for
                        the specified address

long GetSpd(long ea
 Get modification of SP made by the current instruction


        arguments:      ea      - address of the instruction
        returns:        Get modification of SP made at the specified location
                        If the specified location doesn't contain a SP
                        change point, return 0
                        Otherwise return delta of SP modification

long GetSpDiff(long ea
 Setup modification of SP made by the current instruction


        arguments:      ea      - address of the instruction
                        delta   - the difference made by the current
                                  instruction.
        returns:        1-
-failed

success SetSpDiff(long ea,long delta
 retrieve number of entry points
        arguments:      none
        returns:        number of entry points

long GetEntryPointQty(void
 add entry point
        arguments:      ordinal  - entry point number
                                   if entry point doesn't have an ordinal
                                   number,
'ordinal' should be equal to '
'
                        ea       - address of the entry point
                        name     - name of the entry point. If null string,
                                   the entry point won't be renamed.
                        makecode - if 1 then this entry point is a start
                                   of a function. Otherwise it denotes data
                                   bytes.
        returns:        0 - entry point with the specifed ordinal already
                                exists
                        1 - ok

success AddEntryPoint(long ordinal,long ea,char name,long makecode
 retrieve entry point ordinal number
        arguments:      index -
GetEntryPointQty
1
        returns:        0 if entry point doesn't exist
                        otherwise entry point ordinal
```

```
long GetEntryOrdinal(long index
 retrieve entry point address
        arguments:      ordinal - entry point number
                                  it is returned by GetEntryPointOrdinal

        returns:        -1 if entry point doesn't exist
                        otherwise entry point address.
                        If entry point address is equal to its ordinal
                        number, then the entry point has no ordinal.

long GetEntryPoint(long ordinal
 rename entry point
        arguments:      ordinal - entry point number
                        name    - new name
        returns:        !
- ok

success RenameEntryPoint(long ordinal,char name
 find next address with fixup information
        arguments:      ea - current address
        returns:        -
- no more fixups
                        otherwise returns the next address with
                                                fixup information

long GetNextFixupEA(long ea
 find previous address with fixup information
        arguments:      ea - current address
        returns:        -
- no more fixups
                        otherwise returns the previous address with
                                                fixup information

long GetPrevFixupEA(long ea
 get fixup target type
        arguments:      ea - address to get information about
        returns:        -
- no fixup at the specified address
                        otherwise returns fixup target type:
#define FIXUP
MASK      0x7
 mask for fixup types:
#define FIXUP
BYTE      0
   Low-order byte (
-bit displacement or

   low byte of 16-bit offset
#define FIXUP
OFF16     1
   16-bit offset.
#define FIXUP
SEG16     2
   16-bit base
logical segment base

(selector
#define FIXUP
PTR32     3
   32-bit long pointer (
-bit base:
-bit

   offset
#define FIXUP
OFF32     4
   32-bit offset.
#define FIXUP
PTR48     5
   48-bit pointer (
-bit base:
-bit offset)
#define FIXUP
HI8       6
   high 8 bits of 16bit offset
#define FIXUP
REL       0x08
 fixup is relative to linear address

 specified in
#define FIXUP
SELFREL   0x10
 self-relative?
#define FIXUP
EXTDEF    0x20
 target is a location (otherwise - segment)
#define FIXUP
UNUSED    0x40
 fixup is ignored by IDA

long GetFixupTgtType(long ea
 get fixup target selector
        arguments:      ea - address to get information about
        returns:        -
- no fixup at the specified address
                        otherwise returns fixup target selector

long GetFixupTgtSel(long ea
 get fixup target offset
        arguments:      ea - address to get information about
        returns:        -
- no fixup at the specified address
                        otherwise returns fixup target offset

long GetFixupTgtOff(long ea
 get fixup target displacement
        arguments:      ea - address to get information about
        returns:        -
- no fixup at the specified address
                        otherwise returns fixup target displacement

long GetFixupTgtDispl(long ea
 set fixup information
        arguments:      ea        - address to set fixup information about
```

```
                        type      - fixup type. see GetFixupTgtType

                                    for possible fixup types.
                        targetsel - target selector
                        targetoff - target offset
                        displ     - displacement
        returns:        none

void SetFixup(long ea,long type,long targetsel,long targetoff,long displ
 delete fixup information
        arguments:      ea - address to delete fixup information about
        returns:        none

void DelFixup(long ea
 mark position
        arguments:      ea      - address to mark
                        lnnum   - number of generated line for the '
'
                        x       - x coordinate of cursor
                        y       - y coordinate of cursor
                        slot    - slot number:
20
                                  if the specifed value is not within the
                                  range, IDA will ask the user to select
slot.
                        comment - description of the mark.
                                  Should be not empty.
        returns:        none

void MarkPosition(long ea,long lnnum,long x,long y,long slot,char comment
 get marked position
        arguments:      slot    - slot number:
20
                                  if the specifed value is <
 0
                                  range, IDA will ask the user to select
slot.
        returns:        -
- the slot doesn't contain a marked address
                        otherwise returns the marked address

long GetMarkedPos(long slot
 get marked position comment
        arguments:      slot    - slot number:
20
        returns:        null string if the slot doesn't contain
                                        a marked address
                        otherwise returns the marked address comment

char GetMarkComment(long slot
 get number of defined structure types
        arguments:      none
        returns:        number of structure types

long GetStrucQty(void
 get index of first structure type
        arguments:      none
        returns:        -1 if no structure type is defined
                        index of first structure type.
                        Each structure type has an index and ID.
                        INDEX determines position of structure definition
                         in the list of structure definitions. Index 1
                         is listed first, after index 2 and so on.
                         The index of a structure type can be changed any
                         time, leading to movement of the structure
definition
                         in the list of structure definitions.
                        ID uniquely denotes a structure type. A structure
                         gets a unique ID at the creation time and this ID
                         can't be changed. Even when the structure type gets
                         deleted, its ID won't be resued in the future.

long GetFirstStrucIdx(void
 get index of last structure type
        arguments:      none
        returns:        -1 if no structure type is defined
                        index of last structure type.
                        See GetFirstStrucIdx
 for the explanation of
                        structure indices and IDs.

long GetLastStrucIdx(void
 get index of next structure type
        arguments:      current structure index
        returns:        -1 if no (more) structure type is defined
                        index of the next structure type.
                        See GetFirstStrucIdx
 for the explanation of
                        structure indices and IDs.

long GetNextStrucIdx(long index
 get index of previous structure type
        arguments:      current structure index
        returns:        -1 if no (more) structure type is defined
                        index of the presiouvs structure type.
                        See GetFirstStrucIdx
 for the explanation of
                        structure indices and IDs.

long GetPrevStrucIdx(long index
 get structure index by structure ID
        arguments:      structure ID
        returns:        -1 if bad structure ID is passed
                        otherwise returns structure index.
                        See GetFirstStrucIdx
 for the explanation of
                        structure indices and IDs.

long GetStrucIdx(long id
 get structure ID by structure index
        arguments:      structure index
        returns:        -1 if bad structure index is passed
                        otherwise returns structure ID.
                        See GetFirstStrucIdx
 for the explanation of
                        structure indices and IDs.

long GetStrucId(long index
 get structure ID by structure name
        arguments:      structure type name
        returns:        -1 if bad structure type name is passed
                        otherwise returns structure ID.

long GetStrucIdByName(char name
 get structure type name
        arguments:      structure type ID
        returns:        -1 if bad structure type ID is passed
                        otherwise returns structure type name.

char GetStrucName(long id
 get structure type comment
        arguments:      id         - structure type ID
                        repeatable -
: get repeatable comment
                                     0: get regular comment
        returns:        null string if bad structure type ID is passed
                        otherwise returns comment.

char GetStrucComment(long id,long repeatable
 get size of a structure
        arguments:      id         - structure type ID
        returns:        -1 if bad structure type ID is passed
                        otherwise returns size of structure in bytes.

long GetStrucSize(long id
 get number of members of a structure
        arguments:      id         - structure type ID
        returns:        -1 if bad structure type ID is passed
                        otherwise returns number of members.

long GetMemberQty(long id
 get previous offset in a structure
        arguments:      id     - structure type ID
                        offset - current offset
        returns:        -1 if bad structure type ID is passed
                           or no (more) offsets in the structure
                        otherwise returns previous offset in a structure.
                        NOTE: IDA allows 'holes' between members of a
                              structure. It treats these 'holes'
                              as unnamed arrays of bytes.
                        This function returns a member offset or a hole
offset.
                        It will return size of the structure if input
                        'offset' is bigger than the structure size.

long GetStrucPrevOff(long id,long offset
 get next offset in a structure
        arguments:      id     - structure type ID
                        offset - current offset
        returns:        -1 if bad structure type ID is passed
                           or no (more) offsets in the structure
                        otherwise returns next offset in a structure.
                        NOTE: IDA allows 'holes' between members of a
                              structure. It treats these 'holes'
                              as unnamed arrays of bytes.
                        This function returns a member offset or a hole
offset.
                        It will return size of the structure if input
                        'offset' belongs to the last member of the structure.

long GetStrucNextOff(long id,long offset
 get offset of the first member of a structure
        arguments:      id            - structure type ID
        returns:        -1 if bad structure type ID is passed
                           or structure has no members
                        otherwise returns offset of the first member.
                        NOTE: IDA allows 'holes' between members of a
                              structure. It treats these 'holes'
                              as unnamed arrays of bytes.

long GetFirstMember(long id
 get offset of the last member of a structure
        arguments:      id            - structure type ID
        returns:        -1 if bad structure type ID is passed
                           or structure has no members
                        otherwise returns offset of the last member.
                        NOTE: IDA allows 'holes' between members of a
                              structure. It treats these 'holes'
                              as unnamed arrays of bytes.

long GetLastMember(long id
 get offset of a member of a structure by the member name
        arguments:      id            - structure type ID
                        member
name   - name of structure member
        returns:        -1 if bad structure type ID is passed
                           or no such member in the structure
                        otherwise returns offset of the specified member.

long GetMemberOffset(long id,char member
name
 get name of a member of a structure
        arguments:      id            - structure type ID
                        member
offset - member offset. The offset can be
                                        any offset in the member. For
example,
                                        is a member is 4 bytes long and
starts
                                        at offset 2, then 2,
,5 denote
                                        the same structure member.
        returns:        -1 if bad structure type ID is passed
                           or no such member in the structure
                        otherwise returns name of the specified member.

char GetMemberName(long id,long member
offset
```

```
 get comment of a member
        arguments:      id            - structure type ID
                        member
offset - member offset. The offset can be
                                        any offset in the member. For
example,
                                        is a member is 4 bytes long and
starts
                                        at offset 2, then 2,
,5 denote
                                        the same structure member.
                        repeatable -
: get repeatable comment
                                     0: get regular comment
        returns:        null string if bad structure type ID is passed
                           or no such member in the structure
                        otherwise returns comment of the specified member.

char GetMemberComment(long id,long member
offset,long repeatable
 get size of a member
        arguments:      id            - structure type ID
                        member
offset - member offset. The offset can be
                                        any offset in the member. For
example,
                                        is a member is 4 bytes long and
starts
                                        at offset 2, then 2,
,5 denote
                                        the same structure member.
        returns:        -1 if bad structure type ID is passed
                           or no such member in the structure
                        otherwise returns size of the specified
                                          member in bytes.

long GetMemberSize(long id,long member
offset
 get type of a member
        arguments:      id            - structure type ID
                        member
offset - member offset. The offset can be
                                        any offset in the member. For
example,
                                        is a member is 4 bytes long and
starts
                                        at offset 2, then 2,
,5 denote
                                        the same structure member.
        returns:        -1 if bad structure type ID is passed
                           or no such member in the structure
                        otherwise returns type of the member, see bit
                        definitions above. If the member type is a structure
                        then function GetMemberStrid
 should be used to
                        get the structure type id.
```

```
long GetMemberFlag(long id,long member
offset
 get structure id of a member
        arguments:      id            - structure type ID
                        member
offset - member offset. The offset can be
                                        any offset in the member. For
example,
                                        is a member is 4 bytes long and
starts
                                        at offset 2, then 2,
,5 denote
                                        the same structure member.
        returns:        -1 if bad structure type ID is passed
                           or no such member in the structure
                        otherwise returns structure id of the member.
                        If the current member is not a structure, returns -
.

long GetMemberStrId(long id,long member
offset
 define a new structure type
        arguments:      index         - index of new structure type
                        If another structure has the specified index,
                        then index of that structure and all other
                        structures will be increentedfreeing the specifed
                        index. If index is
, then the biggest index
                        number will be used.
                        See GetFirstStrucIdx
 for the explanation of
                        structure indices and IDs.

                        name - name of the new structure type.

        returns:        -1 if can't define structure type because of
                        bad structure name: the name is ill-formed or is
                        already used in the program.
                        otherwise returns ID of the new structure type

long AddStruc(long index,char name
 delete a structure type
        arguments:      id            - structure type ID
        returns:        0 if bad structure type ID is passed
                        1 otherwise the structure type is deleted. All data
                        and other structure types referencing to the
                        deleted structure type will be displayed as array
                        of bytes.

success DelStruc(long id
 change structure index
        arguments:      id      - structure type ID
                        index   - new index of the structure
                        See GetFirstStrucIdx
 for the explanation of
                        structure indices and IDs.
        returns:        !
```

```
- ok

long SetStrucIdx(long id,long index
 change structure name
        arguments:      id      - structure type ID
                        name    - new name of the structure
        returns:        !
- ok

long SetStrucName(long id,char name
 change structure comment
        arguments:      id      - structure type ID
                        comment - new comment of the structure
                        repeatable -
: change repeatable comment
                                     0: change regular comment
        returns:        !
- ok

long SetStrucComment(long id,char comment,long repeatable
 add structure member
        arguments:      id      - structure type ID
                        name    - name of the new member
                        offset  - offset of the new member
                        flag    - type of the new member. Should be one of
                                  FF
BYTE
PACKREAL (see above)
                                  combined with FF
DATA
                        typeid  - structure id if 'flag'
STRU
                                  Denotes type of the member is the member
                                  itself is a structure. Otherwise should be
                                  -
.
                                  if isOff0(flag) then typeid specifies
                                  the offset base.
                                  if isASCII(flag) then typeid specifies
                                  the string type (ASCSTR

                        nitems  - number of items in the new member
        returns:        0 -
, otherwise error code:
#define STRUC
ERROR
MEMBER
NAME   1
 already have member with this name (bad name)
#define STRUC
ERROR
MEMBER
OFFSET 2
 already have member at this offset
#define STRUC
ERROR
MEMBER
```

```
SIZE   3
 bad number of items or bad sizeof(type)

long AddStrucMember(long id,char name,long offset,long flag,
                    long typeid,long nitems
 delete structure member
        arguments:      id            - structure type ID
                        member
offset - offset of the member
        returns:        !
.
                        NOTE: IDA allows 'holes' between members of a
                              structure. It treats these 'holes'
                              as unnamed arrays of bytes.

long DelStrucMember(long id,long member
offset
 change structure member name
        arguments:      id            - structure type ID
                        member
offset - offset of the member
                        name          - new name of the member
        returns:        !
.

long SetMemberName(long id,long member
offset,char name
 change structure member type
        arguments:      id            - structure type ID
                        member
offset - offset of the member
                        flag    - new type of the member. Should be one of
                                  FF
BYTE
PACKREAL (see above)
                                  combined with FF
DATA
                        typeid  - structure id if 'flag'
STRU
                                  Denotes type of the member is the member
                                  itself is a structure. Otherwise should be
                                  -
.
                                  if isOff0(flag) then typeid specifies
                                  the offset base.
                                  if isASCII(flag) then typeid specifies
                                  the string type (ASCSTR

                        nitems  - number of items in the member
        returns:        !
.

long SetMemberType(long id,long member
offset,long flag,long typeid,long nitems
 change structure member comment
        arguments:      id      - structure type ID
                        member
```

```
offset - offset of the member
                        comment - new comment of the structure member
                        repeatable -
: change repeatable comment
                                     0: change regular comment
        returns:        !
- ok

long SetMemberComment(long id,long member
offset,char comment,long repeatable
 get number of enum types
        arguments:      none
        returns:        number of enumerations

long GetEnumQty(void
 get ID of the specified enum by its serial number
        arguments:      idx - number of enum (
GetEnumQty
)
        returns:        ID of enum or -1 if error

long GetnEnum(long idx
 get serial number of enum by its ID
        arguments:      enum
- ID of enum
        returns:        (
GetEnumQty
) or -1 if error

long GetEnumIdx(long enum
 get enum ID by the name of enum
        arguments:      name - name of enum
        returns:        ID of enum or -1 if no such enum exists

long GetEnum(char name
 get name of enum
        arguments:      enum
- ID of enum
        returns:        name of enum or empty string

char GetEnumName(long enum
 get comment of enum
        arguments:      enum
- ID of enum
                        repeatable -
:get regular comment
                                     1:get repeatable comment
        returns:        comment of enum

char GetEnumCmt(long enum
,long repeatable
 get size of enum
        arguments:      enum
- ID of enum
        returns:        number of constants in the enum
                        Returns 0 if enum
id is bad.
```

```
long GetEnumSize(long enum
 get flag of enum
        arguments:      enum
- ID of enum
        returns:        flags of enum. These flags determine representation
                        of numeric constants (binary,octal,decimal,
)
                        in the enum definition. See start of this file for
                        more information about flags.
                        Returns 0 if enum
id is bad.

long GetEnumFlag(long enum
 get member of enum - a symbolic constant ID
        arguments:      name - name of symbolic constant
        returns:        ID of constant or -1

long GetConstByName(char name
 get value of symbolic constant
        arguments:      const
- id of symbolic constant
        returns:        value of constant or 0

long GetConstValue(long const
 get id of enum by id of constant
        arguments:      const
- id of symbolic constant
        returns:        id of enum the constant belongs to.
                        -1 if const
id is bad.

long GetConstEnum(long const
 get id of constant
        arguments:      enum
- id of enum
                        value   - value of constant
        returns:        id of constant or -1 if error

long GetConst(long enum
,long value
 get first constant in the enum
        arguments:      enum
- id of enum
        returns:        value of constant or -1 no constants are defined
                        All constants are sorted by their values
                        as unsigned longs.

long GetFirstConst(long enum
 get last constant in the enum
        arguments:      enum
- id of enum
        returns:        value of constant or -1 no constants are defined
                        All constants are sorted by their values
                        as unsigned longs.

long GetLastConst(long enum
 get next constant in the enum
        arguments:      enum
- id of enum
                        value   - value of the current constant
        returns:        value of a constant with value higher than the
specified
                        value.
-1 no such constants exist.
                        All constants are sorted by their values
                        as unsigned longs.

long GetNextConst(long enum
,long value
 get prev constant in the enum
        arguments:      enum
- id of enum
                        value   - value of the current constant
        returns:        value of a constant with value lower than the
specified
                        value.
-1 no such constants exist.
                        All constants are sorted by their values
                        as unsigned longs.

long GetPrevConst(long enum
,long value
 get name of a constant
        arguments:      const
- id of const
        returns:        name of constant

char GetConstName(long const
 get comment of a constant
        arguments:      const
- id of const
                        repeatable -
:get regular comment
                                     1:get repeatable comment
        returns:        comment string

char GetConstCmt(long const
,long repeatable
 add a new enum type
        arguments:      idx - serial number of the new enum.
                              If another enum with the same serial number
                              exists, then all enums with serial
                              numbers >
 the specified idx get their
                              serial numbers incremented (in other words,
                              the new enum is put in the middle of the list
                              of enums

                              If idx >
 GetEnumQty
 then the new enum is
                              created at the end of the list of enums.
                        name - name of the enum.
                        flag - flags for representation of numeric constants
                               in the definition of enum.
        returns:        id of new enum or -
.

long AddEnum(long idx,char name,long flag
 delete enum type
        arguments:      enum
- id of enum

void DelEnum(long enum
 give another serial number to a enum
        arguments:      enum
- id of enum
                        idx     - new serial number.
                              If another enum with the same serial number
                              exists, then all enums with serial
                              numbers >
 the specified idx get their
                              serial numbers incremented (in other words,
                              the new enum is put in the middle of the list
                              of enums

                              If idx >
 GetEnumQty
 then the enum is
                              moved to the end of the list of enums.
        returns:        comment string

success SetEnumIdx(long enum
,long idx
 rename enum
        arguments:      enum
- id of enum
                        name    - new name of enum
        returns:        1-
-failed

success SetEnumName(long enum
,char name
 set comment of enum
        arguments:      enum
- id of enum
                        cmt     - new comment for the enum
                        repeatable -
:set regular comment
                                     1:set repeatable comment
        returns:        1-
-failed

success SetEnumCmt(long enum
,char cmt,long repeatable
 set flag of enum
        arguments:      enum
- id of enum
                        flag - flags for representation of numeric constants
                               in the definition of enum.
        returns:        1-
-failed

success SetEnumFlag(long enum
,long flag
 add a member of enum - a symbolic constant
        arguments:      enum
- id of enum
                        name    - name of symbolic constant. Must be unique
                                  in the program.
                        value   - value of symbolic constant.
        returns:        0-
, otherwise error code:
#define CONST
ERROR
NAME  1
 already have member with this name (bad name)
#define CONST
ERROR
VALUE 2
 already have member with this value
#define CONST
ERROR
ENUM  3
 bad enum id

long AddConst(long enum
,char name,long value
 delete a member of enum - a symbolic constant
        arguments:      enum
- id of enum
                        value   - value of symbolic constant.
        returns:        1-
-failed

success DelConst(long enum
,long value
 rename a member of enum - a symbolic constant
        arguments:      const
- id of const
                        name     - new name of constant
        returns:        1-
-failed

success SetConstName(long const
,char name
 set a comment of a symbolic constant
        arguments:      const
- id of const
                        cmt     - new comment for the constant
                        repeatable -
:set regular comment
                                     1:set repeatable comment
        returns:        1-
-failed

success SetConstCmt(long const
,char cmt,long repeatable

The arrays are virtual. IDA allocates space for and keeps only the specified
elements of an array. Array index is 32-bit long. Actually, each array
may keep a set of strings and a set of long(32bit) values.
 create array
        arguments:      name - name of array. There are no restrictions
                               on the name (its length should be less than
                               120 characters, though)
        returns:        -
- can't create array (it already exists)
                        otherwise returns id of the array

long CreateArray(char name
 get array id by its name
        arguments:      name - name of existing array.
        returns:        -
- no such array
                        otherwise returns id of the array

long GetArrayId(char name
 rename array
        arguments:      id      - array id returned by CreateArray
 or
                                  GetArrayId

                        newname - new name of array. There are no
                                  restrictions on the name (its length should
                                  be less than 120 characters, though)
        returns:        1-
-failed

success RenameArray(long id,char newname
 delete array
   This function deletes all elements of the array.
        arguments:      id      - array id

void DeleteArray(long id
 set 32bit value of array element
        arguments:      id      - array id
                        idx     - index of an element
                        value   - 32bit value to store in the array
        returns:        1-
-failed

success SetArrayLong(long id,long idx,long value
 set string value of array element
        arguments:      id      - array id
                        idx     - index of an element
                        str     - string to store in array element
        returns:        1-
-failed

success SetArrayString(long id,long idx,char str
 get value of array element
        arguments:      tag     - tag of array, specifies one of two
                                  array types:
```

```
#define AR
LONG '
'
 array of longs
#define AR
STR  '
'
 array of strings
                        id      - array id
                        idx     - index of an element
        returns:        value of the specified array element.
                        note that this function may return char or long
                        result. Unexistent array elements give zero as
                        a result.

char or long GetArrayElement(long tag,long id,long idx
 delete an array element
        arguments:      tag     - tag of array (
LONG or AR
)
                        id      - array id
                        idx     - index of an element
        returns:        1-
-failed

success DelArrayElement(long tag,long id,long idx
Decision to convert to instruction/data is made by IDA
 get index of the first existing array element
        arguments:      tag     - tag of array (
LONG or AR
)
                        id      - array id
        returns:        -
- array is empty
                        otherwise returns index of the first array element

long GetFirstIndex(long tag,long id
Building list of the signatures.
WRAP
Edit a enum member


This command allows you to rename enum member. A enum member
is a symbolic constant. Its name should be unique in the program.

See also
:1222[Edit|Enums] submenu.
Actions %s and %s have the same hotkey %
Too many actions are defined!

This type of output files is not supported.
WRAP
Delete a enum member


Please remember that deleting a member deletes also all information
about the member, including comments, member name etc.
```

See also
:1222[Edit|Enums] submenu.
Kernel analysis options

Create offsets and segments using fixup info

        IDA will use relocation information to make the disassembly
        nicer. In particular, it will convert all data items with
        relocation information to words or dwords like this:

```
                dd offset label
                dw seg seg000
```

        If an instruction has a relocation information attached to it,
        IDA will convert its immediate operand to an offset or segment:

```
                mov     eax, offset label
```

        You can display the relocation information attached to the current
        item by using show
[internal] flags command.

Mark typical code sequences as code

        IDA knows some typical code sequences for each processor.
        For example, it knows about typical sequence

```
                push    bp
                mov     bp, sp
```

        If this option is enabled, IDA will search for all typical sequences
        and convert them to instructions even if there are no references
        to them. The search is performed at the loading time.

Delete instructions with no xrefs

        This option allows IDA to undefine unreferences instructions.
        For example, if you
[undefine] an instruction at the start of a
        function, IDA will trace execution flow and delete all instructions
        that lose references to them.

Trace execution flow

        This options allows IDA to trace execution flow and convert all
        references bytes to
[instructions

Create functions if call is present

        This options allows IDA to create
[function]
(proc) if a call
        instruction is present. For example, the presence of:

```
                call loc
1234

        leads to creation of a function at label loc
1234
```

Analyse and create all xrefs

        Without this option IDA will not thoroughly analyse the program.
        If this option is disabled, IDA will simply trace execution flow,
        nothing more (no xrefs, no additional checks, etc)

Use flirt signatures

        Allows usage of FLIRT technology

Create function if data xref data
code32 exists

        If IDA encounters a data references from DATA segment to 32bit
        CODE segment, it will check for the presence of meaningful
        (disassemblable) instruction at the target. If there is an
        instruction, it will mark is as an instruction and will create
        a function there.

Rename jump functions as j

        This option allows IDA to rename simple functions containing only

```
                jmp somewhere
```

        instruction to
somewhere
.

Rename empty functions as nullsub

        This option allows IDA to rename empty functions containing only
        a
return
 instruction as
nullsub

 is replaced by a serial number:

Create stack variables

        This option allows IDA to automatically create stack variables and
        function parameteres.

Trace stack pointer

```
        This option allows IDA to
[trace] value of SP register.

Create ascii string if data xref exists

        If IDA encounters a data reference to an undefined item, it
        checks for the presence of ASCII string at the target. If the length
        of ASCII string is big enough (more than 4 chars in 16bit or data
        segments; more than 16 chars otherwise
 IDA will automatically create
        an
[ASCII] string.

Convert 32bit instruction operand to offset

        This option works only in 32bit segments.
        If an instruction has an immediate operand and the operand
        can be represented as a meaningful offset expression, IDA will
        convert it to an offset. However, the value of immediate operand
        must be higher than 0x10000.

Create offset if data xref to seg32 exists

        If IDA encounters a data reference to 32bit segment and the target
        contains 32bit value which can be represented as an offset
expression,
        IDA will convert it to an offset

Make final analysis pass

        This option allows IDA to coagulate all
[unexplored] bytes
        by converting them to data or instructions.

Locate and create jump tables

        This option allows IDA to try to guess address and size of
[jump]
        tables. Please note that disabling this option will not disable
        the recognition of C-style typical switch constructs.

Coagulate data in the final pass

        This option is meaningful only if
Make final analysis pass

        is enabled. It allows IDA to convert
[unexplored] bytes
        to data arrays in the data segments. If this option is disabled,
        IDA will coagulate only code segments.
Building list of the modules.
The output database %s already exists. Do you want to overwrite it?
 get index of the last existing array element
        arguments:      tag     - tag of array (
LONG or AR
)
                        id      - array id
        returns:        -
- array is empty
                        otherwise returns index of the last array element

long GetLastIndex(long tag,long id
 get index of the next existing array element
        arguments:      tag     - tag of array (
LONG or AR
)
                        id      - array id
                        idx     - index of the current element
        returns:        -
- no more array elements
                        otherwise returns index of the next array element

long GetNextIndex(long tag,long id,long idx
 get index of the previous existing array element
        arguments:      tag     - tag of array (
LONG or AR
)
                        id      - array id
                        idx     - index of the current element
        returns:        -
- no more array elements
                        otherwise returns index of the previous array element

long GetPrevIndex(long tag,long id,long idx

  IDA can keep information about source files used to create the program.
  Each source file is represented by a range of addresses.
  A source file may contains several address ranges.
 Mark a range of address as belonging to a source file
   An address range may belong only to one source file.
   A source file may be represented by several address ranges.
        ea1     - linear address of start of the address range
        ea2     - linear address of end of the address range
        filename- name of source file.
   returns:
-failed.

success AddSourceFile(long ea1,ulong ea2,char filename
 Get name of source file occupying the given address
        ea - linear address
   returns: NULL - source file information is not found
            otherwise returns pointer to file name

char GetSourceFile(long ea
 Delete information about the source file
        ea - linear address belonging to the source file
   returns: NULL - source file information is not found
            otherwise returns pointer to file name

success DelSourceFile(long ea
 set source line number
        arguments:      ea      - linear address
                        lnnum   - number of line in the source file
        returns:        nothing
```

```
void SetLineNumber(long ea,long lnnum
 get source line number
        arguments:      ea      - linear address
        returns:        number of line in the source file or -1

long GetLineNumber(long ea
 delete information about source line number
        arguments:      ea      - linear address
        returns:        nothing

void DelLineNumber(long ea

        Convenience function.
        See
:1575[
 functions for bit definitions.

StringStp(
)            SetCharPrm(
ASCII
BREAK,
)
LowVoids(
)             SetLongPrm(
)
HighVoids(
)            SetLongPrm(
HIGH
)
TailDepth(
)            SetLongPrm(
MAXREF,
)
Analysis(
)             SetCharPrm(
AUTO,
)
Tabs(
)                 SetCharPrm(
ENTAB,
)
Comments(
)             SetCharPrm(
CMTFLAG
)
                                ?
ALLCMT|GetCharPrm(
CMTFLAG

                                :
ALLCMT
GetCharPrm(
CMTFLAG

Voids(
)                SetCharPrm(
VOIDS,
)
Indent(
)               SetCharPrm(
INDENT,
)
CmtIndent(
)            SetCharPrm(
COMMENT,
)
AutoShow(
)             SetCharPrm(
AUTOSHOW,
)
MinEA
                 GetLongPrm(
)
MaxEA
                 GetLongPrm(
)
BeginEA
               GetLongPrm(
BEGIN
)
set
start
)         SetLongPrm(
START
)
set
start
)         SetLongPrm(
START
Error packing database
:1745
```

This file is self-loading, IDA can't disassemble it.
WRAP
Self-loading NE files

There are so-called self-loading 16bit programs in MS Windows.
These programs load theirselves into the memory. IDA can't disassemble them because the method of loading is unknown - each program may have its own method of loading.
The color codes

You can change the colors in menu Options|Colors, Customize button.
You may have up to 4 different color palettes and switch between them on fly. The color palette is saved in IDACOLOR.CF file.

Each line prefix has its own color code depending on the current item:
(the fourth color palette values are shown)

```
Line prefixes
   Library function             BRIGHT CYAN ON BLUE
```

Regular function	WHITE ON BLUE
Instruction	BROWN ON BLUE
Data	WHITE ON BLUE
Unexplored	WHITE ON BLACK
Externs	BRIGHT MAGENTA ON BLUE
Current item	BRIGHT BLUE ON BLUE
Current line	YELLOW ON BLUE
Default	BLACK ON BLACK (not used)

If the current item or current line are BLACK ON BLACK, then they will not be highlighted.

The rest of the line is colored with the following codes:

Keywords

Instruction	WHITE ON BLUE
Directive	YELLOW ON BLUE
Macro name	MAGENTA ON BLUE
Register name	WHITE ON BLUE
Other	WHITE ON BLUE

Names

Dummy data	WHITE ON BLUE
Dummy code	WHITE ON BLUE
Dummy unexplored	MAGENTA ON BLUE
Hidden	GREY ON BLUE
Library function	BRIGHT CYAN ON BLUE
Local variable	GREEN ON BLUE
Regular data	YELLOW ON BLUE
Regular code	YELLOW ON BLUE
Regular unexplored	RED ON BLUE
Demangled	BRIGHT GREEN ON BRIGHT BLUE
Segment name	YELLOW ON BLUE
Imported name	LIGHT MAGENTA ON BLUE

Constants

Suspicious (void)	BRIGHT RED ON BLUE
Char in instruction	BRIGHT CYAN ON BLUE
String in instruction	BRIGHT CYAN ON BLUE
Number in instruction	BRIGHT GREEN ON BLUE
Char in data	BRIGHT GREEN ON BLUE
String in data	BRIGHT GREEN ON BLUE
Number in data	WHITE ON BLUE

Xrefs

Code	GREEN ON BLUE
Data	CYAN ON BLUE
Code to tail	BRIGHT RED ON BLUE
Data to tail	MAGENTA ON BLUE

Comments

Automatic	BROWN ON BLUE
Regular	BRIGHT WHITE ON BLUE

Repeatable	BROWN ON BLUE
Extra line	YELLOW ON BLUE
Other	
Punctuation	WHITE ON BLUE
Opcode bytes	BRIGHT GREEN ON BLUE
Manual operand	BRIGHT WHITE ON BLUE
Errors	RED ON BLACK
Selected	BLACK ON WHITE
Default	YELLOW ON BLUE

Other Default color code is used if a token has no color attached to it.