# Module idc

IDC compatibility module

This file contains IDA built-in function declarations and internal bit definitions. Each byte of the program has 32-bit flags (low 8 bits keep the byte value). These 32 bits are used in get_full_flags/get_flags functions.

This file is subject to change without any notice. Future versions of IDA may use other definitions.

| Classes | |
|---|---|
| | DeprecatedIDCError<br>Exception for deprecated function calls |

| Functions | |
|---|---|
| | set_inf_attr(offset, value) |
| | get_name_ea_simple(name)<br>Get linear address of a name |
| | get_event_module_size()<br>Get module size for debug event |
| | writelong(handle, dword, mostfirst) |
| | batch(batch)<br>Enable/disable batch mode of operation |
| | get_sp_delta(ea)<br>Get modification of SP made by the instruction |
| | set_segm_attr(segea, attr, value)<br>Set segment attribute |
| | del_hash_string(hash_id, key)<br>Delete a hash element. |
| | get_frame_id(ea)<br>Get ID of function frame structure |
| | create_word(ea)<br>Convert the current item to a word (2 bytes) |
| | set_bpt_attr(address, bptattr, value)<br>modifiable characteristics of a breakpoint |
| | substr(s, x1, x2) |
| | find_unknown(ea, flag) |
| | get_segm_start(ea)<br>Get start address of a segment |
| | isBin1(F) |
| | isBin0(F) |
| | value_is_pvoid(var) |
| | set_array_string(array_id, idx, value)<br>Sets the string value of an array element. |
| | qsleep(milliseconds)<br>qsleep the specified number of milliseconds This function suspends IDA for the specified amount of time |

| | |
|---|---|
| | byte_value(F)<br>Get byte value from flags Get value of byte provided that the byte is initialized. |
| | enable_tracing(trace_level, enable)<br>Enable step tracing |
| | get_sreg(ea, reg)<br>Get value of segment register at the specified address |
| | create_struct(ea, size, strname)<br>Convert the current item to a structure instance |
| | GetLocalType(ordinal, flags)<br>Retrieve a local type declaration |
| | get_bpt_qty()<br>Get number of breakpoints. |
| | get_fixup_target_flags(ea)<br>Get fixup target flags |
| | SaveFile(filepath, pos, ea, size)<br>Save from IDA database to file |
| | get_event_exc_code()<br>Get exception code for EXCEPTION event |
| | delete_array(array_id)<br>Delete array, by its ID. |
| | call_system(command)<br>Execute an OS command. |
| | del_struc_member(sid, member_offset)<br>Delete structure member |
| | is_event_handled()<br>Is the debug event handled? |
| | find_suspop(ea, flag) |
| | rotate_byte(x, count) |
| | create_double(ea)<br>Convert the current item to a double floating point (8 bytes) |
| | set_tail_owner(tailea, funcea)<br>Change the function chunk owner |
| | set_segm_combination(segea, comb)<br>Change combination of the segment |
| | is_union(sid)<br>Is a structure a union? |
| | create_yword(ea)<br>Convert the current item to a ymm word (32 bytes/256 bits) |
| | get_next_offset(sid, offset)<br>Get next offset in a structure |
| | set_func_flags(ea, flags)<br>Change function flags |
| | get_member_cmt(sid, member_offset, repeatable)<br>Get comment of a member |
| | set_bmask_cmt(enum_id, bmask, cmt, repeatable)<br>Set bitmask comment (only for bitfields) |
| | is_data(F) |
| | |

| | |
|---|---|
| | is_struct(F) |
| | get_event_module_name()<br>Get module name for debug event |
| | create_array(name)<br>Create array. |
| | isExtra(F) |
| | find_code(ea, flag) |
| | xtol(s) |
| | get_event_exc_info()<br>Get info for EXCEPTION event |
| | set_member_type(sid, member_offset, flag, typeid, nitems, target=-1, tdelta=0, reftype=2)<br>Change structure member type |
| | force_bl_call(ea)<br>Force BL instruction to be a call |
| | get_prev_offset(sid, offset)<br>Get previous offset in a structure |
| | GetFloat(ea)<br>Get value of a floating point number (4 bytes) This function assumes number stored using IEEE format and in the same endianness as integers. |
| | get_event_module_base()<br>Get module base for debug event |
| | get_fixup_target_sel(ea)<br>Get fixup target selector |
| | is_head(F) |
| | resume_process() |
| | is_float(F) |
| | gen_flow_graph(outfile, title, ea1, ea2, flags)<br>Generate a flow chart GDL file |
| | delete_all_segments()<br>Delete all segments, instructions, comments, i.e. |
| | read_dbg_dword(ea)<br>Get value of program double-word using the debugger memory |
| | get_event_exit_code()<br>Get exit code for debug event |
| | get_next_hash_key(hash_id, key)<br>Get the next key in the hash. |
| | eval_idc(expr)<br>Evaluate an IDC expression |
| | guess_type(ea)<br>Guess type of function/variable |
| | get_hash_string(hash_id, key)<br>Gets the string value of a hash element. |
| | filelength(handle) |
| | find_data(ea, flag) |
| | form(format, *args) |
| | create_byte(ea) |

| | |
|---|---|
| | Convert the current item to a byte |
| | is_defarg1(F) |
| | get_event_pid()<br>Get process ID for debug event |
| | create_tbyte(ea)<br>Convert the current item to a tbyte (10 or 12 bytes) |
| | value_is_string(var) |
| | get_strlit_contents(ea, length=-1, strtype=0)<br>Get string contents |
| | is_tbyte(F) |
| | hasName(F) |
| | get_member_qty(sid)<br>Get number of members of a structure |
| | get_curline()<br>Get the disassembly line at the cursor |
| | get_next_index(tag, array_id, idx)<br>Get index of the next existing array element. |
| | get_tinfo(ea)<br>Get type information of function/variable as 'typeinfo' object |
| | del_struc(sid)<br>Delete a structure type |
| | find_imm(ea, flag, value) |
| | process_ui_action(name, flags=0)<br>Invokes an IDA UI action by name |
| | prev_head(ea, minea=0)<br>Get previous defined item (instruction or data) in the program |
| | get_segm_attr(segea, attr)<br>Get segment attribute |
| | write_dbg_memory(ea, data)<br>Write to debugger memory. |
| | GetDisasm(ea)<br>Get disassembly line |
| | choose_func(title)<br>Ask the user to select a function |
| | is_dword(F) |
| | add_enum_member(enum_id, name, value, bmask)<br>Add a member of enum - a symbolic constant |
| | get_next_module(base)<br>Enumerate process modules |
| | strlen(s) |
| | get_bytes(ea, size, use_dbg=False)<br>Return the specified number of bytes of the program |
| | get_color(ea, what)<br>Get item color |
| | get_func_attr(ea, attr)<br>Get a function attribute |
| | |

| | |
|---|---|
| | define_local_var(start, end, location, name)<br>Create a local variable |
| | is_unknown(F) |
| | get_operand_value(ea, n)<br>Get number used in the operand |
| | get_member_name(sid, member_offset)<br>Get name of a member of a structure |
| | get_member_offset(sid, member_name)<br>Get offset of a member of a structure by the member name |
| | SetPrcsr(processor) |
| | get_event_bpt_hea()<br>Get hardware address for BREAKPOINT event |
| | next_func_chunk(funcea, tailea)<br>Get the next function chunk of the specified function |
| | get_fixup_target_type(ea)<br>Get fixup target type |
| | set_reg_value(value, name)<br>Set register value |
| | set_local_type(ordinal, input, flags)<br>Parse one type declaration and store it in the specified slot |
| | clear_trace(filename)<br>Clear the current trace buffer |
| | set_struc_idx(sid, index)<br>Change structure index |
| | create_qword(ea)<br>Convert the current item to a quadro word (8 bytes) |
| | get_prev_enum_member(enum_id, value, bmask)<br>Get prev constant in the enum |
| | get_member_flag(sid, member_offset)<br>Get type of a member |
| | add_auto_stkpnt(func_ea, ea, delta)<br>Add automatical SP register change point |
| | get_first_member(sid)<br>Get offset of the first member of a structure |
| | set_array_params(ea, flags, litems, align)<br>Set array representation format |
| | find_defined(ea, flag) |
| | is_enum0(F) |
| | is_enum1(F) |
| | move_segm(ea, to, flags)<br>Move a segment to a new address This function moves all information to the new address It fixes up address sensitive information in the kernel The total effect is equal to reloading the segment to the target address |
| | demangle_name(name, disable_mask)<br>demangle_name a name |
| | is_defarg0(F) |
| | get_array_id(name) |

| | |
|---|---|
| | Get array array_id, by name. |
| | del_enum_member(enum_id, value, serial, bmask)<br>Delete a member of enum - a symbolic constant |
| | get_segm_name(ea)<br>Get name of a segment |
| | get_xref_type()<br>Return type of the last xref obtained by [RD]first/next[B0] functions. |
| | get_member_id(sid, member_offset)<br>Returns: -1 if bad structure type ID is passed or there is no member at the specified offset. |
| | value_is_func(var) |
| | read_dbg_byte(ea)<br>Get value of program byte using the debugger memory |
| | apply_type(ea, py_type, flags=1)<br>Apply the specified type to the address |
| | is_seg1(F) |
| | is_seg0(F) |
| | get_idb_path()<br>Get IDB full path |
| | get_array_element(tag, array_id, idx)<br>Get value of array element. |
| | create_oword(ea)<br>Convert the current item to an octa word (16 bytes/128 bits) |
| | value_is_float(var) |
| | set_func_attr(ea, attr, value)<br>Set a function attribute |
| | loadfile(filepath, pos, ea, size) |
| | set_hash_long(hash_id, key, value)<br>Sets the long value of a hash element. |
| | get_segm_end(ea)<br>Get end address of a segment |
| | is_oword(F) |
| | get_reg_value(name)<br>Get register value |
| | set_frame_size(ea, lvsize, frregs, argsize)<br>Make function frame |
| | AddSeg(startea, endea, base, use32, align, comb) |
| | fgetc(handle) |
| | read_dbg_qword(ea)<br>Get value of program quadro-word using the debugger memory |
| | get_last_index(tag, array_id)<br>Get index of last existing array element. |
| | ftell(handle) |
| | set_segm_name(ea, name)<br>Change name of the segment |
| | ltoa(n, radix) |
| | del_array_element(tag, array_id, idx) |

| | |
|---|---|
| | Delete an array element. |
| | gen_file(filetype, path, ea1, ea2, flags)<br>Generate an output file |
| | sel2para(sel)<br>Get a selector value |
| | del_segm(ea, flags)<br>Delete a segment |
| | set_default_sreg_value(ea, reg, value)<br>Set default segment register value for a segment |
| | isDec0(F) |
| | isDec1(F) |
| | print_insn_mnem(ea)<br>Get instruction mnemonics |
| | get_next_seg(ea)<br>Get next segment |
| | get_bpt_ea(n)<br>Get breakpoint address |
| | get_ordinal_qty()<br>Get number of local types + 1 |
| | get_member_strid(sid, member_offset)<br>Get structure id of a member |
| | get_first_module()<br>Enumerate process modules |
| | get_fixup_target_dis(ea)<br>Get fixup target displacement |
| | is_stkvar0(F) |
| | is_stkvar1(F) |
| | op_plain_offset(ea, n, base)<br>Convert operand to an offset (for the explanations of 'ea' and 'n' please see op_bin()) |
| | rename_array(array_id, newname)<br>Rename array, by its ID. |
| | is_stroff1(F) |
| | is_stroff0(F) |
| | rotate_left(value, count, nbits, offset)<br>Rotate a value to the left (or right) |
| | isRef(F) |
| | to_ea(seg, off)<br>Return value of expression: ((seg<<4) + off) |
| | get_hash_long(hash_id, key)<br>Gets the long value of a hash element. |
| | generate_disasm_line(ea, flags)<br>Get disassembly line |
| | find_binary(ea, flag, searchstr, radix=16, from_bc695=False) |
| | set_flag(off, bit, value) |
| | make_array(ea, nitems)<br>Create an array. |

| | |
|---|---|
| | validate_idb_names(do_repair=0)<br>check consistency of IDB name records |
| | get_enum_member(enum_id, value, serial, bmask)<br>Get id of constant |
| | get_numbered_type_name(ordinal)<br>Retrieve a local type name |
| | get_first_seg()<br>Get first segment |
| | parse_decls(inputtype, flags=0)<br>Parse type declarations |
| | del_stkpnt(func_ea, ea)<br>Delete SP register change point |
| | create_float(ea)<br>Convert the current item to a floating point (4 bytes) |
| | get_event_id()<br>Get ID of debug event |
| | print_decls(ordinals, flags)<br>Print types in a format suitable for use in a header file |
| | expand_struc(sid, offset, delta, recalc)<br>Expand or shrink a structure type |
| | op_offset_high16(ea, n, target)<br>Convert operand to a high offset High offset is the upper 16bits of an offset. |
| | get_last_hash_key(hash_id)<br>Get the last key in the hash. |
| | get_event_exc_ea()<br>Get address for EXCEPTION event |
| | is_flow(F) |
| | create_dword(ea)<br>Convert the current item to a double word (4 bytes) |
| | LoadFile(filepath, pos, ea, size)<br>Load file into IDA database |
| | set_segm_alignment(ea, alignment)<br>Change alignment of the segment |
| | value_is_int64(var) |
| | is_pack_real(F) |
| | import_type(idx, type_name)<br>Copy information from type library to database Copy structure, union, or enum definition from the type library to the IDA database. |
| | set_segm_type(segea, segtype)<br>Set segment type |
| | create_pack_real(ea)<br>Convert the current item to a packed real (10 or 12 bytes) |
| | remove_fchunk(funcea, tailea)<br>Remove a function chunk from the function |
| | SizeOf(typestr)<br>Returns the size of the type. |
| | is_tail(F) |
| | |

| | |
|---|---|
| | plan_and_wait(sEA, eEA, final_pass=True)<br>Perform full analysis of the range |
| | fputc(byte, handle) |
| | is_code(F) |
| | del_bpt(ea)<br>Delete breakpoint |
| | del_items(ea, flags=0, size=1)<br>Convert the current item to an explored item |
| | selector_by_name(segname)<br>Get segment selector by name |
| | get_fixup_target_off(ea)<br>Get fixup target offset |
| | get_func_flags(ea)<br>Retrieve function flags |
| | get_bmask_name(enum_id, bmask)<br>Get bitmask name (only for bitfields) |
| | add_struc(index, name, is_union)<br>Define a new structure type |
| | get_frame_regs_size(ea)<br>Get size of saved registers in function frame |
| | read_selection_end()<br>Get end address of the selected range |
| | read_selection_start()<br>Get start address of the selected range returns BADADDR - the user has not selected an range |
| | get_event_tid()<br>Get type ID for debug event |
| | func_contains(func_ea, ea)<br>Does the given function contain the given address? |
| | set_fixup(ea, fixuptype, fixupflags, targetsel, targetoff, displ)<br>Set fixup information |
| | parse_decl(inputtype, flags)<br>Parse type declaration |
| | get_fchunk_attr(ea, attr)<br>Get a function chunk attribute |
| | first_func_chunk(funcea)<br>Get the first function chunk of the specified function |
| | get_enum_member_name(const_id)<br>Get name of a constant |
| | get_prev_func(ea)<br>Find previous function |
| | is_byte(F) |
| | value_is_long(var) |
| | strstr(s1, s2) |
| | set_segment_bounds(ea, startea, endea, flags)<br>Change segment boundaries |
| | get_func_cmt(ea, repeatable) |

| | |
|---|---|
| | Retrieve function comment |
| | get_last_enum_member(enum_id, bmask)<br>Get last constant in the enum |
| | split_sreg_range(ea, reg, value, tag=2)<br>Set value of a segment register. |
| | hasUserName(F) |
| | get_fchunk_referer(ea, idx)<br>Get a function chunk referer |
| | set_member_cmt(sid, member_offset, comment, repeatable)<br>Change structure member comment |
| | get_event_ea()<br>Get ea for debug event |
| | fopen(f, mode) |
| | get_spd(ea)<br>Get current delta for the stack pointer |
| | find_selector(val)<br>Find a selector which has the specifed value |
| | readlong(handle, mostfirst) |
| | atol(s) |
| | atoa(ea)<br>Convert address value to a string Return address in the form 'seg000:1234' (the same as in line prefixes) |
| | find_func_end(ea)<br>Determine a new function boundaries |
| | get_first_enum_member(enum_id, bmask)<br>Get first constant in the enum |
| | is_manual1(F) |
| | is_manual0(F) |
| | get_frame_lvar_size(ea)<br>Get size of local variables in function frame |
| | add_struc_member(sid, name, offset, flag, typeid, nbytes, target=-1, tdelta=0, reftype=2)<br>Add structure member |
| | is_strlit(F) |
| | append_func_tail(funcea, ea1, ea2)<br>Append a function chunk to the function |
| | set_name(ea, name, flags=0)<br>Rename an address |
| | fclose(handle) |
| | gen_simple_call_chart(outfile, title, flags)<br>Generate a function call graph GDL file |
| | get_first_hash_key(hash_id)<br>Get the first key in the hash. |
| | set_color(ea, what, color)<br>Set item color |
| | rotate_dword(x, count) |
| | get_min_spd_ea(func_ea) |

| | |
|---|---|
| | Return the address with the minimal spd (stack pointer delta) If there are no SP change points, then return BADADDR. |
| | get_prev_fchunk(ea)<br>Get previous function chunk |
| | isHex1(F) |
| | isHex0(F) |
| | get_first_index(tag, array_id)<br>Get index of the first existing array element. |
| | get_next_enum_member(enum_id, value, bmask)<br>Get next constant in the enum |
| | get_member_size(sid, member_offset)<br>Get size of a member |
| | set_segm_class(ea, segclass)<br>Change class of the segment |
| | get_frame_args_size(ea)<br>Get size of arguments in function frame which are purged upon return |
| | get_local_tinfo(ordinal)<br>Get local type information as 'typeinfo' object |
| | GetDouble(ea)<br>Get value of a floating point number (8 bytes) This function assumes number stored using IEEE format and in the same endianness as integers. |
| | fprintf(handle, format, *args) |
| | add_enum(idx, name, flag)<br>Add a new enum type |
| | save_database(idbname, flags=0)<br>Save current database to the specified idb file |
| | idadir()<br>Get IDA directory |
| | get_next_func(ea)<br>Find next function |
| | get_prev_hash_key(hash_id, key)<br>Get the previous key in the hash. |
| | EVAL_FAILURE(code)<br>Check the result of eval_idc() for evaluation failures |
| | set_array_long(array_id, idx, value)<br>Sets the long value of an array element. |
| | isOct0(F) |
| | isOct1(F) |
| | set_segm_addressing(ea, bitness)<br>Change segment addressing |
| | is_off1(F) |
| | is_off0(F) |
| | op_stroff(ea, n, strid, delta)<br>Convert operand to an offset in a structure |
| | get_enum_member_cmt(const_id, repeatable)<br>Get comment of a constant |
| | create_strlit(ea, endea) |

| | |
|---|---|
| | Create a string. |
| | add_segm_ex(startea, endea, base, use32, align, comb, flags)<br>Create a new segment |
| | set_bpt_cond(ea, cnd, is_lowcnd=0)<br>Set breakpoint condition |
| | read_dbg_word(ea)<br>Get value of program word using the debugger memory |
| | send_dbg_command(cmd)<br>Sends a command to the debugger module and returns the output string. |
| | get_bmask_cmt(enum_id, bmask, repeatable)<br>Get bitmask comment (only for bitfields) |
| | get_last_member(sid)<br>Get offset of the last member of a structure |
| | has_value(F) |
| | AutoMark(ea, qtype)<br>Plan to analyze an address |
| | rotate_word(x, count) |
| | savefile(filepath, pos, ea, size) |
| | add_func(start, end=4294967295)<br>Create a function |
| | writestr(handle, s) |
| | fseek(handle, offset, origin) |
| | get_module_name(base)<br>Get process module name |
| | is_double(F) |
| | set_func_cmt(ea, cmt, repeatable)<br>Set function comment |
| | set_member_name(sid, member_offset, name)<br>Change structure member name |
| | set_bmask_name(enum_id, bmask, name)<br>Set bitmask name (only for bitfields) |
| | is_loaded(ea)<br>Is the byte initialized? |
| | readshort(handle, mostfirst) |
| | get_item_size(ea)<br>Get size of instruction or data item in bytes |
| | print_operand(ea, n)<br>Get operand of an instruction or data |
| | get_func_off_str(ea)<br>Convert address to 'funcname+offset' string |
| | writeshort(handle, word, mostfirst) |
| | get_module_size(base)<br>Get process module size |
| | MakeVar(ea)<br>Mark the location as "variable" |
| | SetType(ea, newtype) |

| | |
|---|---|
| | Set type of function/variable |
| | find_text(ea, flag, y, x, searchstr, from_bc695=False) |
| | readstr(handle) |
| | get_frame_size(ea)<br>Get full size of function frame |
| | toggle_bnot(ea, n)<br>Toggle the bitwise not operator for the operand |
| | is_char1(F) |
| | is_char0(F) |
| | is_word(F) |
| | get_prev_index(tag, array_id, idx)<br>Get index of the previous existing array element. |
| | get_event_info()<br>Get debug event info |
| | is_qword(F) |
| | get_next_fchunk(ea)<br>Get next function chunk |
| | update_hidden_range(ea, visible)<br>Set hidden range state |
| | get_name(ea, gtn_flags=0)<br>Get name at the specified address |
| | add_default_til(name)<br>Load a type library |
| | next_head(ea, maxea=4294967295)<br>Get next defined item (instruction or data) in the program |
| | get_str_type(ea)<br>Get string type |
| | set_hash_string(hash_id, key, value)<br>Sets the string value of a hash element. |
| | get_operand_type(ea, n)<br>Get type of instruction operand |
| | set_fchunk_attr(ea, attr, value)<br>Set a function chunk attribute |
| | get_func_name(ea)<br>Retrieve function name |
| | get_segm_by_sel(base)<br>Get segment by segment base |
| | is_align(F) |
| | get_type(ea)<br>Get type of function/variable |
| | add_bpt(ea, size=0, bpttype=12)<br>Add a new breakpoint |
| | get_inf_attr(offset) |
| | can_exc_continue()<br>Can it continue after EXCEPTION event? |
| | force_bl_jump(ea) |

| | |
|---|---|
| | Some ARM compilers in Thumb mode use BL (branch-and-link) instead of B (branch) for long jumps, since BL has more range. |
| | get_bpt_attr(ea, bptattr)<br>Get the characteristics of a breakpoint |
| | process_config_line(directive)<br>Parse one or more ida.cfg config directives |
| | MakeFunction(start, end=ida_idaapi.BADADDR) |
| | FindBinary(ea, flag, searchstr, radix=16) |
| | FindText(ea, flag, y, x, text) |
| | MakeStr(ea, endea) |
| | GetProcessorName() |
| | SegStart(ea) |
| | SegEnd(ea) |
| | SetSegmentType(ea, type) |
| | here() |
| | is_mapped(ea) |

| Variables | |
|---|---|
| | __EA64__ = `False` |
| | SendDbgCommand = `send_dbg_command` |
| | ApplyType = `apply_type` |
| | GetManyBytes = `get_bytes` |
| | GetString = `get_strlit_contents` |
| | ClearTraceFile = `clear_trace` |
| | NextHead = `next_head` |
| | ParseTypes = `parse_decls` |
| | PrevHead = `prev_head` |
| | ProcessUiAction = `process_ui_action` |
| | SaveBase = `save_database` |
| | Eval = `eval_idc` |
| | ARGV = `['tools/docs/hrdoc.py']`<br>The command line arguments passed to IDA via the -S switch. |
| | __package__ = `None` |

Function Details

## get_name_ea_simple(name)

Get linear address of a name

Parameters:

- **name** - name of program byte

Returns:
address of the name BADADDR - No such name

## get_event_module_size()

Get module size for debug event

Returns:
module size

## batch(batch)

Enable/disable batch mode of operation

Parameters:

- **batch** - batch mode 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

## get_sp_delta(ea)

Get modification of SP made by the instruction

Parameters:

- **ea** - end address of the instruction i.e.the last address of the instruction+1

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

## set_segm_attr(segea, attr, value)

Set segment attribute

Parameters:

- **segea** - any address within segment
- **attr** - one of SEGATTR_... constants

**Note:** Please note that not all segment attributes are modifiable. Also some of them should be modified using special functions like set_segm_addressing, etc.

## del_hash_string(hash_id, key)

Delete a hash element.

Parameters:

- **hash_id** - The hash ID.
- **key** - Key of an element

Returns:
1 upon success, 0 otherwise.

## get_frame_id(ea)

Get ID of function frame structure

Parameters:

- **ea** - any address belonging to the function

Returns:
ID of function frame or None In order to access stack variables you need to use structure member manipulaion functions with the obtained ID.

## create_word(ea)

Convert the current item to a word (2 bytes)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## set_bpt_attr(address, bptattr, value)

modifiable characteristics of a breakpoint

Parameters:

- **address** - any address in the breakpoint range
- **bptattr** - the attribute code, one of BPTATTR_* constants BPTATTR_CND is not allowed, see set_bpt_cond()
- **value** - the attibute value

Returns:
success

## get_segm_start(ea)

Get start address of a segment

Parameters:

- **ea** - any address in the segment

Returns:

start of segment BADADDR - the specified address doesn't belong to any segment

### set_array_string(array_id, idx, value)

Sets the string value of an array element.

Parameters:

- **array_id** - The array ID.
- **idx** - Index of an element.
- **value** - String value to store in the array

Returns:
1 in case of success, 0 otherwise

### qsleep(milliseconds)

qsleep the specified number of milliseconds This function suspends IDA for the specified amount of time

Parameters:

- **milliseconds** - time to sleep

### byte_value(F)

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.

### enable_tracing(trace_level, enable)

Enable step tracing

Parameters:

- **trace_level** - what kind of trace to modify
- **enable** - 0: turn off, 1: turn on

Returns:
success

### get_sreg(ea, reg)

Get value of segment register at the specified address

Parameters:

- **ea** - linear address
- **reg** - name of segment register

Returns:
the value of the segment register or -1 on error

**Note:** The segment registers in 32bit program usually contain selectors, so to get paragraph pointed to by the segment register you need to call sel2para() function.

## create_struct(ea, size, strname)

Convert the current item to a structure instance

Parameters:

- **ea** - linear address
- **size** - structure size in bytes. -1 means that the size will be calculated automatically
- **strname** - name of a structure type

Returns:
1-ok, 0-failure

## GetLocalType(ordinal, flags)

Retrieve a local type declaration

Parameters:

- **flags** - any of PRTYPE_* constants

Returns:
local type as a C declaration or ""

## get_bpt_qty()

Get number of breakpoints.

Returns:
number of breakpoints

## get_fixup_target_flags(ea)

Get fixup target flags

Parameters:

- **ea** - address to get information about

Returns:
0 - no fixup at the specified address otherwise returns fixup target flags

## SaveFile(filepath, pos, ea, size)

Save from IDA database to file

Parameters:

- **filepath** - path to output file
- **pos** - position in the file
- **ea** - linear address to save from
- **size** - number of bytes to save

Returns:
0 - error, 1 - ok

## get_event_exc_code()

Get exception code for EXCEPTION event

Returns:
exception code

## delete_array(array_id)

Delete array, by its ID.

Parameters:

- **array_id** - The ID of the array to delete.

## call_system(command)

Execute an OS command.

Parameters:

- **command** - command line to execute

Returns:
error code from OS

**Note:** IDA will wait for the started program to finish. In order to start the command in parallel, use OS methods. For example, you may start another program in parallel using "start" command.

## del_struc_member(sid, member_offset)

Delete structure member

Parameters:

- **sid** - structure type ID
- **member_offset** - offset of the member

Returns:
!= 0 - ok.

**Note:** IDA allows 'holes' between members of a structure. It treats these 'holes' as unnamed arrays of bytes.

## is_event_handled()

Is the debug event handled?

Returns:
boolean

## create_double(ea)

Convert the current item to a double floating point (8 bytes)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## set_tail_owner(tailea, funcea)

Change the function chunk owner

Parameters:

- **tailea** - any address in the function chunk
- **funcea** - the starting address of the new owner

Returns:
False if failed, True if success

**Note:** The new owner must already have the chunk appended before the call

## set_segm_combination(segea, comb)

Change combination of the segment

Parameters:

- **segea** - any address in the segment
- **comb** - new combination of the segment (one of the sc... constants)

Returns:
success (boolean)

## is_union(sid)

Is a structure a union?

Parameters:

- **sid** - structure type ID

Returns:
1: yes, this is a union id 0: no

**Note:** Unions are a special kind of structures

## create_yword(ea)

Convert the current item to a ymm word (32 bytes/256 bits)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## get_next_offset(sid, offset)

Get next offset in a structure

Parameters:

- **sid** - structure type ID
- **offset** - current offset

Returns:
-1 if bad structure type ID is passed, ida_idaapi.BADADDR if no (more) offsets in the structure, otherwise returns next offset in a structure.

**Notes:**

- 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.
- Union members are, in IDA's internals, located at subsequent byte offsets: member 0 -> offset 0x0, member 1 -> offset 0x1, etc...

## set_func_flags(ea, flags)

Change function flags

Parameters:

- **ea** - any address belonging to the function
- **flags** - see get_func_flags() for explanations

Returns:
!=0 - ok

## get_member_cmt(sid, member_offset, repeatable)

Get comment of a member

Parameters:

- **sid** - 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 2,3,4,5 denote the same structure member.
- **repeatable** - 1: get repeatable comment 0: get regular comment

Returns:
None if bad structure type ID is passed or no such member in the structure otherwise returns comment of the specified member.

## set_bmask_cmt(enum_id, bmask, cmt, repeatable)

Set bitmask comment (only for bitfields)

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant
- **cmt** - comment repeatable - type of comment, 0-regular, 1-repeatable

Returns:
1-ok, 0-failed

## get_event_module_name()

Get module name for debug event

Returns:
module name

## create_array(name)

Create array.

Parameters:

- **name** - The array name.

Returns:
-1 in case of failure, a valid array_id otherwise.

## get_event_exc_info()

Get info for EXCEPTION event

Returns:

info string

## set_member_type(sid, member_offset, flag, typeid, nitems, target=-1, tdelta=0, reftype=2)

Change structure member type

Parameters:

- **sid** - structure type ID
- **member_offset** - offset of the member
- **flag** - new type of the member. Should be one of FF_BYTE..FF_PACKREAL (see above) combined with FF_DATA
- **typeid** - if isStruc(flag) then typeid specifies the structure id for the member if is_off0(flag) then typeid specifies the offset base. if is_strlit(flag) then typeid specifies the string type (STRTYPE_...). if is_stroff(flag) then typeid specifies the structure id if is_enum(flag) then typeid specifies the enum id if is_custom(flags) then typeid specifies the dtid and fid: dtid|(fid<<16) Otherwise typeid should be -1.
- **nitems** - number of items in the member
- **target** - target address of the offset expr. You may specify it as -1, ida will calculate it itself
- **tdelta** - offset target delta. usually 0
- **reftype** - see REF_... definitions

Returns:
!=0 - ok.

**Note:** The remaining arguments are allowed only if is_off0(flag) and you want to specify a complex offset expression

## force_bl_call(ea)

Force BL instruction to be a call

Parameters:

- **ea** - address of the BL instruction

Returns:
1-ok, 0-failed

## get_prev_offset(sid, offset)

Get previous offset in a structure

Parameters:

- **sid** - structure type ID
- **offset** - current offset

Returns:
-1 if bad structure type ID is passed, ida_idaapi.BADADDR if no (more) offsets in the structure, otherwise returns previous offset in a structure.

**Notes:**

- 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.
- Union members are, in IDA's internals, located at subsequent byte offsets: member 0 -> offset 0x0, member 1 -> offset 0x1, etc...

## GetFloat(ea)

Get value of a floating point number (4 bytes) This function assumes number stored using IEEE format and in the same endianness as integers.

Parameters:

- **ea** - linear address

Returns:
float

## get_event_module_base()

Get module base for debug event

Returns:
module base

## get_fixup_target_sel(ea)

Get fixup target selector

Parameters:

- **ea** - address to get information about

Returns:
BADSEL - no fixup at the specified address otherwise returns fixup target selector

## gen_flow_graph(outfile, title, ea1, ea2, flags)

Generate a flow chart GDL file

Parameters:

- **outfile** - output file name. GDL extension will be used
- **title** - graph title
- **ea1** - beginning of the range to flow chart
- **ea2** - end of the range to flow chart.
- **flags** - combination of CHART_... constants

**Note:** If ea2 == BADADDR then ea1 is treated as an address within a function. That function will be flow charted.

## delete_all_segments()

Delete all segments, instructions, comments, i.e. everything except values of bytes.

## read_dbg_dword(ea)

Get value of program double-word using the debugger memory

Parameters:

- **ea** - linear address

Returns:
The value or None on failure.

## get_event_exit_code()

Get exit code for debug event

Returns:
exit code for PROCESS_EXITED, THREAD_EXITED events

## get_next_hash_key(hash_id, key)

Get the next key in the hash.

Parameters:

- **hash_id** - The hash ID.
- **key** - The current key.

Returns:
the next key, 0 otherwise

## eval_idc(expr)

Evaluate an IDC expression

Parameters:

- **expr** - an expression

Returns:
the expression value. If there are problems, the returned value will be "IDC_FAILURE: xxx" where xxx is the error description

**Note:** Python implementation evaluates IDC only, while IDC can call other registered languages

## guess_type(ea)

Guess type of function/variable

Parameters:

- **ea** - the address of the object, can be the structure member id too

Returns:
type string or None if failed

## get_hash_string(hash_id, key)

Gets the string value of a hash element.

Parameters:

- **hash_id** - The hash ID.
- **key** - Key of an element.

Returns:
the string value of the element, or None if no such element.

## create_byte(ea)

Convert the current item to a byte

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## get_event_pid()

Get process ID for debug event

Returns:
process ID

## create_tbyte(ea)

Convert the current item to a tbyte (10 or 12 bytes)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## get_strlit_contents(ea, length=-1, strtype=0)

Get string contents

Parameters:

- **ea** - linear address
- **length** - string length. -1 means to calculate the max string length
- **strtype** - the string type (one of STRTYPE_... constants)

Returns:
string contents or empty string

## get_member_qty(sid)

Get number of members of a structure

Parameters:

- **sid** - structure type ID

Returns:
-1 if bad structure type ID is passed otherwise returns number of members.

**Note:** Union members are, in IDA's internals, located at subsequent byte offsets: member 0 -> offset 0x0, member 1 -> offset 0x1, etc...

## get_curline()

Get the disassembly line at the cursor

Returns:
string

## get_next_index(tag, array_id, idx)

Get index of the next existing array element.

Parameters:

- **tag** - Tag of array, specifies one of two array types: AR_LONG, AR_STR
- **array_id** - The array ID.
- **idx** - Index of the current element.

Returns:
-1 if no more elements, otherwise returns index of the next array element of given type.

## get_tinfo(ea)

Get type information of function/variable as 'typeinfo' object

Parameters:

- **ea** - the address of the object

Returns:
None on failure, or (type, fields) tuple.

## del_struc(sid)

Delete a structure type

Parameters:

- **sid** - 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.

## process_ui_action(name, flags=0)

Invokes an IDA UI action by name

Parameters:

- **name** - Command name
- **flags** - Reserved. Must be zero

Returns:
Boolean

## prev_head(ea, minea=0)

Get previous defined item (instruction or data) in the program

Parameters:

- **ea** - linear address to start search from
- **minea** - the search will stop at the address minea is included in the search range

Returns:
BADADDR - no (more) defined items

## get_segm_attr(segea, attr)

Get segment attribute

Parameters:

- **segea** - any address within segment
- **attr** - one of SEGATTR_... constants

## write_dbg_memory(ea, data)

Write to debugger memory.

Parameters:

- **ea** - linear address
- **data** - string to write

Returns:
number of written bytes (-1 - network/debugger error)

Thread-safe function (may be called only from the main thread and debthread)

## GetDisasm(ea)

Get disassembly line

Parameters:

- **ea** - linear address of instruction

Returns:
"" - could not decode instruction at the specified location

**Note:** this function may not return exactly the same mnemonics as you see on the screen.

## choose_func(title)

Ask the user to select a function

Arguments:

Parameters:

- **title** - title of the dialog box

Returns:
-1 - user refused to select a function otherwise returns the selected function start address

## add_enum_member(enum_id, name, value, bmask)

Add a member of enum - a symbolic constant

Parameters:

- **enum_id** - id of enum
- **name** - name of symbolic constant. Must be unique in the program.
- **value** - value of symbolic constant.
- **bmask** - bitmask of the constant ordinary enums accept only ida_enum.DEFMASK as a bitmask all bits set in value should be set in bmask too

Returns:
0-ok, otherwise error code (one of ENUM_MEMBER_ERROR_*)

## get_next_module(base)

Enumerate process modules

Parameters:

- **base** - previous module's base address

Returns:
next module's base address or None on failure

## get_bytes(ea, size, use_dbg=False)

Return the specified number of bytes of the program

Parameters:

- **ea** - linear address
- **size** - size of buffer in normal 8-bit bytes
- **use_dbg** - if True, use debugger memory, otherwise just the database

Returns:
None on failure otherwise a string containing the read bytes

## get_color(ea, what)

Get item color

Parameters:

- **ea** - address of the item
- **what** - type of the item (one of CIC_* constants)

Returns:
color code in RGB (hex 0xBBGGRR)

## get_func_attr(ea, attr)

Get a function attribute

Parameters:

- **ea** - any address belonging to the function
- **attr** - one of FUNCATTR_... constants

Returns:
BADADDR - error otherwise returns the attribute value

## define_local_var(start, end, location, name)

Create a local variable

Parameters:

- **start** - start of address range for the local variable
- **end** - end of address range for the local variable
- **location** - the variable location in the "[bp+xx]" form where xx is a number. The location can also be specified as a register name.
- **name** - name of the local variable

Returns:
1-ok, 0-failure

**Note:** For the stack variables the end address is ignored. If there is no function at 'start' then this function. will fail.

## get_operand_value(ea, n)

Get number used in the operand

This function returns an immediate number used in the operand

Parameters:

- **ea** - linear address of instruction
- **n** - the operand number

Returns:
value operand is an immediate value => immediate value operand has a displacement => displacement operand is a direct memory ref => memory address operand is a register => register number operand is a register phrase => phrase number otherwise => -1

## get_member_name(sid, member_offset)

Get name of a member of a structure

Parameters:

- **sid** - 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 2,3,4,5 denote the same structure member.

Returns:
None if bad structure type ID is passed or no such member in the structure otherwise returns name of the specified member.

## get_member_offset(sid, member_name)

Get offset of a member of a structure by the member name

Parameters:

- **sid** - 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.

**Note:** Union members are, in IDA's internals, located at subsequent byte offsets: member 0 -> offset 0x0, member 1 -> offset 0x1, etc...

## get_event_bpt_hea()

Get hardware address for BREAKPOINT event

Returns:
hardware address

## next_func_chunk(funcea, tailea)

Get the next function chunk of the specified function

Parameters:

- **`funcea`** - any address in the function
- **`tailea`** - any address in the current chunk

Returns:
the starting address of the next function chunk or BADADDR

**Note:** This function returns the next chunk of the specified function

## get_fixup_target_type(ea)

Get fixup target type

Parameters:

- **`ea`** - address to get information about

Returns:
0 - no fixup at the specified address otherwise returns fixup type

## set_reg_value(value, name)

Set register value

Parameters:

- **`name`** - the register name
- **`value`** - new register value

**Note:** The debugger should be running It is not necessary to use this function to set register values. A register name in the left side of an assignment will do too.

## set_local_type(ordinal, input, flags)

Parse one type declaration and store it in the specified slot

Parameters:

- **ordinal** - slot number (1...NumberOfLocalTypes) -1 means allocate new slot or reuse the slot of the existing named type
- **input** - C declaration. Empty input empties the slot
- **flags** - combination of PT_... constants or 0

Returns:
slot number or 0 if error

## set_struc_idx(sid, index)

Change structure index

Parameters:

- **sid** - structure type ID
- **index** - new index of the structure

Returns:
!= 0 - ok

**Note:** See get_first_struc_idx() for the explanation of structure indices and IDs.

## create_qword(ea)

Convert the current item to a quadro word (8 bytes)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## get_prev_enum_member(enum_id, value, bmask)

Get prev constant in the enum

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant ordinary enums accept only ida_enum.DEFMASK as a bitmask
- **value** - value of the current constant

Returns:
value of a constant with value lower than the specified value. idaapi.BADNODE no such constants exist. All constants are sorted by their values as unsigned longs.

## get_member_flag(sid, member_offset)

Get type of a member

Parameters:

- **sid** - 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 2,3,4,5 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.

## add_auto_stkpnt(func_ea, ea, delta)

Add automatical SP register change point

Parameters:

- **func_ea** - function start
- **ea** - linear address where SP changes usually this is the end of the instruction which modifies the stack pointer (insn.ea+insn.size)
- **delta** - difference between old and new values of SP

Returns:
1-ok, 0-failed

## get_first_member(sid)

Get offset of the first member of a structure

Parameters:

- **sid** - structure type ID

Returns:
-1 if bad structure type ID is passed, ida_idaapi.BADADDR if structure has no members, otherwise returns offset of the first member.

**Notes:**

- IDA allows 'holes' between members of a structure. It treats these 'holes' as unnamed arrays of bytes.
- Union members are, in IDA's internals, located at subsequent byte offsets: member 0 -> offset 0x0, member 1 -> offset 0x1, etc...

## set_array_params(ea, flags, litems, align)

Set array representation format

Parameters:

- **ea** - linear address

- **flags** - combination of AP_... constants or 0
- **litems** - number of items per line. 0 means auto
- **align** - element alignment
  - -1: do not align
  - 0: automatic alignment
  - other values: element width

Returns:
1-ok, 0-failure

## move_segm(ea, to, flags)

Move a segment to a new address This function moves all information to the new address It fixes up address sensitive information in the kernel The total effect is equal to reloading the segment to the target address

Parameters:

- **ea** - any address within the segment to move
- **to** - new segment start address
- **flags** - combination MFS_... constants

Returns:
MOVE_SEGM_... error code

## demangle_name(name, disable_mask)

demangle_name a name

Parameters:

- **name** - name to demangle
- **disable_mask** - a mask that tells how to demangle the name it is a good idea to get this mask using get_inf_attr(INF_SHORT_DN) or get_inf_attr(INF_LONG_DN)

Returns:
a demangled name If the input name cannot be demangled, returns None

## get_array_id(name)

Get array array_id, by name.

Parameters:

- **name** - The array name.

Returns:
-1 in case of failure (i.e., no array with that name exists), a valid array_id otherwise.

## del_enum_member(enum_id, value, serial, bmask)

Delete a member of enum - a symbolic constant

Parameters:

- **enum_id** - id of enum
- **value** - value of symbolic constant.
- **serial** - serial number of the constant in the enumeration. See op_enum() for for details.
- **bmask** - bitmask of the constant ordinary enums accept only ida_enum.DEFMASK as a bitmask

Returns:
1-ok, 0-failed

## get_segm_name(ea)

Get name of a segment

Parameters:

- **ea** - any address in the segment

Returns:
"" - no segment at the specified address

## get_xref_type()

Return type of the last xref obtained by [RD]first/next[B0] functions.

Returns:
constants fl_* or dr_*

## get_member_id(sid, member_offset)

Parameters:

- **sid** - structure type ID
- **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 2,3,4,5 denote the same structure member.

Returns:
-1 if bad structure type ID is passed or there is no member at the specified offset. otherwise returns the member id.

## read_dbg_byte(ea)

Get value of program byte using the debugger memory

Parameters:

- **ea** - linear address

Returns:
The value or None on failure.

## apply_type(ea, py_type, flags=1)

```
Apply the specified type to the address

@param ea: the address of the object
@param py_type: typeinfo tuple (type, fields) as get_tinfo() returns
            or tuple (name, type, fields) as parse_decl() returns
            or None
           if specified as None, then the
           item associated with 'ea' will be deleted.
@param flags: combination of TINFO_... constants or 0
@return: Boolean
```

## get_idb_path()

Get IDB full path

This function returns full path of the current IDB database

## get_array_element(tag, array_id, idx)

Get value of array element.

Parameters:

- **tag** - Tag of array, specifies one of two array types: AR_LONG, AR_STR
- **array_id** - The 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.

## create_oword(ea)

Convert the current item to an octa word (16 bytes/128 bits)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## set_func_attr(ea, attr, value)

Set a function attribute

Parameters:

- **ea** - any address belonging to the function
- **attr** - one of FUNCATTR_... constants
- **value** - new value of the attribute

Returns:
1-ok, 0-failed

## set_hash_long(hash_id, key, value)

Sets the long value of a hash element.

Parameters:

- **hash_id** - The hash ID.
- **key** - Key of an element.
- **value** - 32bit or 64bit value to store in the hash

Returns:
1 in case of success, 0 otherwise

## get_segm_end(ea)

Get end address of a segment

Parameters:

- **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

## get_reg_value(name)

Get register value

Parameters:

- **name** - the register name

Returns:
register value (integer or floating point)

**Note:** The debugger should be running. otherwise the function fails the register name should be valid. It is not necessary to use this function to get register values because a register name in the script will do too.

## set_frame_size(ea, lvsize, frregs, argsize)

Make function frame

Parameters:

- **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

## read_dbg_qword(ea)

Get value of program quadro-word using the debugger memory

Parameters:

- **ea** - linear address

Returns:
The value or None on failure.

## get_last_index(tag, array_id)

Get index of last existing array element.

Parameters:

- **tag** - Tag of array, specifies one of two array types: AR_LONG, AR_STR
- **array_id** - The array ID.

Returns:
-1 if the array is empty, otherwise index of first array element of given type.

## set_segm_name(ea, name)

Change name of the segment

Parameters:

- **ea** - any address in the segment
- **name** - new name of the segment

Returns:
success (boolean)

## del_array_element(tag, array_id, idx)

Delete an array element.

Parameters:

- **tag** - Tag of array, specifies one of two array types: AR_LONG, AR_STR
- **array_id** - The array ID.
- **idx** - Index of an element.

Returns:
1 in case of success, 0 otherwise.

## gen_file(filetype, path, ea1, ea2, flags)

Generate an output file

Parameters:

- **filetype** - type of output file. One of OFILE_... symbols. See below.
- **path** - the output file path (will be overwritten!)
- **ea1** - start address. For some file types this argument is ignored
- **ea2** - end address. For some file types this argument is ignored
- **flags** - bit combination of GENFLG_...

Returns:
number of the generated lines. -1 if an error occurred OFILE_EXE: 0-can't generate exe file, 1-ok

## sel2para(sel)

Get a selector value

Parameters:

- **sel** - the selector number

Returns:
selector value if found otherwise the input value (sel)

**Note:** selector values are always in paragraphs

## del_segm(ea, flags)

Delete a segment

Parameters:

- **ea** - any address in the segment
- **flags** - combination of SEGMOD_* flags

Returns:
boolean success

## set_default_sreg_value(ea, reg, value)

Set default segment register value for a segment

Parameters:

- **ea** - any address in the segment if no segment is present at the specified address then all segments will be affected
- **reg** - name of segment register
- **value** - default value of the segment register. -1-undefined.

## print_insn_mnem(ea)

Get instruction mnemonics

Parameters:

- **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.

## get_next_seg(ea)

Get next segment

Parameters:

- **ea** - linear address

Returns:
start of the next segment BADADDR - no next segment

## get_bpt_ea(n)

Get breakpoint address

Parameters:

- **n** - number of breakpoint, is in range 0..get_bpt_qty()-1

Returns:
address of the breakpoint or BADADDR

## get_ordinal_qty()

Get number of local types + 1

Returns:
value >= 1. 1 means that there are no local types.

## get_member_strid(sid, member_offset)

Get structure id of a member

Parameters:

- **sid** - 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 2,3,4,5 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 -1.

### get_first_module()

Enumerate process modules

Returns:
first module's base address or None on failure

### get_fixup_target_dis(ea)

Get fixup target displacement

Parameters:

- **ea** - address to get information about

Returns:
0 - no fixup at the specified address otherwise returns fixup target displacement

### op_plain_offset(ea, n, base)

Convert operand to an offset (for the explanations of 'ea' and 'n' please see op_bin())

# 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!',0

Then you need to specify a linear address of the segment base to create a proper offset:

op_plain_offset(["seg000",0x2000],0,0x10000);

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:

Please note that the outer offsets are meaningful only for Motorola 680x0.

Parameters:

- **ea** - linear address
- **n** - number of operand
  - 0 - the first operand
  - 1 - the second, third and all other operands
  - -1 - all operands
- **base** - base of the offset as a linear address If base == BADADDR then the current operand becomes non-offset

## rename_array(array_id, newname)

Rename array, by its ID.

Parameters:

- **id** - The ID of the array to rename.
- **newname** - The new name of the array.

Returns:
1 in case of success, 0 otherwise

## rotate_left(value, count, nbits, offset)

Rotate a value to the left (or right)

Parameters:

- **value** - value to rotate
- **count** - number of times to rotate. negative counter means rotate to the right
- **nbits** - number of bits to rotate
- **offset** - offset of the first bit to rotate

Returns:
the value with the specified field rotated all other bits are not modified

## get_hash_long(hash_id, key)

Gets the long value of a hash element.

Parameters:

- **hash_id** - The hash ID.
- **key** - Key of an element.

Returns:
the 32bit or 64bit value of the element, or 0 if no such element.

## generate_disasm_line(ea, flags)

Get disassembly line

Parameters:

- **ea** - linear address of instruction
- **flags** - combination of the GENDSM_ flags, or 0

Returns:
"" - could not decode instruction at the specified location

**Note:** this function may not return exactly the same mnemonics as you see on the screen.

## make_array(ea, nitems)

Create an array.

Parameters:

- **ea** - linear address
- **nitems** - size of array in items

**Note:** This function will create an array of the items with the same type as the type of the item at 'ea'. If the byte at 'ea' is undefined, then this function will create an array of bytes.

## validate_idb_names(do_repair=0)

check consistency of IDB name records

Parameters:

- **do_repair** - try to repair netnode header it TRUE

Returns:
number of inconsistent name records

## get_enum_member(enum_id, value, serial, bmask)

Get id of constant

Parameters:

- **enum_id** - id of enum
- **value** - value of constant
- **serial** - serial number of the constant in the enumeration. See op_enum() for details.
- **bmask** - bitmask of the constant ordinary enums accept only ida_enum.DEFMASK as a bitmask

Returns:
id of constant or -1 if error

## get_numbered_type_name(ordinal)

Retrieve a local type name

Parameters:

- **ordinal** - slot number (1...NumberOfLocalTypes)

  returns: local type name or None

## get_first_seg()

Get first segment

Returns:
address of the start of the first segment BADADDR - no segments are defined

## parse_decls(inputtype, flags=0)

Parse type declarations

Parameters:

- **inputtype** - file name or C declarations (depending on the flags)
- **flags** - combination of PT_... constants or 0

Returns:
number of parsing errors (0 no errors)

## del_stkpnt(func_ea, ea)

Delete SP register change point

Parameters:

- **func_ea** - function start
- **ea** - linear address

Returns:
1-ok, 0-failed

## create_float(ea)

Convert the current item to a floating point (4 bytes)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## get_event_id()

Get ID of debug event

Returns:
event ID

## print_decls(ordinals, flags)

Print types in a format suitable for use in a header file

Parameters:

- **ordinals** - comma-separated list of type ordinals
- **flags** - combination of PDF_... constants or 0

Returns:
string containing the type definitions

## expand_struc(sid, offset, delta, recalc)

Expand or shrink a structure type

Parameters:

- **id** - structure type ID
- **offset** - offset in the structure
- **delta** - how many bytes to add or remove
- **recalc** - recalculate the locations where the structure type is used

Returns:
!= 0 - ok

## op_offset_high16(ea, n, target)

Convert operand to a high offset High offset is the upper 16bits of an offset. This type is used by TMS320C6 processors (and probably by other RISC processors too)

Parameters:

- **ea** - linear address
- **n** - number of operand
  - 0 - the first operand
  - 1 - the second, third and all other operands
  - -1 - all operands
- **target** - the full value (all 32bits) of the offset

## get_last_hash_key(hash_id)

Get the last key in the hash.

Parameters:

- **hash_id** - The hash ID.

Returns:
the key, 0 otherwise.

## get_event_exc_ea()

Get address for EXCEPTION event

Returns:
adress of exception

## create_dword(ea)

Convert the current item to a double word (4 bytes)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## LoadFile(filepath, pos, ea, size)

Load file into IDA database

Parameters:

- **filepath** - path to input file
- **pos** - position in the file
- **ea** - linear address to load
- **size** - number of bytes to load

Returns:
0 - error, 1 - ok

## set_segm_alignment(ea, alignment)

Change alignment of the segment

Parameters:

- **ea** - any address in the segment
- **alignment** - new alignment of the segment (one of the sa... constants)

Returns:
success (boolean)

## import_type(idx, type_name)

Copy information from type library to database Copy structure, union, or enum definition from the type library to the IDA database.

Parameters:

- **idx** - the position of the new type in the list of types (structures or enums) -1 means at the end of the list
- **type_name** - name of type to copy

Returns:
BADNODE-failed, otherwise the type id (structure id or enum id)

## set_segm_type(segea, segtype)

Set segment type

Parameters:

- **segea** - any address within segment
- **segtype** - new segment type:

Returns:
!=0 - ok

## create_pack_real(ea)

Convert the current item to a packed real (10 or 12 bytes)

Parameters:

- **ea** - linear address

Returns:
1-ok, 0-failure

## remove_fchunk(funcea, tailea)

Remove a function chunk from the function

Parameters:

- **funcea** - any address in the function
- **tailea** - any address in the function chunk to remove

Returns:
0 if failed, 1 if success

## SizeOf(typestr)

Returns the size of the type. It is equivalent to IDC's sizeof(). Use name, tp, fld = idc.parse_decl() ; SizeOf(tp) to retrieve the size

Returns:
-1 if typestring is not valid otherwise the size of the type

## plan_and_wait(sEA, eEA, final_pass=True)

Perform full analysis of the range

Parameters:

- **sEA** - starting linear address
- **eEA** - ending linear address (excluded)
- **final_pass** - make the final pass over the specified range

Returns:
1-ok, 0-Ctrl-Break was pressed.

## del_bpt(ea)

Delete breakpoint

Parameters:

- **ea** - any address in the process memory space:

Returns:
success

## del_items(ea, flags=0, size=1)

Convert the current item to an explored item

Parameters:

- **ea** - linear address
- **flags** - combination of DELIT_* constants
- **size** - size of the range to undefine

Returns:
None

## selector_by_name(segname)

Get segment selector by name

Parameters:

- **segname** - name of segment

Returns:
segment selector or BADADDR

## get_fixup_target_off(ea)

Get fixup target offset

Parameters:

- **ea** - address to get information about

Returns:
BADADDR - no fixup at the specified address otherwise returns fixup target offset

## get_func_flags(ea)

Retrieve function flags

Parameters:

- **ea** - any address belonging to the function

Returns:
-1 - function doesn't exist otherwise returns the flags

## get_bmask_name(enum_id, bmask)

Get bitmask name (only for bitfields)

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant

Returns:
name of bitmask or None

## add_struc(index, name, is_union)

Define a new structure type

Parameters:

- **index** - index of new structure type If another structure has the specified index, then index of that structure and all other structures will be incremented, freeing the specifed index. If index is == -1, then the biggest index number will be used. See get_first_struc_idx() for the explanation of structure indices and IDs.
- **name** - name of the new structure type.
- **is_union** - 0: structure 1: union

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

## get_frame_regs_size(ea)

Get size of saved registers in function frame

Parameters:

- **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 None

## read_selection_end()

Get end address of the selected range

Returns:
BADADDR - the user has not selected an range

## get_event_tid()

Get type ID for debug event

Returns:
type ID

## func_contains(func_ea, ea)

Does the given function contain the given address?

Parameters:

- **func_ea** - any address belonging to the function
- **ea** - linear address

Returns:
success

## set_fixup(ea, fixuptype, fixupflags, targetsel, targetoff, displ)

Set fixup information

Parameters:

- **ea** - address to set fixup information about
- **fixuptype** - fixup type. see get_fixup_target_type() for possible fixup types.
- **fixupflags** - fixup flags. see get_fixup_target_flags() for possible fixup types.
- **targetsel** - target selector
- **targetoff** - target offset
- **displ** - displacement

Returns:

none

## parse_decl(inputtype, flags)

Parse type declaration

Parameters:

- **inputtype** - file name or C declarations (depending on the flags)
- **flags** - combination of PT_... constants or 0

Returns:
None on failure or (name, type, fields) tuple

## get_fchunk_attr(ea, attr)

Get a function chunk attribute

Parameters:

- **ea** - any address in the chunk
- **attr** - one of: FUNCATTR_START, FUNCATTR_END, FUNCATTR_OWNER, FUNCATTR_REFQTY

Returns:
desired attribute or -1

## first_func_chunk(funcea)

Get the first function chunk of the specified function

Parameters:

- **funcea** - any address in the function

Returns:
the function entry point or BADADDR

**Note:** This function returns the first (main) chunk of the specified function

## get_enum_member_name(const_id)

Get name of a constant

Parameters:

- **const_id** - id of const

Returns: name of constant

## get_prev_func(ea)

Find previous function

Parameters:

- **ea** - any address belonging to the function

Returns:
BADADDR - no more functions otherwise returns the previous function start address

## set_segment_bounds(ea, startea, endea, flags)

Change segment boundaries

Parameters:

- **ea** - any address in the segment
- **startea** - new start address of the segment
- **endea** - new end address of the segment
- **flags** - combination of SEGMOD_... flags

Returns:
boolean success

## get_func_cmt(ea, repeatable)

Retrieve function comment

Parameters:

- **ea** - any address belonging to the function
- **repeatable** - 1: get repeatable comment 0: get regular comment

Returns:
function comment string

## get_last_enum_member(enum_id, bmask)

Get last constant in the enum

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant (ordinary enums accept only ida_enum.DEFMASK as a bitmask)

Returns:
value of constant or idaapi.BADNODE no constants are defined All constants are sorted by their values as unsigned longs.

## split_sreg_range(ea, reg, value, tag=2)

Set value of a segment register.

Parameters:

- **ea** - linear address
- **reg** - name of a register, like "cs", "ds", "es", etc.
- **value** - new value of the segment register.
- **tag** - of SR_... constants

**Note:** 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.

## get_fchunk_referer(ea, idx)

Get a function chunk referer

Parameters:

- **ea** - any address in the chunk
- **idx** - referer index (0..get_fchunk_attr(FUNCATTR_REFQTY))

Returns:
referer address or BADADDR

## set_member_cmt(sid, member_offset, comment, repeatable)

Change structure member comment

Parameters:

- **sid** - structure type ID
- **member_offset** - offset of the member
- **comment** - new comment of the structure member
- **repeatable** - 1: change repeatable comment 0: change regular comment

Returns:
!= 0 - ok

## get_event_ea()

Get ea for debug event

Returns:
ea

## get_spd(ea)

Get current delta for the stack pointer

Parameters:

- **ea** - end address of the instruction i.e.the last address of the instruction+1

Returns:

The difference between the original SP upon entering the function and SP for the specified address

## find_selector(val)

Find a selector which has the specifed value

Parameters:

- **val** - value to search for

Returns:
the selector number if found, otherwise the input value (val & 0xFFFF)

**Note:** selector values are always in paragraphs

## atoa(ea)

Convert address value to a string Return address in the form 'seg000:1234' (the same as in line prefixes)

Parameters:

- **ea** - address to format

## find_func_end(ea)

Determine a new function boundaries

Parameters:

- **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

## get_first_enum_member(enum_id, bmask)

Get first constant in the enum

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant (ordinary enums accept only ida_enum.DEFMASK as a bitmask)

Returns:
value of constant or idaapi.BADNODE no constants are defined All constants are sorted by their values as unsigned longs.

## get_frame_lvar_size(ea)

Get size of local variables in function frame

Parameters:

- **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 None

## add_struc_member(sid, name, offset, flag, typeid, nbytes, target=-1, tdelta=0, reftype=2)

Add structure member

Parameters:

- **sid** - structure type ID
- **name** - name of the new member
- **offset** - offset of the new member -1 means to add at the end of the structure
- **flag** - type of the new member. Should be one of FF_BYTE..FF_PACKREAL (see above) combined with FF_DATA
- **typeid** - if isStruc(flag) then typeid specifies the structure id for the member if is_off0(flag) then typeid specifies the offset base. if is_strlit(flag) then typeid specifies the string type (STRTYPE_...). if is_stroff(flag) then typeid specifies the structure id if is_enum(flag) then typeid specifies the enum id if is_custom(flags) then typeid specifies the dtid and fid: dtid|(fid<<16) Otherwise typeid should be -1.
- **nbytes** - number of bytes in the new member
- **target** - target address of the offset expr. You may specify it as -1, ida will calculate it itself
- **tdelta** - offset target delta. usually 0
- **reftype** - see REF_... definitions

Returns:
0 - ok, otherwise error code (one of STRUC_ERROR_*)

**Note:** The remaining arguments are allowed only if is_off0(flag) and you want to specify a complex offset expression

## append_func_tail(funcea, ea1, ea2)

Append a function chunk to the function

Parameters:

- **funcea** - any address in the function
- **ea1** - start of function tail
- **ea2** - end of function tail

Returns:
0 if failed, 1 if success

**Note:** If a chunk exists at the specified addresses, it must have exactly the specified boundaries

## set_name(ea, name, flags=0)

Rename an address

Parameters:

- **ea** - linear address
- **name** - new name of address. If name == "", then delete old name
- **flags** - combination of SN_... constants

Returns:
1-ok, 0-failure

## gen_simple_call_chart(outfile, title, flags)

Generate a function call graph GDL file

Parameters:

- **outfile** - output file name. GDL extension will be used
- **title** - graph title
- **flags** - combination of CHART_GEN_GDL, CHART_WINGRAPH, CHART_NOLIBFUNCS

## get_first_hash_key(hash_id)

Get the first key in the hash.

Parameters:

- **hash_id** - The hash ID.

Returns:
the key, 0 otherwise.

## set_color(ea, what, color)

Set item color

Parameters:

- **ea** - address of the item
- **what** - type of the item (one of CIC_* constants)
- **color** - new color code in RGB (hex 0xBBGGRR)

Returns:
success (True or False)

## get_min_spd_ea(func_ea)

Return the address with the minimal spd (stack pointer delta) If there are no SP change points, then return BADADDR.

Parameters:

- **func_ea** - function start

Returns:
BADDADDR - no such function

## get_prev_fchunk(ea)

Get previous function chunk

Parameters:

- **ea** - any address

Returns:
the starting address of the function chunk or BADADDR

**Note:** This function enumerates all chunks of all functions in the database

## get_first_index(tag, array_id)

Get index of the first existing array element.

Parameters:

- **tag** - Tag of array, specifies one of two array types: AR_LONG, AR_STR
- **array_id** - The array ID.

Returns:
-1 if the array is empty, otherwise index of first array element of given type.

## get_next_enum_member(enum_id, value, bmask)

Get next constant in the enum

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant ordinary enums accept only ida_enum.DEFMASK as a bitmask
- **value** - value of the current constant

Returns:
value of a constant with value higher than the specified value. idaapi.BADNODE no such constants exist. All constants are sorted by their values as unsigned longs.

## get_member_size(sid, member_offset)

Get size of a member

Parameters:

- **sid** - 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 2,3,4,5 denote the same structure member.

Returns:
None if bad structure type ID is passed, or no such member in the structure otherwise returns size of the specified member in bytes.

## set_segm_class(ea, segclass)

Change class of the segment

Parameters:

- **ea** - any address in the segment
- **segclass** - new class of the segment

Returns:
success (boolean)

## get_frame_args_size(ea)

Get size of arguments in function frame which are purged upon return

Parameters:

- **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

## get_local_tinfo(ordinal)

Get local type information as 'typeinfo' object

Parameters:

- **ordinal** - slot number (1...NumberOfLocalTypes)

Returns:
None on failure, or (type, fields, name) tuple.

## GetDouble(ea)

Get value of a floating point number (8 bytes) This function assumes number stored using IEEE format and in the same endianness as integers.

Parameters:

- **ea** - linear address

Returns:
double

### add_enum(idx, name, flag)

Add a new enum type

Parameters:

- **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 >= get_enum_qty() or idx == idaapi.BADNODE 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 BADADDR

### save_database(idbname, flags=0)

Save current database to the specified idb file

Parameters:

- **idbname** - name of the idb file. if empty, the current idb file will be used.
- **flags** - combination of ida_loader.DBFL_... bits or 0

### idadir()

Get IDA directory

This function returns the directory where IDA.EXE resides

### get_next_func(ea)

Find next function

Parameters:

- **ea** - any address belonging to the function

Returns:
BADADDR - no more functions otherwise returns the next function start address

### get_prev_hash_key(hash_id, key)

Get the previous key in the hash.

Parameters:

- **hash_id** - The hash ID.
- **key** - The current key.

Returns:
the previous key, 0 otherwise

## EVAL_FAILURE(code)

Check the result of eval_idc() for evaluation failures

Parameters:

- **code** - result of eval_idc()

Returns:
True if there was an evaluation error

## set_array_long(array_id, idx, value)

Sets the long value of an array element.

Parameters:

- **array_id** - The array ID.
- **idx** - Index of an element.
- **value** - 32bit or 64bit value to store in the array

Returns:
1 in case of success, 0 otherwise

## set_segm_addressing(ea, bitness)

Change segment addressing

Parameters:

- **ea** - any address in the segment
- **bitness** - 0: 16bit, 1: 32bit, 2: 64bit

Returns:
success (boolean)

## op_stroff(ea, n, strid, delta)

Convert operand to an offset in a structure

Parameters:

- **ea** - linear address
- **n** - number of operand

- 0 - the first operand
- 1 - the second, third and all other operands
- -1 - all operands

- **strid** - id of a structure type
- **delta** - struct offset delta. usually 0. denotes the difference between the structure base and the pointer into the structure.

## get_enum_member_cmt(const_id, repeatable)

Get comment of a constant

Parameters:

- **const_id** - id of const
- **repeatable** - 0:get regular comment, 1:get repeatable comment

Returns:
comment string

## create_strlit(ea, endea)

Create a string.

This function creates a string (the string type is determined by the value of get_inf_attr(INF_STRTYPE))

Parameters:

- **ea** - linear address
- **endea** - ending address of the string (excluded) if endea == BADADDR, then length of string will be calculated by the kernel

Returns:
1-ok, 0-failure

**Note:** The type of an existing string is returned by get_str_type()

## add_segm_ex(startea, endea, base, use32, align, comb, flags)

Create a new segment

Parameters:

- **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** - 0: 16bit segment, 1: 32bit segment, 2: 64bit segment
- **align** - segment alignment. see below for alignment values
- **comb** - segment combination. see below for combination values.
- **flags** - combination of ADDSEG_... bits

Returns:
0-failed, 1-ok

## set_bpt_cond(ea, cnd, is_lowcnd=0)

Set breakpoint condition

Parameters:

- **ea** - any address in the breakpoint range
- **cnd** - breakpoint condition
- **is_lowcnd** - 0 - regular condition, 1 - low level condition

Returns:
success

## read_dbg_word(ea)

Get value of program word using the debugger memory

Parameters:

- **ea** - linear address

Returns:
The value or None on failure.

## send_dbg_command(cmd)

Sends a command to the debugger module and returns the output string. An exception will be raised if the debugger is not running or the current debugger does not export the 'send_dbg_command' IDC command.

## get_bmask_cmt(enum_id, bmask, repeatable)

Get bitmask comment (only for bitfields)

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant
- **repeatable** - type of comment, 0-regular, 1-repeatable

Returns:
comment attached to bitmask or None

## get_last_member(sid)

Get offset of the last member of a structure

Parameters:

- **sid** - structure type ID

Returns:
-1 if bad structure type ID is passed, ida_idaapi.BADADDR if structure has no members, otherwise returns offset of the last member.

**Notes:**

- IDA allows 'holes' between members of a structure. It treats these 'holes' as unnamed arrays of bytes.
- Union members are, in IDA's internals, located at subsequent byte offsets: member 0 -> offset 0x0, member 1 -> offset 0x1, etc...

## add_func(start, end=4294967295)

Create a function

Parameters:

- **start** - function bounds
- **end** - function bounds

  If the function end address is BADADDR, then IDA will try to determine the function bounds automatically. IDA will define all necessary instructions to determine the function bounds.

Returns:
!=0 - ok

**Note:** an instruction should be present at the start address

## get_module_name(base)

Get process module name

Parameters:

- **base** - the base address of the module

Returns:
required info or None

## set_func_cmt(ea, cmt, repeatable)

Set function comment

Parameters:

- **ea** - any address belonging to the function
- **cmt** - a function comment line
- **repeatable** - 1: get repeatable comment 0: get regular comment

## set_member_name(sid, member_offset, name)

Change structure member name

Parameters:

- **sid** - structure type ID
- **member_offset** - offset of the member
- **name** - new name of the member

Returns:
!= 0 - ok.

## set_bmask_name(enum_id, bmask, name)

Set bitmask name (only for bitfields)

Parameters:

- **enum_id** - id of enum
- **bmask** - bitmask of the constant
- **name** - name of bitmask

Returns:
1-ok, 0-failed

## get_item_size(ea)

Get size of instruction or data item in bytes

Parameters:

- **ea** - linear address

Returns:
1..n

## print_operand(ea, n)

Get operand of an instruction or data

Parameters:

- **ea** - linear address of the item
- **n** - number of operand: 0 - the first operand 1 - the second operand

Returns:
the current text representation of operand or ""

## get_func_off_str(ea)

Convert address to 'funcname+offset' string

Parameters:

- **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

## get_module_size(base)

Get process module size

Parameters:

- **base** - the base address of the module

Returns:
required info or -1

## MakeVar(ea)

Mark the location as "variable"

Parameters:

- **ea** - address to mark

Returns:
None

**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.

## SetType(ea, newtype)

Set type of function/variable

Parameters:

- **ea** - the address of the object
- **newtype** - the type string in C declaration form. Must contain the closing ';' if specified as an empty string, then the item associated with 'ea' will be deleted.

Returns:
1-ok, 0-failed.

## get_frame_size(ea)

Get full size of function frame

Parameters:

- **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

## toggle_bnot(ea, n)

Toggle the bitwise not operator for the operand

Parameters:

- **ea** - linear address
- **n** - number of operand
  - 0 - the first operand
  - 1 - the second, third and all other operands
  - -1 - all operands

## get_prev_index(tag, array_id, idx)

Get index of the previous existing array element.

Parameters:

- **tag** - Tag of array, specifies one of two array types: AR_LONG, AR_STR
- **array_id** - The array ID.
- **idx** - Index of the current element.

Returns:
-1 if no more elements, otherwise returns index of the previous array element of given type.

## get_event_info()

Get debug event info

Returns:
event info: for THREAD_STARTED (thread name) for LIB_UNLOADED (unloaded library name) for INFORMATION (message to display)

## get_next_fchunk(ea)

Get next function chunk

Parameters:

- **ea** - any address

Returns:
the starting address of the next function chunk or BADADDR

**Note:** This function enumerates all chunks of all functions in the database

## update_hidden_range(ea, visible)

Set hidden range state

Parameters:

- **ea** - any address belonging to the hidden range
- **visible** - new state of the range

Returns:
!= 0 - ok

## get_name(ea, gtn_flags=0)

Get name at the specified address

Parameters:

- **ea** - linear address
- **gtn_flags** - how exactly the name should be retrieved. combination of GN_ bits

Returns:
"" - byte has no name

## add_default_til(name)

Load a type library

Parameters:

- **name** - name of type library.

Returns:
1-ok, 0-failed.

## next_head(ea, maxea=4294967295)

Get next defined item (instruction or data) in the program

Parameters:

- **ea** - linear address to start search from
- **maxea** - the search will stop at the address maxea is not included in the search range

Returns:
BADADDR - no (more) defined items

## get_str_type(ea)

Get string type

Parameters:

- **ea** - linear address

Returns:
One of STRTYPE_... constants

## set_hash_string(hash_id, key, value)

Sets the string value of a hash element.

Parameters:

- **hash_id** - The hash ID.
- **key** - Key of an element.
- **value** - string value to store in the hash

Returns:
1 in case of success, 0 otherwise

## get_operand_type(ea, n)

Get type of instruction operand

Parameters:

- **ea** - linear address of instruction
- **n** - number of operand: 0 - the first operand 1 - the second operand

Returns:
any of o_* constants or -1 on error

## set_fchunk_attr(ea, attr, value)

Set a function chunk attribute

Parameters:

- **ea** - any address in the chunk
- **attr** - only FUNCATTR_START, FUNCATTR_END, FUNCATTR_OWNER
- **value** - desired value

Returns:
0 if failed, 1 if success

## get_func_name(ea)

Retrieve function name

Parameters:

- **ea** - any address belonging to the function

Returns:
null string - function doesn't exist otherwise returns function name

## get_segm_by_sel(base)

Get segment by segment base

Parameters:

- **base** - segment base paragraph or selector

Returns:
linear address of the start of the segment or BADADDR if no such segment

## get_type(ea)

Get type of function/variable

Parameters:

- **ea** - the address of the object

Returns:
type string or None if failed

## add_bpt(ea, size=0, bpttype=12)

Add a new breakpoint

Parameters:

- **ea** - any address in the process memory space:
- **size** - size of the breakpoint (irrelevant for software breakpoints):
- **bpttype** - type of the breakpoint (one of BPT_... constants)

Returns:
success

**Note:** Only one breakpoint can exist at a given address.

## can_exc_continue()

Can it continue after EXCEPTION event?

Returns:
boolean

## force_bl_jump(ea)

Some ARM compilers in Thumb mode use BL (branch-and-link) instead of B (branch) for long jumps, since BL has more range. By default, IDA tries to determine if BL is a jump or a call. You can override IDA's decision using commands in Edit/Other menu (Force BL call/Force BL jump) or the following two functions.

Force BL instruction to be a jump

Parameters:

- **ea** - address of the BL instruction

Returns:
1-ok, 0-failed

## get_bpt_attr(ea, bptattr)

Get the characteristics of a breakpoint

Parameters:

- **ea** - any address in the breakpoint range
- **bptattr** - the desired attribute code, one of BPTATTR_... constants

Returns:
the desired attribute value or -1

## process_config_line(directive)

Parse one or more ida.cfg config directives

Parameters:

- **directive** - directives to process, for example: PACK_DATABASE=2

**Note:** If the directives are erroneous, a fatal error will be generated. The settings are permanent: effective for the current session and the next ones