IDA Pro 4.9

What's new in this release: overview of major features.

Debugger for WindowsCE based ARM Devices.

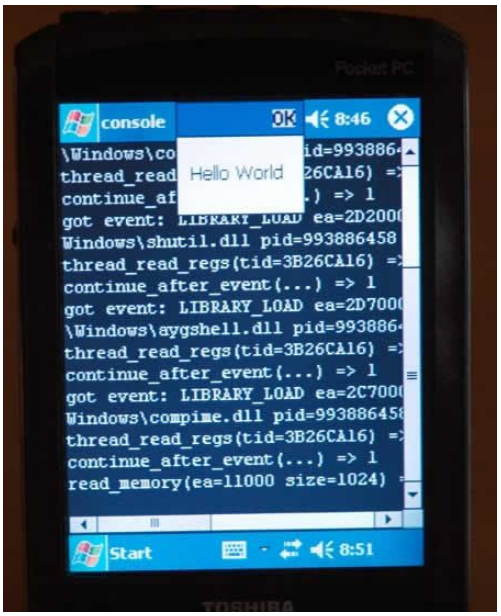

Illustration 1: The IDA Pro Windows CE remote debugger.

The Windows 32-bit versions of IDA now offer a Windows CE ARM remote debugger. At this point, only those versions support the debugger since it relies on the ActiveSync API which is available only on this platform. That feature is available for free to all existing IDA Pro customers with an active license. It is available as a separate additional module to new customers.

The debugger supports the Intel xScale hardware data breakpoint capabilities and uses software breakpoints otherwise. Please note the following about Windows CE/ARM breakpoints: since “single step” is not supported by the generic ARM architecture, it is emulated by software breakpoints. This works in most cases but single stepping in system areas and coredll.dll is not possible. Since a breakpoint in coredll.dll or the system area would be visible to all processes but would only be handled in the context of the debugged process, it would freeze the system and require a hard reset. The IDA Pro WinCE debugger attempts to prevent that situation by refusing any write into coredll.dll or to any address >= 0x80000000.

Since it is impossible to put a breakpoint in the forbidden areas, the chances of breaking into or pausing a running application when it is in the kernel or in coredll.dll are slim. Currently IDA checks if the PC of the process is in the allowed range and, if it is the case, creates a breakpoint to pause the process. Otherwise the @hlpHelpProcessPause[pause process] command fails.

The debugger uses the kernel memory tables to find out the memory layout and assumes that the kernel memory layout (addresses greater than 0x80000000) never changes, so this information is gathered only once at the beginning of the debugging session.

There is no need to start the debugger server manually. IDA will use the existing ActiveSync connnection to communicate with the device and download the debugger server to it. If

the input file is not present on the device, it will also be copied to the device.

On Windows CE the running process is mapped to two different areas: to the process slot and slot #0. IDA Pro uses slot#0 as the main image of the process and does not know about this memory mapping.

Better handling of PDB files

IDA uses a new fully automated method of downloading PDB files from the Microsoft Symbol Server. The web browser is not involved in the process anymore and one does not need to unpack the downloaded file manually. One single click downloads the file, unpacks it and applies it to the disassembly.

Debugger

Apart from WindowsCE ARM debugger, there are other improvements as well. Three of them are important enough to be noted here:

- plugin writers will appreciate the improved speed of the trace mechanism.
- We provide the source code of a sample Universal PE Unpacker debugger plugin module to demonstrate this improvement in a real life situation.
- the user may decide at the last moment how to proceed after an exception and override the default settings. this improvement combined with the special handling of the popf instruction makes it possible to debug programs using EXCEPTION_BREAKPOINT and EXCEPTION_SINGLE_STEP for their own uses.

Stable SDK

We recognize that a growing number of IDA plugins have proven very useful and will try to make plugin user's lives a bit easier: starting from this 4,9 release we freeze the IDA API: **well written plugins, even in the binary form, will be backward compatible**. The IDA Pro API will continue to be improved but will attempt not modify the interface of the existing functions.

Updated demangler

The built-in demangler has been improved to support GNU v3.x demangling scheme and new Microsoft compilers. No need to connect an external demangler to see nice names in the disassembly listing, everything is automatic as usual.

Support for MIPS Architecture Release 2

We have added support for the MIPS32 architecture release 2.

Support of large databases

Given the rapid increase of application sizes, we got rid of the limit on the size of the Btree database used by IDA: we expect the new database engine to be capabable of handling very large binaries, consisting of up to 100MB of code.

3-byte items

Microcontroller users will appreciate this: IDA Pro now supports 3-byte items. You can even configure the order of byte in the listing. Currently available for Motorola 6812, other processors modules will be updated progressively.

What's New in this release: detailed list of features

PROCESSOR MODULES

- **MIPS: MIPS32 release 2 and MIPS5 instructions are supported**
- **New processor: Atmel OAK DSP (courtesy of Ivan Litvin)**
- 6812: IDA does not create an xref for (L)BRN instructions anymore; BRN and CPS instruction are replaced by SKIP1/SKIP2 pseudoinstructions
- ARM: better handling of glue code
- ARM: IDA knows that STR Rn, [SP,-4]! changes the stack pointer
- ARM: jump table recognition has been improved
- ARM: tail call using the 'B' instruction is recognized if it is preceded with LDM SP!,...
- ARM: updated WinCE coredll.ids and added undocumented functions
- ARM: MOV PC,... is recognized as an instruction which spoils register values
- C166: it is possible to use zero offsets, enum members and other representations for indirect addressing like [rX]. In other words, [rX] can be replaced by [rX+symbol_denoting_zero] using appropriate commands in IDA
- IBM PC: parameter comments are applied to all instructions, not only on restrict set of instructions; parameter location were not always correct for borland and watcom __fastcall calling conventions
- IBM PC: Watcom & GNU __fastcall calling convention is now supported.
- IBM PC: SEH_prolog function is now recognized and handled by IDA.
- M68K: addi.l #imm,... automatically selects the number sign
- MIPS: default MIPS processor is simple MIPS, not R5900 anymore.
- PPC: the processor module can display up to 4 operands per instruction

File Formats

- **TDS: IDA knows about Borland's TDS debug information files**
- **PDB: IDA uses a new fully automated method of downloading PDB files from the Microsoft Symbol Server**
- COFF: gcc link directives are recognized
- COFF: files with no sections were considered as incorrect files; now ida accepts them (but this might lead to misdetections)
- COFF: IBM PC: ida sets up the compiler by demangling all names and selecting the compiler which gives most demangled names
- ELF: added support for ARM LOPROC symbol type
- ELF: communal variables in Kylix files are loaded into the database
- ELF: IDA knows about DT_PLTGOT record
- ELF: much better patching for PIC mode; additional loading options flag in added to the user interface
- ELF: support for new ABI v3.0 has been added
- ELF: try to load unknown elf images upon user request
- ELF: IDA tries to detect the GNUC++ compiler based on the symbols in the input file
- EPOC: default string style is unicode
- EPOC: thumb mode rom images are supported
- PE: DllEntryPoint function prototype has been added
- PE unpacker plugin: this plugin uses the debugger to let the program to unpack itself in the memory and as soon as the execution reaches the original entry point, it suspends the program; the user then may take a memory snapshot

KERNEL

- **the demangler has been improved to support GNU v3.x demangling scheme and new Microsoft compilers**
- **btree databases bigger than 512MB are supported**
- **IDA supports 3-byte items. The user can specify the order of bytes using the TRIBYTE_ORDER ida.cfg parameter**
- automatically detect Visual C++ object files and set the target compiler
- 'force zero offset' applied to a structure offset operand which is represented as the structure size appends the first structure field to the operand representation
- array indexes are displayed for the arrays of terse structures
- better treatment of wrongly created function tails: a wrong tail might have led to other wrong tails, so after converting it to a function we have to recreate all other function tails

- it is possible to rename a register to itself (useful to create register comments)
- while forming a function prototype if undecorating the name fails, try to demangle it and get the bare name
- better analysis of unicode strings: ida creates them only on 4byte boundaries
- more strings are discovered and created during the analysis
- when the default string type is unicode, IDA checks for C strings as well
- FLAIR: REL_I386_CLR_TOKEN relocation type is supported
- FLIRT: Visual C++ signatures has been updated
- IDA creates thunk functions for ibm pc more aggressively
- IDA doesn't display anonymous structures in the structure selection lists (for new databases)
- IDA tries to apply type information for names like j_func and func_0
- initialized union instances are allowed; ida will display the union using the definition of the first union field
- ntddk.til has been updated; wdm.til has been added

IDC & SDK

- IDC: error message about calling an undefined function includes the function name
- IDC: GetString() function has been added
- IDC: MakeUnknown() function to undefine a range of addresses has been added
- IDC: isEnabled() macro allows to check if an address is valid
- SDK: IDA API has been frozen for future binary compatibility
- SDK: add_regvar() modifies existing regvar definitions to define a new variable which overlaps the existing variables
- SDK: add_sourcefile() function does not fail if there already was a source file defined at the specified range; in this case it will delete or modify the old definitions to make a hole big enough to hold the new file
- SDK: add_stkvar2() function to add stack variables from plugins; tds plugin uses it
- SDK: added a flag to allow the use of constructs not supported by the target assembler. It is INFFL_ALLASM bit in inf.s_genflags. The inf.use_allasm() function checks this bit
- SDK: areacb_t::make_hole() function can be used to make holes in area definitions
- SDK: AUTOHIDE NONE and similar keywords in the dialog box messages can appear not only in the format string but also in the final string as well
- SDK: closing_comment() to get the comment closing sequence
- SDK: debugger structures are aligned at 4 bytes to avoid problems on ARM processor
- SDK: is_ret_insn() function and callback are added
- SDK: make_visible_name() has additional argument - output buffer size

- SDK: new function calc_bare_name() to get the smallest possible form of a name (try to undecorate and demangle)
- SDK: new function parse_types(); IDC: new function ParseTypes()
- SDK: new function: get_flags_ex(). The new function get_flags_novalue() built on top of get_flags_ex() does not return the MS_VAL and FF_IVL fields of the flags and therefore is much faster for remote debugging; it is not exported yet (todo later!) but the kernel uses it during the segment deletion which improves the speed
- SDK: ph.guess_memory_model is replaced by ph.setup_til
- SDK: rotate_left() function is exported
- SDK: SetFlags, SetFlbits, ClrFlbits() functions do not modify the MS_VAL and FF_IVL flag fields; uFlag global variable does not contain these fields; the kernel updates the uFlag automatically (if the modified address is equal to cmd.ea), so there is no need to update uFlag manually in the emulator function of processor modules
- SDK: setup_selector() allocates a new selector for values not fitting in 16 bits only for IBM PC. Other processors will use 32-bit segment bases
- SDK: the processor name is stored in the debugger description structure to allow instant debugging for processors different from IBM PC
- SDK: added new data type: 3byte; doData() function is removed because it can be replaced by do_data_ex()
- SDK: btoa..() functions accept a buffer for the answer; atoa() function is obsolete and removed (use ea2str())
- SDK: close_chooser() to close open non-modal list views
- SDK: dbg->stopped_at_debug_event() does take require 'event' as the parameter;
- SDK: elnum_t is replaced with int
- SDK: func_does_return() is added; it is better to use this function instead of examining FUNC_NORET flag directly
- SDK: get_name_expr() accepts a buffer for the answer and returns the answer length; new function: out_name_expr() is easier to use in the processor modules; append_disp() is replaced with print_disp() which does not append but simply uses the given buffer and returns the answer length
- SDK: IDC compile/run functions return the error message in the specified buffer (before it was a in a static storage); the return value is a bool meaning success of the operation
- SDK: interface to netnode functions returning big objects has been changed. Now these functions put the results into the specified bugger. This is a big change in IDA API leading to the modifications of many functions. This change is a small step in the direction of multithreaded model.
- SDK: is_alloca_probe notification code has been added
- SDK: lexical analyzer is thread-safe and can be called from several threads for different inputs
- SDK: new function qerrstr() and qerrcode()
- SDK: nexthat, prevthat functions accept 'user_data' parameter for thread-safe handling

- SDK: now the memory config and contents are not automatically refreshed at each debug notification; the plugin must call invalidate_dbgmem_config() and/or invalidate_dbgmem_contents() to get the current view
- SDK: PLUGIN_PROC and PLUGIN_FIX flags are added; plugin management is moved to the kernel
- SDK: refresh_navband() to refresh the navigation band
- SDK: standard file functions like fopen() are not visible by default - use qfopen() and similar functions; the standard functions can be enabled by the USE_STANDARD_FILE_FUNCTIONS preprocessor symbol
- SDK: strarray() accepts a buffer for the answer; qstrerror() accepts a buffer for the answer; ivalue1,2,3 functions are deleted
- SDK: take_memory_snapshot() function is added
- SDK: term_database() function is added
- SDK: zip_inflate/zip_deflate() functions accept 'user_data' parameter
- SDK: added get_reg_name() to get the name of the specified register
- SDK: bring_debugger_to_front() is added
- SDK: dbg_trace can be used to filter trace events
- SDK: find_text() function is exported
- SDK: get_first_module(), get_next_module() debugger functions are added
- SDK: new dbg_process_attach & dbg_process_detach notifications are generated in all cases ([request_]attach|detach_process() or not)
- SDK: new function add_menu_item(), del_menu_item() (only in the gui version for the moment)
- SDK: the asynchronous start_process() command now terminates as soon as the process is started and generates a 'dbg_process_start' notification
- SDK: ua_emu() and ua_out() functions are not in IDA API anymore
- SDK: user-defined form buttons can be configured in the dialog definition

User Interface

- it is possible to add/delete exception codes from the user interface
- a disassembly/hex view can be attached to a hex/disassembly view or to a register value when debugging (through the 'Synchronize with' command in the view's popup menu)
- gui: register variable definition ranges are displayed in the hints
- gui: text entry dialog boxes are resizeable
- gui: Imports and Exports windows now automatically select the nearest entry just below the current address if any
- maximal size of instant idc script executed by Shift-F2 is increased to 16KB

- 'jump to file offset' treats the input as a hex number by default
- reaction time to cancelling a file load has been much improved
- ui: 'make array' command is faster because it does not calculate additional information which was displayed on the dialog box
- ui: it is possible to jump to the structure definition by pressing Enter on the "size structname" expression in the disassembly view
- ui: register variable definitions are printed at the beginning of the register definition area (was at the beginning of the function)
- ui: editing a standard structure makes it non-standard
- ui: if IDA detects that the new name entered by the user is already used in the program, it proposes to set 'create anyway' flag

Debugger

- remote debugging: debugger module for Windows CE running on ARM
- the decision whether to mask/unmask an exception can be taken upon resuming program execution
- 'Run to cursor' now continues the execution if used at IP (useful to iterate in loops)
- diminished the wait time of stopping a task which does not stop reason from 4 seconds to 3 seconds
- added 'Enable breakpoints' and 'Disable breakpoints' commands in popup menu of the Breakpoints list - these commands also accept multiple selection
- ida knows about popf instructions and that they might set the trace bit; if the currently stepped instruction sets the trace bit, then the resulting exception is reported back to the application - this allows to single step such instruction with preserving all application functionality; the user may also cancel the 'continue execution' command
- unexpected trace bit exceptions are passed to the application first; this allows to debug a program using exception 0x80000004 internally
- ida proposes to copy the local input file to the remote computer if it is missing

Bugfixes

- 'make array' would ignore instructions when calculating the maximal array size
- 'wait' box might stay on the screen after a long operation
- a non-modal chooser was stealing the focus of a (modal) dialog box if opened from inside this one
- ARM module was creating sp based stack variables too early - this could lead to superfluous stack variables in some cases
- arrows referencing addresses outside of the screen were sometimes displayed

- Borland RTTI plugin was crashing at the dialog box
- c166: buffer for the memory mappings display was too short for 64bit 16 mappings
- checking a manual operand might fail
- corrected description of askbuttons() function
- database garbage collection errors would lead to immediate exit (the error message was printed to stdout); now ida reports about them using a message box
- debugger: an error message was displayed when cancelling the 'Process options...' dialog box
- debugger: IDA was sometimes unable to obtain the debug privilege in order to debug a system process
- debugger: register values were not properly displayed with large fonts
- debugger: the 'Command "BreakpointToggle" failed' message was displayed when cancelling the 'Breakpoint settings' dialog box
- debugger: tracing could cause infinite loops once the trace buffer was full
- debugger: tracing was erroneously disabled if it was started inside a debugger segment, the 'Trace over debugger segments' option was enabled, and the debugger segment was calling code inside the database segment
- disassembly and hex views were refreshed too often, causing slow browsing
- dsp56k was not disassembling some movem instructions
- editing anterior/posterior comments would remove the last empty line
- elf loader would skip sections with wrong type even if the user wanted to load them in the manual mode
- EPOC ROM images were truncated by the size of the ROM header
- error message about using register names while the debugger was not active would display garbage instead of the register name
- FR call instruction was not disassembled correctly
- IBM PC 'assembly' command would not accept labels in instruction operands
- IBM PC: movhpd, movldp, movntdq instruction do not allow register-to-register encodings but IDA was disassembling them
- IDA could automatically destroy an instruction if a data item overlapping it is defined
- IDA could hang trying to display very long string result in the calculator
- IDA was proposing the default array size of one for uninitialized data if the debugger was active
- IDA was using the debuggee command line switches when the debugger was started with the -r switch
- IDC DelConst(Ex) function was not available
- IDC: empty statement after 'else' would lead to incorrect execution
- if the entry point ends up having a name different from "start" then its name in the entry

point list would be missing

- if the last segment in the program was close to the memory top, the binary search could fail
- if the plugin had PLUGIN_UNL then its init() would not be called when the plugin was invoked the second time
- in some very rare cases the list of names of a debugged program might get corrupted
- information about original bytes patched when the debugger was active would be lost
- it was impossible to assign Fx keys as IDC hotkeys
- it was impossible to use a negated symbol constant
- it was not possible to debug 32-bit applications under Windows64
- java: floating point numbers were not displayed correctly
- loading old databases under linux for big endian processors might lead to problems due to unexpected saving of database at the load time
- mc68k was always using offset base 0 for lea/pea instructions
- mc68k: pea/rts might create wrong cref; execution flow was always stopping at it
- MIPS module was not disassembling some MIPS32 opcodes
- multi monitor systems: IDA was positioning windows on the main monitor regardless where they were before
- mysterious access violations when renaming a stack variable are fixed
- only the first hardware breakpoint was working correctly
- PE files with unaligned section pointers would be loaded incorrectly
- PE modules were not accepted for imported function names even if they contained exported names
- PIC config files were wrong
- renaming a stack variable of a function with a prototype might lead to wrong prototype and subsequent crash
- repeatable comments for structure offsets at 0 were not printed if the 0 constant was not explicitly present in the instruction
- rotate_left() all 32-bits on 32-bit ida was not working
- some functions would not be destroyed at the end of a debugging session
- some imported functions would not have correct attributes because the imports segment was created after the attempt to set their types; now the type libraries for imported modules are loaded after creation of import segments (it concerns mainly pe files)
- some processor modules were not taking into account the possibility of custom instruction types
- tab order in 'Breakpoint settings' dialog box was not properly defined
- text version: warning messages in the batch mode were displayed as garbage in the messages window

- the 'Clear trace' command from the 'Edit' menu in the 'Trace window' was not properly refreshing the window if no trace line was selected
- the definition of a structure member of a unicode string type had wrong length
- the pdb plugin was not unloaded after the use despite of the 'PLUGIN_UNL' flag
- the segment register window by default had too narrow columns in the 64-bit version
- the stack trace window was not restored at the beginning of a debugging session
- the status line was not immediately redrawn after a function renaming
- the system menu icon of a maximized MDI window was continuously appearing/disappearing when starting various commands such as 'D'
- the type of structures with floating pointer members was not correctly guessed/determined
- TMS320C54 and TMS320C55 files could hang when loading wrong input file
- using floating point constants in IDC might lead to runtime errors
- VS.NET: double floating point constants were displayed incorrectly in the 64bit version
- when removing a function tail, the stack change points of the tail were not removed
- XlatAsciiOutput was not working correctly in all cases
- 'jump to end' in the structures window was jumping to the last field of the last structure instead of positioning the cursor at the structure end
- bin_search() was not working properly when the debugger was active; take_memory_snapshot() recalculates the tail flags (required to display properly data arrays after a snapshot)
- dragging toolbars out of debug registers window would change its size unnecessarily
- ida could erroneously report a mismatch between the database and the executable file if the input file name was changed after initializing the debugger; now we pass the input file name to the 'start_process' function and to 'get_process_info' function
- ida could hang if the structure definitions were corrupted
- IDA was always proposing enum_1 as the default enumeration name
- it was possible to create mutually nested structure types
- it was possible to use a name starting with a digit using the MakeName() IDC function
- modifying the stack frame of a deleted function could lead to a crash
- on Windows, breakpoints occuring while another thread is terminating could cause the debugger to freeze
- possible endless loop in get_bmask_enum()
- resolve_typedef() result was not always checked against NULL
- trying to fix the suspended threads problem when creating a breakpoint in shared code of a multi-threaded application

DataRescue SA/NV
40 Blvd Piercot
4000 Liège
Belgium
t - +32-4-3446510
f - +32-4-3446514
info@datarescue.com