Subject: Do we have support for Windows on ARM64?
Posted by Tom1 on Thu, 09 Jan 2025 09:38:04 GMT
View Forum Message <> Reply to Message

Hi,

I noticed that new Windows laptops have appeared on the market with ARM64 architecture processors (e.g. Qualcomm Snapdragon X Plus X1P-42-100). Is U++ ready for this?

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Tue, 04 Feb 2025 07:35:10 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 09 January 2025 10:38Hi,

I noticed that new Windows laptops have appeared on the market with ARM64 architecture processors (e.g. Qualcomm Snapdragon X Plus X1P-42-100). Is U++ ready for this?

Tom

No. But it will probably not be an issue. CLANG-MINGW seems to support these for a long time. And ARM is not a problem - we run on MacOS M1 and Rapsberry PI without issue. Also windows emulate x86.

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Tue, 04 Feb 2025 11:09:25 GMT
View Forum Message <> Reply to Message

Thanks Mirek,

I guess this support will need to be sorted out within a couple of years, i.e. just before clients start to wonder why our apps do not install/run on their brand new Windows laptops.

Visual Studio 2022 (x64) can target ARM64 too, so could we be lucky enough to have it working by just adding a suitable BM?

Anyway, I do not have such a laptop, so I will not be able to test this yet.

Best regards,

Tom

EDIT: OK, they may run in emulation, but I mean natively...

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Tue, 04 Feb 2025 11:44:34 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 04 February 2025 12:09Thanks Mirek,

I guess this support will need to be sorted out within a couple of years, i.e. just before clients start to wonder why our apps do not install/run on their brand new Windows laptops.

Visual Studio 2022 (x64) can target ARM64 too, so could we be lucky enough to have it working by just adding a suitable BM?

Yes. CLANG Toolchain we use even supports it already...

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Sun, 30 Nov 2025 15:18:30 GMT
View Forum Message <> Reply to Message

Hi,

I discovered a nice black friday deal and finally decided to order a laptop with snapdragon and Windows 11 ARM. Any suggestions on how to get Ultimate++ up and running natively on this platform? I guess I will get the laptop in just a few days...

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Sun, 30 Nov 2025 19:18:40 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sun, 30 November 2025 16:18Hi,

I discovered a nice black friday deal and finally decided to order a laptop with snapdragon and Windows 11 ARM. Any suggestions on how to get Ultimate++ up and running natively on this platform? I guess I will get the laptop in just a few days...

Best regards,

Tom

From what I know, and I might be wrong, start with running U++ non-natively. Then fiddle with compiler/build methods.

https://github.com/mstorsjo/llvm-mingw

which we use supports ARM (but I tend to delete it from the archive for now).

IMO, over speaking should not be all that hard.

Keep us updated, this is interesting.

Mirek

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Fri, 05 Dec 2025 09:53:55 GMT
View Forum Message <> Reply to Message

Hi Mirek,

OK, now I have the new laptop with Windows 11 ARM64 up and running. The nightly upp installed easily and runs in the Prism emulation (embedded in Windows 11 ARM by default) very well and does not feel sluggish at all.

The next step I took was downloading ARM64 version of the llvm tools from:

https://github.com/mstorsjo/llvm-mingw/releases/download/20251202/llvm-mingw-20251202-ucrt-aarch64.zip

Then I extracted the contents and replaced the entire C:\upp\bin\clang -directory with the contents from this archive. I opened theide and the UWord from examples. As I tried to compile it, the compiler first complained about the 'Common option -mpopcnt' which is not supported by aarch64. So I removed that option.

I also changed the PATH executable directory: C:\upp\bin\clang\x86_64-w64-mingw32\bin path to point at C:\upp\bin\clang\aarch64-w64-mingw32\bin instead.

I made those changes to a new build method 'CLANGARM64' using 'CLANG' builder. I know this is not complete and probably needs more changes for at least the other paths, includes and libraries, depending on the program being built.

Next issue was found on Core\Debug.cpp around line 270. The 'qword esp = ep->ContextRecord->Rsp;' and 'dword esp = ep->ContextRecord->Esp;' do not exist here. I do not know how to really fix this, but as a quick and dirty workaround I decided to replace that code with just 'qword esp = 0;' and consequently the UWord compiled, linked and runs OK!

So, I guess all this is doable, but requires some tuning in TheIDE so that everything runs natively out of the box. Can you take a more professional look at this based on this description?

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Fri, 05 Dec 2025 10:06:11 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 05 December 2025 10:53Hi Mirek,

OK, now I have the new laptop with Windows 11 ARM64 up and running. The nightly upp installed easily and runs in the Prism emulation (embedded in Windows 11 ARM by default) very well and does not feel sluggish at all.

The next step I took was downloading ARM64 version of the llvm tools from:

https://github.com/mstorsjo/llvm-mingw/releases/download/202
51202/llvm-mingw-20251202-ucrt-aarch64.zip

Then I extracted the contents and replaced the entire C:\upp\bin\clang -directory with the contents from this archive. I opened theide and the UWord from examples. As I tried to compile it, the compiler first complained about the 'Common option -mpopcnt' which is not supported by aarch64. So I removed that option.

I also changed the PATH executable directory: C:\upp\bin\clang\x86_64-w64-mingw32\bin path to point at C:\upp\bin\clang\aarch64-w64-mingw32\bin instead.

I made those changes to a new build method 'CLANGARM64' using 'CLANG' builder. I know this is not complete and probably needs more changes for at least the other paths, includes and libraries, depending on the program being built.

Next issue was found on Core\Debug.cpp around line 270.

Slightly off topic, you do know about "Misc/Copy current position" ?

Quote:
The 'qword esp = ep->ContextRecord->Rsp;' and 'dword esp = ep->ContextRecord->Esp;' do not exist here. I do not know how to really fix this, but as a quick and dirty workaround I decided to replace that code with just 'qword esp = 0;' and consequently the UWord compiled, linked and runs OK!

That whole thing can go away I guess, it is fossil...

Quote:
So, I guess all this is doable, but requires some tuning in TheIDE so that everything runs natively out of the box. Can you take a more professional look at this based on this description?

I guess there 2 hard problems ahead:

- libraries that we ship with (esp OpenSSL). that is even more sad as updating libraries takes awful amount of time, now there would be another architecture...
- debugging. That might possibly work (test please), but at minimum the dissassembler only supports x86.

Mirek

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Fri, 05 Dec 2025 11:34:05 GMT
View Forum Message <> Reply to Message

Hi,

Quote:
Slightly off topic, you do know about "Misc/Copy current position" ?

I have no idea what this is... should I??

Quote:
I guess there 2 hard problems ahead:

- libraries that we ship with (esp OpenSSL). that is even more sad as updating libraries takes awful amount of time, now there would be another architecture...
- debugging. That might possibly work (test please), but at minimum the dissassembler only supports x86.

- I agree, libraries cause a constantly increasing amount of work and adding another architecture is again a multiplier to this work.

- Debugging does not work here. Tried to run UWord in debugger and TheIDE reported: "Error! SetThreadContext failed. The thread context could not be updated because this has been restricted for the process. (1660)"

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Fri, 05 Dec 2025 11:38:24 GMT
View Forum Message <> Reply to Message

Now tried to "Setup > Upgrade TheIDE" in order to get native TheIDE if that has any effect for debugging. However, I was unable to compile it:
Saving
----- ide/Common (GUI CLANG BLITZ WIN32) (1 / 40)
BLITZ: ComDlg.cpp Module.cpp Util.cpp
----- ide/Core (GUI CLANG BLITZ WIN32) (2 / 40)
BLITZ: Ide.cpp Cache.cpp Core.cpp Util.cpp Builder.cpp PPinfo.cpp Assembly.cpp Package.cpp Workspace.cpp usc.cpp BinObj.cpp Signature.cpp Host
.cpp Logger.cpp
----- ide/LayDes (GUI CLANG BLITZ WIN32) (3 / 40)
BLITZ: sdiff.cpp laylib.cpp layusc.cpp property.cpp textprop.cpp fontprop.cpp propane.cpp item.cpp layout.cpp visgen.cpp laydes.cpp layfile.cp
p laywin.cpp
----- ide/Builders (GUI CLANG WIN32) (4 / 40)
Precompiling header: Builders.h
ide/Common: 3 file(s) built in (0:01.51), 505 msecs / file
ide/Core: 14 file(s) built in (0:02.48), 177 msecs / file
ide/LayDes: 13 file(s) built in (0:03.67), 282 msecs / file
CppBuilder.cpp
MakeFile.cpp
CCJ.cpp
GccBuilder.cpp
MscBuilder.cpp
JavaBuilder.cpp
ScriptBuilder.cpp
Cocoa.cpp
AndroidProject.cpp
AndroidApplicationMakeFile.cpp
AndroidMakeFile.cpp
AndroidModuleMakeFile.cpp
AndroidBuilder.cpp
AndroidBuilderCommands.cpp
AndroidBuilderUtils.cpp
AndroidModuleMakeFileBuilder.cpp
Blitz.cpp
Build.cpp
Install.cpp
BuilderUtils.cpp
ide/Builders: 21 file(s) built in (0:07.57), 360 msecs / file
Creating library...
C:/upp/bin/clang/bin/llvm-ar.exe: warning: creating C:/upp/out/uppsrc/ide/Builders/CLANGARM64.Gui\Builders.a
C:/upp/out/uppsrc/ide/Builders/CLANGARM64.Gui\Builders.a (1124408 B) created in (0:00.07)
----- ide/Debuggers (GUI CLANG BLITZ WIN32) (5 / 40)
BLITZ: Terminal.cpp Disas.cpp GdbCmd.cpp GdbData.cpp Gdb.cpp GdbMem.cpp GdbUtils.cpp

```
Debug.cpp Mem.cpp Sym.cpp Exp.cpp PrettyUpp.cpp PrettyStd
   .cpp Scripts.cpp Pretty.cpp Visualise.cpp Data.cpp Tree.cpp Stack.cpp Code.cpp
Cpu.cpp
Pdb.cpp
----- ide/Browser ( GUI CLANG BLITZ WIN32 ) (6 / 40)
BLITZ: Util.cpp TopicBase.cpp File.cpp Topic.cpp Template.cpp Link.cpp TopicWin.cpp Move.cpp
CodeRef.cpp TopicI.cpp
----- CodeEditor ( GUI CLANG BLITZ WIN32 ) (7 / 40)
BLITZ: Register.cpp HighlightOut.cpp Syntax.cpp Style.cpp RegisterSyntax.cpp CSyntax.cpp
CInit.cpp CHighlight.cpp CLogic.cpp DiffSyntax.cpp Ta
   gSyntax.cpp PythonSyntax.cpp LogSyntax.cpp EditorBar.cpp FindReplace.cpp Lang.cpp
CodeEditor.cpp
----- CtrlLib ( GUI CLANG BLITZ WIN32 ) (8 / 40)
BLITZ: CtrlLibInit.cpp SmartText.cpp LabelBase.cpp DisplayPopup.cpp Button.cpp Switch.cpp
VirtualButtons.cpp EditField.cpp Text.cpp LineEdit.c
   pp DocEdit.cpp ScrollBar.cpp HeaderCtrl.cpp ArrayCtrl.cpp MultiButton.cpp PopupTable.cpp
PopUpList.cpp DropList.cpp DropChoice.cpp Static.
   cpp Splitter.cpp FrameSplitter.cpp SliderCtrl.cpp ColumnList.cpp Progress.cpp AKeys.cpp
RichTextView.cpp Prompt.cpp Help.cpp DateTimeCtrl.
   cpp SuggestCtrl.cpp Bar.cpp ToolButton.cpp ToolBar.cpp ToolTip.cpp StatusBar.cpp
TabCtrl.cpp TreeCtrl.cpp DropTree.cpp DlgColor.cpp ColorP
   opup.cpp ColorPusher.cpp FileList.cpp FileSel.cpp FileSelUtil.cpp PrinterJob.cpp Windows.cpp
Win32.cpp Gtk.cpp TrayIconWin32.cpp TrayIconX
   11.cpp TrayIconGtk.cpp Update.cpp CtrlUtil.cpp BeginnerInfo.cpp LNGCtrl.cpp Ch.cpp
ChGtk3.cpp ChCoco.cpp
MenuItem.cpp
MenuBar.cpp
ChWin32.cpp
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:1:26: error: no member named 'Rax' in
'_CONTEXT'
   1 | CPU_REG(CV_AMD64_AL    , Rax, REG_L, NULL, 0)
     | ~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
  19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
     |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:2:26: error: no member named 'Rcx' in
'_CONTEXT'
   2 | CPU_REG(CV_AMD64_CL    , Rcx, REG_L, NULL, 0)
     | ~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
  19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
     |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:3:26: error: no member named 'Rdx' in
```

```
'_CONTEXT'
    3 | CPU_REG(CV_AMD64_DL    , Rdx, REG_L, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:4:26: error: no member named 'Rbx' in
'_CONTEXT'
    4 | CPU_REG(CV_AMD64_BL    , Rbx, REG_L, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:5:26: error: no member named 'Rax' in
'_CONTEXT'
    5 | CPU_REG(CV_AMD64_AH    , Rax, REG_H, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:6:26: error: no member named 'Rcx' in
'_CONTEXT'
    6 | CPU_REG(CV_AMD64_CH    , Rcx, REG_H, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:7:26: error: no member named 'Rdx' in
'_CONTEXT'
    7 | CPU_REG(CV_AMD64_DH    , Rdx, REG_H, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:8:26: error: no member named 'Rbx' in
'_CONTEXT'
    8 | CPU_REG(CV_AMD64_BH    , Rbx, REG_H, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
```

```
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:9:26: error: no member named 'Rax' in
'_CONTEXT'
    9 | CPU_REG(CV_AMD64_AX    , Rax, REG_X, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:10:26: error: no member named 'Rcx' in
'_CONTEXT'
   10 | CPU_REG(CV_AMD64_CX    , Rcx, REG_X, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:11:26: error: no member named 'Rdx' in
'_CONTEXT'
   11 | CPU_REG(CV_AMD64_DX    , Rdx, REG_X, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:12:26: error: no member named 'Rbx' in
'_CONTEXT'
   12 | CPU_REG(CV_AMD64_BX    , Rbx, REG_X, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                      ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:14:26: error: no member named 'Rsp' in
'_CONTEXT'
   14 | CPU_REG(CV_AMD64_SP    , Rsp, REG_X, NULL, 0)
      | ~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
```

```
    |                                                  ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:15:26: error: no member named 'Rbp' in
'_CONTEXT'
  15 | CPU_REG(CV_AMD64_BP    , Rbp, REG_X, NULL, 0)
     | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
  19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
     |                                                  ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:16:26: error: no member named 'Rsi' in
'_CONTEXT'
  16 | CPU_REG(CV_AMD64_SI    , Rsi, REG_X, NULL, 0)
     | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
  19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
     |                                                  ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:17:26: error: no member named 'Rdi' in
'_CONTEXT'
  17 | CPU_REG(CV_AMD64_DI    , Rdi, REG_X, NULL, 0)
     | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
  19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
     |                                                  ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:18:26: error: no member named 'Rax' in
'_CONTEXT'
  18 | CPU_REG(CV_AMD64_EAX   , Rax, REG_E, NULL, 0)
     | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
  19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
     |                                                  ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:19:26: error: no member named 'Rcx' in
'_CONTEXT'
  19 | CPU_REG(CV_AMD64_ECX   , Rcx, REG_E, NULL, 0)
     | ~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
  19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
     |                                                  ~~~~~~~~~~~~~ ^
In file included from C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:20:
C:\upp\upp.src\uppsrc\ide\Debuggers/amd64.cpu:20:26: error: no member named 'Rdx' in
```

```
'_CONTEXT'
   20 | CPU_REG(CV_AMD64_EDX   , Rdx, REG_E, NULL, 0)
      |                        ^~~~~~~~~~~~~~~~~~~
C:\upp\upp.src\uppsrc\ide\Debuggers\Cpu.cpp:19:85: note: expanded from macro 'CPU_REG'
   19 | #define CPU_REG(sym, context_var, kind, name, flags) case sym: return
ctx.context64.context_var;
      |                                                     ~~~~~~~~~~~~~ ^
fatal error: too many errors emitted, stopping now [-ferror-limit=]
20 errors generated.
In file included from
C:/upp/out/uppsrc/ide/Debuggers/CLANGARM64.Blitz.Gui\ide/Debuggers$blitz.cpp:46:
C:\upp\upp.src\uppsrc\ide\Debuggers\Debug.cpp:255:29: error: no member named 'Rsp' in
'_CONTEXT'
  255 |         f.sp = win64 ? c.context64.Rsp : c.context32.Esp;
      |                        ~~~~~~~~~~~~ ^
C:\upp\upp.src\uppsrc\ide\Debuggers\Debug.cpp:420:46: error: no member named 'Rip' in
'_CONTEXT'
  420 |                                          && bp_set.Find((win64 ? t.context64.Rip : t.context32.Eip)
- 1) >= 0
      |                                                              ~~~~~~~~~~~~ ^
C:\upp\upp.src\uppsrc\ide\Debuggers\Debug.cpp:427:21: error: no member named 'Rip' in
'_CONTEXT'
  427 |                                                t.context64.Rip--;
      |                                               ~~~~~~~~~~~~ ^
C:\upp\upp.src\uppsrc\ide\Debuggers\Debug.cpp:528:21: error: no member named 'EFlags' in
'_CONTEXT'
  528 |                 context.context64.EFlags |= 0x100;
      |                ~~~~~~~~~~~~~~~~~~ ^
C:\upp\upp.src\uppsrc\ide\Debuggers\Debug.cpp:539:21: error: no member named 'EFlags' in
'_CONTEXT'
  539 |                 context.context64.EFlags &= ~0x100;
      |                ~~~~~~~~~~~~~~~~~~ ^
In file included from
C:/upp/out/uppsrc/ide/Debuggers/CLANGARM64.Blitz.Gui\ide/Debuggers$blitz.cpp:162:
C:\upp\upp.src\uppsrc\ide\Debuggers\Stack.cpp:28:44: error: no member named 'Rip' in
'_CONTEXT'
   28 |                 stackFrame.AddrPC.Offset = ctx.context64.Rip;
      |                                            ~~~~~~~~~~~~~ ^
C:\upp\upp.src\uppsrc\ide\Debuggers\Stack.cpp:29:47: error: no member named 'Rbp' in
'_CONTEXT'
   29 |                 stackFrame.AddrFrame.Offset = ctx.context64.Rbp;
      |                                               ~~~~~~~~~~~~~ ^
C:\upp\upp.src\uppsrc\ide\Debuggers\Stack.cpp:30:47: error: no member named 'Rsp' in
'_CONTEXT'
   30 |                 stackFrame.AddrStack.Offset = ctx.context64.Rsp;
      |                                               ~~~~~~~~~~~~~ ^
In file included from
C:/upp/out/uppsrc/ide/Debuggers/CLANGARM64.Blitz.Gui\ide/Debuggers$blitz.cpp:169:
```

```
C:\upp\upp.src\uppsrc\ide\Debuggers\Code.cpp:52:28: error: no member named 'Rip' in
'_CONTEXT'
   52 |                return context.context64.Rip;
      |                           ~~~~~~~~~~~~~~~~~ ^
C:\upp\upp.src\uppsrc\ide\Debuggers\Code.cpp:259:21: error: no member named 'Rip' in
'_CONTEXT'
  259 |                context.context64.Rip = a;
      |              ~~~~~~~~~~~~~~~~~ ^
10 errors generated.
ide/Debuggers: 22 file(s) built in (0:01.75), 79 msecs / file
ide/Browser: 10 file(s) built in (0:02.84), 284 msecs / file
CodeEditor: 17 file(s) built in (0:03.07), 181 msecs / file
CtrlLib: 62 file(s) built in (0:09.70), 156 msecs / file

There were errors. (0:17.51) (completed at 13:33)
```

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Fri, 05 Dec 2025 12:18:09 GMT
View Forum Message <> Reply to Message

Yes, that is sort of expected...

Anyway, you can #ifdef debugger out, but you will still have a problem with openssl...

Here are my notes about how to compile openssl:

https://github.com/ultimatepp/ultimatepp/blob/master/uppbox/ Scripts/updateinfo.txt

I think it should basically work for ARM too. Maybe with some additional adjustments.

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Sun, 07 Dec 2025 13:07:43 GMT
View Forum Message <> Reply to Message

Hi Mirek,

It seems that stripping out the debugger is not quite straight forward... There are a lot of connection points for various parts. BTW: How does debugger work on new Apple hardware, or is that even comparable to ARM64 in any sense?

Also, openssl compilation seems to require more wisdom that I can offer at this time. Just

wondering, if openssl could be converted into a package to make it compile along with the rest of the packages?

The third issue is that some external hardware related binary dependencies I have with my software are not (yet) available for ARM64 architecture, so that will definitely cause significant delays for migration. To summarize, the rest of the world is not there yet, but I'm pleased to see that we can indeed compile and run some U++ based programs natively already.

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Sun, 07 Dec 2025 17:52:44 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sun, 07 December 2025 14:07Hi Mirek,

It seems that stripping out the debugger is not quite straight forward... There are a lot of connection points for various parts. BTW: How does debugger work on new Apple hardware, or is that even comparable to ARM64 in any sense?

It is supposed to work in the same way as on Rapsberry PI - it is posix environment, so GDB is used as (poor) debugger backend. Supposed, as to have working gdb on MacOS is not exactly easy, so there is no debugger in theide on MacOS at this time.

Quote:
Also, openssl compilation seems to require more wisdom that I can offer at this time. Just wondering, if openssl could be converted into a package to make it compile along with the rest of the packages?

Maybe, but if, then it requires more wisdom than I can offer...

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Mon, 08 Dec 2025 13:19:23 GMT
View Forum Message <> Reply to Message

Hi,

Quote:
Maybe, but if, then it requires more wisdom than I can offer...

OK, I deserved that!

BTW: I just noticed that I do not seem to get email notifications about new messages here as I used to. The Post Notification switch is still enabled just as before...

I will try to dig into openssl compilation challenge, but can't make any promises.

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Mon, 08 Dec 2025 22:10:11 GMT
View Forum Message <> Reply to Message

Hi,

I composed a new way to build openssl static libraries for Windows (x86, x86_64 and arm64), using CLANG (mingw-llvm) cross compilation on Linux Mint. This might even work on other Ubuntu based distros...

There are two scripts, one for downloading and preparing the required tools and downloading the sources for openssl. The second is for building the static libraries *.a for all three architectures.

USAGE: Just copy the scripts to your home directory and first run the preparing script once and then the build script once -- or until you get working results. (It is possible that further dependencies might be missing from the preparing script, so let me know if this is the case.)

The builder script creates an openssl folder in your home directory that contains only the headers and the static libraries for each architecture.

I hope this makes it easier to update openssl. Please test.

Best regards,

Tom

File Attachments
1) openssl-windows-static-library-builder-on-linux-mint-22.2.zip, downloaded 157 times

---

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Mon, 08 Dec 2025 22:50:38 GMT
View Forum Message <> Reply to Message

Hi,

Still in trouble with commenting out debugger. Any suggestions on how to do that cleanly and efficiently?

Another issue that surfaced is libclang which is missing too. How to build or is it available readily in some location?

I think I need some #define that tells in package manager and code about ARM64 architecture so that parts can be included and excluded depending on that. Windows + ARM seems a bit different than just plain old Windows...

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Tue, 09 Dec 2025 07:08:20 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 08 December 2025 23:10
I hope this makes it easier to update openssl.

Unfortunately we still to support MSVC...

Subject: Re: Do we have support for Windows on ARM64?
Posted by mirek on Tue, 09 Dec 2025 07:15:14 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 08 December 2025 23:50Hi,

Still in trouble with commenting out debugger. Any suggestions on how to do that cleanly and efficiently?

IDK, I guess the best option is to fix it I guess he is unhappy about CPU registers...

Quote:
Another issue that surfaced is libclang which is missing too. How to build or is it available readily in some location?

https://github.com/llvm/llvm-project/releases

You install win arm64 and fetch libclang.dll from files...

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Tue, 09 Dec 2025 16:55:01 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 09 December 2025 09:15Tom1 wrote on Mon, 08 December 2025 23:50Hi,

Still in trouble with commenting out debugger. Any suggestions on how to do that cleanly and efficiently?

IDK, I guess the best option is to fix it I guess he is unhappy about CPU registers...

Quote:
Another issue that surfaced is libclang which is missing too. How to build or is it available readily in some location?

https://github.com/llvm/llvm-project/releases

You install win arm64 and fetch libclang.dll from files...
Thanks Mirek,

libclang.lib, libclang.dll, etc. can be found just there.

I managed to block all calls to debugger parts and comment them somewhat out and TheIDE even started natively. Of course as that was a quick and dirty test only, I reverted to the original and started to figure out the Debugger.

Yes, the debugger battle starts with registers. The cvconst.h file is missing ARM64 registers, but they can be found in:

https://github.com/microsoft/microsoft-pdb/blob/master/inclu de/cvconst.h

Actually, the above file is MIT licensed, so could that possibly be used instead of the original (LGPL)? Don't know about the other consequences though...

The registers are different and Context needs ARM64_NT_CONTEXT in place of CONTEXT.

Next, a arm64.cpu needs to be created, but that's not straight forward.

After this everything gets dark... there are a lot of #ifdef s around the code and we should add a whole bunch more to have it go the ARM64 way. But I need better understanding about the Debugger and ARM64 registers to figure out how everything is supposed to be dealt with. I will return if I get something useful done here. Currently I'm not so sure I will get anywhere with this...

Best regards,

Tom

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Tue, 09 Dec 2025 23:28:33 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 09 December 2025 09:08Tom1 wrote on Mon, 08 December 2025 23:10
I hope this makes it easier to update openssl.

Unfortunately we still to support MSVC...

Maybe this helps... based it on MSBT22 tool chain and Strawberry Perl. Those installed, it should do the job. (At least it works here.) The libs for all architectures can be found in the source dir/dist/lib/<arch>.

Best regards,

Tom

File Attachments

1) openssl_build.bat, downloaded 187 times

Subject: Re: Do we have support for Windows on ARM64?
Posted by Tom1 on Mon, 15 Dec 2025 20:37:32 GMT
View Forum Message <> Reply to Message

Hi Mirek,

OK, now I have something to show. The attached file includes all the code changes required for compiling TheIDE on Windows 11 on ARM64.

- The prior scripts for creating the different versions of openssl library should be used to create the required binaries.
- The SDL2, MySQL and Postgresql are not currently supported on arm64.
- MSVS2026 (community) support has been added to Automatic build methods setup. This includes x86, x64 and arm64.
- Automatic build methods setup now also supports setting up clang-mingw for compiling on arm64.

- A slight change in library locations for openssl:
  bin/openssl/x86

bin/openssl/x64
bin/openssl/arm64

- And the same for llvm library too:
bin/llvm/x86
bin/llvm/x64
bin/llvm/arm64

(These library location changes are needed for cross platform functionality. We can compile at least on ARM64 for all three architectures now. Possibly the other way around too... I hope.)

- Various other tunings have been done to get this work.

It is also worth noting, that a slightly wider installation of clang-mingw is needed in order to support arm64 (aarch64). This is easily obtained from the source you already pointed out.

As a reference test, I compiled examples/UWord for all three architectures on Windows 11 ARM using both clang-mingw and MSVS2026.

NOTE: Unfortunately, while debugger related stuff compiles already and registers are defined, I could not make it work with ARM64.

Please take a look and test... I surely would not mind if the debugger issue was solved.

Best regards,

Tom

[EDIT] Now there's MSBT2026 support included too.

File Attachments

1) upp-arm64.zip, downloaded 158 times