Go 64!

Bring you 32-bit applications
into 64-bit future

Primož Gabrijelčič

thedelphigeek.com

gabr42@gmail.com

gabr42

github.com/gabr42

linkedin.com/in/gabr42

DelphiDay italian conference

9,10 Giugno 2026
Piacenza

wintech italia

OPEN-SOURCE PROJECTS

github.com/gabr42

OmniThreadLibrary

Parallel programming library for VCL/Windows

OmniThreadLibrary-NG

Platform-independent parallel programming library [WIP]

GpDelphiUnits

Various helper units

The Delphi Geek

random ramblings on Delphi, programming, Delphi programming, and all the rest

Tuesday, April 21, 2026

OmniThreadLibrary-NG teaser

Normal suite (default — Stress excluded)

Platform	Found	Ignored	Passed	Failed	Time
Win32	322	0	322	0	40 s
Win64	322	0	322	0	42 s
Linux64	325	3	322	0	32 s
Android	325	3	322	0	38 s
Total			1288	0	~2 min 32 s

Stress only (--include:Stress)

Platform	Found	Ignored	Passed	Failed	Time
Win32	7	0	7	0	293 s
Win64	7	0	7	0	293 s
Linux64	7	0	7	0	275 s
Android	7	0	7	0	284 s
Total			28	0	~19 min 5 s

A bit of history,
A bit of future

Platforms through the time

- Windows 16-bit [Delphi 1, 1995]
- Windows 32-bit [Delphi 2, 1996]
- Windows 64-bit [Delphi XE2, 2011]
- Delphi 13, 2025: ~~Android32~~, Android64, iOS64, Linux64, Mac64, MacARM64
- Delphi 13.1: WindowsARM64EC
  - https://www.embarcadero.com/products/rad-studio/whats-new-in-13-florence

Today: Migration from Win32 to Win64

- Reasons for migration
- Problems and solutions
- Testing
- Using AI helpers

Reasons for migration

Why go from Win32 to Win64?

- Limitations of 32-bit address space (≈ 2–3 GB)
  - {$SetPEFlags IMAGE_FILE_LARGE_ADDRESS_AWARE}
- Better utilization of modern CPUs
- The future of Windows is 64-bit
  - Do not expect 32-bit support on new architectures
  - https://docwiki.embarcadero.com/RADStudio/Florence/en/ARM64EC
- (External libraries that are no longer supported in 32-bit)

“Change platform and recompile“?

- The code often compiles without errors
- Errors are hidden and depend on the data
- Problems often don't surface until the code is in production

Differences between Win32 and Win64

What is different?

- Pointers: 32 → 64 bits
- Record alignment
- Size of certain data types (NativeInt, LongInt …)
  - cardinalVar := cardinal (pointerVar)
- External DLLs must be 64-bit
  - Problem if you only have 32-bit versions
- Assembler, passing parameters by stack, handling exceptions

Data type size

Type	Windows 32, POSIX 32	Windows 64, POSIX 64
Integer	4	4
Cardinal	4	4
Int64	8	8
Pointer	4	8
NativeInt/NativeUInt	4	8
WPARAM/LPARAM/LRESULT	4	8

	Windows 32 and 64, POSIX 32	POSIX 64
LongInt/LongWord	4	8

Pointers

- `cardinalVar := cardinal(pointerVar);`
  - Sometimes it works, sometimes it doesn't, depending on the value in *pointerVar*

- `PostMessage(Handle, 123, 0, cardinal(pointer));`
  - => NativeUInt(pointer)
  - => LPARAM(pointer)

Record size and field alignment

- SizeOf(record) may change
  - Due to changes in the size of basic types
  - Due to changes in ‘Record field alignment’
    - In Windows32, it is ‘Double word’ or ‘Quad word’, depending on the Delphi version

- Particularly dangerous with file/network I/O
- Be careful if you use external libraries (C, C++)
- Be careful with Windows API calls if you have your own declarations
  - Windows API may expect a different size or different field alignment

Assembler and Win64

- Different register names
- Additional/modified instructions
- Inline assembler does not exist in Win64
- We still have "pure assembler" functions
  - Only on Intel platforms
- Assembler code must be adapted/rewritten

Assembler: Migration strategies

- Complete port to Pascal (recommended)
  - If speed is not an issue
- Dual implementation
  - Inline in pascal unit
  - External code – object file
    - {$L ‘helper.obj’}
    - function zlibVersion: MarshaledAString; cdecl; external object 'zutil.o’;
  - External code – library
    - WinARM64EC can load Intel DLLs
    - function zlibVersion: MarshaledAString; cdecl; external 'libzlib.dll’;
    - function zlibVersion: MarshaledAString; cdecl; external 'libzlib.a';

Components and external libraries

- Upgrade to a newer version
- “Do it yourself” – if you have the source code
- Replace components/libraries
- Partial upgrade to 64-bit code

Partial migration

- A 32-bit program that offloads some functionality to 64-bit code
- A 64-bit program that offloads some functionality to 32-bit code
- A 32-bit program **cannot load** a 64-bit DLL
  - And vice versa
- A 32-bit program **can run** a 64-bit program
  - And vice versa

Examples from real code

cardinal(pointer)

```
var headerEnd: cardinal;

methodInfoHeader := ObjAuto.GetMethodInfo(
    WorkerIntf.Implementor, methodName);

headerEnd := cardinal(methodInfoHeader) + methodInfoHeader^.Len;
```

- This code worked until Delphi 13.1!

```
if PInteger(@opsStage)^ <> NativeInt(nil) then begin
    Assert(PInteger(@opsStageEx)^ = NativeInt(nil));
```

Code from an external library

```
function CompatibleTimeSetEvent(Delay, Resolution: UINT;
  TimeProc: TFNTimeCallBack; User: DWORD; Event: UINT):
MMResult;


FEndOfStreamTimer := CompatibleTimeSetEvent(Delay,
TIMEOUT_RESOLUTION, @EndOfStreamTimer, Cardinal(Self),
TIME_ONESHOT);
```

Low-level hacks

```
if ImageNTHeaders.FileHeader.Machine
   <> {$IFDEF CPUX64} $8664 {$ELSE} $14C {$ENDIF}
then
```

- CPUX64 or CPU64BITS
  - https://docwiki.embarcadero.com/RADStudio/en/Conditional_compilation_(Delphi)

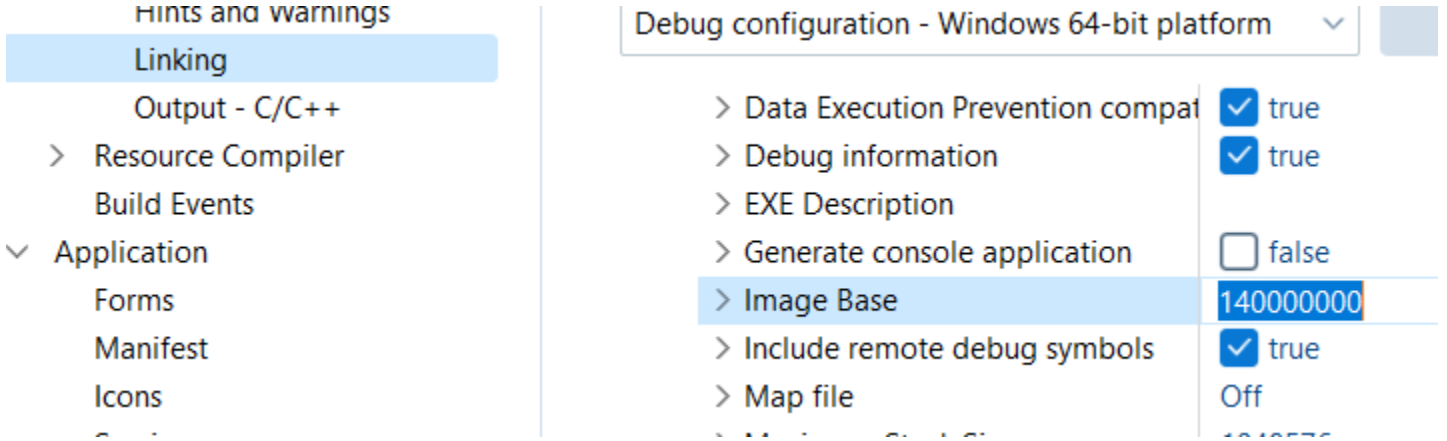
Windows API

```
SP_DEVICE_INTERFACE_DETAIL_DATA_A = packed record
    cbSize: DWORD;
    DevicePath: array [0..ANYSIZE_ARRAY - 1] of AnsiChar;
{$IFDEF WIN64}
    Padding: array [0..2] of AnsiChar;
{$ENDIF WIN64}
  end;
```

Testing

Pointers

- Change image base to be > 4 GB
  - Project, Options, Building, Delphi Compiler, Linking, Image Base
- Or use FastMM4
  - AlwaysAllocateTopDown
    - FastMM4Options.inc
    - Not needed in Delphi 13
  - FullDebugMode
  - CheckHeapForCorruption (slow!)



Static code analysis - manual

- cardinal(, integer(, longint(, longword(, pointer
- SetWindowLong, GetWindowLong, SendMessage, PostMessage
  - SetWindowLongPtr
- GWL_USERDATA, GWL_WNDPROC

Combining signed and unsigned 64-bit types - treated as an unsigned type	True
Implicit cast of one integer type to another may result in data loss	True
Implicit conversion from wide integral type to narrow type, may result in data loss	True
Unsafe typecast	True

Static code analysis - automated

- FixInsight
- Peganza
- AI

AI

LLM AI as assistant

- Identifying problem areas
- Explaining why the code is dangerous
- Gradual migration of units

AI prompta example

- “Check unit xxx for 64-bit migration problems”

- Better: “What would be a good query to give Claude to check existing codebase for 64-bit migration problems?”

Rewriting assembler code

- „Can you convert this function to Win64 and also write a pure pascal version?“
- **Required**: „Please verify that all three versions are functionally identical.“
- **Required²**: Testing!

Final thoughts

Recap

- Win64 migration is not a trivial problem
- Problems can remain hidden for a long time
- AI is a great assistant, not a replacement