

# Re: Native Code vs PCode

---

*Source:*

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2005-09/msg00270.html>

---

- *From:* "M. Posseth" <[MPosseth@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:MPosseth@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 2 Sep 2005 04:58:03 -0700
- 

for those interested of performing some benchmarks

copy this in a module

and start testing what is for you, the optimal compiler settings

#####

Option Explicit

```

//*****
// Api declares
//*****
Public Declare Function QueryPerformanceCounter Lib "kernel32"
(lpPerformanceCount As LARGE_INTEGER) As Long
Public Declare Function QueryPerformanceFrequency Lib "kernel32"
(lpFrequency As LARGE_INTEGER) As Long

Public Type LARGE_INTEGER
lowpart As Long
highpart As Long
End Type

//*****
// Variables
//*****
Private m_Initialized As Boolean
Private m_Frequency As Double '// Frequency of the hi-res counter
Private m_Start As LARGE_INTEGER '// Hi-res timer value (Before timing)
Private m_End As LARGE_INTEGER '// Hi-res timer value (After timing)
Private m_CallingDelay As Double '// Time required to call the stoptiming
function
Private m_isTiming As Boolean '// Timing in progress flag
Private m_TimingComplete As Boolean

```

## Re: Native Code vs PCode

```
//*********************************************************************
// Const
//*********************************************************************
//--- Conversion units ---
Public Const Units_MilliSeconds = &H0
Public Const Units_MicroSeconds = &H1
Public Const Units_Seconds = &H2
Public Const Units_NoConversion = &H3
Public Enum UnitsType
    Milliseconds = Units_MilliSeconds
    MicroSeconds = Units_MicroSeconds
    seconds = Units_Seconds
    NoConversion = Units_NoConversion
End Enum

//*********************************************************************
// Timing functions
//*********************************************************************
Public Function InitializeTimer() As Boolean

    //**** Get the high resolution counter frequency (Beats per seconds) ****
    Dim iFreq As LARGE_INTEGER
    If (QueryPerformanceFrequency(iFreq) = 0) Then

        //**** There performance counter doesn't work... ****
        Exit Function
    End If
    m_Frequency = LargeIntToDouble(iFreq)
    m_Initialized = True

    InitializeTimer = True
End Function
Public Sub StartTiming()
    If (m_isTiming) Then Exit Sub

    //**** If the timer is not initialized, initialize it ****
    If (Not m_Initialized) Then
        If (Not InitializeTimer) Then Exit Sub
    End If

    //**** Calculate the time required to call the StopTiming function ****
    m_isTiming = True
    Call QueryPerformanceCounter(m_Start)
    m_CodeTiming.StopTiming
    m_CallingDelay = (LargeIntToDouble(m_End) - LargeIntToDouble(m_Start))

    //**** Read the current value of the hi-res timer ****
    m_isTiming = True

```

## Re: Native Code vs PCode

```
Call QueryPerformanceCounter(m_Start) '/***** This MUST be the last
line in the sub ****
```

```
End Sub
```

```
Public Sub StopTiming()
```

```
If (Not m_isTiming) Then Exit Sub
```

```
/***** Read the current value of the hi-res timer ****
```

```
Call QueryPerformanceCounter(m_End)
```

```
m_isTiming = False
```

```
m_TimingComplete = True
```

```
End Sub
```

```
Public Function ReadCodeTiming(Optional Units As UnitsType =
Units_MicroSeconds) As Double
```

```
Dim nCounts As Long
```

```
nCounts = (LargeIntToDouble(m_End) - LargeIntToDouble(m_Start)) -
m_CallingDelay
```

```
If (nCounts < 0) Then nCounts = 0
```

```
Select Case Units
```

```
Case Units_MicroSeconds
```

```
ReadCodeTiming = (nCounts / (m_Frequency / 1000000))
```

```
Case Units_MilliSeconds
```

```
ReadCodeTiming = (nCounts / (m_Frequency / 1000))
```

```
Case Units_Seconds
```

```
ReadCodeTiming = (nCounts / m_Frequency)
```

```
Case Else
```

```
ReadCodeTiming = nCounts
```

```
End Select
```

```
End Function
```

```
/******
```

```
// Conversion functions
```

```
/******
```

```
Public Function LargeIntToDouble(LINT As LARGE_INTEGER) As Double
```

```
Const Shift32bit = (4# * 1024# * 1024# * 1024#)
```

```
Dim LoInt As Double
```

```
LoInt = LINT.lowpart
```

```
If (LoInt < 0) Then
```

```
LoInt = (LoInt + Shift32bit)
```

```
End If
```

```
LargeIntToDouble = (LINT.highpart * Shift32bit) + LoInt
```

```
End Function
```

Re: Native Code vs PCode

#####

"M. Posseth" wrote:

- > well ...
- >
- > I justed wanted to make clear what is exactly the difference as an extra
- > comment to Tom`s reply
- >
- > my own experience tells me that Tom is absolutely right in his statement
- > "It's easy enough to build & test each way though if you want to be certain."
- >
- > in my situation ,, ( programs with lots of array`s , database handling ,
- > generating of text files ) native code is superior to p-code in execution
- > speed . i know this for the simple fact that i benchmarked my app in all
- > situations ( p-code , native code with and without the different optimization
- > options )
- >
- > Also be aware of the following , P-code can eventually be easier to
- > decompile as native code , this might also be something worth mentioning

> "Tony Proctor" wrote:

- >> I guess you haven't seen any of the previous threads on this subject, e.g.
- >>

[http://groups.google.ie/group/microsoft.public.vb.general.discussion/browse\\_frm/thread/0d579a81dc11e19c/117fe49d](http://groups.google.ie/group/microsoft.public.vb.general.discussion/browse_frm/thread/0d579a81dc11e19c/117fe49d)

- >>> The point being that P-code is interpreted rather than being converted to
- >>> native code via a JIT compiler

>>> Tony Proctor

>>> "M. Posseth" <MPosseth@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message  
>>> [news:33DEBA46-0AAB-4FDE-BDF2-D25993AD2C82@xxxxxxxxxxxxxxxxxxxx](mailto:news:33DEBA46-0AAB-4FDE-BDF2-D25993AD2C82@xxxxxxxxxxxxxxxxxxxx)

- >>>> If you have the Professional or Enterprise edition of Visual Basic, you
- >>>> can
- >>>> compile your code either in standard Visual Basic p-code format or in
- >>>> native
- >>>> code format. Native code compilation provides several options for
- >>>> optimizing
- >>>> and debugging that aren't available with p-code.

- >>>>> P-code, or pseudo code, is an intermediate step between the high-level
- >>>>> instructions in your Basic program and the low-level native code your
- >>>>> computer's processor executes. At run time, Visual Basic translates each
- >>>>> p-code statement to native code. By compiling directly to native code

## Re: Native Code vs PCode

>> format,  
>>> you eliminate the intermediate p-code step.  
>>>  
>>> You can debug compiled native code using standard native code debugging  
>>> tools, such as the debugging environment provided by Visual C++. You can  
>>> also  
>>> use options available in languages such as Visual C++ for optimizing and  
>>> debugging native code. For example, you can optimize code for speed or for  
>>> size.  
>>>  
>>> Note All projects created with Visual Basic use the services of the  
>>> run-time DLL (MSVBVM60.DLL). Among the services provided by this DLL are  
>>> startup and shutdown code for your application, functionality for forms  
>> and  
>>> intrinsic controls, and run-time functions like Format and CLng.  
>>>  
>>> Compiling a project with the Native Code option means that the code you  
>>> write will be fully compiled to the native instructions of the processor  
>>> chip, instead of being compiled to p-code. This will greatly speed up  
>> loops  
>>> and mathematical calculations, and may somewhat speed up calls to the  
>>> services provided by MSVBVM60.DLL. However, it does not eliminate the need  
>>> for the DLL.  
>>>  
>>> regards  
>>>  
>>> Michel Posseth [MCP]  
>>>  
>>> "Tom Esh" wrote:  
>>>  
>>>> On Fri, 2 Sep 2005 08:15:42 +0200, "Ivan Debono"  
>>>> <ivanmdeb@xxxxxxxxxxxx> wrote:  
>>>>  
>>>>> Hi all,  
>>>>>  
>>>>> I've got some activex dlls that access a database, retrieve recordsets,  
>> loop  
>>>>> through them, calculate stuff, etc...  
>>>>>  
>>>>> Would it make a big difference if I compiled to PCode instead of Native  
>> Code  
>>>>> (Fast Code)??  
>>>>>  
>>>>> The difference is in the size which is important for me as I need to  
>> package  
>>>>> everything in a setup file and make it available on the internet for  
>>>>> downloading.  
>>>>>  
>>>>> Very unlikely you would see much improvement with native unless you're  
>>>>> doing a ~lot~ of calculations. It's easy enough to build & test each  
>>>>> way though if you want to be certain.

>>>>  
>>>> -Tom  
>>>> MVP – Visual Basic  
>>>> (please post replies to the newsgroup)  
>>>>  
>>  
>>  
>>  
>>  
.

---

• **References:**

- ◆ **Native Code vs PCode**  
    ◇ From: Ivan Debono
  - ◆ **Re: Native Code vs PCode**  
    ◇ From: Tom Esh
  - ◆ **Re: Native Code vs PCode**  
    ◇ From: M. Posseth
  - ◆ **Re: Native Code vs PCode**  
    ◇ From: Tony Proctor
  - ◆ **Re: Native Code vs PCode**  
    ◇ From: M. Posseth
- 
- Prev by Date: **Re: Native Code vs PCode**
  - Next by Date: **VB control Help**
  - Previous by thread: **Re: Native Code vs PCode**
  - Next by thread: **Re: Native Code vs PCode**
  - Index(es):
    - ◆ **Date**
    - ◆ **Thread**