

Re: Release Dll Problem – Optimization Issue

Source:

<http://www.tech-archive.net/Archive/PocketPC/microsoft.public.pocketpc.developer/2008-06/msg00251.html>

- *From:* sid <mailsid@xxxxxxxxx>
 - *Date:* Mon, 30 Jun 2008 05:49:08 -0700 (PDT)
-

Hi Guys

Just needed to update on the current problem I asked....

The problem has been rectified after hours of long searching/reading the documentations on compiler optimisations....These are some steps I have followed in accordance with the rules for "optimising for speed" on embedded platforms and it turned out to be extremely effective.

* If optimising for speed on smaller size processors, avoid using pointers or taking the address of variables (especially in function arguments). This limits the compiler in using fast registers for the manipulation of data. Using pointers and dereferencing also uses extra steps in the compiled code to determine the location of the data before accessing it.

(NOTE: Use pointers when optimising for SIZE)

* On smaller size microprocessors, when optimising for speed, avoid the use of function arguments wherever possible. Sometimes speed optimisations try to push the variables into registers rather than stack. Smaller processors have limited number of all-purpose registers and you can end up in address-collisions (unlikely, but atleast i experienced it) and slow-stack retrieval (push/ pop) operations. I re-structured lot of my functions to take zero or limited number of arguments, and that too only simple datatypes as INT, LONG etc. No LPTSTR's coming into argument list now for me and it helped a lot.

* Although a bad programming practice, but its recommended to use global variables instead of clogging the function argument list. Again this helped me a lot during my re-structuring of the code and it seemed like an appropriate tradeoff.

* Since the number of registers are limited, using static variables wherever necessary, may improve the execution speed of the code as well as decrease the size of the code. When the number of registers are limited, declaring variables as "static" gives them a fixed location in memory. This helps eliminate the variable being used in

Re: Release Dll Problem – Optimization Issue

stack space, and speeds up access to the variable.

* carefully assess variables which can be declared as const/ volatile and make use of it. It will help the compiler to determine the type of data when optimising. For example declaring "const" variables will help the compiler to make memory optimisations.

* Lastly, although I havent done so far since I do not completely understand it, is the optimisation due to loop unrolling. It is however unnecessary, if the compiler automatically does it when optimising for speed. Loop unrolling is the process of taking looped code and repeating the number of iterations of the code without looping. This would increase teh code size but increase the speed. I serisouly do not know how to check/ apply this point, but hunt is still on..

By applying the above steps I have already achieved 3.5–4.5 times faster code than its predecessor version. This is a lot of boost for the time–critical application I am working on. I am sure to achieve another 100–300% gain on top of it, but I guess it will take some time since the code is quite big and it can be disastrous to change anything blindly...so insetad of going one–module at a time, I am going one–function at a time to assess the best possible optimisation route...

Cheers
Sid

.