[7.44] Local subroutines execute faster than global subroutines

TI Basic subroutines can be defined outside of the calling routine (an global subprogram), or they can be defined within the calling routine (a local subprogram). As an example, this program *t()* calls the external function *sub1()*:

```
t()
Prgm
Local k,r
For k,1,500
 sub1(k)→r
EndFor
EndPrgm

sub1(x)
Func
Return x^2
EndFunc
```

sub1() would be defined locally in *t()* like this:

```
t()
Prgm
Local k,r,sub1

Define sub1(x)=Func
 Return x^2
EndFunc

For k,1,500
 sub1(k)→r
EndFor
EndPrgm
```

Local subroutine calls are slightly faster, as this table shows.

	Global	Local
Unarchived	37.58 mS	37.04 mS
Archived	38.02 mS	36.94 mS

The times in the table are for one function call, averaged over 500 calls as shown in the examples above. The test routines ran on a HW2 TI-92 Plus, AMS 2.05. The local routine, archived, is about 3% faster than the global routine. This is not significant for most applications, but if your program makes thousands of calls to the subroutine, you may save a few seconds.

Defining the subroutine locally reduces the number of variables in a folder, and makes it a bit easier to remove the program if needed. If the subprogram has no use outside of its calling routine, there is little point to defining it globally.