[7.14] Determine calculator model and ROM version in programs

It can be useful for your program to determine the model and ROM version of the calculator on which it is running. For example, since the 92+ LCD screen is 240 x 128 pixels, and the 89 screen is only 160 x 100 pixels, programs that take advantage of the larger 92+ screen won't work well on the 89: all the output won't be shown. The 92+ has many functions not included in the 92, so if your program uses those functions, it won't run on the 92.

If you want your programs to determine if they are running on an 89 or a 92+, Frank Westlake provides this code to identify the calculator model:

```
¨Identify Model
Local j,tmp,timodel
list▸mat(getConfg(),2)→tmp
For j,1,rowDim(tmp)
 If tmp[j,1]="Screen Width" Then
  If tmp[j,2]=240 Then:"TI-92"→timodel
  ElseIf tmp[j,2]=160 Then:"TI-89"→timodel
  Else:""→timodel
  EndIf
 EndIf
EndFor
```

When this code finishes, the variable 'timodel' is "TI-92", "TI-89", or "" if the calculator seems to be neither model. Frank notes that this is slow, but reliable.

Lars Fredericksen offers this code which also determines if the calculator is a TI92 or TI92II:

```
¨Identify model
Local j, tmp, timodel
 ""→timodel
 getconf()→tmp
 If getType(tmp)="EXPR" Then
   "TI-92"→timodel
 Else
  For j,1,dim(tmp),2
   If tmp[j]="Screen Width" Then
    If tmp[j+1]=240 Then
     "TI-92p"→timodel
    ElseIf tmp[j+1]=160 Then
     "TI-89"→timodel
    EndIf
    Exit
   EndIf
  EndFor
 endif
```

After execution, the local variable *timodel* holds the strings "TI-92" for a TI-92, "TI-92p" for a TI92 Plus, or "TI-89" for a TI-89.

Frank Westlake has found that the product ID can also be used to identify both the calculator model as well as the AMS ROM version. Specifically:

Product ID	Calculator Model	AMS ROM version
03-0-0-2A	TI-89	1.00
03-1-3-66	TI-89	1.05
01-0-0-4A	TI-92+	1.00
01-0-1-4F	TI-92+	1.01
01-1-3-7D	TI-92+	1.05

Frank also provides these functions to identify the model and ROM version.

First, this is an example function which evaluates some user function *func1()* if the model is an 89 or 92+, and the version is 1.05 or greater. Otherwise, the function returns *undef*.

```
example()
func
local timodel
model()→timodel
if (timodel="TI-89" or timodel="TI-92+") and version()≥1.05:return func1()
return undef
endfunc
```

This function returns the product ID as a string, or "TI-92" if the calculator is a TI-92 without the Plus module:

```
pid()
func
© Product ID
local i,m
1→i
getconfg()→m
if gettype(m)="EXPR":return "TI-92"
while m[i]≠"Product ID"
i+1→i
endwhile
return m[i+1]
endfunc
```

This function calls *pid()* and returns the model number as a string.

```
model()
func
@Identify model
Local tmp
pid()→tmp
if mid(tmp,2,1)="1":return "TI-92+"
if mid(tmp,2,1)="3":return "TI-89"
return tmp
endfunc
```

This function calls *pid()* and returns the ROM version as a floating-point number.

```
version()
func
@Identify version
Local tmp
pid()→tmp
if mid(tmp,4,3)="0-0":return 1.0
if mid(tmp,4,3)="0-1":return 1.01
if mid(tmp,4,3)="1-3":return 1.05
return 0
endfunc
```

Frank also offers these comments on his code:

"In most cases none of this will be necessary. In the rare case that it is, it will probably be more useful to incorporate fragments into a single function or program making the overall code much smaller.

I claim no rights to any of this code, it is all public domain.

There doesn't appear to be any way to determine hardware version programatically."

(Credit to Frank Westlake and Lars Fredericksen)