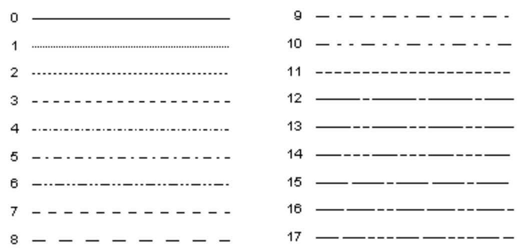
SetDashPattern

`SetDashPattern` *`dashNumber`*`, {`*`d1`*`, `*`s1`*` [, `*`d2`*`, `*`s2`*`]…}`

The SetDashPattern operation defines a dashed-line pattern for a user-defined dashed line. These dashed lines are used by the drawing tools and the Modify Waves Appearance dialog, and are elsewhere referred to as “line styles”.

Parameters

dashNumber specifies which dash pattern is to be set. It must be between 1 and 17. Dash pattern 0 is reserved for a solid line.



{*d1,s1* [,*d2,s2*]…} defines the dash pattern. The dash pattern consists of 1 to 8 “dash,skip” pairs. Each pair consists of the number of drawn points followed by the number of skipped points.

d1 specifies the number of drawn points and *s1* specifies the number of skipped points in the first “dash,skip” pair. *d2* and *s2* specify the number of drawn and skipped points in the second pair and so on. Each draw or skip value must be between 1 and 127.

Details

SetDashPattern updates *all* graphs, panels and layouts so that any dashed lines will be updated with the new pattern. If you repeatedly call SetDashPattern from within a macro, you should precede the commands with the PauseUpdate operation to prevent multiple updates (which would be slow).

Dashed lines may also be redefined by the Dashed Lines dialog which you can choose from the Misc menu.

The dashed line patterns are saved as part of the experiment. When a new experiment is opened, the preferred dash patterns are restored.

Some programs and printer drivers do not properly render dashed lines with many “dash,skip” pairs.

Examples

```
Make test; Display test
SetDashPattern 17, {20,3,15,8}       // sets last dashed line pattern
ModifyGraph lstyle(test)=17          // apply pattern to trace
```

See Also

PauseUpdate and **ResumeUpdate** operations, and **Dashed Lines** on page III-496.

SetDataFolder

`SetDataFolder` *`dataFolderSpec`*

The SetDataFolder operation sets the current data folder to the specified data folder.

Parameters

dataFolderSpec can be a simple name (MyDataFolder), a path (root:MyDataFolder) or a string expression containing a name or path. It can also be a data folder reference created by the **DFREF** keyword or returned by **GetDataFolderDFR**.

If *dataFolderSpec* is a path it can be a partial path relative to the current data folder (:MyDataFolder) or an absolute path starting from root (root:MyDataFolder).

Examples

```
SetDataFolder foo              // Sets CDF to foo in the current data folder
SetDataFolder :bar:foo         // Sets CDF to foo in bar current data folder
SetDataFolder root:foo         // Sets CDF to foo in the root data folder
DFREF savedDF= GetDataFolderDFR()    // Remember current data folder
NewDataFolder/O/S root:MyDataFolder  // Set CDF to a new data folder
```