AddResource Make arbitrary data in memory into a resource

#include <Resources.h> **Resource Manager**

void **AddResource(***theHandle*, *rType*, *rID*, *rName* **)**;

Handle	*theHandle* ;	a handle leading to data to store as a resource
ResType	*rType* ;	a 4-byte resource type
short	*rID* ;	a 2-byte resource ID
ConstStr255Param	*rName* ;	address of length-prefixed name; "\p" = none

Given a handle to any type of data, this assigns a resource ID, type, and name to that data and inserts an entry into the **resource map** for the current resource file. It also tags the file for update.

theHandle is a handle leading to any type of data *except an existing resource*. It is typically a handle obtained via **NewHandle**, or possibly one used in **DetachResource**. When storing Standard Resource Types, you should format this data to match the standard layout.

rID specifies the desired resource ID to be assigned. You may call **UniqueID** or **Unique1ID** to obtain a value for this parameter.

rType is a 4-byte ResType structure (any 32-bit long integer is valid, but 4-character constants such as 'PICT' or 'STR ' are normally used). It specifies the four bytes identifying the type of the resource.

rName is the address of a length-prefixed Pascal-style string containing the desired name for the resource. You may specify no name by using an empty string (e.g., "\p").

Returns: none (if *theHandle* is NIL or is already a resource handle, **ResError** will return the addResFailed error. If this call can not be completed because of memory shortage or other reasons, **ResError** will return an appropriate operating system error code).

Notes: **AddResource** converts an existing handle (either one that points to any existing data or even an empty handle) into a handle recognized by the **Resource Manager** and saved in the **resource map**. This affects the map of the current resource file (see **UseResFile**).

This function automatically sets the resChanged bit of the resource attribute. Thus, when the current resource file is closed or updated the resource data and map will be written to disk (see **ChangedResource**). All other attributes are cleared; use **SetResAttrs** before writing the resource if you want different attributes.

This is the opposite of **DetachResource**, which converts a resource handle into a generic handle. To duplicate a resource, use **DetachResource** followed by **AddResource**.

Example

```
#include <Resources.h>

Handle      textHandle;
short       rID;

PtrToHand( "some text", &textHandle, 10 );      /* make a handle */

while ( (rID=UniqueID('TEXT')) < 128 )          /* generate unique ID */
    ;
AddResource( textHandle, 'TEXT', rID, "\pMyTextName" );
if ( ResError() ) { /* ... an error occurred ... */ }
WriteResource( textHandle );                    /* force the update */
```