# Apple II
# File Type Notes

Developer Technical Support

## File Type: $C7 (199)
**Auxiliary Type: All**

Full Name: Control Panel
Short Name: Control Panel

Revised by: Dave "Flag Bits" Lyons — May 1992
Written by: Matt Deatherage & Darryl Lovato — September 1989

Files of this type contain control panels (formerly called CDevs) for the Apple IIGS Control Panels New Desk Accessory.
**Changes since December 1991:** Updated for Control Panels NDA 2.0 in System 6.0.

---

Files of type $C7 contain control panels. When deciding whether to write an NDA or a control panel, keep in mind that control panels normally don't take any RAM when they are not in use, don't take up space in the Apple menu, and automatically re-open at the same screen position where they were last used. On the other hand, control panels are limited to a single code segment, and the main window has a fixed size.

Before System 6, the Control Panels NDA presented control panels (then called "CDevs" to the user) in a single window. In System 6, each control panel appears in its own window.

## Auxiliary Type

The auxiliary type of CDevs is defined bit by bit. Currently, only bit 15 is defined—it's the "inactive" bit. As with desk accessories, FSTs, and setup files, the control panel is not loaded or used if this bit is set. All other bits are reserved and must be set to zero. (In 6.0, inactive control panels do not appear in the Control Panels NDA's list, but the user can still open them directly from the Finder.)

## How Control Panels Works

The Control Panels NDA lets the user choose control panels, and it communicates with open control panels using a small collection of messages. Most events the Control Panels NDA receives from the system are handled by calling `TaskMasterDA`. At certain times, the Control Panels NDA sends messages to control panels.

The Control Panels NDA takes care of nearly everything necessary, including tracking controls. Control panel windows are usually just windows full of extended controls; the control panel receives a `HitCDEV` message every time the user adjusts the value of one of the controls.

**Every** control in the control panel window must be an extended control. Older, non-extended controls are not allowed; all controls **must** be created with `NewControl2`. When one of these controls is "hit," the Control Panels NDA calls the control panel with the `HitCDEV` message, the control handle, and the control ID. This allows the control panel to respond to user actions. User interface items beyond extended controls (for example, modal windows) must be handled entirely by the control panel (that is, the Control Panels NDA is not involved).

**Note:** Setting the `fInWindowOnly` bit of Pop-up menu controls is not recommended.

### The Control Panels Window

In version 1.0 of the Control Panel NDA there is a single window, and exactly one control panel is always active in a portion of that window.

With System 6, this is no longer true. Many control panel messages include "the control panel's window pointer" as one of the parameters. This is guaranteed to be the window containing the control panel's controls, but little else is guaranteed.

For example, do not draw outside the area containing your control panel's controls; do not compute other window sizes from the size of this window; and do not assume that the Control Panels NDA will offset your controls' coordinates by the same amount version 1.0 did.

Do not hard-code any window coordinates. The Control Panels NDA shifts all your controls by some amount horizontally and vertically, but this amount will not stay the same between different versions of the Control Panels NDA (it can be zero). If you draw things besides controls in the window, compute the coordinates relative to a control on the fly.

In System 6.0, each control panel gets its own window.

## Resource Fork

The Control Panels NDA opens your control panel's resource fork differently depending on whether the machine was booted from an AppleShare file server or from a local volume.

When the machine was booted from AppleShare, your resource fork is opened with read-only access so that more than one user can have your control panel open at once. When the machine was booted locally, your resource fork is opened with "as allowed" access (this means you will have read/write access if the control panel file is unlocked and was not already opened read-only by some other part of the system).

When your control panel receives the `BootCDEV` message at boot time, its resource fork is always open read-only.

## File Format

A control panel is defined by three resources (additional resources may be present). The data fork is normally empty, but a control panel that requires System 6 or later may put code in the data fork in OMF format (it's up to the control panel to determine its own pathname and use `InitialLoad2` to load code from the data fork—a control panel can find its pathname by using `GetCurResourceFile`, `GetOpenFileRefNum`, and `GetRefInfo`).

The three required resources are the CDev code resource (type `rCDEVCode`=$8018, ID=$00000001), the CDev flags resource (type `rCDEVFlags`=$8019, ID $00000001) and the CDev's icon (type `rIcon`=$8001, ID=$00000001).

It is a good idea to make sure each released version of your control panel file has a different creation date, since the system caches certain information about your control panel in the CDev.Data file. The system uses the creation date to notice that a new version of your CDev is present.

You may also want to delete the `*:System:CDevs:CDev.Data` file, if it exists, as part of your CDev installation process.

### The Icon Resource

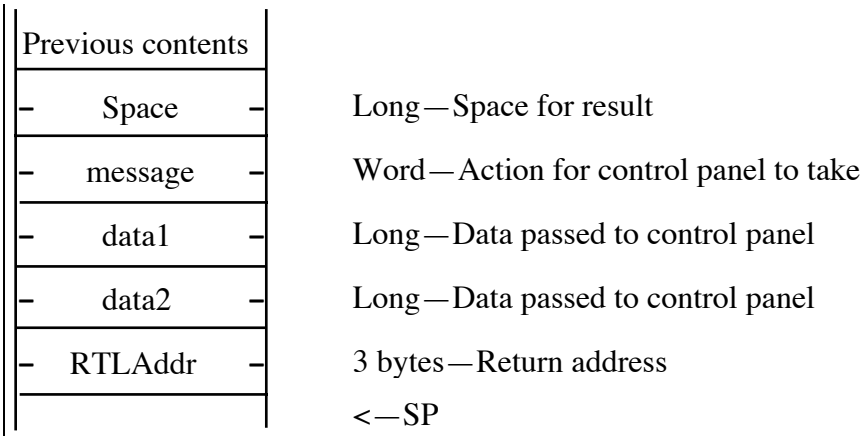
Each control panel's icon is a standard icon resource. This icon appears in the Control Panels window; it is also displayed at boot time if the CDev has any initialization code (described later).

If the icon is to be displayed during boot time (before System 6.0), it must be exactly 28 pixels wide. In 6.0, this restriction is gone, but 28 is still a nice width.

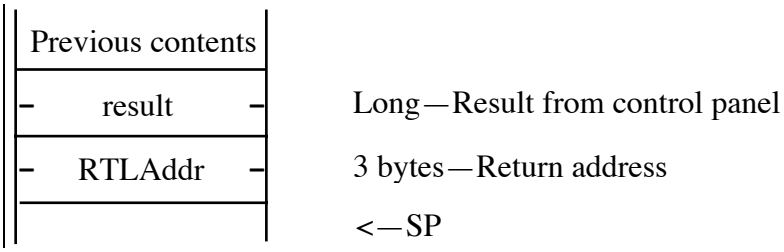
### The CDEV Code Resource

The `rCDevCode`(1) resource contains code to do the real work. A code resource has the same format as an OMF load file; the code resource converter (which is part of the system) is responsible for loading code resources. Eventually, `InitialLoad2` loads the code from memory. This process gives the `rCDevCode` resource a maximum size of 64K.

When the control panel code gets control, the stack is as follows:




The control panel must remove the input parameters from the stack and perform an `RTL`, so the calling routine may then pull the four-byte `result` parameter off the stack. Just before the control panel code RTLs, the stack must be formatted as follows:




This function, like nearly all toolbox functions, is a "Pascal" function, and may be declared in Pascal as follows:

```
function MyControlPanel(message: Integer; data1, data2: Longint): LongInt;
```

It may be declared in C as follows:

```
pascal Long MyControlPanel(message, data1, data2)
int message;
long data1, data2;
```

`Data1` and `Data2` depend on the value of `message`; `message` is the parameter that tells the CDev code what needs to be done. Higher-level language control panels can easily be arranged as a giant `switch` (or `case`, as the case may be) statement.

There are twelve defined "CDev" messages. Where parameters are not listed, they are undefined.

**Message 1: MachineCDEV**

The Control Panels NDA always compares the Apple IIGS ROM version against the minimum ROM version you put in the CDev Flags resource. If the machine's ROM version is too low, the control panel does not appear (and cannot be opened).

The `MachineCDEV` message was not supported before System 6.0. In 6.0, if the `wantMachine` bit is set in the CDev Flags resource, the control panel receives `MachineCDEV` when the user attempts to open it. The input parameters are undefined. Return a nonzero result to allow the open, or return zero to abort the open. When returning zero, you may want to display an alert explaining why the control panel cannot be opened.

**Message 2: BootCDEV**

If the `wantBoot` flag is set in the CDev Flags resource, this routine is called during the IIGS boot sequence. The parameters are undefined before 6.0. In 6.0, data1 is defined to point to a data word that is initialially zero. If you set bit 0 of this word while handling the `BootCDEV` message, the Control Panels NDA will draw an "X" over your icon (but it will not call `SysBeep2` for you; do that yourself if appropriate).

`BootCDEV` is called only during a real boot—it doesn't get control on a switch back to GS/OS from ProDOS 8. The Control Panels NDA draws the icon (from the icon resource) on the boot screen. (Before 6.0, the icon must be exactly 28 pixels wide if it is drawn at boot time.)

At best, the machine state during this call can be termed bad. QuickDraw II is not even available. Be sure to save and restore any system resources you use, including the data bank register and the direct page register.

**Note**: If your CDev expects to receive a `BootCDEV` message, it should still behave gracefully if `BootCDEV` was never received and the user attempts to use the control panel (for example, tell the user to put the file into the CDevs folder and restart the system).

In System 5.0.x, the user could drag your control panel into the `CDevs` folder and then try to use it without restarting. In System 6.0, control panels are directly launchable from the Finder, but only the ones in the `CDevs` folder receive `BootCDEV` messages.

**Message 3: Reserved**

This message is reserved for future use as a shutdown message.

**Message 4: InitCDEV**

If the `wantInit` flag is set in the CDev Flags resource, this routine is called with `data1` equal to the control panel's window pointer. When `InitCDEV` is called, `CreateCDEV` (message 7) has already been called. Controls should have been created in `CreateCDEV`, and this routine is an ideal place to initialize the controls before they are displayed.

### Message 5: CloseCDEV

This routine is called if the `wantClose` bit is set in the CDev Flags resource. If so, `CloseCDEV` is called when your control panel is closing. This is a good place to dispose of any memory you allocated or to save settings that need to be saved. The disposal of the control panel's controls is handled by the Control Panels NDA. The window pointer is in `data1`.

### Message 6: EventsCDEV

If the `wantEvents` bit is set in the CDev Flags resource, the Control Panels NDA calls this routine with `data1` as a pointer to the event record (this is an Event Manager event record, not a `TaskMaster`-style task record). The window pointer is in `data2`. The Control Panels NDA, like all NDAs, is passed events, which the it then handles by using the `TaskMasterDA` call. This routine is called before `TaskMasterDA` is called, so the control panel can change the event record before the Control Panels NDA handles it.

### Message 7: CreateCDEV

This routine is only called if the `wantCreate` bit is set in the CDev Flags resource. When called, the control panel's window pointer is in `data1`. The control panel must create any controls it has during this call. The control panel's resource fork is open during this call, so Resource Manager calls may be made (and controls may be created from resources in the control panel file). All control rectangles are relative to the upper-left corner of the part of the window a control panel owns (in 6.0 this happens to be the whole window). The Control Panels NDA handles setting the offsets of the controls to the proper place in the window. Initialization of the controls must be done in the `InitCDEV` call.

If the `wantCreate` bit is not set, the control panel must contain an `rControlList` (type=$8003) resource with ID $00000001. The Control Panels NDA automatically creates your controls from the resource.

### Message 8: AboutCDEV

If the `wantAbout` bit is set in the CDev Flags resource, the Control Panels NDA calls this routine when the user selects "Help" while your control panel's icon is selected. The window pointer to the help window is in `data1`. The Control Panels NDA takes care of the icon, author, version string and the "OK" button. The easiest way to handle help is simply to create a static text control with the help text in it.

If the `wantAbout` bit is **not** set, your control panel must have an `rControlList` resource with ID $00000002. When the user selects "Help" while your control panel's is selected, the Control Panels NDA uses this resource to create your additional About controls.

**Note:** In 6.0, when a control panel receives the `AboutCDEV` message, the Font Manager and TextEdit are always started. The Control Panels NDA can display a control panel's About box without ever opening the control panel. Making TextEdit available avoids a potential incompatibility with some control panels

(such as General) that start up TextEdit on receiving `AboutCDEV`, assuming they will have a chance to shut it back down later, on receiving `CloseCDEV`.

**Message 9: RectCDEV**

Normally, the Control Panels NDA uses the rectangle in the CDev Flags resource for the control panel's display rectangle. However, if the `wantRect` bit is set in the CDev Flags resource, this routine is called before the control panel is displayed with `data1` containing a pointer to the display rectangle. The rectangle may be modified by this routine. This gives control panels the chance to use different sized rectangles for different occasions. For example, on ROM 03, the serial port control panels show fewer parameters when the port is set to AppleTalk (since fewer parameters are changeable). In that instance, the `RectCDEV` routine changes the rectangle to be smaller.

**Message 10: HitCDEV**

If the CDev wants to know when a control has been hit, it can set the `wantHit` bit in the CDev Flags resource. When called, the handle to the control in question is in `data1` and that control's ID is in `data2`. The control panel may then take action based upon the control selection.

If you need the window pointer, you can get it from the `ctlOwner` field of the control record handle in `data1`.

**Note:** If your control panel contains any extended List controls, the toolbox automatically creates a scroll bar control for each list. These scroll bars are standard (not extended) controls; this is the exception to the rule that all control panel controls must be extended. When the user tracks the scroll bar, the `HitCDEV` `data1` parameter is a valid control handle, but `data2` is an unpredictable large value (because no control ID is available for a non-extended control). In 6.0, the control ID returned in this case is always $FFFFFFFF.

**Message 11: RunCDEV**

This routine is called if the `wantRun` bit in the CDev flags resource is set. It enables control panels to receive a call as often as the Control Panels NDA receives run events from `SystemTask` (currently **once** per second).

The control panel's window pointer is in `data1`. (This is true even before 6.0, but it was not previously documented.)

**Message 12: EditCDEV (6.0 and later)**

This routine is called if the `wantEdit` bit in the CDev flags resource is set, when the user chooses Undo, Cut, Copy, Paste, or Clear from the Edit menu (if the items have the proper item numbers), and when the user types Command-Z, -X, -C, or -V.

The control panel's window pointer is in `data2`. The low word of `data1` indicates what kind of edit operation is happening. The codes are the same as what `SystemEdit` passes to NDAs (Toolbox Reference 1, page 5-7):

$0005 Undo
$0006 Cut
$0007 Copy
$0008 Paste
$0009 Clear

All other codes are reserved for future use.

## The CDEV Flags resource

The CDEV Flags resource tells the Control Panels NDA which messages the control panel accepts. It also tells the Control Panels NDA certain things about the operating environment required for the CDev.

| | | | |
|---|---|---|---|
| flags | (+000) | **Word** | The flags word tells the Control Panels NDA which messages (defined in the discussion of the rCDevCode resource) the control panel wants:<br>Bits 15 – 12: Reserved, must be zero.<br>Bit 11: `wantEdit`<br>Control panel wants edit events.<br>Bit 10: `wantRun`<br>Control panel wants run events.<br>Bit 9: `wantHit`<br>Control panel wants control hits.<br>Bit 8: `wantRect`<br>Control panel wants rectangle messages.<br>Bit 7: `wantAbout`<br>Control panel wants "about" messages.<br>Bit 6: `wantCreate`<br>Control panel wants create messages.<br>Bit 5: `wantEvents`<br>Control panel wants event records.<br>Bit 4: `wantClose`<br>Control panel wants close messages.<br>Bit 3: `wantInit`<br>Control panel wants initialization message.<br>Bit 2: `wantShutDown`<br>Reserved, must be zero.<br>Bit 1: `wantBoot`<br>Control panel wants boot messages.<br>Bit 0: `wantMachine` |

| | | | |
|---|---|---|---|
| | | | Control panel wants machine messages (6.0). |
| enabled | (+002) | **Byte** | If this value is zero, the control panel is never activated. NOT USED. |
| version | (+003) | **Byte** | An integer version number assigned by the author. |
| machine | (+004) | **Byte** | This byte contains a minimum ROM version required for the control panel. For most control panels this is 1, but some (requiring, for example, hardware text page two shadowing) want 3 in this byte. |
| reserved | (+005) | **Byte** | Reserved, must be zero. |
| data rectangle | (+006) | **4 Words** | QuickDraw II rectangle within which the control panel is displayed. The top left of this rectangle **must** be (0,0). |
| name | (+014) | **16 Bytes** | A **string** (Pascal) giving the name of the control panel. Names longer than 15 bytes are not allowed. Note that this field requires 16 bytes regardless of the **string** length. |
| author | (+030) | **33 Bytes** | A **string** (Pascal) giving the name of the control panel's author. Names longer than 32 bytes are not allowed. Note that this field requires 33 bytes regardless of the string length. |
| version | (+063) | **9 Bytes** | A **string** (Pascal) giving the version of the control panel. Strings are typically of the format "v1.0". Version strings longer than eight bytes are not allowed. Note that this field requires nine bytes regardless of the string length. |

### Opening Additional Resource Files

The Control Panels NDA, not any individual control panel, owns the Resource Manager search path that is in effect when a control panel routine gets control. While handling a message, you may temporarily open additional resources files in the same search path, but you must close them and call `SetCurResourceFile` to its previous value before returning control to the Control Panels NDA.

There may be extra resource files in the search path that you know nothing about, so do not assume that your extra file is adjacent to your control panel's resource file in the search path.

### Color Table Swapping

Since control panels generally assume the sixteen standard 640-mode dithered colors are available, Control Panels NDA 2.0 automatically provides a standard color table whenever the "Control Panels" window or any individual control panel window is in front. (It ought to do the same thing for the Help and credits windows, but it does not.)

The color table provided in 640 mode is identical to the default 640-mode color table.

The color table provided in 320 mode provides colors almost identical to the default 640 colors. This is **not** the same as the default 320-mode color table. (See Apple IIGS Technical Note #63, Table 3.)

### Programmatic Interface to the Control Panels NDA

You can use `SendRequest` in the System 6 Tool Locator to ask the Control Panels NDA to do two things for you: Open the main window, or open a control panel from a pathname.

You must send the requests by name to "`Apple~Control Panel~`".

Request code $9001 is `cpOpenControlPanels`. `dataIn` is reserved and must be zero.

Request code $9000 is `cpOpenCDev`. `dataIn` and `dataOut` are as defined for the `finderSaysOpenFailed` request (see the Finder 6.0 documentation). You can also open a control panel by pathname by sending `finderSaysBeforeOpen`, as permitted in the Finder documentation.

## Further Reference

- *Apple IIGS Toolbox Reference*, Volumes 1-3
- System 6.0 Documentation