26th January 1996

Support Group Application Note
Number: 293
Issue: 0.11
Author: DW

Acorn

CDFS Audio SWI Calls

This document details the publically releasable list of CDFS Audio SWI calls, which are valid for CDFS versions 2.21 through 2.28. Versions of CDFS later than 2.28 may not adhere strictly to the SWI interface described herein.

Applicable Hardware :	All systems fitted with RISC OS 3.1 or later, running CDFS 2.21 or 2.28	Related Application Notes:	273: CD ROM Drives and their Handling under RISC OS Explained

Support Group
Acorn Computers Limited
Acorn House
Vision Park
Histon, Cambridge
CB4 4AE

Introduction

CDFS 2.21 is made up of three modules; the CDFS module itself, the CDFSDriver module and the CDFSFiler module. CDFS 2.28 comprises modified versions of CDFS, CDFSDriver and CDFSFiler, along with a number of demand-loading drivers designated CDFSSoft* tailored to particular models of CD ROM drive.
In general, the SWIs provided by these modules are reserved for internal use, and you must not use them in your own code, otherwise unpredictable results may occur. You should access files stored on CD ROMs using the same * commands and SWI calls with which you would access any other filing system; see the FileSwitch chapter of the RISC OS 3 Programmer's Reference Manual.

The SWI Calls

All CDFS audio SWIs conform to the RISC OS standard insofar as:

• Any registers not explicitly defined as result parameters are preserved

• The N, Z and C flags are preserved

• Errors are indicated by setting V and returning with R0 pointing at an error block. If the bit 17 clear version is called, control passes to the error handler instead.

• The SWI cannot be called from the background unless specifically documented

• All SWIs may be called from interrupt and event routines

but:

• No CDFSDriver SWIs preserve the interrupt state

• No CDFSDriver SWIs are re-entrant

CDFS SWI Calls

CDFS_ConvertDriveToDevice (SWI &41E80)
Converts a logical drive number to a physical drive number
On entry:
R0 = logical drive number
On exit:
R0 preserved
R1 = physical drive number
Use:
This call converts a logical drive number to a physical drive number. You can then use the physical drive number to produce a CDFS control block, which you have to pass when calling CDFSDriver SWIs. For details of how to construct the block, see below.

This call returns an error if the logical drive is not known.

CDFSDriver SWI Calls

CDFS Control Blocks
Before you can access a CD ROM using any of the CDFSDriver SWIs, you need to set up a CDFS control block. When you call any of the SWIs, you pass a pointer to the block in R7.

A CDFS control block describes a physical device to the CDFS_Driver module, so it can use the correct routines. Consequently there must be one block for each CD ROM drive to be accessed. Once set up, the block remains valid until the next CDFS initialisation. CDFS may alter a block's contents, but you must not do so yourself, nor must you save a block and use it in another session.

Each CDFS control block is five words long. To construct a block, you must:

1. Allocate memory for the block

2. Call CDFS_ConvertDrivetoDevice to convert that drive's logical number to a physical drive number

3. Copy the returned physical drive number into the block as follows:

Bits from physical drive number	Place in:	Word	Bits	of CDFS control block
0-2		0	0-2	
3-4		1	0-1	
5-7		2	0-2	
8-15		3	0-7	
16-31		4	0-15	

All bits of the control block not mentioned above must be initialised to zero. The whole process can be done as:

```
word0 = (drive_number AND 7)
word1 = (drive_number AND &18)>>3
word2 = (drive_number AND &E0)>>5
word3 = (drive_number AND &FF00)>>8
word4 = (drive_number AND &FFFF0000)>>16
```

Addressing Modes
Locations on a CD are specified by one-word addresses. You can use three different addressing modes:

Addressing mode	Meaning
0	Logical block addressing
1	Red Book addressing
2	Physical block addressing

Logical block addressing
A logical block address gives an offset in blocks from the start of the CD's data area.

Red Book addressing
A Red Book address gives a time offset from the CD's first frame, made up as follows from the MM:SS:FF location defined in the Red Book:

Byte	**Mnemonic**	**Meaning**
0	FF	Frame number (0-74)
1	SS	Seconds (0-59)
2	MM	Minutes (0-99)
3		Undefined

There are 75 frames per second.

Physical Block Addressing
A physical block address gives an offset in blocks from the first block on the CD.

CD_OpenDrawer (SWI &41247)
Stops the CD spinning, and opens the drawer / ejects the caddy
On entry:
R7 = pointer to CDFS control block
On exit:
R7 preserved
Use:
This call stops the CD spinning, opens the drawer and / or ejects the caddy. The call fails and returns an error if the drawer has previously been locked using CD_EjectButton or *Lock.

CD_EjectButton (SWI &41248)
Enables or disables CD ejection.
On entry:
R0 = 0 (to enable ejection) or 1 (to disable ejection)
R7 = pointer to CDFS control block
On exit:
R0, R7 preserved
Use:
This call enables or disables CD ejection. It is used by the *Lock command.

CD_EnquireAddress (SWI &41249)
Returns the current address of the head.
On entry:
R0 = addressing mode, for returned head location
R7 = pointer to CDFS control block
On exit:
R0 = head address
R7 preserved
Use:
This call returns the current address of the head. You can use it while playing audio to determine which section of audio is playing.

CD_PlayAudio (SWI &4124B)
Attempts to play a section of CD as audio, specified by address
On entry:
R0 = addressing mode of parameters
R1 = start address of play section
R2 = end address of play section
R7 = pointer to CDFS control block

On exit:
R0-R2, R7 preserved
Use:
This call attempts to play a section of CD as audio, specified by address. If the section contains non-audio information, the results are drive-dependent; you will not necessarily get an error returned, and some drives may still attempt to play the data. We therefore recommend you first use calls to CD_EnquireTrack to find out if the section you wish to play consists entirely of audio.

CD_PlayTrack (SWI &4124C)
Attempts to play a section of a CD as audio, specified by track.
On entry:
R0 = track number at which to start playing (1-99)
R1 = 255 to play to end of CD, or 254 to play to end of track
R7 = pointer to CDFS control block
On exit:
R0, R1, R7 preserved
Use:
This call attempts to play a section of a CD as audio, specified by track. If the section contains non-audio information, an error is returned.

CD_AudioPause (SWI &4124D)
Controls pausing of audio playing.
On entry:
R0 = 0 to turn off pausing, if pause was in effect, or 1 to turn on pausing if audio was playing
R7 = pointer to CDFS control block
On exit:
R0, R7 preserved
Use:
This call controls pausing of audio playing.

CD_EnquireTrack (SWI &4124E)
Returns the track range of a CD, or information on a specified track.
On entry:
R0 = 0 to return track range, or a valid track number
R1 = pointer to 5 byte block
R7 = pointer to CDFS control block
On exit:
R0, R1, R7 preserved
Use:
This call returns the track range of a CD, or information on a specified track. The information is returned in the block pointed to by R1.
If R0 = 0 on entry, the track range is returned in the block as follows:

Byte	Contents
0	First track on CD (1-99)
1	Last ttrack on CD (1-99)
2-4	Corrupted

Track numbers do not necessarily start at 1.

If R0 is a valid track number on entry, then information about that track is returned in the block:

Byte	**Contents**
0-3	Logical block address of track start (ie Mode 0 address)
4	Audio control bits specifying the type of information the track holds:

Bits	**Meaning**
0	0 indicates audio information, 1 indicates data information
1	0 indicates 4 channel sound, 1 indicates 2 channel sound
2-7	Reserved

CD_StopDisc (SWI &41252)
Stops a CD playing.
On entry:
R7 = pointer to CDFS control block
On exit:
R7 preserved
Use:
This call stops a CD playing. You can use it to prematurely stop an audio play request.

CD_DiscUsed (SWI &41253)
Returns the size of the CD currently in the drive.
On entry:
R0 = addressing mode for returned size
R1 = pointer to 8 byte block
R7 = pointer to CDFS control block
On exit:
R0 preserved
R1 corrupted
R7 preserved
Use:
This call returns the size of the CD currently in the drive. The call works with audio CDs as well as with CD ROMs. The size is returned in the block pointed to by R1 on entry.

Bytes	**Contents**
0-3	Size of CD
4-7	Size of blocks on CD in bytes (usually 2048)

Note that R1 may be corrupted on exit.

CD_AudioStatus (SWI &41254)
Returns the current audio status of the drive.
On entry:
R7 = pointer to CDFS control block
On exit:
R0 = audio status
R7 preserved
Use:
This call returns the current audio status of the drive, which may have one of the following values:

Value	**Meaning**
0	Currently playing audio
1	Currently paused
2	Reserved (has no meaning at present)
3	Audio play successfully completed
4	Error occurred during last audio play
5	Audio play has not been requested

To maintain drive independence, you should always treat values 3 and 5 in the same way.