Using File Navigation API Functions in an IACD System

Seong–Ae Jin *Internet Audio Group*

ABSTRACT

When a CD is inserted, the Internet Audio CD (IACD) system reads all of the file information – structure, size, and attributes – on the CD and stores this information to memory in a data structure. You can use this information to implement all of the functions related to any files. However, since the file system structure is included in the system core code, a system developer cannot handle this code directly. If you want to make a product based on the IACD system, you must get information about file system through the API function. This report provides an easy way to handle these API functions.

Contents

List of Figures

List of Tables

1 Introduction

When a CD is inserted, the IACD system reads the file information on the CD and stores it in memory in a data structure. This data structure contains all of the information about the files, such as hierarchy, size, name, and other characteristics. The information contained in the data structure can be used to implement all of the functions related to any files.

Generally each product has its own specification of file navigation; however, the file system is included in the internal core system code, which makes it impossible for product developers to access the code directly. The only way to manipulate this information is by using an application programming interface (API). This report provides knowledge about some basic API functions that are included in the IACD system, and how these functions can be used in the new product system code.

2 Data Structure of File System in IACD

As you already know, when a CD is inserted, the IACD system generates a file system structure that includes all the file information contained on the CD. There are two types of data structures in the IACD system. Before the API function is explained, a short review about data structures of the internal file system will be provided so that you will use these API functions correctly.

2.1 root_node Structure

When the IACD system creates a file system, it stores all the information about the file in a general data structure. For this purpose, the IACD system uses the **root_node** internal data structure.

The root_node has seven fields as defined in ISO9660.h.

Table 1. root_node Structure

Type	Name	Description
root_node *	next	Points to the next root_node
char	flag	Types of files d: directory p: parent_child node f: normal file–music file l: play list file
int	file_id	Ordered number of files
char *	name	File name
root_node *	parent_child	
root_node *	sort_Next	Points to the next node
packet_struct *	packet	Points to the structure of packet information

When the IACD system generates the file system, one root_node structure is used for storing the information about one file only. But the next field system links all of the root_nodes. This linked order becomes the basic order of the files. There is also a hierarchy between files included the directory structure. The IACD system uses two root_nodes for a directory. One is used for the storing the directory's information (the same as the file), and the other is used for the hierarchical information of the directory.

Figure 1 shows the internal structure of root_nodes. There is one directory and three files: File1, File2, and File3 (section 1). The arrows show that the **next** field links the three files in all three nodes. Another file node (section 2) is linked by the **next** field, and indicates either the start or end of the file in its directory. The node is also linked by the **parent_child** field (section 2)with the **directory** node (section 3). This additional node is called the parent_child node of the directory.

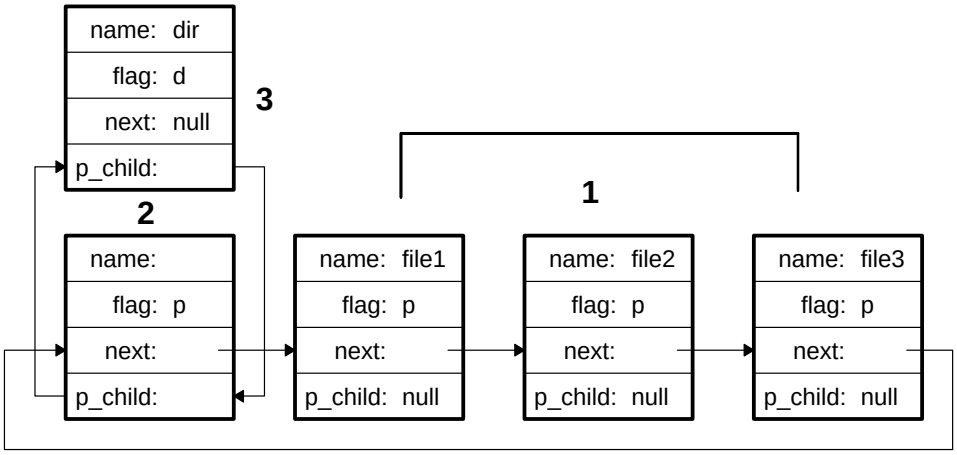

Figure 1. Structure of the File System

2.2 file_node Structure

File navigation encompasses one basic node for searching. The system always stores the value of the basic node in the global memory area. In the IACD system, the name of this node pointer is **TargetFileNode**, and it is included in the **LoadData** global variable area. The TargetFileNode is a base point for searching and playing. When you insert a CD and it begins to play music, the music becomes a TargetFileNode. When the music is stopped, the file or directory displayed in LCD becomes a Target File Node. When you want to “find the next song” on the CD, you are finding the next song from the current TargetFileNode. So, a directory or a file can be a TargetFileNode.

The type of TargetFileNode (CD or file) is a structure of the file_node. The type of file_node structure includes structure of cd_state as well as root_node; therefore, you can handle all file and CD information at any point.

There is another pointer in the file_node structure in the IACD system. The name of this pointer is **nextTargetFileNode**. After setting the TargetFileNode (e.g., song or music on a CD), the next song that plays after the TargetFileNode will be a nextTargetFileNode (or the next song). This node is used for pre-fetching data.

3 Two Navigation Methods

The IACD system has two different file types, so two navigation methods are provided. One method is the copy navigation method and the other is the non-copy navigation method. The copy navigation method system has an extra memory space in the global variable area named targetFileNode. Its address is stored in TargetFileNode, so **TargetFileNode** always points to the **targetFileNode**. When the system sets one of the files on the CD to TargetFileNode, it copies all of the field, root_node, to the targetFileNode area. When the system changes a TargetFileNode, it doesn't modify the value of the pointer named TargetFileNode. Instead, the system copies all field values of the new file to the targetFileNode. This is the basic navigation method of IACD.

This method, however, is not suitable for some situations where the developer wants to maintain a pointer without memory space for convenience. The IACD system provides another navigation method. When we set the new file in the basic file navigation mode, it doesn't copy all the field, root_node, of that file; it only modifies the pointer value. Before using this method, it is better to define a new pointer of root_node. Also, if the TargetFileNode is used in a non-copy navigation method for navigation pointers, please pay attention to an inconsistency of the pointer TargetFileNode and the real field value of targetFileNode.

4 API Functions

4.1 API Function Related to Field Value

FSIsDir

Description	Returns the flag of the file_node pointed to by the first argument. This function always checks the HasFileTree value of the CDDrive, so caution is needed when using this non-copy navigation mode.
Arguments	**pFSFile– pointer to file_node
Return Value	Returns the second field value of the file_node structure, flag.

FSRIsDir

Description	Returns the flag of the root_node pointed to by the first argument.
Arguments	**pFile – pointer to root_node
Return Value	Returns the second field value of the root_node structure, flag.

FSGetFileID

Description	Returns the value of the file_id of the file_node pointed to by the first argument. Before using this function file_d, the value should be set.
Arguments	**pFSFile– pointer to file_node
Return Value	Returns the third field value of the file_node structure, file_id.

FSRSetFileID

Description	Sets the value of the file_id of the root_node pointed to by the first argument, to the second argument.
Arguments	**pFile– pointer to root_node
Return Value	Always returns zero.

4.2 API Functions Related to Hierarchy

FSCdRoot

Description	Changes the first argument for the first file or directory in the root directory. Both navigation modes are possible.
Arguments	**pFSFile– pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Depends on the field value of the new pFSFile. When the copy navigation mode is used, all the fields of pFSFile have their specific names and functions. One of the fields of the pFSFile is a CDDrive structure and one of the fields of the CDDrive structure is HasFileTree. If HasFileTree equals 0, then the current file system is not valie and cannot change the pFSFile to the new system. It returns a 1. Another special case is if the new pFSFile is the last file of the file system it returns 1. Otherwise, it returns 0.

FSRCdRoot

Description	Changes the value of the first argument pFile, for the first file or directory in the root directory. Only non-copy navigation modes are provided.
Arguments	**pFile– pointer to root_node
Return Value	Depends on the field value of a new pFile. When the new pFile is a last file of the file system, it returns 1. Otherwise it returns 0.

FSCd

Description	Changes the value of the first argument, pFSFile, to the directory or up to its directory. It is useless if pFSFile is a file (flag of pFSFile is f). When pFSFile is a directory (flag of pFSFile is ‘d’), it sets the new pFSFile to its parent-child noded that is linked by the parent-child field. When pFSFile points a parent-child node of the file system (flag of the pFSFile is ‘p’), it sets the new pFSFile to its directory, linked by the parent-child field. Two navigation modes are also possible.
Arguments	**pFSFile– pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	One of the fields of the pFSFile is a CDDrive structure and one of the fields of of the CDDrive structure is HasFileTree. If HasFileTree is 0, then the current file system is not valid and cannot change the pFSFile to the new system. It returns 1. Depends on the flag value of the old pFSFile. When the flag of the old pFSFile is either p or d, it returns 0. Otherwise it returns 1.

FSRCd

Description	Almost the same as the FSCd function, except the type of argument. It has only one argument, pFile, whose type is root_node. It provides only non-copy navigation mode. If the flag of the old pFile is d, it sets the new pFile to the first node in the directory. If the flag of the old pFile is p, it sets the new pFile to the directory node .
Arguments	**pFile– pointer to root_node
Return Value	Depends on the old pFile's field value. It returns 0 when the flag of the old pFile is either d or p. Otherwise it returns 1.

FSDirDown

Description	Changes the value of the first argument, pFSFile, into its directory. It is useless if the first argument isn't a directory. If the flag of pFSFile is d, it sets the new pFSFile to the first node in its directory. Otherwise, it makes no difference.
Arguments	**pFile– pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Depends on the flag of the old pFSFile. If the flag is d, ir returns 0. Otherwise it returns 1.

FSDirUp

Description	Changes the value of the first argument, pFSFile, out of its directory. No matter what the flag of the old pFSFile, the function changes the first argument for the directory node, in which the old pFSFile is included. If the old pFSFile is included in the root directory, this sets the new pFSFile to the first file or directory in root. Two navigation modes are also possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	If the old pFSFile is included int he root directory, it returns 1. Otherwise, it returns 0.

4.3 API Function for Finding Files

Based on our IACD system, most of the system developer's work with the file system will be finding a new file node. The following are the search functions.

FSFindNext

Description	Changes the first argument in the pFSFile for the next file or folder node in the file system. There are two types of next nodes in the files system. One is a normal next type and it uses the next field of file_node. This is a default next type. The other is a sorted next type and it uses the sortNext field of the file_node. Please use caution when the sort next type is selected. This field should be set before using this function. Ensure the file system included in IACD system doesn't set this field value. Two navigation modes are also possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode nexttype – selection type of next mode 1: sorted next 0: normal next
Return Value	When the HasFileTree is not valid and the copy navigation mode is selected, it returns 1. It returns –1 when the next field (and normal next type is selected) or sortNextfield (and sorted next type is selected) value is NULL. The return value depends on the next type. When we use the normal next type, it returns 1 when the new pFSFile is the last file in its directory (i.e., the flag of new pFSFile is p). It also returns 1 when we use the sort next type and when the new the pFSFile is the last file in its file system (i.e., the flag of new pFSFile is d). Otherwise, it returns 0.

FSRFindNext

Description	Changes the first argument, pFile, for the next file or folder node in file system. It is almost same to FSFindNext. Its type of first argument is root_node, not file_node. Because of the type of the first argument, only non-copy navigation mode is provided. Two types of next mode are possible.
Arguments	**pFile – pointer to root_node nexttype – selection type of next mode 1: sorted next 0: normal next
Return Value	It returns –1 when the next field (and normal next type is selected) or sortNextfield (and sorted next type is selected) value is NULL. The return value depends on the next type. When we use the normal next type, it returns 1 when new pFile is the last file in the directory (i.e., the flag of new pFSFile is p). It also returns 1 when we use the sort next type and new the pFSFile is the last file in its file system (i.e., the flag of new pFSFile is d). Otherwise, it returns 0.

FSFindPrev

Description	Changes the first argument, pFSFile, for the previous or folder node in file system. There are two types of next modes in the file system. One is a normal next type and uses the next field of file_node. This is the default next type. The other is a sorted next type and uses the sortNext field of the file_node. Use caution when selecting the sort next type. This field should be set before using this function. Ensure that the file system included in the IACD system doesn't set this value. Two types of navigation mode are possible.
Arguments	**pFSFile – pointer to file_node copy – Selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode nexttype – selection type of next mode 1: sorted next 0: normal next
Return Value	When the HasFileTree is not valid and the copy navigation mode is selected, it returns 1. It returns –1 when the next field (and normal next type is selected) or sortNextfield (and sorted next type is selected) value is NULL. The return value depends on the next type. When we use the normal next type, it returns 1 when new pFile is the last file in the directory (i.e., the flag of new pFSFile is p). It also returns 1 when we use the sort next type and the new pFSFile is the last file in its file system (i.e., the flag of new pFSFile is d). Otherwise, it returns 0.

FSRFindPrev

Description	Changes the first argument, pFile, for the previous file or folder node in the file system. It is almost the same as FSFindPrev. Its type of first argument is root_node, not file_node. Because of the type of the first argument, only non-copy navigation mode is provided. Two next modes are possible.
Arguments	**pFile – pointer to root_node nexttype – selection type of next mode 1: sorted next 0: normal next
Return Value	It returns –1 when the next field (and normal next type is selected) or sortNextfield (and sorted next type is selected) value is NULL. The return value depends on the next type. When we use the normal next type, it returns 1 when new pFile is the last file in the directory (i.e., the flag of new pFSFile is p). It also returns 1 when we use the sort next type and the new pFSFile is the last file in its file system (i.e., the flag of new pFSFile is d). Otherwise, it returns 0.

FSFindFirst

Description	Changes the first argument, pFile, for the first file or folder in the directory. Only copy navigation mode is possible.

Arguments	**pFSFile – pointer to file_node
Return Value	Returns the same value as FSFindNext.

FSFindFile

Description	Changes the first argument, pFile, for the new file whose path is defined by the second argument. Only copy navigation mode is possible.
Arguments	**pFSFile – pointer to file_node path – path of new pFSFile Example of path is “Eric Clapton\Great Hits\Stairway.mp3”
Return Value	When the HasFileTree is not valid or the file cannot be found, it returns 1. It returns 0 when the file has been found.

FSSearchFileId

Description	Finds the node whose file_id is a second argument in the file system. It then changes the first argument to the node we have found. Both navigation modes are possible.
Arguments	**pFile – pointer to root_node file_id – file_id value you want to find copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	When the function fails to find the file, it returns 1. Otherwise, it returns 0.

FSRSearchFileId

Description	Finds the root_node whose file_id is a second argument in the file system. It then changes the value of the first argument. Only non-copy navigation mode is possible.
Arguments	**pFile – pointer to root_node file_id – file_id value you want to find
Return Value	When the function fails to find the file, it returns 1. Otherwise, it returns 0.

FSFindFirstSong

Description	Changes the first argument, pFSFile, for the first file, not the directory in the whole file system. If the first node in the file system is a directory, it finds the file in that directory. Two navigation modes are possible.
Arguments	**pFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Returns the same value as FSFindNext.

FSFindFirstSongFromHere

Description	Changes the first argument, pFSFile, for the first file from its current position. If the next node is a directory, it finds the first file in that directory. Two navigation modes are possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Returns the same value as FSFindNext.

FSFindFirstSortedSong

Description	This changes the first argument, pFSFile, for the first file from the of current pFSFile position. In this function, we use the sixth field of file_node when we find the next node pFSFile. The name of the field is sortNext. Before using this function, please check whether or not this field is set. In general, this field doesn't have a value. Only non-copy mode is possible.
Arguments	**pFSFile – pointer to file_node
Return Value	Has no return value.

FSFindNextInAll

Description	Changes the first argument, pFSFile, for the next file or folder in all file systems. Its function is almost the same as FSFindNext. The only difference is when the flag of the new pFSFile is either 'd' or 'p', it moves into or out of its directory once more. Two navigation modes are possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	When the flag of the new pFSFile is r, it means this is the end of the file in the file system. It returns 1. Otherwise, it returns 0.

FSRFindNextInAll

Description	Changes the first argument, pFile, for the next file or folder in all file systems. Its function is almost the same as FSRFindNext. The only difference is when the flag of the new pFile is either 'd' or 'p', it moves into or out of its directory once more. Only non-copy navigation mode is possible.
Arguments	**pFile – pointer to root_node
Return Value	When the flag of the new pFile is r, it means this is the end of the file in the file system. It returns 1. Otherwise, it returns 0.

FSFindPrevInAll

Description	Changes the first argument, pFSFile, for the previous file or folder in all file systems. Its function is almost the same as FSFindPrev. The only difference is when the flag of the new pFSFile is either ‘d’ or ‘p’, it moves into or out of its directory once more. Two navigation modes are possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	When the flag of the new pFSFile is r, it means this is the end of the file in the file system. It returns 1. Otherwise, it returns 0.

FSFindNextInDir

Description	Finds the next file only in the directory, and changes the pFSFile to it. If there is no next file in the directory, it doesn’t change the value of the pFSFile. Two navigation modes are possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Returns a 1 when the old pFSFile is the last file in that directory. Otherwise, it returns 0.

FSRFindNextInDir

Description	Finds the next file only in the directory, and changes the pFile to it. If there is no next file in the directory, it doesn’t change the value of the pFile. Only non-copy navigation mode is possible.
Arguments	**pFile – pointer to root_node
Return Value	Returns a 1 when the old pFile is the last file in that directory. Otherwise, it returns 0.

FSFindPrevInDir

Description	Finds the previous file only in the directory, and changes the pFSFile to it. If there is no previous file in the directory, it doesn’t change the value of the pFSFile. Two navigation modes are possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Returns a 1 when the old pFile is the first file in that directory. Otherwise, it returns 0.

FSFindNextInSort

Description	Finds the next file in the sortNext list and changes the pFSFile to it. If there is no next file in the directory, it doesn't change the value of the pFSFile. Two navigation modes are possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Returns the same value as FSFindNext.

FSFindPrevInSort

Description	Finds the previous file in the sortNext list and changes the pFSFile to it. If there is no next file in the directory, it doesn't change the value of the pFSFile. Two navigation modes are possible.
Arguments	**pFSFile – pointer to file_node copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Returns the same value as FSFindPrev.

FSRandomTrackChangeInAll

Description	Changes the first argument for the next nth file from the current position of the first argument in all the file system. The value of the jumping track is provided by the third argument. For this function, all the node files should have their own file_id value. Because this function searches the new file based on the file_id value, check whether or not this file_id is set before using this function. If the first argument is not a file, before searching, the function finds the first file from its current position; then it searches for the new file from there. Both navigation modes are possible and two next types are provided.
Arguments	**CurrentFileNode – pointer to file_node nexttype – selection type of next mode 1: sorted next 0: normal text number – value of jumping track copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	There is no return value.

FSRandomTrackChangeInDir

Description	Changes the first argument for the next nth file from the current position of the first argument only in its directory. The value of the jumping track is provided by the second argument. When the function finds the files, it searches only in the current directory, not nested. This means that if there are folders int he current directory, each folder counts as one, no matter how many files belong to that folder. If the value of thye third argument is larger than the number of files from the first argument, nothing happens. Both navigation modes are possible and two next types are possible.
Arguments	**CurrentFileNode – pointer to file_node number – value of jumping track nexttype – selection type of next mode 1: sorted next 0: normal text copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Return value is the same as FSFindNext.

4.4 Other API Functions

FSFileInit

Description	Initializes the first argument to the field value of the second argument. The file_node structure is composed of two parts. One is node of file and the other is a pointer of the CDDrive. In this function we set the field of node by the first node of the file system, that is, the first file or directory node in the root directory. The pointer of the CDDrive is set by the second argument.
Arguments	**pCDFileNode – pointer to file_node *pCDDrive – pointer to cdstate
Return Value	When the parent _child field of the second argument's Root is NULL, it returns -1, which means that the file system structure is not valid. If there is a valid file structure but there is no file, it returns 1. Otherwise it returns ISO9660_NO_ERROR.

FSGetDiscStatistics

Description	Initializes the field value of the second argument to its appropriate value by using the first argument. The discstat structure has three fields: numFiles, numFolders, and totalSize. The set numFiles function sets the total number of files in the CD. Similarly, the set numFolder function, along with the totalSize function, sets the total number of folders in CD.
Arguments	*pCDDrive – pointer to cdstate *State – pointer to discstat
Return Value	If the file system in the CDDrive is not valid it returns 1; otherwise, else it returns 0.

FSGetDecoderStatistics

Description	Initializes the field value of the second argument to its appropriate value using the first argument. The decoderStat structure has a three fields, extentions, numFiles, and totalSize. The function set the numFiles includes the total number of files whose extension is the same as the first field of decoderStat. The total size of the file with the same extension is also added.
Arguments	*pCDDrive – pointer to cdstate *State – pointer to discstat
Return Value	If the file system in the CDDrive is not valid it returns 1; otherwise, else it returns 0.

FSMove

Description	Copies the value of the second argument to the first argument. Therefore, the two arguments will point to the same node in the file system.
Arguments	**pReturn – pointer to file_node **pFile – pointer to file_node
Return Value	When the flag of the second argumennt is 'p' it returns 1. Otherwise it returns 0.

FSRMove

Description	Copies the value of the second argument to the second argument. Therefore, the two arguments will point to the same node in the file system.
Arguments	**pReturn – pointer to file_node **pFile – pointer to root_node
Return Value	When the flag of the second argument is ‘p’ it returns 1. Otherwise it returns 0.

FSNodeDistance

Description	Calculates the distance between two file nodes. Two next modes are possible, but there is a restriction if the normal next mode is selected. The node provided by arguments should be in the same directory. Also, the folder node is considered to be one node.
Arguments	**Currentfile_node – pointer to file_node **Endfile_node – pointer to file_node nexttype – selection type of next mode 1: sorted next 0: normal next
Return Value	If the function cannot count the distance, it returns -1. Otherwise, it returns its distance.

FSNumberOfFileAfterNode

Description	This function calculates the distance from the first argument to the end of the node. Depending on the next mode type, the end node is different. When we select th enormal next mode, the end node will be an end file node in its directory where the first argument is included. And if the sort next mode is selected, the last file in the sortNext field will be an end node.
Arguments	**Currentfile_node – pointer to file_node nexttype – selection type of next mode 1: sorted next 0: non-copy navigation mode copy – selection flag of navigation mode 1: copy navigation mode 0: non-copy navigation mode
Return Value	Returns its distance.

FSRFMove

Description	The second argument is a pointer to root_node. Generally, this is a pointer to the real node of the file system. In the system, there is a base file_node, TargetFileNode, where sometimes you must copy the result of temporary navigation. This function modifies the first argument to point to the same node pointed to by the second argument.
Arguments	**pReturn – pointer to file_node **pFile – pointer to root_node
Return Value	Returns 1 when the flag of the second argument is ‘p’. Otherwise, it returns 0.

FSFRMove

Description	The second argument is a pointer to file_node. This function is useful when you need to initialize the temporary navigation pointer with the system base file_node, TargetFileNode.
Arguments	**pReturn – pointer to root_node **pFile – pointer to file_node
Return Value	Returns 1 when the flag of the second argument is ‘p’. Otherwise, it returns 0.

FSRGetRealNode

Description	The first argument is a pointer to file_node. In our IACD system, the only structure of file_node is TargetFileNode. The TargetFileNode always points to the targetFileNode, and this targetFileNode is in the global memory area, not in real memory space stored in the file system. Sometimes you need the real address value of the current TargetFileNode and this function provides the real address of the current first argument in the file system memory.
Arguments	**pFSile – pointer to file_node
Return Value	Always returns 0.

5 References

TMS320C54x DSP CPU and Peripherals Reference Set Volume 1 (SPRU131).

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265