

## Flashdisk Support with the Rabbit 3000

PCMCIA/PC Card flashdisks and CompactFlash modules are inexpensive, commonly available devices used for storing large quantities of data (currently up to 800 MB). Most of these devices have a common ATA interface, and are quite simple to interface to a Rabbit 3000-based board. This technical note describes one method of interfacing a PCMCIA flashdisk to a Rabbit-based board, including a method of retaining the FAT-based file system commonly placed on these devices to allow direct PC access of data written by the Rabbit device.

CompactFlash devices have an interface that is a subset of the PCMCIA interface, and can be supported with minimal changes to the description in this technical note (or without changes via an inexpensive CompactFlash-to-PCMCIA converter).

The code described here was tested with SanDisk 4MB and 8MB flashdisks, and a SanDisk 8MB CompactFlash module with a PCMCIA converter.

### Hardware Interface

The PCMCIA/PC Card standard provides for three access modes from a processor: memory-mapped, I/O-based, and “True IDE.” These options are provided to support the external I/O access methods of various microprocessors, and several are possible for use with a Rabbit CPU.

This technical note describes the True IDE method of accessing the flashdisk, which is enabled by connecting the /OE (PCMCIA pin 9) signal to ground. Accesses will then occur via the /IOWR and /IORD signals.

**NOTE:** Hot-swapping is not officially supported in this mode.

The connections necessary to access a flashdisk with the Rabbit 3000 CPU are described in Table 1. Only three address lines and eight data lines are used in this implementation (although 16-bit data transfers are possible as well). Fourteen signal lines from the Rabbit are necessary, although a number of other pins on the PCMCIA interface must be connected to either ground or  $V_{CC}$ . Most flashdisks are capable of running with  $V_{CC}$  at either 3.3 V or 5 V.

**Table 1. PCMCIA/Rabbit 3000 interface, required connections**

Pin(s)	PCMCIA Signal	R3000 Signal	Description
1, 34, 35, 68	GND	GND	
17,51	V <sub>CC</sub>	V <sub>CC</sub>	
2	D3	PA3	
3	D4	PA4	
4	D5	PA5	
5	D6	PA6	
6	D7	PA7	
7	/CS0	PE0	Described as /CE1 in modes other than True IDE.
8	A10	GND	
9	/OE	GND	Enables True IDE mode.
11	A9	GND	
12	A8	GND	
15	/WE	V <sub>CC</sub>	Not used in True IDE mode.
23-27	A3-A7	GND	
27	A2	PB4	
28	A1	PB3	
29	A0	PB2	
30	D0	PA0	
31	D1	PA1	
32	D2	PA2	
42	/CS1	V <sub>CC</sub>	Described as /CE2 in modes other than True IDE.
44	/IORD	/IORD	
45	/IOWR	/IOWR	
56	/CSEL	GND	
58	/RESET	GND	
61	/REG	V <sub>CC</sub>	

## Software Support

The library `ATAINTERFACE.LIB` provides a simple interface to the flashdisk. There are only three functions available at the user level; they are described below.

### **`void ataInit()`**

This function initializes the flashdisk by enabling LBA addressing and 8-bit accesses.

### **`void ataWriteSector(unsigned long sector, char *buffer);`**

This function writes one sector's worth of data to the sector number provided. Most flashdisks use 512-byte sectors.

### **`void ataReadSector(unsigned long sector, char *buffer);`**

This function read one sector's worth of data into the buffer provided. Most flashdisks use 512-byte sectors.

A sample program, `FLASHDISK.C`, demonstrates the use of these functions by writing raw data to multiple sectors on a flashdisk, then reading the data back.

## Access To Data From a PC

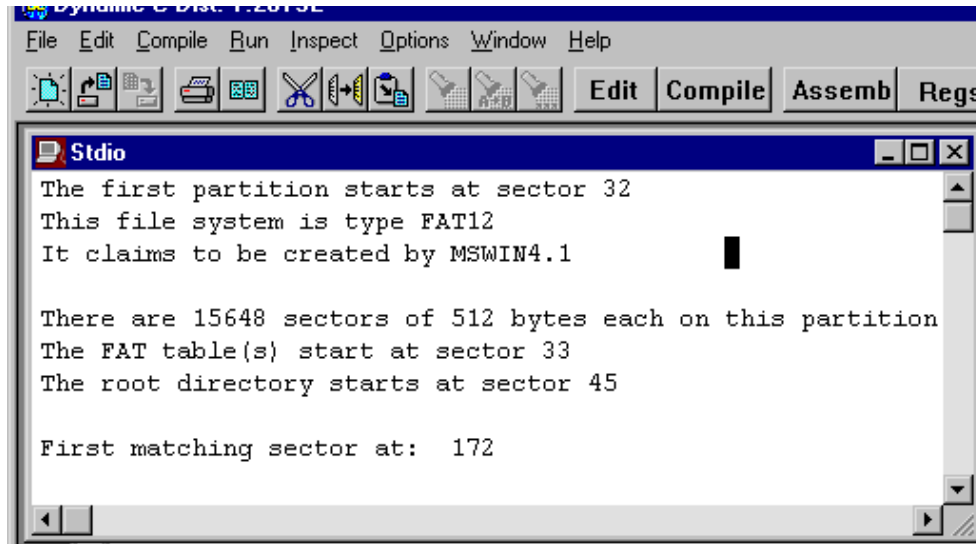
Direct access from a PC to the data written on a flashdisk is obviously very desirable. However, even though the FAT-based file systems common to PCs running Microsoft Windows are not that sophisticated, fully supporting them would take up significant resources on an 8-bit single-board computer. A simpler method of reading data written to a flashdisk by a Rabbit-based device is described below.

The FAT-based file systems (FAT12, FAT16, and FAT32) supported by Microsoft Windows have a simple layout, which is exactly predictable when starting with a freshly-formatted device. In particular, if a single large file is created after the format, the starting sector can be found and the file should continue sequentially through the sectors on the device. In addition, there is no checksum calculated for the sectors of a file, so the actual data in the sectors can be changed by separate accesses, such as from a Rabbit-based device.

Here is the sequence to produce a flashdisk format usable by the Rabbit:

1. Format the flashdisk.
2. Create a dummy file on your PC's hard drive almost as large as the free space on the flashdisk. The program `MAKEDUMMY.EXE` (with source) is provided with this technical note for this purpose. The dummy file, `DUMMY.TXT`, will be located in the same directory as `MAKEDUMMY.EXE`. Note that a particular string must be written at the start of this file to find its first sector.
3. Copy `DUMMY.TXT` over to the flashdisk.

4. Remove the flashdisk and attach it to your Rabbit-based board. Run the `FINDFIRSTSECTOR.C` program provided with this technical note. This program returns the information similar to the following:



```
Dynamic C Disk: 1.2013L
File Edit Compile Run Inspect Options Window Help
[Icons] Edit Compile Assemb Regs
Studio
The first partition starts at sector 32
This file system is type FAT12
It claims to be created by MSWIN4.1

There are 15648 sectors of 512 bytes each on this partition
The FAT table(s) start at sector 33
The root directory starts at sector 45

First matching sector at: 172
```

Write down the value for “First matching sector.” This is the first sector of the large file you created, in this case, 172.

5. When you write code for the Rabbit board, use that sector as the starting sector of the data you store on the flash device.
6. When you wish to read the data, install the flash card in your PC and copy the file back to your hard drive. If necessary, write an application to pull relevant blocks of data from the file.

The sample program `FLASHFAT.C` provides a simple example of writing some data to a flashdisk with a FAT-based file system via the method above. Note that any addition or deletion of files on the flashdisk may make the `FINDFIRSTSECTOR.C` results invalid —reformat the flashdisk if you make any changes to files on it.

### Multiple Files

By copying several smaller files after the flashdisk format with different strings at the start, multiple files (starting at different sector numbers and each continuing sequentially) can be accessed. Change the search string in `FINDFIRSTSECTOR.C` to the appropriate value for each file and use the resulting sector number to access that file. As above, any addition or deletion of files after the initial copies may make the results from `FINDFIRSTSECTOR.C` invalid.

#### Z-World, Inc.

2900 Spafford Street  
Davis, California 95616-6800  
USA

Telephone: (530) 757-3737  
Fax: (530) 757-3792

[www.zworld.com](http://www.zworld.com)

#### Rabbit Semiconductor

2932 Spafford Street  
Davis, California 95616-6800  
USA

Telephone: (530) 757-8400  
Fax: (530) 757-8402

[www.rabbitsemiconductor.com](http://www.rabbitsemiconductor.com)