CTAN008: Life-cycle Management for Cactus Flash Memory cards and drives

Covered Products: CF / PC Card / SSD / DOM with product code -203 or -204 under Microsoft Windows NT/2000/XP

1. Introduction

Life-cycle management for flash-based storage media is important because of limited life for NAND flash devices. Advanced ECC error detection/correction, dynamic wear-leveling and media defect management can alleviate the effects of continuous programming and erase on the physical flash media. However, these techniques cannot extend the physical media life indefinitely, therefore Cactus Technologies has implemented life-cycle management technologies for customers to determine the optimum time to renew existing Cactus Technologies industrial-grade flash storage cards and drives.

2. The Problem

The intelligent controller on-board Cactus Technologies industrial-grade flash cards/drives reserve a percentage of NAND flash blocks as spare blocks which can be used for bad sector/block remapping during the life of the device. We consider a flash card or drive as having reached its end of useful life when all the spare blocks have been used for defect sector/block remapping. With no available spare blocks for defect management, any unrecoverable ECC read/write errors will be considered by the host as a physical defect. For the end user, this will show up as groups of bad sectors on the flash card/drive. To many applications, when bad sectors appear from the physical drive, this implies incurred data losses due to unrecoverable read errors, and it is unacceptable from a data reliability point of view.

From the previous overview, the criteria for end-of-life for Cactus Technologies industrial-grade flash memory cards and drives may be defined as the moment where the number of available spare blocks for defect remapping has reached less than 5% of total reserved blocks. The flash card/drive will continue to function, but its data reliability may deteriorate as the number of spare blocks for defect remapping decreases.

3. The Solution

Host designers can take advantage of the integral life-cycle management features on the Cactus Technologies intelligent controller to determine the best time to renew cards/drives before they reached the end-of-life scenario to ensure maximum data integrity.

Cactus Technologies Limited provides a Windows NT – based OS driver and application programming interface (API) for host designers to determine the number of remaining spare blocks and a simple TRUE/FALSE status to determine whether a card/drive should be renewed. The interface works with the -203/-204 models and its programming interface is described below.

The Windows NT – based Cactus EOL library consists of an low-level driver to allow direct access to Cactus Technologies devices connected to the ATA bus and PCMCIA-ATA adapters under Windows NT/2000 and XP, and an user-level DLL for host designers to obtain the end-of-life information from supported Cactus Technologies flash cards/drives connected to the system via either PCMCIA-ATA or True IDE mode connections. Due to the use of low-level driver for direct disk access, administrator-level access is required to use the Cactus EOL library.

The Cactus EOL DLL library exports the following API functions for host designers to obtain end-of-life information in terms of initial and remaining spare blocks on specified ATA interfaces.

- `CactusSetATADeviceEnum()`
- `CactusGetNumOfSpareBlocks()`

The DLL API function call `CactusGetNumOfSpareBlocks()` may be used to query for the end-of-life information from Cactus Technologies CF and PCMCIA flash cards operating under either True IDE or PCMCIA ATA mode, and multiple products may be queried on different ATA port addresses. By default, the master and slave devices on the following port addresses will be queried: 0x1F0, 0x170, 0x1F8, 0x168.

The `_ATACACTUSSPAREBLOCKSINFO` data structure contains the following life-cycle management information:

1. Number of initial spare blocks on the card/drive
2. Number of remaining spare blocks on the card/drive
3. Card service status (TRUE/FALSE) determined as the 5% remaining spare block threshold.

Host designers may use the card service status for determining the optimum time to renew existing cards / drives, or utilize the information to their custom requirements.

To change the default port addresses, the `CactusSetATADeviceEnum()` function may be called to specify the I/O port address for all ATA (both native and PCMCIA-ATA) controller on the system that may be connected to Cactus Technologies flash cards/drives.

The `_ATADEVICEENUMINFO` data structure allows the host designer to specify non-standard ATA ports or PCMCIA-ATA port addresses. The data structure contains a 16-bit port address and 16-bit device number to indicate master/slave configuration.

The appendix of this application note has sample code to show the DLL usage on a Cactus Technologies industrial-grade CF card.

The CactusEOL Windows library are available from Cactus Technologies sales representatives worldwide.

4. Version History

Version	***Date***	***Change***
1.00	December 12, 2007	Initial Version
1.01	June 3, 2008	Minor edits

Appendix: Demonstration code for Cactus EOL library for Windows

```
#undef  UNICODE
#include <windows.h>
#include <stdio.h>
#include "CactusEOLDll.h"
#define ATA_MAX_LOCAL_DEVICE_NUMBER 16

DWORD QueryEOL(void)
{
    HMODULE    hModule;

    DWORD    status;
    int      cactus_device_num;
    int       n;

    _ATACACTUSSPAREBLOCKSINFO   spare_blocks_info[ATA_MAX_LOCAL_DEVICE_NUMBER];
    _ATADEVICEENUMINFO   device_info[ATA_MAX_LOCAL_DEVICE_NUMBER];
    __CactusGetNumOfSpareBlocks CactusGetNumOfSpareBlocks;
    __CactusSetATADeviceEnum CactusSetATADeviceEnum;

    hModule = LoadLibrary( "CactusEOLDll.dll");

    device_info[0].port   = 0x1f0;
    device_info[0].device = 0x00;
    device_info[1].port   = 0x1f0;
    device_info[1].device = 0x01;
    device_info[2].port   = 0x170;
    device_info[2].device = 0x00;
    device_info[3].port   = 0x170;
    device_info[3].device = 0x01;
    // a PCMCIA IDE/ATAPI controller is present on this I/O port address as displayed Device
    // Properties page under Device Manager
    device_info[4].port   = 0xFFF0;
    device_info[4].device = 0x00;
    device_info[5].port   = 0xFFF0;
    device_info[5].device = 0x01;
    // this is an optional tertiary IDE port
    device_info[6].port   = 0x1E8;
    device_info[6].device = 0x00;
    device_info[7].port   = 0x1E8;
    device_info[7].device = 0x01;

    if( hModule != NULL )
    {
        CactusGetNumOfSpareBlocks = (__CactusGetNumOfSpareBlocks)GetProcAddress(
               hModule,
               "CactusGetNumOfSpareBlocks"
        );

       CactusSetATADeviceEnum = (__CactusSetATADeviceEnum)GetProcAddress(
               hModule,
               “CactusSetATADeviceEnum”
        );

       if( CactusSetATADeviceEnum != NULL )
        {
                status = CactusSetATADeviceEnum( 8, device_info );
                if (status)
                    return 1;
        }

        if( CactusGetNumOfSpareBlocks != NULL )
        {
                status =  CactusGetNumOfSpareBlocks( &cactus_device_num, spare_blocks_info );
```

```
        }
        FreeLibrary( hModule );
    }

    if( status ) // get spare blocks fail
    {
        printf( ">>  Get spare blocks error, the error code is 0x%x\n",status );
    }
    else // get spare blocks success
    {
        if( cactus_device_num == 0 )
        {
            printf( ">> There is no Cactus memory products installed\n",status );
        }

        for( n=0; n<cactus_device_num; n++ )
        {
                 printf( ">>CactusDevice[%d], the info as follows:\n", n+1 );
                 printf( ">>Port=0x%x, Device=%d, Initial Spare Blocks=%d,
                           Current Spare Blocks=%d\n",
                           spare_blocks_info[n].port,
                           spare_blocks_info[n].device,
                           spare_blocks_info[n].initial_spare_blocks,
                           spare_blocks_info[n].current_spare_blocks,
                           spare_blocks_info[n].service_status
                       );

                 if( spare_blocks_info[n].service_status == 0 )
                 {
                     printf( " >>Service Status =%s\n"," Need to be replaced now");
                 }
                 else
                 {
                     printf( " >>Service Status =%s\n"," OK ");
                 }
         }
    }
    return 0;
}
```