GR-USB/HOST#
マスストレージクラスドライバ
ユーザーズガイド

第 1.02 版

2015 年 12 月

株式会社グレープシステム

[注意事項]

はじめに

本書は、GR-USB/HOST#におけるマスストレージクラス（以下、MSC）ドライバの概要や導入手順、および使用方法について記述したものです。

改訂履歴

Rev.	日付	改訂内容
1.00	2008 年 7 月	初版リリース
1.01	2008 年 10 月	全般 ・以下のコールバックに関するメンバ名変更による修正 grp_msc_reg 構造体 pfnMscEvCallback ← pfnCallback 6.2.2 章 ・構造体のメンバ名変更によるサンプルアプリケーションの変更に伴う、説明の修正
1.02	2015 年 12 月	定義値の初期値を変更

用語

用語	説明
デバイス ID	USB デバイスアドレスとユニークな数値を組み合わせた弊社ドライバ独自のデバイス識別子
ATAPI	AT Attachment Packet Interface　サブクラスの一つ
BOT	Bulk Only Transport　プロトコルの一つ
CB	CBI (with no command completion interrupt)　プロトコルの一つ
CBI	Control/Bulk/Interrupt Transport　プロトコルの一つ
CMEM	共通メモリ管理モジュールの略　弊社ドライバ独自の処理でキャッシュやバウンダリ問題などの対応に利用される
CRC	Cyclic Redundancy Check の略
EOP	End-of-Packet の略
FS	Full Speed（max 12Mbps）の略
FSIF	File System InterFace の略　弊社 MSC ドライバ上に存在するモジュール
HC	Host Controller の略
HCD	Host Controller Driver の略
HCM	Host Controller Driver Modules の略
HS	High Speed（max 480Mbps）の略
LS	Low Speed（max 1.5Mbps）の略
MSC	マスストレージクラスの略
OVC	Over Current の略　過電流のこと
PID	Packet ID の略
RHUB	Root HUB の略　ホストコントローラのポートを監視/制御する
SCSI	Small Computer System Interface の略　サブクラスの一つ
SFF-8070i	ATAPI for Floppies　サブクラスの一つ
UFI	USB Floppy Interface の略　サブクラスの一つ
USB	Universal Serial Bus の略
USB マスストレージ統合キット	弊社が提供するファイルシステムから USB ホストコントローラドライバまでのパッケージ製品の名称

参照ドキュメント

以下のドキュメントも併せて参照ください。

MSC

『GR-USB HOST# MSC API 仕様書』

GR-VOS

『GR-VOS ポーティングガイド』

GR-USB/HOST#、CMEM

『GR-USB HOST# ユーザーズガイド』

仕様書

『Universal Serial Bus Specification Revision 2.0』

『Universal Serial Bus Mass Storage Class Bulk-Only Transport Revision 1.0』

『Universal Serial Bus Mass Storage Class Control/Bulk/Interrupt(CBI) Transport Revision 1.1』

目次

1 MSCドライバの概要

MSC（マスストレージクラス）ドライバの概要について説明します。

1.1 MSCドライバとは

MSC ドライバとは、USB のクラス仕様で定義されたマスストレージクラスに準拠したドライバで、主に USB メモリや USB ハードディスクなどのストレージデバイスにアクセスするためのものです。本ドライバでは、接続された機器のプロトコル/サブクラスを意識することなくアクセスすることが可能です。

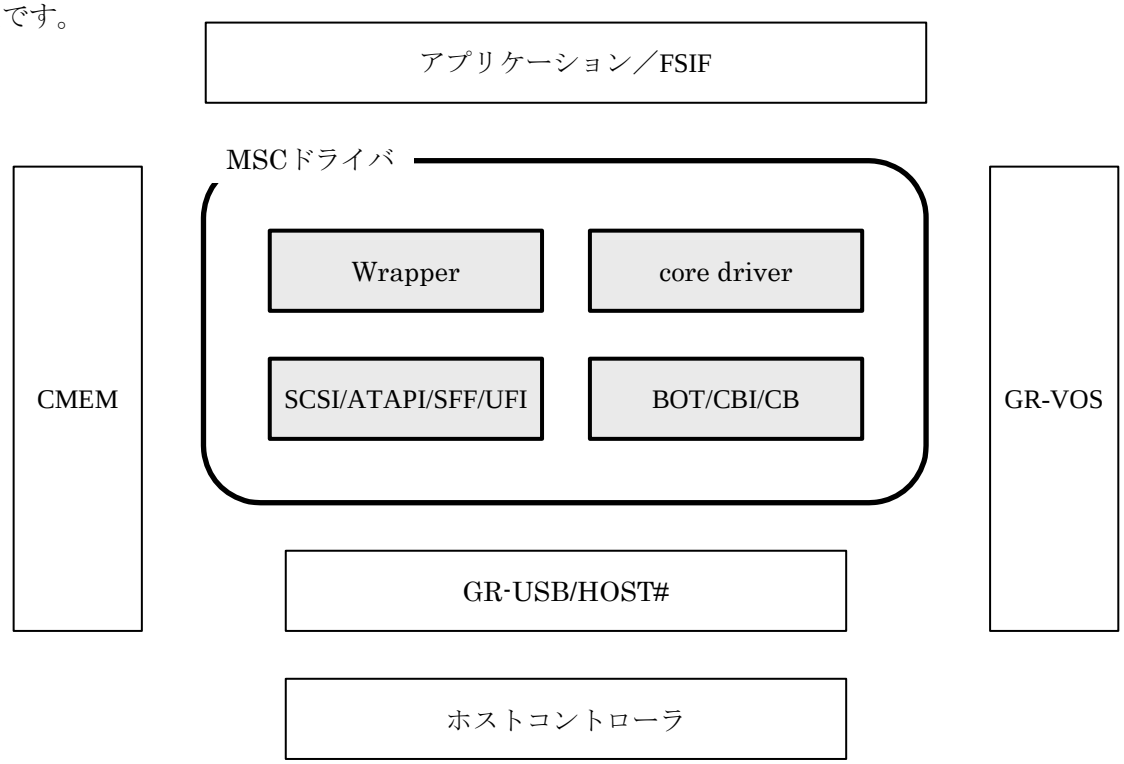

図 1-1 MSC ドライバの概要

（1）BOT/CBI/CB

本ドライバでサポートしている通信プロトコルです。

（2）SCSI/ATAPI/SFF/UFI

本ドライバでサポートしているサブクラスです。

（3）core driver

MSC ドライバのコアドライバです。

初期化処理や登録処理など上位アプリケーションへの API を提供します。

（4）Wrapper

上位アプリケーションからサブクラスを意識することなくコマンドを実行するためのラッパー関数を提供します。

1.2 他モジュールとの関連

以下に、MSC ドライバと他モジュールとの関連図を示します。

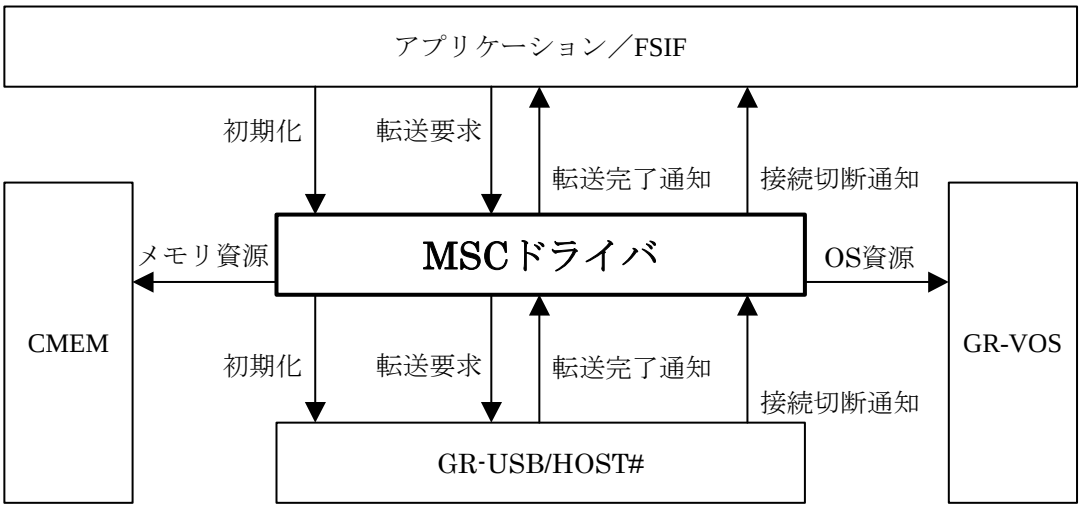

図 1-2 MSC 関連図

（1）GR-VOS

仮想 OS モジュールです。
Grapeware は全て、GR-VOS の API を介して OS の資源を使用します。
使用する OS、環境向けにポーティングする必要があります。

（2）GR-USB/HOST#

USB に関する通信要求、ドライバ制御要求の API を提供するモジュールです。
使用する USB ホストコントローラ向けにポーティングする必要があります。

（3）CMEM

CMEM（共通メモリ管理モジュール）とは、GR-USB/HOST#で用いるスケジューリングリスト、制御データ・バッファ、アプリケーション・データ・バッファの領域を管理することを目的にした、共通インターフェースです。

（4）アプリケーション/FSIF

アプリケーションとは、本ドライバの API を利用しユーザーの作成されるソフトウェアを指します。
また FSIF とは、弊社が提供する USB マスストレージ統合キットのモジュールであり、本ドライバと GR-FILE の間に位置します。そして、GR-FILE に対してデバイスドライバの機能を提供します。

1.3 機能

MSC ドライバは、以下の機能を提供します。

（1）初期設定に関する API

MSC を使用するアプリケーションに対し、初期化処理およびデバイスの登録処理（接続の通知をあげてよいプロトコル-サブクラスの組み合わせを登録します）を行うための API を提供します。

（2）基本機能

接続された MS デバイスに対する、オープン/クローズ処理やコマンドに対する実行/キャンセルなどを行うための API を提供します。

BOT にて必要な、GetMaxLUN の機能やサブクラスを取得する機能なども提供します。

（3）コマンド

接続された MS デバイスのプロトコルやサブクラスを意識せずに処理が行えるよう、以下のコマンドに対応した API を提供します。

・READ
・WRITE
・INQUIRY
・READ CAPACITY
・MODE SENSE
・TEST UNIT READY
・REQUEST SENSE

2 開発手順

本モジュールの使用方法の基本的な流れを以下に記します。

インストール

↓

納品された CD から、開発用 PC へインストールします。
インストールの手順に関しましては「3 インストール手順」を参照ください。

動作環境の構築

↓

本モジュールを使用するのに必要な周辺モジュールの準備をします。
詳しくは「5.1 動作環境の準備」を参照ください。

ポーティング

↓

本モジュールを使用する環境に合わせてポーティングします。
ポーティングする項目、手順などに関しましては、「5.2 ポーティング手順」を参照ください。

ライブラリファイル作成

↓

本モジュールのライブラリファイルを作成します。
ライブラリファイルの作成手順などに関しましては、「5.3 ライブラリファイル作成手順」を参照ください。

サンプルアプリケーション動作

↓

本モジュールを使用したアプリケーションを作成するためのサンプルとして、サンプルアプリケーションが付属しています。これは同時に、ポーティングした本モジュールの基本的な動作確認も行ないます。従って、アプリケーションを作成する前に、一度サンプルアプリケーションを動作させて頂くことを推奨します。
サンプルアプリケーションの実行ファイル作成手順、内容などに関しては、「6.3 実行ファイル作成手順」、「6.4 サンプルアプリケーション」を参照ください。

アプリケーション作成

↓

本モジュールを使用したアプリケーションを作成します。
アプリケーションに提供される API 仕様、アプリケーションの作成方法に関しましては、「6.1 API仕様」、「6.2 アプリケーション作成」を参照ください。

動作確認

実装した本モジュールの動作確認を行います。
動作確認方法に関しましては、「7 動作確認方法」を参照ください。

3 インストール手順

CD の中にある”ms_c”フォルダ以下を、開発に使用する PC の適切な場所にコピーして頂くだけで、インストールは完了です。

4 フォルダ構成

MSC（マスストレージクラス）ドライバのフォルダ構成は以下のとおりです。

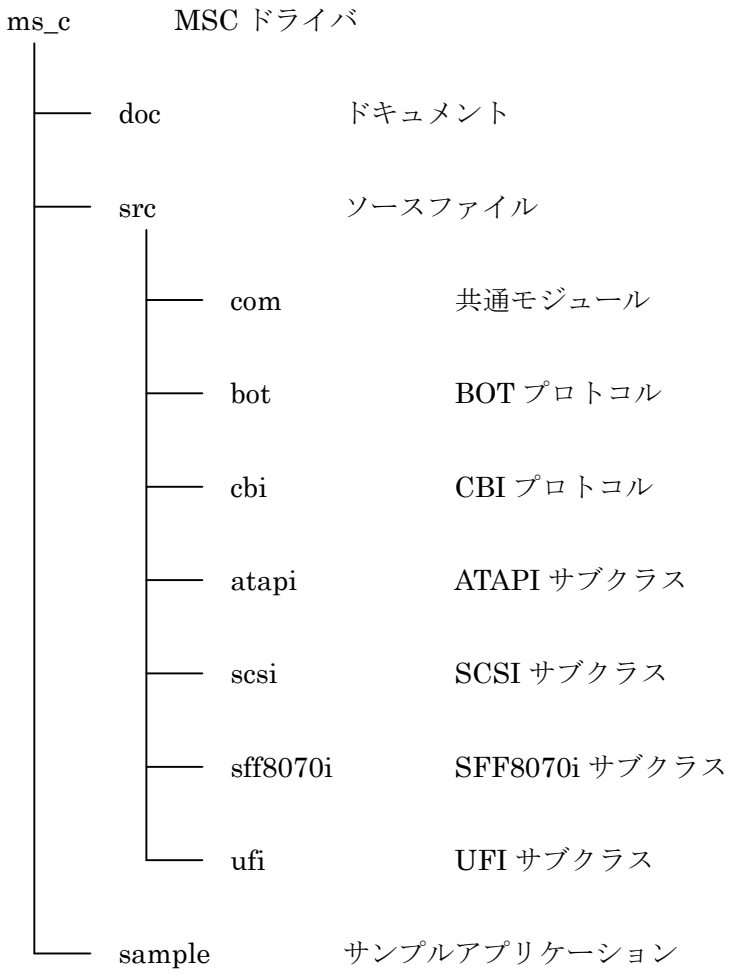

表 4-1ファイル一覧

ms_c	
relnote.txt	リリースノート
GR-USB HOST# MSC ユーザーズガイド.pdf（本ドキュメント）	
doc	
GR-USB HOST# MSC API 仕様書.pdf	
com	
grp_msc_drv.h	MSC ドライバヘッダファイル
grp_msc_config.h	MSC ドライバ設定用ヘッダファイル
grp_msc_local.h	MSC ドライバローカルヘッダファイル
grp_msc_debug.h	MSC ドライバデバッグ用ヘッダファイル
grp_msc.h	MSC ドライバヘッダファイル
grp_msc_drv.c	MSC ドライバモジュール
grp_msc_wrap.c	MSC ドライバラッパーモジュール
bot	
grp_msc_bot.c	BOT プロトコルモジュール
grp_msc_bot.h	BOT プロトコルヘッダファイル
cbi	
grp_msc_cbi.c	CBI プロトコルモジュール
grp_msc_cbi.h	CBI プロトコルヘッダファイル
atapi	
grp_msc_atapi.c	ATAPI サブクラスモジュール
grp_msc_atapi.h	ATAPI サブクラスヘッダファイル
grp_atapi_local.h	ATAPI サブクラスローカルヘッダファイル
scsi	
grp_msc_scsi.c	SCSI サブクラスモジュール
grp_msc_scsi.h	SCSI サブクラスヘッダファイル
grp_scsi_local.h	SCSI サブクラスローカルヘッダファイル
sff8070i	
grp_msc_sff8070i.c	SFF8070i サブクラスモジュール
grp_msc_sff8070i.h	SFF8070i サブクラスヘッダファイル
grp_sff8070i_local.h	SFF8070i サブクラスローカルヘッダファイル
ufi	
grp_msc_ufi.c	UFI サブクラスモジュール
grp_msc_ufi.h	UFI サブクラスヘッダファイル
grp_ufi_local.h	UFI サブクラスローカルヘッダファイル
sample	
msctest.c	サンプルアプリケーションモジュール
msctest.h	サンプルアプリケーションヘッダファイル

5 準備

本章では、MSC（マスストレージクラス）ドライバを使用するための準備について説明します。

5.1 動作環境の準備

MSC ドライバを使用するには、以下の周辺モジュールの準備が必要となります。それぞれのモジュールの準備について説明します。

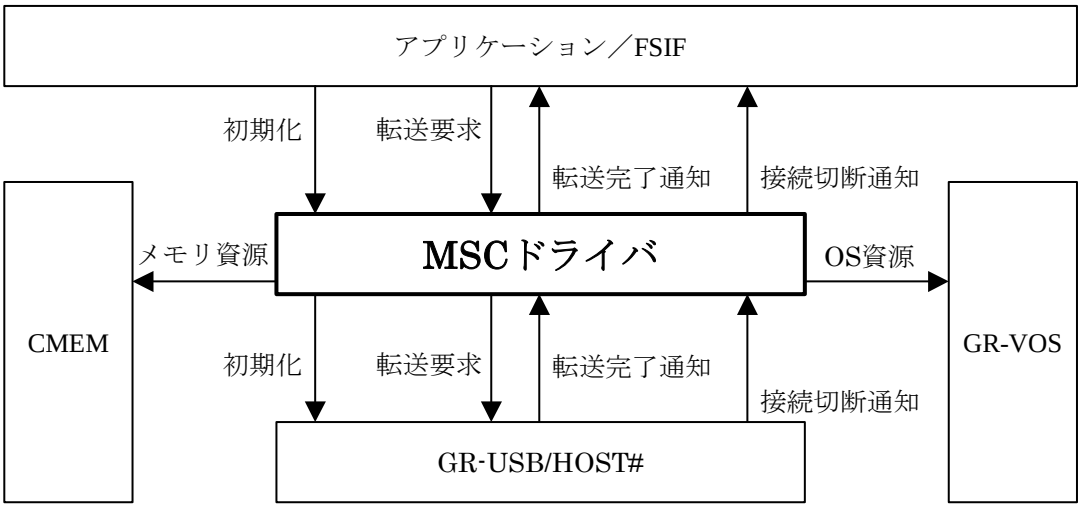

図 5-1 周辺モジュール

（1）GR-VOS

使用する OS、環境向けにポーティングし、GR-VOS に付属のテストアプリケーションを用いて事前に動作確認を済ませておく必要があります。

また、システム全体で使用する OS 資源の数を把握し、それを満たすだけの OS 資源を提供できるよう設定しておく必要があります。

MSCドライバが使用するOS資源に関しましては、「OS資源」を参照ください。

（2）GR-USB/HOST#

使用する USB ホストコントローラ向けにポーティングし、GR-USB/HOST#に付属のテストアプリケーションを使用して事前に動作確認を済ませておく必要があります。

（3）CMEM

CMEM は関数のインターフェースのみを提供しているため、内部処理は全てポーティングを行う必要があります。

また、システム全体で使用する CMEM メモリ資源の種類や数を把握し、それを満たすだけのメモリ資源を提供できるよう設定しておく必要があります。

MSCドライバが使用するメモリ資源に関しましては、「メモリ資源」を参照ください。

5.2 ポーティング手順

MSC ドライバを使用するには、MSC ドライバ自身を環境に合わせてポーティングする必要があります。ここでは、MSC ドライバのポーティング項目、仕様などについて説明します。

5.2.1 ポーティング対象ファイル

MSC ドライバのポーティング対象となるファイルを以下に記します。

・grp_msc_config.h

5.2.2 定義の設定

（1）プロトコル

プロトコルの設定を行います。

使用するプロトコルの定義を有効に、使用しないプロトコルの定義はコメントアウトして無効にしてください。

表 5-1 プロトコル

#	定義名	初期設定	内容
1	GRP_MSC_USE_BOT	定義	BOT（Bulk Only Transport）プロトコル
2	GRP_MSC_USE_CBI	未定義	CBI（Control Bulk Interrupt Transport）プロトコル

（2）サブクラス

サブクラスの設定を行います。

使用するサブクラスの定義を有効に、使用しないサブクラスの定義はコメントアウトして無効にしてください。

表 5-2 サブクラス

#	定義名	初期設定	内容
1	GRP_MSC_USE_ATAPI	未定義	ATAPI
2	GRP_MSC_USE_SCSI	定義	SCSI
3	GRP_MSC_USE_SFF8070I	未定義	SFF8070i
4	GRP_MSC_USE_UFI	未定義	UFI

5.2.3 定数の設定

（1）接続・登録数

同時に接続できるデバイス数や、登録できるプロトコルとサブクラスの組み合わせの数の設定を行います。

ご利用の環境に合わせて設定してください。

表 5-3 接続・登録数

#	定義名	初期設定	内容
1	GRP_MSC_DEVICE_MAX	GRP_USB_HOST_DEVICE_MAX	同時に接続できるデバイス数
2	GRP_MSC_REG_MAX	1	登録できるプロトコルとサブクラスの組み合わせの数

※GRP_MSC_DEVICE_MAX について

本定義は、以下のような場合、設定した値分の MS デバイスが接続することが出来ないことにご注意ください。

GRP_USB_HOST_DEVICE_MAX を「3」と設定し、HUB＋MS デバイス×2 を接続した場合、USB デバイスが 3 台接続されているため、これ以上、USB デバイスが接続することが出来ません。そのため、GRP_MSC_DEVICE_MAX を「3」と設定しても、3 台接続することは出来ません。

※GRP_MSC_REG_MAX の設定値について

本定義は、上位アプリケーションから登録するクラス-サブクラスの組み合わせの数を設定値としてください。弊社クラスドライバでは、クラス 2 種類、サブクラス 4 種類を定義していますのでデフォルトの設定値は 2×4＝8 としています。

5.3 ライブラリファイル作成手順

MSC ドライバを使用する場合、まずは MSC ドライバだけで 1 つのライブラリファイルを作成し、それをアプリケーションとリンクさせるようにすることで、整理され、快適に使用することができます。（もちろんライブラリファイル化せず、直接アプリケーションとリンクしても構いません。）
ここでは、MSC ドライバのライブラリファイルを作成する手順について説明します。

（1）コンパイル対象のファイル

ms_c フォルダ以下の全ての C ソースファイルが対象となります。

（2）インクルードパス

ms_c フォルダ以下の全てのヘッダファイルや、周辺モジュールの必要なファイルが参照できるよう、以下のパスを設定してください。

MSCドライバ

＜インストール先＞¥ms_c¥src¥com
＜インストール先＞¥ms_c¥src¥bot
＜インストール先＞¥ms_c¥src¥cbi
＜インストール先＞¥ms_c¥src¥atapi
＜インストール先＞¥ms_c¥src¥scsi
＜インストール先＞¥ms_c¥src¥sff8070i
＜インストール先＞¥ms_c¥src¥ufi

GR-VOS

＜インストール先＞¥gr_vos¥common
＜インストール先＞¥gr_vos¥inc

GR-USB/HOST#

＜インストール先＞¥grp_usb_h¥src¥usbm¥common
＜インストール先＞¥grp_usb_h¥src¥usbm¥usbd
＜インストール先＞¥grp_usb_h¥src¥usbm¥cnf_sft
＜インストール先＞¥grp_usb_h¥src¥hcm¥cmem

（3）マクロ定義

MSC ドライバでは、コンパイル時に定義が必要なマクロは特にありません。

（4）出力ファイル

出力ファイルを”ライブラリファイル”形式に設定してください。
ライブラリファイルの名前は特に指定はありませんが、どのモジュールのライブラリファイルか判別しやすい名前にすることを推奨します。（例：grusbh_msc.lib）

（5）コンパイル・リンク

（1）で示した対象ファイルを全てコンパイル、リンクし、1 つのライブラリファイルを生成してください。

6 使用方法

本章では、MSC（マスストレージクラス）ドライバを使用する方法について説明します。

6.1 API仕様

MSC ドライバがアプリケーションに対して提供する API 関数を以下に記します。

初期化・登録関数

grp_msc_Init	MSC ドライバの初期化
grp_msc_Register	デバイスの登録

基本関数

grp_msc_Open	デバイスのオープン
grp_msc_Close	デバイスのクローズ
grp_msc_ReqCmd	コマンドの要求
grp_msc_Cancel	コマンドのキャンセル
grp_msc_Abort	全コマンドのキャンセル
grp_msc_Reset	デバイスのリセット
grp_msc_GetMaxLun	Get Max LUN 要求
grp_msc_GetMaxLunCancel	Get Max LUN のキャンセル
grp_msc_GetSubClass	サブクラスの取得

コマンド関数

grp_msc_ReadSector	READ SECTOR
grp_msc_WriteSector	WRITE SECTOR
grp_msc_Inquiry	INQUIRY
grp_msc_ReadFormatCapacity	READ FORMAT CAPACITY
grp_msc_ReadCapacity	READ CAPACITY
grp_msc_ModeSense	MODE SENSE
grp_msc_TestUnitReady	TEST UNIT READY
grp_msc_RequestSense	REQUEST SENSE

API 仕様の詳細に関しましては、別冊『GR-USB/HOST# MSC API 仕様書』を参照ください。

6.2 アプリケーション作成方法

MSC ドライバの提供する API を使用してアプリケーションを作成する方法について説明します。

6.2.1 必要なヘッダファイル

MSC ドライバを使用するアプリケーションは、以下のヘッダファイルをインクルードする必要があります。

＜インストール先＞¥gr_vos¥inc¥grp_vos.h

＜インストール先＞¥ms_c¥src¥com¥grp_msc.h

＜インストール先＞¥gr_usb_h¥src¥usbm¥common¥grp_usbc.h

6.2.2 API関数の使い方

ここでは基本的な API 関数の使い方について、サンプルアプリケーションのプログラムを基に説明します。

サンプルアプリケーションに関しましては、「6.4 サンプルアプリケーション」を参照ください。

（1）初期化

```
/* Initialize GR-VOS */
lStat = grp_vos_Init();
if( lStat != GRP_VOS_POS_RESULT )
{
    return 1;
}

/* Initialize GR-USB/Host */
lStat = grp_usbc_HostInit();
if( lStat != GRP_USBC_OK )
{
    return 2;
}

/* Initialize MSC Driver */
lStat = grp_msc_Init(GRP_USB_NULL);
if( lStat != GRP_MSC_OK )
{
    return 3;
}

/* Register Mass Storage Device Type */
tMscReg.ucSubClass          = GRP_MSC_UFI_CODE;
tMscReg.ucProtocol          = GRP_MSC_CBI_CODE;
tMscReg. pfnMscEvCallback   = _MscTest_ConDis_Callback;
tMscReg.pvUserRef           = GRP_USB_NULL;
tMscReg.ulMode              = GRP_MSC_REG_PROTOCOL;
lStat = grp_msc_Register( &tMscReg );
if( lStat != GRP_MSC_OK )
{
    return 4;
}
```

① ② ←

```
    （中略）

    /* Enable GR-USB/Host interrupt */
    lStat = grp_usbc_Enable();                                    ←―― ③
    if( lStat != GRP_USBC_OK )
    {
        return 5;
    }
```

周辺モジュールと MSC ドライバの初期化を行います（①）。MSC ドライバの初期化の前に、GR-VOS と GR-USB/HOST#の初期化を行う必要があります。
各モジュールの初期化が終わりましたら、次にデバイスの登録を行います（②）。マスストレージクラスで取り扱いたいデバイスのサブクラスとプロトコルの組み合わせを登録します。組み合わせが複数ある場合は、この処理を繰り返し行ってください。ここで登録したサブクラスとプロトコルの組み合わせに一致したデバイスが接続・切断されると、登録時に指定したコールバック関数が呼ばれます。
初期化、登録が終わりましたら、最後に GR-USB/HOST#の起動処理を行います（③）。この処理により、USB/Host の割込みが許可されます。

（２）接続・切断

```
static grp_s32 _MscTest_ConDis_Callback(grp_msc_notify *ptMscNotify)
{
    if( ptMscNotify->iEvent == GRP_MSC_ATTACHED )
    {
        l_tMscHdr = ptMscNotify->hMscHdr;                         ←―― ④
        l_ulConnectFlag = FLAG_ON;
    }
    else if( ptMscNotify->iEvent == GRP_MSC_DETACHED )
    {
        l_ulConnectFlag = FLAG_OFF;
    }
}
```

初期化で登録したデバイスが接続・切断されると、登録時に指定したコールバック関数が呼び出されます。
接続時には、パラメータで渡されるマスストレージハンドルを保存する必要があります（④）。このハンドルは以降の転送などで使用します。

（3）オープン

```
l_tMscCmd.hMscHdr = l_tMscHdr;

/* Open Device */
lStat = grp_msc_Open(&l_tMscCmd);                    ←―― ⑤
if( lStat != GRP_MSC_OK )
{
    return 1;
}
```

登録したデバイスの使用には、まずオープン処理を行います（⑤）。デバイスの指定方法として、接続時に保存したマスストレージハンドルを用います。

（4）転送要求

```
l_tMscCmd.hMscHdr          = l_tMscHdr;
l_tMscCmd.ucLun            = 0;
l_tMscCmd.pucReqBuffer     = l_aucSectorBuf;
l_tMscCmd.ulReqLength      = INQUIRY_DATA_SIZE;          ⑥
l_tMscCmd.pfnCallback      = _MscTest_Command_Callback;
l_tMscCmd.pvUserRef        = USER_PTR;
l_tMscCmd.ucDir            = GRP_USBD_TX_IN;
l_tMscCmd.ulCmdLength      = SBC_INQUIRY_SIZE;
/* Make Command Content */
pucStr = (SBC_INQUIRY_STR*)l_tMscCmd.ucCmdContent;
grp_std_memset( pucStr, 0, GRP_MSC_CMD_LENGTH );
pucStr->ucCommand = SBC_INQUIRY_COMMAND;
pucStr->ucAllocationLen = (grp_u8)INQUIRY_DATA_SIZE;    ⑦
pucStr->ucCmddtEvpd = 0;
pucStr->ucPageOperation = 0;
pucStr->ucControl = 0;

/* Request Command */
if( grp_msc_ReqCmd(&l_tMscCmd) == GRP_MSC_OK )    ←―― ⑧
{
```

ここではサブクラスが SCSI のデバイスに対し、INQUIRY コマンドを発行する手順を示します。

まずは MSC コマンド情報構造体（grp_msc_cmd）のメンバに必要な情報をセットします（⑥）。次に MSC コマンド情報構造体のコマンド領域（ucCmdContent）に、コマンドを設定します（⑦）。この時はサブクラスを意識する必要があります。

MSC コマンド情報構造体の各メンバに必要な情報をセットし終わりましたら、最後に転送要求をかけます（⑧）。転送が完了しますと、⑥でセットしたコールバック関数が呼び出されます。

（5）クローズ

```
l_tMscCmd.hMscHdr = l_tMscHdr;

/* Close Device */
lStat = grp_msc_Close(&l_tMscCmd);                    ←―― ⑨
if( lStat != GRP_MSC_OK )
{
      return 1;
}
```

デバイス使用完了後には、デバイスのクローズを行います（⑨）。

6.3 実行ファイル作成手順

ここでは MSC ドライバを使用したアプリケーションの実行ファイルを作成する手順について説明します。

なお、「ライブラリファイル作成手順」に従って、既に MSC ドライバのライブラリファイルを作成していることを前提とします。また、MSC ドライバを使用するために必要な周辺モジュールに関しても、それぞれのモジュール毎にライブラリファイルが作成されていることを前提とします。

（1）リンク対象のファイル

以下のファイルがリンクの対象となります。

- アプリケーションソースファイル
- MSC ドライバのライブラリファイル
- 各周辺モジュールのライブラリファイル（GR-VOS や GR-USB/HOST#など）
- その他のライブラリ（OS や標準ライブラリなど）

（2）インクルードパス

アプリケーションをコンパイルするためには、MSC ドライバ、周辺モジュールのそれぞれの必要なヘッダファイルが参照できるよう、以下のパスを設定してください。

MSCドライバ

＜インストール先＞¥ms_c¥src¥com
＜インストール先＞¥ms_c¥src¥bot
＜インストール先＞¥ms_c¥src¥cbi
＜インストール先＞¥ms_c¥src¥atapi
＜インストール先＞¥ms_c¥src¥scsi
＜インストール先＞¥ms_c¥src¥sff8070i
＜インストール先＞¥ms_c¥src¥ufi

GR-VOS

＜インストール先＞¥gr_vos¥common
＜インストール先＞¥gr_vos¥inc

GR-USB/HOST#

＜インストール先＞¥grp_usb_h¥src¥usbm¥common
＜インストール先＞¥grp_usb_h¥src¥usbm¥usbd
＜インストール先＞¥grp_usb_h¥src¥usbm¥cnf_sft
＜インストール先＞¥grp_usb_h¥src¥hcm¥cmem

（3）出力ファイル

出力ファイル形式を、環境に合わせた実行ファイルの形式に設定してください。

（4）コンパイル・リンク

アプリケーションをコンパイルしその他のライブラリとリンクして、1 つの実行ファイルを生成してください。

6.4 サンプルアプリケーション

MSC ドライバに付属のサンプルアプリケーションについて説明します。

6.4.1 ファイル構成

サンプルアプリケーションファイルを以下に記します。

- msctest.c
- msctest.h

6.4.2 使用方法

アプリケーションタスクから、サンプルアプリケーションの MSC ドライバテストメイン関数（MscTest_Main）を呼び出してください。

サンプルアプリケーション内部では、GR-VOS の初期化と自タスクの登録、GR-USB/HOST#の初期化と起動処理も行っております。

6.4.3 動作内容

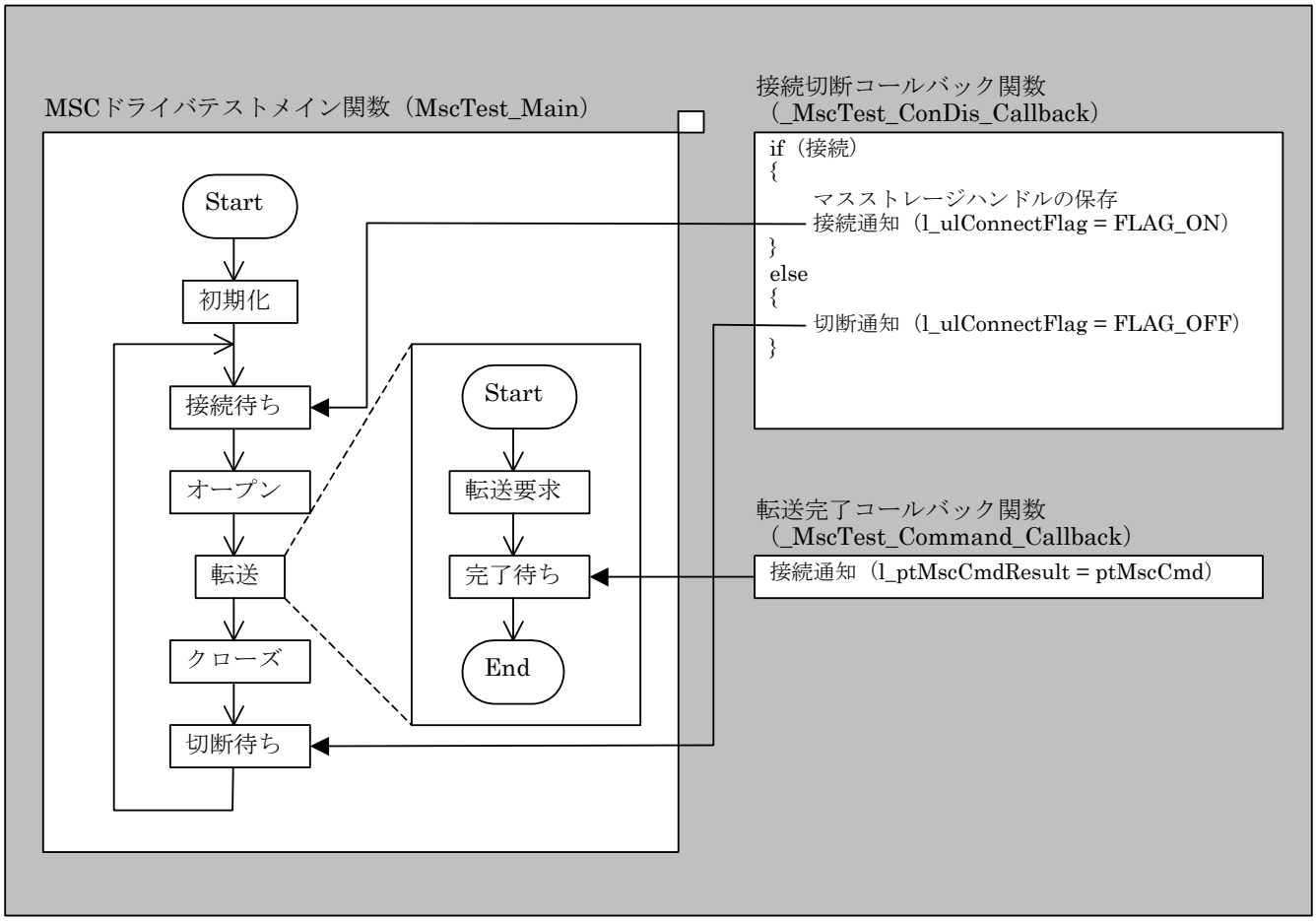

図 6-1 サンプルアプリケーション動作内容

MSC ドライバテストメイン関数は、初期化、オープン、転送、クローズと一連の動作のテストを行います。途中でエラーが発生しますと、そこで無限ループに入り止ります。従って、クローズが正常に終了したことは、一連のテストが正常に終了したことを意味します。
一連のテストが終了すると切断待ちで止まりますので、デバイスを切断・接続して頂くことで、再度テストを実施することができます。

7 動作確認方法

本章では、ポーティングした MSC（マスストレージクラス）ドライバを動作確認する方法について説明します。

7.1 初期化処理の確認

MSC ドライバでは、grp_msc_Init（grp_msc_drv.c）にて初期化が行われます。
以下の項目について正常に動作が行われているかご確認ください。

Check

□ （1）grp_vos_CreateXXX でエラーとなっていないか
⇒エラーとなる場合、リソース数が足りているか GR-VOS の設定をご確認ください。

□ （2）BOT プロトコルを使用する場合、grp_msc_BotInit は呼ばれているか
⇒呼ばれていない場合、GRP_MSC_USE_BOT（grp_msc_cfg.h）の定義が有効になっているかご確認ください。

□ （3）CBI プロトコルを使用する場合、grp_msc_CbiInit は呼ばれているか
⇒呼ばれていない場合、GRP_MSC_USE_CBI（grp_msc_cfg.h）の定義が有効になっているかご確認ください。

＜_msc_SubClassInit 内＞

□ （4）ATAPI サブクラスを使用する場合、grp_msc_AtapiInit は呼ばれているか
⇒呼ばれていない場合、GRP_MSC_USE_ATAPI（grp_msc_cfg.h）の定義が有効になっているかご確認ください。

□ （5）SCSI サブクラスを使用する場合、grp_msc_ScsiInit は呼ばれているか
⇒呼ばれていない場合、GRP_MSC_USE_SCSI（grp_msc_cfg.h）の定義が有効になっているかご確認ください。

□ （6）SFF8070i サブクラスを使用する場合、grp_msc_Sff8070iInit は呼ばれているか
⇒呼ばれていない場合、GRP_MSC_USE_SFF8070I（grp_msc_cfg.h）の定義が有効になっているかご確認ください。

□ （7）UFI サブクラスを使用する場合、grp_msc_UfiInit は呼ばれているか
⇒呼ばれていない場合、GRP_MSC_USE_UFI（grp_msc_cfg.h）の定義が有効になっているかご確認ください。

7.2 登録処理の確認

MSC ドライバでは、grp_msc_Register（grp_msc_drv.c）にてデバイスの登録が行われます。
以下の項目について正常に動作が行われているかご確認ください。

Check

□ （1）_msc_SearchEmptyReg でエラーとなっていないか

⇒エラーとなる場合、デバイス登録数の設定 GRP_MSC_REG_MAX（grp_msc_cfg.h）をご確認ください。

7.3 最初のデバイス接続処理の確認

初期化、登録、GR-USB/HOST#の起動処理（grp_usbc_Enable）後、最初にデバイスを接続した際に、以下の項目について正常に動作が行われているかご確認ください。

Check

□ （1）_msc_EventCallback（grp_msc_drv.c）が呼び出されるか

⇒呼び出されない場合、MSC デバイスの登録処理や、GR-USB/HOST#のポーティングをご確認ください。

Appendix A 内部構造

MSC（マスストレージクラス）ドライバの内部構造について説明します。

A.1 内部構成

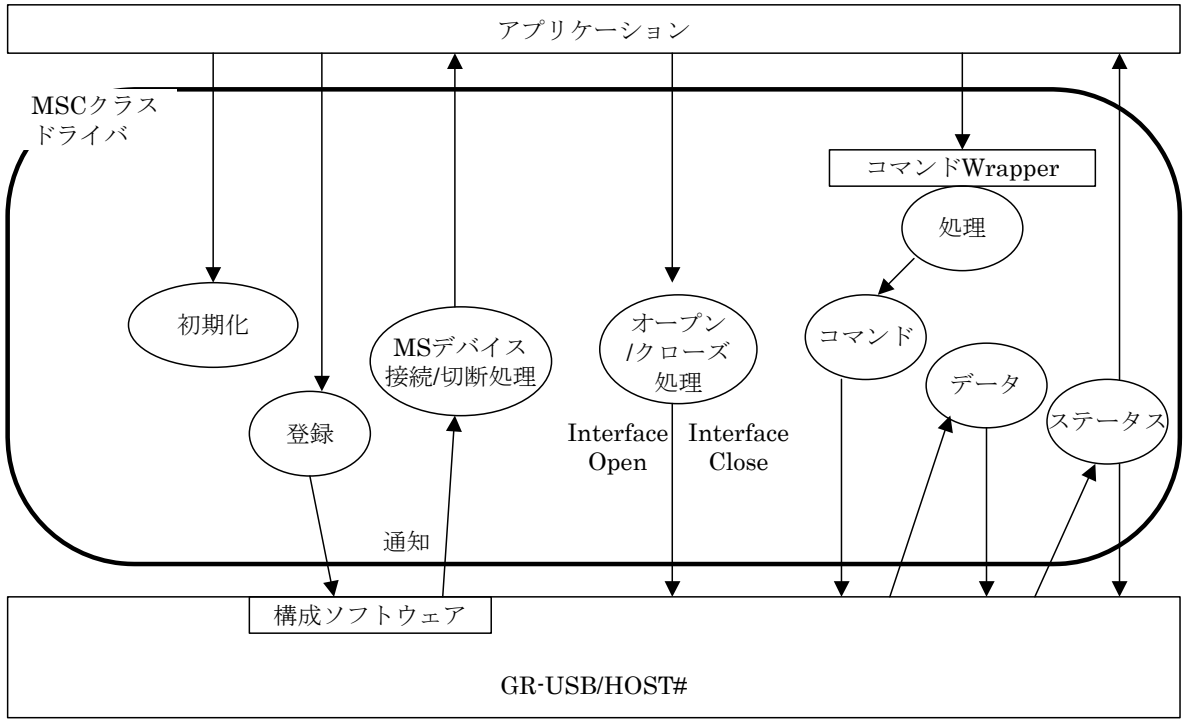

図 A-1 内部構成

・初期化は、アプリケーションより実行します。
・初期化後に MS クラスを登録を実行します。必要なプロトコル/サブクラスの組み合わせ分、登録処理を実行してください。
・MS デバイスが接続されると、構成ソフトウェアより接続が通知されます。
・そのままアプリケーションへ通知しますので、コマンドを発行する前にオープンの処理を実行してください。
・コマンドは、プロトコル/サブクラスを意識せず実行することが出来ます。
・内部では、コマンドを発行、コールバック通知によりデータ部を発行、コールバック通知によりステータス部を発行し、そのコールバックにより完了を上位へ通知します。
・デバイスが切断された時や MS デバイスにアクセスしない場合、クローズ処理を実行してください。

A.2 OS資源

MSC ドライバで使用する OS 資源は以下のとおりです。

セマフォ : 1

（1）セマフォ

#	名前	初期カウント値	用途
1	“sMSC”	1	MSC デバイス情報制御用

A.3 メモリ資源

MSC ドライバで使用する CMEM のメモリ資源は以下のとおりです。

#	名前	ID	サイズ	個数
1	GRP_CMEM_xxx_MSC_COMMAND	0x80000007	32	GRP_USB_HOST_DEVICE_MAX
2	GRP_CMEM_xxx_MSC_STATUS	0x80000008	16	GRP_USB_HOST_DEVICE_MAX

xxx : ”ID”、”BSIZE”（サイズ）、”BNUM”（個数）が入ります

Appendix B 注意事項

MSC（マスストレージクラス）ドライバをご利用にあたっての注意事項について記します。

B.1 データバッファ領域

MSC ドライバでデータ転送を伴う通信を要求する場合、ユーザーは CMEM（共通メモリ管理モジュール）よりバッファ領域を確保し、その領域を使用する必要があります。

※CMEM では、バッファ領域のキャッシュ/非キャッシュの問題や、特定のホストコントローラに存在するバウンダリ問題の対策などのメモリ管理を対応するため、内部ロジックをポーティングしていただく必要があります。

しかし、使用されるホストコントローラや環境によっては、CMEM の機能を必要としないケースもあります。この場合、サンプルとして提供される CMEM にて動作させることが可能です。詳細につきましては、各ホストコントローラドライバ説明書をご覧ください。

B.2 USBメモリのサブクラスについて

市販されている USB メモリには以下の組み合わせが存在します。

・SCSI-BOT

市販されている USB メモリの大半が、この組み合わせになっています。

・SFF8070i-BOT

この組み合わせの USB メモリを数機種確認しております。なお、この組み合わせの場合、HUB を内蔵しているタイプが多いようです。

・ATAPI-BOT

この組み合わせの USB メモリを 1 機種確認しております。

弊社ドライバでは、上記 3 種類の組み合わせおよび HUB クラスを実装することで、多くの市販されている USB メモリを扱うことが可能となります。

もし、接続できない USB メモリなどがありましたら、メーカ名および型番を明記の上、弊社サポートまでご連絡ください。

B.3 ベンダ固有サブクラスについて

ベンダ固有のサブクラスは、お客様にて実装していただく必要があります。実装方法につきましては、別途、弊社サポートまでお問い合わせください。

Appendix C トレースログ機能

弊社ドライバには、トレースログ機能がございます。
トレースログ機能は、弊社ドライバ内の動きを関数単位で記録する機能です。
不具合などが発生した際にログの取得をお願いする場合がございます。

ログ取得手順

（1）マクロ定義に「GRP_USBC_DBG_MDL」を追加します。

（2）grp_usbc_dbg.h の「GRP_DBG_MAX_TRACE」を設定してください。デフォルトでは「1024」となっています。

（3）ドライバのライブラリファイル、およびアプリケーションの実行ファイルを作成します。

（4）プログラムを実行し、問題発生後に停止します。（この間のトレースログが記録されます）

（5）grp_usbc_dbg.c 内の「g_iDbgMdlCnt」の値と、「g_aucDbgMdlTbl[][]」の値をダンプし、ご連絡ください。（デフォルトの設定ですと、4×1024＝4096 バイトの情報量となります）

※上記設定のみでは、ホストコントローラドライバは含まれません。
ホストコントローラドライバのログを取得する方法に関しましては、ホストコントローラドライバ説明書を参照ください。

Appendix D Grapeware技術サポート

D.1 サポート時間

午前 10：00 より午後 5：30 まで
（土日祝日、年末年始休日、指定休日を除く）

D.2 質問手順

問題発生

↓

製品、マニュアルについて不明な点がある。あるいは製品が仕様どおりに動作しない、不具合がある、などの問題に直面された場合。

保守サービス有効期限の確認

↓

Grapeware 技術サポートに問い合わせる前に、保守サービス有効期限を確認ください。
保守サービス有効期限内である場合、次のステップに進んでください。
保守サービス有効期限を過ぎている場合、弊社営業担当者まで問い合わせください。

Grapeware技術サポートに問い合わせ

Grapeware 技術サポートへの問い合わせはメール・電話・FAX で受け付けております。

メール：gr@support.grape.co.jp
電話：045-222-3762
FAX：045-222-3760

※問題や、その対応の正確なログを残すために、弊社ではメールを推奨しております。

問い合わせの際は、以下の情報をお知らせください。
＜情報＞

- カスタマー番号
- 御社名、担当者様名
- 問題の発生する製品名、バージョン
- 問題の内容（問題発生手順、頻度など）

また、開発環境（CPU、ツールなど）に関する情報も併せてお知らせ頂けますと、サポートをよりスムーズに進められる場合がございます。
別紙『質問フォーム.txt』もご利用ください。

問い合わせに対し、2 営業日以内に回答、もしくは（回答まで至らなかった場合は）状況報告をさせて頂きます。

GR-USB/HOST#
マスストレージクラスドライバ
ユーザーズガイド

発行年月：2015 年 12 月　第 1.02 版
発行：株式会社グレープシステム

E-Mail　：　gr@support.grape.co.jp
URL　　：　http://www.grape.co.jp