

Memory card Library.

Memory card Library V1.0

Author: Kevin Thompson
Date: 18 September 1997
Email: Kevin_Thompson@PlayStation.sony.com

Usage:

The library has to be recompiled for every different project. This is due to the Memory card warning messages contained within the code. If you go into memcard.h and change the following #defines

```
#define GAME_TITLE "TestTitle" // place your title here
#define NUMBER_OF_BLOCKS_USED 1 // number of blocks your game uses
#define FORWARD_BUTTON "X" // button to go forward in game
#define RETURN_BUTTON "Triangle" // button to go backwards in game
```

It is also possible to change the Authorised names as seen below. But I do not recomend this as your game could then fail QA.

// these are changeable.

```
#define AUTH_NAMES1 "Memory card"
#define AUTH_NAMES2 "Memory card slot"
```

You will also have to change the KanjiFntPrint I have in the message functions to use your internal printf function.

There is one other thing that will be needed to changed, This is the name name of your printf function

ReCompiling the Libraries:

Type Wmake in the src directory.

Description:

This Library Is somthing I coded to make Programming the Memory cards as easy as possible using the Lowlevel functions supplied libcard.lib.

I have also supplied the source code to the library. This is so you can see just how the code is working and if needed alter and re-compile.

Functions:

Source-

NOUBOC.c

Description-

Finds out how many blocks have been used on a Memory card.

Syntax-

```
int _mc_NumberOfUsedBlocksOnCard(FILE_HEADER *fileheader)
```

Return Value-

Returns how many Used Blocks there are on the Memory card.

Remarks-

Adds the BlockEntries of the FILE_HEADER passed.

Source-

NOUFOC.c

Description-

Finds out how many blocks are free on a Memory card.

Syntax-

```
int _mc_NumberOfFreeBlocksOnCard(FILE_HEADER *fileheader)
```

Return Value-

Returns how many Free Blocks there are on the Memory card.

Remarks-

Minuses the BlockEntries of the FILE_HEADER passed from 15.

Source-

FileONMC.c

Description-

Finds out if a file of the same name already exists on the Memory card. The file name must always include BU** on the front else an error will occur.

Syntax-

```
int _mc_FileExistsOnCard(char *fileName)
```

Return Value-

Returns 1 if the file exists on the Memory card else returns 0.

Remarks-

This function works by trying to open the file, if the file opens then the file does not exist else the file is already there.

Source-

DCEvenSw.c

Description-

Gets the Memory card events. in Software

Syntax-

```
int _mc_GetCardEventSw(void)
```

Return Value-

Returns Event Number.

```
EVENT_IOE           = Memory card OK
EVENT_ERROR
EVENT_TIMEOUT       = Memory card Not Present
EVENT_NEWCARD
```

Remarks-

The #defines are contained within memcard.h

Tests Events Ev0,Ev1,Ev2,Ev3.

Source-

DCEvenHw.c

Description-

Gets the Memory card events. HardWare.

Syntax-

```
int _mc_GetCardEventHw(void)
```

Return Value-

Returns Event Number.

```
EVENT_IOE           = Memory card OK
EVENT_ERROR
EVENT_TIMEOUT       = Memory card Not Present
EVENT_NEWCARD
```

Remarks-

The #define's are contained within memcard.h

Tests Events Ev10,Ev11,Ev12,Ev13.

Source-

CCEvenSw.c

Description-

Clears the Memory cards SW Events

Syntax-

```
void _mc_ClearEventSw(void)
```

Return Value-

None

Remarks-

Source-

CCEvenHw.c

Description-

Clears the Memory cards HW Events

Syntax-

void _mc_ClearEventHw(void)

Return Value-
None

Remarks-

Source-
FormatC.c

Description-
Formats a selected Memory card.

Syntax-
int _mc_FormatCard(long channel)

Return Value-
Return 1 if Format Successful, 0 if not.

Remarks-
#define's for the Channels are contained within Memcard.h

Source-
CardStat.c

Description-
Gets the Status of the Memory card.

Syntax-
int _mc_GetCardStatus(long channel, int StartUp)

Return Value-
Returns the Status of the Card, i.e., Formatted, UnFormatted.

Remarks-
#defines for the status of the Memory card's are contained in
Memcard.h

StartUp Defines if you wish to check a new memory card. This is
also used if you want to check if the Memory card is UnFormatted.

Source-
Delete.c

Description-
Deletes A Save from A Memory card.

Syntax-
int _mc_DeleteFileFromCard(char *drive, char *name)

Return Value-
Returns 1 if Deletion successful else returns 0.

Remarks-

Source-

Enable.c

Description-

Enables the Memory card Events.

Syntax-

int _mc_EnableCardEvents(void)

Return Value-

Returns 1 if successful

Remarks-

This code should only be called after _mc_DisableCardEvents to re-enable the events.

IT IS NOT PART OF THE INITIALISATION.

Source-

Disable.c

Description-

Disables The Memory card Events.

Syntax-

int _mc_DisableCardEvents(void)

Return Value-

Returns 1 if successfull.

Remarks-

This Function Only Disables the events. It does not close them nor does it call StopCard().

Source-

STDini.c

Desctiption-

Initialises and opens the Memory card Events

Syntax-

int _mc_InitializeCardAndEventsStandard(int shared)

Return Value-

Returns 1 if successful.

Returns 0 if the shared argument is not valid.

Remarks-

Initialises and opens all needed events and calls needed functions to initalise the cards correctly.

The Shared Argument is to say if the Memory card is going to be used with a MultiTap. There are Some Defines in Memcard.h to cover this.

Source-
GFHFC.c

Description-
Loads a selected file from the Selected Memory card.

Syntax-
int _mc_LoadFromCard(long channel, char *fileName, FILE_HEADER *fileHeader, int DownloadSize)

Return Value-
Return 0 if the file could not be opened
Return 1 if successful
Return 2 if an error occurs during download.

Remarks-
The Download size is the size of the file you wish to download in bytes.

Source-
FFileNF.c

Description-
Downloads the dirEntry information to the DirEntry Structure passed.

Syntax-
int _mc_ReadDirectoryInformation(long port, char *drive, struct DIRENTRY *d)

Return Value-
Returns the amount of saves contained on the card.

Remarks-

Source-
ASC2SJIS.c

Description-
Converts an ascii string into a Kanji String.

Syntax-
void _mc_AsciiStringToSJIS(unsigned char *string, unsigned char *dest)

Return Value-
None

Remarks-

Source-

icon.c

Description-

Downloads A Memory card Icon into a selects point in Vram

Syntax-

```
int _mc_LoadIconDataIntoVram(int u, int v, int w, int h, int cu,
int cv, char *icon, char *clut)
```

Return Value-

None

Remarks-

u,v,w,h are for the Icon Position.

cu,cv are for the Clut Position.

Source-

BlWrite.c

Description-

Writes a Block of data to a Selected Memory card

Syntax-

```
int _mc_BlockWrite(int blocks, char *buffer, char *SaveName)
```

Return Value-

Returns 0 if the Select file already exists on the card.

Returns 1 if the save has been made successfully.

Returns 2 if the Save could not be opened.

Return 3 if there was an error during writing.

Remarks-

The save name has to have the corrcct BU***: added to the front or an error will occur when trying to open the file.

MESSAGE FUNCTIONS BELOW

Source-

LFMCMes.c

Description-

Contains all messages needed When Loading From the Memory card.
when there is only one slot available.

Syntax-

```
int _mc_LoadingFromTheMemoryCardMessages(int message, int mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 8 messages

Source-

LFMCMSLA.c

Description-

Contains all messages needed When Loading From the Memory card. when there are multiple slots available.

Syntax-

```
int _mc>LoadingFromMemoryCardMultipleSlotsAvMessages(int message,  
char *slot, int mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 8 messages

Source-

OWDOTMCM.c

Description-

Contains messages for when Overwriting Data on the Memory card

Syntax-

```
int _mc>OverWrittingDataFromOnMemoryCardMessages(int message, int  
mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 6 Messages.

Source-

smcmsame.c

Description-

Contains messages for use when Saving to the Memory card with multiple slots available.

Syntax-

```
int _mc_SavingToMemoryCardMultipleSlotsAvMessages(int message,
int blocksfree, char *slot, int mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 16 Messages

Source-

STCMes.c

Description-

Contains messages for use when Saving to the Memory card with One Slot available

Syntax-

```
int _mc_SavingToTheMemoryCardMessages(int message, int
blocksfree, int mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 16 Messages.

Source-

BootAM.c

Description-

Contains all messages for use when the game is booting and only one slot available. (Automated saves Messages)

Syntax-

```
int _mc_BootUpAutoMessages(int message, int FreeBlocks, int mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 4 Functions.

Source-

BootM.c

Description-

Contains all messages for use when the game is booting and only one slot available.

Syntax-

```
int _mc_BootUpMessage(int message, int mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 4 Functions.

Source-

STIMMess.c

Description-

Message for when saving to the PlayStaion Memory not the Memory card.

Syntax-

```
int _mc_SavingToTheInternalMemoryMessages(int message, int mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Contains 1 Message.

Source-

CDOTMCMes.c

Description-

Messages for use when checking the Data on the Present Memory card.

Syntax-

```
int _mc_CheckingDataOnCardMessages(int message, char *slot,int  
mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.

Source-

DDFMCMes.c

Description-

Messages For use when deleting data from the Memory card.

Syntax-

```
int _mc_DeletingDataFromTheMemoryCardMessages(int message, int  
mess)
```

Return Value-

Returns Text string (specified when the Fnt is opened).

Remarks-

Within this function I am using KanjiFntPrint with a specific position on screen (mess).

This function will have to be changed by the programme to include your game's own Printf function.