The EasySDK Guide

Table of contents

1 Hardware Description

An EasyFlash cartridge consists of two flash memory chips, called LOROM chip and HIROM chip. In the current hardware version, each of them has a size of 512 KiB.

Only 8 KiB of memory per chip can be seen in the address space of the C64 at once. Therefore EasyFlash supports banking. When each chip has a size of 512 KiB, there are 512 / 8 = 64 banks. The EasyFlash cartridge allows software to select the active bank.

The mapping of the chips into the C64 memory is controlled by two lines of the Expansion Port: /GAME and /EXROM. The "/" means that their meaning is inverted, i.e. 0 or low means active.

The possible states of these lines and their meaning is listed in Table 1.

/GAME	/EXROM	Comment
1	1	Cartridge memory invisible ("Invisible Mode")
1	0	8 KiB from Chip 0 at $8000 ("8K Mode")
0	0	8 KiB from Chip 0 at $8000, 8 KiB from Chip 1 at $A000 ("16K Mode")
0	1	8 KiB from Chip 0 at $8000, 8 KiB from Chip 1 at $E000 ("Ultimax Mode")

Table 1: States of /GAME and /EXROM lines

These lines can be controlled by software, we'll come back to it soon.

The resulting memory model is shown in Fig. 1.

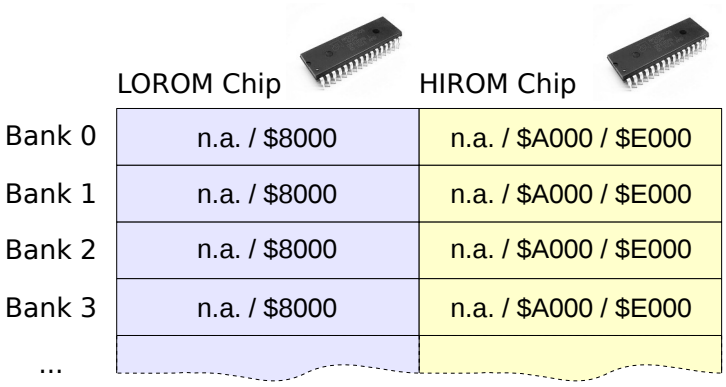

Fig. 1: EasyFlash memory model

If the Boot switch is in position "Boot" (or jumper open) and the computer is reset, EasyFlash is started normally: The Ultimax memory configuration is set, bank 0 selected. The CPU starts at the reset vector at $FFFC. As in Ultimax mode the HIROM chip is banked in at $E000, the flash must contain a valid reset vector and some start-up code. This is described in a separate chapter.

If the Boot jumper is in position "Disable" (or jumper closed) EasyFlash starts with cartridge memory hidden. However, software like EasyProg can write to the flash memory because it can override the jumper setting.

The software can chose the active bank and set the memory location by using two I/O registers. They are described in the following chapters.

Additionally an EasyFlash cartridge has 256 Bytes of RAM. This memory is always visible. It can be used to save small portions of code or data in a way which is unlikely to interfere with other software.

1.1 Register $DE00 – EasyFlash Bank

This is a write-only register located at $DE00. The active flash memory bank is chosen using this register. Basically it selects the state of the higher address lines. The bank selection can be made regardless if the flash memory is actually visible according to the memory configuration or not. The selected bank is valid for both chips, LOROM and HIROM.

Bit	7	6	5	4	3	2	1	0
Meaning	0	0	Bank number					

Table 2: Register $DE00

The value after reset is $00. The bits 6 and 7 should always remain 0, unless stated otherwise in this document.

1.2 Register $DE02 – EasyFlash Control

This is a write-only register located at $DE02. The software can control the memory configuration of the cartridge using this register. Additionally it can be used to set the state of the status LED.

Bit	7	6	5	4	3	2	1	0
Meaning	L	0	0	0	0	M	X	G

Table 3: Register $DE02

The value after reset is $00.

Position	**Name**	**Comment**
7	L	Status LED, 1 = on
6..3	0	reserved, must be 0
2	M	GAME mode, 1 = controlled by bit G, 0 = from jumper "boot"
1	X	EXROM state, 0 = /EXROM high
0	G	GAME state if M = 1, 0 = /GAME high

Table 4: Bits of $DE02

Following memory configurations are possible:

MXG	**Type**
000	GAME from jumper, EXROM high (i.e. Ultimax or Off)
001	Reserved, don't use this
010	GAME from jumper, EXROM low (i.e. 16K or 8K)
011	Reserved, don't use this
100	Cartridge ROM off (RAM at $DF00 still available)
101	Ultimax (Low bank at $8000, high bank at $e000)
110	8k Cartridge (Low bank at $8000)
111	16k cartridge (Low bank at $8000, high bank at $a000)

Table 5: Configuration of cartridge memory maps

Most likely the software is only interested in setting combinations with M = 1. Of course the

usual memory configurations selected with the 6510 register $01 do also apply.

1.3 RAM at $DF00

From $DF00 to $DFFF there are 256 bytes of RAM. This memory is not part of the normal 64 KiB of internal C64 RAM but part of the I/O memory space.

2 Conventions

Conventions can make the life easier and look professional. So let's invent some.

2.1 Address Format Convention

Addresses of EasyFlash have a special format in this document and related tools:

BB:C:FFFF

BB	Bank number, currently 00..3F
C	Chip number, 0 for LOROM chip or 1 for HIROM chip
FFFF	Offset in this bank and chip, 0000..1FFF

Table 6: Address format convention

The address 00:1:1FFC means bank 0, HIROM chip, offset 0x1FFC. This is the address which has to contain a reset vector to make the cartridge work.

2.2 Scan the Keyboard

When a cartridge is plugged into the C64, often the user has no other way to get to the BASIC prompt than unplugging the cartridge.

EasyFlash has a switch or a jumper to avoid booting. However, let's make it even easier for our users:

When you implement start-up code for EasyFlash, scan the keyboard as early as possible. If the Stop key is held down, make the cartridge invisible and call the kernal's reset vector. You can hide (“kill”) the cartridge by writing $04 to $DE02.

3 Supported cartridge types

Different types of cartridge images can be written to and started from an EasyFlash cartridge. Let's have find out how these work.

3.1 Normal 8K Cartridges

Normal 8K cartridges consist of up to 8 KiB of ROM visible at LOROM. This type of cartridges pull down /EXROM and leave /GAME high.

The memory is visible at $8000 and contains some magic bytes. The C64 Kernal detects these bytes in the reset code. If such a cartridge is found, it gets started with JMP($8000).

EasyFlash cartridges always start in Ultimax mode. Therefore there is a small boot code at the end of the HIROM flash chip on bank 0, i.e. at 00:1:FFxx. This code copies itself to RAM and checks whether the Stop key is pressed (refer to chapter 2.2). If not, it starts the 8K Cartridge.

To do this, it does some initializations which would be done by the kernel normally. Then it sets the 8K mode using $DE02. Finally it does JMP($8000).

Xxxxxxxxx

Bild

xxxxxxxxxx

4 EasyFS Cartridges

4.1 Introduction

EasyFS means Easy File System. This is a simple ROM file system which simplifies implementing software for EasyFlash.

EasySDK comes with some tools and code snippets to create and read EasyFS.

Of course developers are free to develop cartridges which do not use EasyFS by directly accessing the I/O registers described in this document.

As already mentioned earlier, EasyFlash cartridges always start in Ultimax mode. Therefore there is a small boot code at the end of the HIROM flash chip on bank 0, i.e. at 00:1:FFxx. Cartridges using EasyFS always use the 16 KiB cartridge mode, which is configured by the start-up code by setting /GAME and /EXROM active (low). All explanations in this chapter refer to this configuration.

If your cartridge makes use of the EasySDK File System (EasyFS), it contains a directory structure. This structure resides at 00:1:0000. This means it is visible at $A000 in 16 KiB mode when bank 0 is selected.

Currently a directory shall not contain more than 255 entries (excluding end mark). We will see that each entry has a size of 24 Bytes, so the whole directory can have a size of up to $1800 Bytes. Behind the directory there is space of $0800 bytes for the start-up code.

An example memory layout of an EasySDK cartridge using EasyFS is shown in Fig. 2.

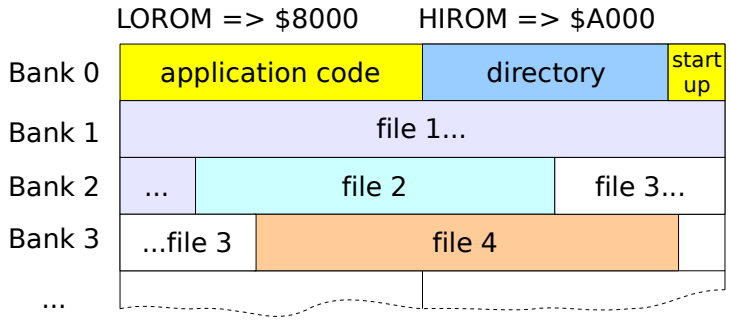

Fig. 2: An EasyFS cartridge

Remember that HIROM is banked in in Ultimax mode at $E000 after a reset. Therefore the start-up code contains the reset vector.

4.2 Format of EasyFS Directory Entries

Field	Type	Comment
Name	16 bytes	File name, 16 characters PETSCII (?), 0-padded
Flags	1 byte	File type and some flags
Bank	1 byte	Bank where the file starts, 0..63
Bank High	1 byte	Reserved for high byte of bank, currently always 0
Offset	2 bytes	File start offset in this bank 0x0000..0x3FFF, little endian
Size	3 bytes	File size, little endian

The flags field has following structure:

Bit	7	6	5	4	3	2	1	0
Meaning	H	R	R	Type				

The bit *H* means hidden. If this is 1, this file is not intended to be seen by a user, a file browser should not show it.

`Reserved bits marked with R must be set to 1.`

Table 7 shows the possible values for *Type*.

Value	Type
$00	`Files with this type are marked as invalid, they must be skipped. Note that the flags of this file may be not 0.`
$01	`Normal PRG file with 2 bytes start address`
...	
$10	Normal 8 kByte cartridge (0x8000..0x9FFF)
$11	Normal 16 kByte cartridge (0x8000..0x9FFF, 0xA000..0xBFFF)
$12	Normal Ultimax cartridge (0x8000..0x9FFF, 0xE000..0xFFFF)
$13	Normal Ultimax cartridge, first bank not used (0xE000..0xFFFF)
...	
$1F	End of directory. This entry is only a terminator.

Table 7: EasyFS file types

Note that erased flash memory does have the value $FF, so all bits are *one*. This means that an entry located in erased flash has the type 0x1f naturally.

Furthermore, when a flash is written, bits can only turn from *one* to *zero*. It is not possible to change a bit from *zero* to *one* when writing to flash memory. That's why the file type $00 marks a deleted or invalid file. $00 can be written regardless from the old file type by only changing *ones* to *zeros*.

The meaning of the file types $10 to $13 will be described in chapter 4.3.

4.3 Hybrid cartridges

An image may also contain one or more normal 8 kByte, 16 kByte or Ultimax sub-modules. These have to be placed bank-aligned, because they are not loaded, but banked in and started. This special feature may be useful for something like CRT collections.

That is the reason a directory may also have entries which refer to sub-cartridges.

Fig. 3 shows an example hybrid cartridge which contains a 16K sub-cartridge.

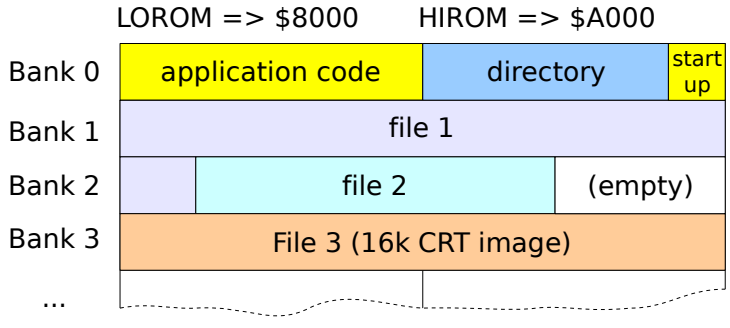

Fig. 3: Hybrid cartridge containing a normal 16K cartridge

4.4 Start-Up Code

Todo: Vorbereiteter Start-Up-Code kopiert Code zum Zugriff auf EasyFS nach $CF00, schaltet in den 16 KiB-Modus und ruft Code an $8000 auf.

5 Flash Write API

Todo: EasyProg soll automatisch auf die letzte Kachel ein paar Funktionen legen, die das Schreiben in den Flash ermöglichen. Sinnvoll für Spielstände o.ä.?

Aber nur, wenn die Kachel nicht ohnehin benutzt wird.

Dieses API ist nicht im eigentlichen CRT-Image enthalten, damit es in zukünftigen Versionen mit anderen Flash-Typen auch noch funktioniert.

Offen: Und wie geht das in Emulatoren?