Table of contents

1. Features and appearance

Flash cart for Super Nintendo systems

- Internal FLASH ROM size up to 128MBit (16Mb).
- Maximum supported ROM size up to 48MBit (6Mb).
- 32kb FRAM or Static RAM memory for game saves store.
- USB[1] development port. Support Assembler Studio, and step by step debug **with code execution on real console**.
- Upgradable OS from USB[1] and from external memory card.
- Game saves can be stored or loaded from SD card.
- SD/MMC/SDHC cards support.
- FAT12/FAT16/ FAT32 support. 32GB max external card size.
- SMARD DMA Module (Support fast external memory access up to 11Mbit/ps).
- Support FX **Capcom CX4** chip functionality.
- Memory paging support (64 ready to run maximum games store on one card).
- Automatic Save RAM memory management (extract need to run save ram then game store to internal ready to run memory).
- PAL and NTSC systems support, multiregional CIC (developed by Ikari_01 special thanks him for this).

1 - USB port available only on ENCHANCED 64SZ+UF card (see 2.0 possible configurations).

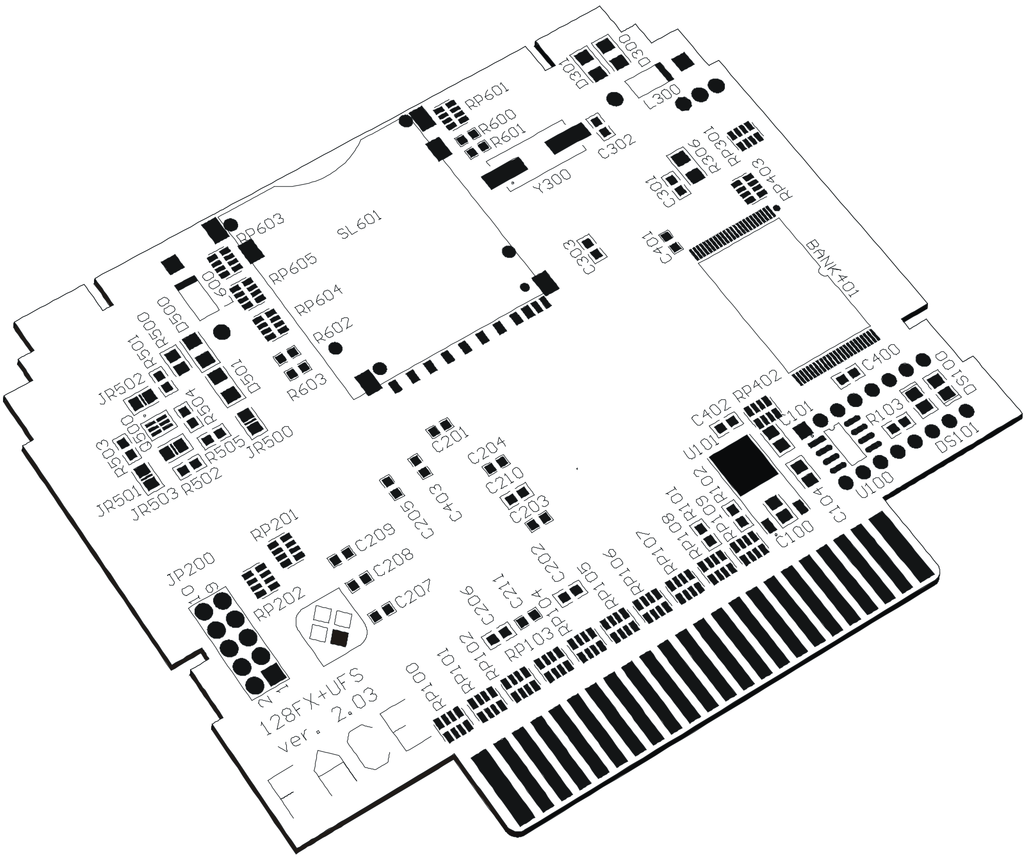

Figure 1.1. Device PCB top view.

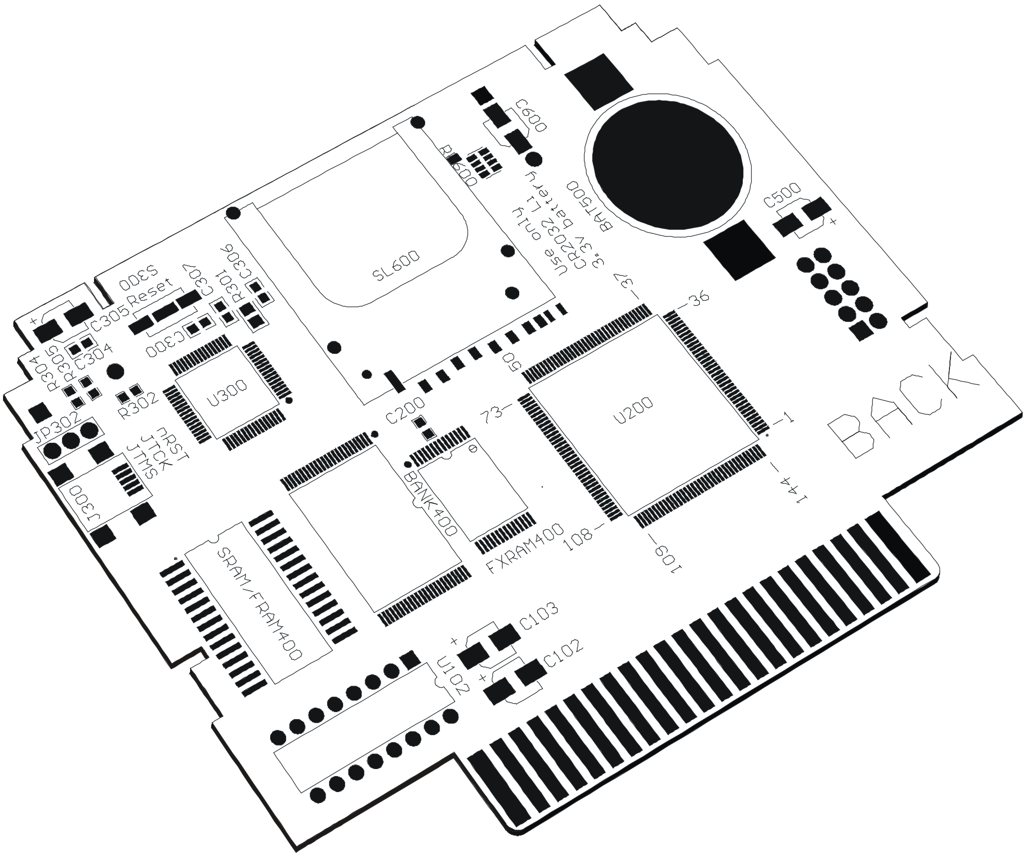

Figure 1.2. Device PCB bottom view.

2. Possible configurations

The device may have several types of configurations.

- Internal Memory Configurations: 8,12 or 16 Mb (this memory need to run games).
- Memory Card slots configuration, - External, Internal or both Card slots.
- Save RAM memory configuration, - 32k Nonvolatile Ferroelectric memory (FRAM) or battery backed Static RAM.
- USB copy and Debug interface, - can be installed or not. If USB Interface installed device Named as 64SZ+UF+E (E - Enhanced device feature)

3. Memory MAP

The device has three memory modules, - built-in FLASH ROM memory, Save RAM memory and working SMART DMA memory.

Addressing these types of memory, you can see in Table 3.1.

3.1 Memory Architectural Overview

Then SMART DMA in BUSY state all data reads from any memory card regions is invalid, except CORE Registers and SDMA Status Register (See 5.1 SMART DMA memory map).

Table 3.1. Device memory map.

Mode 20* (LOROM)

FF - F0	EF - E0	DF - D0	CF - C0	BF - B0	AF - A0	9F - 90	8F - 80	7F - 7E	7D - 70	6F - 60	5F - 50	4F - 40	3F - 30	2F - 20	1F - 10	0F - 00	BANK	ADDRESS
INTERNAL FLASH ROM BANKs 0/1								SNES RAM	INTERNAL FLASH ROM BANKs 0/1								FFFF	
RAM	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA		RAM	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	CX4 / SDMA	8000	
																	7FFF	
	FX REGs	FX REGs	FX REGs	FX REGs	FX REGs	FX REGs	FX REGs			FX REGs	FX REGs	FX REGs	FX REGs	FX REGs	FX REGs	FX REGs	6000	
	CORE REGs	CORE REGs	CORE REGs	CORE REGs	CORE REGs	CORE REGs	CORE REGs			CORE REGs	CORE REGs	CORE REGs	CORE REGs	CORE REGs	CORE REGs	CORE REGs	3FFF	
																	3000	
																	2007	
																	2000	
																	0000	

Mode 21* (HIROM)

FF-C0	BF - B0	AF - A0	9F - 90	8F - 80	7F - 7E	7D-40	3F - 30	2F - 20	1F - 10	0F - 00	BANK	ADDRESS
INTERNAL FLASH ROM ($000000-$3FFFFF)	INTERNAL FLASH ROM ($008000:$3FFFFF)				SNES RAM	INTERNAL FLASH ROM ($000000-$3D0000)	INTERNAL FLASH ROM ($008000:$3FFFFF)				FFFF	
	RAM	RAM	CX4 / SDMA	CX4 / SDMA			RAM	RAM	CX4 / SDMA	CX4 / SDMA	8000	
											7FFF	
	FX REGs	FX REGs	FX REGs	FX REGs			FX REGs	FX REGs	FX REGs	FX REGs	6000	
	CORE REGs	CORE REGs	CORE REGs	CORE REGs			CORE REGs	CORE REGs	CORE REGs	CORE REGs	3FFF	
											3000	
											2007	
											2000	
											0000	

Mode 25* (HIROM)

FF-C0	BF - B0	AF - A0	9F - 90	8F - 80	7F - 7E	7D-40	3F - 30	2F - 20	1F - 10	0F - 00	BANK	ADDRESS
INTERNAL FLASH ROM ($000000-$03FFFF)	INTERNAL FLASH ROM ($008000:$3FFFFF)				SNES RAM	INTERNAL FLASH ROM ($400000-$7DFFFF)	INTERNAL FLASH ROM ($408000:$7FFFFF)				FFFF	
	RAM	RAM	CX4 / SDMA	CX4 / SDMA			RAM	RAM	CX4 / SDMA	CX4 / SDMA	8000	
											7FFF	
	FX REGs	FX REGs	FX REGs	FX REGs			FX REGs	FX REGs	FX REGs	FX REGs	6000	
	CORE REGs	CORE REGs	CORE REGs	CORE REGs			CORE REGs	CORE REGs	CORE REGs	CORE REGs	3FFF	
											3000	
											2007	
											2000	
											0000	

A shaded area is main memory area, not shaded is memory image.

* - SNES console can support two different types of memory models, it's calls: HIROM and LOROM, 64SZ + UF device can switch between them, memory model changing provides by MCR register (see 4.3 MCR - Memory Configuration Register and 4.6 SRCR - Save RAM Configuration Register).

CORE REGs is a main CPLD core registers (see 4. Device FPGA Core for more detail)
RAM – Save RAM memory use in some games for store scores and game states. Before Use this memory block user must configure SAVE RAM by SRCR register (see 4.6 SRCR - Save RAM Configuration Register)
CX4/SDMA – is working SMART DMA Memory (see 5. SMART Direct Memory Access Module, for memory map and functions).

4. Device CPLD Core

4.1 Introduction

This section discusses the CPLD core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

4.2 Architectural Overview

Figure 4-1. Block Diagram of the CPLD Core Architecture

4.3 MCR - Memory Configuration Register

Access address: $xx2000 (xx – any address in range $00:$3F).

MCU register provide card change configuration, like ROM size and memory model, also able to disable access to the core module and SMATDMA module.
To prevent unauthorized writes, these register protected by WUA sequence module (see 4.9 Write Unlock Algorithm).

Bit	MCR0	MCR1	MCR2	MCR3	MCR4	MCR5	MCR6	MCR7
Name	RSC0	RSC1	RSC2	CRCD	LCA	LSDMA	MF	HIROM
Access Type	R/WUA	R/WUA	R/WUA	R/WUA	R/WUA	R/WUA	R	R/WUA

• RSC[2..0] - ROM Size Configuration

RSC[2..0] Bits group produce ROM size emulation. Then ROMs packed into device, device can cat out not used by selected ROM address space. See Table 4.1 for detail.

Table 4.1. ROM size supported configurations.

RSC[2..0]	Description
0	No Sized
1	4 MegaBits
2	8 MegaBits
3	16 MegaBits
4	24 MegaBits
5	32 MegaBits
6	Unused - Means No Sized
7	Unused - Means No Sized

• CRCD – Core Registers Clear Disable then external reset

Disable core reset then user push reset button on console. Actually device return to current game, not into OS or Bootloader.

• LCA – Lock Core Access

Then LCA Bit set as ONE, access to 64SZ+UF core will be blocked. This bit clear only then external reset applied.

• LSDMA – Lock Smart DMA and CX4 Access

Then LSDMA Bit set as ONE, access to 64SZ+UF Smart AMD and CX4. This bit clear only then external reset applied.

• MF – Memory Flash size detection bit (Read Only bit).
If bits set, flash memory chips installed in hardware have size is 4 Mbytes, not set – 8 MBytes. This bit is read only and cannot be changed by software.

• HIROM - HiROM CARD Configuration bit.
If HIROM bit set ONE, CARD switched into HiROM memory mode.

4.4 FRAR - FLASH ROM Access Register

Access address: $xx2001 (xx – any address in range $00:$3F).

FRAR register provide lock/unlock ROM writes support, ROM state and ROM reset support.
To prevent unauthorized writes, these register protected by WUA sequence module (see 4.9 Write Unlock Algorithm).

Bit	FRAR0	FRAR1	FRAR2	FRAR3	FRAR4	FRAR5	FRAR6	FRAR7
Name	RDY	NRRESET	UNLOCK	UWP0	UWP1	UWP2	UWP3	UWP4
Access Type	R	R/WUA	R/WUA=1	R/WUA=1	R/WUA=1	R/WUA=1	R/WUA=1	R/WUA=1

• RDY – FLASH ROM State.
RDY Register bit report FLASH ROM state, ONE – assume that ROM ready for write data or erase BANK. ZERO – ROM busy by command (erase or write).

• NRRESET – FLASH ROM Reset.
NRRESET Register bit, write into this register ONE to activate reset. NRRESET target connects to RESET pin on FLASH ROM chip(s), while access to reset, user mast be sure that no need access to reset ROM (for sample, Reset must be activated and deactivated from SRAM memory).

• UNLOCK – Unlock Write Into ROM.
UNLOCK Register bit activate write access into FLASH ROM memory, by default write access is disabled (UNLOCK = 1). Write into UNLOCK bit ZERO to unlock write into FLASH ROM memory, this bit does not produce write access into locked boot sectors, See UWP0/4.

• UWP0/4 – Unprotect Write into Page 0/4.
UWP Register bits produce protect support for boot sectors on device, by default FLASH ROM pages 0-4 is protected, write into UWP0/4 bit(s) ZERO to unprotect page 0/4.

4.5 MPSR - Memory Page Shift Register

Access address: $xx2002 (xx – any address in range $00:$3F).

MPSR register provide memory map shift and can shift memory map in range from 0 to 255PG (1PG = 64Kb).
To prevent unauthorized writes, these register protected by WUA sequence module (see 4.9 Write Unlock Algorithm).

Bit	MPSR0	MPSR1	MPSR2	MPSR3	MPSR4	MPSR5	MPSR6	MPSR7
Name	PGAD0	PGAD1	PGAD2	PGAD3	PGAD4	PGAD5	PGAD6	PGAD7
Access Type	R/WUA	R/WUA	R/WUA	R/WUA	R/WUA	R/WUA	R/WUA	R/WUA

• PGAD[7..0] – PAGE ROM Addressing group.

PGAD[7..0] Register bit group can separate all Internal ROM memory AREA into 256 pages with 512KiloBits each bank size. This bit group use to change memory according. By default PGAD[7..0] = $00.

4.6 SRCR - Save RAM Configuration Register

Access address: $xx2003 (xx – any address in range $00:$3F).

SRCR register provide Save RAM size managing and Save RAM memory map shift, SRCR can shift memory map in range from 0 to 15PG (1PG = 2Kb).

To prevent unauthorized writes, these register protected by WUA sequence module (see 4.9 Write Unlock Algorithm).

Bit	SRCR0	SRCR1	SRCR2	SRCR3	SRCR4	SRCR5	SRCR6	SRCR7
Name	SRSC0	SRSC1	SRSC2	HROM25	SRPA0	SRPA1	SRPA2	SRPA3
Access Type	R/WUA	R/WUA	R/WUA	0	R/WUA	R/WUA	R/WUA	R/WUA

• SRSC[2:0] - Save RAM Size Configuration.

SRSC Register bits group produce Save RAM size emulation, it may be used then 64SZ+UF CARD has onboard multi ROM storage with Save RAM page separation, or if need produce fixed Save RAM size, or remove Save RAM from card (some ROMs has Save RAM size check). See Table 4.2 for detail.

Table 4.2. Save RAM size supported configurations.

SRSC[2..0]	Description
0	NO SRAM
1	16 KiloBits
2	32 KiloBits
3	64 KiloBits
4	128 KiloBits
5	256 KiloBits
6	* 512 KiloBits
7	* 1024 KiloBits

* Not used in current device, seems NO SRAM.

• HROM25 – HiROM mode 25 CARD Configuration bit.

If bits HIROM and HROM25 set, CARD switched into HiROM memory mode 25. HROM25 supported by CORÉ version 2.08 or higher.

• SRPA[3:0] - Save RAM PAGE Addressing group.

SRPA Register bit group can separate 256KiloBits Internal RAM AREA into 16 pages with 16KiloBits each bank size (32KBytes=16x2KBytes). This bits group use to change Save RAM memory according.

4.7 SSR - SPI Machine State Register

Access address: $xx2004 (xx – any address in range $00:$3F).

SSR register use by SMART DMA to detect card move/remove and write protect card status detection.

Bit	SSR0	SSR1	SSR2	SSR3	SSR4	SSR5	SSR6	SSR7
Name	CD0	CW0	RES0	RES1	CD1	CW1	RES2	RES3
Access Type	R	R	0	0	R	R	0	0

• CD0/CD1 – Card Detect in slot 0/1.

CD Register bit signaling SD/MMC card slot state, if this bit ZERO, card inserted into slot, otherwise – not inserted.

• CW0/CW1 – Card Write Protected slot 0/1 detection.

CW Register bit signaling SD/MMC card slot write state, if this bit ONE, card inserted into slot is write protected otherwise, - unprotected.

4.8 CVR - Core Version Register

Access address: $xx2007 (xx – any address in range $00:$3F).

CVR Register consist current version of 64SZ+UF CPLD core.

Bit	EXR0	EXR1	EXR2	EXR3	EXR4	EXR5	EXR6	EXR7
Name	MJV0	MJV1	MJB2	MJV3	MIV0	MIV1	MIV2	MIV3
Access Type	R	R	R	R	R	R	R	R

• MJV[3..0] – Major Core version.

MJV[3..0] Produce major core version in range from 1.x:15.x.

• MIV[3..0] – Minor Core version.

MIV[3..0] Produce minor core version in range from x.0:x.15 ..

4.9 WUA - Write Unlock Algorithm

WUA - Core Unlock Algorithm use in cart to prevent unauthorized write access into some registers.

WUA work same as JEDEC write algorithm in FLASH ROMs, before write into WUA protected register user must write into cart unlock sequence.

Unlock sequence consist three cycles, see table 4.4.

Table 4.4. Registers write access map.

Regsiter	Lenght	Bus Write operations							
		1st		2nd		3rd		4th	
		Addr	Data	Addr	Data	Addr	Data	Addr	Data
$xx2000 (MCR)	4	$AAA	$55	$555	$AA	$AAA	$30	RA	RD
$xx2001 (FRAR)	4	$AAA	$55	$555	$AA	$AAA	$30	RA	RD
$xx2002 (MPSR)	4	$AAA	$55	$555	$AA	$AAA	$30	RA	RD

$xx2003 (SRCR)	4	$AAA	$55	$555	$AA	$AAA	$30	RA	RD
$xx2004 (SSR)	1	RA	RD						
$xx2007 (CVR*)	0	n / a	n / a						

RA – Register Address.
RD – Register Data.
* - Write into register CVR is unavailable.

5. SMART Direct Memory Access Module

This section discusses the SMART DMA features, fast access to SPI machine, fast division, multiply, FAT file systems and CX4 chip support.

5.1 SMART DMA memory map.

SMART DMA Module provides support for the most resource consuming functions like multiply, division, trigonometric and etc. functions.

SMART DMA Module has 4096 Bytes of SRAM which can be accessed from SNES Bus, this memory also use for CX4 chip support. See table 5.1 for more detail.

Table 5.1. SMART DMA memory map.

Address	Name	Description
$6000 ~ $6BFF	Register File	SDMA and CX4 Work RAM (WRAM)
$6C00 ~ $7F3F	4928x8Bit RAM Buffer 1	Only SDMA First Buffer (FBRAM)
$7F40 ~ $7BAF	12x24Bit Registers	CX4 Registers group one
$7FB0 ~ $7FFF	80x8Bit RAM Buffer 2	Second Buffer (SBRAM, not used in SDMA)

CX4 and SMART DMA has 3 registers, 2 for commands, and 1 for status.
Status register, named **SR** has address 0x7F5E, and can be read any time (even when the SMART DMA is busy processing commands)
First command register named **CR** has address 0x7F4F, and use for commands send to SMART DMA, writes in this register is possible only when the SMART DMA not busy processing commands.
Second command register named **CLDA** (CX4 Load Memory Access) has address 0x7F47, and use only for CX4 special LDMAC command, writes in this register is possible only when the SMART DMA not busy processing commands.

Table 5.2. SMART DMA Registers MAP.

Address	Register Name
$7F80	R0
$7F83	R1
$7F86	R2
$7F89	R3
$7F8C	R4
$7F8F	R5
$7F92	R6
$7F95	R7
$7F98	R8
…	…

Before SMART DMA performs any operation, the user must fulfill three conditions:

1. Wait while bit 4 in SR register is clear.
2. Fill data need for function processing and write command into CR register (see SMART DMA features).
3. Wait while bit 4 in SR register is clear.

5.2 SMART DMA features

This section describe SMART DMA File System IO Functions. All parameters must be LSB (Least Significant Byte).

Init drive.

Parameters:

R0 = $*0000NN*

NN - Drive Number (0 or 1).

CR = $*90*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero. See table 5.3 File function return codes.

FindFirst identical command.

Parameters:

WRAM = "*szString*"

"*szString*" = Directory name. Zero ended ASCII String ("Directory/Sub Directory")

CR = $*91*

Returns:

If function success return value in LOBYTE of R0 is ZERO over wise non zero. See table 5.3 File function return codes.

FindNext identical command.

Parameters:

CR = $*92*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

If function success, SMART DMA fill into WRAM File Info Structure, this function only valid after FindFirst command:

Table 5.3. File Info structure

Address	Size (bytes)	Name	Description
$6000	4	dwSize	File size
$6003	2	wDate	Date stamp
$6005	2	wTime	Time stamp
$6007	1	bFileAttr	File attributes
$6008	13	szFileName	8.3 File Name ascii zero ended string
$601B	2	wLFNLen	Long file name linght
$601D	wLFNLen	xchFileName	Unicode Long File Name

FOpen identical command.

Parameters:

R0 = $*00FFNN*

NN – File number (0 or 1, FOpen command can process 2 files at same time, it means you can open in first stage file 0, in second stage file 1 and read data from file 0 and write read data into file 1, See file commands FRead/FWrite)

FF – open flags, it can by any combinations of:

FA_READ = $01
FA_WRITE = $02
FA_CREATE_ALWAYS = $08
FA_OPEN_ALWAYS = $10
FA_CREATE_NEW = $04
WRAM = = "*szString*"
"*szString*" = File name. Zero ended ASCII String ("sample.txt")
CR = $*93*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

FRead identical command.

Parameters:

R0 = $*SSSSNN*

NN – File number (0 or 1, FRead command can process 2 files at same time, it means you can open in first stage file 0, in second stage file 1 and read data from file 0 and write read data into file 1, See file commands FOpen, FWrite)

SSSS – Data size for read, maximum data size is $C00 and this parameter cannot be $000.

CR = $*94*

Result:

If function success return value in R0[7..0] is ZERO over wise non zero, See table 5.3 File function return codes.

If function success, SMART DMA fill into WRAM file data and R0[23..8] contains the number of readed data bytes.

FWrite identical command.

Parameters:

R0 = $*SSSSNN*

NN – File number (0 or 1, FRead command can process 2 files at same time, it means you can open in first stage file 0, in second stage file 1 and read data from file 0 and write read data into file 1, See file commands FOpen, FWrite)

SSSS – Data size for write, maximum data size is $C00 and this parameter cannot be $000.

WRAM = *lpBytes[$C00]*

lpBytes[$C00] – Data for write process

CR = $*95*

Result:

If function success return value in R0[7..0] is ZERO over wise non zero, See table 5.3 File function return codes.

If function success, R0[23..8] contains the number of written data bytes.

FClose identical command.

Parameters:

R0 = $*0000NN*

NN – File number (0 or 1)

CR = $*96*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

FSeek identical command.

Parameters:

R0 = $*0000NN*

NN – File number (0 or 1)

R1 = $*SSSSSS*

SSSSSS – Seek position from file start.

CR = $*97*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

FState identical command.

Parameters:

WRAM = = "*szString*"

"*szString*" = File name. Zero ended ASCII String ("sample.txt")

CR = $98

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

If function success, SMART DMA fill into WRAM File Info Structure, see table 5.2 for detail.

FDelete identical command.

Parameters:

WRAM = = "*szString*"

"*szString*" = File name. Zero ended ASCII String ("sample.txt")

CR = $*99*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

MKDir identical command.

Parameters:

WRAM = "*szString*"

"*szString*" = Directory name. Zero ended ASCII String ("Directory")

CR = $*9A*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

FRename identical command.

Parameters:

WRAM = File name for rename. Zero ended ASCII String ("old.txt")

WRAM + $600 = New file name. Zero ended ASCII String ("new.txt")

CR = $*9B*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.

FGetFree command.

FGetFree return number of free clusters on drive.

Parameters:

R0 = $*0000NN*

NN – Number of previously initialized drive (0 or 1)

CR = $*9C*

Result:

If function success return value in LOBYTE of R0 is ZERO over wise non zero, See table 5.3 File function return codes.
If function success WRAM[3..0] consist number of free clusters. Return number is DWORD - Most Signed Byte.

DMAMemoryMove command.

Move File To File/File, File to Memory or Memory to File data.

Parameters:

WRAM = *lpBytes[$C00]*
R0 = $*SSSSFN*
R1 = *[$DDDDDD]*
R2 = $*WWRRBB*

lpBytes[$C00] – Data for write process (use only if command parameters, – WRAM copy to Flash Memory Destination).
N – Command Type.
F – Transfer direction.

- N=1 – Transfer File to file
  F=0 – Transfer File[0] to File[1]
  F=1 – Transfer File[1] to File[0]
- N=2:
  F=0 – Transfer File[0] to Flash Memory Destination indicated by **R1** register
  F=1 – Transfer File[*1*] to Flash Memory Destination indicated by **R1** register
  F>1 – Transfer WRAM to Flash Memory Destination indicated by **R1** register
- N=3 – Transfer Any Memory Destination (**R1**) to File[*F*]
- N=4 – Transfer File[*F*] to Ram Memory Destination indicated by **R1** register

SSSS – Data size, maximum data size is $C00 and this parameter cannot be $000.
[$DDDDDD] - Flash Memory Destination (address)
BB – Flash BANK (see 4.5 MPSR - Memory Page Shift Register).
RR – SRAM Setup Register (see 4.6 SRCR - Save RAM Configuration Register).
WW – Write Data Algorithm (0 – Standard Unlock Bypass by bytes, 1 – Octuple Bytes Program).

CR = $*9D*

Result:

R0 Consists File Read Statuses, R1 Consists File Write Statuses, bits 8-23 of R0 and R1 consists numbers of processed bytes for Read (R0[23..8]), and Write (R1[23..8]).
If function success return value in R0[7..0] and R1[7..0] is ZERO otherwise non zero, See table 5.3 File function return codes.

FilesList command.

This command process files list enumeration.

Parameters:

CR = $*9E*

Result:

If function success return value of R0[7..0] is ZERO and R0[23..8] consist number of enumerate entries, R1[15..0] number of stored bytes.
This function only valid after FindFirst or FindNext command.

case 0x9F: //

FlashCMD.

Erase FLASH Page or Read MF ID.

Parameters:

R0 = $*00BBNN*

R1 = *[$DDDDDD]*

CR = $*9F*

NN – Command Type (0 – read manufacturer ID, 0x01:0xFF – Erase FLASH sector).

BB – Flash BANK (see 4.5 MPSR - Memory Page Shift Register).

[$DDDDDD] - Flash Memory Destination (address)

Result:

If function success return value of R0[7..0] is ZERO, R0[23..8] and R2[23..8] consist manufacturer id numbers (only for read manufacturer ID command).

WriteFlash.

Write SMART DMA Buffered data to FLASH.

Parameters:

WRAM = *lpBytes[$C00]*

R0 = $*SSSSBB*

R1 = *[$DDDDDD]*

R2 = *$0000WW*

CR = $*A0*

lpBytes[$C00] – Data for write process.

BB – Flash BANK (see 4.5 MPSR - Memory Page Shift Register).

SSSS – Data size, maximum data size is $C00 and this parameter cannot be $000.

[$DDDDDD] - Flash Memory Destination (address)

WW – Write Data Algorithm (0 – Standard Unlock Bypass by bytes, 1 – Octuple Bytes Program).

Result:

If function success return value of R0[7..0] is ZERO.

ReadLongFlashAddress.

Read FLASH data into SMART DMA buffer.

Parameters:

R0 = $*SSSSBB*

R1 = *[$DDDDDD]*

CR = $*A1*

BB – Flash BANK (see 4.5 MPSR - Memory Page Shift Register).

SSSS – Data size, maximum data size is $C00 and this parameter cannot be $000.

[$DDDDDD] - Flash Memory Source (address)

Result:

If function success return value of R0[7..0] is ZERO, and WRAM consist received data.

Disable.

Disable command, permanent switch off access to FILE IO functions while external reset does not applied (SNES RESET button).

Parameters:

CR = $*AF*

Result:

Permanent disable FILE IO Functions.

Table 5.3. File function return codes.

Name	Value	Description
_OK	$0	Success
_DISK_ERR	$1	Disk error
_INT_ERR	$2	Internal error
_NOT_READY	$3	Drive is not ready
_NO_FILE	$4	Reached to end of File table No file to open
_NO_PATH	$5	Could not reach the dir
_INVALID_NAME	$6	NULL string or cannot create dot entry
_DENIED	$7	No free cluster End Of Table and could not stretch No free entry or too many SFN collision Not empty sub-dir (for Delete function)
_EXIST	$8	Object name is already exist
_INVALID_OBJECT	$9	File system object is invalid
_WRITE_PROTECTED	$A	Disk is write protected
_INVALID_DRIVE	$B	Invalid Drive number
_NOT_ENABLED	$C	Is the file system object is not available
_NO_FILESYSTEM	$D	No valid FAT partition is found
_INV_PARAMS	$E	Invalid Parameter(s)
_TIMEOUT	$F	Timeout achieved while function processed

5.3 CX4 enhanced chip features

The SMART DMA can process ALL CX4 functions, it gives the device compatibility with games which using CX4 enhanced chip.
Supported by SMART DMA CX4 command you can see in table 5.4.

Table 5.4. CX4 Commands List.

Command Value	Description
$00	Sprite
$01	Draw wireframe
$05	Propulsion
$0D	Set vector length
$10	Polar to rectangular
$13	Polar to rectangular
$15	Pythagorean
$1F	Arctangent
$22	Trapezoid
$25	Multiply
$2D	Transform Coordinates
$40	Sum
$54	Square
$5C	Immediate Registers
$89	Immediate ROM

SMART DMA process shift command, then user writes into device sequence described below:

$7F4D <= $0E
$7F4E <= $00
$7F48 <= $00
$7F4F <= $XX Any numbers between 0 and $3F.