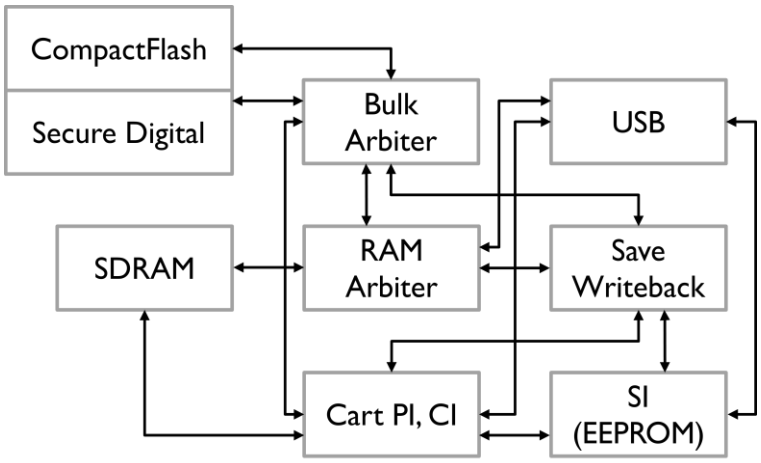
HARDWARE USAGE GUIDE

(FIRMWARES 2.XX)

OVERVIEW

All of the 64drive's functionality is facilitated by the onboard FPGA. Automatically configured on every power-on, the FPGA's job is to coordinate communication between components on the 64drive circuit board such as SDRAM, USB, and the PI bus. In addition to emulating the N64's cartridge bus protocol, the FPGA also emulates all N64 save types.

BLOCK DIAGRAM



MODES OF OPERATION

	Bootloader	CIC Emu	Save Writeback	Save Emulation	CI Access	USB
MENU MODE	✓	✓	✓	✓	✓	✓
DEV MODE		[✓ HW2 Only]		✓	✓	✓

MENU MODE

Upon power-up in an N64 console, the 64drive boots into menu mode. In this mode, the N64 boots from the bootloader, which then locates MENU.BIN from the memory card, transfers to SDRAM and runs it.

DEV MODE

When the user plugs in a USB cable the 64drive switches to dev mode. This mode most closely resembles a real, single-image cartridge. It is intended that the user uploads an image to SDRAM using PC-side software prior to booting the N64.

While save chips may be emulated, any changes in their data will not be flushed back to the memory. You must manually upload and download the contents of these save memory banks with the companion PC software.

CI (CARTRIDGE INTERFACE)

OVERVIEW

The 'CI' interface is a small range of memory in the upper area of the PI address space that contains registers and some block memory associated with the 64drive hardware.

The interface is mapped into the cartridge address space at the following base locations:

BASE ADDRESS	NAME	HARDWARE AVAILABILITY
0x1800 0000	Regular address mode	HW1, HW2
0x1F80 0000	Extended address mode	HW2 only

REGISTERS

ADDRESS	SIZE	FUNCTION	READ	WRITE
BASE+0x0000	512 bytes	General Buffer	✓	✓
BASE+0x0200	16 bits	Status	✓	
BASE+0x0208	32 bits	Command		✓
BASE+0x0210	32 bits	LBA		✓
BASE+0x0218	32 bits	Length	✓	✓
BASE+0x0220	32 bits	Result	✓	
BASE+0x02E8	32 bits	SDRAM Size (bytes)	✓	
BASE+0x02EC	32 bits	Hardware Magic	✓	
BASE+0x02F0	32 bits	Hardware Variant	✓	
BASE+0x02F4	32 bits	Persistent Variable Storage [**Reserved**]	✓	✓
BASE+0x02F8	16 bits	Button Status	✓	
BASE+0x02FA	16 bits	Upgrade Module Status	✓	
BASE+0x02FC	16 bits	Revision Number	✓	
BASE+0x0400	8 bits	USB Command/Status	✓	✓
BASE+0x0404	32 bits	USB Param0/Result0	✓	✓
BASE+0x0408	32 bits	USB Param1/Result1	✓	✓
BASE+0x0420	8 bits	WIFI Command/Status	✓	✓
BASE+0x0424	32 bits	WIFI Param0/Result0	✓	✓
BASE+0x0428	32 bits	WIFI Param1/Result1	✓	✓

BASE+0x1000	2048 bytes	EEPROM contents	✓	✓
BASE+0x1800	1024 bytes	Save Writeback LBA List (256 dwords)	✓	✓

TO PERFORM A COMMAND:

1. Check the status register to make sure CI is idle
2. Write all necessary parameters to the appropriate registers
3. Write the command code to the command register
4. Poll status for completion (same as step 1)

CI STATUS REGISTER TABLE

Bit Range	[15:12]	[11:8]	[7:4]	[3:0]
Function	0x0 – CI Idle 0x1 – CI Busy	Reserved	Reserved	Reserved

CUI STATUS REGISTER TABLE

Bit Range	[7:4]	[3:0]
Meaning	WRITE STATUS	ARM STATUS
Function	0x0 – Idle 0xF – Busy	0x0 – Idle 0xF – Busy 0x1 – Armed, no data 0x2 – Unarmed, data rx

CWI STATUS REGISTER TABLE

Bit Range	[7:4]	[3:0]
Meaning	WRITE STATUS	ARM STATUS
Function	0x0 – Idle 0xF – Busy	0x0 – Idle 0xF – Busy 0x1 – Armed, no data 0x2 – Unarmed, data rx

CI COMMAND TABLE

COMMAND	ID	PARAMETERS
READ SECTOR INTO BUFFER	0x01	LBA
READ MULTIPLE SECTORS TO SDRAM	0x03	LBA, RAMADDR, NUM_SEC
WRITE BUFFER INTO SECTOR	0x10	LBA
WRITE MULTIPLE SECTORS FROM SDRAM	0x13	LBA, RAMADDR, NUM_SEC
RE-INITIALIZE SD CONTROLLER	0x1F	-
SET SAVE TYPE	0xD0	SAVE_TYPE
DISABLE SAVE WRITEBACK	0xD1	-
ENABLE SAVE WRITEBACK	0xD2	-
DISABLE BYTESWAP ON LOAD	0xE0	-
ENABLE BYTESWAP ON LOAD	0xE1	-
ENABLE CARTROM WRITES	0xF0	-
DISABLE CARTROM WRITES	0xF1	-
ENABLE EXTENDED ADDRESS MODE	0xF8	-
DISABLE EXTENDED ADDRESS MODE	0xF9	-
START FIRMWARE UPGRADE	0xFA	MAGIC
SET CF PULSE WIDTH	0xFD	NUM_CYCLES
ABORT COMMAND	0xFF	-

All reserved bitfields should be masked off and ignored, as they may be used in future versions for additional functionality.

CUI COMMAND TABLE

COMMAND	ID	PARAMETERS
ARM USB FIFO DMA	0x0A	SOURCE_ADDR, SOURCE_LEN
DISARM USB FIFO DMA	0x0F	-
WRITE USB FIFO DMA	0x08	TARGET_ADDR, MAX_LEN

CWI COMMAND TABLE

COMMAND	ID	PARAMETERS
ARM WIFI UART DMA	0x0A	SOURCE_ADDR, SOURCE_LEN
DISARM WIFI UART DMA	0x0F	-
WRITE WIFI UART DMA	0x08	TARGET_ADDR, MAX_LEN

COMMAND LISTING

MEMORY CARD ACCESS

When power is first applied, the 64drive tests to see if a CF card is inserted. If so, it will always use it until power is lost. If no CF was detected, SD will be used instead. SD is only initialized once upon powerup, so hot-swapping is not supported.
On HW2 (Rev B), only microSD is supported.

READ SECTOR INTO BUFFER

Reads a single sector from the memory card into the main buffer.

BASE+0x0210	[LBA REG]	LBA (sector number)

READ MULTIPLE SECTORS TO SDRAM

Copies any number of sectors from the memory card to an address in SDRAM.

BASE+0x0210	[LBA REG]	LBA (sector number)
BASE+0x0218	[LENGTH REG]	Number of sectors
BASE+0x0004	[in buffer]	SDRAM address (addressed in 16bit word units)

WRITE SECTOR FROM BUFFER

Writes the FPGA's 512 byte buffer to the memory card sector specified in the LBA register.

BASE+0x0210	[LBA REG]	LBA (sector number)

WRITE MULTIPLE SECTORS FROM SDRAM

Copies any number of sectors to the memory card from an address in SDRAM.

BASE+0x0210	[LBA REG]	LBA (sector number)
BASE+0x0218	[LENGTH REG]	Number of sectors
BASE+0x0004	[in buffer]	SDRAM address (addressed in 16bit word units)

RE-INIT SD CONTROLLER – **FIRMWARE 2.??+ ONLY**

Restarts the SD card controller. Necessary if a card has been removed, or if the interface hangs.

SET CF PULSE WIDTH

Sets the pulse width (in number of clock cycles) of OE and WE for the CompactFlash interface (HW1 only). Currently the core FPGA logic runs at 50MHz so each cycle period is 20ns. Setting this value too low may cause data corruption. The bootloader optimizes this number on bootup to get faster transfers.

`BASE+0x0000`	Clock cycles (Default: 10 → 200ns)

DISABLE BYTESWAP ON LOAD

Sets normal byte ordering (Z64).

ENABLE BYTESWAP ON LOAD

Causes all data transferred directly from the memory card to SDRAM to be byteswapped, for loading V64 format images. This has no effect on copying from the buffer to SDRAM. Transfers (writes) back to memory card are not affected. The default is `DISABLED`.

SAVE MEMORY EMULATION

SET SAVE TYPE

The 64drive supports all save types found in commercial software images. This command sets the type of save that the FPGA should emulate.

Note: Due to limitations in HW1 (Rev A), the Pokemon Stadium 2 images are handled specifically.

BASE+0x0000	Save type: 0 = None 1 = EEPROM 4Kbit 2 = EEPROM 16Kbit 3 = SRAM 256Kbit 4 = FlashRAM 1Mbit 5 = SRAM 768Kbit (for Dezaemon 3D) 6 = FlashRAM 1Mbit (for Pokemon Stadium 2)

⚠ WARNING

Do not enable saving until the LBA writeback list is populated. The LBAs are zero upon powerup. If the SWB (Save Writeback module) finds a zero-value LBA, it will cancel the operation to prevent FAT corruption.

DISABLE SAVE WRITEBACK

Inhibits the Save Writeback module (SWB), preventing any save persistence. This is automatically set upon USB cable insertion (**USB MODE**).

ENABLE SAVE WRITEBACK

Default mode when booting in menu mode. Throughout gameplay in **MENU MODE**, periodically flushes new savegame data back to the memory card via the Save Writeback LBA list.

MISCELLANEOUS

ENABLE CARTROM WRITES

Normally, reads from cartridge space (0x1000 0000-CI BASE) are mapped to SDRAM, while writes to this space are ignored. This command allows writes to this range to fall through to SDRAM. Necessary for setting up a multi-block DMA from RAM to memory card. The default is DISABLED.

DISABLE CARTROM WRITES

Locks cartridge ROM (SDRAM) against direct modification by the N64. This is the default state on boot in both menu and dev mode.

START FIRMWARE UPGRADE

Initiates the on-chip firmware flashing engine. Special firmware blob must be written to SDRAM in preparation. The upgrade module will read the blob and perform several verifications.

If all checks pass, it flashes the configuration ROM with the new data.

If there is any failure, the module will stop and lock itself out until the next power cycle.

The upgrade module can be kick-started either from CI or the USB interface. In each case, the same status codes are read out to check on the upgrade's progress.

BASE+0x0000	Magic (0x55504752)

ENABLE EXTENDED ADDRESS MODE – **HW2 FIRMWARE 2.06+ ONLY**

Sets cartridge ROM emulation range:

(0x1000 0000- 0x1F7F FFFF) are mapped to SDRAM for a total of 240Mbyte SDRAM exposed to the N64.

Sets CI BASE address to 0x1F80 0000.

The default is DISABLED.

DISABLE EXTENDED ADDRESS MODE – **HW2 FIRMWARE 2.06+ ONLY**

Resets cartridge ROM emulation range:

(0x1000 0000-0x13FF FFFF) are mapped to SDRAM for a total of 64Mbyte SDRAM exposed to the N64.

Resets CI BASE address to 0x1800 0000.

As default, this mode is functionally identical on HW1 and HW2 units, and is for backwards compatibility with both earlier homebrew and some commercial games that expect to see unmapped space up high.

ABORT COMMAND – **FIRMWARE 2.??+ ONLY**

Aborts some commands currently in progress. For example, can be used to stop a lengthy memory card DMA.

ADDITIONAL REGISTERS

In addition to the registers used for commands, there are several other registers that contain information about the device.

SDRAM SIZE

The total number in bytes of SDRAM on the device.

HW1, Rev A	67108864 bytes (64Mbyte).
HW2, Rev B	268435456 bytes (256Mbyte). NOTE: Only 240MB is mapped into cartridge space due to addressing limitations of the N64 PI bus.

DEVICE MAGIC

A four-character ASCII string with the product series code – “UDEV”

DEVICE VARIANT

An ASCII variant letter identifying the specific variation of the device. Existing hardware revisions return a letter and an ASCII null terminator.

HW1, Rev A	0x4100 (“A “)
HW2, Rev B	0x4200 (“B “)

Future revisions may not contain a null terminator, as in the case of a two-letter variant.

Make sure to mask off the upper 16 bits and check the lower 16 bits only.

REVISION (VERSION) REGISTER

The lower 16 bits of this register contain the FPGA configuration revision number. Make sure to mask away the upper 16 bits, since those bits are unspecified and vary at runtime.

For example, a value of decimal “203” means version 2.03.

This information is displayed in the About tab of the menu.

PERSISTENT VARIABLE STORAGE

A 32-bit register that is used by the bootloader. On startup, it writes its own version number to this register so that the menu can read it later on. Soft reset behavior is determined by the value written here by the menu when starting software.

Applications should not touch this value.

BUTTON REGISTER

The state of the hardware button. The button input is not filtered, so debouncing should be performed in software. Nonzero when depressed, zero otherwise.

UPGRADE STATUS REGISTER

The current response from the Upgrade Module (UPG).

Bit Range	[15]	[14:4]	[3:0]
Function	`0x0` – Not valid `0x1` – Status Valid	Reserved	`0x1` – Ready `0x2` – Verifying `0x3-0x6` – Erasing `0x7-0xA` – Writing `0xC` – Success `0xD` – General fail `0xE` – Bad variant `0xF` – Verify failed

EEPROM CONTENTS

A 2048-byte buffer that stores the contents of the EEPROM save chip being emulated. The menu writes here when loading an image if that image has EEPROM save data associated with it. During gameplay, when the game writes save data it will be stored in this buffer.

If the **SET SAVE TYPE** command is executed with the parameter '1', the valid size is 512 bytes. If it is executed with '2', the valid size is 2048 bytes.

⚠ WARNING

This memory is muxed three ways among USB, CI, and the Save Writeback module, be careful of monopolizing this region.

SAVE WRITEBACK LBA LIST

A 1024-byte (256 LBA) buffer that stores a list of LBAs occupied by the image's save file. The LBAs are stored as 32 bits each, although internally 27 bit LBA addressing is used.

Upon a save writeback event (usually triggered by the N64 modifying the save memory followed by a timeout) the save data is flushed back to the memory card by writing the save data to these sectors. Such an event can be noticed by carefully watching the status LED on the circuit board inside the case. It will flash in a regular pattern for about half a second.

In 1.xx series firmware this only happened due to Reset/NMI firing. In 2.xx this can occur at any time.

The writeback state machine performs writes to contiguous runs of LBAs in a single pass so that a non-fragmented file will be written in 1 pass. If the save file is fragmented, it will be written in multiple runs.

USB INTERFACE

OVERVIEW

USB connectivity on recent HW2/RevB units is provided by a FT232H USB 2.0 high-speed chip running in Synchronous FIFO mode, while on older HW1/RevA, a very similar FT2232H running in Asynchronous FIFO mode is used.

In order to provide a more useful experience, 64drive uses a simple block/command based protocol that is handled completely in hardware. To provide some more useful high-level operations such as sending atomic blocks to and from the device we need to build a simple framework on top of the dumb FIFO – encapsulating data blocks and device operations.

INTERFACE OPERATIONS

Target-side operations are those driven by the software running on the N64. For example, a program could print debug messages over the USB pipe to be displayed on the PC, or request the PC send the contents of a certain file. These operations are passed transparently through the host-side interface by the hardware.

Host-side operations are those controlled directly by the PC and deal with operations such as loading an image, setting the save type, dumping a chunk from cartridge ROM, and so on. All host-side operations are handled transparently by the hardware, requiring no intervention from the N64. Furthermore, most operations can even be performed while the N64 is streaming data.

⚠ **Firmware 2.05+ is required to use the Target-side interface.**

⚠ **Firmware 2.00+ is required to use the Host-side interface.**

Target-side operations require PC software to be running to monitor USB. The USB loader example provides a debug server that can be extended to provide desired functionality.

TARGET-SIDE USB INTERFACE

ARM USB FIFO DMA

Sets up a buffer and prepares the hardware for a block of data to be received from the host PC.
Data must be sent by the PC either before or during the operation.
DMA will remain in progress until the N64 accepts the data.

Please refer to the CUI register and command tables described earlier. The steps on N64 are as follows:

1. Read status register and confirm interface may be armed.

`BASE+0x0400` [Command/Status]	Must be `0x0` or `0x2`

2. Write the details of the buffer space in cart SDRAM you want to allocate:

`BASE+0x0400` [Command/Status]	`0x0A` (ARM)
`BASE+0x0404` [Param0/Result0]	Buffer address in SDRAM (addressed in 16bit word units) Must be 64bit aligned.
`BASE+0x0408` [Param1/Result1]	Maximum length that can be copied to buffer. Must be 32bit aligned. Maximum 8MByte

3. Check status until data is sent by PC (please see Host-side interface command `0x40`)

`BASE+0x0400` [Command/Status]	0x1 – Armed, no data yet 0x2 – Unarmed, data received in buffer

4. Read back the result of the operation.

`BASE+0x0400` [Command/Status]	Must be `0x2` (data in buffer)
`BASE+0x0404` [Param0/Result0]	[31:24] – Block type (8 bits) [23:0] – Block bytes transferred in this arming
`BASE+0x0408` [Param1/Result1]	[23:0] – Block bytes remaining to be transferred

5. Repeat from step 1 if more data still remains. This allows large blocks of data to be received even if the armed buffer is as small as 8 bytes, though typical buffers could be around 64KB.

Data may be sent by the PC first, if so, it will remain in the FIFO hardware until it is serviced by the N64. This data must be serviced or it will block any other commands from being sent by the PC, even on the Host-side interface.

WRITE USB FIFO DMA

Copies a block of data anywhere in cartridge RAM to the USB interface.

Data must be received by the PC either before or during the operation. Generally, the PC should be running the debug server to service the data and prevent the pipe from clogging.

DMA will remain in progress until the host PC accepts the data.

Please refer to the CUI register and command tables described earlier. The steps on N64 are as follows:

1. Read status register and confirm interface will allow a write.

`BASE+0x0400`	[Command/Status]	Must be `0x0`

2. Write the details of the buffer space in cart SDRAM you want to send:

`BASE+0x0400`	[Command/Status]	`0x08` (WRITE)
`BASE+0x0404`	[Param0/Result0]	Buffer address in SDRAM (addressed in 16bit word units) Must be 64bit aligned.
`BASE+0x0408`	[Param1/Result1]	[31:24] – Block type (8 bits) [23:0] – Block bytes to send Must be 32bit aligned. Maximum 8MByte

3. Check status until data is received by PC

`BASE+0x0400`	[Command/Status]	0xF – Busy / waiting 0x0 – Idle / data sent

4. Done

DISARM USB FIFO DMA

Cancels an armed buffer. USB subsystem will stop waiting for data.

Please refer to the CUI register and command tables described earlier. The steps on N64 are as follows:

1. Read status register and confirm interface is armed and not busy.

`BASE+0x0400`	[Command/Status]	Must be `0x1`

2. Write the command:

`BASE+0x0400`	[Command/Status]	`0x0F` (DISARM)

3. Done

HOST-SIDE USB INTERFACE

COMMAND TABLE

COMMAND	ID	P#	PARAM1	PARAM2	D_TX	D_RX	CMP
LOAD FROM PC	0x20	2	BANK_ADDR	BANK_ID_LEN	✓	-	✓
DUMP TO PC	0x30	2	BANK_ADDR	BANK_ID_LEN	-	✓	✓
TARGET-SIDE FIFO	0x40	1	CHK_TYPE_LEN	-	✓	-	✓
SET SAVE TYPE	0x70	1	SAVE_TYPE	-	-	-	✓
SET CIC TYPE	0x72	1	CIC_TYPE	-	-	-	✓
SET CI EXTENDED	0x74	1	ENABLE	-	-	-	✓
VERSION REQUEST	0x80	-	-	-	-	✓	✓
UPGRADE START	0x84	-	-	-	-	-	✓
UPGRADE REPORT	0x85	-	-	-	-	✓	✓
STD ENTER	0x88	-	-	-	-	-	✓
STD LEAVE	0x89	-	-	-	-	-	✓
STD PI READ 32	0x90	1	PI_ADDR	-	-	✓	✓
STD PI WRITE 32	0x91	1	PI_ADDR	-	✓	-	✓
STD PI RD BURST	0x92	1	PI_ADDR	-	-	✓	✓
STD PI WR BURST	0x93	1	PI_ADDR	-	✓	-	-
STD PI WRITE VEND	0x94	1	PI_ADDR	-	✓	-	-
STD PI WRITE VEND	0x95	1	PI_ADDR	-	✓	-	-
STD PI WRITE 16	0x96	1	PI_ADDR	-	✓	-	-
STD PI WRITE 16 DQ7 EXIT	0x97	1	PI_ADDR	-	✓	-	-
STD SI OP	0x98	3	-	-	✓	✓	✓

As a reminder, these are all commands sent by the PC. For running N64 software to communicate, please see Target-side Interface.

SENDING A COMMAND

⚠ All data sent through USB should be treated as big-endian, including command parameters and arguments.

1. Write the 4-byte command packet: `{CMD_ID 0x43 0x4D 0x44}`
   For example, SET CIC TYPE would be `0x72434D44`
2. Write any parameters or variable-length data required by the command.
   All parameters are 32bits each, data is always 32bit aligned length.
3. Receive any data caused by the command's execution, if applicable.
4. Receive the 4-byte completion packet: `{0x43 0x4D 0x50 CMD_ID}`
   Throw an error if the packet doesn't match.
   Some commands do not send this packet to speed up repeated operation.

For example, for the command DUMP TO PC:

- Write `{0x30434D44}` (command header)
- Write `{0x00002000}` (parameter 1: ram address `0x2000` bytes)
- Write `{0x01200000}` (parameter 2: bank `01`, length `0x200000` bytes)
- Receive `0x200000` bytes of data
- Receive `{0x434D5030}` completion packet

COMMAND DESCRIPTIONS

LOAD FROM PC

	Description	Bytes
Data to send	Bank offset in bytes	4
	Bank index	1
	Length in bytes (32bit aligned)	3
	Desired data	< 8 Mbyte
Data to receive	-	-

Loads the contents of a specified bank with data sent by PC. Can be used for transferring image to Cart ROM, loading a save game to emulated EEPROM, and so on.

The bank index can be any of the following:

Index	Description	Size
0	Invalid	-
1	Cartridge ROM (actually SDRAM)	64 Mbyte (240 Mbyte HW2)
2	SRAM (256 kbit)	32 Kbyte
3	SRAM (768 kbit)	96 Kbyte
4	FlashRAM	128 Kbyte
5	FlashRAM (Pokemon Stadium 2 on HW1)	128 Kbyte
6	EEPROM (4k, 16k)	512 byte, 2 Kbyte

DUMP TO PC

	Description	Bytes
Data to send	Bank offset in bytes	4
	Bank index	1
	Length in bytes (32bit aligned)	3
Data to receive	Desired data	< 8 Mbyte

Reads the contents of any bank. Use the same bank index list as the previous command.

TARGET-SIDE FIFO

	Description	Bytes
Data to send	Chunk type	1
	Length in bytes (32bit aligned)	3
	Desired data	< 8 Mbyte
Data to receive	-	-

Sends the desired data to be received by an armed Target-side buffer on N64. Please see the Target-side interface section for additional context.

SET SAVE TYPE

	Description	Bytes
Data to send	Save type	4
Data to receive	-	-

Sets the save type to be emulated. Only bits [3:0] of the 32bit word are used.
Refer to the CI Set Save Type command for the list of values.

SET CIC TYPE

	Description	Bytes
Data to send	CIC word	4
Data to receive	-	-

Requires HW2/RevB
Sets the CIC type to be emulated as soon as the N64 power is turned on. Only affects images booted from USB mode. This word is decoded as follows:

Bit	Description
[31]	Override default type, must be 1
[3:0]	CIC Index
	0 - 6101
	1 - 6102
	2 - 7101
	3 - 7102
	4 – 6103/7103
	5 – 6105/7105
	6 – 6106/7106
	7 - 5101

Regardless of the selected CIC, the region is still handled automatically by the cartridge to match your N64. Of course, the desired CIC type must match the image's header you load for it to properly boot. By loading your image, setting the proper save type, and setting the CIC type, you can create an electrically identical cartridge that can even be booted with a real Gameshark.

SET CI EXTENDED ADDRESSING

	Description	Bytes
Data to send	Enable?	4
Data to receive	-	-

Requires HW2/RevB

Enable[0] sets whether CI registers should be relocated higher into PI space to make room for ROM data.

This works the same way as the previous CI Enable/Disable CI Extended Address Mode commands.

While it's possible to upload data exceeding the 64mbyte limit into the device over USB, the N64 will be unable to access it unless you enable extended addressing.

Please note that if the software uses the CI registers it must adjust to the higher register locations.

VERSION REQUEST

	Description	Bytes
Data to send	-	-
Data to receive	Version word	4

Returns the device firmware and variant.

Bit	Description
[31:16]	Device variant, e.g. 0x4200 for RevB
[15:0]	Device firmware in decimal For example, 2.05 is 0x00CD

UPGRADE START

	Description	Bytes
Data to send	-	-
Data to receive	-	-

Starts the upgrade engine on the device. Firmware upgrade blob must be uploaded to Cartridge ROM at 0x0 prior.

UPGRADE REPORT

	Description	Bytes
Data to send	-	-
Data to receive	Upgrade status	4

Returns the current status of the Upgrade Module. Please see the CI register Upgrade Module Status for the table of values.

STANDALONE ENTER

	Description	Bytes
Data to send	-	-
Data to receive	-	-

Causes the device to enter Standalone Mode. This is a special mode where bus emulation is disabled, and instead the 64drive actually drives the PI and SI busses as if it were a real N64.
Requires the UltraSave adapter to connect with either a 64dd or real cartridge.

STANDALONE LEAVE

	Description	Bytes
Data to send	-	-
Data to receive	-	-

Leaves Standalone Mode and returns to normal operation.

STANDALONE PI READ 32

	Description	Bytes
Data to send	PI Address	4
Data to receive	Desired bus data	4

Executes a 32-bit read from the desired address and sends back the data.
For example, to dump the first 32 bits of a cartridge using the UltraSave adapter, read from the address `0x10000000` and the returned data should be `0x80371240`.
In case there is no device attached or the address is unmapped, the return data will be open-bus, where the data will simply be the last 16 bits of the address repeated twice.

STANDALONE PI WRITE 32

	Description	Bytes
Data to send	PI Address	4
	Desired write data	4
Data to receive	-	-

Executes a 32-bit write at the desired address.
For example, to overwrite a value in a cartridge that has SRAM attached, write to the address `0x08000000` with any data. Of course, any existing data will be corrupted.

STANDALONE PI READ BURST

	Description	Bytes
Data to send	PI Address	4
	Burst length	4
Data to receive	Desired bus data	Any

Executes a burst read of the specified length. The acceptable burst length depends on the physical hardware connected and mapped at the location read. For example, normal cartridge mask ROMs only support 512 byte burst length, while SRAM support longer lengths.

STANDALONE PI WRITE BURST

	Description	Bytes
Data to send	PI Address	4
	Burst length	4
	Desired data	Burst length
Data to receive	-	-

Executes a burst write of the specified length.
Please note that this command does NOT respond with a completion packet.

STANDALONE PI WRITE 16

	Description	Bytes
Data to send	PI Address	4
	Desired data (only upper halfword used)	4
Data to receive	-	-

Writes a halfword to the desired address. Not for use with official N64 hardware. Could be used for special hardware like UFLC carts, official flash carts, and Gamesharks, all of which utilize CFI/JEDEC standard 16bit bus ops. Returns as soon as the word is written to bus.
Please note that this command does NOT respond with a completion packet.

STANDALONE PI WRITE 16 WITH DQ7 EXIT

	Description	Bytes
Data to send	PI Address	4
	Desired data (only upper halfword used)	4
Data to receive	-	-

Writes a halfword to the desired address. Not for use with official N64 hardware. Could be used for special hardware like UFLC carts, official flash carts, and Gamesharks, all of which utilize CFI/JEDEC standard 16bit bus ops.
As per CFI standard, internally polls DQ7 until the data matches the word written, which signifies write completion. Useful when burning a flash chip.
Please note that this command does NOT respond with a completion packet.

STANDALONE SI OPERATION

	Description	Bytes
Data to send	Operation bit write length	1
	Operation bit read length	1
	Operation write phase [79:64]	2
	Operation write phase [63:32]	4
	Operation write phase [31:0]	4
Data to receive	Operation read phase [63:32]	4
	Operation read phase [31:0]	4

Executes an SI operation. Can be used for interacting with all SI devices such as controllers, EEPROMs, and the RTC chip in Doubutsu no Mori.
Bits are written onto the SI data line, shifting out of the write_phase[79] shift register, which rotates left.

Bits are then read from the SI data line, shifting into the read_phase[0] shift register, which rotates left.
For example, to probe if an EEPROM is attached, write 8 bits and read back 24 bits of status:

- Write {0x434D4498} (command header)
- Write {0x08182000} (0x08 write, 0x18 read, null data)
- Write {0x00000000} (null data)
- Write {0x00000000} (null data)
- Receive 8 bytes of data: {0x00000000 0x00008000} which signifies 64 pages
- Receive {0x434D5098} completion packet

WI-FI INTERFACE

OVERVIEW

Wireless 802.11b/g/n is available on HW2/RevB units in the form of an ESP8266 system-on-chip module, referred to as the ESP module for simplicity.

Implementation is very similar to the USB Target-side interface, though physically the interface is a serial UART that runs much slower than USB. Additionally, the interface has more commands to support toggling various general-purpose I/O on the ESP module, firmware upgrades, adjusting baud rate, and rebooting the module.

Further information will be released as the interface is more fully developed.

END OF DOCUMENT

REVISION INFORMATION

November 4, 2014	First internal draft	
January 7, 2015	First public release	Initial release with FW 2.00
May 3, 2017	Second public release	Added information about HW2
December 29, 2017	Third public release	Added USB protocol details
February 18, 2018	Fourth public release	Fixed USB command packet format
December 10, 2018	Fifth public release	Added USB command for CI extended addressing

COPYRIGHT INFORMATION

To submit any inaccuracies or inconsistencies in this document, please e-mail support at the above web address.