The Definitive DI / DVD Interface Thread
https://www.gc-forever.com/forums/viewtopic.php?f=26&t=3362

The Definitive DI / DVD Interface Thread

by **hornpipe2** Posted: **Sun Mar 20, 2016 11:04 pm**

This is a massive compilation thread for people interested in doing DVD drive replacement efforts. Unknown or unverified information will be listed in *red italics*.

The GC motherboard interfaces with the DVD drive assembly through the P9 connector on the motherboard. YAGCD has this info on the pinout.
DVDConnector-1.jpg
(27.16 KiB) Not downloaded yet
Gamecube_dvd_connector-3.jpg
(21.12 KiB) Not downloaded yet

pin	Signal
1	AISLR (audio bus)
2	5V
3	AISD (audio bus)
4	5V
5	AISCLK (audio bus)
6	5V
7	DIHSTRB
8	5V
9	DIERRB
10	Ground
11	DIBRK
12	DICOVER
13	DIDSTRB
14	DIRSTB
15	DIDIR
16	Ground
17	DID7
18	Ground
19	DID6
20	Ground
21	DID5
22	Ground
23	DID4
24	Ground
25	DID3
26	Ground
27	DID2
28	MONI
29	DID1
30	MONOUT
31	DID0
32	Ground

All signals lines are **3.3V**. It may help to group the pins by function. Pins in red are host-controlled (driven by GC motherboard), in blue are DVD-controlled, and in green are bidirectional. Black is for non-signal wires.

Pins 11, 28 and 30 need direction / definition

Pin: Signal
--- Streaming Audio ---
1: AISLR ("Audio: Stereo L/R")
3: AISD ("Audio: Data")
5: AISCLK ("Audio: Clock")

--- Data Communication ---
7: DIHSTRB ("Host Strobe")
13: DIDSTRB ("Device Strobe")
15: DIDIR ("Direction Line": indicates who drives the Data Bus)
17, 19, 21, 23, 25, 27, 29, 31: DID0-7 (8-bit bidirectional Data Bus)

--- Interrupt ---
14: DIRSTB ("Reset": Cube may instruct the DVD drive to "reset")
9: DIERRB ("Error State": indicates drive error to mainboard)
12: DICOVER ("Door/Cover State": indicates door/cover is open, to mainboard)
11 DIBRK

--- Other ---
28 MONI
30 MONOUT

--- Power/Ground ---
2, 4, 6, 8: +5V
10, 16, 18, 20, 22, 24, 26, 32: Ground

Ok, that covers the electrical connections... On to PROTOCOL.

On power-up the Gamecube will *raise DRSTB, which starts the DVD processor working*. The DVD drive should be reading from DID0-7 in Input mode, and it should also monitor DIHSTRB and DIDIR. (Any time the GC raises DIDIR, it wants the DVD to run the data bus. Any time it drops to Low, the DVD should stop what it's sending, and flip from output to input.)

When the Cube wants to talk to the Drive, it will send a 12-byte transaction, sometime after raising DIHSTRB first. Timing looks something like this:
GC_TIMING.gif
(4.22 KiB) Not downloaded yet
STRB lines are latched on a low->high transition, so as you can see in this chart twelve strobes occur. *The drive is safe to raise DIDSTRB after receiving 9 bytes;* this indicates to the Cube that the DVD drive wants to talk. Wait for the cube to lift DIDIR ("DVD drives the bus") and drop DIHSTRB ("Cube not requesting write now"), and then the DVD drive may respond.

DVD to GC communication works the same way, but the signals are reversed: DVD drive will strobe DIDSTRB and clock data out on the bidi bus. Drop DIDSTRB if you are out of data to send, or if the Cube raises DIHSTRB or drops DIDIR.

Gamecube will speak to DVD drive at a rate of ~20 MHZ (DIHSTRB rate). The DVD drive typically responds at 13.5 MHZ. *Different (slower) rates are OK. The system becomes unstable at ??.??? Mhz, because the DVD drive is sending data too fast for the GC FIFO and it overflows. There is no way to check for this condition.*

Those signaling details aside, here are the messages the DVD drive might get from the Gamecube: http://www.gc-forever.com/wiki/index.ph ... DICommands

Re: The Definitive DI / DVD Interface Thread

by **hornpipe2** Posted: **Sun Mar 20, 2016 11:05 pm**

Audio Streaming
... deserves its own section. The DVD drive has a built-in ability to decode certain on-disk regions of ADPCM sound. These are sent back on special pins which route to the sound processor. Thus, the CPU is freed from having to decode audio itself: it merely issues start/stop/queue commands and lets the drive do all the heavy lifting.
In order for this to work properly, a few things are needed:
* DVD drive has to be able to decode the ADPCM sound and send it back on the Audio pins (1, 3) at the rate requested by the Cube (pin 5).
* Audio data must be aligned on 32kb sectors on disk, because the seek command is only this granular. Get the alignment wrong and only static will play.

Some timing diagrams for audio streaming. The output is Cube-controlled, and could be 32khz, 48khz *or slightly faster (48043hz)*. In any case it's 16-bit PCM data, stereo, so the overall AISCLK rate is (rate * 16 * 2)hz. You can see AISLR flipping back and forth every 16 bits, indicating left- or right-speaker data to be sent. MSB is first, LSB last.
AI_CLK2.gif
(9.58 KiB) Not downloaded yet
AI_CLK1.gif
(11.13 KiB) Not downloaded yet
Unseen (below) notes that this scheme is essentially **host-controlled I2S**, where the receiver (cube) runs AISCLK at the rate it wants, and swaps AISLR every 16 bits to indicate whether DVD should send L or R stereo. The drive returns AISD based on the speed the Cube sends. You can read more about it here: https://en.wikipedia.org/wiki/I%C2%B2S

Since DVD emulators would be doing the audio decoding yourself, you'll need the code to do it...
I don't have any easy answers (yet), but here's a Winamp plugin that claims to do it. And several other video game systems too. So good luck isolating the GC parts 😜
https://gitlab.kode54.net/kode54/vgmstream/

You might also get useful info from StreamADPCM::Decode from the Dolphin emu.
https://github.com/dolphin-emu/dolphin/ ... mADPCM.cpp

Commands, taken from... somewhere:

> Upon Power up or reset , 2 commands must be issued for proper use of audio streaming:
>
> DVDReadDiskID A8000040,00000000,00000020
> DVDLowAudioBufferConfig E4xx00yy,00000000,00000020
>
> xx=byte 8 [0 or 1] from the disk header retrieved from DVDReadDiskID
> yy=0 (if xx=0) or 0xA (if xx=1)
>
> All immediate commands [Ex] respond with 4 bytes.
>
> DVDLowRequestAudioStatus E2xx0000
> xx=0 is stream playing?
> response: 0,0,0,y
> y=1 yes;0=no
>
> xx=1 what is the current address
> response: y,y,y,y
> y=address of current audio buffer that is burning on disk with 0x2000 granularity
> remember address is actaul byte offset>>2
> i.e. it has a granularity of 0x8000 bytes which is 0x2000
>
> xx=2 what was the start address of currently playing audio stream
> response: starting address of current audio stream >>2

xx=3 what was the length of currently playing audio stream
response: length of current audio stream

DVDCancelStreamAsync E1010000,xxxxxxxx,yyyyyyyy
response: 0,0,0,0
cancel current audio stream

DVDStopStreamAtEndAsync E1000000,xxxxxxxx,yyyyyyyy
response: 0,0,0,0
start audio stream at x with length y
if x and y are zero then will stop audio stream at end of buffer.
if not zero then will add another stream to queue, when current one finishes
then next one will be played.
Queue is only 1 level deep, and is replaced by last command issued.

Some other commands:
DVDLowInquiry 12000000,00000000,00000020
response: 32 bytes with what appears to Date/Version info
00,00,00,00
20,02,04,02 Date code
61,00,00,00 Version
rest are zero

DVDLowSeek AB000000
seek to offset, reponse 4 bytes

DVDLowStopMotor E3000000
stop motor

To properly handle Door open and DIERRB:
When door is opened and command is issued [sometimes software doesnt check status of door open]
assert DIERRB until E0000000 command is issued.
Deassert DIERRB and send 4 byte response:

Error Responses:
high byte:
00 Ready.
01 Cover is opened.
02 Disk change.
03 No Disk.
04 Motor stop.
05 Disk ID not read.

low bytes:
000000 No error.
030200 No Seek complete.
031100 UnRecoverd read error.
052000 Invalid command operation code.
052001 Audio Buffer not set.
052100 Logical block address out of range.
052400 Invalid Field in command packet.
052401 Invalid audio command.
052402 Configuration out of permitted period.
062800 Medium may have changed.
023A00 Medium not present / Cover opened.
0B5A01 Operator medium removal request.

056300 End of user area encountered on this track.
020401 Disk ID not read.
020400 Motor stopped.
040800 Transfer protocol error.

Re: The Definitive DI / DVD Interface Thread

by **hornpipe2** Posted: **Sun Mar 20, 2016 11:28 pm**

Some useful links and references for this post:
* This awesome forum post: viewtopic.php?f=26&t=3234
* DI Commands on the Wiki: http://www.gc-forever.com/wiki/index.ph ... DICommands
* HyperIris did one too: viewtopic.php?t=184

Also, someone has compiled a ZIP file of info from the CrazyNation IDE interface - perhaps the first proof-of-concept for the drive emulator. You can get it from my Dropbox here.
https://dl.dropboxusercontent.com/u/564 ... VD_Emu.zip

Re: The Definitive DI / DVD Interface Thread

by **Unseen** Posted: **Mon Mar 21, 2016 10:39 am**

Thanks for compiling that information!

There are some minor details in the audio part that are currently not clear to me though:

The AI signals look as if it's just I2S - an I2S audio source can be either self-clocked or clocked by the receiver. Is AISCLK driven by the drive or the Gamecube? AISLR should be driven by the drive in either case.

Is the audio sample rate really 48000 Hz or is it the slightly higher rate of the Cube, 48042.xxx Hz?

Re: The Definitive DI / DVD Interface Thread

by **tueidj** Posted: **Mon Mar 21, 2016 11:54 am**

The clock can be 32KHz rather than 48KHz (although no commercial games use it), it's definitely controlled by the cube.

Re: The Definitive DI / DVD Interface Thread

by **meneerbeer** Posted: **Mon Mar 21, 2016 1:57 pm**

I think Crazynation wrote some C code for the audio decompression. I will see if I can find it and upload it.

I still want to try to get my emulator working.. Discs are recognized. They just seem to fail when booting.

Re: The Definitive DI / DVD Interface Thread

by **hornpipe2** Posted: **Mon Mar 21, 2016 6:16 pm**

Thanks for the tips, I've edited the above to note some changes. Keep them coming :)

I'm also planning to start one of these (based around some fast ~100mhz ARM microcontroller and an SD card) so I'll add my own notes as I discover issues.

Re: The Definitive DI / DVD Interface Thread

by **hornpipe2** Posted: **Mon Mar 21, 2016 6:52 pm**

> ***Unseen wrote:***

> AISLR should be driven by the drive in either case.

Actually... AISLR may be Cube controlled too. Check this VHDL for the CrazyNation version?

```
--GC
DID: inout std_logic_vector(7 downto 0);
DIHSTRB : in std_logic;
DIDIR : in std_logic;
DIDSTRB : out std_logic;
DIERRB : out std_logic;
DIBRK : inout std_logic:='Z';
DICOVER : out std_logic;
DIRSTB : in std_logic;

AISCLK : in std_logic;
AISD : out std_logic;
AISLR : in std_logic;
```

Re: The Definitive DI / DVD Interface Thread

by **meneerbeer** Posted: **Mon Mar 21, 2016 7:03 pm**

> ***hornpipe2 wrote:***
> I'm also planning to start one of these (based around some fast ~100mhz ARM microcontroller and an SD card) so I'll add my own notes as I discover issues.

Interesting. I have been thinking about that as well, but I am much more a hardware designer, so I usually settle with an FPGA.  What kind of chip is on the DVD drive anyways? Also just a standard microprocessor?

I really need to make some time to work on mine again. It would be nice if it would at least work without audio streaming.

The ADPCM decode code from crazynation can be found below. I believe he would send the decoded data over PC to the FPGA.

```
// GC DVD ADPCM decoding algorithm
// NOTE: the ADPCM format used by the dvd drive is *NOT* the same as the ADPCM forma
// they are similiar but the dvd adpcm format requires less multiplies
// previous public decoder by Shinji Chiba is innacurate
// -Crazynation

#include <stdio.h>

int predL1,predL2,predR1,predR2;
int new_pred;

init_predictors()
{
        predL1=0;
        predL2=0;
        predR1=0;
        predR2=0;
}
int decode(int predictor1,int predictor2,int nibble,int index)
{
        int i,j,k;
        short  scalex,scaley;
```

```
	int data1,data2;


	switch(index>>4)
	{
		case 0:
			scalex=0;
			scaley=0;
			break;
		case 1:
			scalex=0x3C;
			scaley=0;
			break;
		case 2:
			scalex=0x73;
			scaley=0xFFCC;
			break;
		case 3:
			scalex=0x62;
			scaley=0xFFC9;
			break;
		default:
			break;
	}
	i=predictor1*scalex;
	j=predictor2*scaley;
	i=i+j+0x20;
	i=i>>6;
	if(i<(int)0xFFE00000)
		i=0xFFE00000;
	else if(i>0x1FFFFF)
		i=0x1FFFFF;

	data1=i;

	j=(nibble&0xF)<<12;
	j=(short)j>>(index&0xF);

	i=j<<6;
	i=i+data1;

	data2=i;
	new_pred=data2;

	j=data2>>6;
	if(j<(int)0xFFFF8000)
		j=0x8000;
	else if(j>0x7FFF)
		j=0x7FFF;

	return j;




}
char wav_header[]={0x52,0x49,0x46,0x46,0x26,0x40,0x21,0x00,0x57,0x41,0x56,0x45,0x66,
0x10,0x00,0x00,0x00,0x01,0x00,0x02,0x00,0x80,0xBB,0x00,0x00,0x00,0xEE,0x02,0x00,
0x04,0x00,0x10,0x00,0x64,0x61,0x74,0x61,0x00,0x40,0x21,0x00};
```

```
int main(int argc, char *argv[])
{
        int i,j,k;
        int count,length,offset;
        int nibble_high,nibble_low;
        FILE *fin,*fout;
        unsigned char buffer[32];
        short wav[128];


        if(argc<3)
        {
                printf("decode <adpcm in file> <output wav file>\n");
                exit(1);
        }
        fin=fopen(argv[1],"rb");
        if(fin==0)
        {
                printf("cant open input file %s\n",argv[1]);
                exit(1);
        }

        fout=fopen(argv[2],"wb");
        if(fout==0)
        {
                printf("cant open output file %s\n",argv[2]);
                exit(1);
        }
        fseek(fin,0,SEEK_END);
        length=ftell(fin);
        fseek(fin,0,SEEK_SET);

        init_predictors();
        offset=0;
        fwrite(wav_header,44,1,fout);
//------------------------------------------------------------------------------
decode:
        i=fread(buffer,32,1,fin);
        if(i==0)
                exit(1);

        if(buffer[0]!=buffer[2])
        {
                printf("invalid ADPCM file\n");
                exit(1);
        }

        if(buffer[1]!=buffer[3])
        {
                printf("invalid ADPCM file\n");
                exit(1);
        }
        count=0;
        do{

                nibble_high=buffer[count+4]>>4;
                nibble_low=buffer[count+4]&0xF;

                k=decode(predL1,predL2,nibble_low,buffer[0]);
```

```
                wav[count*2]=k;

                predL2=predL1;
                predL1=new_pred;

                k=decode(predR1,predR2,nibble_high,buffer[1]);
                wav[(count*2)+1]=k;

                predR2=predR1;
                predR1=new_pred;

        count++;
        }while(count<28);

        fwrite(wav,28*4,1,fout);

        offset+=32;
        if(offset<length)
                goto decode;

        length=ftell(fout);
        fseek(fout,4,SEEK_SET);
        i=length-0x6;
        fwrite(&i,4,1,fout);
        fseek(fout,0x28,SEEK_SET);
        i=length-0x2C;
        fwrite(&i,4,1,fout);




}
```

Re: The Definitive DI / DVD Interface Thread

by **Streetwalker** Posted: **Mon Mar 21, 2016 7:33 pm**

> ***meneerbeer wrote:***
> What kind of chip is on the DVD drive anyways? Also just a standard microprocessor?

Some 8 bit MCU (MN10200).

Re: The Definitive DI / DVD Interface Thread

by **Unseen** Posted: **Mon Mar 21, 2016 7:41 pm**

> ***hornpipe2 wrote:***
>
> > ***Unseen wrote:***
> > AISLR should be driven by the drive in either case.
>
> Actually... AISLR may be Cube controlled too. Check this VHDL for the CrazyNation version?

Indeed, I misremembered that - when an I2S receiver is the master device, the only signal driven by the sender is the data line.

Re: The Definitive DI / DVD Interface Thread

by **pyroholic** Posted: **Wed Mar 23, 2016 6:26 pm**

> ***Unseen wrote:***
>
> > ***hornpipe2 wrote:***
> >
> > > ***Unseen wrote:***
> > > AISLR should be driven by the drive in either case.
> >
> > Actually... AISLR may be Cube controlled too. Check this VHDL for the CrazyNation version?
>
> Indeed, I misremembered that - when an I2S receiver is the master device, the only signal driven by the sender is the data line.

Yes that is correct, the cube drives clock and l/r channel strobe. It toggles every 16-bits so a period is 32-bits which is equat to a clock frequency of ~1.54 MHz at 48kHz and 1.02MHz at 32kHz. I used the audio-streaming code from dolphin when i wrote my VHDL core for streaming audio and it seems correct, sounds decent at least. The dolphin decoder is the same as for the winamp plugin.

As for cube/dvd speed on the bus, from my findings using a logic analyzer the cube seems to be transmitting at 20Mbyte/s where the strobe signal toggles as a 20MHz clock. The drive seems to be shuffling data at 13.5MHz, which was calculated by measuring the time of ~100 clocks divided by the same number of clock flanks.

The one signal i have not been able to figure out is DIBRK, which is listed as bidirectional in crazynations code but not handled. I have mapped it as IO in my implementation set to high impedance, and has never seen it toggle from 1 to 0. I have not seen it toggle while sniffing on the original drive connected to the cube either. But my guess is that it is handled using some kind of handshake if it should really be bidirectional.

Re: The Definitive DI / DVD Interface Thread

by **hornpipe2** Posted: **Tue Mar 29, 2016 6:49 pm**

I assumed DIBRK was marked as bidirectional just because they didn't know what to do with it either :) Maybe the Cube raises it if you send data too fast.

Any idea on MONI or MONOUT? Some kind of serial debug thing or something, I guess.

Have you tried going faster than 13.5MHz to Cube?

Re: The Definitive DI / DVD Interface Thread

by **pyroholic** Posted: **Tue Mar 29, 2016 7:13 pm**

Currently i use a system clock of 125 MHz divided by six to generate the strobe signal, so it toggles every third clock resulting in 20.83 MHz and so far no issues.

I measure zero ohms resistance between moni / monout pins and grund on the drive board so i guess they are really ground pins.

Re: The Definitive DI / DVD Interface Thread

by **tueidj** Posted: **Wed Mar 30, 2016 2:34 am**

BRK is signalled to the drive on error I.e. when it takes too long to respond. It's meant to cancel the current operation.

Re: The Definitive DI / DVD Interface Thread

by **pyroholic** Posted: **Wed Mar 30, 2016 6:54 am**

Ok, I guess that might be why I've never seen it toggle. Perhaps it is asserted before a dre happens.

My guess is that the cube asserts the break signal for a short period and then releases it, the dvd

responds by holding the break signal on the asserted level until the break is complete. It should be possible to verify with some custom code that starts a break request while observing with a logic analyzer or scope.

Re: The Definitive DI / DVD Interface Thread

by **hornpipe2** Posted: **Wed Mar 30, 2016 2:35 pm**

> ***meneerbeer wrote:***
>
> > ***hornpipe2 wrote:***
> > I'm also planning to start one of these (based around some fast ~100mhz ARM microcontroller and an SD card) so I'll add my own notes as I discover issues.
>
> Interesting. I have been thinking about that as well, but I am much more a hardware designer, so I usually settle with an FPGA. What kind of chip is on the DVD drive anyways? Also just a standard microprocessor?

After some research, I think ARM-based uC is not the answer. 20MHz bitbanged parallel access is just too fast for most microcontrollers to keep up with. For industry protocols like SPI or I2C there is often on-chip hardware to make it go fast, but for a custom protocol you are limited to how fast you can toggle GPIO pins. Even with a 700MHz Raspberry Pi, it craps out at 22MHz using C.

Now to see how much VHDL I remember from one semester of college :)

The chips currently on the drive, according to YAGDC, are:

```
Matsushita MN103S13BGA Optical Disk Controller
Matsushita MN102H60GFA MicroComputer clocked at 34mhz
```

with some custom ROM. MN103 probably runs the motor and laser etc, while MN102 does the computation.

Re: The Definitive DI / DVD Interface Thread

by **pyroholic** Posted: **Wed Mar 30, 2016 3:55 pm**

You could maybe get away with the writes from the drive to the cube using a bit-bang approach but most likely slow. Reading the commands transferred by the cube would most likely be very difficult. It should be possible to make a solution with a cpld or fpga to handle communication but you would still want a fast way to shuffle data to the cube. An FPGA with a softcore and sd-card for storing games should work. If you want to do audio streaming you will probably need to be able to buffer a bit some amount data so then some external RAM would be good.

Re: The Definitive DI / DVD Interface Thread

by **nobodylemons** Posted: **Fri Feb 09, 2018 6:26 pm**

Now that this thread has been dead for 3 years, the raspberry pis have gotten quite a bit faster. The pi 3's clock speed is now 1.2GHz instead of 700MGhz. Is this fast enough to act as a drive replacement for the Gamecube using GPIO pins?

Re: The Definitive DI / DVD Interface Thread

by **Streetwalker** Posted: **Fri Feb 09, 2018 9:22 pm**

No. CPU speed has little to do with this, most of the SoC is still pretty much the same, especially the IO hardware which is what we're interested in. Don't expect it to ever be viable. FPGAs are cheap anyway.