GCpad IP Core Integration Manual

Author: Arnim Läuger
arniml@opencores.org

Rev. 1.0
October 13, 2004

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
1.0	13-Oct-2004	Arnim Läuger	First release.

Contents

1

Introduction

The GCpad core interfaces to the gamepad used with the Nintendo Gamecube video gaming system. The core communicates to the gamepad using its proprietary communication protocol and offers the retrieved information for further processing.

To suit the needs of the integrating system, two different flavours of the core are available:

- For simple applications, the basic flavour manages all communication issues with the gamepad and provides the current status of the buttons and analog axes at its interface. The integrating system does not need to interfere with gamepad communication and can statically read the button and axes status information.
- The full flavour allows full control of the gamepad communication to the integrating system. This flavour offers a command and response interface which is driven by the system to send arbitrary commands to the gamepad. The response of the gamepad is available for further processing.

This document indents to aid the integration of both flavours of the core.

The GCpad core is part of the gamepads controller core collection. The complete collection is maintained and released on the OpenCores web server. You can access the gamepads project at

http://www.opencores.org/projects.cgi/web/gamepads/overview

Updates of the GCpad core can be optained via the project pages.

2 Architecture

The following Figure 1 gives an overview of the core's architecture.

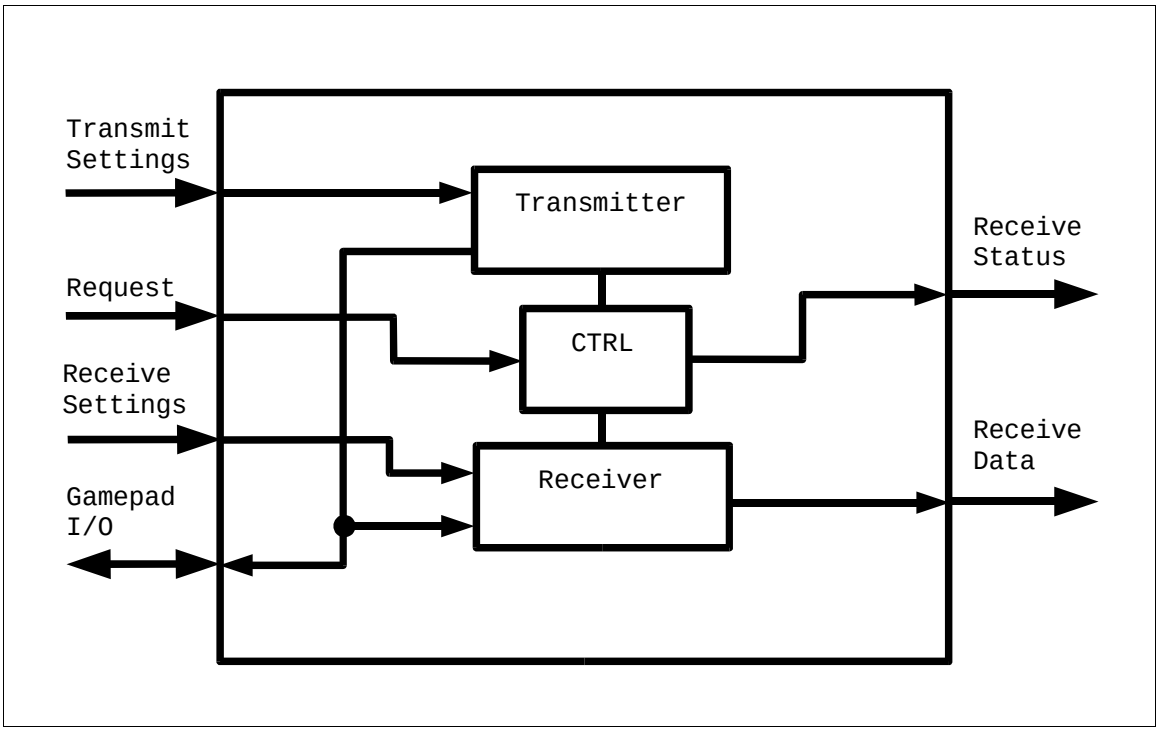

Figure 1: Architecture of the GCpad core

The central element is the control until `CTRL`. It manages both the `Transmitter` and `Receiver` unit and controls one atomic command/response sequence. The `CTRL` unit is triggered by the request line and instructs the `Transmitter` to issue the appied command. When the command has been transmitted, the `Receiver` is actively monitoring the I/O line for any incoming response data from the gamepad.

The received data is presented as a 64 bit value at the core's interface. In addition, the receive status is output constantly, indicating whether the command/response sequence has finished and signalling a timeout while reading the response.

The transmit settings comprise the command value and the length of the command. The receive settings specify the expected number of bytes to be received from the gamepad.

3

Operation

This section describes the functionality of the GCpad core. Beginning with the description of a command/response sequence, the details of communicating with the Gamecube controller pad are given.

Command and Response Sequence

Whenever the core is triggered by the `pad_request_i` line it will initiate a command and response sequence. Such a sequence comprises the transmission of the specified command and reception of the gamepad's response.

Command Transmission

The command is specified by the `tx_command_i` input bus. The value on this 24 bit wide bus is transmitted MSB first to the gamepad. The value on `tx_size_i` specifies how many bytes have to be transmitted. It is a binary coded, 2 bit vector ranging from 1 to 3.

When the transmit size is set to 3, the full 24 bits (3 bytes) of `tx_command_i` will be transmitted to the gamepad. For sizes 2 and 1, only the first 16 resp. 8 bits of `tx_command_i` are transmitted. The first bit which is sent is always the MSB (number 23).

The communication protocol requires an additional stopbit following the command payload. Therefore, the first unused bit of `tx_command_i` must have the value '1'. This is bit number 15 for transmit size 1 and bit number 7 for transmit size 2. The Transmitter automatically adds the '1' bit for transmit size 3.

Response Reception

After transmitting the command, the GCpad core switches into reception mode and monitors the I/O line to the gamepad. Each incoming bit is sampled by a majority detector inside the Receiver and added to an internal shift register. When the expected number of bytes has received as specified by `rx_size_i`, the response data is output to `rx_data_o` and the response phase terminates.

Like `tx_size_i`, `rx_size_i` is a binary encoded vector which allows to set the expected number of received bytes to the range 1 to 8. The upper limit is 8 although a larger value can be applied. The behaviour of the Receiver is undefined for values larger than 8.

The incoming data is presented at the 64 bit wide vector `rx_data_o`. Data is shifted in MSB first, so the first received bit is located at bit position 63 when 8 bytes are received. The following bits are loacted at positions 62 down to 0. For less than 8 bytes, the first bit is located at a lower position, always filling the output vector from n down to 0. Index n is calculated as follows

$$n = 2^{\text{rx\_data\_i}} - 1$$

Position n contains the MSB of the response, i.e. the bit that was received first.

Response reception is aborted when no response from the gamepad is detected or when a timeout occurs.

Receive Status

There are two output signals that show the status of the command and response sequence: `pad_avail_o` and `pad_timeout_o`

Whenever a sequence has been triggered via `pad_request_i`, both outputs are cleared to '0'. As soon as the Receiver has terminated, `pad_avail_o` is set to '1' and keeps this level until a new sequence is initiated. In case a timeout occured or no data from the gamepad was received at all, `pad_timeout_o` is activated together with `pad_avail_o`.

The system must neither change `tx_data_i`, `tx_size_i` or `rx_size_i` nor trigger a new sequence via `pad_request_i` until `pad_avail_o` has changed from '0' to '1'. A valid response is only visible at `rx_data_o` if pad_timeout_o is at '0' while `pad_avail_o` is '1'.

Basic Core Flavour

All said above applies to the full flavour of the GCpad core (entity `gcpad_full`). The basic flavour (entity `gcpad_basic`) differs from this in the way that command code, size and response size are fixed to single values. Furthermore, the response data is already decoded by terms of button and axes status.

The command/response sequence cycles automatically and generates a quasi-static output at the button and axes signals. This eases integration of the core in systems that do not contain facilities that can manage the core, trading off functionality.

4

Parametrization

The GCpad core can be parametrized by VHDL generics in several ways. Table 1 specifies the parameters and their respective meaning.

Generic Name	Value	Description
reset_level_g	0 or 1	Active level of the asynchronous reset input reset_i 0 ... low active 1 ... high active
clocks_per_1us_g	2 ... n	Number of clk_i periods within 1 µs

Table 1: List of generic parameters for the GCpad core

The generic `reset_level_g` is straight forward. It gives the integrator the possibility to specify the active level of `reset_i`. Nonetheless, `reset_i` will always have asynchronous characteristic. This cannot be changed.

The generic `clocks_per_1us_g` is necessary to adjust the counters inside the Transmitter and Receiver unit to the clock frequency of `clk_i`. These counters operate on multiples of this parameter instead of a "real" timebase, the value of this parameter is important. It should match quite exactly the final frequency of `clk_i` in the target design as a inaccuracy will lead sooner or later to failures in the communcation with the gamepad.

5 Clocks

Table 2 specifies the clocks of the GCpad core.

Name	Source	Rates (MHz)		Remarks	Description
		Max	Min		
clk_i	System Clock	N/A	2	Minimum frequency required for gamepad communication	System clock for synchronous operations.

Table 2: List of clocks

The system clock `clk_i` requires a minimum frequency of 2 MHz. This is necessary to support the time base of 1 µs for communication with the gamepad. To sample the incoming data stream corretly, the Nyquist criteria has to be fulfilled. The maximum frequency that has to be detected is 1 MHz so the Receiver needs at least 2 MHz.

6

IO Ports

Table 3 specifies the ports of the GCpad core in full flavour.

Name	Width	Direction	Description
clk_i	1	Input	System clock
reset_i	1	Input	Asynchronous system reset
pad_request_i	1	Input	Request to execute a command/response sequence
pad_avail_o	1	Output	Response available indication
pad_timeout_o	1	Output	Timeout indication
tx_size_i	2	Input	Number of bytes for transmission
tx_command_i	24	Input	Command vector for transmission (MSB transmitted first)
rx_size_i	4	Input	Number of expected bytes for response
rx_data_o	64	Output	Response data
pad_data_io	1	Bidir	Signal line to the gamepad

Table 3: List of IO ports for the full flavour

Table 4 specifies the ports of the GCpad core in basic flavour.

Name	Width	Direction	Description
clk_i	1	Input	System clock
reset_i	1	Input	Asynchronous system reset
pad_request_i	1	Input	Request to execute a command/response sequence
pad_avail_o	1	Output	Response available indication
pad_data_io	1	Bidir	Signal line to the gamepad
but_a_o	1	Output	State of button A
but_b_o	1	Output	State of button B
but_x_o	1	Output	State of button X
but_y_o	1	Output	State of button Y
but_start_o	1	Output	State of button Start
but_tl_o	1	Output	State of shoulder button TL (digital contact)
but_tr_o	1	Output	State of shoulder button TR (digital contact)
but_left_o	1	Output	State of digital pad Left
but_right_o	1	Output	State of digital pad Right

Name	Width	Direction	Description
but_up_o	1	Output	State of digital pad Up
but_down_o	1	Output	State of digital pad Down
ana_joy_x_o	8	Output	Position of analog joystick X-axis 0 ◄— 128 —► 255 left center right
ana_joy_y_o	8	Output	Position of analog joystick Y-axis 0 ◄— 128 —► 255 down center up
ana_c_x_o	8	Output	Position of analog C X-axis 0 ◄— 128 —► 255 left center right
ana_c_y_o	8	Output	Position of analog C Y-axis 0 ◄— 128 —► 255 down center up
ana_l_o	8	Output	Position of analog shoulder TL 0 ——————► 255 out in
ana_r_o	8	Output	Position of analog shoulder TR 0 ——————► 255 out in

Table 4: List of IO ports for the basic flavour

Appendix A

Protocol

The communication protocol of the Gamecube controller is proprietary information of Nintendo. It has been reverse engineered by third parties and is therefore not known in full detail. This section gives an overview of what is known so far.

Electrical Layer

Information is exchanged between the gamepad and the host side over a single wire. Both sides seem to implement a open-collector style output driver. Therefore, an external pull-up resistor of 1 kΩ to the 3.43 V supply is required. The electrical '1' is represented by inactivity of both output drivers, whereas the electrical '0' is represented by an active pull to GND.

Bit Transmission Layer

Single bits are encoded by the duty cycle of the signal on the I/O line. The period of the base clock is 5 µs resulting in 200 kBits/s. The logic '0' is encoded by a signal period with duty cycle 80:20. The logic '1' is encoded by a signal period with duty cycle 20:80. The following Figure 2 depicts both encodings.

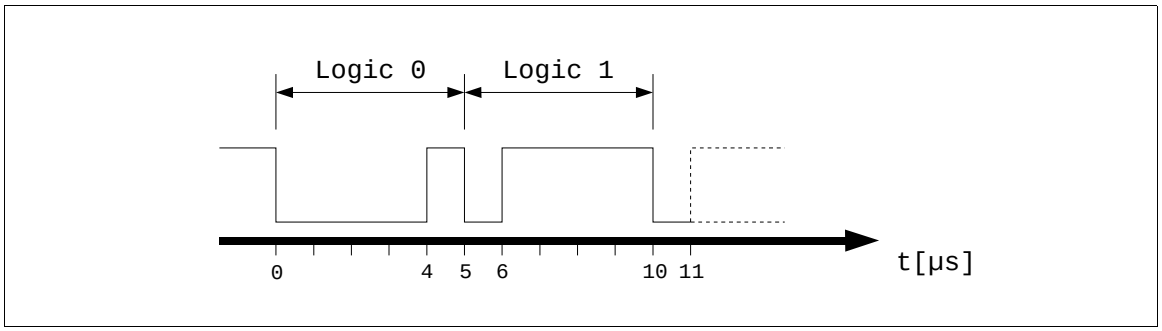

Figure 2: Logic encodings for '0' and '1'

Other sources define the signal period with 4 µs and the duty cycle as 25:75 / 75:25. The experience gatherd with the GCpad core showed that the gamepad does not respond when sticking to the 4 µs scheme. It was definitely necessary to enlarge the period to 5 µs

when transmitting the command. On the other hand, the Receiver is not restricted to a tightly fixed scheme. It will accept any signal period up to 5 µs with the duty cycle varying between 49:51 and 20:80 (and vice versa).

Data Link Layer

The data link layer is quite simple. A stopbit consisting of a logic '1' is appended to both the command and response string. It is not know whether the absence of this stopbit for the command results in a sort of error condition in the gamepad logic.

As written above, the Transmitter automatically adds this stopbit at the end of a 3 byte wide command string. The user has to specify this stopbit explicitely in the command input vector for command strings smaller than 3 byte.

Protocol Layer

Only few is known about the possible commands for the Gamecube pad. Especially no information at all is available on the communcation with the Gameboy Advance adapter. This section lists the known commands and their respective response.

- Init
  Opcode: 0x00
  Length: 1 byte
  Response: 0x0900 for standard Gamecube controller
  See YAGCD for all known types
- Poll
  Opcode: 0x4000302
  Length: 3 bytes
  Response: 8 bytes
  See Table 5 for details
- Rumble
  Opcode: 0x0000000X
  X = 1: turn rumble on
  X = 0: turn rumble off
  Length: 3 bytes
  Response: Unknown

Bit #	Description
63	Error Status
62	Error Latch
61	Always read as '0'
60	Button Start
59	Button Y
58	Button X
57	Button B

Bit #	Description
56	Button A
55	Always read as '1'
54	Button shoulder TL
53	Button shoulder TR
52	Button Z
51	Digital pad Up
50	Digital pad Down
49	Digital pad Right
48	Digital pad Left
47 ... 40	Analog joystick X-axis (left is near 0, center around 128, right is near 255)
39 ... 32	Analog joystick Y-axis (down is near 0, center around 128, up is near 255)
31 ... 24	Analog C X-axis (left is near 0, center around 128, right is near 255)
23 ... 16	Analog C Y-axis (down is near 0, center around 128, up is near 255)
15 ... 8	Analog shoulder TL (out is near 0, in is near 255)
7 ... 0	Analog shoulder TR (out is near 0, in is near 255)

Table 5: Format of the response to a poll command

References

James' excellent page covering many details of the Gamecube controller protocol

http://www.int03.co.uk/crema/hardware/gamecube/gc-control.htm

Yet Another Gamecube Documentation

http://www.gc-linux.org/docs/yagcd/index.html

Refer to sections 5.8, 9.1 and 9.2.