PC Commands and UDMA Reference Manual *

Cicuttin Andres, Crespo Maria,
Garcia Luis, Mannatunga Kasun, Valinoti Bruno

November 2018

*This document is based on "Commands for PC Parallel Port – FPGA Communication " Crespo Maria Liz and Cicuttin Andres. It is a work in progess.

1 Introduction

This document defines a command structure and a communication method, regardless of the physical layer or the type of networks over which they communicate.

2 Scope of document

To describe all commands implemented in the next table:

Command	Description
x_read	reads a single 32 bits data from memory
x_write	writes a single 32 bits data to memory
x_read_reg	reads the N 32 bits Comblock register
x_write_reg	writes the N 32 bits Comblock register
x_read_fifo	reads N 32 bits data from Comblock fifo
x_write_fifo	writes N 32 bits data to Comblock fifo
x_read_mem	reads N 32 bits data words from memory
x_write_mem	writes N 32 bits data words to memory
x_udma	executes a DMA transfer between hardware resources

Table 1: Command reference.

3 Commands

3.1 Read single data (x_read command)

Reads a single 32 bits data mapped in memory, the only parameter the commnad uses is a 32 bits address, this address is intrepreted as an hexadecimal as default, but can be interpreted as a decimal or binary. The command accept one option argument $< -r >$ to select the return value radix, as default is an ASCII value.

As is shown below, the command reads the address ¡addr¿ and returns its value to the standard output as the $< -r >$ modifier imposes.

x_read $< addr > [-r][< outputformat >]$

where $< addr >$: address value, is a 32 bits address and can be written as an hexadecimal value as 0x00000012 or 0x12, like a binary values as 0b00000000000000000000000000010010 and like a decimal value as 0d18.

output format

- d: decimal radix
- h: hexadecimal radix

- b: binary radix
- a: ascii value

3.2 Write single data (x_write command)

Writes a single 32 bits data mapped in memory, the commnad takes two parameters, a 32 bits address and the data to be written. The address is intrepreted as an hexadecimal as default, but can be a decimal or binary. The data can be expressed in hexadecimal, binary, decimal or as an ASCII value.

As is shown below, the command writes in the address ¡addr¿ a 32 bits data value.

x_write $< addr >< data >$

where $< addr >$: address value, is a 32 bits address and can be written as an hexadecimal value as 0x00000012 or 0x12, like a binary values as 0b00000000000000000000000000010010 and like a decimal value as 0d18.

And $< data >$: is a 32 bits data value that can be written as an hexadecimal value as 0x00000012 or 0x12, like a binary values as 0b00000000000000000000000000010010, like a decimal value as 0d18 or like an ASCII as ”DOCE”.

3.3 Read data block (x_read_mem command)

Reads a block of 32 bits width data mapped in memory, the commnad can accept four arguments, a 32 bits address, the number of words to read, the radix of the return data and the increment in words after each memory reading.

The address is intrepreted as an hexadecimal as default, but can be a decimal or a binary value.

The numbers of words to be read is 1 as default, if 0 means infinit reading, other value the quantity desired.

The increment as default is 1, means that reads one word after the other, but can be set as any other natural number.

The radix argument $< -r >$ to select the return value representation, as default is an ASCII value.

As is shown below, the command reads the data block starting in the address ¡addr¿, as a N words length incrementing the ”pointer” by an $< inc >$ between sucesive data words and returns its value to the standard output as the $< -r >$ modifier imposes.

x_read_mem $< addr >< N >< inc > [-r][< outputformat >]$

where $< addr >$: address value, is a 32 bits address and can be written as an hexadecimal value as 0x00000012 or 0x12, like a binary values as 0b00000000000000000000000000010010 and like a decimal value as 0d18.

$< N >$: is an integer value

$< inc >$: a natural value

output format

- d: decimal radix
- h: hexadecimal radix
- b: binary radix
- a: ascii value

3.4 Write data block (x_write_mem command)

Writes a block of 32 bits width data mapped in memory form a file, the commnad can accept four arguments, a 32 bits address, the number of words to write, the data radix and the increment in words after each memory writting.

The address is intrepreted as an hexadecimal as default, but can be a decimal or a binary value.

The numbers of words to be read is 1 as default, if 0 means infinit reading, other value the quantity desired.

The increment as default is 1, means that reads one word after the other, but can be set as any other natural number.

The radix argument $< -r >$ to select the input value representation, as default is an ASCII value.

As is shown below, the command writes the data block starting in the address ¡addr¿, as a N words length incrementing the ”pointer” by an $< inc >$ between sucesive data words and reads the input file as the $< -r >$ modifier imposes.

x_write_mem $< addr >< N >< inc > [-r][< inputfile >]$

3.5 Universal Direct Memory Access (x_udma command)

Moves data between memory mapped resources, nowadays, from RAM to RAM, FIFO to RAM, FIFO to fifo, RAM to FIFO. The command must be followed by five arguments, a 32bits source addres, 32 bits destiny address, step increment source address, step increment destiny address and the numbers of words to be moved.

The address is intrepreted as an hexadecimal as default, but can be a decimal or a binary value.

The increment as default is 1, means that moves one word after the other, but can be set as any other natural number, if 0.

The number of words, if 0 means an infinite number any other means the quantity axpresed as a natural number.

x_udma $< src\_addr >< dst\_addr >< src\_inc >< dst\_inc >< N >$