# ImageMan Register map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register's name | Address | Size (bytes) | Purpose | Place |
| --- | 1 | 1 | Not mapped.  For future groth. | --- |
| Dbg\_reg | 2🡪4 | 3 | Write to / read from this address in SDRAM at Debug mode | Mem\_mng\_top |
| Left\_frame\_reg | 5 | 1 | Left frame of image | Disp\_ctrl\_top |
| Right\_frame\_reg | 6 | 1 | Right frame of image | Disp\_ctrl\_top |
| Top\_frame\_reg | 7 | 1 | Upper frame of image | Disp\_ctrl\_top |
| Buttom\_frame\_reg | 8 | 1 | Lower frame of image | Disp\_ctrl\_top |
| Rd\_burst\_len\_reg | 9🡪A | 2 | Read burst length from SDRAM / Register in Debug mode | tx\_path |
| Dbg\_cmd\_reg | B | 1 | (Clear on Read)  When its value is 0x1, then a Wishbone Read transaction is executed from register / SDRAM, and transmitted through UART. | tx\_path |
| Reg\_addr | C | 1 | Address of register to be read at Debug Mode | tx\_path |
| Type\_reg | D | 1 | Type of message | Mem\_mng\_top, |
| Type\_reg | E | 1 | Type of message | disp\_ctrl\_top, |
| Type\_reg | F | 1 | Type of message | tx\_path |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Type\_reg | 10 | 1 | Type of message | Img\_man\_top |
| x\_start\_reg | 11 | 2 | X crop coordinate | Img\_man\_top |
| y\_start\_reg | 13 | 2 | Y crop coordinate | Img\_man\_top |
| zoom\_reg | 15 | 2 | Zoom ratio | Img\_man\_top |
| cos\_reg | 17 | 2 | Cosine of rotation angle, multiplied by 0x100 | Img\_man\_top |
| Sin\_reg | 19 | 2 | Sine of rotation angle, multiplied by 0x100 | Img\_man\_top |

# Type Register Values

Type Register addresses: 0xD, 0xE, 0xF.

* Type\_register [0]: '0' for Normal Mode, '1' for Debug Mode.
* Type\_register [1]: '0' for Image Transaction, '1' for Summary Transaction.
* Type\_register [2]: Displayed image from VESA generator: '0' for Image Transaction (From SDRAM), '1' for Synthetic Pattern Generator.
* Type\_Register [7]: '0' for Data Transmission, '1' for Registers Transmission.

Type register (all the 3, from address 0xD to 0xF) will be automatically updated at each message transmission from the RX\_PATH.

# Wishbone Cycles

Refer to [Top Architecture](https://runlen.googlecode.com/svn/Docs/Meeting_Summary/Top%20Architecture.docx) or to [Wishbone Specifications](https://runlen.googlecode.com/svn/branches/Spec/Wishbone_Bus/wbspec_b4.pdf).

# Register Access

## Limitations

**Important Note:** Two groups of registers cannot be written / read at the same transaction.

For example:

1. DBG\_REG (Address: 0x2; Burst length: 0x2 = 3 bytes) transaction is OK.
2. DBG\_REG & Left\_Frame\_Reg (Address: 0x2; Burst length: 0x3 = 4 bytes) cannot be executed, and will cause the system to stop function until Reset is executed.

## Write to Registers

***Note***: *Type\_reg should not be written by the software. It is done automatically by the design at each UART message transmission (message = SOF, TYPE … EOF). Type\_reg* *value will be set to the TYPE of the UART message*.

In order to write to register, the following UART message should be sent:

* SOF (=0x64)
* TYPE, where the MSB is '1'. (i.e: 0x80 = write to registers)
* ADDRESS, which is the register's address. (i.e: 2 for dbg\_reg)
* LENGTH, which is the burst length minus 1. (i.e: 2 for dbg\_reg, which is 3 bytes wide)
* PAYLOAD, which is the register's value to be written. (i.e: for dbg\_reg, PAYLOAD might be 0x[01 FA 00] )
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

This will generate Wishbone Write transaction to the relevant register.

## Read Registers

In order to read value from the SDRAM, a UART write transmission to *reg\_addr* (#12 = 0xC in the table above) register should be written, with the register's address to be read, according to the table above.

The following UART messages should be sent:

Message #1 – Register's address

* SOF (=0x64)
* TYPE, where the MSB is '1'. (i.e: 0x80 = write to registers)
* ADDRESS of the *reg\_addr* register (0xB)
* LENGTH, which is the burst length minus 1. In this case – 0 (burst of 1 byte)
* PAYLOAD, which is the register's address to be read. (i.e: 0x5 – left frame register)
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

Message #2 – Burst length

* SOF (=0x64)
* TYPE, where the MSB is '1'. (i.e: 0x80 = write to registers)
* ADDRESS of the *rd\_burst\_len\_reg* register (0x9).
* LENGTH, which is the burst length minus 1. In this case – 0 (burst of 1 byte).
* PAYLOAD, which is the register's length to be read. (i.e: for left frame register, length is 0, which represents 1 byte)
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

Message #3 – Execute read command

* SOF (=0x64)
* TYPE, where the MSB is '1'. (i.e: 0x80 = write to registers)
* ADDRESS of the *dbg\_cmd\_reg* register (0xA).
* LENGTH, which is the burst length minus 1. In this case – 0 (burst of 1 byte).
* PAYLOAD, which should command to start the read sequence. In this case: 0x1.
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

## Register's structure

The registers are wrapped by *wbs\_reg* component, which translates Wishbone transaction to write / read from a specific register, according to the table above.

# SDRAM Access at Debug Mode

## Write to SDRAM in Debug Mode

In order to write to SDRAM to specific address, a UART write transmission to *dbg\_reg* (#2🡪4 in the table above) register should be written, with the SDRAM's address to be read.

The following UART messages should be sent:

Message #1 – Debug Register address

* SOF (=0x64)
* TYPE, where the MSB is '1'. (i.e: 0x80 = write to registers)
* ADDRESS of the *dbg\_reg* register (0x2)
* LENGTH, which is the burst length minus 1. In this case – 2 (burst of 3 bytes = 24 bit address)
* PAYLOAD, which is the register's address to be read. (i.e: 0x[00 00 00])
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

Message #2 – Data to SDRAM

* SOF (=0x64)
* TYPE, where the MSB is '0' and LSB is '1'. (i.e: 0x1 = write to SDRAM in DBG Mode)
* ADDRESS should be 0x0.
* LENGTH, which is the burst length minus 1. For example, for 512 bytes burst, 0x1FF (511 decimal) should be used..
* PAYLOAD, which is the SDRAM data. Payload size should match the LENGTH.
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

## Read SDRAM in Debug Mode

In order to read SDRAM value, a UART write transmission to *dbg\_reg* (#2🡪4 in the table above) register should be written, with the SDRAM's address to be read.

The following UART messages should be sent:

Message #1 – Debug Register address

* SOF (=0x64)
* TYPE, where the MSB is '1'. (i.e: 0x80 = write to registers)
* ADDRESS of the *dbg\_reg* register (0x2)
* LENGTH, which is the burst length minus 1. In this case – 2 (burst of 3 bytes = 24 bit address)
* PAYLOAD, which is the register's address to be read. (i.e: 0x[00 00 00])
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

Message #2 – Burst length

* SOF (=0x64)
* TYPE, where the MSB is '1'. (i.e: 0x80 = write to registers)
* ADDRESS of the *rd\_burst\_len\_reg* register (0x9).
* LENGTH, which is the burst length minus 1. In this case – 0 (burst of 1 byte).
* PAYLOAD, which is the SDRAM bytes to be read. (i.e: 0x10)
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)

Message #3 – Execute read command

* SOF (=0x64)
* TYPE, where the MSB and LSB are '1'. (i.e: 0x81 = Write to Registers + Debug Mode)
* ADDRESS of the *dbg\_cmd\_reg* register (0xA).
* LENGTH, which is the burst length minus 1. In this case – 0 (burst of 1 byte).
* PAYLOAD, which should command to start the read sequence. In this case: 0x1.
* CRC, which is the checksum for TYPE🡪PAYLOAD (inclusive)
* EOF (=0xC8)