# ImageMan Register map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register's name | Address | Size (bytes) | Purpose | Place |
| Type\_reg | 1 | 1 | Type of message | Mem\_mng\_top, disp\_ctrl\_top, tx\_path |
| 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 | 1 | Read burst length from SDRAM / Register in Debug mode | Tx\_path (not implemented yet) |
| Dbg\_cmd\_reg | A | 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 (not implemented yet) |
| Reg\_addr | B | 1 | Address of register to be read at Debug Mode | Tx\_path (not implemented yet) |
| x\_start\_reg | E | 2 | X crop coordinate | Img\_man\_top |
| y\_start\_reg | 10 | 2 | Y crop coordinate | Img\_man\_top |
| zoom\_reg | 12 | 2 | Zoom ratio | Img\_man\_top |
| cos\_reg | 14 | 2 | Cosine of rotation angle, multiplied by 0x100 | Img\_man\_top |
| Sin\_reg | 16 | 2 | Sine of rotation angle, multiplied by 0x100 | Img\_man\_top |

# Type Register Values

* 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 will be updated at each message transmission from the RX\_PATH.

# Register Access

## 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

***Note***: *This functionality is not implemented yet. It will be implemented in the TX\_PATH.*

In order to read register's value, a UART write transmission to *reg\_addr* (#12 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.

# 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).