Reality Coprocessor (RCP)

The RCP is the workhorse of the Nintendo 64. It contains a secondary MIPS R4000-based processor with vector command extensions, known as the Reality Signal Processor (RSP). It also contains SGI's rasterizer hardware, known as the Reality Display Processor (RDP). The RCP also contains a collection of registers controlling virtually all peripheral access on the N64 including video, audio and serial among other things. The main processor communicates with the various RCP components using memory-mapped registers, separated into discrete areas of memory based on function.

NOTE: All RCP registers are 32 bits unless otherwise noted. All registers are also read/write enabled unless otherwise noted.

Register Memory Map

The following is a high level memory map of the RCP. Sections of memory are separated by function as detailed below. Every division is exactly 1MB in size, though the valid address space of accessible registers is far shorter.

Memory Region	Function
0x04040000	SP registers
0x04100000	DP command registers
0x04200000	DP span registers
0x04300000	MI registers
0x04400000	VI registers
0x04500000	AI registers
0x04600000	PI registers
0x04700000	RI registers
0x04800000	SI registers

Video Interface (VI)

The video interface is found at memory offset 0x04400000. The VI registers allow precise control of the video output from the N64. They control hardware setup for timing, color depth and resolution among other things. It is important to note that all registers must be set up properly to ensure proper video output using the N64. The VI does not include services for drawing to the screen as this is done to a frame buffer that is then passed to the VI using the VI_ORIGIN_REG register detailed below. All registers are listed with their mnemonic first, followed by their offset from the video interface base address.

Registers

- **VI_CONTROL_REG (0x00)**

The VI_CONTROL_REG register uses only the bottom 17 bits as mapped out below.

Function	Bits	Description
Color Depth	1:0	Sets the frame buffer bit depth: **00** – Blank (no data or sync) **01** – Reserved **10** – 16 bit color (0bRRRRRGGGGGBBBBBA) **11** – 32 bit color (0xRRGGBBAA)
Gamma Dither	2	Enables gamma dither. Usually left unset. No effect if Gamma Boost is unset.
Gamma Boost	3	Enables gamma boost. Usually left unset.
Divot	4	Enables divot. Usually used in conjunction with anti-aliasing.
Reserved	5	Always zero.
Serrate	6	Set this if running in interlaced mode. Clear otherwise.
Diagnostic	7	Always zero.
Anti-Alias Mode	9:8	Sets the anti-alias mode: **00** – Anti-aliasing and resampling, always fetch extra lines **01** – Anti-aliasing and resampling, fetch when needed **10** – Resampling only **11** – Neither (replicate pixels, no interpolation)
Undefined	10	Always zero.
Diagnostic	11	Always zero.
Pixel Advance	15:12	Typical observed value is 0x3.
Dither Filter	16	Enable dither filter. Usually used in conjunction with 16 bit color to smooth out gradients.

The gamma dither and gamma boost bits are somewhat curious. It is recommended to leave these on, but a better picture color wise can be produced when disabling them. Many games leave the gamma boost bit off. Disabling gamma boost gives a smoother gradient of colors, while enabling it produces a much brighter but more washed out picture. The gamma dither bit only affects the picture when gamma boost is enabled. The gamma dither bit will slightly brighten the darker colors while leaving the lighter colors unaffected.

One caveat to be careful of is that anti-aliasing must be enabled when running in 320x240 16bpp mode. If you do not at least enable resampling anti-aliasing when running in this mode, you will get corrupted video. All other modes do not seem to have this restriction.

- **VI_DRAM_ADDR_REG (0x04)**

The VI_DRAM_ADDR_REG register allows a frame buffer in memory to be assigned as the currently used frame buffer. This allows a programmer to design a single, double or even triple buffered system without relying on hardware. This should be updated when the system is in vertical blank to swap buffers for the smoothest possible video update. Frame buffers can be anywhere in memory, but it's a good idea to have them aligned on a boundary that makes it possible to DMA to them. Also, be sure to pass the uncached memory address to the VI by or'ing with the bit mask `0xA0000000` (MIPS `kseg1`) or you will have to manually invalidate the cache before swapping buffers.

- **VI_H_WIDTH_REG (0x08)**

The VI_H_WIDTH_REG register controls the horizontal width in pixels of the frame buffer. Typical values are `0x0140` for 320x240 video, or `0x0280` for 640x480 video. This value, combined with the horizontal and vertical scaling, allows for the video mode width and height to be set.

- **VI_V_INTR_REG (0x0C)**

The VI_V_INTR_REG register works in combination with the MI to assert an interrupt when the video hardware hits a certain line (usually used to set up a vblank interrupt). Don't forget to set the associated mask bits in the MI or you will not get an interrupt. To clear the interrupt once it is asserted, write to the VI_V_CURRENT_LINE_REG register.

- **VI_V_CURRENT_LINE_REG (0x10)**

The VI_V_CURRENT_LINE_REG register is updated once per vertical line to reflect the current line being displayed. In interlaced mode, the least significant bit will be constant for the current field. A write to this register does not change the vertical line currently being displayed. Instead, it clears any currently asserted interrupt caused by the VI_V_INTR_REG register.

- **VI_TIMING_REG (0x14)**

The VI_TIMING_REG register controls several timing aspects of the video signal. It is mapped out into different fields as detailed below.

Function	Bits	Description
HSync Width	7:0	Specifies the horizontal sync width in pixels.
Burst Width	15:8	Specifies the color burst width in pixels.
VSync Width	19:16	Specifies the vertical sync height in lines.
Burst Start	29:20	Specifies the color burst start as an offset in pixels from the horizontal sync.

- **VI_V_SYNC_REG (0x18)**

The VI_V_SYNC_REG register controls the number of lines per frame (or the number of half-lines per field). Values are typically taken straight from the NTSC or PAL specifications. For NTSC, this is `0x20D` (525 lines). For PAL, this is `0x271` (625 lines).

- **VI_H_SYNC_REG (0x1C)**

The VI_H_SYNC_REG register is split into two fields as detailed below.

Function	Bits	Description
Line Duration	11:0	Duration of a line represented in ¼ pixels.
HSync Period	20:16	Leap pattern used only in PAL formats.

- **VI_H_SYNC_LEAP_REG (0x20)**

The VI_H_SYNC_LEAP_REG register is split into two identical fields, set to the same value as the line duration in the VI_H_SYNC_REG register.

Function	Bits	Description
Line Duration	11:0	Duplicate of data represented in 11:0 of VI_H_SYNC_LEAP_REG.
Line Duration	27:16	Duplicate of data represented in 11:0 of VI_H_SYNC_LEAP_REG.

- **VI_H_VIDEO_REG (0x24)**

The VI_H_VIDEO_REG register is split into two fields that specify the start and end of the active horizontal video in pixels. The difference between these values is normally 640 pixels.

Function	Bits	Description
Horizontal End	9:0	End of active video measured in pixels.
Horizontal Start	25:16	Start of active video measured in pixels.

- **VI_V_VIDEO_REG (0x28)**

The VI_V_VIDEO_REG register is split into two fields that specify the start and end of the active vertical video in lines. The difference between these values is normally 474 lines.

Function	Bits	Description
Vertical End	9:0	End of active video measured in lines.
Vertical Start	25:16	Start of active video measured in lines.

- **VI_V_BURST_REG (0x2C)**

The VI_V_BURST_REG register is split into two fields that specify the start and end of the color burst in vertical lines. This in combination with the VI_TIMING_REG register allows for full control over video timing.

Function	Bits	Description
Color Burst End	9:0	End of active video measured in lines.
Color Burst Start	25:16	Start of active video measured in lines.

- **VI_X_SCALE_REG (0x30)**

The VI_X_SCALE_REG register controls the horizontal scaling up factor between the frame buffer and the final image as displayed on the screen. It also controls the horizontal subpixel offset. Both values are specified as 2.10 fixed point integers.

Function	Bits	Description
Horizontal Scale	11:0	Reciprocal of the horizontal scale value.
Subpixel Offset	27:16	Horizontal subpixel offset.

- **VI_Y_SCALE_REG (0x34)**

The VI_Y_SCALE_REG register controls the horizontal scaling up factor between the frame buffer and the final image as displayed on the screen. It also controls the vertical subpixel offset. Both values are specified as 2.10 fixed point integers.

Function	Bits	Description
Vertical Scale	11:0	Reciprocal of the vertical scale value.
Subpixel Offset	27:16	Vertical subpixel offset.

Sample VI Values

The following is the register setup for a 320x240 16 bit, resampling only anti-aliased video mode for NTSC, PAL and PAL-M.

Register	NTSC Value	PAL Value	MPAL Value
VI_CONTROL_REG	0x0000320e	0x0000320e	0x0000320e
VI_DRAM_ADDR_REG	*Unspecified*	*Unspecified*	*Unspecified*
VI_H_WIDTH_REG	0x00000140	0x00000140	0x00000140
VI_V_INTR_REG	0x00000200	0x00000200	0x00000200
VI_V_CURRENT_LINE_REG	*Unspecified*	*Unspecified*	*Unspecified*
VI_TIMING_REG	0x03e52239	0x0404233a	0x04651e39
VI_V_SYNC_REG	0x0000020d	0x00000271	0x0000020d
VI_H_SYNC_REG	0x00000c15	0x00150c69	0x00040c11
VI_H_SYNC_LEAP_REG	0x0c150c15	0x0c6f0c6e	0x0c190c1a
VI_H_VIDEO_REG	0x006c02ec	0x00800300	0x006c02ec
VI_V_VIDEO_REG	0x002501ff	0x005f0239	0x002501ff
VI_V_BURST_REG	0x000e0204	0x0009026b	0x000e0204
VI_X_SCALE_REG	0x00000200	0x00000200	0x00000200
VI_Y_SCALE_REG	0x00000400	0x00000400	0x00000400

Setup and Operation

The proper way to set up a frame buffer on the N64 is simple. Most register values can be simply copied from the table above as they relate to the actual generation of the video signal and not software parameters relating to a buffer in memory. The VI_DRAM_ADDR_REG register is normally left set to zero while setting other registers to avoid flicker or garbage appearing on the screen. The N64 will output video at 640x480 regardless of the frame buffer width or height. The VI_X_SCALE_REG and VI_Y_SCALE_REG registers allow one to scale an arbitrary buffer in memory to fit in the 640x480 region. The VI_H_WIDTH_REG register allows the frame buffer to be set to arbitrary width and only serves to allow the N64 to determine where a particular line ends in the frame buffer.

Curiously, the N64 treats the vertical component as half-lines and as such a scaling factor of 1X will set the vertical resolution to 240 pixels. A horizontal scaling factor of 1X will set the horizontal resolution to 640 pixels as expected. The horizontal and vertical scaling factors can be set independent of the VI_H_WIDTH_REG register. No vertical component is specified for the frame buffer as the N64 hardware will use as many lines in the frame buffer as are needed to render the image to the screen.

Double or triple buffering is achieved in software by changing the VI_DRAM_ADDR_REG register while in the vertical retrace period. An interrupt is generated on the half-line specified by the VI_V_INTR_REG register. This means that even on interlaced televisions, an interrupt will only happen once the entire scene has been drawn (even and odd lines). It is recommended to pass to the VI_DRAM_ADDR_REG register the uncached memory address of the frame buffer as this prevents artifacts due to caching problems. When using the RDP to render a scene, the Color Image Buffer is set to these addresses.

Sample Video Mode Configurations

320x240		
Register	**Setting**	
H_WIDTH	0x140	320
X_SCALE	0x200	2X
Y_SCALE	0x400	1X

640x240		
Register	**Setting**	
H_WIDTH	0x280	640
X_SCALE	0x400	1X
Y_SCALE	0x400	1X

640x480		
Register	**Setting**	
H_WIDTH	0x280	640
X_SCALE	0x400	1X
Y_SCALE	0x800	.5X

Rasterizer Interface (DP)

The Reality Display Processor control interface is found at `0x04100000`. The DP registers allow commands to be sent to the RDP and for RDP settings to be toggled. The RDP works in tandem with the Video Interface to allow for hardware-drawn polygons. The RDP contains its own texture memory (4KB) from which it can texture rectangles or triangles. The RDP also has the capability of converting textures from other color modes as well as drawing shaded and z-buffered triangles. Under normal game operation, the RDP is usually controlled via the RSP as the drawing commands are very low level. However, it is possible to control the RDP directly from a running program on the N64 without coding a specialized program for the RSP. Sending a command to the RDP is done in a similar fashion to sending a DMA request to one of the DMA controllers onboard the N64.

- **DP_START_REG (`0x00`)**

The DP_START_REG register specifies the start location in RDRAM or RSP DMEM where an RDP command can be found. The RDP will pull the command from RDPAM or RSP DMEM depending on the state of the DMEM DMA bit in the DP_STATUS_REG register. When in RDRAM mode, the RDP can handle any valid address, but be careful as caching issues may arise with cached areas of RDRAM. When in RSP DMEM mode, the RDP expects the address to be in the range of `0x000` to `0xFFF`. This is identical to the valid data memory range with respect to the RSP. This should be written to before the DP_END_REG.

- **DP_END_REG (`0x04`)**

The DP_END_REG register specifies the end location in RDRAM or RSP DMEM where an RDP command can be found. The same restrictions that apply to the range of addresses passed to DP_START_REG apply here as well. Writing to this register will kick off the process of loading a command from memory and performing it, so the DP_END_REG should be written to after the DP_START_REG. Commands should be issued one at a time from memory, not as a block of commands.

- **DP_CURRENT_REG (`0x08`)**

The DP_CURRENT_REG reflects the current location of the RDP in loading commands from memory.

- **DP_STATUS_REG (0x0C)**

The DP_STATUS_REG register allows arbitrary modes to be set on the RDP. Most notably, this is how one would tell the RDP from which processor to pull its data. The bits have different meaning depending on whether it is read from or written to as detailed below.

Reading from DP_STATUS_REG		
Function	**Bit**	**Description**
DMEM DMA Status	0	This bit will be set if DMEM DMA mode is set. Use this bit to determine where the RDP will expect commands to come from.
Freeze Status	1	This bit will be set if Freeze has been set.
Flush Status	2	This bit will be set if Flush has been set.
Start GCLK	3	*Unknown.*
TMEM Busy	4	This bit will be set if the TMEM is in use on the RDP.
Pipe Busy	5	This bit will be set if the graphics pipe is in use on the RDP.
Command Busy	6	This bit will be set if the RDP is currently executing a command.
Command Buffer Busy	7	This bit will be set if the RDP command buffer is in use.
DMA Busy	8	This bit will be set if the RDP is in the process of copying a command to the command buffer.
End Valid	9	*Unknown. Normally set to zero.*
Start Valid	10	*Unknown. Normally set to zero.*

Possibly the only bits worth mentioning are the DMEM DMA Status, End Valid and Start Valid bits. The DMEM DMA bit indicates whether the RDP is in the correct mode to receive commands from the RDRAM. While the actual meaning of the End Valid and Start Valid bits is unknown, they are often used in commercial games to tell if it is safe to issue the next command. Before issuing any command to the RDP, check to see that both the Start Valid and End Valid bits are cleared. Waiting until these bits are both cleared before issuing another command ensures that the next command won't overwhelm the RDP command buffer. Failing to do so could cause the N64 to freeze.

Writing to DP_STATUS_REG		
Function	**Bit**	**Description**
Clear DMEM DMA Mode	0	Writing a 1 to this bit will set the RDP to pull commands from RDRAM.
Set DMEM DMA Mode	1	Writing a 1 to this bit will set the RDP to pull commands from the RSP DMEM.
Clear Freeze	2	Writing a 1 to this bit will clear the RDP freeze bit.
Set Freeze	3	Writing a 1 to this bit will set the RDP freeze bit.
Clear Flush	4	Writing a 1 to this bit will clear the RDP flush bit.
Set Flush	5	Writing a 1 to this bit will set the RDP flush bit.
Clear TMEM Counter	6	Writing a 1 to this bit will set the DP_TMEM_REG register to 0.
Clear Pipe Counter	7	Writing a 1 to this bit will set the DP_PIPEBUSY_REG register to 0.
Clear Command Counter	8	Writing a 1 to this bit will set the DP_BUFBUSY_REG register to 0.
Clear Clock Counter	9	Writing a 1 to this bit will set the DP_CLOCK_REG register to 0.

- **DP_CLOCK_REG (0x10)**

The functionality of the DP_CLOCK_REG register is unknown. It can be reset using the DP_STATUS_REG.

- **DP_BUFBUSY_REG (0x14)**

The functionality of the DP_BUFBUSY_REG register is unknown. It can be reset using the DP_STATUS_REG, and counts up to a predetermined value.

- **DP_PIPEBUSY_REG (0x18)**

The functionality of the DP_PIPEBUSY_REG register is unknown. It can be reset using the DP_STATUS_REG, and counts up to a predetermined value.

- **DP_TMEM_REG (0x1C)**

The functionality of the DP_TMEM_REG register is unknown. It can be reset using the DP_STATUS_REG, and counts up to a predetermined value.

Rasterizer Command Set

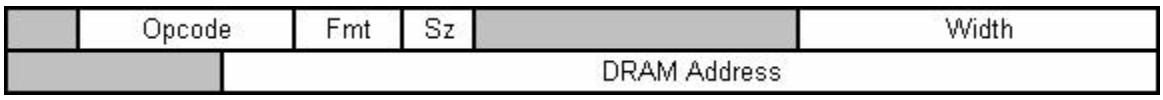
The rasterizer is controlled by a set of commands, most of them 64 bits in length. They can be split into three different categories: configuration, drawing and pipeline control. The RDP is capable of rasterizing rectangles (textured or untextured) and a variety of triangles. Coordinates of geometry to be rasterized are given with respect to a virtual frame buffer (also known as screen space) which is four times larger in width and height to the screen resolution. It is possible that this is controlled by the Pixel Advance bits in the VI_CONTROL_REG register but this is unverified.

All commands are presented with their common name, followed by the expected data format as passed to the RDP and finally a description of the command and its use. Illegal values for a particular command are highlighted in red.

Configuration Commands

These commands allow one to configure the RDP for various rasterization options as well as set up and load textures into TMEM.

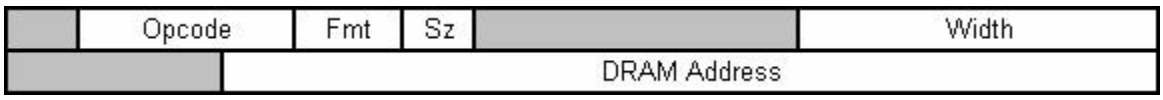
- **Set Color Image (64 bits)**



Function	Bits	Description
Opcode	61:56	`0x3f`
Color Format	55:53	Controls the output format of the rasterized image: **000** – RGBA **001** – YUV **010** – Color Index **011** – Intensity Alpha **011** – Intensity
Pixel Size	52:51	Size of an individual pixel in terms of bits: **00** – 4 bits **01** – 8 bits (Color Index) **10** – 16 bits (RGBA) **11** – 32 bits (RGBA)
Width	41:32	Width of frame buffer in memory minus one. This width should correspond to the width of the frame buffer given to the VI_H_WIDTH_REG register.
DRAM Address	25:0	The address in memory where the frame buffer resides.

The Set Color Image command sets the location of the frame buffer in memory where the RDP should rasterize geometry. The Color Format bits are normally set to RGBA and the Pixel Size set to match the bit depth specified in the VI_CONTROL_REG register. However, if desired the RDP can be used to draw to a color indexed frame buffer which can be later used as a texture for another primitive. The DRAM Address can be modified at any time as long as a Sync Full is called to ensure all geometry has been rendered to the frame buffer. In this way, it is possible to use double or triple buffering techniques that take advantage of the RDP for some drawing.

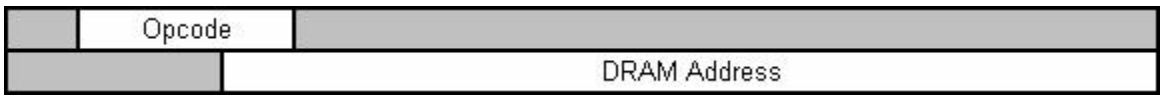
- **Set Texture Image (64 bits)**



Function	Bits	Description
Opcode	61:56	`0x3d`
Color Format	55:53	Controls the input format of the texture buffer in RDRAM: **`000`** – RGBA **`001`** – YUV **`010`** – Color Index **`011`** – Intensity Alpha **`011`** – Intensity
Pixel Size	52:51	Size of an individual pixel in terms of bits: **`00`** – 4 bits (Color Index, IA, I) **`01`** – 8 bits (Color Index, IA, I)) **`10`** – 16 bits (RGBA, YUV, IA) **`11`** – 32 bits (RGBA)
Width	41:32	Width of texture buffer in memory minus one. This is provided merely for convenience, so individual textures can be easily copied out of a composite image map.
DRAM Address	25:0	The address in memory where the texture buffer resides.

The Set Texture Image command sets the location of the current texture buffer in memory where subsequent Load Tile commands source their data. Individual textures are not loaded by this command as it only gives the RDP information on where textures appear in RDRAM and how to interpret the memory. Before a texture can be used by a primitive, it still needs to be loaded into texture memory. When rendering to a frame buffer that will be used as a texture buffer in the future, be sure to perform a Sync Full to ensure all primitives are rendered to the texture memory.

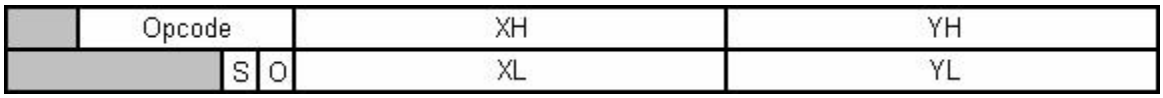
- **Set Z Image (64 bits)**



Function	Bits	Description
Opcode	61:56	0x3e
DRAM Address	25:0	The address in memory where the z-buffer resides.

The Set Z Image command sets the location of the z-buffer in memory where the RDP should read and write pixel depths. The format is a 16 bit integer per pixel with the value of 0 denoting infinitely far away. It is recommended to clear the z-buffer using the RDP itself; many games will set the z-buffer as the color image and draw a black rectangle the size of the screen to zero the depth at every pixel. The width and height of the z-buffer in memory should match the color image. This command is not required unless triangles with depth are desired. If no z-buffering is used, the RDP will draw geometry in the order received on the pipeline.

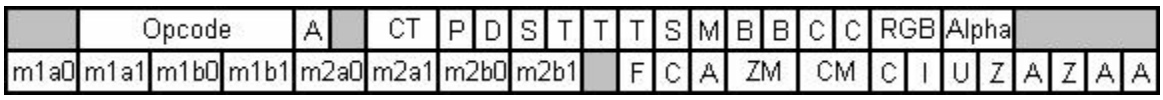
- **Set Scissor (64 bits)**



Function	Bits	Description
Opcode	61:56	0x2d
XH	55:44	X coordinate of the top left corner of the scissor box.
YH	43:32	Y coordinates of the top left corner of the scissor box.
Scissor Field	25	Set to enable scissoring of odd or even lines for interlaced displays.
Odd/Even	24	Specifies which field lines to skip: **0** – Skip all odd lines **1** – Skip all even lines
XL	23:12	X coordinate of the bottom right corner of the scissor box.
YL	11:0	Y coordinate of the bottom right corner of the scissor box.

The Set Scissor command tells the RDP to rasterize geometry falling inside the scissor box. Coordinates are given with respect to screen space. If needed, even or odd field lines can be skipped for a given scene. If no clipping is desired, set the scissor box corners to the absolute minimum and maximum valid pixel locations.

- **Set Other Modes (64 bits)**



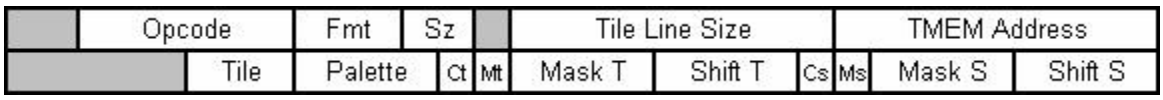
Function	Bits	Description
Opcode	61:56	`0x2f`
Atomic Primitive Enable	55	Force primitive to be written to the frame buffer before processing next primitive.
Cycle Type	53:52	Pipeline rasterization mode: **`00`** – 1 Cycle **`01`** – 2 Cycle **`10`** – Copy **`11`** – Fill
Perspective Correction Enable	51	Enable perspective correction on textures.
Detail Texture Enable	50	Enable detail texture.
Sharpen Texture Enable	49	Enable sharpen texture.
Texture LOD Enable	48	Enable texture level of detail (mipmapping).
TLUT Enable	47	Enable texture lookup table. This is useful when textures are color mapped but the desired output to the frame buffer is RGB.
TLUT Type	46	Type of texels (texture color values) in TLUT table: **`0`** – 16 bit RGBA (`0bRRRRRGGGGGBBBBBA`) **`1`** – 16 bit IA (`0xIIAA`)
Sample Type	45	Type of texture sampling: **`0`** – 1x1 (point sample) **`1`** – 2x2
Mid Texel Enable	44	Enable 2x2 half-texel sampling for texture filter.
Bi-Linear Interpolation 0	43	Enables bi-linear interpolation for cycle 0 when 1 Cycle or 2 Cycle mode is enabled.
Bi-Linear Interpolation 1	42	Enables bi-linear interpolation for cycle 1 when 1 Cycle or 2 Cycle mode is enabled.
Color Convert	41	Color convert the texel outputted by the texture filter on cycle 0.
Chroma Key Enable	40	Enable chroma keying.

Function	Bits	Description
RGB Dither Type	39:38	Type of dithering done to RGB values in 1 Cycle or 2 Cycle modes: **00** – Magic square matrix **01** – Bayer matrix **10** – Noise **11** – No dither
Alpha Dither Type	37:36	Type of dithering done to alpha values in 1 Cycle or 2 Cycle modes: **00** – Pattern **01** – ~Pattern **10** – Noise **11** – No dither
Blend m1a,0	31:30	Multiply blend 1a input in cycle 0.
Blend m1a,1	29:28	Multiply blend 1a input in cycle 1.
Blend m1b,0	27:26	Multiply blend 1b input in cycle 0.
Blend m1b,1	25:24	Multiply blend 1b input in cycle 1.
Blend m2a,0	23:22	Multiply blend 2a input in cycle 0.
Blend m2a,1	21:20	Multiply blend 2a input in cycle 1.
Blend m2b,0	19:18	Multiply blend 2b input in cycle 0.
Blend m2b,1	17:16	Multiply blend 2b input in cycle 1.
Force Blend	14	Enable force blend.
Coverage Enable	13	Enable use of coverage bits in alpha calculation.
Alpha Multiply Enable	12	Enable multiplying coverage bits by alpha value for final pixel alpha: **0** – Alpha = CVG **1** – Alpha = CVG * A
Z Mode	11:10	Mode select for Z buffer: **00** – Opaque **01** – Interpenetrating **10** – Transparent **11** – Decal
Coverage Mode	9:8	Mode select for handling coverage values: **00** – Clamp **01** – Wrap **10** – Force to full coverage **11** – Don’t write back

Function	Bits	Description
Color on Coverage	7	Only update color on coverage overflow. Useful for transparent surfaces.
Image Read Enable	6	Enable coverage read/modify/write access to frame buffer.
Update Z Enable	5	Enable writing new Z value if color write is enabled.
Z Compare Enable	4	Enable conditional color write based on depth comparison.
Anti-alias Enable	3	Enable anti-aliasing based on coverage bits if force blend is not enabled.
Z Source	2	Select the source of the Z value: `0` – Pixel Z `1` – Primitive Z
Alpha Compare	1	Select source for alpha compare: `0` – Random noise `1` – Blend alpha
Alpha Compare Enable	0	Enable conditional color write based on alpha compare.

Many of these modes are purely optional and only affect the quality of the final image in the frame buffer. However, a select few of the modes must be set to a specific value for certain RDP commands to work properly. For these options, the command in question will specify which modes need to be set in the Set Other Modes command to operate properly. The texture filter referred to multiple times is simply the 1 Cycle or 2 Cycle pipeline mode. This is not enabled when doing certain simple geometry such as untextured rectangles. The coverage bits are only accessible to the RDP and the DAC chips doing the video conversion to display to the screen. The RDRAM in the N64 is actually 9 bits wide, but only the lower 8 bits are accessible to software. One bit of coverage is assigned to each byte in the frame buffer. This means that 16 bit RGBA buffers will include 2 coverage bits per pixel, and 32 bit RGBA buffers will include 4 coverage bits per pixel. Unless certain modes are set as specified above, these coverage bits will not be used in the final calculation when sending pixel colors to the TV.

- **Set Tile (64 bits)**

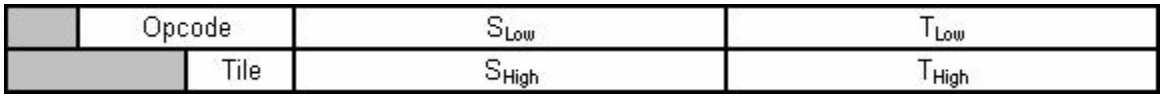


Function	Bits	Description
Opcode	61:56	0x35
Color Format	55:53	Controls the output format of the rasterized image: **000** – RGBA **001** – YUV **010** – Color Index **011** – Intensity Alpha **011** – Intensity
Pixel Size	52:51	Size of an individual pixel in terms of bits: **00** – 4 bits **01** – 8 bits (Color Index) **10** – 16 bits (RGBA) **11** – 32 bits (RGBA)
Tile Line Size	49:41	The width of the texture to be stored in terms of 64 bit words.
TMEM Address	40:32	The location in texture memory to store the texture in terms of 64 bit words. This implies that textures must be 8 byte aligned in TMEM.
Tile	26:24	The tile ID to give this texture.
Palette	23:20	For 4 bit sprites, this is the palette number. In effect, this is the upper 4 bits to make an 8 bit color index pixel which will then be looked up in a palette.
Clamp Enable T	19	Enables clamping in the T direction when texturing primitives.
Mirror Enable T	18	Enable mirroring in the T direction when texturing primitives.
Mask T	17:14	Number of bits to mask or mirror in the T direction.
Shift T	13:10	Level of detail shift in the T direction.
Clamp Enable S	9	Enables clamping in the S direction when texturing primitives.
Mirror Enable S	8	Enable mirroring in the S direction when texturing primitives.
Mask S	7:4	Number of bits to mask or mirror in the S direction.
Shift S	3:0	Level of detail shift in the S direction.

The Set Tile command tells the RDP about a texture that is to be loaded out of a previously specified Texture Image. To calculate the Tile Line Size field, multiply the width of the texture by the bit depth (in bytes), round up to the next multiple of 8, and

then divide the resulting number by 8. For both Mask S and Mask T, a mask value of 2 would result in the bit mask of 0x3. If mirror enable is off the coordinate is masked to this mask value, allowing tiling. If mirror enable is on tiling is enabled and the coordinate is mirrored every other tile. If this value is 0 then the RDP defaults to clamping for the coordinate. Mirroring, tiling and clamping can only be done on a power-of-two boundary, but textures that do not need these features can be of arbitrary size as long as they fit within TMEM.

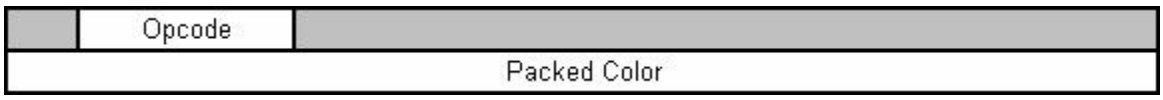
- **Load Tile (64 bits)**



Function	Bits	Description
Opcode	61:56	0x34
S_{Low}	55:44	Low S coordinate of texture (10.5 fixed point format).
T_{Low}	43:32	Low T coordinate of texture (10.5 fixed point format).
Tile	26:24	The tile ID to load this texture into.
S_{High}	23:12	High S coordinate of texture (10.5 fixed point format).
T_{High}	11:0	High T coordinate of texture (10.5 fixed point format).

The Load Tile command tells the RDP to pull actual texture data out of a texture set by Set Texture Image and place it in TMEM in a texture slot defined by Set Tile. The tile ID references a tile that has been set up by Set Tile. S,T_{Low} specifies the location of the top left corner where texture copying should start. S,T_{High} specifies the location of the bottom right corner where texture copying should stop. Note that a texture copy includes the row and column specified by S,T_{High}. To load simple textures out of RDRAM into TMEM one would use 0,0 for S,T_{Low} and S,T_{High} would be set to the width and height of the sprite minus one. Note that the RDP stores the S,T_{Low} used when loading a texture into TMEM and calculates the mapping of texture space to screen space relative to this offset.

- **Set Fill Color (64 bits)**



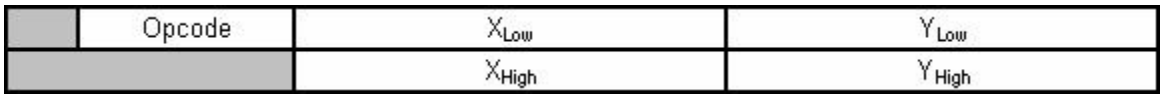
Function	Bits	Description
Opcode	61:56	0x37
Packed Color	31:0	Color to fill non-textured rectangles with.

The Set Fill Color command tells the RDP what color to use when rasterizing a Fill Rectangle command. For 32 bit color mode, this is the RGBA value to set each pixel to. For 16 bit color mode, this is two 16 bit colors packed into one 32 bit field. It is possible to rasterize vertically striped rectangles by specifying two different 16 bit colors in the Set Fill Color command.

Drawing Commands

The drawing commands are dependent on proper RDP setup using the correct configuration commands. Effort will be given to ensure valid setup steps are specified for each of the drawing commands.

- **Fill Rectangle (64 bits)**



Function	Bits	Description
Opcode	61:56	0x36
X_{Low}	55:44	Low X coordinate of rectangle (10.2 fixed point format).
Y_{Low}	43:32	Low Y coordinate of rectangle (10.2 fixed point format).
X_{High}	23:12	High X coordinate of rectangle (10.2 fixed point format).
Y_{High}	11:0	High Y coordinate of rectangle (10.2 fixed point format).

The Fill Rectangle command will draw a rectangle to the frame buffer using the color specified in a Set Fill Color command. Before using the Fill Rectangle command, the RDP must be set up in fill mode using the Cycle Type parameter in the Set Other Modes command.

- **Texture Rectangle (128 bits)**

	Opcode	X_{Low}	Y_{Low}
	Tile	X_{High}	Y_{High}

S	T
DsDx	DtDy

Function	Word	Bits	Description
Opcode	0	61:56	0x24
X_{Low}	0	55:44	Low X coordinate of rectangle (10.2 fixed point format).
Y_{Low}	0	43:32	Low Y coordinate of rectangle (10.2 fixed point format).
Tile	0	26:24	Tile ID of the texture to use.
X_{High}	0	23:12	High X coordinate of rectangle (10.2 fixed point format).
Y_{High}	0	11:0	High Y coordinate of rectangle (10.2 fixed point format).
S	1	63:48	S coordinate of texture at the top left corner of the rectangle (s.10.5 fixed point format).
T	1	47:32	T coordinate of texture at the top left corner of the rectangle (s.10.5 fixed point format).
DsDx	1	31:16	Change in S per change in X (s.5.10 fixed point format).
DtDy	1	15:0	Change in T per change in Y (s.5.10 fixed point format).

The Texture Rectangle command will draw a rectangle to the frame buffer using a texture loaded with the Load Texture command. Before using the Fill Rectangle command, the RDP must be set up in copy mode using the Cycle Type parameter in the Set Other Modes command. The S,T coordinate is the S,T coordinate of the texture in RDRAM, not in TMEM. For example, consider the following setup. There is a 40x40 texture in RDRAM that is pointed to with the Set Texture Image command. A 20x20 texture is loaded into memory using a Load Tile command with parameters S,T_{Low} equal to 20,20 and S,T_{High} equal to 39,39. A 20x20 textured rectangle is drawn using a Texture Rectangle command with parameters X,Y_{Low} equal to 100,100 and X,Y_{High} equal to 119, 119. The S,T parameters in this case would be 20,20, **not** 0,0. This is because S,T is specified relative to the original texture, not the locally copied out portion in TMEM. To display sprites, one would normally set DsDx to 4.0 and DtDy to 1.0. It is suspected that DsDx needs to be 4.0 due to the Pixel Advance parameter given in the VI setup. Effectively, DsDx and DtDy can be viewed as the inverse of the scaling value desired for horizontal and vertical texture scaling.

Pipeline Commands

The pipeline commands do not have any parameters. They simply instruct the graphics pipeline to stall until a particular event. This is useful for such operations as ensuring primitives are affected by texture loads and forcing rasterization of all primitives before swapping buffers.

- **Sync Load (64 bits)**

Function	Bits	Description
Opcode	61:56	0x31

Sync Load will stall any subsequent texture load commands until all preceding geometry has been rasterized. This should be placed before a load texture command that is overwriting a texture or TLUT used by up to two preceding primitives.

- **Sync Tile (64 bits)**

Function	Bits	Description
Opcode	61:56	0x28

Sync Tile will stall any tile load operations until any preceding command finishes reading from texture memory. It is unclear how this differs significantly from Sync Load.

- **Sync Pipe (64 bits)**

Function	Bits	Description
Opcode	61:56	0x27

Sync Pipe will stall the pipeline until any primitive on the pipeline is finished using previously issued configuration commands. This encompasses all stalls in Sync Load as well as several other configuration commands. The Primitive Color and Primitive Depth commands are exempt from needing a Sync Pipe. If one is careful to order RDP commands properly, this can be left out. For example, set texture image does not affect the rasterization of triangles or rectangles, so it can be called immediately after a rectangle or triangle command without the need of a Sync Pipe to separate. Note that calling this spuriously will cause rendering artifacts such as the RDP rendering geometry to the wrong frame buffer. In essence, do not call Sync Pipe when there is nothing that needs to be waited on.

- **Sync Full (64 bits)**

Function	Bits	Description
Opcode	61:56	0x29

Sync Full will stall the pipeline until the last command that reads or writes from the frame buffer finishes. This is useful for ensuring an entire scene is fully drawn before swapping frame buffers or reusing a frame buffer as a texture image. Before pointing the VI at the frame buffer in memory, ensure that this command is called so that all geometry appears on the screen without flicker. Due to the fact that often a program must know when the RDP has finished processing, Sync Full will cause a DP interrupt when the RDP completes processing and encounters the Sync Full. This is the only command that causes a DP interrupt and thus can be useful for programs that must wait for rasterization to complete before continuing. See the MIPS Interface (MI) registers for more details.

Glossary

- **Fixed Point Format –** A format used to represent fractions without resulting to a floating point value. Usually specified as M.N where M is the number of bits in the whole number and N is the number of bits in the fraction. A M.N fixed point number is interpreted as MN * 2^{-N}. Sometimes represented as s.M.N which signifies a M.N fixed point number with a leading sign bit. A s.M.N fixed point number is interpreted as -1^{s} * MN * 2^{-N}
- **Half Line –** Used to refer to a vertical line with respect to interlaced video modes. For all intents and purposes, a half line can be thought of as a single row of pixels to be displayed on the screen. In an interlaced setup, odd half lines will be displayed one frame, followed by even frames on the next.
- **RDRAM** – Main memory of the Nintendo 64.
- **Screen Coordinates** – Coordinate system used to reference pixels as rendered on the screen. Usually denoted in X,Y fashion and 0,0 is the top left pixel on the screen.
- **Texture Coordinates –** Coordinate system used to reference pixels as found in a texture. Usually denoted in S,T fashion and 0,0 is the top left pixel in a particular texture.
- **TMEM** – Texture memory. The RDP has 4KB and uses the bottom 2KB for textures and the top 2KB for palette entries. In some cases, textures can span all 4KB.