Tags (ARM to VC)

VideoCore

Get firmware revision

- Tag: 0x00000001
- Request:
  - Length: 0
- Response:
  - Length: 4
  - Value:
    - u32: firmware revision

Hardware

Get board model

- Tag: 0x00010001
- Request:
  - Length: 0
- Response:
  - Length: 4
  - Value:
    - u32: board model

Get board revision

- Tag: 0x00010002
- Request:
  - Length: 0
- Response:
  - Length: 4
  - Value:
    - u32: board revision

Get board MAC address

- Tag: 0x00010003
- Request:
  - Length: 0
- Response:
  - Length: 6
  - Value:
    - u8...: MAC address in network byte order

Get board serial

- Tag: 0x00010004
- Request:
  - Length: 0
- Response:
  - Length: 8
  - Value:
    - u64: board serial

Get ARM memory

- Tag: 0x00010005
- Request:
  - Length: 0
- Response:
  - Length: 8
  - Value:
    - u32: base address in bytes
    - u32: size in bytes

Future formats may specify multiple base+size combinations.

Get VC memory

- Tag: 0x00010006
- Request:
  - Length: 0
- Response:
  - Length: 8
  - Value:
    - u32: base address in bytes
    - u32: size in bytes

Future formats may specify multiple base+size combinations.

Get clocks

- Tag: 0x00010007
- Request:
  - Length: 0
- Response:
  - Length: variable (multiple of 8)
  - Value:
    - u32: parent clock id (0 for a root clock)
    - u32: clock id
    - (repeated)

Returns all clocks that exist **in top down breadth first order**. Clocks that depend on another clock should be defined as children of that clock. Clocks that depend on no other clocks should have no parent. Clock IDs are as in the clock section below.

(swarren: This clock message is much less well defined than the other clock message below. To be useful, you'd need to either return a clock name along with each clock so the caller knew what they all were, or pre-define the list of valid clock IDs as the other clock message does below)

(lp0: I've now made it clear that the clock IDs are as below, which is why I reserved 0)

Config

Get command line

- Tag: 0x00050001
- Request:
  - Length: 0
- Response:
  - Length: variable
  - Value:
    - u8...: ASCII command line string

Caller should not assume the string is null terminated.

Shared resource management

Get DMA channels

- Tag: 0x00060001
- Request:
  - Length: 0
- Response:
  - Length: 4
  - Value:
    - u32: mask
  - Mask:
    - Bits 0-15: DMA channels 0-15 (0=do not use, 1=usable)
    - Bits 16-31: reserved for future use

Caller assumes that the VC has enabled all the usable DMA channels.

Power

Unique device IDs:

- 0x00000000: SD Card
- 0x00000001: UART0
- 0x00000002: UART1

- 0x00000003: USB HCD
- 0x00000004: I2C0
- 0x00000005: I2C1
- 0x00000006: I2C2
- 0x00000007: SPI
- 0x00000008: CCP2TX
- 0x00000009: Unknown (RPi4)
- 0x0000000a: Unknown (RPi4)

Get power state

- Tag: 0x00020001
- Request:
  - Length: 4
  - Value:
    - u32: device id
- Response:
  - Length: 8
  - Value:
    - u32: device id
    - u32: state
  - State:
    - Bit 0: 0=off, 1=on
    - Bit 1: 0=device exists, 1=device does not exist
    - Bits 2-31: reserved for future use

Response indicates current state.

Get timing

- Tag: 0x00020002
- Request:
  - Length: 4
  - Value:
    - u32: device id
- Response:
  - Length: 8
  - Value:
    - u32: device id
    - u32: enable wait time in microseconds

Response indicates wait time required after turning a device on before power is stable. Returns 0 wait time if the device does not exist.

Set power state

- Tag: 0x00028001
- Request:
    - Length: 8
    - Value:
        - u32: device id
        - u32: state
    - State:
        - Bit 0: 0=off, 1=on
        - Bit 1: 0=do not wait, 1=wait
        - Bits 2-31: reserved for future use (set to 0)
- Response:
    - Length: 8
    - Value:
        - u32: device id
        - u32: state
    - State:
        - Bit 0: 0=off, 1=on
        - Bit 1: 0=device exists, 1=device does not exist
        - Bits 2-31: reserved for future use

Response indicates new state, with/without waiting for the power to become stable.

Clocks

Unique clock IDs:

- 0x000000000: reserved
- 0x000000001: EMMC
- 0x000000002: UART
- 0x000000003: ARM
- 0x000000004: CORE
- 0x000000005: V3D
- 0x000000006: H264
- 0x000000007: ISP
- 0x000000008: SDRAM
- 0x000000009: PIXEL
- 0x00000000a: PWM
- 0x00000000b: HEVC
- 0x00000000c: EMMC2
- 0x00000000d: M2MC
- 0x00000000e: PIXEL_BVB

(swarren: I imagine there are more clocks than that; the clock message earlier returned clock parent information, and I doubt any of the clocks listed here are parents of each-other. At the very least I'd expect one more defined clock ID to represent the root crystal/PLL, and I'd expect there are a actually a variety of intermediate clocks between this and these peripherals).

All clocks are the **base clocks** for those peripherals, e.g. 3MHz for UART, 50/100MHz for EMMC, not the dividers applied using the peripheral.

Get clock state

- Tag: 0x00030001
- Request:
  - Length: 4
  - Value:
    - u32: clock id
- Response:
  - Length: 8
  - Value:
    - u32: clock id
    - u32: state
  - State:
    - Bit 0: 0=off, 1=on
    - Bit 1: 0=clock exists, 1=clock does not exist
    - Bits 2-31: reserved for future use

Set clock state

- Tag: 0x00038001
- Request:
  - Length: 8
  - Value:
    - u32: clock id
    - u32: state
  - State:
    - Bit 0: 0=off, 1=on
    - Bit 1: 0=clock exists, 1=clock does not exist
    - Bits 2-31: reserved for future use (set to 0)
- Response:
  - Length: 8
  - Value:
    - u32: clock id
    - u32: state
  - State:
    - Bit 0: 0=off, 1=on
    - Bit 1: 0=clock exists, 1=clock does not exist
    - Bits 2-31: reserved for future use

(swarren: it doesn't seem to make sense for the request to have a "clock exists" bit, only the response)

(lp0: the lack of a response makes it unclear if the tag itself was unsupported)

Get clock rate

- Tag: 0x00030002
- Request:
  - Length: 4
  - Value:
    - u32: clock id
- Response:
  - Length: 8
  - Value:
    - u32: clock id
    - u32: rate (in Hz)

Next enable rate should be returned even if the clock is not running. A rate of 0 is returned if the clock does not exist.

Get clock rate measured

- Tag: 0x00030047
- Request:
  - Length: 4
  - Value:
    - u32: clock id
- Response:
  - Length: 8
  - Value:
    - u32: clock id
    - u32: rate (in Hz)

Get the true/actual clock rate (instead of the last requested value returned by "Get clock rate"/0x0003002) which respects clamping, throttling and clock divider limitations.

Set clock rate

- Tag: 0x00038002
- Request:
  - Length: 12
  - Value:
    - u32: clock id
    - u32: rate (in Hz)
    - u32: skip setting turbo
- Response:
  - Length: 8

        - Value:
            - u32: clock id
            - u32: rate (in Hz)

Next supported enable rate should be returned even if the clock is not running. A rate of 0 is returned if the clock does not exist. The clock rate may be clamped to the supported range.

By default when setting arm freq above default, other turbo settings will be enabled (e.g. voltage, sdram and gpu frequencies). You can disable this effect by setting "skip setting turbo" to 1.

(swarren: As lp0 mentioned in the issue, min/max rates, and a list of valid parents would be useful response data from some message. Also, a set parent message might be useful if there are messages to get parenting information)

Get max clock rate

- Tag: 0x00030004
- Request:
    - Length: 4
    - Value:
        - u32: clock id
- Response:
    - Length: 8
    - Value:
        - u32: clock id
        - u32: rate (in Hz)

Return the maximum supported clock rate for the given clock. Clocks should not be set higher than this.

Get min clock rate

- Tag: 0x00030007
- Request:
    - Length: 4
    - Value:
        - u32: clock id
- Response:
    - Length: 8
    - Value:
        - u32: clock id
        - u32: rate (in Hz)

Return the minimum supported clock rate for the given clock. This may be used when idle.

Get turbo

- Tag: 0x00030009
- Request:

  - Length: 4
  - Value:
    - u32: id
- Response:
  - Length: 8
  - Value:
    - u32: id
    - u32: level

Get the turbo state for index id. id should be 0. level will be zero for non-turbo and one for turbo.

Set turbo

- Tag: 0x00038009
- Request:
  - Length: 8
  - Value:
    - u32: id
    - u32: level
- Response:
  - Length: 8
  - Value:
    - u32: id
    - u32: level

Set the turbo state for index id. id should be zero. level will be zero for non-turbo and one for turbo. This will cause GPU clocks to be set to maximum when enabled and minimum when disabled.

Voltage

Unique voltage IDs:

- 0x000000000: reserved
- 0x000000001: Core
- 0x000000002: SDRAM_C
- 0x000000003: SDRAM_P
- 0x000000004: SDRAM_I

Get voltage

- Tag: 0x00030003
- Request:
  - Length: 4
  - Value:
    - u32: voltage id
- Response:

    - Length: 8
    - Value:
        - u32: voltage id
        - u32: value (offset from 1.2V in units of 0.025V)

The voltage value may be clamped to the supported range. A value of 0x80000000 means the id was not valid.

Set voltage

- Tag: 0x00038003
- Request:
    - Length: 8
    - Value:
        - u32: voltage id
        - u32: value (offset from 1.2V in units of 0.025V)
- Response:
    - Length: 8
    - Value:
        - u32: voltage id
        - u32: value (offset from 1.2V in units of 0.025V)

The voltage value may be clamped to the supported range. A value of 0x80000000 means the id was not valid.

Get max voltage

- Tag: 0x00030005
- Request:
    - Length: 4
    - Value:
        - u32: voltage id
- Response:
    - Length: 8
    - Value:
        - u32: voltage id
        - u32: value (offset from 1.2V in units of 0.025V)

Return the maximum supported voltage rate for the given id. Voltages should not be set higher than this.

Get min voltage

- Tag: 0x00030008
- Request:
    - Length: 4
    - Value:
        - u32: voltage id

- Response:
  - Length: 8
  - Value:
    - u32: voltage id
    - u32: value (offset from 1.2V in units of 0.025V)

Return the minimum supported voltage rate for the given id. This may be used when idle.

Get temperature

- Tag: 0x00030006
- Request:
  - Length: 4
  - Value:
    - u32: temperature id
- Response:
  - Length: 8
  - Value:
    - u32: temperature id
    - u32: value

Return the temperature of the SoC in thousandths of a degree C. id should be zero.

Get max temperature

- Tag: 0x0003000a
- Request:
  - Length: 4
  - Value:
    - u32: temperature id
- Response:
  - Length: 8
  - Value:
    - u32: temperature id
    - u32: value

Return the maximum safe temperature of the SoC in thousandths of a degree C. id should be zero. Overclock may be disabled above this temperature.

Allocate Memory

- Tag: 0x0003000c
- Request:
  - Length: 12
  - Value:
    - u32: size
    - u32: alignment

        - u32: flags
- Response:
    - Length: 4
    - Value:
        - u32: handle

Allocates contiguous memory on the GPU. size and alignment are in bytes. flags contain:

```
MEM_FLAG_DISCARDABLE = 1 << 0, /* can be resized to 0 at any time. Use for cached data */
MEM_FLAG_NORMAL = 0 << 2, /* normal allocating alias. Don't use from ARM */
MEM_FLAG_DIRECT = 1 << 2, /* 0xC alias uncached */
MEM_FLAG_COHERENT = 2 << 2, /* 0x8 alias. Non-allocating in L2 but coherent */
MEM_FLAG_L1_NONALLOCATING = (MEM_FLAG_DIRECT | MEM_FLAG_COHERENT), /* Allocating in L2 */
MEM_FLAG_ZERO = 1 << 4,  /* initialise buffer to all zeros */
MEM_FLAG_NO_INIT = 1 << 5, /* don't initialise (default is initialise to all ones */
MEM_FLAG_HINT_PERMALOCK = 1 << 6, /* Likely to be locked for long periods of time. */
```

Lock memory

- Tag: 0x0003000d
- Request:
    - Length: 4
    - Value:
        - u32: handle
- Response:
    - Length: 4
    - Value:
        - u32: bus address

Lock buffer in place, and return a bus address. Must be done before memory can be accessed

Unlock memory

- Tag: 0x0003000e
- Request:
    - Length: 4
    - Value:
        - u32: handle
- Response:
    - Length: 4
    - Value:
        - u32: status

Unlock buffer. It retains contents, but may move. Needs to be locked before next use. status=0 is success.

Release Memory

- Tag: 0x0003000f
- Request:
  - Length: 4
  - Value:
    - u32: handle
- Response:
  - Length: 4
  - Value:
    - u32: status

Free the memory buffer. status=0 is success.

Execute Code

- Tag: 0x00030010
- Request:
  - Length: 28
  - Value:
    - u32: function pointer
    - u32: r0
    - u32: r1
    - u32: r2
    - u32: r3
    - u32: r4
    - u32: r5
- Response:
  - Length: 4
  - Value:
    - u32: r0

Calls the function at given (bus) address and with arguments given. E.g. r0 = fn(r0, r1, r2, r3, r4, r5); It blocks until call completes. The (GPU) instruction cache is implicitly flushed. Setting the lsb of function pointer address will suppress the instruction cache flush if you know the buffer hasn't changed since last execution.

Get Dispmanx Resource mem handle

- Tag: 0x00030014
- Request:
  - Length: 4
  - Value:
    - u32: dispmanx resource handle
- Response:
  - Length: 8
  - Value:
    - u32: 0 is successful

        - u32: mem handle for resource

Gets the mem_handle associated with a created dispmanx resource. This can be locked and the memory directly written from the arm to avoid having to copy the image data to GPU.

Get EDID block

- Tag: 0x00030020
- Request:
    - Length: 4
    - Value:
        - u32: block number
- Response:
    - Length: 136
    - Value:
        - u32: block number
        - u32: status
        - 128 bytes: EDID block

This reads the specified EDID block from attached HDMI/DVI device. There will always be at least one block of 128 bytes, but there may be additional blocks. You should keep requesting blocks (starting from 0) until the status returned is non-zero.

Frame Buffer

- All tags in the request are processed in one operation.
- It is not valid to mix Test tags with Get/Set tags in the same operation and no tags will be returned.
- Get tags will be processed after all Set tags.
- If an allocate buffer tag is omitted when setting parameters, then no change occurs unless it can be accommodated without changing the buffer base or size.
- When an allocate buffer response is returned, the old buffer area (if the base or size has changed) is implicitly freed.

For example:

1. The current values/defaults are loaded into a temporary struct
2. The tags are used to overwrite some or all of the values
3. Validation of Test/Set tags occurs
4. The Set changes are applied and responses based on the requested Get/Test/Set tags are written to the buffer

Duplicating the same tag in one request/response is prohibited. The expected result is either an error or implementation specified undefined behaviour (such as only using the last instance of the tag).

Allocate buffer

- Tag: 0x00040001
- Request:
  - Length: 4
  - Value:
    - u32: alignment in bytes
- Response:
  - Length: 8
  - Value:
    - u32: frame buffer base address in bytes
    - u32: frame buffer size in bytes

If the requested alignment is unsupported then the current base and size (which may be 0 if not allocated) is returned and no change occurs.

Release buffer

- Tag: 0x00048001
- Request:
  - Length: 0
- Response:
  - Length: 0

Releases and disables the frame buffer.

Blank screen

- Tag: 0x00040002
- Request:
  - Length: 4
  - Value:
    - u32: state
  - State:
  - Bit 0: 0=off, 1=on
  - Bits 1-31: reserved for future use (set to 0)
- Response:
  - Length: 4
  - Value:
    - u32: state
  - State:
  - Bit 0: 0=off, 1=on
  - Bits 1-31: reserved for future use

Get physical (display) width/height

Note that the "physical (display)" size is the size of the allocated buffer in memory, not the resolution of the video signal sent to the display device.

- Tag: 0x00040003
- Request:
  - Length: 0
- Response:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels

Test physical (display) width/height

- Tag: 0x00044003
- Request:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels
- Response:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set physical (display) width/height

- Tag: 0x00048003
- Request:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels
- Response:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels

The response may not be the same as the request so it must be checked. May be the previous width/height or 0 for unsupported.

Get virtual (buffer) width/height

Note that the "virtual (buffer)" size is the portion of buffer that is sent to the display device, not the resolution the buffer itself. This may be smaller than the allocated buffer size in order to implement panning.

- Tag: 0x00040004
- Request:
  - Length: 0
- Response:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels

Test virtual (buffer) width/height

- Tag: 0x00044004
- Request:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels
- Response:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set virtual (buffer) width/height

- Tag: 0x00048004
- Request:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels
- Response:
  - Length: 8
  - Value:
    - u32: width in pixels
    - u32: height in pixels

The response may not be the same as the request so it must be checked. May be the previous width/height or 0 for unsupported.

Get depth

- Tag: 0x00040005
- Request:
    - Length: 0
- Response:
    - Length: 4
    - Value:
        - u32: bits per pixel

Test depth

- Tag: 0x00044005
- Request:
    - Length: 4
    - Value:
        - u32: bits per pixel
- Response:
    - Length: 4
    - Value:
        - u32: bits per pixel

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set depth

- Tag: 0x00048005
- Request:
    - Length: 4
    - Value:
        - u32: bits per pixel
- Response:
    - Length: 4
    - Value:
        - u32: bits per pixel

The response may not be the same as the request so it must be checked. May be the previous depth or 0 for unsupported.

Get pixel order

- Tag: 0x00040006
- Request:
    - Length: 0
- Response:
    - Length: 4

    - Value:
        - u32: state
- State:
    - 0x0: BGR
    - 0x1: RGB

Test pixel order

- Tag: 0x00044006
- Request:
    - Length: 4
    - Value:
        - u32: state (as above)
- Response:
    - Length: 4
    - Value:
        - u32: state (as above)

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set pixel order

- Tag: 0x00048006
- Request:
    - Length: 4
    - Value:
        - u32: state (as above)
- Response:
    - Length: 4
    - Value:
        - u32: state (as above)

The response may not be the same as the request so it must be checked.

Get alpha mode

- Tag: 0x00040007
- Request:
    - Length: 0
- Response:
    - Length: 4
    - Value:
        - u32: state
- State: * 0x0: Alpha channel enabled (0 = fully opaque) * 0x1: Alpha channel reversed (0 = fully transparent) * 0x2: Alpha channel ignored

Test alpha mode

- Tag: 0x00044007
- Request:
    - Length: 4
    - Value:
        - u32: state (as above)
- Response:
    - Length: 4
    - Value:
        - u32: state (as above)

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set alpha mode

- Tag: 0x00048007
- Request:
    - Length: 4
    - Value:
        - u32: state (as above)
- Response:
    - Length: 4
    - Value:
        - u32: state (as above)

The response may not be the same as the request so it must be checked.

Get pitch

- Tag: 0x00040008
- Request:
    - Length: 0
- Response:
    - Length: 4
    - Value:
        - u32: bytes per line

Get virtual offset

- Tag: 0x00040009
- Request:
    - Length: 0
- Response:
    - Length: 8
    - Value:

        - u32: X in pixels
        - u32: Y in pixels

Test virtual offset

- Tag: 0x00044009
- Request:
    - Length: 8
    - Value:
        - u32: X in pixels
        - u32: Y in pixels
- Response:
    - Length: 8
    - Value:
        - u32: X in pixels
        - u32: Y in pixels

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set virtual offset

- Tag: 0x00048009
- Request:
    - Length: 8
    - Value:
        - u32: X in pixels
        - u32: Y in pixels
- Response:
    - Length: 8
    - Value:
        - u32: X in pixels
        - u32: Y in pixels

The response may not be the same as the request so it must be checked. May be the previous offset or 0 for unsupported.

Get overscan

- Tag: 0x0004000a
- Request:
    - Length: 0
- Response:
    - Length: 16
    - Value:
        - u32: top in pixels

            - u32: bottom in pixels
            - u32: left in pixels
            - u32: right in pixels

Test overscan

- Tag: 0x0004400a
- Request:
    - Length: 16
    - Value:
        - u32: top in pixels
        - u32: bottom in pixels
        - u32: left in pixels
        - u32: right in pixels
- Response:
    - Length: 16
    - Value:
        - u32: top in pixels
        - u32: bottom in pixels
        - u32: left in pixels
        - u32: right in pixels

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set overscan

- Tag: 0x0004800a
- Request:
    - Length: 16
    - Value:
        - u32: top in pixels
        - u32: bottom in pixels
        - u32: left in pixels
        - u32: right in pixels
- Response:
    - Length: 16
    - Value:
        - u32: top in pixels
        - u32: bottom in pixels
        - u32: left in pixels
        - u32: right in pixels

The response may not be the same as the request so it must be checked. May be the previous overscan or 0 for unsupported.

Get palette

- Tag: 0x0004000b
- Request:
  - Length: 0
- Response:
  - Length: 1024
  - Value:
    - u32...: RGBA palette values (index 0 to 255)

Test palette

- Tag: 0x0004400b
- Request:
  - Length: 24..1032
  - Value:
    - u32: offset: first palette index to set (0-255)
    - u32: length: number of palette entries to set (1-256)
    - u32...: RGBA palette values (offset to offset+length-1)
- Response:
  - Length: 4
  - Value:
    - u32: 0=valid, 1=invalid

Response is the same as the request (or modified), to indicate if this configuration is supported (in combination with all the other settings). Does not modify the current hardware or frame buffer state.

Set palette

- Tag: 0x0004800b
- Request:
  - Length: 24..1032
  - Value:
    - u32: offset: first palette index to set (0-255)
    - u32: length: number of palette entries to set (1-256)
    - u32...: RGBA palette values (offset to offset+length-1)
- Response:
  - Length: 4
  - Value:
    - u32: 0=valid, 1=invalid

The response may not be the same as the request so it must be checked. Palette changes should not be partially applied.

Set Cursor Info

- Tag: 0x00008010

- Request:
  - Length: 24
  - Value:
    - u32: width
    - u32: height
    - u32: (unused)
    - u32: pointer to pixels
    - u32: hotspotX
    - u32: hotspotY
- Response:
  - Length: 4
  - Value:
    - u32: 0=valid, 1=invalid

Format is 32bpp (ARGB). Width and height should be >= 16 and (width * height) <= 64.

Set Cursor State

- Tag: 0x00008011
- Request:
  - Length: 16
  - Value:
    - u32: enable (1=visible, 0=invisible)
    - u32: x
    - u32: y
    - u32: flags
- Response:
  - Length: 4
  - Value:
    - u32: 0=valid, 1=invalid

The flags control: bit0: clean=display coords, set=framebuffer coords

if Set Cursor Info hasn't been called a default cursor will be used (64x64 with hotspot at 0,0).