RADIANT Board Manager Documentation

P.S. Allison, 1/25/21

# Communications Protocol

The RADIANT is commanded primarily via a simple UART packet interface operating at 1 Mbaud (8 bits, no parity, 1 stop bit) with no flow control. From the user’s point of view, both the board manager *and* the FPGA are commanded via the same packets and same interface, with an address space split between the two.

Packets are Consistent Overhead Byte Stuffing (COBS) encoded using 0x00 as a packet delimiter, and consist of a 24-bit address+read/write indicator followed by either number of bytes requested (for reads) or data (for writes).

**Note**: all packets shown here are *before* COBS encoding. Packets are limited to a payload size of 254 bytes. With COBS encoding, this *guarantees* that all packets have a fixed size of payload+2 bytes, while *ensuring* that the packet delimiter (0x00) *always* indicates a packet boundary.

Read Format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **READ REQUEST** | | | | |
| *Byte 0* | | *Byte 1* | *Byte 2* | *Byte 3* |
| [7] | [6:0] | [7:0] | [7:0] | [7:0] |
| 1 | ADDR[22:0] | ADDR[15:8] | ADDR[7:0] | # of bytes requested – 1 |

Write Format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **WRITE REQUEST** | | | | |
| *Byte 0* | | *Byte 1* | *Byte 2* | *Byte 3..N (<253)* |
| [7] | [6:0] | [7:0] | [7:0] | [7:0] |
| 0 | ADDR[22:0] | ADDR[15:8] | ADDR[7:0] | DATA[0..N-3] |

In this way up to 250 bytes can be read or written at any given time.

*Typically* the data is processed in incrementing addresses: that is, a 4 byte read from address 0x00 will return address 0x00, 0x01, 0x02, 0x03. *Note*: see the “burst address mode” for clarification here.

## Addressing Notes

The address specified in read/write requests are *byte addresses*, and both the board manager and FPGA all have internal *32-bit little endian* address spaces. This means that *although the addresses are written big-endian* (which is needed to allow quick switching at the board manager level), the data *comes out* ***little-endian***. For example, a read to the IDENT register in the Board Manager returns, in order, 0x4D, 0x42, 0x44, 0x52.

## FPGA Burst Address Mode

Writes/reads to the FPGA are *typically* done with incrementing addresses: an 8-byte write to address 0x20 will write to 0x20-0x27, for instance. However, some operations on the FPGA require repeated writes to the *same address*. In this case, one can switch to *burst address mode*, which requires setting bit 3 in the CONTROL register of the **board manager**. Burst address mode works because for communication between the FPGA and Board Manager, ADDR[22] would always be 0.

Burst address mode consists of 3 separate “sub” modes which are controlled by the reset/mode register in the ***FPGA****.* ***Note the difference:***burst mode *overall* is controlled by a board manager register, whereas burst *addressing type* is controlled by the FPGA. This allows changing from burst<->normal access without any accesses to the FPGA.

* Burst byte address: address never changes (8 byte write to 0x20 writes 8 times to 0x20)
* Burst word address: address increments modulo 2 (8 byte write to 0x20 writes 2 bytes to 0x20/0x21 4 times)
* Burst dword address: address increments modulo 4 (8 bytes write to 0x20 writes 4 bytes to 0x20/0x21/0x22/0x23 2 times)

Switching between modes is detailed in the FPGA register space documentation of the RESET\_MODE register.

# Board Manager Register Space

The RADIANT Board Manager has a 32-bit register space accessed in the same way that the FPGA registers are accessed (via COBS-encoded packets with a 23-bit address + r/w bit, plus a number of bytes requested or data to be written). Board manager registers are accessed as an address space from 0x400000-0x7FFFFF.

Board manager registers should only be accessed at 32-bit offsets (e.g. register 0, 4, 8, 16, etc.), and it’s probably easier to just only do 4 byte reads + writes, although the firmware will not crash or break if non-32 bit values are requested/read.

|  |  |  |
| --- | --- | --- |
| **Address (Base: 0x400000)** | **Name** | **Description** |
| 0x0 | IDENT | “RDBM” (constant returned value: 0x5244424D) |
| 0x4 | DATEVERSION | Version info + date encoding |
| 0x8 | STATUS | Status register (state of various pins) |
| 0xC | CONTROL | Control register (control of some pins) |
| 0x10 | ANAV10 | Analog readback of 1.0V |
| 0x14 | ANAV18 | Analog readback of 1.8V |
| 0x18 | ANAV25 | Analog readback of 2.5V |
| 0x1C | ANALEFT | Analog readback of LEFT monitor (ch 0-11) |
| 0x20 | ANARIGHT | Analog readback of RIGHT monitor (ch 12-23) |
| 0x24 | SPIOUTLSB | Board SPI output (to attenuators) |
| 0x28 | SPIOUTMSB | Board SPI output (to sig generator) |
| 0x30-0x3C | reserved | Reserved |
| 0x40 | GPIO0 | Quad 0 GPIO |
| 0x44 | GPIO1 | Quad 1 GPIO |
| 0x48 | GPIO2 | Quad 2 GPIO |
| 0x4C | GPIO3 | Quad 3 GPIO |
| 0x50 | GPIO4 | Quad 4 GPIO |
| 0x54 | GPIO5 | Quad 5 GPIO |
| 0x58 | SIGGPIO | Signal generator GPIO |
| 0x5C-0x7C | reserved | reserved |
| 0x80 | TDBIAS0 | Trigger diode bias ch 0 |
| 0x84 | TDBIAS1 | Trigger diode bias ch 1 |
| 0x88 | TDBIAS2 | Trigger diode bias ch 2 |
| 0x8C | TDBIAS3 | Trigger diode bias ch 3 |
| 0x90 | TDBIAS4 | Trigger diode bias ch 4 |
| 0x94 | TDBIAS5 | Trigger diode bias ch 5 |
| 0x98 | TDBIAS6 | Trigger diode bias ch 6 |
| 0x9C | TDBIAS7 | Trigger diode bias ch 7 |
| 0xA0 | TDBIAS8 | Trigger diode bias ch 8 |
| 0xA4 | TDBIAS9 | Trigger diode bias ch 9 |
| 0xA8 | TDBIAS10 | Trigger diode bias ch 10 |
| 0xAC | TDBIAS11 | Trigger diode bias ch 11 |
| 0xB0 | TDBIAS12 | Trigger diode bias ch 12 |
| 0xB4 | TDBIAS13 | Trigger diode bias ch 13 |
| 0xB8 | TDBIAS14 | Trigger diode bias ch 14 |
| 0xBC | TDBIAS15 | Trigger diode bias ch 15 |
| 0xC0 | TDBIAS16 | Trigger diode bias ch 16 |
| 0xC4 | TDBIAS17 | Trigger diode bias ch 17 |
| 0xC8 | TDBIAS18 | Trigger diode bias ch 18 |
| 0xCC | TDBIAS19 | Trigger diode bias ch 19 |
| 0xD0 | TDBIAS20 | Trigger diode bias ch 20 |
| 0xD4 | TDBIAS21 | Trigger diode bias ch 21 |
| 0xD8 | TDBIAS22 | Trigger diode bias ch 22 |
| 0xDC | TDBIAS23 | Trigger diode bias ch 23 |
| 0xE0 | VPEDLEFT | Pedestal voltage for ch0-11 |
| 0xE4 | VPEDRIGHT | Pedestal voltage for ch12-23 |

# Register descriptions

## 0x08: STATUS register

The STATUS register covers several status input pins. The bit breakdown is

* Bit 0: FPGA\_DONE (1 if FPGA configuration is complete)
* Bit 1: /MGTDET (0 if an MGT connection is detected – note this is the connector next to the FPGA)
* Bit 2: SD\_DETECT (1 if a micro-SD card is inserted, 0 otherwise)
* Bit 3: PG1V0 (Power Good 1.0V)
* Bit 4: PG1V8 (Power Good 1.8V)
* Bit 5: PG2V5 (Power Good 2.5V)
* Bit 6: PG2V6 (Power Good 2.6V)
* Bit 7: PG3V1 (Power Good 3.1V)

## 0x0C: CONTROL register

The CONTROL register covers several control outputs. These are mostly unused at the moment, but the used bits are:

* Bit 2: /BM\_EN\_10MHz (1 to *disable* the onboard 10 MHz clock)
* Bit 3: BURST bit for FPGA-bound packets (if 1, the BURST bit gets set for outgoing FPGA packets and the address *does not increment* in the same way)

## 0x10-0x20: Analog readbacks

These values are all analog inputs to the Board Manager. They are 16-bit values referenced to 3.3V (e.g. the value is val\*3.3V/65535).

## 0x40-0x54: Quad GPIOs

These are the I2C GPIOs for each quad (group of channels). Quad 0 controls channels 0-3, quad 1 controls channels 4-7, etc.

There are 8 controllable bits in each register. The bits are:

* Bit 0: SEL\_CAL (set to 1 to enable calibration for this quad)
* Bit 1: ATT\_LE (set to 1 to drive LE – latch enable - for attenuators in this quad)
* Bit 2: BIST (set to 1 to enable built-in self-test mode for this quad)
* Bit 3: green/red LED (set to 1 to drive green LED, set to 0 to drive red LED)
* Bit 4: TRIG\_EN (enable 3.0V trigger voltage)
* Bit 5: LAB\_EN (enable 2.6V LAB voltage)
* Bit 6: DIP switch bit 0 (controls TRIG\_EN at startup)
* Bit 7: DIP switch bit 1 (controls LAB\_EN at startup)

Bits 6 and 7 cannot be modified (read-only). All others are read-write.

### GPIO Usage

There are several ways to use the quad GPIOs. Specifically, the BIST bit enables driving the analog monitors from the LABs/quads for use in self-testing. This usage will not be covered here (always keep BIST bit 0).

The SEL\_CAL bit switches all 4 channels in a quad to CALIBRATION mode. Only 1 quad on each side (e.g. one quad of 0, 1, 2 and one quad of 3, 4, 5) can be switched into calibration mode at a time. In calibration mode, either an FPGA-derived impulse or an approximately sine-wave signal is fed to each LAB instead of the RF input (depending on the state of the CAL\_PULSE pins in the SIGGPIO).

The ATT\_LE bit drives the LE (latch enable) for all 8 attenuators attached to this quad. To set a specific attenuator:

* Write control word (A[7:0],D[7:0]) to SPIOUTLSB
* Set ATT\_LE to 1
* Set ATT\_LE to 0

A[7:0] here is the address of the attenuator – only A[2:0] are used. Attenuator addresses are 0, 2, 4, 6 = signal attenuators for quad ch 0, 1, 2, 3 (e.g. channel % 4) and 1, 3, 5, 7 are trigger attenuators for channels 0, 1, 2, 3 (again channel % 4).

**Example**: to set the trigger attenuator for channel 14 (which is the 2nd channel on quad 3, so attenuator address 5) to 50, you would:

* Write 0x532 to SPIOUTLSB. Here 0x5 is the attenuator address, and 0x32 (50) is the setting.
* Read GPIO3 to ‘val’
* Write “val | 0x2” to GPIO3
* Write “val” to GPIO3

**Note:** SPIOUTLSB only uses the low 16 bits of the word written.

## 0x58: SIGGPIO

This is the GPIO connected to the test signal generation section. The bits are

* Bit 0: CAL\_FIL0 (0th bit of test signal filter selection)
* Bit 1: SIG\_LE (Latch enable for the signal generator)
* Bit 2: CAL\_FIL1 (1st bit of test signal filter selection)
* Bit 3: /CAL\_FIL1 (*must be* opposite of bit 2)
* Bit 4: CAL\_PULSE (1 if test signal = pulse, 0 if test signal = sine wave)
* Bit 5: /CAL\_PULSE (*must be* opposite of bit 4)
* Bit 6: SG\_ENABLE (1 if the signal generator is powered on)
* Bit 7: SG\_MUXOUT (read-only MUXOUT bit from signal generator)

The signal generator is an ADF4351, which generates a variable-amplitude, variable frequency output from a reference clock. Programming the ADF4351 is not covered here – Python scripts exist to configure the signal generator.

The ADF4351 is connected to the same SPI bus used for programming the attenuators, but it uses the SPIOUTMSB register. **Note**: SPIOUTMSB uses *all 32 bits* of the word written.

The “CAL\_FIL” bits select which banks are used to filter the sine wave. **Only** use one of the settings below.

|  |  |  |  |
| --- | --- | --- | --- |
| **/CAL\_FIL1** | **CAL\_FIL1** | **CAL\_FIL0** | Frequency Band |
| 1 | 0 | 0 | 50-100 MHz |
| 1 | 0 | 1 | 100-300 MHz |
| 0 | 1 | 0 | 300-600 MHz |
| 0 | 1 | 1 | 600 MHz+ |

## 0x80-0xDC: Trigger DAC outputs

These registers control the trigger diode bias voltage (should be ~1.2V). These registers are 12-bit values ranging from 0-2.0V. That is, a value of 1.2V would be ~2500.

## 0xE0-0xE4: Pedestal DAC outputs

These registers control the LAB4 pedestal outputs. These registers are 12-bit values ranging from 0-3.3V. That is, a value of 1.2V would be ~1500. **Note:** the default pedestal should probably be like, 0.8V or something. I think we used 0.74V in the LAB4D paper.