Efabless Ravenna PicoRV32 SoC

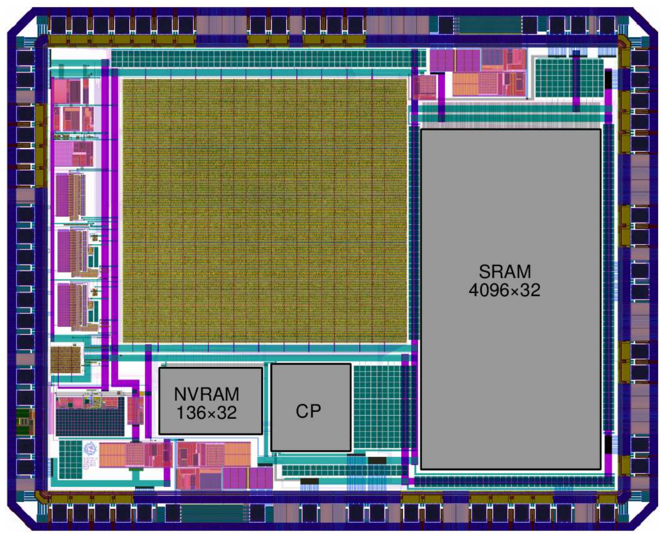
Description:

The efabless Ravenna PicoRV32 is a small RISC-V microprocessor based on the simple 2-cycle PicoRV32 RISC-V core implementing the RV32IMC instruction set (see http://riscv.org/)

Core:

The processor core is the PicoRV32 design (see http://github.com/cliffordwolf/picorv32). The full core description is available from the github site. The hardware implementation is the "large" variant, incorporating options IRQ, MUL, DIV, BARREL_SHIFTER, and COMPRESSED_ISA (16-bit instructions).

Core clock rate: 80 MHz maximum over all conditions.



Ravenna PicoRV32 SoC die (2.573mm × 2.068mm)

Features:

The SoC design incorporates on-board analog functions, including the following:

- 2 10-bit SAR ADCs
- 1 10-bit DAC
- 1 analog comparator
- 1 100kHz RC oscillator
- 1 1.235V bandgap reference
- 1 high temperature alarm

Digital functions/features of the SoC include:

- 1 QSPI flash controller
- 1 UART
- 1 SPI master
- 1 I^2C master
- 1 counter-timer
- 16 general-purpose digital input/output channels
- 4k word (4096 bytes × 32 bits) on−board SRAM
- 136 (128) word (128 bytes × 32 bits) on−board NVRAM

Process:

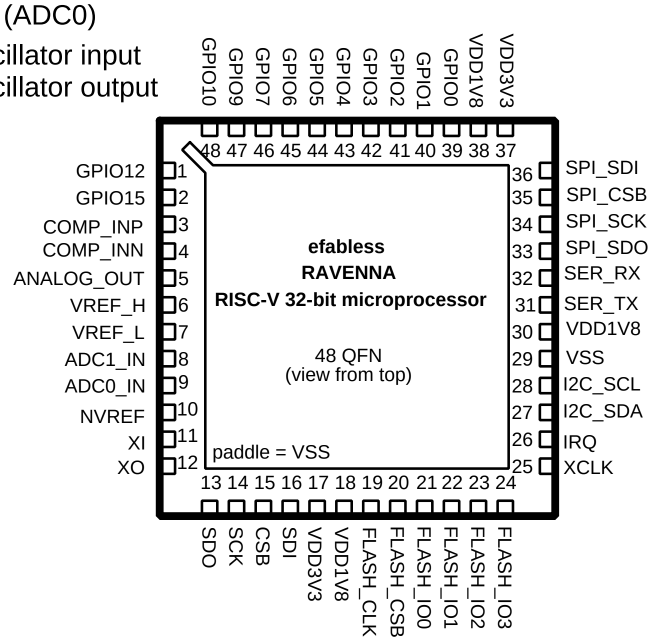
The efabless Ravenna PicoRV32 is fabricated in X-Fab 0.18µm CMOS technology.

Version:

This document corresponds to version 1 of the Ravenna processor (November 2020).

Pin Description (48QFN)

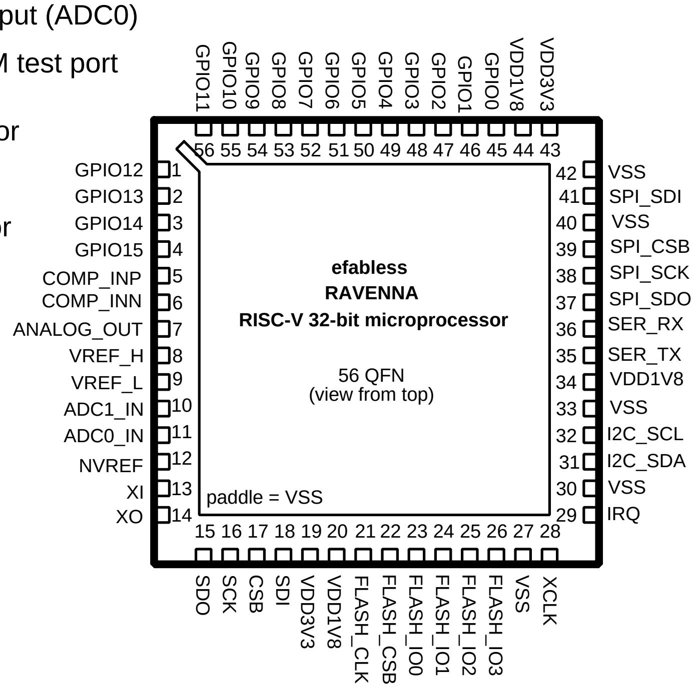
Pin #	*Name*	*Type*	*Summary description*
39–48 1–2	GPIO0–7,9,10 GPIO12,15	Digital I/O	General purpose configurable digital I/O with pullup/pulldown, input or output, and enable/disable.
13	SDO	Digital out	Housekeeping serial interface data output
14	SCK	Digital in	Housekeeping serial interface clock
15	CSB	Digital in	Housekeeping serial interface chip select
16	SDI	Digital in	Housekeeping serial interface data input
19	FLASH_CLK	Digital out	QSPI clock
20	FLASH_CSB	Digital out	QSPI chip select
21–24	FLASH_IO0–3	Digital I/O	QSPI flash bidirectional data input/output
25	XCLK	Digital in	External CMOS 3.3V optional clock source
26	IRQ	Digital in	External interrupt
27	I2C_SDA	Digital out	I^2C master data channel
28	I2C_SCL	Digital in	I^2C master clock channel
31	SER_TX	Digital out	UART transmit channel
32	SER_RX	Digital in	UART receive channel
33	SPI_SDO	Digital out	Serial interface master data output
34	SPI_SCK	Digital out	Serial interface master clock
35	SPI_CSB	Digital out	Serial interface master chip select
36	SPI_SDI	Digital in	Serial interface masterdata input
3	COMP_INP	Analog in	Comparator positive input
4	COMP_INN	Analog in	Comparator negative input
5	ANALOG_OUT	Analog out	Selectable bandgap or DAC output
6	VREF_H	Analog in	ADC and DAC voltage reference (high)
7	VREF_L	Analog in	ADC and DAC voltage reference (low)
8	ADC1_IN	Analog in	ADC input (ADC1)
9	ADC0_IN	Analog in	ADC input (ADC0)
11	XI	Crystal in	Crystal oscillator input
12	XO	Crystal out	Crystal oscillator output
17,37	VDD3V3	3.3V Power (in)	
18,30 38	VDD1V8	1.8V Power (out)	
29 paddle	VSS	Ground	



Standard package:	QFN48
Package size:	7 mm × 7 mm
Pin pitch:	0.5 mm
Paddle size:	5.63 mm × 5.63 mm

Pin Description (56QFN)

Pin #	*Name*	*Type*	*Summary description*
45–56 1–4	GPIO0–11 GPIO12–15	Digital I/O	General purpose configurable digital I/O with pullup/pulldown, input or output, and enable/disable.
15	SDO	Digital out	Housekeeping serial interface data output
16	SCK	Digital in	Housekeeping serial interface clock
17	CSB	Digital in	Housekeeping serial interface chip select
18	SDI	Digital in	Housekeeping serial interface data input
21	FLASH_CLK	Digital out	QSPI clock
22	FLASH_CSB	Digital out	QSPI chip select
23–26	FLASH_IO0–3	Digital I/O	QSPI flash bidirectional data input/output
28	XCLK	Digital in	External CMOS 3.3V optional clock source
29	IRQ	Digital in	External interrupt
31	I2C_SDA	Digital out	I^2C master data channel
32	I2C_SCL	Digital in	I^2C master clock channel
35	SER_TX	Digital out	UART transmit channel
36	SER_RX	Digital in	UART receive channel
37	SPI_SDO	Digital out	Serial interface master data output
38	SPI_SCK	Digital out	Serial interface master clock
39	SPI_CSB	Digital out	Serial interface master chip select
41	SPI_SDI	Digital in	Serial interface masterdata input
5	COMP_INP	Analog in	Comparator positive input
6	COMP_INN	Analog in	Comparator negative input
7	ANALOG_OUT	Analog out	Selectable bandgap or DAC output
8	VREF_H	Analog in	ADC and DAC voltage reference (high)
9	VREF_L	Analog in	ADC and DAC voltage reference (low)
10	ADC1_IN	Analog in	ADC input (ADC1)
11	ADC0_IN	Analog in	ADC input (ADC0)
12	NVREF	Analog out	NVRAM test port
13	XI	Crystal in	Crystal oscillator input
14	XO	Crystal out	Crystal oscillator output
19,43	VDD3V3	3.3V Power (in)	
20,34 44	VDD1V8	1.8V Power (out)	
27,30 40,42 paddle	VSS	Ground	

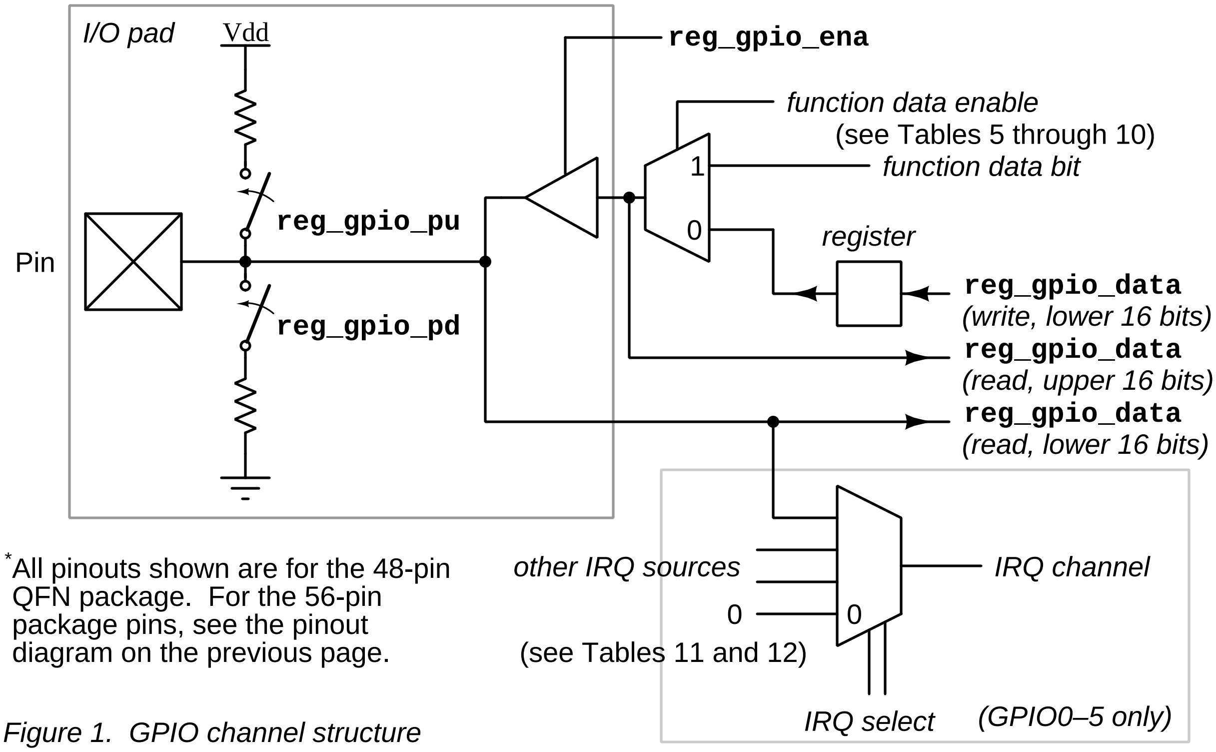


Standard package:	QFN56
Package size:	8 mm × 8 mm
Pin pitch:	0.5 mm
Paddle size:	6.63 mm × 6.63 mm

General Purpose I/O

GPIO0 to GPIO15 (pins 39–48 and 1–2)*

The GPIO pins are sixteen assignable digital inputs or outputs. The basic function of each GPIO is illustrated below. All writes to **`reg_gpio_data`** are registered. All reads from **`reg_gpio_data`** are immediate.



*All pinouts shown are for the 48-pin QFN package. For the 56-pin package pins, see the pinout diagram on the previous page.

Figure 1. GPIO channel structure

GPIO memory address map:

C header name	*address*	*description*
`reg_gpio_data`	**`0x03000000`**	GPIO input/output (lower 16 bits) GPIO output readback (upper 16 bits)
`reg_gpio_enb`	**`0x03000004`**	GPIO output enable (0 = output, 1 = input)
`reg_gpio_pub`	**`0x03000008`**	GPIO pullup enable (0 = pullup, 1 = none)
`reg_gpio_pdb`	**`0x0300000c`**	GPIO pulldown enable (0 = pulldown, 1 = none)
`reg_comp_out_dest`	**`0x0300006c`**	Comparator output destination (low 2 bits)
`reg_rcosc_out_dest`	**`0x03000074`**	RC oscillator output destination (low 2 bits)
`reg_xtal_out_dest`	**`0x030000a0`**	Crystal output destination (low 2 bits)
`reg_pll_out_dest`	**`0x030000a4`**	PLL clock output destination (low 2 bits)
`reg_trap_out_dest`	**`0x030000a8`**	Trap output destination (low 2 bits)
`reg_irq7_source`	**`0x030000b0`**	IRQ 7 input source (low 2 bits)
`reg_irq8_source`	**`0x030000b4`**	IRQ 8 input source (low 2 bits)
`reg_overtemp_out_dest`	**`0x030000e8`**	Over-temperature destination (low 2 bits)

GPIO description, continued.

In the memory-mapped register descriptions below, each register is shown as 32 bits corresponding to the data bus width of the PicoRV32 processor. Addresses, however, are in bytes. Depending on the instruction and data type, the entire 32-bit register can be read in one instruction, or one 16-bit word, or one 8-bit byte.

Table 1 **reg_gpio_data**

0x03000003	0x03000002	0x03000001	0x03000000	address
GPIO output readback		GPIO input/output		value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bits 0 to 15 and bits 16 to 31 correspond to GPIO channels 0 to 15, respectively.
Writing to the address low 16 bits always sets the registered value at the GPIO.
Writing to the address high 16 bits has no effect.
Reading from the address low 16 bits reads the value at the corresponding chip pin.
Reading from the address high 16 bits reads the value at the multiplexer output (see diagram).

Table 2 **reg_gpio_enb**

0x03000007	0x03000006	0x03000005	0x03000004	address
(undefined, reads zero)		GPIO output enable (inverted)		value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bits 0 to 15 correspond to GPIO channels 0 to 15, respectively.
Output enable is sense inverted. Bit value 0 indicates an output channel; 1 indicates an input.

Table 3 **reg_gpio_pub**

0x0300000b	0x0300000a	0x03000009	0x03000008	address
(undefined, reads zero)		GPIO pin pull-up (inverted)		value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bits 0 to 15 correspond to GPIO channels 0 to 15, respectively.
Pullup enable is sense inverted. Bit value 0 indicates pullup is active; 1 indicates pullup inactive.

Table 4 **reg_gpio_pdb**

0x0300000f	0x0300000e	0x0300000d	0x0300000c	address
(undefined, reads zero)		GPIO pin pull-down (inverted)		value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bits 0 to 15 correspond to GPIO channels 0 to 15, respectively.
Pulldown enable is sense inverted. Bit value 0 indicates pullup is active; 1 indicates pulldown is inactive.

GPIO description, continued.

Table 5 **`reg_comp_out_dest`**

`0x0300006f`	`0x0300006e`	`0x0300006d`	**`0x0300006c`**	address
(undefined, reads zero)			comparator dest.	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The two low bits of this register direct the output of the comparator to one of two GPIO channels or IRQ (interrupt) channel, according to the following table:

Register byte **`0x0300006c`** value		Comparator output directed to this channel
0	**`00`**	(none)
1	**`01`**	GPIO0
2	**`10`**	GPIO1
3	**`11`**	IRQ9

Table 6 **`reg_rcosc_out_dest`**

`0x03000077`	`0x03000076`	`0x03000075`	**`0x03000074`**	address
(undefined, reads zero)			RC osc. destination	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The two low bits of this register direct the output of the RC oscillator to one of three GPIO channels, according to the following table:

Register byte **`0x03000074`** value		RC oscillator output directed to this channel
0	**`00`**	(none)
1	**`01`**	GPIO2
2	**`10`**	GPIO3
3	**`11`**	GPIO4

Table 7 **`reg_xtal_out_dest`**

`0x030000a3`	`0x030000a2`	`0x030000a1`	**`0x030000a0`**	address
(undefined, reads zero)			crystal osc. dest.	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The two low bits of this register direct the output of the crystal oscillator to one of three GPIO channels, according to the following table:

Register byte **`0x030000a0`** value		Crystal oscillator output directed to this channel
0	**`00`**	(none)
1	**`01`**	GPIO5
2	**`10`**	GPIO6
3	**`11`**	GPIO7

GPIO description, continued.

Table 8 **`reg_pll_out_dest`**

0x030000a7	0x030000a6	0x030000a5	**0x030000a4**	address
(undefined, reads zero)			PLL clock dest.	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The two low bits of this register direct the output of the core clock to one of three GPIO channels, according to the following table:

Register byte **0x030000a4** value		Clock output directed to this channel
0	**00**	(none)
1	**01**	Core PLL clock to GPIO8
2	**10**	Selected clock (PLL or XCLK) to GPIO9*
3	**11**	NVram test clock to GPIO10

Note that a high rate core clock (*e.g.,* 80 MHz) may be unable to generate a full swing on the GPIO outputs.

Table 9 **`reg_trap_out_dest`**

0x030000ab	0x030000aa	0x030000a9	**0x030000a8**	address
(undefined, reads zero)			trap signal dest.	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The two low bits of this register direct the output of the crystal oscillator to one of three GPIO channels, according to the following table:

Register byte **0x030000a8** value		Trap signal output directed to this channel
0	**00**	(none)
1	**01**	GPIO11*
2	**10**	GPIO12
3	**11**	GPIO13*

Table 10 **`reg_overtemp_out_dest`**

0x030000eb	0x030000ea	0x030000e9	**0x030000e8**	address
(undefined, reads zero)			over-temp dest.	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The two low bits of this register direct the output of the over-temperature alarm to one of two GPIO channels or an IRQ channel, according to the following table:

Register byte **0x030000e8** value		Over-temperature alarm directed to this channel
0	**00**	(none)
1	**01**	GPIO14*
2	**10**	GPIO15
3	**11**	IRQ10

**(note: not available on the 48 pin package)*

GPIO description, continued.

Table 11 **reg_irq7_source**

0x030000b3								0x030000b2								0x030000b1								**0x030000b0**								address
(undefined, reads zero)																								IRQ 7 source								value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The two low bits of this register direct the input of the selected GPIO channel to the processor's IRQ7 channel:

Register byte **0x030000b0** value		This channel directed to IRQ channel 7
0	**00**	(none)
1	**01**	GPIO0
2	**10**	GPIO1
3	**11**	GPIO2

Table 12 **reg_irq8_source**

0x030000b7								0x030000b6								0x030000b5								**0x030000b4**								address
(undefined, reads zero)																								IRQ 8 source								value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The two low bits of this register direct the input of the selected GPIO channel to the processor's IRQ8 channel:

Register byte **0x030000b4** value		This channel directed to IRQ channel 8
0	**00**	(none)
1	**01**	GPIO3
2	**10**	GPIO4
3	**11**	GPIO5

Housekeeping SPI SDI (pin 16), CSB (pin 15), SCK (pin 14), and SDO (pin 13)

The “housekeeping” SPI is an SPI slave that can be accessed from a remote host through a standard 4-pin serial interface. The SPI implementation is mode 0, with new data on SDI captured on the SCK rising edge, and output data presented on the falling edge of SCK (to be sampled on the next SCK rising edge).

SPI protocol definition

All input is in groups of 8 bits. Each byte is input msb first.

Every command sequence requires one command word (8 bits) followed by one address word (8 bits) followed by one or more data words (8 bits each), according to the data transfer modes defined below.

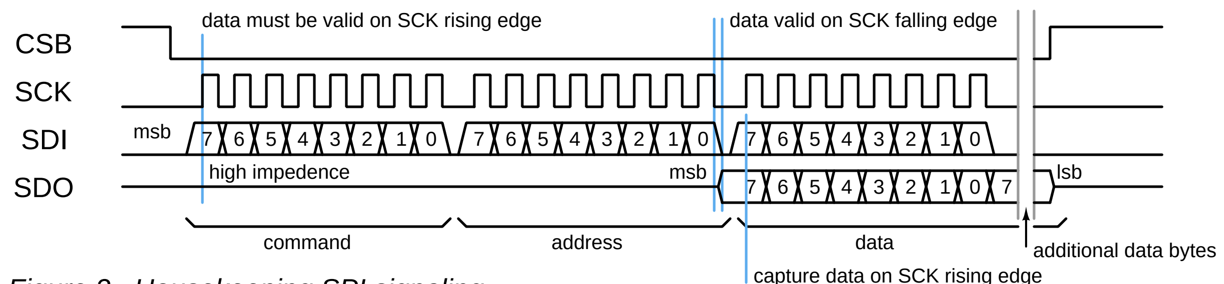


Figure 2. Housekeeping SPI signaling

Addresses are read in sequence from lower values to higher values.

Therefore groups of bits larger than 8 should be grouped such that the lowest bits are at the highest address. Any bits additional to an 8-bit boundary should be at the lowest address.

Data are captured from the register map in bytes on the falling edge of the last SCK before a data byte transfer. Multi-byte transfers should ensure that data do not change between byte reads.

CSB pin must be low to enable an SPI transmission. Data are clocked by pin SCK, with data valid on the rising edge of SCK. Output data are received on the SDO line. SDO is held high-impedance when CSB is high and at all times other than the transfer of data bits on a read command. SDO outputs become active on the falling edge of SCK, such that data are written and read on the same SCK rising edge.

After CSB is set low, the SPI is always in the "command" state, awaiting a new command.

The first transferred byte is the command word, interpreted according to Table 1 below.

Table 13 Housekeeping SPI command word definition

`00000000`	No operation
`10000000`	Write in streaming mode
`01000000`	Read in streaming mode
`11000000`	Simultaneous Read/Write in streaming mode
`11000100`	Pass-through Read/Write in streaming mode
`10nnn000`	Write in n-byte mode (up to 7 bytes).
`01nnn000`	Read in n-byte mode (up to 7 bytes).
`11nnn000`	Simultaneous Read/Write in n-byte mode (up to 7 bytes).

All other words are reserved and act as no-operation if not defined by the SPI slave module.

SPI protocol definition *(continued)*

The two basic modes of operation are "streaming mode" and "n-byte mode". In "streaming mode" operation, data are sent or received continuously, one byte at a time, with the internal address incrementing for each byte. Streaming mode operation continues until CSB is raised to end the transfer.

In "n-byte mode" operation, the number of bytes to be read and/or written is encoded in the command word, and may have a value from 1 to 7 (note that a value of zero implies streaming mode). After n bytes have been read and/or written, the SPI returns to waiting for the next command. No toggling of CSB is required to end the command or to initiate the following command.

Pass-thru mode

The pass-thru mode puts the CPU into immediate reset, then sets FLASH_CSB low to initiate a data transfer to the QSPI flash. After the pass-thru command byte has been issued, all subsequent SPI signaling on SDI and SCK are applied directly to the QSPI flash (pins FLASH_IO0 and FLASH_CLK, respectively), and the QSPI flash data output (pin FLASH_IO1) is applied directly to SDO, until the CSB pin is raised. When CSB is raised, the FLASH_CSB is also raised, terminating the data transfer to the QSPI flash. The CPU is brought out of reset, and starts executing instructions at the program start address.

This mode allows the QSPI flash to be programmed from the same SPI communication channel as the housekeeping SPI, without the need for additional wiring to the QSPI flash chip.

Housekeeping SPI registers

The purpose of the housekeeping SPI is to allow access and modification to values that normally could put the CPU into an unusable state, such as the CPU clock enable, the enable for the crystal oscillator generating the CPU clock, the enable and trim for the PLL that multiplies up the crystal oscillator clock, and the enable for the voltage regulators that generate the CPU's 1.8V supply. The housekeeping SPI runs at 3.3V and so is independent of the 1.8V on-board regulated supply.

Under normal working conditions, the SPI should not need to be accessed unless it is to adjust the clock speed of the CPU. All other functions are purely for test and debug.

All values in the SPI registers are exported to the CPU and can be viewed (read-only) in memory-mapped space (see the memory map documentation).

mask revision register address 0x1 high 4 bits
The 4-bit mask revision for version 1 of the Ravenna chip is 0x0.

manufacturer_ID register address 0x1 low 4 bits and register address 0x2
The 12-bit manufacturer ID for efabless is 0x456

product_ID register address 0x3
The product ID for the Ravenna PicoRV32 is 0x03

xtal osc. enable register address 0x9 bit 0
The crystal oscillator drives the CPU clock, so it cannot be turned on and off by the CPU. By default it is enabled (value 0x1).

1.8V regulator enable register address 0x9 bit 1
The output of the regulator generates the power supply for the CPU, and so it cannot be turned on and off by the CPU. By default it is enabled (value 0x1).

Housekeeping SPI registers *(continued)*

PLL VCO enable register address 0x4 bit 1

The PLL multiplies the crystal oscillator frequency up by a factor of 8. When the VCO is disabled, the PLL is disabled and the CPU clock stops. Default is 0x1 (enabled).

PLL CP enable register address 0x4 bit 2

When the PLL charge pump is disabled but the VCO remains enabled, the VCO is in a free-running mode. The VCO voltage input is connected to pin COMP_INP and adjustible through the range of 0 to VDD3V3 (nominally 3.3V). In addition, the PLL frequency can be trimmed. Default is 0x1 (enabled).

PLL CP bias enable register address 0x4 bit 0

This is a separate control for the current biasing to the PLL charge pump and VCO. Because it biases both, it should remain on at all times regardless of the enable state of the CP or VCO. It should only be disabled when both the CP and VCO are disabled. Default is 0x1 (enabled).

PLL trim register address 0x4 bits 3 through 6

The 4-bit trim value can adjust the VCO frequency over a factor of about four from the slowest (trim value 0x0) to the fastest (trim value 0xf (15)). Default value is 0x0 (slow trim).

PLL bypass register address 0x5 bit 0

When enabled, the PLL bypass switches the clock source of the CPU from the PLL output to the external CMOS clock (pin XCLK). Switching is not guaranteed glitch-free. The default value is 0x0 (CPU clock source is the PLL output).

CPU IRQ register address 0x6 bit 0

This is a dedicated manual interrupt driving the CPU IRQ channel 6. The bit is not self-resetting, so while the rising edge will trigger an interrupt, the signal must be manually set to zero before it can trigger another interrupt.

CPU reset register address 0x7 bit 0

The CPU reset bit puts the entire CPU into a reset state. This will also put analog functions into a disabled state, where analog blocks have enable/disable control. This bit is not self-resetting and must be set back to zero manually to clear the reset state.

CPU trap register address 0x8 bit 0

If the CPU has stopped after encountering an error, it will raise the trap signal. The trap signal can be configured to be read from a GPIO pin, but as the GPIO state is potentially unknowable, the housekeeping SPI can be used to determine the true trap state.

tm_nvcp register address 0xa lower four bits

This 4-bit vector assigns a test mode to the NVRAM for calibration. See the NVRAM calibration section for details.

Housekeeping SPI registers *(continued)*

Table 14 **Housekeeping SPI register map**

Register Address	*msb* 7	6	5	4	3	2	1	*lsb* 0	*comments*
0x00	SPI status and control								unused/ undefined
0x01	mask revision (= 0x0)				manufacturer_ID[11:8] (= 0x4)				read-only
0x02	manufacturer_ID[7:0] (= 0x56)								read-only
0x03	product_ID (= 0x03)								read-only
0x04	unused	PLL trim[3:0]				PLL CP enable	PLL VCO enable	PLL bias enable	default 0x07
0x05	unused							PLL bypass	default 0x00
0x06	unused							CPU IRQ	default 0x00
0x07	unused							CPU reset	default 0x00
0x08	unused							CPU trap	read-only
0x09	unused						1.8V regulator enable	xtal osc. enable	default 0x03
0x0a	unused				tm_nvcp[3:0]				default 0x00
0x0b	undefined								all undefined registers read 0x00

QSPI Flash interface FLASH_IO0–3 (pins 21 to 24), FLASH_CSB (pin 20), and FLASH_CLK (pin 19)

The QSPI flash controller is automatically enabled on power-up, and will immediately initiate a read sequence in single-bit mode with pin FLASH_IO0 acting as SDI (data from flash to CPU) and pin FLASH_IO1 acting as SDO (data from CPU to flash). Protocol is according to, *e.g.,* Cypress S25FL256L.

The initial SPI instruction sequence is as follows:

`0xFF`	Mode bit reset
`0xAB`	Release from deep power-down
`0x03`	Read w/3 byte address
`0x10`	Program start address (**`0x00100000`**) (3 bytes)
`0x00`	
`0x00`	

The QSPI flash continues to read bytes, either sequentially on the same command, or issuing a new read command to read from a new address.

The behavior of the QSPI flash controller can be modified by changing values in the register below:

Table 15 **`reg_spictrl`**

<table>
<tr><td colspan="8">0x02000003</td><td colspan="8">0x02000002</td><td colspan="8">0x02000001</td><td colspan="8">0x02000000</td><td>address</td></tr>
<tr><td></td><td colspan="8">(unused)</td><td colspan="7">(see below)</td><td colspan="4">(unused)</td><td colspan="4"></td><td colspan="2"></td><td colspan="6">(see below)</td><td>value</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>bit</td></tr>
</table>

mask bit	default	description
31	1	QSPI flash interface enable
22–20	0	Access mode (see table below)
19–16	8	Dummy clock cycle count
11–8	0	Bit-bang OE FLASH_IO3–FLASH_IO0
5	0	Bit-bang FLASH_CSB
4	0	Bit-bang FLASH_CLK
3–0	0	Bit-bang value FLASH_IO3–FLASH_IO0

Access mode bit selection (bits 22–20):

0	**`000`**	Single bit per clock
1	**`001`**	Single bit per clock (same as 0)
2	**`010`**	Quad mode (four bits per clock)
3	**`011`**	Quad mode (four bits per clock) + continuous read
4	**`100`**	Dual mode (two bits per clock)
5	**`101`**	Dual mode (two bits per clock) + continuous read
6	**`110`**	Quad DDR mode (eight bits per clock)
7	**`111`**	Quad DDR mode (eight bits per clock) + continuous read

Continuous read mode eliminates the instruction byte whenever the address changes, assuming the same instruction applies until the mode is reset. This results in a modest increase in data throughput.

The SPI flash can be accessed by bit banging when the enable is off. To do this from the CPU, the entire routine to access the SPI flash must be read into SRAM and executed from the SRAM.

External clock XCLK (pin 25)

The external clock functions as a source clock for the two ADCs, and may also be used to clock the whole processor. The processor does not cleanly switch between clock sources, and clock source switching must be done by accessing register 5 in the housekeeping SPI (see *Table 14*).

UART SER_TX (pin 31) and SER_RX (pin 32)

The UART is a standard 2-pin serial interface that can communicate with most similar interfaces at a fixed baud rate. Although the UART operates independently of the CPU, data transfers are blocking operations which will generate CPU wait states until the data transfer is completed.

The behavior of the UART can be modified by changing values in the registers below:

Table 16 **`reg_uart_clkdiv`**

0x0300007b	0x0300007a	0x03000079	0x03000078	address
UART clock divider				value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The entire 32 bit word encodes the number of CPU core cycles to divide down to get the UART data bit rate (baud rate). The default value is 1.

Example: If the external crystal is 12.5MHz, then the core CPU clock runs at 100MHz. To get 9600 baud, 100E6 / 9600 = 10417 (hex value 0x28b1).

Table 17 **`reg_uart_data`**

0x0300007f	0x0300007e	0x0300007d	0x0300007c	address
(unused, value is 0x0)				value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Writing a value to this register will immediately start a data transfer on the SER_TX pin. If a UART write operation is pending, then the CPU will be blocked with wait states until the transfer is complete before starting the new write operation. This makes the UART transmit a relatively expensive operation on the CPU, but avoids the necessity of buffering data and checking for buffer overflow. Reading a value from this register returns 255 (0xff) if no valid data byte is in the receive buffer, and returns the value of the receive buffer otherwise, and clears the receive buffer for additional reads. Note that there is no FIFO associated with the UART.

Interrupt IRQ (pin 28)

The interrupt pin triggers the CPU interrupt channel 5.

I^2C Master

I2C_SDA (pin 26) and I2C_SCL (pin 27)

Table 18 **reg_i2c_config (reg_i2c_control, reg_i2c_prescale)**

0x030000d7	0x030000d6	0x030000d5	0x030000d4	address
(undefined)	control	I^2C clock prescaler		value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bit 22 (control bit 6) is the I^2C interrupt enable. Value 1 = enabled, value 0 = disabled.
Bit 23 (control bit 7) is the I^2C system enable. Value 1 = enabled, value 0 = disabled.

Clock prescaler is a 16-bit integer that defines the I^2C clock period as the divided-down core clock.

Table 19 **reg_i2c_status (reg_i2c_command)**

0x030000db	0x030000da	0x030000d9	0x030000d8	address
(undefined, reads zero)			I^2C status/cmd	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Command (write to 0x030000d8)

Bit 7 Start
Bit 6 Stop
Bit 5 Read
Bit 4 Write
Bit 3 Acknowledge
Bit 0 Interrupt acknowledge

Status (read from 0x030000d8)

Bit 7 Receive acknowledge
Bit 6 I^2C busy (start signal detected)
Bit 5 Arbitration lost
Bit 1 Transfer in progress
Bit 0 Interrupt pending flag

The interrupt pending status bit is set whenever transmission is done or arbitration is lost, regardless of the state of the interrupte enable; and persists until an interrupt acknowledge command is issued. The CPU does not receive an interrupt signal unless the interrupt enable is set in the configuration register.

Note that reg_i2c_status and reg_i2c_command both refer to address 0x03000d8 and may be used interchangeably. Commands are immediate and volatile, and the command word cannot be read back.

The process for reading and writing is largely dependent on the slave device's I^2C protocol definition. Simple read and write commands are outlined on the next page. However, more complicated transmissions involving writing and reading separated by a repeat start signal are possible.

Table 20 **reg_i2c_data**

0x030000df	0x030000de	0x030000dd	0x030000dc	address
(undefined, reads zero)			I^2C data	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The byte at 0x030000dc holds the I^2C data (either read or write)

I^{2}C Master *(continued)*

To write a byte on the I^{2}C interface:

1. Write slave address byte to reg_i2c_data
2. Write command Start + Write (0x90) to reg_i2c_command
3. Wait for transfer-in-progress bit in status (0x02) from reg_i2c_status to clear
4. Check for receive acknowledge bit = 0 in status (0x80) from reg_i2c_status;
   If receive acknowledge = 1, issue command Stop (0x40) and terminate.
5. Write data byte to reg_i2c_data
6. Write command Write (0x10) to reg_i2c_command
7. Wait for transfer-in-progress bit in status (0x02) from reg_i2c_status to clear
8. Repeat from (4) if there are additional bytes to send.
9. Write command Stop (0x40) to reg_i2c_command
10. Check for receive acknowledge bit = 0 in status (0x80) from reg_i2c_status;
    If receive acknowledge = 1, return error to caller; otherwise return success.

To read a byte on the I^{2}C interface:

1. Write slave address byte to reg_i2c_data
2. Write command Start + Write (0x90) to reg_i2c_command
3. Wait for transfer-in-progress bit in status (0x02) from reg_i2c_status to clear
4. Check for receive acknowledge bit = 0 in status (0x80) from reg_i2c_status;
   If receive acknowledge = 1, issue command Stop (0x40) and terminate.
5. Write command Read (0x20) to reg_i2c_command
6. Wait for transfer-in-progress bit in status (0x02) from reg_i2c_status to clear
7. Read data byte from reg_i2c_data
8. Write command Acknowledge (0x08) if there are additional bytes to receive.
9. Repeat from (5) if there are additional bytes to receive.
10. Write command Stop (0x40) to reg_i2c_command

To read a byte on the I^{2}C interface from a device that requires a memory location:

1. Write slave address byte to reg_i2c_data
2. Write command Start + Write (0x90) to reg_i2c_command
3. Wait for transfer-in-progress bit in status (0x02) from reg_i2c_status to clear
4. Check for receive acknowledge bit = 0 in status (0x80) from reg_i2c_status;
   If receive acknowledge = 1, issue command Stop (0x40) and terminate.
5. Write command Start + Write (0x90) to reg_i2c_command
6. Wait for transfer-in-progress bit in status (0x02) from reg_i2c_status to clear
7. Check for receive acknowledge bit = 0 in status (0x80) from reg_i2c_status;
   If receive acknowledge = 1, issue command Stop (0x40) and terminate.
8. Write command Read (0x20) to reg_i2c_command
9. Wait for transfer-in-progress bit in status (0x02) from reg_i2c_status to clear
10. Read data byte from reg_i2c_data
11. Write command Acknowledge (0x08) if there are additional bytes to receive.
12. Repeat from (5) if there are additional bytes to receive.
13. Write command Stop (0x40) to reg_i2c_command

The I^{2}C module is a largely unmodified version of Richard Herveille's code on opencores.org, which can be found here:

`http://opencores.org/ocsvn/i2c/`

Documentation in the opencores.org repository has additional programming examples.

SPI Master SPI_SDI (pin 36), SPI_CSB (pin 35), SPI_SCK (pin 34), and SPI_SDO (pin 33)

Table 21 **`reg_spi_config`**

0x03000033	0x03000032	0x03000031	0x03000030	address
(undefined, reads zero)		SPI master configuration		value
31–24	23–16	15–8	7–0	bit

Configuration bit defintions

Bit 14	SPI interrupt enable	0 = interrupt disabled 1 = interrupt enabled
Bit 13	SPI system enable	0 = SPI disabled 1 = SPI enabled
Bit 12	stream	0 = apply/release CSB separately for each byte 1 = apply CSB until stream bit is cleared (manually)
Bit 11	mode	0 = read and change data on opposite SCK edges 1 = read and change data on the same SCK edge
Bit 10	invert SCK	0 = normal SCK 1= inverted SCK
Bit 9	invert CSB	0 = normal CSB (low is active) 1 = inverted CSB (high is active)
Bit 8	MLB	0 = msb first 1 = lsb first
Bits 7–0	prescaler	count (in master clock cycles) of 1/2 SCK cycle (default value 2)

All configuration bits other than the prescaler default to value zero.

Table 22 **`reg_spi_data`**

0x03000033	0x03000032	0x03000031	0x03000030	address
(undefined, reads zero)			SPI data	value
31–24	23–16	15–8	7–0	bit

The byte at 0x03000030 holds the SPI data (either read or write)

Reading to and writing from the SPI master is simply a matter of setting the required values in the configuration register, and writing values to or reading from reg_spi_data. The protocol is similar to the UART. A write operation will stall the CPU if an incomplete SPI transmission is still in progress. Reading from the SPI will also stall the CPU if an incomplete SPI transmission is still in progress. There is no FIFO buffer for data. Therefore SPI reads and writes are relatively expensive operations that tie up the CPU, but will not lose or overwrite data. Note that there is no FIFO associated with the SPI master.

Comparator COMP_INP (pin 3) and COMP_INN (pin 4)

The analog comparator compares two analog values and produces a digital value 1 or 0 depending on whether or not the positive input is greater than the negative input. The source signal for both the positive input and the negative input is selectable, and the destination of the output value is also selectable.

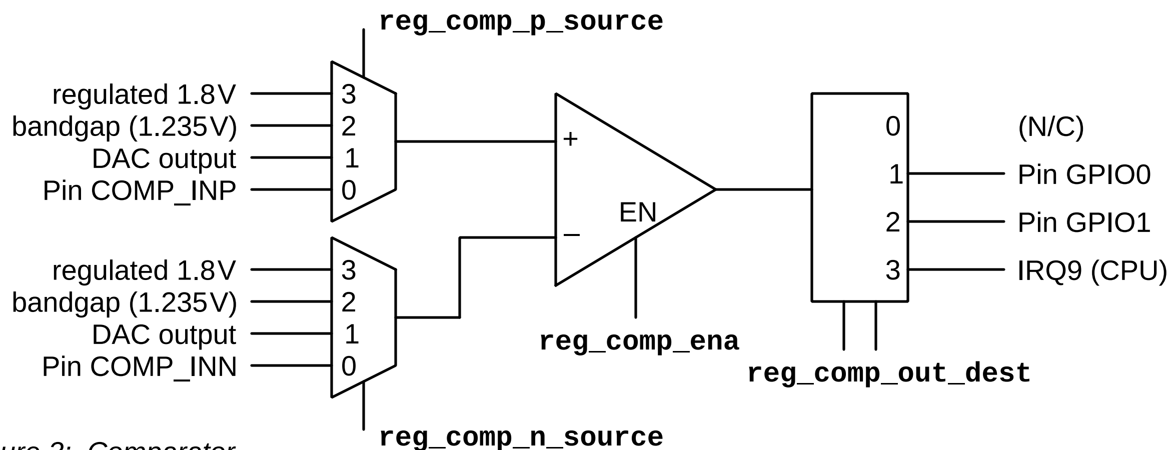


Figure 3: Comparator

Table 23 **`reg_comp_ena`**

<table>
<tr><td colspan="8">0x03000063</td><td colspan="8">0x03000062</td><td colspan="8">0x03000061</td><td colspan="8">0x03000060</td><td>address</td></tr>
<tr><td colspan="24">(undefined, reads zero)</td><td colspan="8">comparator enable</td><td>value</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>bit</td></tr>
</table>

Bit 0 controls the comparator state. Value 1 = enabled, value 0 = disabled.

Table 24 **`reg_comp_n_source`**

<table>
<tr><td colspan="8">0x03000067</td><td colspan="8">0x03000066</td><td colspan="8">0x03000065</td><td colspan="8">0x03000064</td><td>address</td></tr>
<tr><td colspan="24">(undefined, reads zero)</td><td colspan="8">comparator source</td><td>value</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>bit</td></tr>
</table>

The two low bits of this register determine the source signal on the comparator's negative input, according to the following table:

Register byte **`0x03000064`** value		Signal directed to the comparator negative input
0	**`00`**	Pin COMP_INN (pin 4) (default)
1	**`01`**	DAC output value
2	**`10`**	Bandgap output value (1.235V)
3	**`11`**	Regulated 1.8V

`reg_comp_out_dest`: See *Table 5*

Table 25 **`reg_comp_p_source`**

0x0300006b	0x0300006a	0x03000069	**0x03000068**	address
(undefined, reads zero)			comparator source	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The two low bits of this register determine the source signal on the comparator's negative input, according to the following table:

Register byte **0x03000068** value		Signal directed to the comparator negative input
0	**00**	Pin COMP_INP (pin 3) (default)
1	**01**	DAC output value
2	**10**	Bandgap output value (1.235V)
3	**11**	Regulated 1.8V

DAC/bandgap output ANALOG_OUT (pin 37)

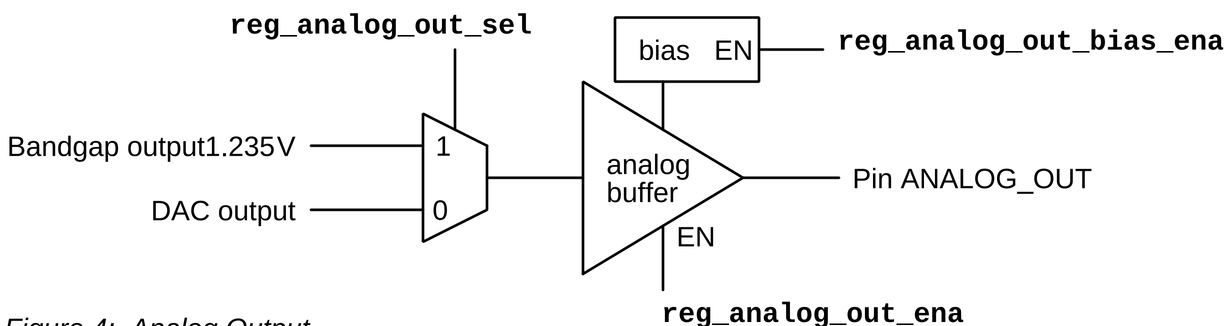


Figure 4: Analog Output

Table 26 **`reg_analog_out_ena`**

0x030000c3	0x030000c2	0x030000c1	**0x030000c0**	address
(undefined, reads zero)			buffer enable	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bit 0 controls the analog output buffer state. Value 1 = enabled, value 0 = disabled.

Table 27 **`reg_analog_out_bias_ena`**

0x030000c7	0x030000c6	0x030000c5	**0x030000c4**	address
(undefined, reads zero)			buffer bias enable	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bit 0 controls the analog output bias current generator state. Value 1 = enabled, value 0 = disabled.

Table 28 **`reg_analog_out_sel`**

0x030000cb	0x030000ca	0x030000c9	**0x030000c8**	
(undefined, reads zero)			buffer input source	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 **1 0**	bit

The two low bits of this register determine the signal that appears at the ANALOG_OUT pin, according to the following table:

Register byte **`0x030000c8`** value		Signal appearing on ANALOG_OUT pin (pin 5)
0	**0**	DAC output value
1	**1**	Bandgap output value (1.235V)

DAC VREF_H (pin 6) and VREF_L (pin 7), ANALOG_OUT (pin 5)

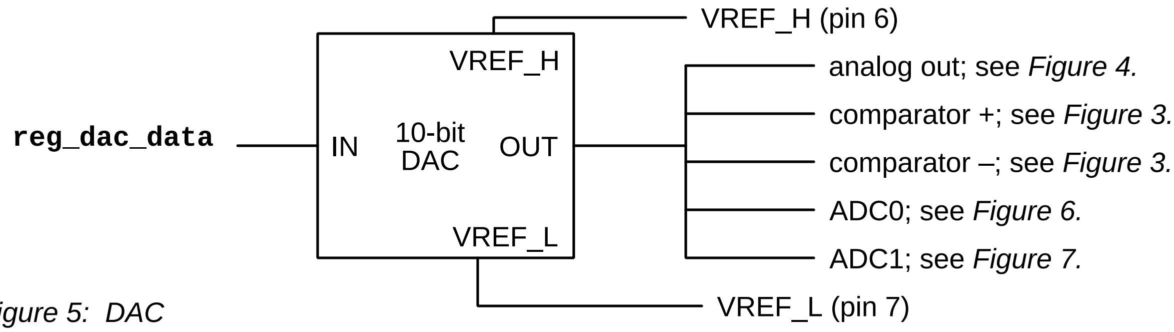


Figure 5: DAC

Table 29 **`reg_dac_ena`**

0x03000053	0x03000052	0x03000051	**0x03000050**	address
(undefined, reads zero)			DAC enable	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 **0**	bit

Bit 0 controls the DAC state. Value 1 = enabled, value 0 = disabled.

Table 30 **`reg_dac_data`**

0x03000057	0x03000056	**0x03000055**	**0x03000054**	address
(undefined, reads zero)			DAC input value	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 **9 8**	**7 6 5 4 3 2 1 0**	bit

The word formed by bits 0 to 9 define the integer value input to the DAC. The DAC output value is defined by the following equation:

$$\text{DAC output value (V)} = \text{VREF\_L} + \frac{(\text{VREF\_H} - \text{VREF\_L}) \cdot \text{reg\_dac\_data}}{1024}$$

ADC0

ADC0_IN (pin 9)
VREF_H (pin 6) and VREF_L (pin 7)

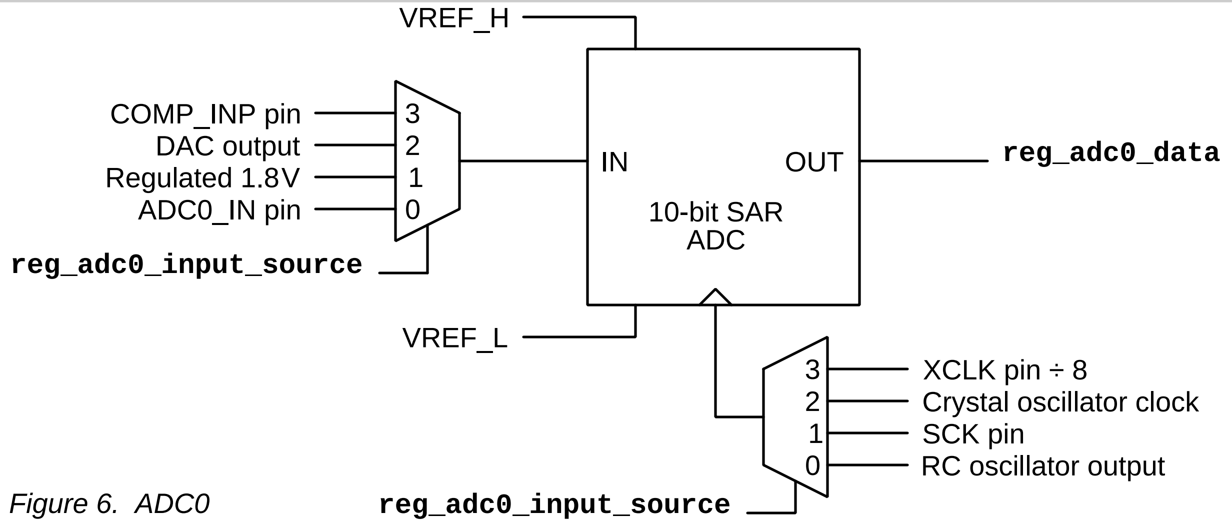


Figure 6. ADC0

Table 31 **`reg_adc0_ena`**

0x03000013								0x03000012								0x03000011								**0x03000010**								address
(undefined, reads zero)																								ADC0 enable								value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Bit 0 controls the ADC0 state. Value 1 = enabled, value 0 = disabled.

Table 32 **`reg_adc0_data`**

0x03000017								0x03000016								**0x03000015**								**0x03000014**								address
(undefined, reads zero)																						ADC0 output value										value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The word formed by bits 0 to 9 define the integer value returned by the ADC after a conversion. The ADC output value is defined by the following equation:

$$\text{reg\_adc0\_data} = 1024 \cdot \frac{\text{(ADC0 input value)} - \text{VREF\_L}}{\text{VREF\_H} - \text{VREF\_L}}$$

Note that pins VREF_L and VREF_H are shared among the ADC0, ADC1, and DAC.

Table 33 **`reg_adc0_done`**

0x0300001b	0x0300001a	0x03000019	**0x03000018**	address
(undefined, reads zero)			ADC0 done	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 **0**	bit

This bit is read-only and indicates that the ADC0 has completed a conversion. The data transfer is not synchronized to the CPU clock and so this bit must be checked before reading reg_adc0_data in order to get a valid result. Value 1 = conversion done; value 0 = conversion in progress.

Table 34 **`reg_adc0_convert`**

0x0300001f	0x0300001e	0x0300001d	**0x0300001c**	address
(undefined, reads zero)			ADC0 start conv.	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 **0**	bit

Set bit 0 to 1 to start a conversion. The bit is not self-resetting. The bit must be manually set to zero before another conversion can begin.

Table 35 **`reg_adc0_clk_source`**

0x03000023	0x03000022	0x03000021	**0x03000020**	address
(undefined, reads zero)			ADC0 clock source	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 **1 0**	bit

The two low bits of this register determine the digital signal that drives the ADC0 clock. Note that the ADC0 has a maximum clock rate of 2 MHz. Depending on the external setup, some signals may not be able to drive the ADC0 correctly. Note that the crystal clock frequency is divided by 8 so that the frequency at the ADC0 is in range for typical 8–10 MHz crystals.

Register byte **`0x03000020`** value		Signal used to clock the ADC0
0	**`00`**	RC oscillator output (100 MHz)
1	**`01`**	SCK (pin 14)
2	**`10`**	Crystal oscillator clock
3	**`11`**	XCLK (pin 25) ÷ 8

Table 36 **`reg_adc0_input_source`**

0x03000027	0x03000026	0x03000025	**0x03000024**	address
(undefined, reads zero)			ADC0 input source	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 **1 0**	bit

The two low bits of this register determine the analog signal that is presented to the ADC0 input, according to the following table:

Register byte **`0x03000024`** value		Signal directed to the ADC0 input
0	**`00`**	Pin ADC0_IN (pin 9) (default)
1	**`01`**	Regulated 1.8V
2	**`10`**	DAC output value
3	**`11`**	Pin COMP_INP (pin 3)

ADC1

ADC1_IN (pin 40)
VREF_H (pin 38) and VREF_L (pin 39)

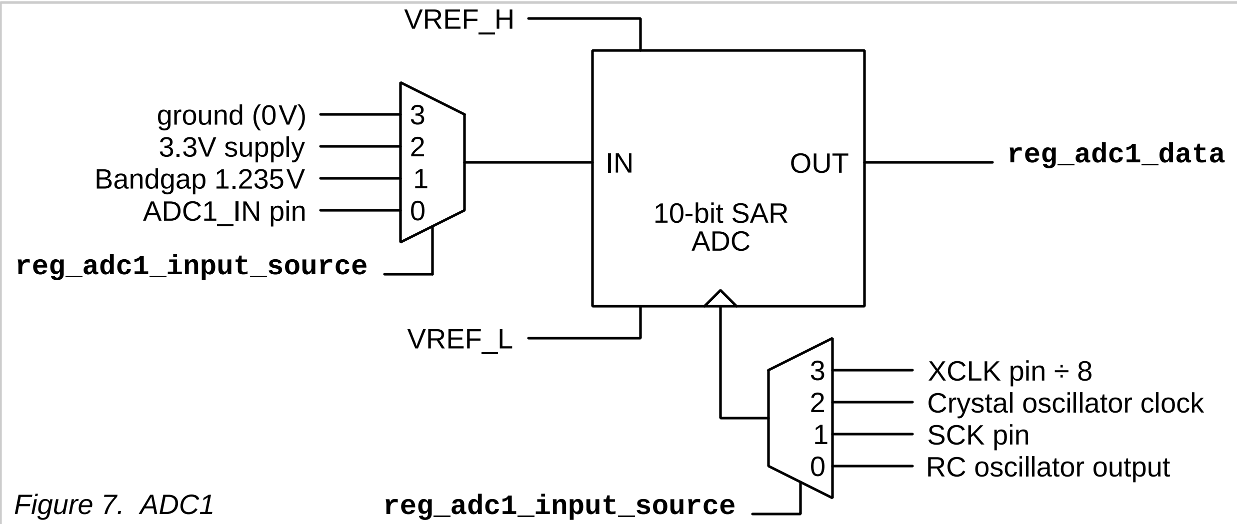


Figure 7. ADC1

Table 37 **reg_adc1_ena**

0x03000033	0x03000032	0x03000031	**0x03000030**	address
(undefined, reads zero)			ADC1 enable	value
31–24	23–16	15–8	7 6 5 4 3 2 1 **0**	bit

Bit 0 controls the ADC1 state. Value 1 = enabled, value 0 = disabled.

Table 38 **reg_adc1_data**

0x03000037	0x03000036	**0x03000035**	**0x03000034**	address
(undefined, reads zero)			ADC1 output value	value
31–24	23–16	15 14 13 12 11 10 **9 8**	**7 6 5 4 3 2 1 0**	bit

The word formed by bits 0 to 9 define the integer value returned by the ADC1 after a conversion. The ADC1 output value is defined by the following equation:

$$\text{reg\_adc1\_data} = 1024 \cdot \frac{\text{(ADC1 input value)} - \text{VREF\_L}}{\text{VREF\_H} - \text{VREF\_L}}$$

Note that pins VREF_L and VREF_H are shared among the ADC0, ADC1, and DAC.

Table 39 **reg_adc1_done**

0x0300003b	0x0300003a	0x03000039	**0x03000038**	address
(undefined, reads zero)			ADC1 done	value
31–24	23–16	15–8	7 6 5 4 3 2 1 **0**	bit

This bit is read-only and indicates that the ADC1 has completed a conversion. The data transfer is not synchronized to the CPU clock and so this bit must be checked before reading reg_adc1_data in order to get a valid result. Value 1 = conversion done; value 0 = conversion in progress.

Table 40 **reg_adc1_convert**

0x0300003f	0x0300003e	0x0300003d	**0x0300003c**	address
(undefined, reads zero)			ADC1 start conv.	value
31–24	23–16	15–8	7 6 5 4 3 2 1 **0**	bit

Set bit 0 to 1 to start a conversion. The bit is not self-resetting. The bit must be manually set to zero before another conversion can begin.

Table 41 **reg_adc1_clk_source**

0x03000043	0x03000042	0x03000041	**0x03000040**	address
(undefined, reads zero)			ADC1 clock source	value
31–24	23–16	15–8	7 6 5 4 3 2 **1 0**	bit

The two low bits of this register determine the digital signal that drives the ADC1 clock. Note that the ADC1 has a maximum clock rate of 2 MHz. Depending on the external setup, some signals may not be able to drive the ADC1 correctly. Note that the crystal clock frequency is divided by 8 so that the frequency at the ADC1 is in range for typical 8–10 MHz crystals.

Register byte **0x03000040** value		Signal used to clock the ADC1
0	**00**	RC oscillator output (100 MHz)
1	**01**	SCK (pin 14)
2	**10**	Crystal oscillator clock
3	**11**	XCLK (pin 24) ÷ 8

Table 42 **reg_adc1_input_source**

0x03000047	0x03000046	0x03000045	**0x03000044**	address
(undefined, reads zero)			ADC1 input source	value
31–24	23–16	15–8	7 6 5 4 3 2 **1 0**	bit

The two low bits of this register determine the analog signal that is presented to the ADC1 input, according to the following table:

Register byte **0x03000044** value		Signal directed to the ADC1 input
0	**00**	Pin ADC1_IN (pin 40) (default)
1	**01**	Bandgap 1.235V output
2	**10**	3.3V power supply
3	**11**	Ground reference (0V)

Crystal oscillator XI (pin 11) and XO (pin 12)

The crystal oscillator is nominally rated for 4MHz but will operate at 12MHz for a core clock rate of 80MHz. The crystal oscillator can be bypassed in two ways: By setting the pll_bypass signal in the housekeeping SPI to value 1 to switch the clock to the XCLK pin, and by turning off the crystal oscillator and PLL charge pump in the housekeeping SPI, allowing the VCO in the PLL to run freely.

When not in PLL bypass mode, the core clock rate is 8 times the crystal frequency. In PLL bypass mode, the core clock rate is equal to the rate of XCLK.

1.8V regulated supply VDD1V8 (pins 18, 30, and 38)

The two on-board voltage regulators supply the 1.8V core power supply for the digital components of the chip, including the CPU. Together the regulators supply 40mA of current, which covers the peak demand of the digital core at 100MHz. The 1.8V supply pins should be connected to external decoupling caps of 3.3µF for each regulator.

Average current draw of the digital core and regulators at 80MHz core clock is (TBD)mA.

3.3V power supply VDD3V3 (pins 17 and 37)

VSS (pin 29 and QFN package paddle)

The power supply for the Raven chip is 3.3V nominal (see maximum ratings). Ground (VSS) is connected to the QFN package paddle as well as the two VSS pins (31 and 34).

Bandgap reference

The bandgap reference generates a nominal 1.235V output that can be routed to the comparator or ADC1, or output on the ANALOG_OUT pin (see Tables 23 and 24 for comparator input routing, Table 42 for the ADC1 input routing, and Table 28 for output routing).

Table 43 **reg_bandgap_ena**

0x030000d3	0x030000d2	0x030000d1	0x030000d0	address
(undefined, reads zero)			Bandgap enable	value
31–24	23–16	15–8	7–0	bit

Bit 0 controls the bandgap state. Value 1 = enabled, value 0 = disabled.

RC oscillator

The on-board RC oscillator generates a 100MHz clock output that can be directed to the ADCs to be used as the ADC conversion clock, or passed directly to a GPIO channel to be output on one of GPIO2, GPIO3, or GPIO6 (see Table 6).

Table 44 **`reg_rcosc_enable`**

0x03000073	0x03000072	0x03000071	**0x03000070**	address
(undefined, reads zero)			RC osc. enable	value
31–24	23–16	15–8	7 6 5 4 3 2 1 0	bit

Bit 0 controls the RC oscillator state. Value 1 = enabled, value 0 = disabled.

Overtemperature alarm

The on-board over-temperature alarm will generate a digital high output when the internal temperature of the chip exceeds a fixed value of approximately 133°C. Hysteresis of 7°C ensures that the signal will remain high until the temperature falls below 126°C. The over−temperator alarm can be routed to a GPIO for monitoring, and can be set to trigger a CPU interrupt state. The value can also be read directly from memory. For output routing, see Table 10.

Table 45 **`reg_overtemp_ena`**

0x030000e3	0x030000e2	0x030000e1	**0x030000e0**	address
(undefined, reads zero)			Overtemp alarm en.	value
31–24	23–16	15–8	7 6 5 4 3 2 1 0	bit

Bit 0 controls the over-temperature alarm state. Value 1 = enabled, value 0 = disabled.

Table 46 **`reg_overtemp_data`**

0x030000e7	0x030000e6	0x030000e5	**0x030000e4**	address
(undefined, reads zero)			Overtemp alarm	value
31–24	23–16	15–8	7 6 5 4 3 2 1 0	bit

Read-only bit 0 is the over-temperature alarm output. Value 1 = alarm triggered (core temperature over 133°C was detected, and current temperature is above 126°C), and value 0 = alarm not triggered (core temperature is below 126°C).

Counter-Timer

The counter/timer is a general-purpose 32-bit adder and subtractor that can be configured for a variety of timing functions including one-shot counts, continuous timing, and interval interrupts. At a core clock rate of 80 MHz, the longest single time interval is 26.84 seconds.

Table 47 **reg_timer_config**

0x030000f7	0x030000f6	0x030000f5	0x030000f4	address
(undefined, reads zero)			Timer config	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Timer configuration bit definitions

Bit 3	Counter/timer enable	1 = counter/timer enabled 0 = counter/timer disabled
Bit 2	Oneshot mode	1 = oneshot mode 0 = continuous mode
Bit 1	Updown	1 = count up 0 = count down
Bit 0	Interrupt enable	1 = interrupt enabled 0 = interrupt disabled

Table 48 **reg_timer_value**

0x030000fb	0x030000fa	0x030000f9	0x030000f8	address
Timer value				value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The value in this register is the current value of the counter. Value is 32 bits. The register is read-write and can be used to reset the timer.

Table 49 **reg_timer_data**

0x030000ff	0x030000fe	0x030000fd	0x030000fc	address
Timer data				value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

The value in this register is the reset value for the comparator.

When enabled, the counter counts up or down from the value set in reg_timer_value at the time the counter is enabled. If counting up, the count continues until the counter reaches reg_timer_data. If counting down, the count continues until the counter reaches zero.

In continuous mode, the counter resets to zero if counting up, and resets to the value in reg_timer_data if counting down, and the count continues immediately. If the interrupt is enabled, the counter will generate an interrupt on every cycle.

In one-shot mode, the counter triggers an interrupt (IRQ channel 13; see next page) when it reaches the value of reg_timer_data (up count) or zero (down count), and stops.

Note: When the counter/timer is disabled, the reg_timer_value remains unchanged, which puts the timer in a hold state. When re-enabled, counting resumes. To reset the timer, write zero to the reg_timer_value register.

Interrupts (IRQ)

The interrupt vector is set to memory addres 0 (bottom of SRAM). The program counter switches to this location when an interrupt is received. To enable interrupts, it is necessary to copy an interrupt handler to memory location 0. The PicoRV32 defines 32 IRQ channels, of which the Ravenna chip uses only a handful, as described in the table below. All IRQ channels not in the list below always have value zero.

Table 50 CPU IRQ channel definitions

IRQ channel	*description*
4	UART data available
5	IRQ external pin (pin 26)
6	Housekeeping SPI IRQ
7	Assignable interrupt (see Table 11)
8	Assignable interrupt (see Table 12)
9	Comparator output, when enabled (see Table)
10	Over-temperature output, when enabled (see Table)
11	I^2C data available, when enabled (see Table)
12	SPI master data available, when enabled (see Table)
13	Timer expired, when enabled (see Table)

The Ravenna PicoRV32 does not enable IRQ QREGS (see PicoRV32 description).

The handling of interrupts is beyond the scope of this document (see RISC-V instruction set description). All interrupts are masked and must be enabled in software.

SRAM

The Ravenna chip has an on-board memory of 1024 words of width 32 bits. The memory is located at address 0 (zero). The SPI flash is configured to overlay the SRAM memory. Therefore the first 1024 words of SRAM are not accessible from the CPU (unless accessed by direct bit-banging of the SPI flash signals from a program copied to, and executed from, SRAM). The SPI flash can be considered an extension of memory above 1k-word. However, SPI flash access severely reduces the processor throughput, as the SPI flash access for read and write takes numerous clock cycles (depending on the SPI flash access mode), and occurs between program counter updates, causing all program reads from the SPI flash to execute a full sequence of command, address, and data transfer.

NVRAM

The Ravenna NVRAM section is a non-volatile memory block that is 136 words by 32 bits and located between addresses 0x02800000 and 0x02800087. The memory consists of two pages. The first page is 128 bits (0x02800000 to 0x0280007f) and is considered user-accessible memory for nonvolatile storage. The remaining area (8 words from 0x02800080 to 0x02800087) is calibration memory and should not be used for other purposes.

`reg_nvram_datatop` **`0x02800000`**

The NVRAM block consists of a volatile memory overlaid on nonvolatile memory. The NVRAM memory can be read and written to like normal SRAM. The control bits are used to store the contents to non-volatile memory, or to restore from non-volatile memory.

Table 51 **`reg_nvramctrl`**

0x0300005b	0x0300005a	0x03000059	**0x03000058**	address
(undefined, reads zero)			NVRAM control	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bit 4 is NVRAM ready. This bit is zero during a non-volatile access. No other memory access to the NVRAM space may be made while this bit is 1. Operations are automatically blocking with wait states.

Bit 3 is NVRAM store. Set high to initiate a non-volatile memory store. Zero manually.
Bit 2 is NVRAM recall. Set high to initiate non-volatile memory recall. Zero manually.
Bit 1 is NVRAM mem select. Set to 0 for store/recall on bank 1, and 1 for store/recall on bank 2.
Bit 0 is NVRAM mem all. Set to 1 for store/recall to both banks. Normally this is left at zero.

Table 52 **`reg_nvram_clkdiv`**

0x0300005f	0x0300005e	0x0300005d	**0x0300005c**	address
(undefined, reads zero)			clock divider	value
31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	bit

Bits 4 to 0 define the clock divider for the NVRAM clock. The core clock is divided by this value + 1 to obtain the NVRAM clock. The NVRAM clock must be exactly 4MHz. The clock can be routed to GPIO 10 to monitor the exact value.

NVRAM Calibration

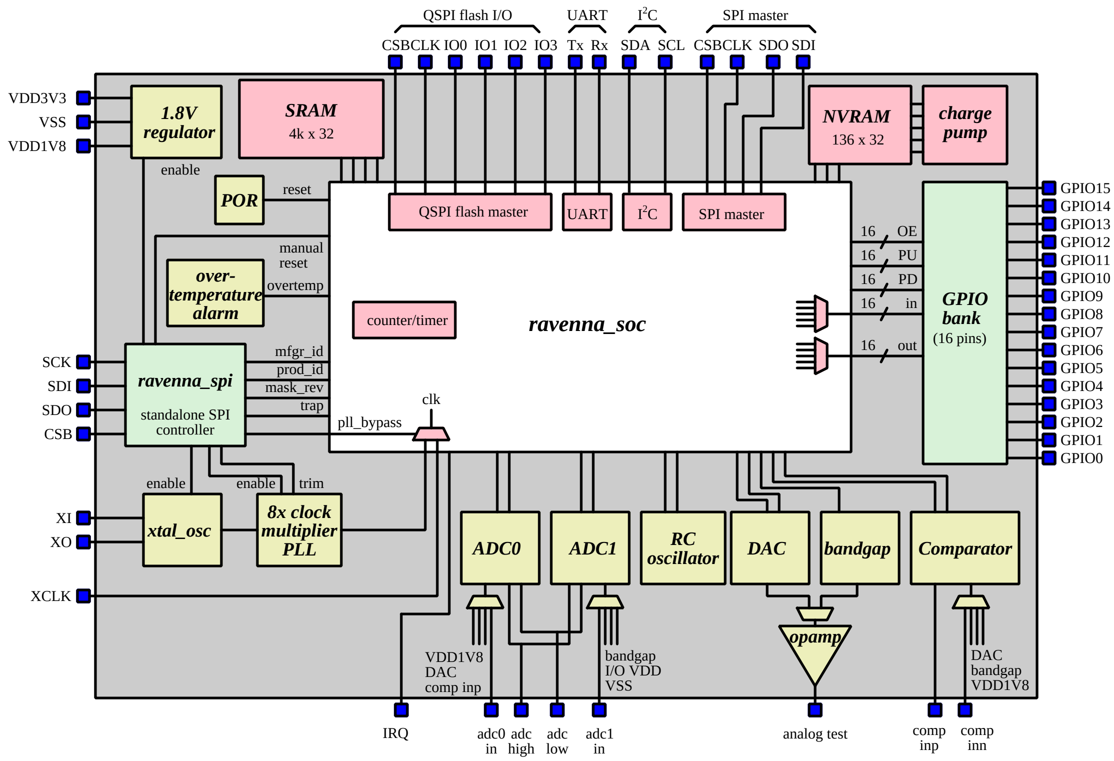
Calibration requires accessing the NVRAM configuration through the housekeeping SPI and setting the 4-bit value **tm_nvcp** to the test number below, and performing the specified test. Analog values to be measured appear on the NVREF pin (pin 10 on the 48-pin QFN package). The measurement of the 4MHz clock can be made from GPIO10 if the value of reg_pll_dest is set to 3 (see Table 8). The NVRAM clock must be checked for 4MHz frequency before running the calibration. Because the NVRAM is driven with an internal clock, clock calibration is not needed other than confirming the value of 4MHz (250ns period).

TRIM is at memory address 0x0280021c (last word in NVRAM, or last 4 bytes). Only values in the low byte are potentially adjusted by the calibration procedure. All other bits in this word must remain zero.

tm_nvcp		
0100	Measure V_{RECALL} at NVREF	Confirm between 0.49 and 0.66V.
1010	Measure V_{BG0} at NVREF	Confirm between 1.10 and 1.20V.
1011	Measure V_{BG} at NVREF	Calculate $V_{OFF} = V_{BG} – V_{BG0}$ (Confirm –0.05 to 0.05V)
0010	Perform complete store cycle (store) and read NVREF at the following times:	
0010	Measure V_{UM1P_MEAS} at NVREF after 100 to 300 µs after store cycle high pulse.	
0010	Measure V_{UM1N_MEAS} at NVREF after 1 to 6 ms after store cycle high pulse.	
	Calculate $V_{UM1P} = V_{UM1P\_MEAS} – V_{OFF}$	Calculate $V_{UM1N} = V_{UM1N\_MEAS} – V_{OFF}$
0011	Perform complete store cycle (store) and read NVREF at the following times:	
0011	Measure V_{UM2P_MEAS} at NVREF after 100 to 300 µs after store cycle high pulse.	
0011	Measure V_{UM2N_MEAS} at NVREF after 1 to 6 ms after store cycle high pulse.	
	Calculate $V_{UM2P} = V_{UM2P\_MEAS} – V_{OFF}$	Calculate $V_{UM2N} = V_{UM2N\_MEAS} – V_{OFF}$
	Read and remember values for TRIM[5:0] (automatically set by the tests above).	
0000	Write positive trim bits from TRIM[2:0] into negative trim bits TRIM[5:3] (if different)	
0110	Measure V_{UM1_MEAS} at NVREF.	Calculate $V_{UM1} = V_{UM1\_MEAS} – V_{OFF}$
0111	Measure V_{UM2_MEAS} at NVREF.	Calculate $V_{UM1} = V_{UM2\_MEAS} – V_{OFF}$
0000	Write back original negative trim bits into TRIM[5:3] (if different)	
	Calculate $N_{RP1} = V_{UM1} / (V_{BG0} – V_{UM1})$	Calculate $N_{RP2} = V_{UM2} / (V_{BG0} – V_{UM2})$
	Calculate $V_{SE1P} = V_{UM1P} \cdot (N_{RP1} + 1)$	Calculate $V_{SE2P} = V_{UM2P} \cdot (N_{RP2} + 1)$
	Calculate $V_{SE1N} = –(28 \cdot V_{UM1N})$	Calculate $V_{SE2N} = –(28 \cdot V_{UM2N})$
	Confirm V_{SE1P} and V_{SE2P} are between 10.94 and 11.54V.	
	Confirm V_{SE1N} and V_{SE2N} are between –10.61 and –10.01V.	
	If all values are validated, then store TRIM into non-volatile memory (store + mem_sel).	

Best practice is not to touch NVRAM page 2 (0x02800200 to 0x0280021f) after calibration, and to never set bits mem_sel or mem_all for NVRAM store. However, if careful not to alter the TRIM bits, then all values in page 2 other than the two TRIM bytes may be used for storage. All NVRAM memory can be used for volatile storage regardless of the non-volatile state.

Ravenna PicoRV32 simplified block diagram



Programming

The RISC-V architecture has a **gcc** compiler. The best reference for getting the correct cross-compiler version is the PicoRV32 source at

`http://github.com/cliffordwolf/picorv32`.

Specifically, see the top-level **`README.md`** file section “Building a pure RV32I Toolchain.”

For programming examples specifically for the Ravenna chip (assuming a correct installation of a RISC-V gcc toolchain as described above), see

`http://github.com/efabless/ravenna-picorv32`

The directory **`verilog/`** contains example source code to program the Ravenna chip along with the header file **`ravenna_defs.h`** that defines the memory-mapped locations as described throughout this text.

The **`verilog/`** directory contains a **`Makefile`** that compiles hex files and runs simulations of a number of test programs that exercise various features of the chip.

Additional documentation exists on the same site for an example demonstration circuit board and driver software.

Additional references

See **`http://riscv.org/`**
`http://riscv.org/software-status/`

Memory Mapped I/O summary by address

Address (bytes)	Function	
`0x00 00 00 00`	Flash SPI / overlaid SRAM (4k words) start of memory block	
`0x00 00 3f ff`	End of SRAM	
`0x00 10 00 00`	Flash SPI start of program block	Program to run starts here on reset.
`0x00 ff ff ff`	Maximum SPI flash addressable space (16MB) with QSPI 3-byte addressing	
`0x01 ff ff ff`	Maximum SPI flash addressable space (32MB)	
`0x02 00 00 00`	SPI master config bit 31 MEMIO enable (reset = 1) 0 = bit-bang mode bit 22 DDR enable bit 21 QSPI enable bit 20 CRM enable bits 19-16 Read latency cycles bits 11-8 I/O output enable bits (bit bang mode) bit 5 Chip select line (bit bang mode) bit 4 Serial clock line (bit bang mode) bits 3-0 Data bits (bit bang mode)	bit names do not correspond exactly to function; see Table 15
`0x02 80 00 00`	Start of NVRAM page 1	
`0x02 80 02 00`	Start of NVRAM page 2	
`0x02 80 02 1c`	Start of NVRAM configuration word (do not overwrite after calibration!)	
`0x02 80 02 1f`	End of NVRAM memory	
`0x03 00 00 00`	GPIO input/output (lower 16 bits)	16 bits of general-purpose digital
`0x03 00 00 04`	GPIO output enable (inverted) (0 = output, 2 = input)	
`0x03 00 00 08`	GPIO pullup enable (0 = pullup, 1 = none)	
`0x03 00 00 0c`	GPIO pulldown enable (0 = pulldown, 2 = none)	
`0x03 00 00 10`	ADC0 enable (low bit 1 = enabled, 0 = disabled)	
`0x03 00 00 14`	ADC0 data (10 bits) read-only	
`0x03 00 00 18`	ADC0 done (low bit 1 = done, 0 = busy) read-only	
`0x03 00 00 1c`	ADC0 start conversion (low bit 1 = convert, 0 = reset)	
`0x03 00 00 20`	ADC0 clock source (low 2 bits) 0 = RC osc 1 = SPI SCK 2 = Xtal clock ÷ 8 3 = XCLK pin	ADC can be manually clocked via the SPI SCK or XCLK pins, or automatically clocked at the rate of the crystal or the RC oscillator.
`0x03 00 00 24`	ADC0 input source (low 2 bits) 0 = external pin 1 = VDD1V8 2 = DAC 3 = comparator P input	ADC normally reads the value of an external pin, but for testing can be set to read the internal values of the core power supply, DAC, or the comparator P input pin.
`0x03 00 00 30`	ADC1 enable (low bit 1 = enabled, 0 = disabled)	
`0x03 00 00 34`	ADC1 data (10 bits) read-only	
`0x03 00 00 38`	ADC1 done (low bit 1 = done, 0 = busy) read-only	
`0x03 00 00 3c`	ADC1 start conversion (low bit 1 = convert, 0 = reset)	
`0x03 00 00 40`	ADC1 clock source (low 2 bits) 0 = RC osc 1 = SPI SCK 2 = Xtal clock ÷ 8 3 = XCLK pin	
`0x03 00 00 44`	ADC1 input source (low 2 bits) 0 = external pin 1 = bandgap 2 = VDD3V3 3 = VSS	ADC can be calibrated by setting input to the 3.3V power supply and ground, and can be used to read the bandgap voltage.

Memory Mapped I/O summary by address *(continued)*

Address (bytes)	Function	
0x03 00 00 50	DAC enable (low bit 1 = enabled, 0 = disabled)	
0x03 00 00 54	DAC value (10 bits)	
0x03 00 00 58	NVRAM control (lower 5 bits) bit 0 = mem_all (access all banks simultaneously) bit 1 = mem_sel (0 = access bank 1; 1 = access bank 2 bit 2 = HR (recall non-volatile values in selected bank(s)) bit 3 = HS (store values in selected non-volatile bank(s)) bit 4 = ready (0 = NVRAM store/recall in progress; 1 = NVRAM ready for access)	
0x03 00 00 5c	NVRAM clock divider NVRAM clock = master clock / (NVRAM clock divider + 1) Default value = 24 (0x18) = 4MHz at a core clock rate of 100MHz	
0x03 00 00 60	Comparator enable (low bit 1 = enabled, 0 = disabled)	
0x03 00 00 64	Comparator N input source (low 2 bits) 0 = external pin 1 = DAC 2 = bandgap 3 = VDD1V8	Comparator can have one input set to a known internal value, either the DAC output, the bandgap voltage, or the core supply voltage.
0x03 00 00 68	Comparator P input source (low 2 bits) 0 = external pin 1 = DAC 2 = bandgap 3 = VDD1V8	
0x03 00 00 6c	Comparator output destination (low 2 bits) 0 = none 1 = GPIO(0) 2 = GPIO(1) 3 = IRQ(9)	Comparator output can be seen directly on GPIO 0 or 1, or used to trigger CPU interrupt IRQ 9. GPIOs 0 and 1 cannot be used for general-purpose I/O when selected for comparator output.
0x03 00 00 70	RC oscillator enable (low bit, 1 = enabled, 0 = disabled)	
0x03 00 00 74	RC oscillator output destination (low 2 bits) 0 = none 1 = GPIO(2) 2 = GPIO(3) 3 = GPIO(4)	RC oscillator output can be passed directly to GPIO 2, 3, or 4. These cannot be used as general-purpose I/O when selected for RC oscillator output.
0x03 00 00 78	UART clock divider select (system clock freq. / baud rate)	
0x03 00 00 7c	UART data (returns 0xffffffff if receiver buffer is empty)	
0x03 00 00 80	SPI configuration byte (low 8 bits) (read-only) Value currently fixed at zero	Most bits of the SPI register map are mirrored to memory locations and can be read by the CPU. These bits are read-only and cannot be modified.
0x03 00 00 84	SPI master enables (low 2 bits) (read-only) bit 0 = 1.8V regulator enable bit 1 = Crystal oscillator enable	
0x03 00 00 88	SPI PLL config (low 7 bits) (read-only) bit 0 = PLL current bias enable bit 1 = PLL VCO enable bit 2 = PLL CP enable bits 3-6 = PLL frequency trim	

Memory Mapped I/O summary by address *(continued)*

Address (bytes)	Function	
0x03 00 00 8c	SPI manufacturer ID (low 12 bits) (= 0x0456) (read-only)	
0x03 00 00 90	SPI product ID (low 8 bits) (= 0x03) (read-only)	
0x03 00 00 94	SPI mask revision (low 4 bits) (= 0x00) (read-only)	
0x03 00 00 98	SPI PLL bypass mode (low bit) (read-only)	
0x03 00 00 a0	Crystal output destination (low 2 bits) 0 = none 1 = GPIO(5) 2 = GPIO(6) 3 = GPIO(7)	The crystal oscillator clock (before the PLL) can be coupled to any of GPIO pins 5, 6, or 7. These GPIOs cannot be used as general-purpose I/O when selected for crystal oscillator clock output.
0x03 00 00 a4	PLL clock output destination (low 2 bits) 0 = none 1 = GPIO(8) 2 = GPIO(9) 3 = GPIO(10)	The PLL clock (crystal oscillator clock multiplied up by 8 times) can be viewed on any of GPIO 8, 9, or 10. These GPIOs cannot be used as general-purpose I/O when selected for PLL clock output. It is unlikely that a full-speed (100MHz) clock will be able to toggle the GPIO at full swing, but a slower clock (5 MHz crystal / 40 MHz core clock) should have a proper output.
0x03 00 00 a8	Trap output destination (low 2 bits) 0 = none 1 = GPIO(11) 2 = GPIO(12) 3 = GPIO(13)	The CPU fault state (trap) can be viewed at GPIO 11, 12, or 13, as a way to monitor the CPU trap externally.
0x03 00 00 b0	IRQ 7 input source (low 2 bits) 0 = none 1 = GPIO(0) 2 = GPIO(1) 3 = GPIO(2)	The GPIO inputs can be used as IRQ event sources and passed to the CPU through IRQ channels 7 and 8. When used as IRQ sources, the corresponding GPIO channel must be first configured as an input.
0x03 00 00 b4	IRQ 8 input source (low 2 bits) 0 = none 1 = GPIO(3) 2 = GPIO(4) 3 = GPIO(5)	
0x03 00 00 b8	SPI master configuration register bits 0–7 = prescaler (core clock / (prescaler + 1) = SPI clock rate / 2) (default 2) bit 8 = mlb (0 = msb first, 1 = lsb first) (default 0) bit 9 = invcsb (0 = csb active low, 1 = csb active high) (default 0) bit 10 = invsck (0 = normal sck, 1 = inverted sck) (default 0) bit 11 = mode (0 = read/write on opposite sck edge, 1 = same edge) (default 0) bit 12 = stream (0 = raise csb after each byte, 1 = keep csb low until stream bit cleared) bit 13 = enable (0 = SPI master disabled, 1 = SPI master enabled) bit 14 = irq enable (0 = disabled, 1 = SPI read valid triggers interrupt channel 12)	
0x03 00 00 bc	SPI master data register (low 8 bits) Write data to send to low byte or read received data from low byte.	
0x03 00 00 c0	Analog output select (low bit) 0 = DAC 1 = bandgap	A single op-amp configured as an analog buffer can be selected to mirror the analog DAC output or the bandgap output.
0x03 00 00 c4	Analog output bias enable (low bit, 1 = enabled, 0 = disabled)	
0x03 00 00 c8	Analog output enable (low bit, 1 = enabled, 0 = disabled)	
0x03 00 00 d0	Bandgap enable (low bit, 1 = enabled, 0 = disabled)	

Memory Mapped I/O summary by address *(continued)*

Address (bytes)	Function
0x03 00 00 d4	I^2C configuration register (lower 24 bits) bits 0–15 = prescaler (core clock / (prescaler + 1) = SCL period / 2) (default 0xffff) bit 30 = interrupt enable (0 = disabled, 1 = I^2C read valid triggers IRQ channel 11) bit 31 = I^2C enable (0 = disabled, 1 = enabled) (default 0)
0x03 00 00 d8	I^2C command and status register (lower 8 bits) command register (write only) bit 0 = interrupt acknowledge bit 3 = acknowledge bit 4 = write bit 5 = read bit 6 = stop bit 7 = start status register (read only) bit 0 = interrupt flag bit 1 = transfer in progress bit 5 = arbitration lost bit 6 = I^2C busy bit 7 = receive acknowledged
0x03 00 00 dc	I^2C data register (lower 8 bits) Write data to send to low byte or read received data from low byte.
0x03 00 00 e0	Over-temperature alarm enable (low bit, 1 = enabled, 0 = disabled)
0x03 00 00 e4	Over-temperature value (low bit, read-only) (1 = over-temperature alarm, 0 = no alarm)
0x03 00 00 e8	Over-temperature destination (low 2 bits) 0 = none 1 = GPIO(14) 2 = GPIO(15) 3 = IRQ(10) The over-temperature sensor triggers at 133 degrees C. It can be viewed at GPIO 14 or 15, or used as an IRQ source to the CPU.
0x03 00 00 f4	Counter/Timer configuration register (lower 4 bits) bit 0 = enable (0 = hold, 1 = count) bit 1 = oneshot (0 = continuous count, 1 = one-shot count) bit 2 = updown (0 = count down, 1 = count up) bit 3 = irq enable (0 = disabled, 1 = trigger IRQ channel 13 on timeout)
0x03 00 00 f8	Counter/Timer current value Set or read the 32-bit current value.
0x03 00 00 fc	Counter/Timer reset value Set or read the 32-bit reset (down-count) or compare (up-count) value.

	minimum	*typical*	*maximum*	*units*
Supply voltage (VDD3V3):	3.0	3.3	3.6	V
Core digital supply voltage (VDD1V8):	1.62	1.8	1.98	V
Junction temperature:	–40	25	175	°C
V_{OH}	$0.8 \cdot$ VDD3V3			V
V_{OL}			0.4	V
Power		94.4		mW
ADC Clock rate:			2	MHz
Temperature alarm		133		°C

Known errors in the efabless Ravenna version 1:

There are no known errors in Ravenna version 1 at this time.