

2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

Regvue
Modern Hardware/Software Interface Documentation

Rob Donnelly NASA-JPL

Josh Geden NASA-JPL



Jet Propulsion Laboratory
California Institute of Technology



Alphabet

Aa Bb Cc Dd

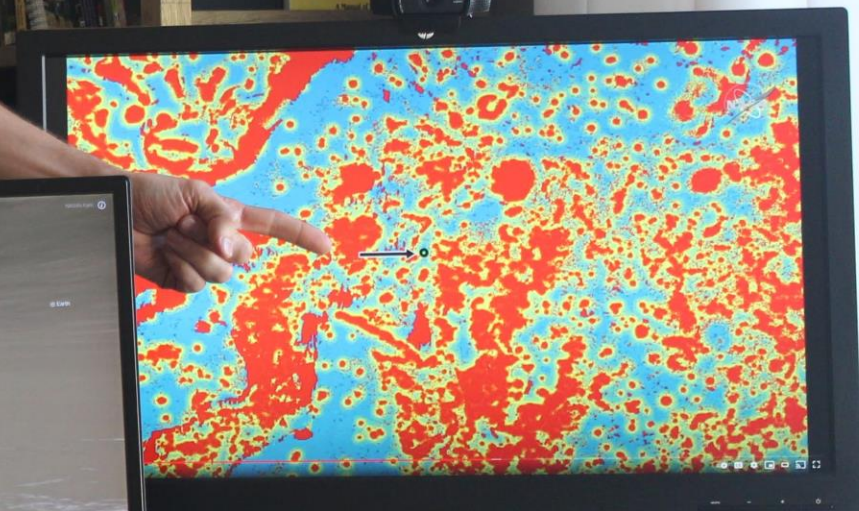
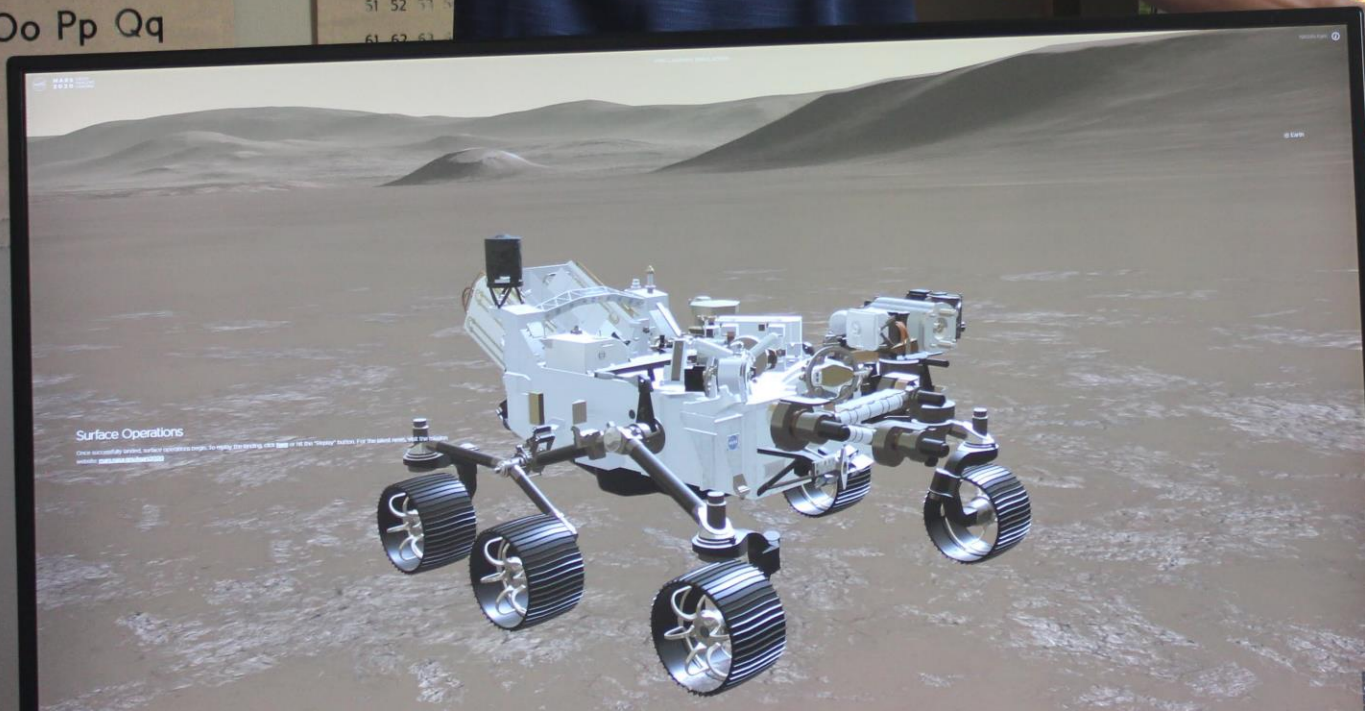
Ee Ff Gg Hh Ii

Jj Kk Ll Mm

Nn Oo Pp Qq

Rr Ss

Ww



A Silly Debug Story

- Manual test of error logic in physical hardware
- Cannot clear the error status bits
- Multiple engineers stumped
- Root cause: mistyped/misread data values
 - E.g. Writing 0x0001000 instead of 0x00010000
 - E.g. Writing 00010030 instead of 0x00010030

A Need Presents

- Humans are error prone
 - Decoding **register** values into **field** values
 - Encoding **field** values into **register** values
 - Decoding **numeric** values into **symbolic** values (enumerations)
 - Byte swapping
- Inefficient information sharing
- Inefficient information access

A Solution Develops

- Modern web technologies
- Static + ***interactive*** content
- Direct permalinks
- Client-side web application
 - No client install required
 - Simple hosting

regvue - SiFive FE310 RISC-V SoC

github.jpl.nasa.gov/pages/regvue/regvue/v1/#/root/uart0/txdata?field=data&data=https://github.jpl.nasa.gov/regvue/demos/raw/main/od...

SiFive FE310 RISC-V SoC v1.0

Search

Ctrl+K

SiFive GitHub

Press Release

clint0x2000000

plic0xc000000

wdog0x10000000

rtc0x10000000

aonclk0x10000000

backup0x10000000

pmu0x10000000

prci0x10008000

otp0x10010000

gpio00x10012000

uart00x10013000

txdata0x10013000

rxdata0x10013004

txctrl0x10013008

rxctrl0x1001300c

ie0x10013010

ip0x10013014

div0x10013018

div_enum0x10013018

qspi00x10014000

pwm00x10015000

i2c00x10016000

Transmit Data FIFO Register - uart0.txdata

0x10013000

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
full	rsvd0																							data							
0	0x0000000																							0x00							
0x00000000																															

Base

Swap

Reset

B

D

H

Byte

Word

POR

Writes to this register push data into the UART transmit FIFO.

Bits	Name	Access	Description
31	full	ro	Indicates whether the transmit FIFO is full. If this bit is 1, writes to this register will be dropped. This bit is updated on write.
30:8	rsvd0	rsvd	Reserved
7:0	data	rw	For writes, the data to push into the FIFO. For reads, the data that was last pushed into the FIFO.

Powered by [regvue v1.1.0](#)



Documentation Comparison



PDF Register Example

24.6.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE**: Bidirectional data mode enable. This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Bit 14 **BIDIOE**: Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.

PDF Register Example

Limited space for
16 bits let alone
32 bits

24.6.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE**: Bidirectional data mode enable. This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected
1: 1-line bidirectional data mode selected

Bit 14 **BIDIOE**: Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode

0: Output disabled (receive-only mode)
1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.

Need to manually
encode/decode field
between field and
register values

Missing
reset value

Missing software
access type

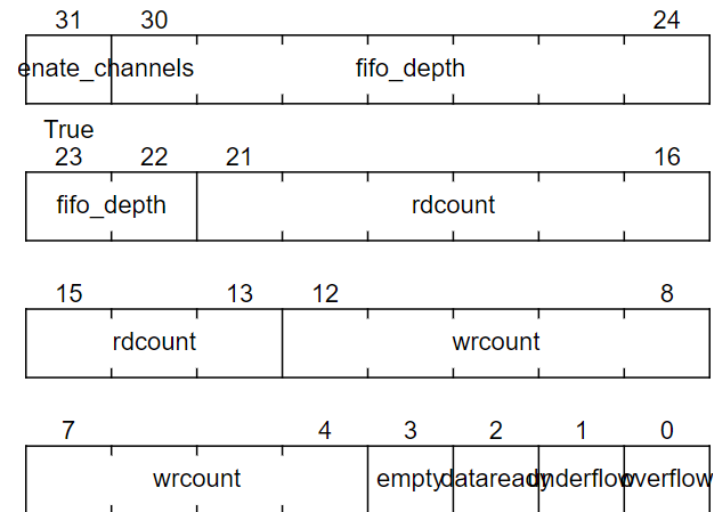
Limited to short,
cryptic identifiers
(presentation > meaning)

HTML Register Example

AUDIO_RX_STAT

Address: $0xf001b000 + 0x10 = 0xf001b010$

Rx data path status



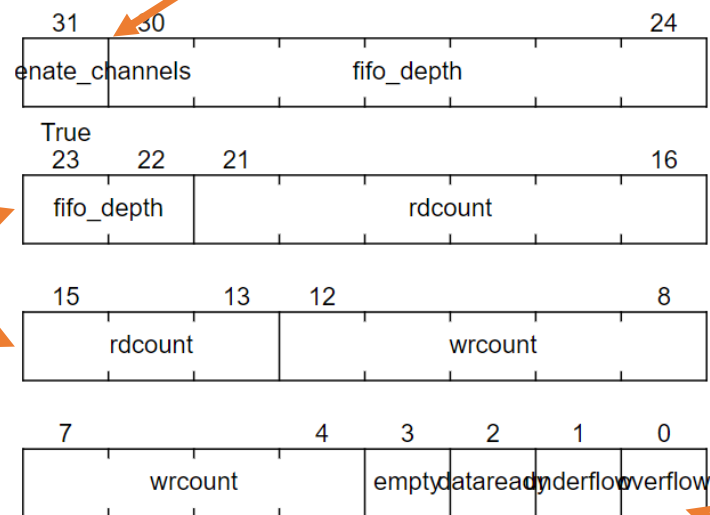
Field	Name	Description
[0]	OVERFLOW	Rx overflow
[1]	UNDERFLOW	Rx underflow
[2]	DATAREADY	256 words of data loaded and ready to read
[3]	EMPTY	No data available in FIFO to read
[12:4]	WRCOUNT	Write count
[21:13]	RDCOUNT	Read count
[30:22]	FIFO_DEPTH	FIFO depth as synthesized
[31]	CONCATENATE_CHANNELS	Receive and send both channels atomically

HTML Register Example

AUDIO_RX_STAT

Address: $0xf001b000 + 0x10 = 0xf001b010$

Rx data path status



Text overflow and truncation
(meaning > presentation)

Missing software
access type

	Name	Description
[0]	OVERFLOW	Rx overflow
[1]	UNDERFLOW	Rx underflow
[2]	DATAREADY	256 words of data loaded and ready to read
[3]	EMPTY	No data available in FIFO to read
[12:4]	WRCOUNT	Write count
[21:13]	RDCount	Read count
[30:22]	FIFO_DEPTH	FIFO depth as synthesized
[31]	CONCATENATE_CHANNELS	Receive and send both channels atomically

Multiple
rows

Text overflow

Regvue Register Example

rx_stat - audio.rx_stat
0xf0019010

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																		
concatenate_channels																fifo_depth										rdcount										wrcount										empty	dataready	underflow	overflow
1																0x000										0x000										0x000										0	0	0	0
0x80000000																																																	

Base Swap

Bits	Name	Access	Description
31	concatenate_channels	rw	Receive and send both channels atomically
30:22	fifo_depth	ro	FIFO depth as synthesized
21:13	rdcount	w1c	Read count
12:4	wrcount	w1c	Write count
3	empty	ro	No data available in FIFO to read
2	dataready	ro	256 words of data loaded and ready to read
1	underflow	w1c	Rx underflow
0	overflow	w1c	Rx overflow

Regvue Register Example

Arbitrary length
identifiers

Full symbolic path

rx_stat - audio.rx_stat
0xf0019010

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																																																
concatenate_channels										fifo_depth										rdcount										wrcount										empty										dataready										underflow										overflow									
1										0x000										0x000										0x000										0										0										0										0									
0x80000000																																																																															

Base Swap
B D H Byte Word

Mouseover
highlighting

Field values

Register value

Sortable

Software
access type

Bits	Name	Access	Description
31	concatenate_channels	rw	Receive and send both channels atomically
30:22	fifo_depth	ro	FIFO depth as synthesized
21:13	rdcount	w1c	Read count
12:4	wrcount	w1c	Write count
3	empty	ro	No data available in FIFO to read
2	dataready	ro	256 words of data loaded and ready to read
1	underflow	w1c	Rx underflow
0	overflow	w1c	Rx overflow

PDF Address Map Example

Bus	Boundary address	Size	Peripheral	Peripheral register map
APB	0x4001 5C00 - 0x4001 7FFF	9 KB	Reserved	-
	0x4001 5800 - 0x4001 5BFF	1 KB	DBGMCU	-
	0x4001 4C00 - 0x4001 57FF	3 KB	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	Section 17.6.16 on page 468
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	Section 17.6.16 on page 468
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	Section 17.5.18 on page 449
	0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	Section 23.7.11 on page 640
	0x4001 3400 - 0x4001 37FF	1 KB	Reserved	-
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1	Section 24.6.8 on page 676
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	Section 13.4.21 on page 294
	0x4001 2800 - 0x4001 2BFF	1 KB	Reserved	-
	0x4001 2400 - 0x4001 27FF	1 KB	ADC	Section 12.11.11 on page 220
	0x4001 1800 - 0x4001 23FF	4 KB	Reserved	-
	0x4001 1400 - 0x4001 17FF	1 KB	USART6	Section 23.7.11 on page 640
	0x4001 0800 - 0x4001 23FF	7 KB	Reserved	-
	0x4001 0400 - 0x4001 07FF	1 KB	EXTI	Section 11.3.7 on page 180
	0x4001 0000 - 0x4001 03FF	1 KB	SYSCFG	Section 9.1.7 on page 151
-	0x4000 8000 - 0x4000 FFFF	32 KB	Reserved	-

PDF Address Map Example

Bus	Boundary address	Size	Peripheral	Peripheral register map
APB	0x4001 5C00 - 0x4001 7FFF	9 KB	Reserved	-
	0x4001 5800 - 0x4001 5BFF	1 KB	DBGMCU	-
	0x4001 4C00 - 0x4001 57FF	3 KB	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	Section 17.6.16 on page 468
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	Section 17.6.16 on page 468
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	Section 17.5.18 on page 449
	0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	Section 23.7.11 on page 640
	0x4001 3400 - 0x4001 37FF	1 KB	Reserved	-
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1	Section 24.6.8 on page 676
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	Section 13.4.21 on page 294
	0x4001 2800 - 0x4001 2BFF	1 KB	Reserved	-
	0x4001 2400 - 0x4001 27FF	1 KB	ADC	Section 12.11.11 on page 220
	0x4001 1800 - 0x4001 23FF	4 KB	Reserved	-
	0x4001 1400 - 0x4001 17FF	1 KB	USART6	Section 23.7.11 on page 640
	0x4001 0800 - 0x4001 23FF	7 KB	Reserved	-
	0x4001 0400 - 0x4001 07FF	1 KB	EXTI	Section 11.3.7 on page 180
	0x4001 0000 - 0x4001 03FF	1 KB	SYSCFG	Section 9.1.7 on page 151
-	0x4000 8000 - 0x4000 FFFF	32 KB	Reserved	-

Verbose

Additional details elsewhere

Regvue Address Map Example

► pwr 0x40007000

► i2c1 0x40005400

i2c2 0x40005800

► iwdg 0x40003000

► wwdg 0x40002c00

► tim1 0x40012c00

► tim3 0x40000400

► tim14 0x40002000

► tim6 0x40001000

► exti 0x40010400

► nvic 0xe000e100

► dma 0x40020000

► rcc 0x40021000

► syscfg 0x40010000

▼ adc 0x40012400

isr 0x40012400

ier 0x40012404

cr 0x40012408

cfgr1 0x4001240c

cfgr2 0x40012410

cfgr1 - adc.cfgr1

0x4001240c

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
rsvd3	awdch					rsvd2	awden	awdsgl	rsvd1				discen	autoff	autdly	cont	ovrmod	exten	rsvd0	extsel		align	res	scandir	dmacfg	dmaen					
0	0x02					0x0	0	0	0x00				0	1	0	1	0	0x0	0	0x0		0	0x0	0	0	0	0	0	0	0	
0x0800A000																															

Base Swap Reset

configuration register 1

Bits ↑	Name	Access	Description
31	rsvd3	rsvd	Reserved
30:26	awdch	rw	Analog watchdog channel selection
25:24	rsvd2	rsvd	Reserved

Regvue Address Map Example

Simple
address map

Hierarchical

Map and register
side-by-side

▶ pwr	0x400007000
▶ i2c1	0x400005400
i2c2	0x400005800
▶ iwdg	0x400003000
▶ wwdg	0x400002c00
▶ tim1	0x400012c00
▶ tim3	0x400000400
▶ tim14	0x400002000
▶ tim6	0x400001000
▶ exti	0x400000000
▶ rcc	0x40021000
▶ syscfg	0x40010000
▼ adc	0x40012400
isr	0x40012400
ier	0x40012404
cr	0x40012408
cfgr1	0x4001240c
cfgr2	0x40012410

cfgr1 - adc.cfgr1
0x4001240c

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	
rsvd3	awdch					rsvd2		awden	awdsgl	rsvd1					discen	autoff	autdly	cont	ovrmod	exten		rsvd0	extsel			align	res		scandir	dmacfg	dmaen	
0	0x02					0x0		0	0	0x00					0	1	0	1	0	0x0	0	0x0		0	0x0		0	0x0	0	0	0	0
0x0800A000																																

Swap
Byte Word

Reset

Bits ↕	Name	Access	Description
31	rsvd3	rsvd	Reserved
30:26	awdch	rw	Analog watchdog channel selection
25:24	rsvd2	rsvd	Reserved



Feature Highlights



Incremental Search

regvue - Multi-SoC Example x +

github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/?data=https://github.jpl.nasa.gov/regvue/demos/raw/main/Various/all.json

Multi-SoC Example (v1.0) Search Ctrl+K GitHub

Multi-SoC Example

[GitHub](#)

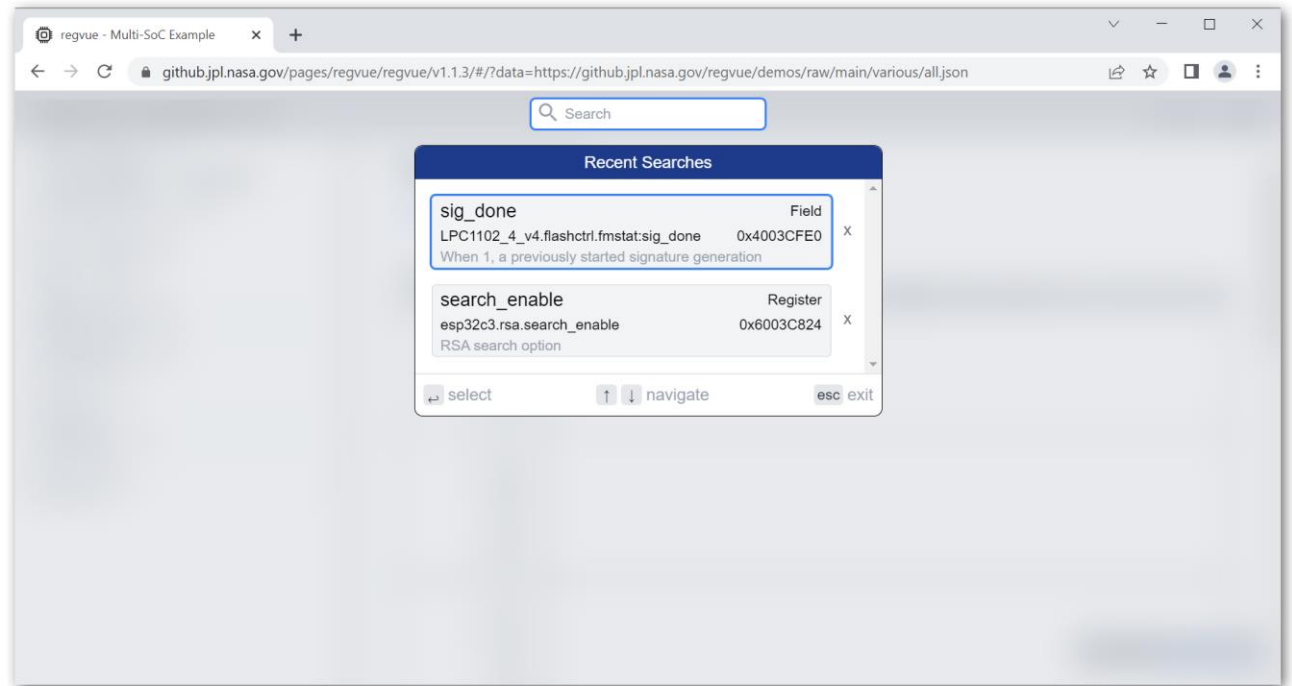
Sub-Elements Map

Offset	Name	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
	ARM_Sample																																
	AT91SAM9CN11																																
	432G200F16																																

Powered by [regvue v1.1.3](#)

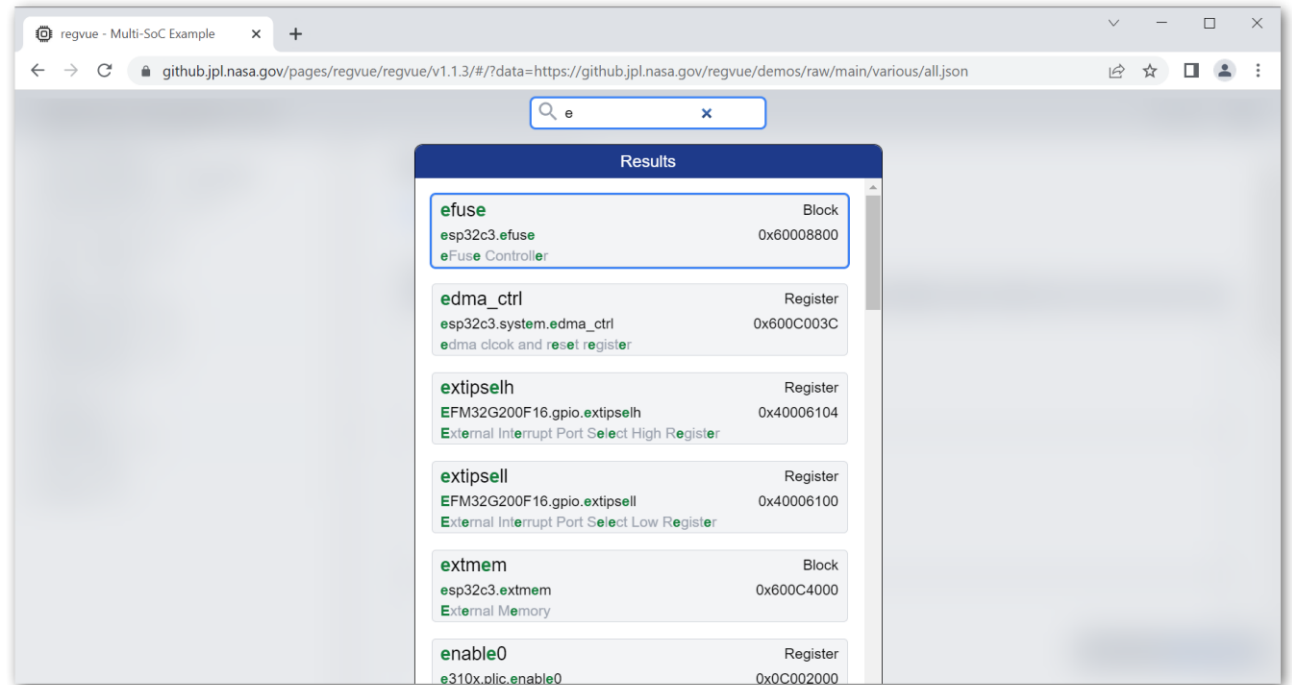
Incremental Search

- Press “Ctrl + k”



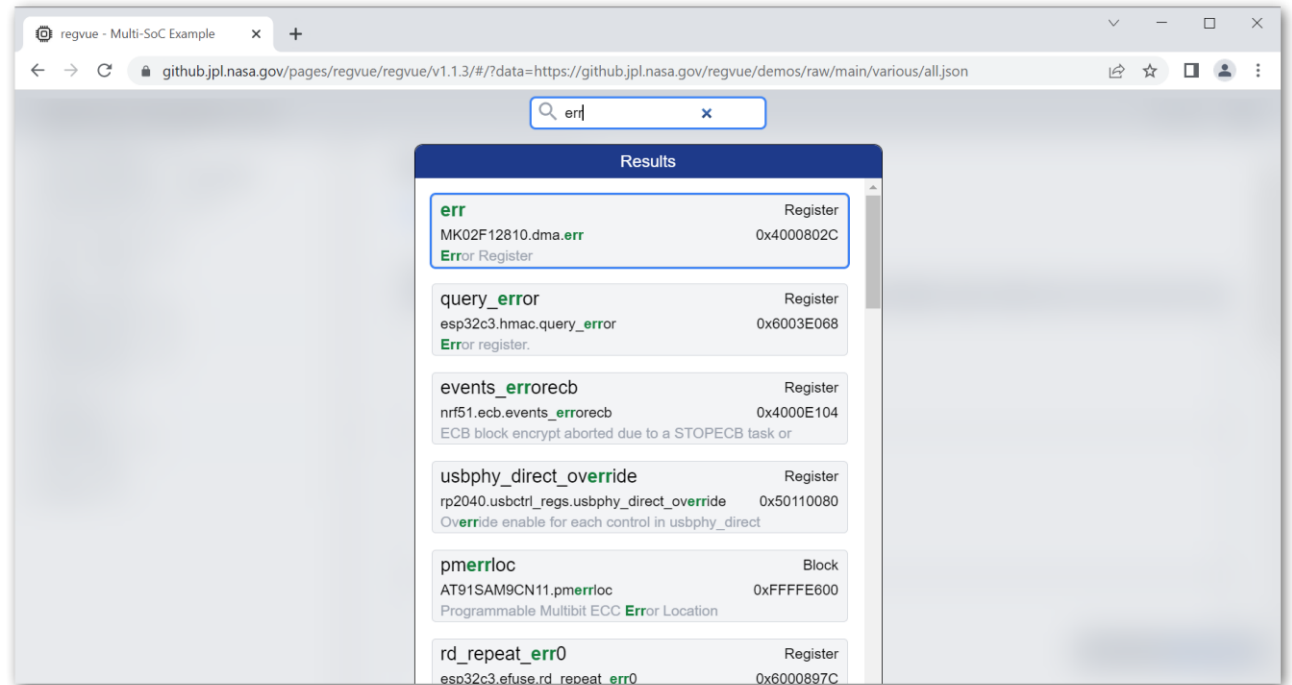
Incremental Search

- Press “Ctrl + k”
- Type “e”



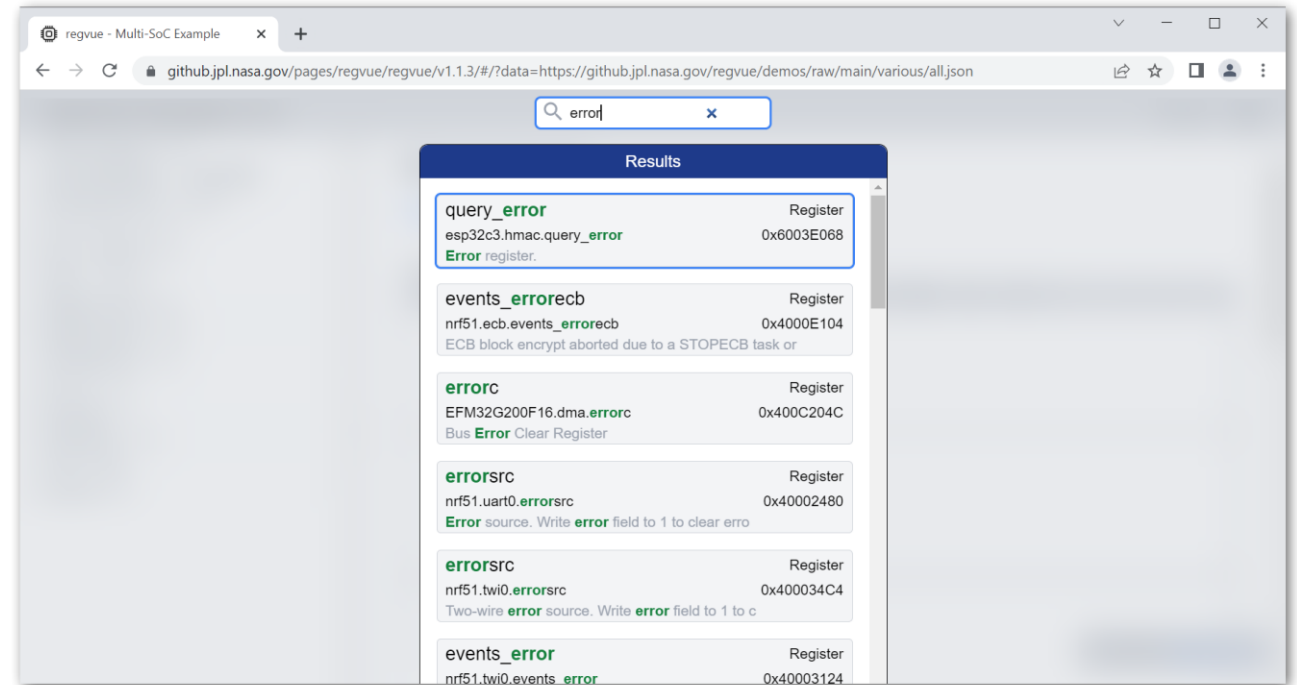
Incremental Search

- Press “Ctrl + k”
- Type “e”
- Type “rr”



Incremental Search

- Press “Ctrl + k”
- Type “e”
- Type “rr”
- Type “or”



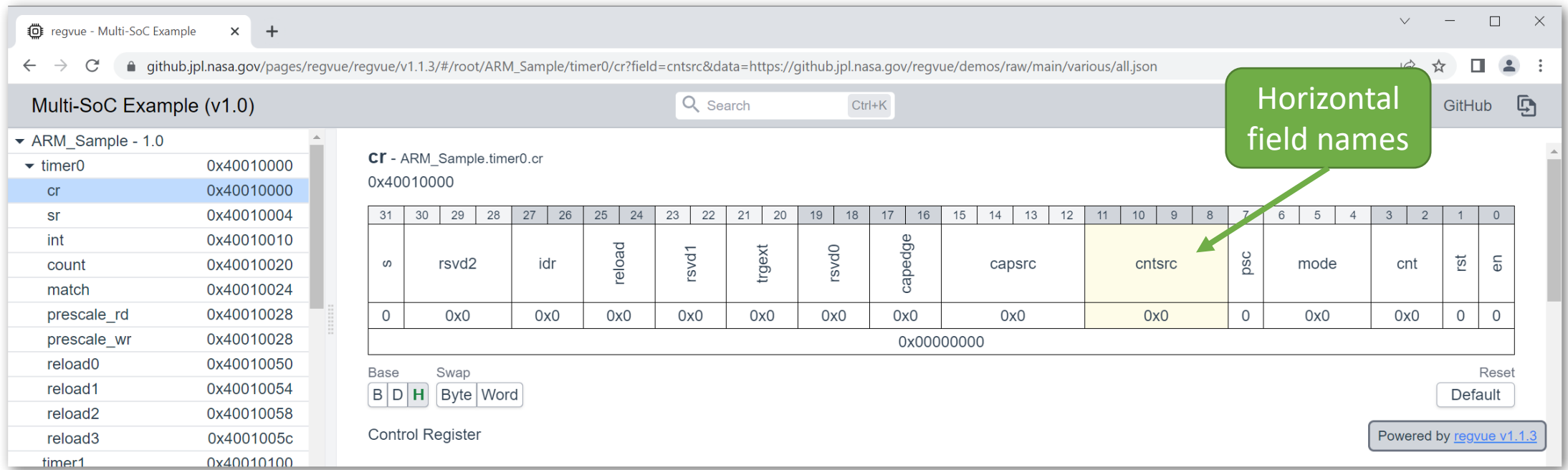
Incremental Search

- Press “Ctrl + k”
- Type “e”
- Type “rr”
- Type “or”
- Press “Enter”

The screenshot shows a web browser window titled "regvue - Multi-SoC Example". The address bar shows a GitHub link. The page title is "Multi-SoC Example (v1.0)". A search bar with "Ctrl+K" is visible. On the left, a list of registers is shown, with "query_error" selected. The main area displays the "query_error" register details, including its address "0x6003e068" and a bitfield diagram. The bitfield diagram shows bits 31:1 and 0, with "rsvd0" and "query_check" fields. Below the bitfield, there are buttons for "Base", "Swap", "Byte", "Word", and "Reset". A table titled "Error register." is shown, with columns "Bits", "Name", "Access", and "Description". The table has two rows: "31:1" for "rsvd0" (Reserved) and "0" for "query_check" (Hmac configuration state). A footer note says "Powered by regvue v1.1.3".

Bits	Name	Access	Description
31:1	rsvd0	rsvd	Reserved
0	query_check	read-only	Hmac configuration state. 0: key are agree with purpose. 1: error

Responsive Design



The screenshot displays the regvue Multi-SoC Example interface in a web browser. The left sidebar shows a tree view of the device components, with 'ARM_Sample - 1.0' expanded and 'timer0' selected. The 'cr' register is highlighted. The main area shows the 'cr' register details, including its address (0x40010000) and a bitfield table. A green callout box labeled 'Horizontal field names' points to the field names in the bitfield table. A black arrow points to the left sidebar.

Multi-SoC Example (v1.0)

Search Ctrl+K

Horizontal field names

cr - ARM_Sample.timer0.cr
0x40010000

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
s	rsvd2			idr	reload	rsvd1		trgext		rsvd0		capedge		capsrc			cntsrc			psc	mode			cnt		rst	en				
0	0x0			0x0	0x0	0x0		0x0		0x0		0x0		0x0			0x0			0	0x0			0x0		0	0		0		
0x00000000																															

Base Swap
B D H Byte Word

Control Register

Reset
Default

Powered by [regvue v1.1.3](#)

Responsive Design

regvue - Multi-SoC Example

github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr...

Multi-SoC Example (v1.0)

Search Ctrl+K

cr - ARM_Sample.timer0.cr
0x40010000

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
s	rsvd2				idr		reload		rsvd1		trgext		rsvd0		capedge		capsrc		cntsrc		psc		mode		cnt		rst		en		
0	0x0				0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0		0x0		0x0		0		0		
0x00000000																															

Base Swap
B D H Byte Word

Reset
Default

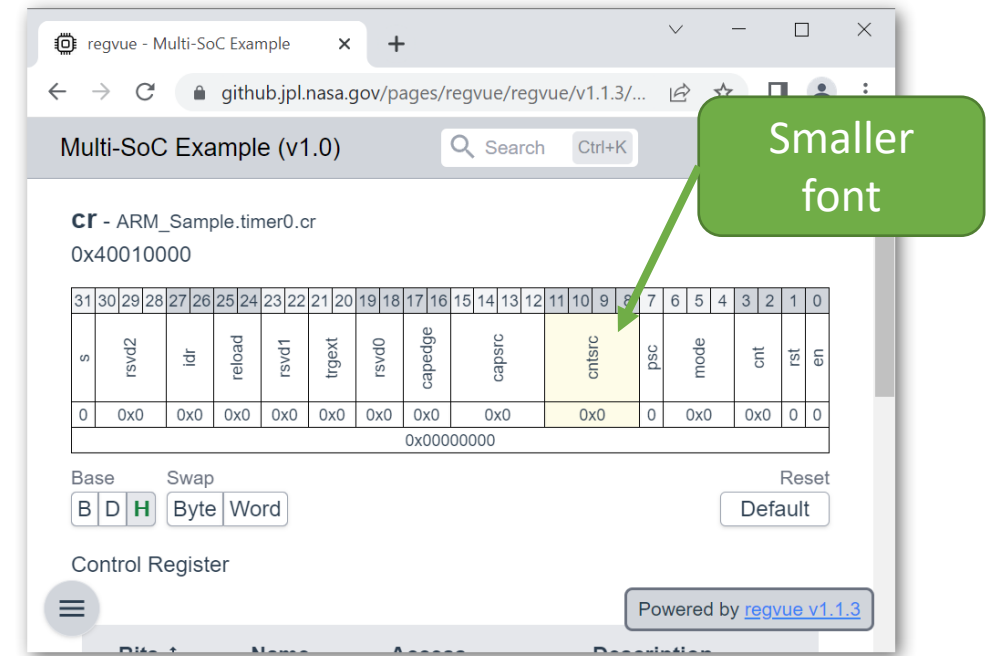
ntrol Register

Powered by [regvue v1.1.3](#)

Vertical field names

Address map hidden

Responsive Design



Encode/Decode

regvue - Multi-SoC Example

github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://gith...

Multi-SoC Example (v1.0)

Search Ctrl+K

GitHub

CR - ARM_Sample.timer0.cr
0x40010000

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
s	rsvd2				idr	reload	rsvd1		trgext	rsvd0		capedge		capsrc				cntsrc				psc	mode		cnt		rst	en			
0	0x0				0x0	0x0	0x0		0x0	0x0		0x0		0x0				0x0				0	0x0		0x0		0	0			
0x00000000																															

Base Swap

B D **H** Byte Word

Reset Default

Control Register

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition
23:22	rsvd1	rsvd	Reserved

Powered by [regvue v1.1.3](#)

Encode/Decode

- Select “psc” field value

The screenshot shows the regvue web interface for a Multi-SoC Example. The browser address bar shows the URL: [github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github...](https://github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github.com/nasa/regvue)

The page title is "Multi-SoC Example (v1.0)". The register being viewed is "CR - ARM_Sample.timer0.cr" with address "0x40010000".

The register fields are displayed in a table with bit positions 31 down to 0:

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		
s	rsvd2				idr		reload		rsvd1		trgext		rsvd0		capedge		capsrc				cntsrc				psc	mode				cnt		rst	en
0	0x0				0x0		0x0		0x0		0x0		0x0		0x0		0x0				0x0				0	0x0				0x0		0	0
0x00000000																																	

Below the register fields, there are controls for "Base" (B, D, H), "Swap" (Byte, Word), and a "Reset" button labeled "Default".

The "Control Register" section contains a table with the following data:

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition
23:22	rsvd1	rsvd	Reserved

A small notification at the bottom right says "Powered by [regvue v1.1.3](#)".

Encode/Decode

- Select “psc” field value
- Type “1”

The screenshot shows the regvue web interface for a Multi-SoC Example. The browser address bar shows the URL: [github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github...](https://github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github.com/nasa/regvue/blob/master/v1.1.3/root/ARM_Sample/timer0/cr)

The page title is "Multi-SoC Example (v1.0)". Below the title, the register is identified as "CR - ARM_Sample.timer0.cr" with address "0x40010000".

A 32-bit register field diagram is shown with bit positions 31 down to 0. The fields are: s (bit 31), rsvd2 (bits 30-28), idr (bits 27-26), reload (bits 25-24), rsvd1 (bits 23-22), trgtxt (bits 21-20), rsvd0 (bits 19-18), capedge (bits 17-16), capsrc (bits 15-14), cntsrc (bits 13-12), psc (bit 7), mode (bits 6-5), cnt (bits 4-3), rst (bit 2), and en (bit 0). The psc field (bit 7) is highlighted with a green box containing the value "1". Below the field diagram, the register value is displayed as "0x00000080".

Below the register value, there are controls for "Base" (B, D, H), "Swap" (Byte, Word), and a "Reset" button labeled "Default".

The "Control Register" section contains a table with the following data:

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition
23:22	rsvd1	rsvd	Reserved

A green callout box with an arrow pointing to the psc field contains the text: "Register value updated as-you-type".

At the bottom right, a small badge says "Powered by [regvue v1.1.3](#)".

Encode/Decode

- Select “psc” field value
- Type “1”
- Select register value

The screenshot shows the regvue web interface for a Multi-SoC Example. The browser address bar shows the URL: github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github.com/nasa/regvue. The page title is "Multi-SoC Example (v1.0)".

The main content area displays the register configuration for **CR - ARM_Sample.timer0.cr** at address **0x40010000**. A 32-bit register field is shown with bit positions 31 down to 0. The fields are:

Bit	Field	Value
31	s	0
30:28	rsvd2	0x0
27:26	idr	0x0
25:24	reload	0x0
23:22	rsvd1	0x0
21:20	trgext	0x0
19:18	rsvd0	0x0
17:16	capedge	0x0
15:14	capsrc	0x0
13:12	cntsrc	0x0
11:10	psc	1
9:8	mode	0x0
7:6	cnt	0x0
5:4	rst	0
3:2	en	0

The register value is displayed as **0x00000080**. Below the register field, there are controls for Base (B, D, H), Swap (Byte, Word), and a Reset button. The Control Register table is also shown.

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition
23:22	rsvd1	rsvd	Reserved

Powered by [regvue v1.1.3](#)

Encode/Decode

- Select “psc” field value
- Type “1”
- Select register value
- Press “Delete”

regvue - Multi-SoC Example

github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://gith...

Multi-SoC Example (v1.0)

Search Ctrl+K

GitHub

CR - ARM_Sample.timer0.cr
0x40010000

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
s	rsvd2		idr		reload		rsvd1		trgext		rsvd0		capedge		capsrc		cntsrc		psc		mode		cnt		rst		en				
0	0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		0x0		1		0x0		0x0		0		0				

Base Swap
B D H Byte Word

Control Register

Empty value

Reset Default

Field	Access	Description
31:0	read-write	Starts and Stops the Timer / Counter
30:28	rsvd	Reserved
27:26	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	read-write	Select RELOAD Register n to reload Timer on condition
23:22	rsvd	Reserved

Powered by [regvue v1.1.3](#)

Encode/Decode

- Select “psc” field value
- Type “1”
- Select register value
- Press “Delete”
- Type “0x”

The screenshot shows the regvue web interface for a Multi-SoC Example. The browser address bar shows the URL: github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github.com/nasa/regvue. The page title is "Multi-SoC Example (v1.0)".

The main content area displays the configuration for the **CR - ARM_Sample.timer0.cr** register, located at address **0x40010000**. A 32-bit register field is shown with bit positions 31 down to 0. The fields are:

Bit	Field	Value
31	s	0
30:28	rsvd2	0x0
27:26	idr	0x0
25:24	reload	0x0
23:22	rsvd1	0x0
21:20	trgext	0x0
19:18	rsvd0	0x0
17:16	capedge	0x0
15:14	capsrc	0x0
13:12	cntsrc	0x0
11:10	psc	0
9:8	mode	0x0
7:6	cnt	0x0
5:4	rst	0
3:2	en	0

Below the register field, there is a "Base" section with buttons for "B", "D", and "H" (selected). A "Swap" section has buttons for "Byte" and "Word". A "Reset" button is labeled "Default". A red arrow points to the "psc" field with the text "Expected a value to follow base encoding".

Below the register field, there is a "Control Register" table:

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition
23:22	rsvd1	rsvd	Reserved

At the bottom right, there is a "Powered by regvue v1.1.3" logo.

Encode/Decode

- Select “psc” field value
- Type “1”
- Select register value
- Press “Delete”
- Type “0x”
- Type “8002”

The screenshot shows the regvue web interface for the Multi-SoC Example. The browser address bar shows the URL: github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github.com/nasa/regvue. The page title is "Multi-SoC Example (v1.0)".

The register configuration is for **CR - ARM_Sample.timer0.cr** at address **0x40010000**. The register is shown as a 32-bit field with bit positions 31 down to 0. The fields are:

Bit	Field	Value
31	s	0
30:28	rsvd2	0x0
27:26	idr	0x0
25:24	reload	0x0
23:22	rsvd1	0x0
21:20	trgext	0x0
19:18	rsvd0	0x0
17:16	capedge	0x0
15:14	capsrc	0x8
13:12	cntsrc	0x0
11:10	psc	0
9:8	mode	0x0
7:6	cnt	0x0
5:4	rst	1
3:2	en	0

The register value is displayed as **0x8002**. Below the register value, there are controls for Base (B, D, H), Swap (Byte, Word), and a Reset button. The text "Control Register" is also present.

A green callout box with arrows pointing to the 'psc' and 'rst' fields contains the text: "Fields values updated as-you-type".

Below the register configuration, there is a table with the following columns: Bits, Name, Access, and Description.

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is modified
25:24	reload	read-write	Select RELOAD Register n to reload timer on condition
23:22	rsvd1	rsvd	Reserved

The page is powered by [regvue v1.1.3](#).

Encode/Decode

- Select “psc” field value
- Type “1”
- Select register value
- Press “Delete”
- Type “0x”
- Type “8002”
- Type “_abcd”

The screenshot shows the regvue web interface for a Multi-SoC Example. The browser address bar shows the URL: [github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github...](https://github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?data=https://github.com/nasa/regvue/blob/v1.1.3/ARM_Sample/timer0/cr)

The page title is "Multi-SoC Example (v1.0)". Below the title, there is a search bar and a "Ctrl+K" button. The main content area displays the register configuration for "ARM_Sample.timer0.cr" at address "0x40010000".

The register configuration is shown as a table with 32 bits (31 to 0) and their corresponding fields:

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			
s	rsvd2				idr		reload		rsvd1		trgext		rsvd0		capedge		capsrc				cntsrc				psc		mode		cnt		rst		en	
1	0x0				0x0		0x0		0x0		0x0		0x0		0x2		0xA				0xB				1		0x4		0x3		0		1	

Below the table, there is a text input field containing "0x8002_abcd".

At the bottom, there is a "Control Register" section with a table of fields:

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition
23:22	rsvd1	rsvd	Reserved

At the bottom right, there is a "Powered by regvue v1.1.3" badge.

Direct Permalinks

The screenshot shows the regvue web interface for a Multi-SoC Example (v1.0). The left sidebar lists various components, including 'ARM_Sample - 1.0' and 'timer0'. The main area displays the 'cr' register (ARM_Sample.timer0.cr) at address 0x40010000. A green callout box with the text 'Shareable direct link to this field' points to the 'idr' field in the register's bit field diagram. The bit field diagram shows fields: s (31), rsvd2 (30:28), idr (27:26), reload (25:24), rsvd1 (23:22), target (21:20), rsvd0 (19:18), cntsrc (11:10), psc (7), mode (6:5), cnt (3:2), rst (1), and en (0). Below the diagram, there are tabs for Base (B, D, H), Swap (Byte, Word), and a Reset button. A table at the bottom provides a detailed description of the register fields.

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition

Powered by [regvue v1.1.3](#)

Direct Permalinks

The screenshot shows the regvue web application interface. The browser address bar displays the URL: `github.jpl.nasa.gov/pages/regvue/regvue/v1.1.3/#/root/ARM_Sample/timer0/cr?field=idr&data=https://github.jpl.nasa.gov/regvue/demos/raw/main/various/all.json`. The left sidebar shows a tree view of the hardware components, with `ARM_Sample - 1.0` expanded and `timer0` selected. The main content area displays the register `cr - ARM_Sample.timer0.cr` at address `0x40010000`. A bitfield table shows the register's structure with fields `s`, `rsvd2`, `idr`, `reload`, `rsvd1`, `rgext`, `rsvd0`, `capedge`, `capsrc`, `cntsrc`, `psc`, `mode`, `cnt`, `rst`, and `en`. The `idr` field is highlighted, and a callout points to its value `0x0`. Another callout points to the `Register description JSON` in the URL. A third callout points to the `Register path` in the sidebar. A fourth callout points to the `Regvue application` header. Below the bitfield table, a table provides a detailed description of the fields.

Bits	Name	Access	Description
31	s	read-write	Starts and Stops the Timer / Counter
30:28	rsvd2	rsvd	Reserved
27:26	idr	read-write	Selects, if Reload Register number is incremented, decremented or not modified
25:24	reload	read-write	Select RELOAD Register n to reload Timer on condition

Powered by [regvue v1.1.3](#)

JSON Input Format

Schema Documentation

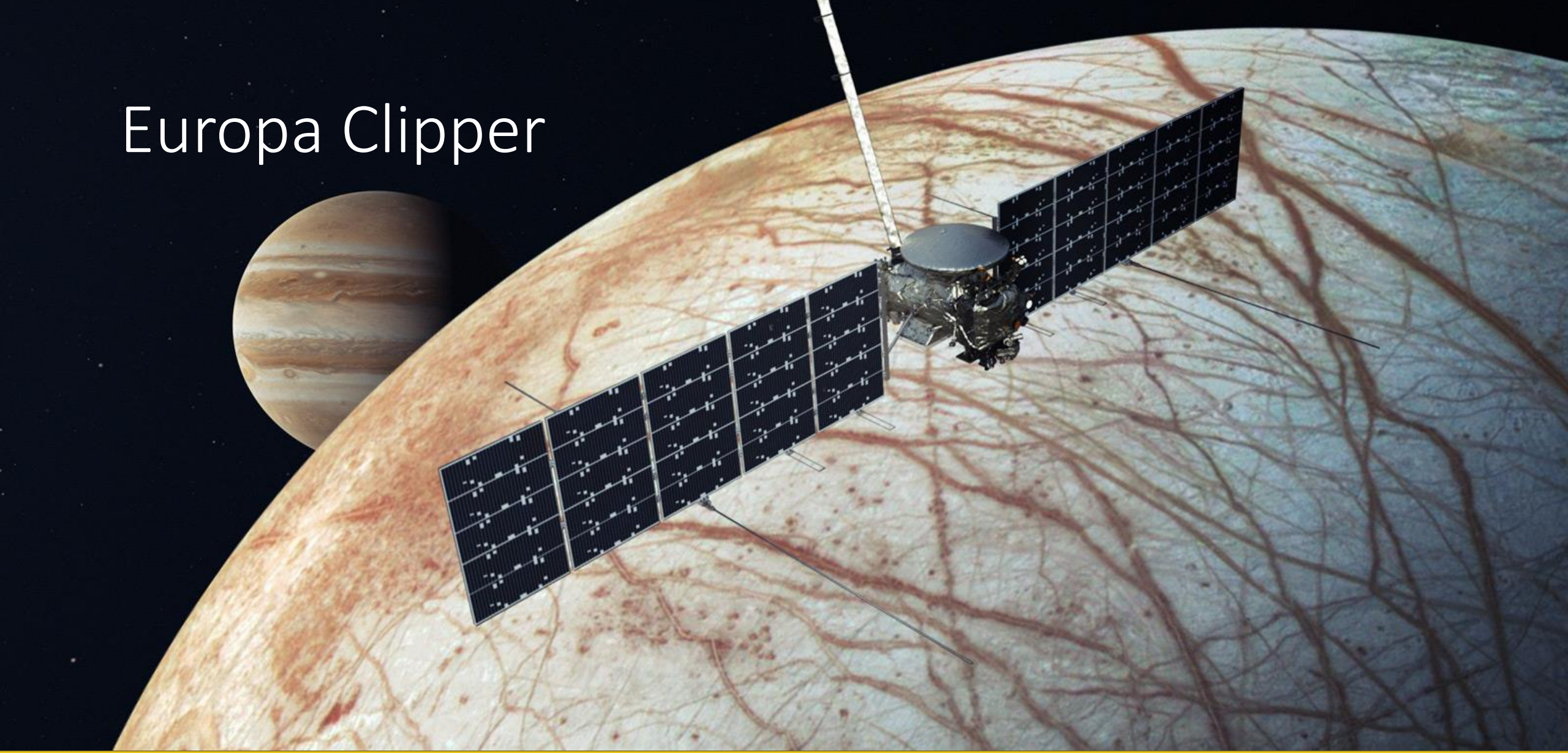
Table of Contents

- 1. Document Object
- 2. Schema Object
- 3. Element IDs
 - └ 3.1. Dot Notation
- 4. Root Object
 - └ 4.1. Root Object Example
- 5. Elements Object
 - └ 5.1. Elements Object Example
- 6. Element Objects
 - └ 6.1. BlockElement Objects
 - └ 6.1.1. Link Objects
 - └ 6.1.2. BlockElement Object Example
 - └ 6.2. RegisterElement Objects
 - └ 6.2.1. Field Objects
 - └ 6.2.1.1. Reset Objects
 - └ 6.2.1.2. EnumValue Objects
 - └ 6.2.2. RegisterElement Object Example
 - └ 6.3. MemoryElement Objects
 - └ 6.4. IncludeElement Objects
 - └ 6.4.1. Include Semantics

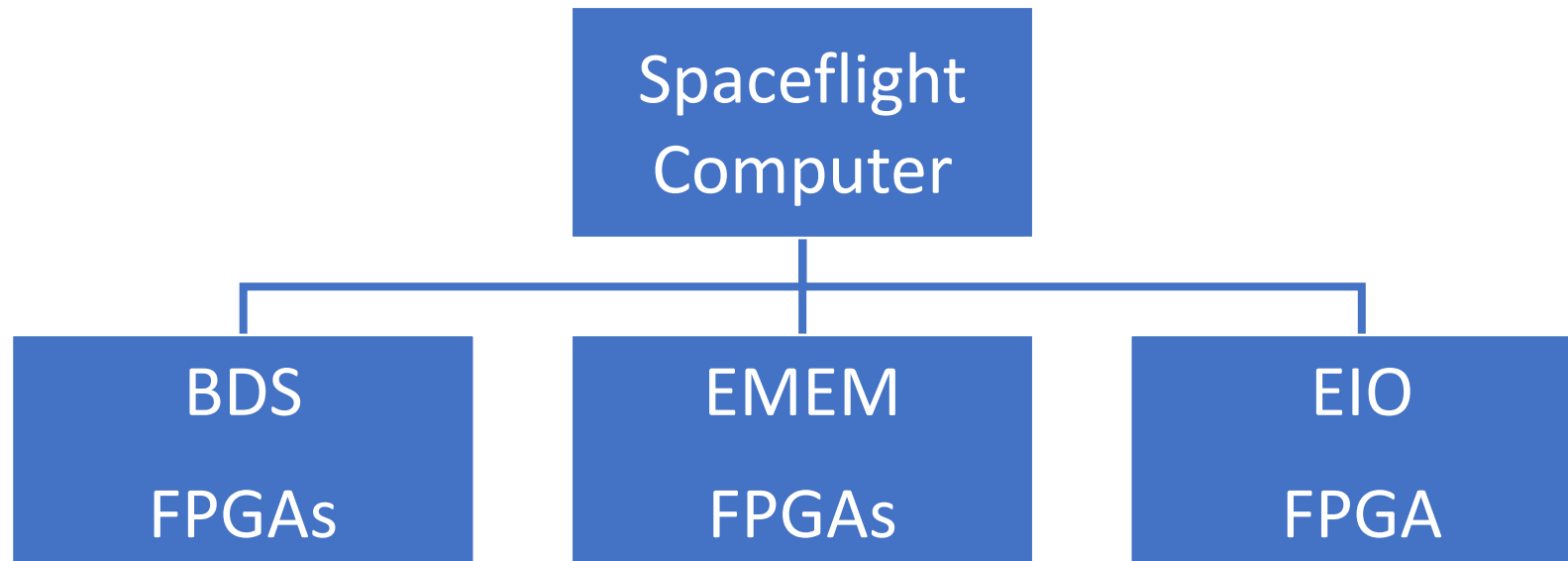
Example JSON

```
{
  "schema": { "name": "register-description-format", "version": "v1" },
  "root": {
    "desc": "Example Design", "version": "v1.0",
    "links": [ { "text": "GitHub", "href": "https://github.com/org/repo" } ],
    "children": [ "blk0" ]
  },
  "elements": {
    "blk0": {
      "type": "blk", "id": "blk0", "name": "blk0",
      "children": [ "blk0.reg0", "blk0.reg1" ]
    },
    "blk0.reg0": {
      "id": "blk0.reg0", "type": "reg", "name": "reg0", "offset": "0", "doc": "An example register.",
      "fields": [
        { "name": "rsvd", "lsb": 5, "nbits": 28, "access": "rw", "doc": "A reserved field." },
        { "name": "command", "lsb": 0, "nbits": 4, "access": "rw", "doc": "The command to execute." }
      ]
    },
    "blk0.reg1": {
      "type": "reg", "id": "blk0.reg1", "name": "reg1", "data_width": 16, "offset": "4",
      "fields": [
        { "name": "f0", "lsb": 0, "nbits": 16, "access": "ro", "reset": "0" }
      ]
    }
  }
}
```


Europa Clipper

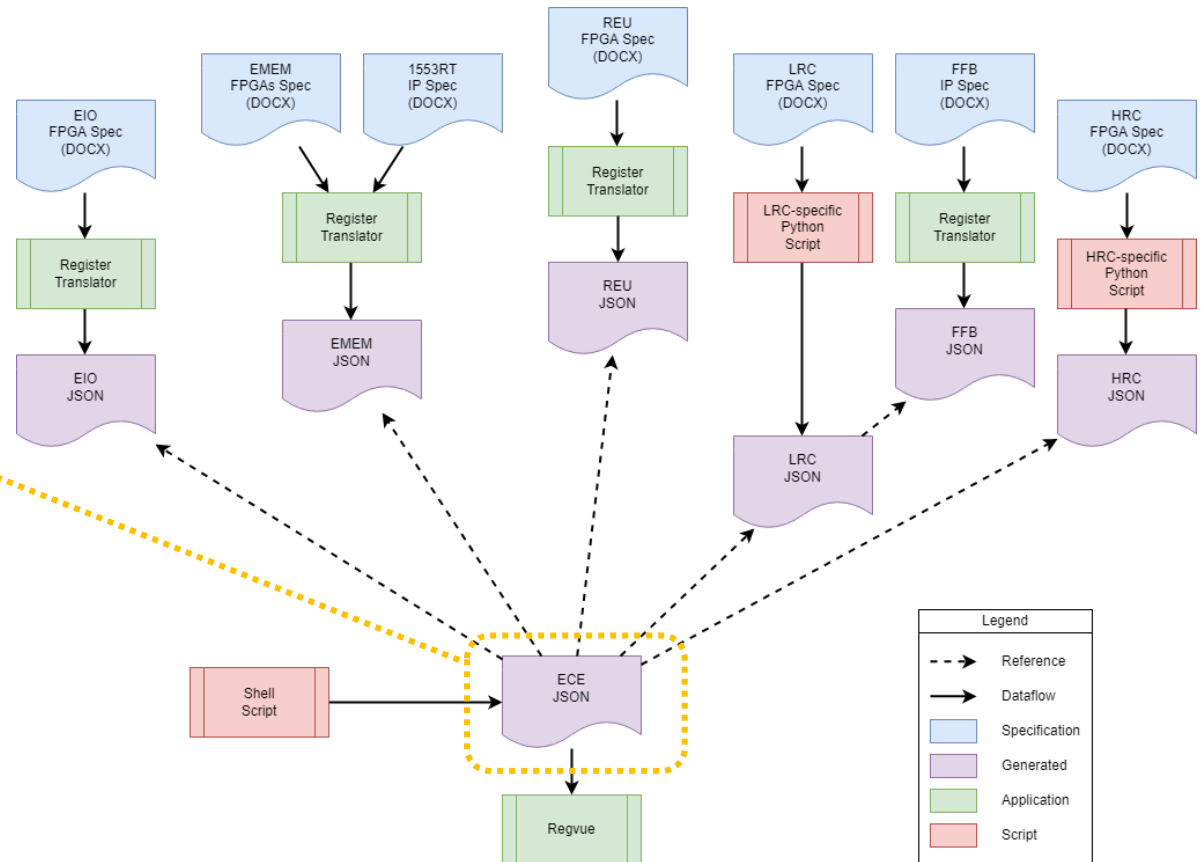


Europa Clipper Avionics

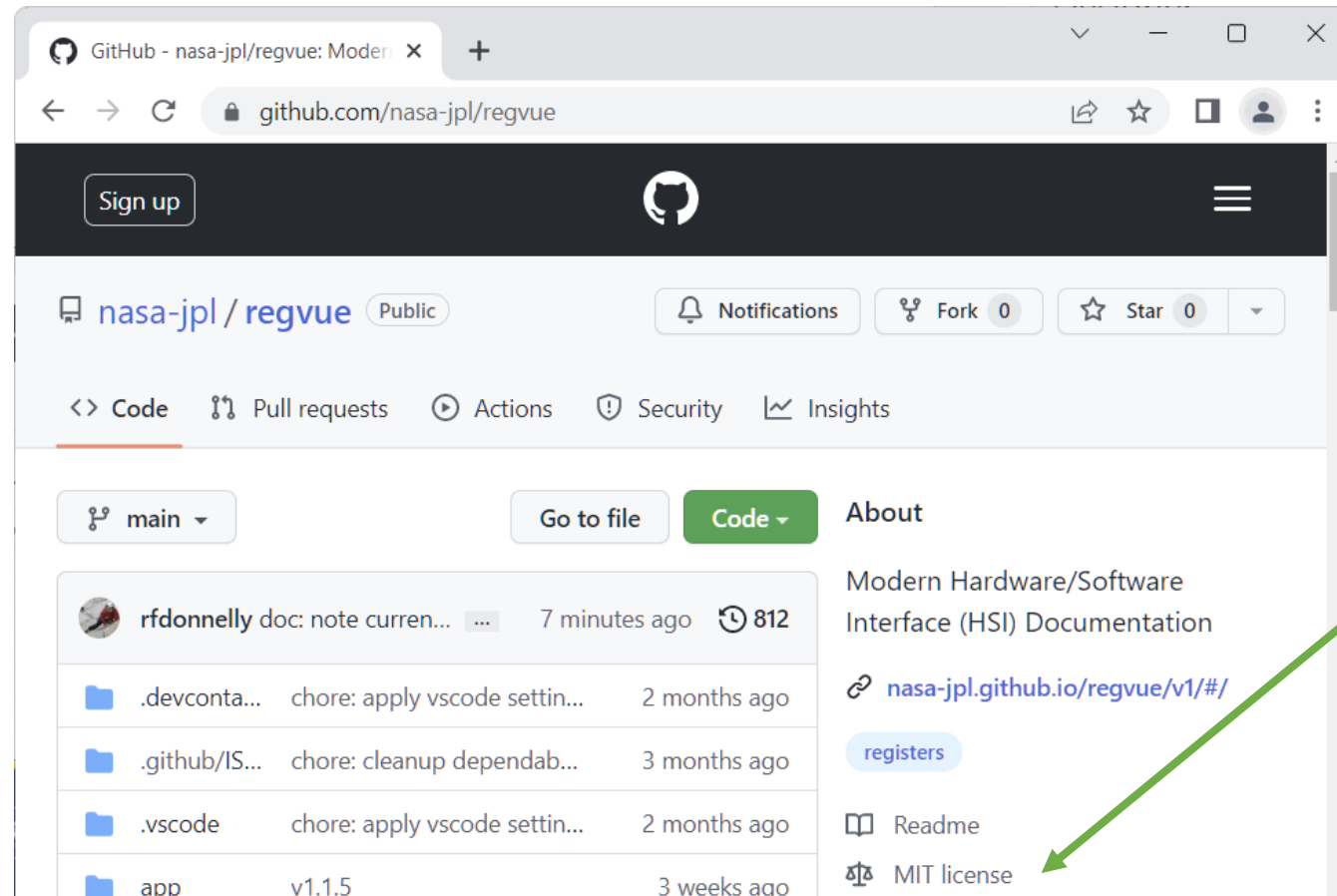


Europa Clipper Regvue JSON

- Convert several Microsoft Word documents to several Regvue JSONs
- Hand created top-level JSON includes other JSONs



Open Source @ <https://github.com/nasa-jpl/regvue>



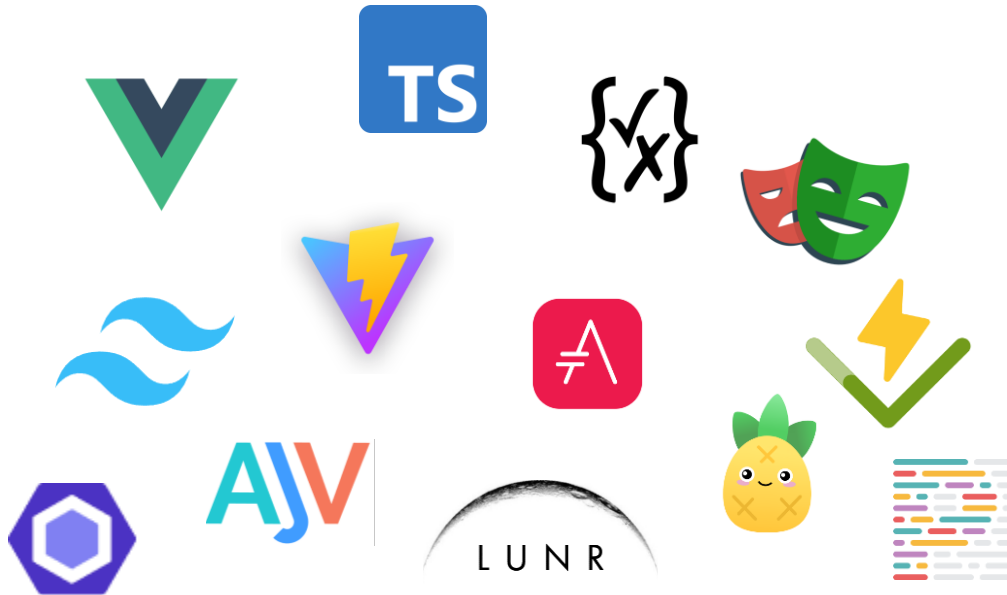
Permissive
license

Summary

- Modern hardware/software interface documentation
- Increases efficiency
 - Direct link to specific design elements
 - Incremental search
- Reduces errors
 - Field/register decode/encode
- Open source

Questions

Open source technologies
used



Funding provided
by

