EduSoC v1.0

Minimalistic RISC-V-oriented SoC for educational purposes

Alexander Kharitonov
University of Stuttgart, ITI

12th July 2023

Overview

EduSoC (Educational System on Chip) is an FPGA-based, RISC-V-oriented system-on-chip infrastructure. It is designed for educational purposes, offering minimal complexity and consisting of a small set of essential modules and peripherals:

- Clock and reset signal generation
- Boot ROM (size and speed configurable)
- Block RAM (size and speed configurable)
- SoC controller with interrupt engine supporting up to 32 interrupts
- GPIO module with change notification interrupt, supporting up to 512 GPIO pins
- Timer module with up to 16 independent timers with configurable periods and interrupts
- PWM module with up to 16 independent pulse width modulated outputs
- VGA controller with 8-bit color framebuffer (size and speed configurable)
- Serial UART bridge for external memory programming and SoC/peripheral control
- Crossbar bus interconnect

In its default configuration, EduSoC is intended for use with a Digilent Arty S7 board and offers a Verilog-compatible module for use with this board (`edusoc_basic`, see Appendix A). However, EduSoC can be configured for any hardware platform, and more peripherals can be added to EduSoC with relative ease.

EduSoC does not target or contain any specific RISC-V core. A simple core interface is provided.

Contents

1. System Structure **4**

2. Memory Map **5**

3. Interfaces **6**
3.1. Memory Bus . . . 6
3.2. Interrupt Bus . . . 7
3.3. SoC Interface . . . 8

4. Interrupts **10**

5. UART Bridge **11**

6. VGA Controller **12**

7. Peripherals **13**
7.1. SoC Control . . . 14
7.1.1. `SOCCON_CONTROL` Register . . . 14
7.1.2. `SOCCON_INT_EN` Register . . . 15
7.1.3. `SOCCON_INT_FLAGS` Register . . . 16
7.2. GPIO . . . 17
7.2.1. `GPIO_PORT_`i Register . . . 18
7.2.2. `GPIO_LATCH_`i Register . . . 18
7.2.3. `GPIO_DIR_`i Register . . . 19
7.2.4. `GPIO_CNR_`i Register . . . 19
7.2.5. `GPIO_CNF_`i Register . . . 20
7.2.6. `GPIO_CN_STATE_`i Register . . . 20
7.2.7. `GPIO_INT_STATUS` Register . . . 21
7.3. Timer . . . 22
7.3.1. `TIMER_CONTROL_`i Register . . . 22
7.3.2. `TIMER_COUNT_`i Register . . . 23
7.3.3. `TIMER_PERIOD_`i Register . . . 24
7.3.4. `TIMER_INT_STATUS` Register . . . 24
7.4. PWM . . . 25
7.4.1. `PWM_CONTROL_`i Register . . . 26
7.4.2. `PWM_VALUE_`i Register . . . 26
7.4.3. `PWM_NEXT_VALUE_`i Register . . . 27

8. Configuration **28**

A. `edusoc_basic`: Verilog-Compatible Interface for Arty S7 **30**

B. Revision History **32**

List of Figures

1. System Block Diagram 4
2. Default Memory Map 5
3. Example Read Waveform 7
4. Example Write Waveform 7
5. UART Bridge Protocol 11
6. VGA Default Screen Layout 12

List of Tables

1. Memory Bus Signals 6
2. Interrupt Bus Signals 7
3. SoC Interface Signals 8
4. Interrupt Mapping 10
5. UART Status Codes 11
6. VGA Pixel Color Format 12
7. SoC Control Registers 14
8. GPIO Registers 17
9. Timer Registers 22
10. PWM Registers 25
11. `edusoc_basic` Interface Signals 30

1. System Structure

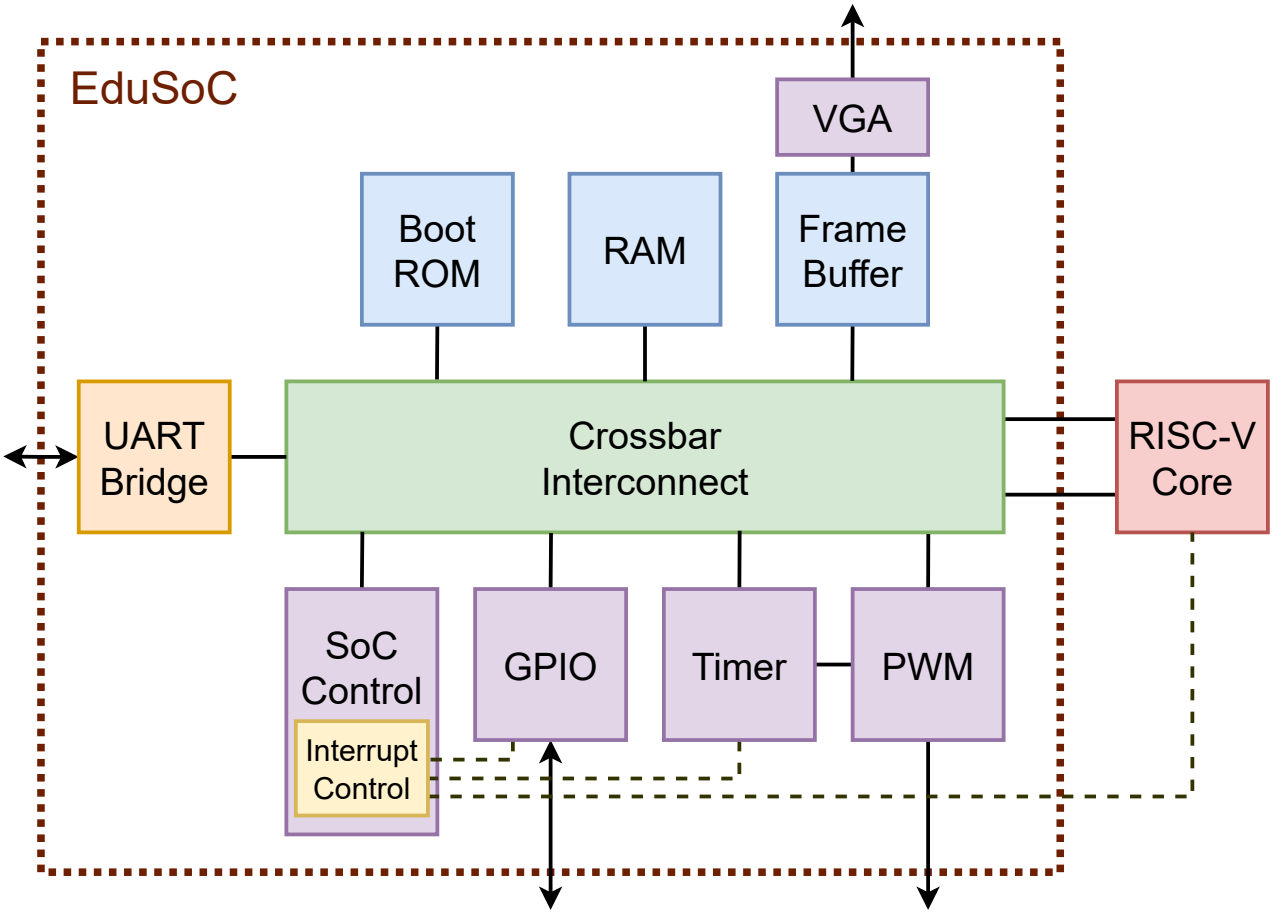

Figure 1: System Block Diagram

EduSoC provides all components within the dotted frame.

At its center, the crossbar interconnect serves as a bus connection between all system components. The interconnect arbitrates three bus masters (the UART bridge and the core's data and instruction buses), where the UART bridge has priority over the other two. The remaining SoC modules are bus slaves.

2. Memory Map

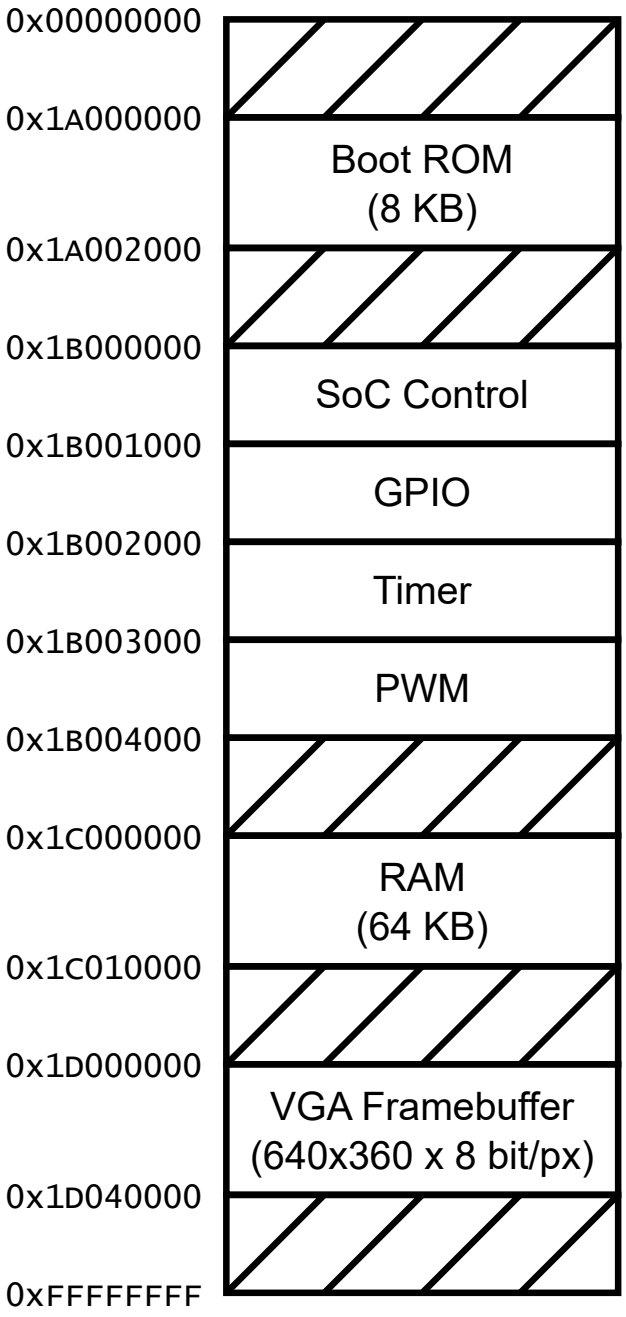

Figure 2: Default Memory Map

The above memory map represents the default configuration of EduSoC, and is fully configurable (see Section 8).

Hatched areas are unused. Writing to them has no effect, and reading from them produces undefined values.

3. Interfaces

3.1. Memory Bus

The EduSoC memory bus has the following signals:

Name	Width (bits)	Direction
`addr`	32	Master ⇒ Slave
`read_data`	32	Slave ⇒ Master
`write_data`	32	Master ⇒ Slave
`write_en`	1	Master ⇒ Slave
`byte_en`	4	Master ⇒ Slave
`req`	1	Master ⇒ Slave
`valid`	1	Slave ⇒ Master

Table 1: Memory Bus Signals

- `addr`: Requested byte address in the memory space. On the SoC level, only 4-byte-aligned requests are supported, so the lowest two address bits are ignored.
- `read_data`: Data read from the requested address. Only valid when `valid` = 1 (see below). Value is not defined for write requests.
- `write_data`: Data to be written to the requested address. Ignored for read requests.
- `write_en`: Whether the current request is a read request (0) or a write request (1).
- `byte_en`: Which of the four bytes of `write_data` should actually be written. Each bit corresponds to one of the four bytes (1 = write, 0 = keep unchanged), with the lowest bit corresponding to the lowest byte. Ignored for read requests.
- `req`: Set high (1) by the master to request bus access. Once set, all master signals should be held at a constant value until the `valid` signal is 1 (see below).
- `valid`: Set high (1) by the slave once the master's request has been completed. For read requests, this means valid read data is available on `read_data`, for write requests, this means the data from `write_data` has been written to the bus.

Once `valid` is asserted high (1), the request is considered complete and potential read data stays available until the request is terminated, either by resetting `req` to 0, or by changing `addr` or `write_en` (which constitutes a new request).

This also means that multiple consecutive requests may be made by keeping `req` = 1 and changing `addr` and/or `write_en` once the previous request is complete.

The following are example waveforms for read and write requests to the memory bus.

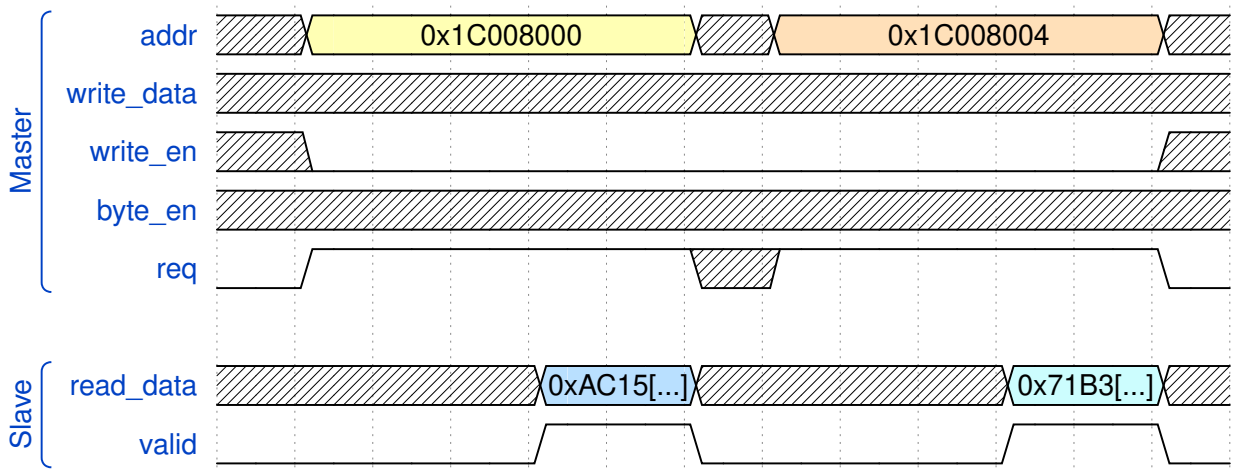

Figure 3: Example Read Waveform

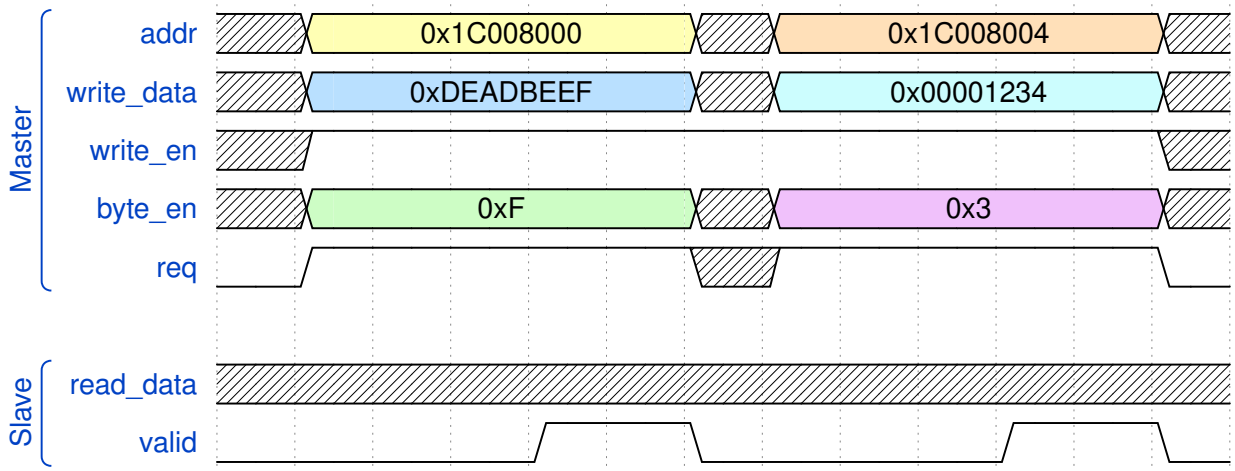

Figure 4: Example Write Waveform

3.2. Interrupt Bus

The EduSoC interrupt bus has the following signals:

Name	Width (bits)	Direction
`irq`	1	Generator ⇒ Handler
`irq_id`	5	Generator ⇒ Handler
`irq_ack`	1	Handler ⇒ Generator
`irq_ack_id`	5	Handler ⇒ Generator

Table 2: Interrupt Bus Signals

- `irq`: Set high (1) by the generator when an interrupt has occurred. Held until the interrupt is is acknowledged by the handler.

- `irq_id`: When `irq` = 1, conveys the ID of the interrupt to be handled (see Section 4).
- `irq_ack`: Set high (1) by the handler to signal that the currently asserted interrupt has been acknowledged and is being handled.
- `irq_ack_id`: When `irq_ack` = 1, conveys the ID of the interrupt that is being acknowledged.

3.3. SoC Interface

The SoC interface, consisting of the I/O interface and the CPU core interface, has the following signals and buses:

Name	Type or width (bits)	Direction
`ext_clk`	1	SoC input
`ext_resn`	1	SoC input
`vga_hsync`	1	SoC output
`vga_vsync`	1	SoC output
`vga_r`	4	SoC output
`vga_g`	4	SoC output
`vga_b`	4	SoC output
`uart_rx`	1	SoC input
`uart_tx`	1	SoC output
`gpio_in`	N	SoC input
`gpio_out`	N	SoC output
`gpio_drive`	N	SoC output
`pwm`	M	SoC output
`core_clk`	1	SoC output
`core_res`	1	SoC output
`control_flags`	16	SoC output
`instr_bus`	Memory Bus	SoC is slave
`data_bus`	Memory Bus	SoC is slave
`core_int_triggers`	16	SoC input
`int_bus`	Interrupt Bus	SoC is generator

Table 3: SoC Interface Signals

The following are central control signals for the system and must be connected. In a simulation environment, they may be left open, as they are driven automatically.

- `ext_clk`: Externally supplied clock signal. Must be driven by a 100 MHz clock.
- `ext_resn`: External reset (active low). Must be held high (1) and may optionally be asserted low (0) to reset the core and SoC.

Next is the VGA video output. For more information about these signals, see Section 6.

- `vga_hsync`: VGA horizontal sync signal.
- `vga_vsync`: VGA vertical sync signal.

- `vga_r`: VGA red color signal (16 levels).
- `vga_g`: VGA green color signal (16 levels).
- `vga_b`: VGA blue color signal (16 levels).

The following is the UART interface for external control and programming of the SoC. See Section 5 for more information.

- `uart_rx`: UART receive line (to SoC).
- `uart_tx`: UART transmit line (from SoC).

The following is the GPIO interface, with bit width $N = 32 \cdot \#\text{ports} \leq 512$. The three signals are designed to be combined into a single N-bit bidirectional I/O port. For more information, see Section 7.2.

- `gpio_in`: GPIO input signals.
- `gpio_out`: GPIO output signals.
- `gpio_drive`: GPIO driver controls. If a bit is 0, that pin of the port is an input ("high impedance"), reading from the corresponding `gpio_in` bit. If a bit is 1, that pin of the port is an output, writing to the corresponding `gpio_out` bit.

The following is the output for the PWM modules, containing M pulse width modulated signals, where $M \leq 16$ is the number of PWM modules in the SoC. See Section 7.4 for more information.

- `pwm`: PWM output signals.

Finally, the following is the CPU core interface. See Sections 3.1 and 3.2 for the memory and interrupt bus descriptions, respectively.

- `core_clk`: CPU core clock.
- `core_res`: CPU core reset, active high. Intended as a synchronous reset, so the core should be reset on the rising edge of `core_clk` if this signal is high (1).
- `control_flags`: User definable control signals for the core. See Section 7.1 for more information.
- `instr_bus`: CPU instruction memory bus. Intended to be used by the core to load program instructions from memory.
- `data_bus`: CPU data memory bus. Intended to be used by the core to read and write memory or peripheral register values.
- `core_int_triggers`: Core interrupt triggers. If asserted high, a corresponding interrupt will be triggered. See Section 4 for more information.
- `int_bus`: CPU interrupt bus.

4. Interrupts

EduSoC contains a simple interrupt controller to notify the core of asynchronous events. It supports up to 32 distinct interrupts, designated by interrupt IDs (0-31).

The priority order of the interrupts is fixed, where interrupts with a lower ID are forwarded to the core/handler first.

16 of these interrupts are SoC interrupts, generated by SoC modules and peripherals. The remaining 16 interrupts are core interrupts, which may be triggered by any events internal to the core. To trigger such an interrupt, the corresponding bit in the `core_int_triggers` signal (see Section 3.3) must be set high (1) for at least one clock cycle. The interrupt will stay asserted until it is acknowledged by the core (i. e. typically handled in software).

The interrupt indices are assigned as follows:

ID	Source	ID	Source	ID	Source	ID	Source
0	Core Interrupt 0	8	Reserved	16	Reserved	24	Core Interrupt 8
1	Core Interrupt 1	9	Reserved	17	Reserved	25	Core Interrupt 9
2	Core Interrupt 2	10	Reserved	18	Reserved	26	Core Interrupt 10
3	Core Interrupt 3	11	Timer	19	Reserved	27	Core Interrupt 11
4	Core Interrupt 4	12	Reserved	20	Reserved	28	Core Interrupt 12
5	Core Interrupt 5	13	Reserved	21	Reserved	29	Core Interrupt 13
6	Core Interrupt 6	14	Reserved	22	Reserved	30	Core Interrupt 14
7	Core Interrupt 7	15	GPIO	23	Reserved	31	Core Interrupt 15

Table 4: Interrupt Mapping

The core should always acknowledge any asserted interrupt as soon as possible, even if that interrupt is not supported or not handled by the core. Otherwise, other interrupts may be missed, especially lower-priority ones.

Interrupts are only asserted to the core if they are enabled first, and they may be cleared by a special register write. See Section 7.1 for more information on interrupt control.

5. UART Bridge

The UART bridge allows EduSoC to be programmed and controlled externally through a UART interface. The bridge functions as a memory bus master which is controlled by a UART protocol.

The UART data rate is 500000 Baud by default, but may be configured, see Section 8.

The UART bridge only supports writing to the memory bus in units of 32 bit words. Each 32 bit (4 byte) word is transmitted in little endian byte order. The UART protocol is as follows:

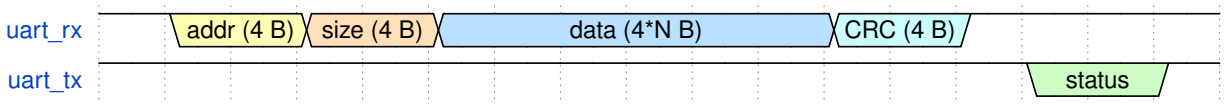

Figure 5: UART Bridge Protocol

First, the starting memory address is sent (i. e. the address of the first word to be written). As described in Section 3.1, only 4-byte-aligned addresses are supported. This is followed by 4 bytes containing the size N of the data to be written, **in units of 32 bit (4 byte) words** (*not* bytes).

The next N 32 bit words, i. e. $4 \cdot N$ bytes, are the actual data to be written, starting at the given address, with further words being written to the consecutively next memory addresses. Finally, the last 32 bit word is a CRC sum of the preceding data words (not including the address and size). The CRC algorithm used is CRC-32C (polynomial `0x1EDC6F41`, input and output reversed, initial value = final XOR = `0xFFFFFFFF`).

Once the bridge has received all of this data, it responds with a single byte indicating the status, which may be one of the following:

Value	Meaning
0x59	Success
0x23	CRC Mismatch
0xE0	Error
Other	Unknown (error)

Table 5: UART Status Codes

Once a success or CRC mismatch status has been sent by the bridge, it is ready for a new transmission. In the last two cases (error), the state of the UART bridge and the destination memory area is undefined, and the bridge may not respond to further UART transmissions. An external system reset is required to return to a known valid state.

In case of a CRC mismatch, data corruption during the transmission is possible. Note that this possibly corrupted data has been written to the destination memory area at this point, so it is recommended to retry the transmission to overwrite the corrupted data.

The EduSoC repository includes a Python script (`uart-programming/uart_upload.py`) implementing this UART interface for the Digilent Arty S7 board. See the header comment in the script itself for more information.

6. VGA Controller

EduSoC features a VGA video interface, requiring only an external DAC module for the analog color signals, for example the Digilent PmodVGA. By default, it is configured for a 640x480 video output at 60 Hz refresh rate, where only the central 640x360 area is backed by actual pixel data. This can be configured, see Section 8.

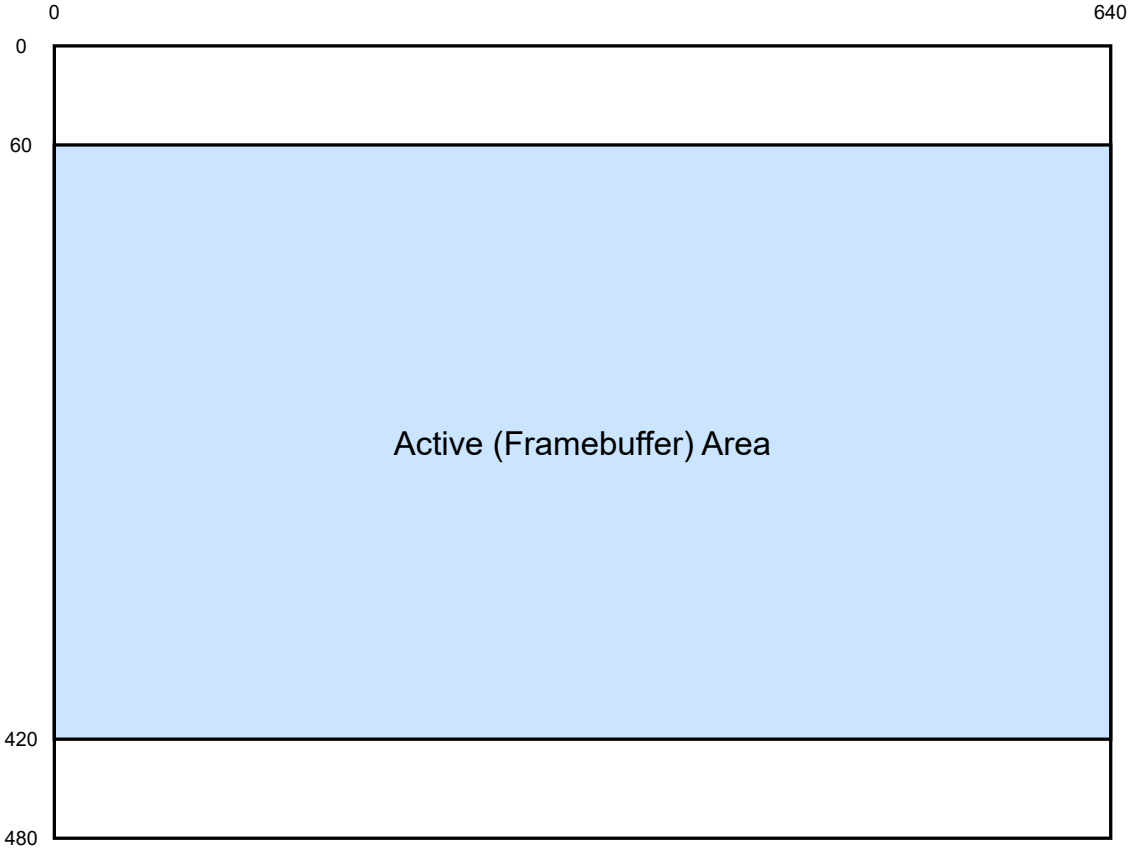

Figure 6: VGA Default Screen Layout

The VGA controller is connected to the framebuffer memory, reading one byte (8 bits) of framebuffer data for each pixel in typical scanline order and generating all VGA interface signals from this data and its given pixel clock. The pixel data format is RGB-332, so for each byte (= pixel), the upper three bits are the red intensity, the next three bits are the green intensity, and the lowest two bits are the blue intensity:

Bit	7	6	5	4	3	2	1	0
Meaning	R[2:0]			G[2:0]			B[1:0]	

Table 6: VGA Pixel Color Format

The framebuffer can be read and written to using the memory bus, like any other memory section, and changes to it will be visible on a connected screen immediately as soon as the changed pixels are drawn in the next frame. Note that, like any other EduSoC memory, the framebuffer contents are not affected by a system reset.

7. Peripherals

EduSoC contains multiple memory-mapped peripheral modules. They are accessible using the main memory bus and have registers to control their function, which are described in the following sections.

All registers in the following peripherals have four separate addresses each (differentiated by the last hexadecimal digit of the address), implementing functions for easy bit manipulations:

- `0x???????0`: Main register. Allows reading the register value and writing a value directly.
- `0x???????4`: SET register. Reads as 0. When written: For every high (1) bit in the written value, the corresponding register bit is set high (1). Low (0) bits in the written value leave the corresponding register bits unchanged.
- `0x???????8`: CLEAR register. Reads as 0. When written: For every high (1) bit in the written value, the corresponding register bit is cleared (set to 0). Low (0) bits in the written value leave the corresponding register bits unchanged.
- `0x???????C`: INVERT register. Reads as 0. When written: For every high (1) bit in the written value, the corresponding register bit is inverted ($0 \leftrightarrow 1$). Low (0) bits in the written value leave the corresponding register bits unchanged.

In the following register descriptions, only the main register address is given, but the SET, CLEAR, and INVERT addresses still function as described above, by using the corresponding last address digit.

The only exceptions are register bits which are read-only or may only be written in certain ways (e. g. clear-only bits) - the SET, CLEAR, and INVERT variants of those registers respect these restrictions, and will not perform any unsupported operations.

The register addresses given in the following register descriptions assume the default memory map described in Section 2.

7.1. SoC Control

The SoC control peripheral implements functions for controlling the SoC and system as a whole, as well as controlling system interrupts.

It allows software- or UART-initiated resets of the core and SoC, allows the core to be halted, and allows detailed control over interrupts (enabling/disabling interrupts globally, enabling/disabling individual interrupts, reading interrupt status, clearing individual interrupts).

Additionally, a 16 bit core control flag signal is provided (see `control_flags` in Section 3.3). These flags may have a user-defined meaning and can also be read or written using the memory bus. Unlike most registers, these flags are *not* affected by system resets, retaining their state.
For the bootloader code and UART programming utilities provided with EduSoC, only one of these flags (flag 0) has a meaning by default - it serves as an indicator that a program has been loaded into RAM. See Appendix A for more information.

This peripheral has the following registers:

Address	Name	Reset Value
0x1B000000	SOCCON_CONTROL	0x00000008[1]
0x1B000010	SOCCON_INT_EN	0x00000000
0x1B000020	SOCCON_INT_FLAGS	0x00000000

Table 7: SoC Control Registers

7.1.1. `SOCCON_CONTROL` Register

Address: 0x1B000000 Reset value: 0x00000008[1]

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	control_flags[15:8]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	control_flags[7:0]							
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0	R/W-0
	—	—	—	—	INTGEN	SOCRES	CORERES	COREHLT

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-16 `control_flags`: Core control flags, see description above. Reset retains values.

Bit 15-4 Unimplemented: Read as 0.

Bit 3 `INTGEN`: Interrupt global enable.
0 = No interrupts will be asserted to the core, irrespective of individual interrupt enable settings.

[1]Upper 16 bits are initially 0x0000, but retain their values during further resets.

1 = Individually enabled interrupts will be asserted to the core.

Bit 2 SOCRES: SoC software reset.

0 = No effect.

1 = SoC will be reset on the next clock cycle (including this register/bit).

Bit 1 CORERES: CPU core reset.

0 = No effect.

1 = CPU core reset line is asserted high (1), leading to a reset on the next CPU core clock cycle.

Bit 0 COREHLT: CPU core halt.

0 = CPU core clock is running.

1 = CPU core clock is halted. Note that this also prevents synchronous core resets from occurring.

7.1.2. SOCCON_INT_EN Register

Address: 0x1B000010 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	INT_ENABLE[31:24]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	INT_ENABLE[23:16]							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	INT_ENABLE[15:8]							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	INT_ENABLE[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 INT_ENABLE: Individual interrupt enable.

0 = Interrupt with the corresponding index will not be asserted to the core.

1 = Interrupt with the corresponding index will be asserted to the core whenever it occurs.

Each bit corresponds to one interrupt ID, e. g. bit 0 corresponds to interrupt ID 0.

When enabling an interrupt, if it has occurred in the past (i. e. its flag is 1, see SOCCON_INT_FLAGS), it will immediately be asserted to the core (assuming INTGEN = 1 in SOCCON_CONTROL).

7.1.3. `SOCCON_INT_FLAGS` Register

Address: `0x1B000020` Reset value: `0x00000000`

Bit Range	Bits (highest to lowest)							
31:24	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	INT_FLAGS[31:24]							
23:16	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	INT_FLAGS[23:16]							
15:8	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	INT_FLAGS[15:8]							
7:0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	INT_FLAGS[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
C = Clearable	-n = Initial Value	1 = Set 0 = Cleared	x = Unknown

Bit 31-0 `INT_FLAGS`: Interrupt occurred flags.

0 = Interrupt with the corresponding index has not occurred.

1 = Interrupt with the corresponding index has occurred and will be asserted to the core if it is enabled.

Each bit corresponds to one interrupt ID, e. g. bit 0 corresponds to interrupt ID 0.

Flags/bits in this register may be cleared (set to 0) using memory writes, which will clear the corresponding interrupts and stop them from being asserted until they reoccur. Attempting to set bits to 1 using memory writes will have no effect.

7.2. GPIO

The GPIO peripheral implements a general-purpose input/output interface for the system, allowing input and output control of up to 512 individual pins, as well as individual-pin input change notification interrupts for responding to external events.

Externally, the GPIO peripheral provides a virtual bidirectional bus with separate input, output, and pin drive/direction signals. These are intended to be combined into a true bidirectional bus on the system level.

Internally, the GPIO pins are divided into 32 pin (32 bit) ports, with a set of registers to control each port. The total number of ports is configurable between 1 port (32 pins total) and 16 ports (512 pins total) (see Section 8), where the default port count is 1.

In the following register descriptions, each type of register is only described once, even if it has multiple copies for the different ports. i is used as a placeholder for the port index in these cases (where the first port is $i = 0$). In register addresses, i should be replaced with the hexadecimal representation of the desired port index.

This peripheral has the following registers:

Address	Name	Reset Value
0x1B001i00	GPIO_PORT_i	0x????????[1]
0x1B001i10	GPIO_LATCH_i	0x00000000
0x1B001i20	GPIO_DIR_i	0x00000000
0x1B001i30	GPIO_CNR_i	0x00000000
0x1B001i40	GPIO_CNF_i	0x00000000
0x1B001i50	GPIO_CN_STATE_i	0x00000000
0x1B0010F0	GPIO_INT_STATUS	0x00000000

Table 8: GPIO Registers

[1] Since the read value of this register depends on external pin states, there is no defined reset value.

7.2.1. `GPIO_PORT_`i Register

Address: `0x1B001`i`00` Reset value: `0x????????`[1]

Bit Range	Bits (highest to lowest)							
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	`PORT_`i[31:24]							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	`PORT_`i[23:16]							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	`PORT_`i[15:8]							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	`PORT_`i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 `PORT_`i: GPIO port i pin states.
0 = Corresponding pin is in a low (0) state.
1 = Corresponding pin is in a high (1) state.

If a pin is configured as an output, the corresponding pin in this register represents its currently driven value.
Otherwise, the corresponding pin in this register represents its current input signal value.

Writing to this register is equivalent to writing to the `GPIO_LATCH_`i register (see below).

7.2.2. `GPIO_LATCH_`i Register

Address: `0x1B001`i`10` Reset value: `0x00000000`

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	`LATCH_`i[31:24]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	`LATCH_`i[23:16]							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	`LATCH_`i[15:8]							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	`LATCH_`i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 `LATCH_`i: GPIO port i output latch states.
0 = Corresponding pin latch is in a low (0) state.
1 = Corresponding pin latch is in a high (1) state.

The latch state of a pin, defined in this register, determines the logic level that will be driven on the pin if it is configured as an output pin. For input-configured pins, the latch value is ignored.

7.2.3. GPIO_DIR_i Register

Address: 0x1B001i20 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIR_i[31:24]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIR_i[23:16]							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIR_i[15:8]							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DIR_i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 DIR_i: GPIO port i pin directions.
0 = Corresponding pin is configured as an input pin.
1 = Corresponding pin is configured as an output pin.

7.2.4. GPIO_CNR_i Register

Address: 0x1B001i30 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNR_i[31:24]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNR_i[23:16]							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNR_i[15:8]							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNR_i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 CNR_i: GPIO port i rising edge change notification enable.
0 = Corresponding pin will not generate change notification interrupts for rising edges.
1 = Corresponding pin will generate change notification interrupts for rising edges ($0 \rightarrow 1$).

7.2.5. GPIO_CNF_i Register

Address: 0x1B001i40 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNF_i[31:24]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNF_i[23:16]							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNF_i[15:8]							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CNF_i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 CNF_i: GPIO port i falling edge change notification enable.
0 = Corresponding pin will not generate change notification interrupts for falling edges.
1 = Corresponding pin will generate change notification interrupts for falling edges (1 $\rightarrow$ 0).

7.2.6. GPIO_CN_STATE_i Register

Address: 0x1B001i50 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	CN_STATE_i[31:24]							
23:16	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	CN_STATE_i[23:16]							
15:8	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	CN_STATE_i[15:8]							
7:0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	CN_STATE_i[7:0]							

Legend:	R = Readable	W = Writable		U = Unimplemented, read as 0
C = Clearable	-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 CN_STATE_i: GPIO port i edge change notification state.
0 = Corresponding pin has not generated a change notification interrupt.
1 = Corresponding pin has generated a change notification interrupt (i. e. it has experienced a signal edge with enabled change notification).

Bits in this register may be cleared (set to 0) using memory writes. Attempting to set bits to 1 using memory writes will have no effect.

7.2.7. `GPIO_INT_STATUS` Register

Address: `0x1B0010F0` Reset value: `0x00000000`

Bit Range	Bits (highest to lowest)							
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	`INT_STATUS[15:8]`							
7:0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	`INT_STATUS[7:0]`							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
C = Clearable	-n = Initial Value	1 = Set 0 = Cleared	x = Unknown

Bit 31-16 Unimplemented: Read as 0.

Bit 15-0 `INT_STATUS`: GPIO ports interrupt status.

0 = Port with corresponding index has not generated a change notification interrupt.

1 = Port with corresponding index has generated a change notification interrupt (i. e. one or more of its pins have experienced a signal edge with enabled change notification).

The purpose of this register is to track which ports have generated change notification interrupts, to allow the software to efficiently find which pin is the latest interrupt source.

Bits in this register may be cleared (set to 0) using memory writes. Attempting to set bits to 1 using memory writes will have no effect.

7.3. Timer

The timer peripheral implements configurable timed event functionality. It consists of up to 16 independent timer modules (configurable, see Section 8), where the default timer count is 2.

Each timer module has a period register determining the tick period in clock cycles (by default, one clock cycle = 40ns, see also Section 8), as well as a counter register that contains the current timer cycle count (within its current period). It can be configured to run in continuous or one-shot mode, and to optionally generate an interrupt on every tick / counter rollover.

In the following register descriptions, each type of register is only described once, even if it has multiple copies for the different timer modules. i is used as a placeholder for the timer index in these cases (where the first timer is $i = 0$). In register addresses, i should be replaced with the hexadecimal representation of the desired timer index.

This peripheral has the following registers:

Address	Name	Reset Value
0x1B002i00	TIMER_CONTROL_i	0x00000000
0x1B002i10	TIMER_COUNT_i	0x00000000
0x1B002i20	TIMER_PERIOD_i	0x00000000
0x1B0020F0	TIMER_INT_STATUS	0x00000000

Table 9: Timer Registers

7.3.1. TIMER_CONTROL_i Register

Address: 0x1B002i00 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
	—	—	—	—	—	—	—	TMRRES
7:0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	—	INT_EN	ONESHOT	ENABLE

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-9 Unimplemented: Read as 0.

Bit 8 TMRRES: Timer counter reset.

0 = No effect.

1 = Timer i will have its counter (COUNT_i) reset to 0 on the next clock cycle, and this bit will be cleared (0). This does not count as a tick / counter rollover.

Bit 7-3 Unimplemented: Read as 0.

Bit 2 `INT_EN`: Timer interrupt enable.

0 = Timer i will not generate any interrupts.

1 = Timer i will generate an interrupt on each tick / counter rollover.

Bit 1 `ONESHOT`: Timer oneshot mode.

0 = Timer i will run continuously while `ENABLE` = 1 until it is disabled manually.

1 = Timer i will run until the next tick / counter rollover if `ENABLE` = 1, at which point `ENABLE` will be cleared (0) automatically.

Bit 0 `ENABLE`: Timer enable.

0 = Timer i is stopped, its counter will not increment.

1 = Timer i is running, its counter will increment and roll over when it reaches the configured period.

7.3.2. `TIMER_COUNT_`i Register

Address: `0x1B002`i`10` Reset value: `0x00000000`

Bit Range	Bits (highest to lowest)							
31:24	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`COUNT_`i[31:24]							
23:16	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`COUNT_`i[23:16]							
15:8	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`COUNT_`i[15:8]							
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`COUNT_`i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 `COUNT_`i: Timer i counter value.

Contains the number of clock cycles that have passed in the current period of timer i. The value in this register should always be less than the timer's period (note the special case for `PERIOD` = 0, see below).

This register is read-only, attempting to write to it will have no effect.

7.3.3. `TIMER_PERIOD_`i Register

Address: 0x1B002i20 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PERIOD_i[31:24]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PERIOD_i[23:16]							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PERIOD_i[15:8]							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PERIOD_i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 PERIOD_i: Timer i period.
0x00000000 = 4,294,967,296 clock cycles
0x00000001 = 1 clock cycle
0x00000002 = 2 clock cycles
...
0xFFFFFFFF = 4,294,967,295 clock cycles

Writing to this register resets the timer's counter (COUNT_i) to 0 without a tick / counter rollover.

7.3.4. `TIMER_INT_STATUS` Register

Address: 0x1B0020F0 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	INT_STATUS[15:8]							
7:0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
	INT_STATUS[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
C = Clearable	-n = Initial Value	1 = Set 0 = Cleared	x = Unknown

Bit 15-0 INT_STATUS: Timer interrupt status.
0 = Timer with corresponding index has not generated a tick / counter rollover interrupt.
1 = Timer with corresponding index has generated a tick / counter rollover interrupt.

Bits in this register may be cleared (set to 0) using memory writes. Attempting to set bits to 1 using memory writes will have no effect.

7.4. PWM

The PWM peripheral implements configurable pulse width modulated signal generation. It consists of up to 16 independent PWM modules (configurable, see Section 8), where the default PWM module count is 6.

Each PWM module depends on one timer from the timer peripheral (see Section 7.3). The PWM signal period is equal to the timer's period, and the timer must be enabled and running continuously for correct PWM signal generation. Multiple PWM modules can use the same timer.

The modulation value (pulse width) of a PWM module is controlled by its value register, which determines the absolute width of each pulse (in clock cycles). For a value of 0, the signal is constantly low (0), for a value greater than or equal to the corresponding timer period, the signal is constantly high (1).

The value register cannot be modified directly, instead, each module has a "next value" register which can be written to, and will be copied to the value register at the start of the following PWM/timer period.

In the following register descriptions, each type of register is only described once, even if it has multiple copies for the different PWM modules. i is used as a placeholder for the module index in these cases (where the first module is $i = 0$). In register addresses, i should be replaced with the hexadecimal representation of the desired module index.

This peripheral has the following registers:

Address	Name	Reset Value
0x1B003i00	PWM_CONTROL_i	0x00000000
0x1B003i10	PWM_VALUE_i	0x00000000
0x1B003i20	PWM_NEXT_VALUE_i	0x00000000

Table 10: PWM Registers

7.4.1. `PWM_CONTROL_`i Register

Address: `0x1B003`i`00` Reset value: `0x00000000`

Bit Range	Bits (highest to lowest)							
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
	—	—	—	—	—	—	—	`ENABLE`
7:0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	`TIMER_ID`			

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-9 Unimplemented: Read as 0.

Bit 8 `ENABLE`: PWM module enable.
0 = PWM module i is disabled, its output is constantly low (0).
1 = PWM module i is enabled and outputting a PWM signal.

Bit 7-4 Unimplemented: Read as 0.

Bit 3-0 `TIMER_ID`: Index of the timer module used for PWM module i. See Section 7.3.

7.4.2. `PWM_VALUE_`i Register

Address: `0x1B003`i`10` Reset value: `0x00000000`

Bit Range	Bits (highest to lowest)							
31:24	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`VALUE_`i[31:24]							
23:16	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`VALUE_`i[23:16]							
15:8	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`VALUE_`i[15:8]							
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	`VALUE_`i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 `VALUE_`i: PWM module i pulse width in clock cycles.

For a value of 0, the PWM signal is constantly low (0), for a value greater than or equal to the corresponding timer period, the PWM signal is constantly high (1).

This register is read-only, attempting to write to it will have no effect. For setting the value, the `PWM_NEXT_VALUE_`i register should be used instead (see below).

7.4.3. PWM_NEXT_VALUE_i Register

Address: 0x1B003i20 Reset value: 0x00000000

Bit Range	Bits (highest to lowest)							
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NEXT_VALUE_i[31:24]							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NEXT_VALUE_i[23:16]							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NEXT_VALUE_i[15:8]							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NEXT_VALUE_i[7:0]							

Legend:	R = Readable	W = Writable	U = Unimplemented, read as 0
-n = Initial Value	1 = Set	0 = Cleared	x = Unknown

Bit 31-0 NEXT_VALUE_i: PWM module i next pulse width in clock cycles.

The value in this register will be copied to PWM_VALUE_i at the start of each PWM/timer period.

8. Configuration

Many of EduSoC's implementation details described above are configurable for individual use cases of EduSoC. The main configuration file for this purpose is `soc/soc_defines.sv`.

This section serves as an overview of the available configuration options and their meaning.

Memory and Bus Configuration Here, the SoC memories, bus behaviour, and memory map are configured.

- `MEM_BOOTROM_ADDR_WIDTH`: Address width, in bits, of the Boot ROM. This also determines the Boot ROM size - for address width m, the Boot ROM size will be 2^m bytes.
- `MEM_RAM_ADDR_WIDTH`: Address width, in bits, of the RAM. This also determines the RAM size - for address width m, the RAM size will be 2^m bytes.
- `MEM_FRAMEBUFFER_ADDR_WIDTH`: Address width, in bits, of the VGA framebuffer. This *does not* determine the framebuffer size - that is defined in the VGA configuration section (see below). This width needs to be large enough to make the entire framebuffer addressable.
- `MEM_*_INIT`: Memory initialization files. Separately configurable for simulation and synthesis.
- `MEM_*_LATENCY`: Latency for the validation of memory bus requests to the corresponding bus slave category, in clock cycles. Actual latency will be one cycle longer than this value.
- `ADDR_SLAVE_COUNT`: Number of memory bus slaves connected to the interconnect. Note that adding or removing slaves requires further changes in the SoC code.
- `ADDR_*_START`: Start address of the given memory bus slave's memory space, inclusive.
- `ADDR_*_END`: End address of the given memory bus slave's memory space, inclusive.

Clock Configuration Here, the frequencies for the different SoC clocks are configured.

- `CLK_MAIN_DIVIDER`: Divider for the generation of the main system (SoC and core) clock. For a divider value n, the resulting frequency is $\frac{800\text{ MHz}}{n}$. Value must be between 1 and 128, inclusive. Default: 32 (25 MHz clock).
- `CLK_VGA_DIVIDER`: Divider for the generation of the VGA pixel clock (must correspond to the implemented VGA resolution). For a divider value n, the resulting frequency is $\frac{800\text{ MHz}}{n}$. Value must be between 1 and 128, inclusive. Default: 32 (25 MHz clock).
- `CLK_UART_DIVIDER_PLL`: Main divider for the generation of the UART clock (see below). Value must be between 1 and 128, inclusive. Default: 100 (8 MHz clock).
- `CLK_UART_DIVIDER_POST`: Post-divider for the generation of the UART clock (see below). Value must be either 1 or a positive multiple of 2. Default: 1 (8 MHz clock).

For the UART clock, given a PLL divider n and a post-divider m, the resulting frequency is $\frac{800\text{ MHz}}{n \cdot m}$. The corresponding UART data rate is $\frac{1}{16}$ of the UART clock frequency, i. e. $\frac{50{,}000{,}000}{n \cdot m}$ Baud.

Video Configuration Here, various Video/VGA parameters are configured.

- `VIDEO_FB_MEM_SIZE`: Framebuffer size, in *bits*. Should be sized accordingly to the chosen active screen area size/resolution. Remember to ensure a sufficient framebuffer address width (see above).
- `VIDEO_VGA_OFFSET`: Vertical offset (in lines/pixels) of the active display area from the top of the overall VGA frame.
- `VIDEO_VGA_*_LINE`: VGA line timings for the start and end of the active display area. See examples in the configuration file.

Changing the overall VGA frame resolution is also possible, but it requires the redefinition of several values in the VGA controller code aside from the configuration values given here, and a different VGA pixel clock (see above).

Peripheral Configuration Here, the SoC peripherals are configured.

- `GPIO_PORT_COUNT`: Number of ports provided by the GPIO peripheral (32 pins per port). Must be between 1 and 16, inclusive. Default: 1 (32 pins).
- `TIMER_COUNT`: Number of timers provided by the timer peripheral. Must be between 1 and 16, inclusive. Default: 2.
- `PWM_MODULE_COUNT`: Number of PWM modules provided by the PWM peripheral. Must be between 1 and 16, inclusive. Default: 6.

A. `edusoc_basic`: Verilog-Compatible Interface for Arty S7

For simple applications on the Arty S7 board with the default SoC configuration, the `edusoc_basic` module is provided. It is fully compatible with Verilog interfaces (as opposed to the SystemVerilog requirement for the generic `edusoc` module) and provides simple connections designed for the Arty S7 board:

Name	Width (bits)	Direction	Name	Width (bits)	Direction
BOARD_CLK	1	SoC input	INSTR_REQ	1	SoC input
BOARD_RESN	1	SoC input	INSTR_VALID	1	SoC output
BOARD_LED	4	SoC output	INSTR_ADDR	32	SoC input
BOARD_LED_RGB0	3	SoC output	INSTR_RDATA	32	SoC output
BOARD_LED_RGB1	3	SoC output	DATA_REQ	1	SoC input
BOARD_BUTTON	4	SoC input	DATA_VALID	1	SoC output
BOARD_SWITCH	4	SoC input	DATA_WE	1	SoC input
BOARD_VGA_HSYNC	1	SoC output	DATA_BE	4	SoC input
BOARD_VGA_VSYNC	1	SoC output	DATA_ADDR	32	SoC input
BOARD_VGA_R	4	SoC output	DATA_WDATA	32	SoC input
BOARD_VGA_G	4	SoC output	DATA_RDATA	32	SoC output
BOARD_VGA_B	4	SoC output	IRQ	1	SoC output
BOARD_UART_RX	1	SoC input	IRQ_ID	5	SoC output
BOARD_UART_TX	1	SoC output	IRQ_ACK	1	SoC input
CPU_CLK	1	SoC output	IRQ_ACK_ID	5	SoC input
CPU_RES	1	SoC output			

Table 11: `edusoc_basic` Interface Signals

The majority of these signals are direct equivalents to generic EduSoC interface connections (see Section 3.3), or fanouts of the buses contained therein:

- `BOARD_CLK` and `BOARD_RESN` correspond to `ext_clk` and `ext_resn`, respectively.
- `BOARD_VGA_*` and `BOARD_UART_*` correspond to `vga_*` and `uart_*`, respectively.
- `CPU_CLK` and `CPU_RES` correspond to `core_clk` and `core_res`, respectively.
- `INSTR_*` correspond to the individual signals of `instr_bus`. As the instruction bus is intended to be a read-only bus, only the corresponding signals are available. Write-related signals are tied to constant 0.
- `DATA_*` correspond to the individual signals of `data_bus`.
- `IRQ*` correspond to the individual signals of `int_bus`.
- Other signals from the SoC interface are not available in this basic interface (`control_flags`, `core_int_triggers`, `gpio_*` and `pwm` aside from exceptions below). If they are needed, please use the generic `edusoc` module instead.

The remaining signals of `edusoc_basic` are intended to be mapped to the corresponding devices available on the Arty S7 board (LEDs, RGB LEDs, buttons, switches). They are internally connected to the GPIO and/or PWM ports:

- `BOARD_LED`[3:0] = GPIO pins 3 to 0 (output only)
- `BOARD_LED_RGB0`[2:0] = GPIO pins 10 to 8 (output only) $\vee$ PWM signals 2 to 0 (bitwise logical OR)
- `BOARD_LED_RGB1`[2:0] = GPIO pins 14 to 12 (output only) $\vee$ PWM signals 5 to 3 (bitwise logical OR)
- `BOARD_BUTTON`[3:0] = GPIO pins 19 to 16 (input only)
- `BOARD_SWITCH`[3:0] = GPIO pins 23 to 20 (input only)

A simple RISC-V bootloader is provided as well (`bootloader.mem`).

In a system with a functional RISC-V core, it configures the GPIO registers according to the above assignments, enabling change notifications for the buttons and switches (though the GPIO interrupt remains disabled), and either jumps to RAM at address `0x1C000080` if control flag 0 is set (see Section 7.1), or displays an idle counting animation on the board LEDs otherwise.

This allows a UART programming script to write a new program to RAM, set control flag 0 and reset the system to let the bootloader launch the newly loaded program.

B. Revision History

- v1.0: Initial Version.