# example_module

**Address width:** 32

**Data width:** 32

**Base address:** 0xFFAA0000

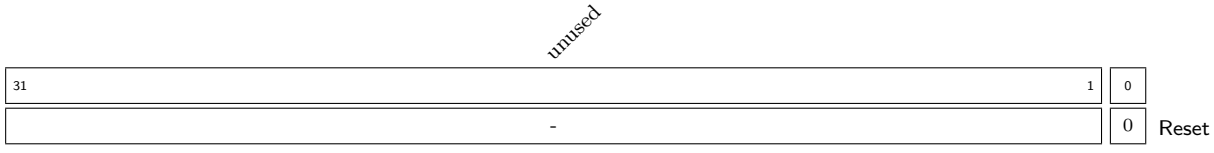
An example module that contain all the register types that are currently supported by uart.

## 1 Register List

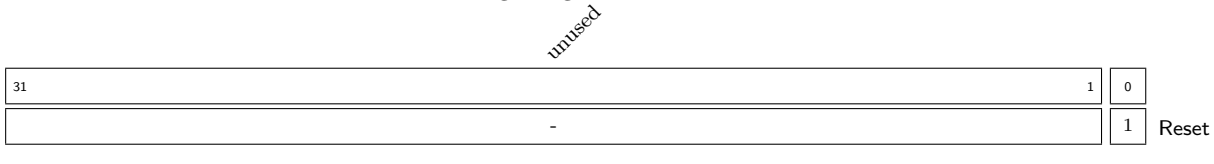
| # | Name | Mode | Address | Type | Length | Reset |
|---|---|---|---|---|---|---|
| 0 | reg0 | RW | 0x00000000 | SL | 1 | 0x0 |
| 1 | reg1 | RW | 0x00000004 | SL | 1 | 0x1 |
| 2 | reg2 | RO | 0x00000008 | SL | 1 | 0x0 |
| 3 | reg3 | RW | 0x0000000C | SLV | 8 | 0x3 |
| 4 | reg4 | RO | 0x00000010 | SLV | 14 | 0x0 |
| 5 | reg5 | RW | 0x00000014 | DEFAULT | 32 | 0xFFFFFFFF |
| 6 | reg6 | RO | 0x00000018 | DEFAULT | 32 | 0x0 |
| 7 | reg7 | RW | 0x0000001C | FIELDS | 21 | 0xAD7 |
| 8 | reg8 | RO | 0x00000020 | FIELDS | 24 | 0x0 |

## 2 Registers

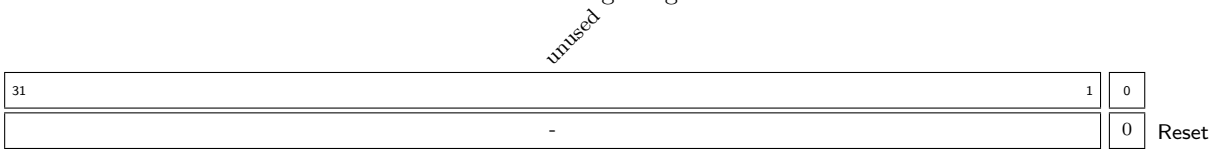
Register 2.1: REG0 - RW (0x00000000)
RW std_logic register that resets to 0x0



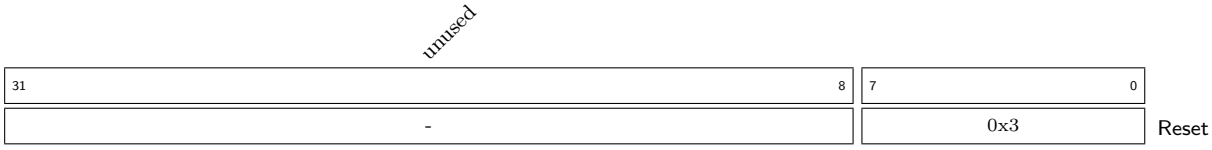

Register 2.2: REG1 - RW (0x00000004)
RW std_logic register that resets to 0x1



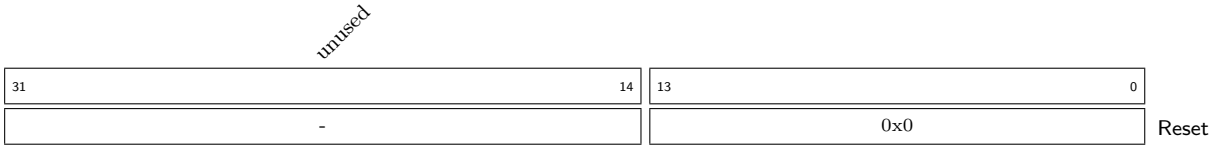
Register 2.3: REG2 - RO (0x00000008)
RO std_logic register



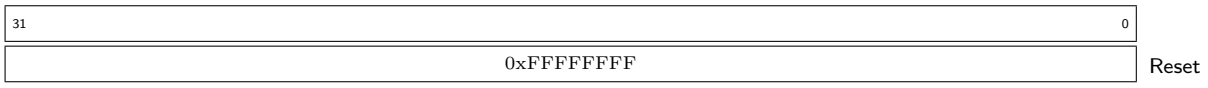

Register 2.4: REG3 - RW (0x0000000C)
RW std_logic_vector[7:0] register that resets to 0x3



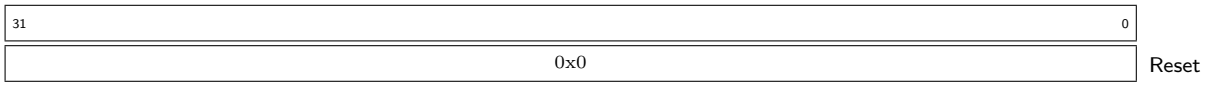

Register 2.5: REG4 - RO (0x00000010)
RO std_logic_vector[13:0]



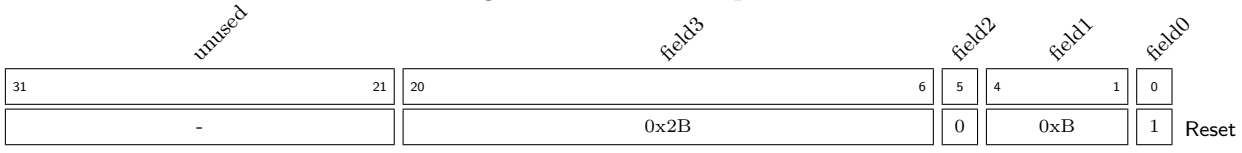

Register 2.6: REG5 - RW (0x00000014)
Default RW register that resets to 0xFFFFFFFF




Register 2.7: REG6 - RO (0x00000018)
Default RO register




Register 2.8: REG7 - RW (0x0000001C)
RW register that have multiple fields



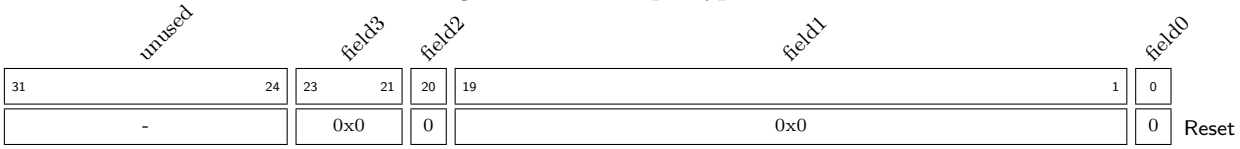

**field0** std_logic that resets to 0x1

**field1** std_logic_vector[3:0] that resets to 0xb is not a valid reset value

**field2** std_logic that resets to 0x1

**field3** std_logic_vector[14:0] that resets to 0x2b

Register 2.9: REG8 - RO (0x00000020)
RO register with multiple types of fields




**field0** std_logic field

**field1** std_logic_vector[18:0] field

**field2** std_logic field

**field3** std_logic_vector[2:0] field

# 3 Example VHDL Register Access

All registers are bundled in records based on their mode. E.g. all RW registers are accessed through the record *bustype_rw_regs*. Access is also dependent on the type of register. All register of type SL, SLV and DEFAULT are all directly accessed by just specifying the mode record signal. E.g. the RW register *reg0* can be assigned a value like this (assuming AXI-bus):

```
axi_rw_regs.reg0 <= (others => '0');
```

Registers of type FIELD cannot be directly accessed without specification of a certain field. This is because the registers are implemented as a record in VHDL (thus a record of records). E.g. if the RO register *reg1* contains the field *field3* it can be accessed like this (assuming AXI-bus):

```
axi_ro_regs.reg1.field3 <= (others => '0');
```