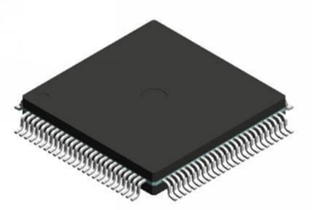
MORAL Peakrad SG13RH (S)

MORAL Rad-Hard Peakrad 32-bit MPU+FPU, 512KB SRAM +EDAC, SpaceWire, MIL-STD 1553C, 12-bit ADC, 12-bit DAC, SPI, I2C, CAN 2.0B

Datasheet - specification data

Features

- Core: **MORAL Peakrad 32-bit** CPU with FPU compliant to IEEE 754-2008 standard, DSP extension, and 2-level Memory Protection Unit. The new ISA is fully **orthogonal**, instructions and addressing modes equally treat each GPR, **regular**, machine modes up to 128 bytes, and **circular,** last GPR is a “logical” neighbor to the first GPR.
- Memory
  - 32-bit on-chip SRAM
  - 512KB of on-chip SRAM extended with EDAC (error detection and correction)
  - Memory controller for accessing external memories (PROM, EEPROM, and SRAM)
- Clock, reset, and supply
  - 1.2V core supply and 3.3V analog components and I/O supply
  - 50MHz system clock
  - 200MHz peripherals clock for PWM and SpaceWire
  - 10 clock cycle power-up and reset procedures
- Operating modes
  - System mode: all instruction and registers available
  - User mode: system instructions and special registers not accessible
- Debug mode
  - Read/write access to GPRs, special registers, and all other peripherals’ registers, as well as internal and external memory and I/O devices
  - Access over AHB master peripherals (UART or SpaceWire)
- 32-bit timers connected to AHB system bus: 2 system timers and 4 user timers



208 pin ceramic package

*illustrative image

- Communication interfaces
  - 2x SpaceWire: include RMAP (remote memory access protocol) function
  - 4x UART interfaces
  - 2x SPI interfaces
  - 2x I^{2}C interfaces: 8-bit oriented up to 1Mbit/s bidirectional and up to 5Mbit/s unidirectional
  - 2x CAN 2.0B interfaces: up to 1 Mbit/s
  - 1 x MIL-STD 1553C bus interface: raw data rate of 1 Mbit/s
- 2x 12-bit DAC converters
- 1x 12-bit ADC converter: select from 8 single-ended or 4 differential channels
- 1x interrupt controller (32 interrupt lines)
- 4x pulse counters
- 1x watchdog timer
- 1x PWM generator: independent 8-channel or complementary in up to four 2-channel pairs
- 64x GPIO ports (shared with internal peripherals I/Os)
- Technology: IHP SG13RH (S) process
- Rad-Hard Characteristics
  - Total ionizing dose (TID) 100 krad (Si)
  - Single event upset (SEU) for the processor’s digital core >30 MeVcm²/mg
  - Single event latch-up (SEL) >60 MeVcm²/mg

This is information on a product not yet in production, final product specification may changes

Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This datasheet provides the device characteristics of the MORAL Peakrad SG13RH (S) microcontroller.

Related documents

- PEAKTOP Instruction Set Architecture manual: “peaktop_isa_v1_3_10_5_r210104.pdf”
- SPI module datasheet: “SPI_Motorola.pdf”
- SpaceWire specification: “SPW_RMAP_Specification_Architecture.pdf”, EADS Astrium
- AMBA specification (ARM Limited, Rev 2.0, May 1999)
- CAN License Conditions (Robert Bosch GmbH, July 2010)
- DCAN Configurable CAN Bus Controller – User Manual (Digital Core Design, 2014)
- NXP Semiconductors - I2C-bus specification and user manual, Rev. 6, April 2014
- MIL-STD 1553C.pdf

2. DESCRIPTION

The MORAL Peakrad SG13H (S) microcontrollers incorporate the high-performance Peaktop 32-bit core with the its unique ISA operating at a 50 MHz frequency, high-speed embedded memory (up to 512 SRAM) with EDAC, and an extensive range of peripherals and I/Os. Communication interfaces (two SpaceWire, four UARTSs, two I2Cs, two SPIs, two CAN 2.0B, and one MIL-STD 1553C), one 12-bit ADC, two 12-bit DAC, six timers, four pulse counters, one watchdog timer, one interrupt controller, and one PWM generator with independent 8-channels or four 2-channel pairs. .

The microcontroller operate in the -40 to +125 °C temperature, 1.2V and 3.3V power supply. Radiation hardness parameters include total ionizing dose (TID) of 100 krad (Si), single event upset (SEU) for the processor's digital core >30 MeVcm^2/mg, and single event latch-up (SEL) >60 MeVcm^2/mg.

The description below provides an overview of the microcontroller functionalities and peripherals. These features make the MORAL Peakrad SG13H (S) microcontrollers suitable for a wide range of space applications, such as propulsion systems, robotics applications, motor & mechanism (antenna pointing), DC/DC conversion, thermal regulation, particle detectors, radiation monitoring, etc.

3. DEFINITIONS

Data/Address format:

The natural data width of the Peaktop core used in the MORAL microcontroller is 32 bits. However, data can be accessed in 8, 16, 32 and 64 bit width. Some registers are defined in 16- or 8-bit format. In this case all remaining higher significant bits are not used/implemented and therefore undefined (usually zero) for read operations. Write operations to those undefined bits are without any effect.

Each register can be addressed by a 32-bit base address of its associated module plus an individual register offset.

Data is generally stored in little endian format (LSB at the lowest byte address of a word). However, the memory controller can dynamically change the endianness of the external memory or IO banks.

Register bit-field definitions:

All registers described in this specification contain bit-fields with the properties “access type” (R/W) and “reset state” (Reset). All access types and reset states are defined below:

Syntax of register access types - user and system mode:

rw	- read and write
ro	- read only
wo	- write only

Syntax of register access types - system mode only:

rw(s)	- read and write, system mode

Register reset states:

0	- single bit, reset to “low” state
0x0	- bit-field, reset to “low” state
1	- single bit, reset to “high” state
010…	- bit-field, reset to binary value (example)
0x0F…	- bit-field, reset to hexadecimal value (example)

4. MORAL MICROCONTROLLER SYSTEM ARCHITECTURE

4.1. MORAL Microcontroller Block Diagram

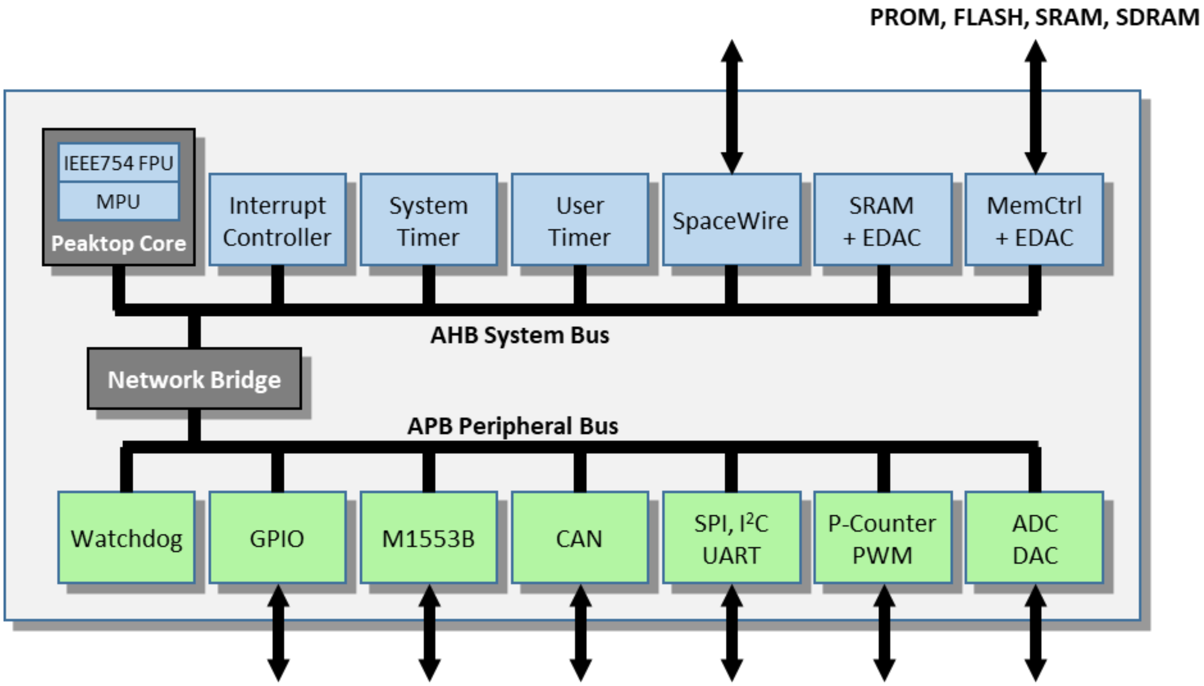


Figure 1: MORAL – Microcontroller Chip Architecture Block Diagram

Implemented IP blocks:

Following is the list of modules and peripherals integrated:

- Peaktop core with FPU, MPU, and DSP extension
- 256/512 Kbyte SRAM with EDAC protection unit
- 1 x Interrupt Controller (32 interrupt lines)
- 6 x Timers (2x system + 4x user timers)
- 1 x 8-channel 12-bit ADC module
- 2 x DAC modules
- 2 x CAN 2.0 modules
- 4 x UART interfaces
- 2 x SPI interfaces
- 2 x SpaceWire interfaces (incl. RMAP protocol)
- 1 x MIL-STD 1553C bus interface (incl. redundant bus)
- 2 x I^{2}C interfaces
- 1 x 8-channel PWM generator
- 64 x GPIO ports (shared with internal peripherals I/Os)
- 4 x Pulse counters
- 1 x Watchdog timer

4.2. Clocks, Power-up and Reset

4.2.1. Clocks

The microcontroller has two external clock input pins:

- CLK is the main system clock operating at 50 MHz.
- PCLK is a peripheral clock operating at 200 MHz required by the PWM and SpaceWire modules

All internal clocks are derived from these two external clocks. The clocks (for some components) can be pre-scaled and turned-off (see Subsection 6.1.4).

4.2.2. Power-up Procedure

Since the microcontroller needs two power supplies (1.2 V core voltage and a 3.3 V I/O voltage), a power-up sequence is possible. The only requirement is to avoid that I/O power becomes active after core power is already active. This may lead to high current peaks, but will not destroy anything. Simultaneously switching-on both power supplies is recommended.

During power-up procedure the reset signal should be active (low). After the power becomes stable the reset signal should stay active for at least 10 clock cycles before releasing it to inactive state (hi).

4.2.3. Reset Procedure

The microcontroller has a synchronous reset signal. All registers and flip-flops in the microcontroller (except for some flip-flops in the SpaceWire modules) are synchronously reset. Although the external reset signal may be activated asynchronously, the reset of the internal flip-flops is synchronized with the system clock. The external reset signal passes through several synchronizers within the chip. Therefore, at least 10 clock cycles with active reset signal are required to properly reset the entire microcontroller.

After releasing the reset to inactive state the first instruction fetch starts always at address 0x0 from an external memory location in bank zero (selected by chip select zero CS0).

Parallel to the external reset, an internal watchdog timer reset is implemented (see Subsection 7.1).

4.3. Debug Support

4.3.1. Debug Unit

The debug unit of the MORAL microcontroller chip is used for two purposes:

- To put the entire chip in ‘chip debug mode’ in which the PEAKTOP core is held totally inactive, i.e., not executing nor fetching instructions or data.
- Once in chip debug mode, enable read/write access to the GPRs and special registers over AHB master peripherals (UART or SpaceWire components). Of course, in this mode, the master peripherals can also access all other registers in all other peripherals, as well as internal and external memory and I/O devices.

Chip debug mode can be activated in one of two ways:

- By hardware: asserting the ‘hold’ input signal of the chip (i.e., transition of ‘hold’ from 0 to 1)
- By software: writing 1 to the DBGM bit of the Debug Unit control register (see Figure 2)

On the other hand chip debug mode is deactivated by:

- Hardware: de-asserting the ‘hold’ input signal of the chip (setting ‘hold’ to 0)
- Software: writing 0 to the DBGM bit of the Debug Unit control register (see Figure 2). However, if ‘hold’ is asserted, writing 0 to the DBGM bit does not have an effect.

If the ‘hold’ signal is asserted to 1 during reset, then the chip immediately enters debug mode after de-asserting the `reset’ signal. Of course, during debugging ‘hold’ has to stay asserted to 1. Otherwise, when ‘hold’ is de-asserted to 0, normal chip operation continues.

Figure 2 shows the control register of the Debug Unit (APB base address: 0x8100 0000).

Bit	31:4	3	2	1	0
Name	reserved	SPW1	SPW0	UART	DBGM
R/W	undefined	rw	rw	rw	rw
Reset		0	0	1	0

Figure 2: Debug Unit Control Register *(Address Offset: 0x00)*

Name	Description
DBGM	Debug mode (0 – debug mode inactive; 1 – debug mode active)
UART	Debug over UART (0 – don’t debug over UART; 1- debug over UART)
SPW0	Debug over SPW0 (0 – don’t debug over SPW 0); 1 – debug over SPW 0)
SPW1	Debug over SPW1 (0 – don’t debug over SPW 1); 1 – debug over SPW 1)

Table 1: Debug Unit Control Register Description

In order to access the control register of the Debug Unit as well as the GPRs and Special Registers of the PEAKTOP core, the following addressing scheme is employed (APB base address: 0x8100 0000):

Offset Address[9:0]	Register Description
0x000	Debug Unit control register (Figure 2)
0x200 – 0x27C	Bank 0: GPRs 0 – 31 (at word-aligned addresses)
0x280 – 0x2FC	Bank 1: GPRs 0 – 31 (at word-aligned addresses)
0x300 – 0x36C	Special Registers 0 – 27 (at word-aligned addresses)

Table 2: Debug Unit Registers Address Range

The GPRs in the two GPR banks and the Special Registers can be accessed only when the chip is in ‘chip debug mode’, while the Debug Unit control register can be accessed anytime.

4.3.2. Chip Debug Mode

As said, in *chip debug mode* the PEAKTOP core is totally inactive. However, upon entering the debug mode in the middle of program execution, all the instructions that are currently in the pipeline will finish execution and then the core will be held inactive. Thus, a delay of up to the execution of 8 instructions may occur. In *chip debug mode* all GPRs and special registers in the PEAKTOP core, as well as all registers of peripherals within the chip can be accessed according to their read/write permissions by the other AHB masters, i.e., the Debug UART (see Subsection 4.3.3) or one of the two SpaceWire components (see Subsection 6.6). Furthermore, internal and external memory and I/O devices can be also accessed.

Breakpoints (i.e., points at which the processor enters the *chip debug mode*) can be software-defined within the program by issuing a store instruction that writes a 1 to the DBGM bit of the control register of the Debug Unit (see Figure 2).

When the ‘*chip debug mode’* is deactivated, the PEAKTOP core continues execution normally after the last executed instruction before holding execution. Of course, since now the pipeline is empty, execution continues with initial latency.

4.3.3. Debug UART

The debug UART is the same UART component described in Subsection 7.7 with the addition of logic to operate not only as an APB slave but also as an AHB master, as well as to perform special debugging functions. Thus, the Debug UART can be dynamically put by software in one of three modes:

- APB slave mode (completely the same as the other three UART components – Subsection 7.7)
- AHB master mode (in this mode only half-duplex operation is possible)
- UART-debug mode (the same as AHB master mode, with additional debug-support features)

The following figures describe the additional Debug UART registers, besides the ones described in Subsection 7.7, i.e., the four registers: data, control, status and scaler of Figure 59 - Figure 62.

The Debug UART (UART0) APB base address is 0x8100 6000:

Bit	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
Name	Start Address																															
R/W	rw																															
Reset	undefined																															

Figure 3: Start Address Register *(Address Offset: 0x20)*

Bit	31:19	18:7	6:4	3	2	1	0
Name	reserved	NRCHNK	MODE	WR	RXIRQEN	TXIRQEN	EN
R/W	undefined	rw					
Reset		0x0					

Figure 4: Command Register *(Address Offset: 0x28)*

Name	Description
EN	AHB mode enable (0-disable; 1-enable)
TXIRQEN	Enable Tx IRQs (0-disable; 1-enable)
RXIRQEN	Enable Rx IRQs (0-disable; 1-enable)
WR	Write command (0-read command; 1-write command)
MODE[6:4]	Machine mode (000-8 bits, 001-16 bits, 010-32 bits, …, 111-1024 bits)
NRCHNK[18:7]	The number of data chunks to be read/written in machine mode MODE

Table 3: Command Register Descriptions

Bit	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
Name	Current Address																															
R/W	ro																															
Reset	undefined																															

Figure 5: Current Address Register *(Address Offset: 0x30)*

Bit	31:29	28:17	16:7	6	5	4	3	2	1	0
Name	reserved	CHNKLEFT	CURBYTE	RXCOMM	RX	TX	AHBERR	RXIRQ	TXIRQ	DBGM
R/W	undefined	ro								
Reset		0x0								

Figure 6: AHB Debug Status Register *(Address Offset: 0x38)*

Name	Description
DBGM	UART in Debug mode (0-UART not in debug mode; 1-UART in debug mode)
TXIRQ	Tx IRQ occurred (0-did not occur; 1-occurred) – reset to 0 on reading this register
RXIRQ	Rx IRQ occurred (0-did not occur; 1-occurred) – reset to 0 on reading this register
AHBERR	AHB error (0-no error; 1-error) – reset to 0 on reading this register
TX	UART transmitting (0-not transmitting; 1-transmitting)
RX	UART receiving (0-not receiving; 1-receiving)
RXCOMM	UART receiving Debug command (for DBGM==1 only) (0-not receiving; 1-receiving)
CURBYTE[16:7]	The current byte which is being received/transmitted in the current data chunk
CHNKLEFT[28:17]	The number of data chunks left to be read/written in machine mode 'MODE'

Table 4: AHB Debug Status Register Descriptions

Address[5:0]	Register Description
0x00	Data Register (see Figure 59)
0x08	Status Register (see Figure 61)
0x10	Control Register (see Figure 60)
0x18	Scaler Reload Register (see Figure 62)
0x20	Start Address Register (see Figure 3)
0x28	Command Register (see Figure 4)
0x30	Current Address Register (see Figure 5)
0x38	AHB Debug Status Register (see Figure 6)

Table 5: Debug UART Registers Address Offsets

4.3.3.1. APB Slave Mode

The operation in the APB slave mode is described in Subsection 7.7. The same four registers are here available under the addresses 0x00 – 0x18 given in Table 5. The other four registers at addresses 0x20 – 0x38 are not used in this mode. This mode supports full-duplex operation.

This mode is the default mode after chip reset. In order to return from another mode to the APB slave mode the *chip debug mode* should be set inactive (by de-asserting the 'hold' signal and/or by resetting the DBGM bit of the Debug Unit control register (see Figure 2) to 0), and by resetting the EN bit of the Debug UART command register to 0 (see Figure 4).

4.3.3.2. AHB Master Mode

AHB master mode is entered by setting the EN bit of the Debug UART command register to 1 (see Figure 4) under the assumption that *chip debug mode* is inactive. In this mode, the UART operates as AHB master in which only half-duplex operation is supported.

In order to transmit over the TxD line a certain number of bytes residing in memory at a given address, this address should be written in the Start address register (Figure 3), and the command register (Figure 4) should be written appropriately, i.e., the EN bit should be 1 and the WR bit should be 0. Furthermore, transmitting for example, five 64-bit values would require writing the MODE field with value 3 and the NRCHNK field with the number 5.

Similarly, in order to set the UART to receive a certain number of bytes at the RxD line and store them in memory at a given address, the start and command registers should be written as in the transmit case with the difference that the WR bit of the command register should be 1.

After transmitting/receiving the programmed number of bytes the Debug UART component raises a single pulse Rx/Tx IRQ, if the RXIRQEN/TXIRQEN bits are 1, respectively.

The status of the UART as well as the current address at which data is transmitted/received can be inspected by reading the Current address and AHB-DEBUG status registers.

4.3.3.3. UART Debug Mode

UART-debug mode is the same as the AHB master mode with some additional protocol that eases debugging. UART-debug mode is entered by setting the EN bit of the Debug UART command register to 1 (see Figure 4) under the assumption that *chip debug mode* is active. In this mode, the UART operates as AHB master in which only half-duplex operation is supported.

Entering UART-debug mode automatically presets the Start and Current address registers (Figure 3 and Figure 5) to the address of the Start address register, while the Command register (Figure 4) is set to receive two 64-bit values. The DBGM, RX and RXCOMM fields of the AHB-DEBUG status register (Figure 6) are also asserted and the CHNKLEFT is set to 2 (TX and CURBYTE are 0). Furthermore, the UART Rx/Tx FIFOs and Rx/Tx shift registers are emptied and the baud rate register is set to a corresponding value for a transmission rate of 115200 bps according to the input 50 MHz. In UART-debug mode this procedure is not only performed on entering the UART-debug mode, but it is also performed after executing each Rx/Tx command programmed in the Command register, thus expecting the following Debug command on the Rx input line.

Once a command is received over the Rx line, the Debug UART starts executing it, i.e., it starts transmitting or expects receiving the commanded number of bytes (calculated by the NRCHNK and MODE fields) at the programmed memory (or register) addresses.

The used UART data frame in this mode is the one of Figure 58, without parity bit and with a single stop bit. The scaler reload register (Figure 62) is zero, which means that the baud rate is 1/8 of the main clock frequency.

In this mode, interrupts are not raised since the 16-bit control register (Figure 60) is set to the value of 0x0600, i.e., only the receiver and transmitter is enabled and everything else is disabled.

4.3.4. PEAKTOP Core Debug Mode

The PEAKTOP core can be put in its own debug mode by writing bit 1 of the System Control Register (see page 53 of [01]). In this mode, the “DEBUG MODE EXCEPTION” is raised after each executed instruction (see page 36 of [01]) of the user program. Note that the *PEAKTOP core debug mode* is independent of the *chip debug mode* described in Subsection 4.3.2 since it is encapsulated in the PEACTOP core. That is, *PEAKTOP core debug mode* can be entered without entering the *chip debug mode*. On the other hand, if the *chip debug mode* is active, the PEAKTOP core is anyway inactive and does not execute instructions, i.e., there is no use of activating *PEAKTOP core debug mode* when the entire chip is in the *chip debug mode*.

In the *PEAKTOP core debug mode*, after each executed user program instruction, the program is transferred to the DEBUG MODE EXCEPTION handler. However, the handler itself is NOT interrupted after each instruction like the user program since all exceptions are automatically disabled upon entering exception handling [01]. The handler program can be used for various diagnostic procedures. Breakpoints (i.e., points at which the processor enters the *PEAKTOP core debug mode*) are here simply implemented by instructions which set bit 1 of the System Control Register.

5. PEAKTOP CORE AND INTERCONNECT SYSTEM

5.1. PEAKTOP Core

One of the main building blocks of the MORAL microcontroller is the PEAKTOP execution unit whose components are shown in Figure 7.

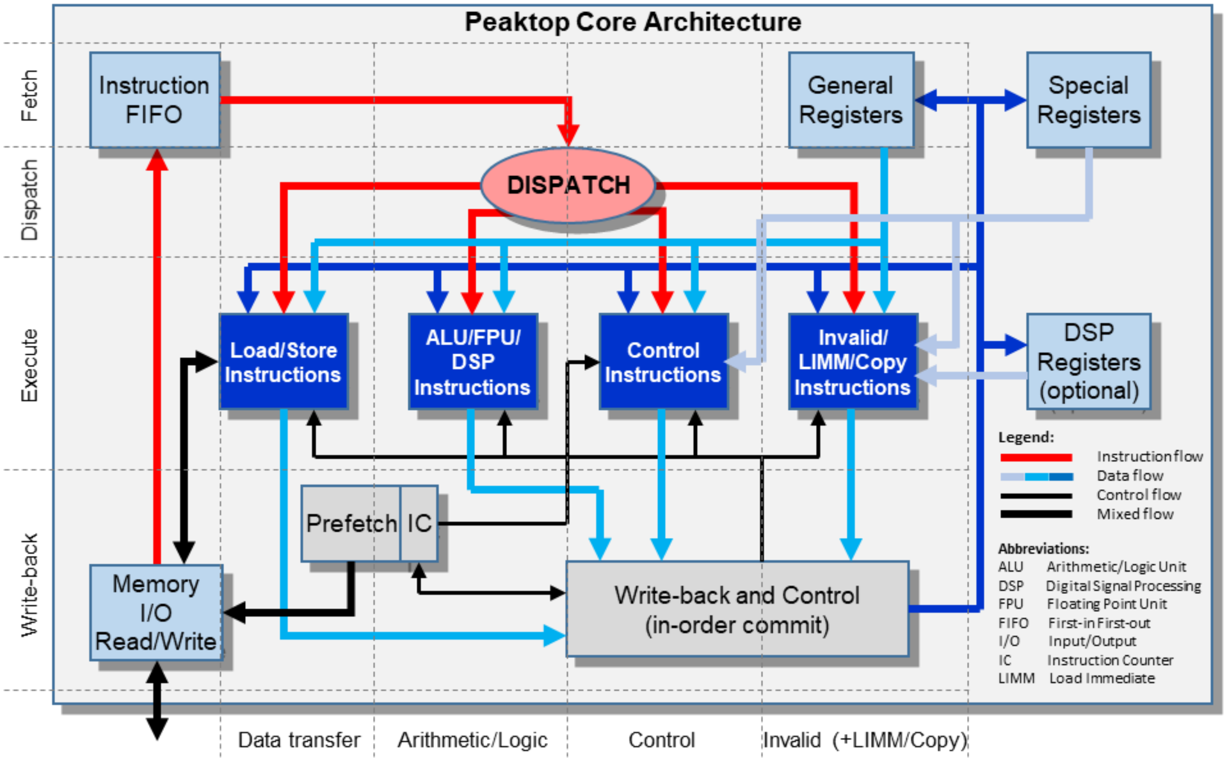


Figure 7: PEAKTOP Execution Unit - Block Diagram

Here we outline the PEAKTOP Instruction Set Architecture (ISA) of the MORAL microcontroller:

- Simple and flexible general-purpose RISC architecture for real-time and embedded processing as well as for general data processing
- 32 General Purpose Registers (GPR) with multiple register banks for fast context switches
- Execution pipeline with optional IEEE 754 compliant Floating Point Unit (FPU) and Digital Signal Processing (DSP) extension
- Orthogonal – all instructions and addressing modes equally treat each GPR
- Regular – machine modes of 1, 2, and 4 bytes; each operation applies to all implemented machine modes
- Circular – The last GPR is a “logical” neighbor to the first GPR, which is important for instructions that output results in consecutive registers (e.g., multiplication of two 32-bit GPRs will produce a 64-bit result to be written-back in two consecutive GPRs)
- Scientific architecture – Orthogonality, regularity and circularity make the architecture a natural platform for scientific problems
- Multiprocessing support (cache coherence, synchronization support, atomic transfers)
- Configurable addressing space: up to 64-bit; virtual address space is up to 128-bits
- 32-bit wide instructions with 0, 1, 2 or 3 operands;

- 12-bit load/store signed address offset; 18-bit load signed/unsigned immediate; 14-bit ALU signed/unsigned immediate; 20-bit signed offset for jumps; 14-bit signed offset for branches;
- Jumps and branches can be IC-relative or absolute, and both using register or offset;
- Aligned instruction memory addressing; data addressing could be both aligned or non-aligned;
- Dedicated hypervisor mode
- Addressing modes are:
    - Register
    - Displacement: base + offset (12-bit signed offset)
    - Indexed: base + index; including automatic pre- and post- increment and decrement of the index according to the number of accessed bytes
    - Immediate: signed/unsigned immediate (18-bit for load and 14-bit for ALU operations)
- 2-level Memory Protection Unit (MPU)
- Optional L1 (separate data/instruction) cache, and L2 cache, with configurable policies such as write-back, write-through, allocate/no-allocate on write, etc.
- Ported GCC compiler for C/C++ and toolchain (assembler, debugger, etc.) are available

The PEAKTOP ISA is described in details in [01].

5.1.1. Floating Point Unit (FPU)

The MORAL microcontroller has a 32-bit (single-precision) FPU compliant to the IEEE 754-2008 standard. The FPU unit implements all Floating Point (FP) instructions according to [01] with the following exceptions:

- FREM and FSQR are not implemented since they introduce significant complexity and area, and their software substitution with FADD, FSUB, FMUL and FDIV is straightforward, i.e., the cost-benefit analyses in light of the purposes of the MORAL project showed that it is better not to implement them.
- FEXT and FSQZ are not implemented since they are not needed: only the 32-bit binary IEEE format is implemented which implies that no conversions between different formats (except between binary FP and integer) is actually required.
- FMAD and FMSU are not implemented for the same reason as FREM and FSQR

5.1.2. Operating System (OS) Support

The OS support provided by the MORAL microcontroller is based on two main facilities:

- PEAKTOP operating modes (see also [01])
- Memory Protection Unit (MPU)

5.1.2.1. PEAKTOP operating modes

The PEAKTOP ISA defines two operating modes:

System	- all instructions can be executed, and all registers can be accessed.
User	- system instructions cannot be executed and some special registers cannot be accessed.

An attempt in user mode to execute a system instruction raises the SYSTEM INSTRUCTION exception (see Subsection 5.1.2.1.3).

The execution starts in system mode. Resetting the 0-th bit of the SYSTEM CONTROL REGISTER switches to execution in user mode. However, only a raised, potent exception, potent interrupt or NMI switches to execution in system mode. Thus, for example, an attempt to write to the SYSTEM CONTROL REGISTER in user mode will raise an exception, which if potent (i.e., not masked and not disabled), will transfer execution to the exception handler in which the operating system can determine the operating mode.

With these simple mechanisms, the PEAKTOP ISA satisfies the virtualization requirements (classic virtualization – trap-and-emulate).

5.1.2.1.1. SYSTEM CONTROL REGISTER

The SYSTEM CONTROL REGISTER has several functions that control the behavior of the system. It is only writable in system mode. The bits and bit-fields of the register are as follows.

Bit	31:8	7:4	3	2	1	0
Name	reserved	GPR Bank	Enable Interrupts	Enable Exceptions	Debug Mode	System Mode
R/W	undefined	ro, rw(s)				ro, rw(s)
Reset		0x0	0	0	0	1

Figure 8: System Control Register (no bus address)

Name	Description
SYSTEM MODE	If 1, the system mode is set. If 0, the user mode is set. This bit is set to 1 after reset, i.e., the system starts in system mode.
DEBUG MODE	If 1, the debug mode is set, in which the DEBUG MODE EXCEPTION is raised after each executed instruction. For normal operation this bit should be 0.
ENABLE EXCEPTIONS	If 1, the exceptions are enabled. If 0, the exceptions are disabled.
ENABLE INTERRUPTS	If 1, the interrupts are enabled. If 0, the interrupts are disabled.
GPR BANK	The number of the currently used GPR bank. Theoretically, up to 16 banks can be implemented, but practically two to four banks are implemented. In the MORAL microcontroller there are two banks. Specifying a number greater than or equal to the number of implemented banks raises the UNIMPLEMENTED GPR BANK exception.
reserved	Not used

Table 6: System Control Register Description

5.1.2.1.2. System Instructions

System instructions are instructions that can be executed only in system mode and are used for operating system protection. An attempt to execute a system instruction in user mode raises the SYSTEM INSTRUCTION exception (see next subsection 3.1.3.1.3).

Any inter-register transfer instruction (COPY) in which the destination is a special register that is non-writable in user mode is a system instruction (see Table 7).

Name	Alias	Register	User acc.	Sys. acc.
spc0	IMP	IMPLEMENTATION REGISTER	r-	r-
spc1	IMP2			
spc2	EST	EXECUTION STATUS	r-	rw
spc3	EXI	EXCEPTION INSTRUCTION	r-	rw
spc4	EXC	EXCEPTION REGISTER	r-	rw
spc5	EXC2			
spc6	EXM	EXCEPTION MASKS	r-	rw
spc7	EXM2			

Name	Alias	Register	User acc.	Sys. acc.
spc8	ETB	EXCEPTION TABLE BASE ADDRESS	r-	rw
spc9	ETB2			
spc10	ITB	INTERRUPT TABLE BASE ADDRESS	r-	rw
spc11	ITB2			
spc12	CID	CORE ID	r-	rw
spc13	PID	PROCESS ID	r-	rw
spc14	PID2			
spc15	SCR	SYSTEM CONTROL REGISTER	r-	rw
spc16	NRP	NMI RETURN POINTER	r-	rw
spc17	NRP2			
spc18	ERP	EXCEPTION RETURN POINTER	r-	rw
spc19	ERP2			
spc20	UCR	USER CONTROL REGISTER	rw	rw
spc21	CRP	CALL RETURN POINTER	rw	rw
spc22	CRP2			
spc23	IRP	INTERRUPT RETURN POINTER	rw	rw
spc24	IRP2			
spc25	DCR	DSP CONFIGURATION REGISTER	rw	rw
spc26	DCR2			

Table 7: **Debug Unit Special Registers** (Base Address: 0x8100 0300)

Thus, a COPY (MOV) instruction whose destination is a register in the range spc0-spc19 is a system instruction. Executing such an instruction in user mode will raise the SYSTEM INSTRUCTION exception.

The return from exception and return from NMI handler (RETE and RETN) instructions are also system instructions. Executing RETE or RETN in user mode will raise the SYSTEM INSTRUCTION exception.

5.1.2.1.3. SYSTEM INSTRUCTION exception

An attempt to execute a system instruction in user mode raises this exception. This exception is raised by RETE, RETN and by any inter-register transfer (COPY) executed in user mode in which the destination is a special register that cannot be written in user mode (see Table 7).

If the exception is impotent (i.e., it is masked or disabled) the exceptional instruction is skipped. The exceptional instruction is written in the EXCEPTION INSTRUCTION register, and the exception is noted in the EXCEPTION REGISTER (see Table 7). No other register change is made.

5.1.2.2. Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) allows organization of the memory in **segments**. The entire memory space is accessible in **system mode**. However, when the MPU is active in **user mode**, only the specified memory segments can be accessed. The MPU is activated/deactivated only in system mode by setting/resetting the **active** bit in the **control register** (see Subsection b)).

All registers in the MPU are 64-bit and can be accessed with byte, halfword, word or doubleword machine mode (8, 16, 32 or 64 bits). However, on write, the entire contents of the register will be changed regardless of the machine mode, therefore the mode should be carefully selected. Furthermore, the access should be aligned to a doubleword address, i.e., the three address LSBs should be zero.

The address ADDR is 32-bit wide, i.e., ADDR[31:0].

5.1.2.2.1. Memory Segment Entries (MSE)

In total, 16 memory segments can be specified in the 16 Memory Segment Entries (MSEs) which are all 64-bit wide. Figure 9: shows the MSE fields. MSE base address is 0x80100008 (= MSE0). The 16 MSE address offsets are 0x08, 0x18, 0x28, …, 0xF8.

Bit	63:35	34:32	31:3	2	1	0
Name	TADR	reserved	BADR	RD	WR	EXE
R/W	rw	undefined	rw	rw	rw	rw
Reset	undefined		undefined	0	0	0

Figure 9: Memory Segment Entry (MSE)

The bit fields of the MSEs are the following:

Name	Description
exe	Memory segment is executable (if this bit value is 1), non-executable (bit value 0)
wr	Memory segment is writable (if this bit value is 1), non-writable (bit value 0)
rd	Memory segment is readable (if this bit value is 1), non-readable (bit value 0)
badr	The base (start) address of the segment
reserved	Reserved 3-bit field
tadr	The top address limit of the segment (higher than the base address)

Table 8: MSE Description

Note that BADR and TADR implicitly assume that the three Least Significant Bits (LSBs) of the address are zero, i.e., BADR and TADR give the higher 29 bits of the address. Thus, the minimal possible size of a segment is 8 bytes;

On reset, the RD, WR and EXE bits are zero, while the TADR and BADR fields are undefined. The reserved field is always read as zero. Access to the MSEs is only allowed in system mode.

a) Access evaluation

When the memory is accessed in user mode, the MPU investigates two conditions in all MSEs starting from MSE0 to MSE15:

1) whether the specified memory access type (**read**, **write**, **instruction fetch**) is permitted according to the values of the RD, WR and EXE bits of the MSE;
2) whether the specified memory address ADDR lies within the specified range in the MSE, i.e., if

$$\mathbf{((BADR \ll 3) \le ADDR[31:3]) \quad and \quad (ADDR[31:3] \le TADR)}$$

where the << operator is a logical left shift (the shift is for three bits in this case). Then, if at least one MSE satisfies both conditions 1) and 2) the MPU grants access to memory. Of course, if more than one MSE satisfy both conditions 1) and 2) the MPU will grant access to memory.

On the other side, if the conditions 1) and 2) are not satisfied by at least one MSE, an I/D SYSTEM BUS ERROR exception is triggered. The violating address ADDR is written in the **Violating address register (VIOL)** together with the violating condition (see Subsection a)).

b) Special segments for peripheral access

An **on-chip** peripheral access is specified when the two Most Significant Bits (MSBs) of the address ADDR are '10'. An **off-chip** peripheral access is specified when the two MSBs of the address ADDR is are '11'.

The **control register** (see Subsection b)) has bits that specify the read, write and execute access of these **special segments** (on-chip and off-chip peripherals).

5.1.2.2.2. MPU registers

The MPU has two 64-bit registers, the read-only Violating address register (VIOL) and the Control register (CNTRL). Base address is 0x8010 0000.

The access to the MPU registers is only allowed in **system mode**.

a) Violating address register (VIOL)

The violating address register contains the address that caused the violation (shown in Figure 10:

Bit	63:32	31:7	6	5	4	3:2	1	0
Name	VADR	reserved	RD	WR	EXE	reserved	OF	NF
R/W	ro	ro	ro			ro	ro	
Reset	undefined	0x0	0x0			0x0	0x0	

Figure 10: Violating Address Register (*Address Offset*: 0x00)

The bit fields of the VIOL register are the following:

Name	Description
NF	Not found – address out of all defined segments
OF	Overflow access on the tadr address (or on the last address of on/off-chip peripheral areas)
reserved	Reserved (unused) 2-bit field
EXE	Execution attempt in a segment that does not allow instruction execution
WR	Write attempt in a segment that is non-writable
RD	Read attempt in a segment that is non-readable
reserved	Reserved (unused) 25-bit field
VADR	Violating address

Table 9: VIOL Register Description

The NF, OF, EXE, WR and RD are automatically reset to zero on a read of the VIOL register.

The VIOL register is updated on a violation if the previous violation was acknowledged by reading this register. If a memory access violation occurs and this register is not read, it will not be updated on successive violations.

The VIOL register is read-only. On reset, the value of the NF, OF, EXE, WR and RD bits of the VIOL register are zero while VADR is undefined. The reserved fields are always read as zero.

b) Control register (CNTRL)

The 64-bit control register is shown in Figure 11:

Bit	63:12	11	10	9	8	7	6	5	4	3:2	1	0
Name	reserved	UF	RF	WF	XF	UN	RN	WN	XN	reserved	FLUSH	ACTIVE
R/W	undefined	rw	rw	rw	rw	rw	rw	rw	rw	undefined	rw	rw
Reset		0	0	0	0	0	0	0	0		0	0

Figure 11: Control Register (*Address Offset*: 0x10)

The fields of the CNTRL register are the following.

Name	Description
ACTIVE	Active MPU (0 – inactive, 1 – active)
FLUSH	Set the RD, WR and EXE bits of all MSEs in the MPU to zero (auto-reset to zero)
reserved	Reserved 2-bit field (can be written and read, but has no function)
XN	Execution from on-chip peripheral area (0 – not allowed, 1 – allowed)
WN	Write to on-chip peripheral area (0 – not allowed, 1 – allowed)
RN	Read from on-chip peripheral area (0 – not allowed, 1 – allowed)
UN	Use the XN, WN and RN bits for on-chip peripheral area control (0 – don't use, 1 – use)
XF	Execution from off-chip peripheral area (0 – not allowed, 1 – allowed)
WF	Write to off-chip peripheral area (0 – not allowed, 1 – allowed)
RF	Read from off-chip peripheral area (0 – not allowed, 1 – allowed)
UF	Use the XF, WF and RF bits for off-chip peripheral area control (0 – don't use, 1 – use)
reserved	Reserved 52-bit field (always zero)

Table 10: CNTRL Register Description

Note that the flush bit is auto-reset to zero whenever written with one. That is, when reading the CNTRL register this bit will always be zero.

If the UN/UF bits are set, an on-chip/off-chip peripheral access is evaluated according to the RN, WN, XN / RF, WF, XF bits respectively, and the MSEs are not consulted at all.

On reset, all bits have the value of zero.

5.1.2.2.3. Addresses of registers and MSEs

In order for the MPU to evaluate an address in user mode, a special **sel** signal is used to signal an address for evaluation. On the other side, for a read/write access to the MSEs or the registers in system mode, a special **hsel** signal is used to signal such access. The registers/MSEs should be mapped in the on-chip address area (i.e., for which the two address MSBs are '10'). The read/write access is additionally differentiated by the **write** signal, which for 0 implies read access, while for 1 implies write access.

The registers/MSEs can be accessed only in **system mode**, otherwise a D SYSTEM BUS ERROR exception is triggered.

For read/write access to the MSEs and the registers a subset of the eight LSBs of the address ADDR is used, i.e., ADDR[7:0]. The base address is 0x8010 0000. Table 11 gives the addresses of two registers and 16 MSEs.

Offset ADDR[7:0] (bin)	ADDR[7:0] (hex)	Access to	Group
0000 0000	0x00	VIOL	Registers
0001 0000	0x10	CNTRL	
0000 1000	0x08	MSE0	MSE
0001 1000	0x18	MSE1	
0010 1000	0x28	MSE2	
0011 1000	0x38	MSE3	
0100 1000	0x48	MSE4	

Offset ADDR[7:0] (bin)	ADDR[7:0] (hex)	Access to	Group
0101 1000	0x58	MSE5	
0110 1000	0x68	MSE6	
0111 1000	0x78	MSE7	
1000 1000	0x88	MSE8	
1001 1000	0x98	MSE9	
1010 1000	0xA8	MSE10	
1011 1000	0xB8	MSE11	
1100 1000	0xC8	MSE12	
1101 1000	0xD8	MSE13	
1110 1000	0xE8	MSE14	
1111 1000	0xF8	MSE15	

Table 11: Offsets of MPU Registers and MSEs

It can be seen from Table 11 that the addresses of the registers are differentiated from the addresses of the MSEs by the ADDR[3] bit. That is,

for ADDR[3] == 0:

- VIOL is accessed if ADDR[7:4] is 0x0
- CNTRL is accessed if ADDR[7:4] is 0x1

while for ADDR[3] == 1:

- MSE0 – MSE15 are accessed correspondingly to the ADDR[7:4] binary code 0 – 15.

The ADDR[2:0] bits are always zero.

5.1.2.2.4. Violation handling

The VIOL register sets automatically the address that caused the violation, and sets additional bits in order for the handler to easily determine the reason for the violation.

An opcode fetch triggers the I SYSTEM BUS ERROR exception, while a read/write violation triggers the D SYSTEM BUS ERROR exception. The EXCEPTION INSTRUCTION (EXI) register further tells whether a load or a store was performed.

5.2. System and Peripheral Bus

The on-chip components are interconnected by AMBA-like buses [04]. Components which are closer to the PEATKOP core are interconnected by the AHB-like bus, while slower peripherals are connected by the APB compatible bus. The AHB and APB buses are connected over a bus bridge. The APB bus is fully compatible to [04], while the AHB bus has the same interface signals as in [04] but a different protocol.

5.2.1. AMBA–like High Speed Bus (AHB)

High performance on-chip peripherals are interconnected to the system network by a high-speed AHB-like system bus (AHB, system bus). The AHB bus in the MORAL microcontroller is not fully compliant to the AMBA 2.0 specification of ARM [04].

5.2.2. AMBA-compliant Peripheral Bus (APB)

Low performance on-chip peripherals are interconnected to the system network by an AMBA 2.0 compliant peripheral bus (APB, peripheral bus). The peripheral bus of the MORAL microcontroller is fully compliant to the ARM APB specification [04].

5.2.3. AHB-APB Network bridge

The high-speed and peripheral buses are interconnected by an AHB-APB network bridge [04].

5.3. On-chip Registers

Table 12 shows all register addresses accessible via the system and peripheral bus with bold type marked base addresses of the related modules.

Address	Register	Address	Register
0x80000000	**IREG module registers:**	**0x8100A000**	**SpaceWire 0 registers:**
0x80000000	IREG version register	0x8100A000	Configuration register
0x80000004	IREG global configuration register	0x8100A004	Control register
0x80000008	IREG on-chip SRAM control register	0x8100A008	Status register
0x8000000C	IREG on-chip SRAM status register	0x8100A00C	Node Address / Dest. Key / PID register
0x80000010	IREG PWM pre-scaler register	0x8100A010	Max TX Packet Size register
0x80000014	IREG SPW0 pre-scaler register	0x8100A014	Max RX Packet Size register
0x80000018	IREG SPW1 pre-scaler register	0x8100A018	Time Out register
0x80100000	**MPU registers:**	0x8100A01C	Time Code register
0x80100000	Violating address register	0x8100A020	IT Pending / Force register
0x80100010	Control register	0x8100A024	IT Clear register
0x80100008	MSE0	0x8100A028	IT Mask register
0x80100018	MSE1	0x8100A02C	TX Segments Descriptor Address register
0x80100028	MSE2	0x8100A030	RX Packets Descriptor Address register
0x80100038	MSE3	0x8100A034	RX Packets Descriptor End Addr. register
0x80100048	MSE4	0x8100A038	Start Address 1 register
0x80100058	MSE5	0x8100A03C	Start Address 2 register
0x80100068	MSE6	0x8100A040	RX areas size register
0x80100078	MSE7	0x8100A044	Current Buffer End register
0x80100088	MSE8	0x8100A048	AHB Burst disable register
0x80100098	MSE9	0x8100A04C	Node Addr. / Dest. Key / PID Mask register
0x801000A8	MSE10	**0x8100B000**	**SpaceWire 1 registers:**
0x801000B8	MSE11	0x8100B000	Configuration register
0x801000C8	MSE12	0x8100B004	Control register
0x801000D8	MSE13	0x8100B008	Status register
0x801000E8	MSE14	0x8100B00C	Node Address / Dest. Key / PID register
0x801000F8	MSE15	0x8100B010	Max TX Packet Size register

Address	Register	Address	Register
0x80300000	**Interrupt controller registers:**	0x8100B014	Max RX Packet Size register
0x80300000	PENDACK register	0x8100B018	Time Out register
0x80300004	SWIRQ register	0x8100B01C	Time Code register
0x80300008	LVLSENS register	0x8100B020	IT Pending / Force register
0x8030000C	POSTPONE register	0x8100B024	IT Clear register
0x80300010	Interrupt Enable register	0x8100B028	IT Mask register
0x80400000	**System Timer 0 registers:**	0x8100B02C	TX Segments Descriptor Address register
0x80400000	Control register	0x8100B030	RX Packets Descriptor Address register
0x80400004	Reload register	0x8100B034	RX Packets Descriptor End Addr. register
0x80400008	Counter register	0x8100B038	Start Address 1 register
0x8040000C	Timeout register	0x8100B03C	Start Address 2 register
0x80500000	**System Timer 1 registers:**	0x8100B040	RX areas size register
0x80500000	Control register	0x8100B044	Current Buffer End register
0x80500004	Reload register	0x8100B048	AHB Burst disable register
0x80500008	Counter register	0x8100B04C	Node Addr. / Dest. Key / PID Mask register
0x8050000C	Timeout register	**0x8100C000**	**SPI 0 registers:**
0x80600000	**User Timer 0 registers:**	0x8100C000	Data register
0x80600000	Control register	0x8100C001	Control register 1
0x80600004	Reload register	0x8100C002	Control register 2
0x80600008	Counter register	0x8100C003	Control register 3
0x8060000C	Timeout register	0x8100C004	Status register
0x80700000	**User Timer 1 registers:**	0x8100C005	Baud Rate register
0x80700000	Control register	**0x8100D000**	**SPI 1 registers:**
0x80700004	Reload register	0x8100D000	Data register
0x80700008	Counter register	0x8100D001	Control register 1
0x8070000C	Timeout register	0x8100D002	Control register 2
0x80800000	**User Timer 2 registers:**	0x8100D003	Control register 3
0x80800000	Control register	0x8100D004	Status register
0x80800004	Reload register	0x8100D005	Baud Rate register
0x80800008	Counter register	**0x8100E000**	**I2C 0 registers:**
0x8080000C	Timeout register	0x8100E000	Data register
0x80900000	**User Timer 3 registers:**	0x8100E001	Control register
0x80900000	Control register	0x8100E002	Address register
0x80900004	Reload register	0x8100E003	Future use
0x80900008	Counter register	0x8100E004	Status register
0x8090000C	Timeout register	0x8100E005	Baud Rate register
0x81000000	**Debug Unit registers:**		
0x81000000	Control register		
0x81000200 - 0x8100027C	Bank 0 GPR 0-31 registers		
0x81000280 - 0x810002FC	Bank 1 GPR 0-31 registers		

Address	Register	Address	Register
0x81000300 – 0x8100036C	SPC 0-27 registers	**0x8100F000**	**I2C 1 registers:**
0x81001000	**Memory Controller registers:**	0x8100F000	Data register
0x81001000	Memory Bank 0 register	0x8100F001	Control register
0x81001004	Memory Bank 1 register	0x8100F002	Address register
0x81001008	Memory Bank 2 register	0x8100F003	Future use
0x8100100C	Memory Bank 3 register	0x8100F004	Status register
0x81001010	Memory Bank 4 register	0x8100F005	Baud Rate register
0x81001014	Memory Bank 5 register	**0x81010000**	**CAN 0 registers:**
0x81001018	Memory Bank 6 register	0x81010000	MR register
0x8100101C	Memory Bank 7 register	0x81010001	CMR register
0x81001020	Memory condition register	0x81010002	SR register
0x81002000	**GPIO registers:**	0x81010003	ISR/IACK register
0x81002000	Data I/O register	0x81010004	IMR register
0x81002008	Set mask register	0x81010005	RMC register
0x81002010	Clear mask register	0x81010006	BTR 0 register
0x81002018	Direction register	0x81010007	BTR 1 register
0x81002020	2nd function register	0x81010008	TXBUF register
0x81003000	**Watchdog register**	0x8101000C	RXBUF register
0x81004000	**ADC registers:**	0x81010010	ACR register
0x81004000	Data register	0x81010014	AMR register
0x81004004	Control register	0x81010018	ECC register
0x81004008	Timer register	0x81010019	RXERR register
0x8100400C	Status register	0x8101001A	TXERR register
0x81004010	NRCHNK30 register	0x8101001B	ALC register
0x81004014	NRCHNK74 register	**0x81011000**	**CAN 1 registers:**
0x81004020	STADR0 register	0x81011000	MR register
0x81004024	STADR1 register	0x81011001	CMR register
0x81004028	STADR2 register	0x81011002	SR register
0x8100402C	STADR3 register	0x81011003	ISR/IACK register
0x81004030	STADR4 register	0x81011004	IMR register
0x81004034	STADR5 register	0x81011005	RMC register
0x81004038	STADR6 register	0x81011006	BTR 0 register
0x8100403C	STADR7 register	0x81011007	BTR 1 register
0x81005000	**PWM registers:**	0x81011008	TXBUF register
0x81005000	START/STATUS 8-bit register	0x8101100C	RXBUF register
0x81005001	CMPL 8-bit register	0x81011010	ACR register
0x81005002	STOP 8-bit register	0x81011014	AMR register
0x81005003	ISTOP 8-bit register	0x81011018	ECC register
0x810050N4	PRESCLHSC 16-bit register N = 0..7 (channel number)	0x81011019	RXERR register
0x810050N6	Timer A 16-bit register N = 0..7 (channel number)	0x8101101A	TXERR register

Address	Register	Address	Register
0x810050N8	Timer B 16-bit register N = 0..7 (channel number)	0x8101101B	ALC register
0x810050NA	Timer C 16-bit register N = 0..7 (channel number)	**0x81012000**	**DAC 0 registers:**
		0x81012000	Data I/O register
0x81006000	**UART 0 registers (Debug unit):**	0x81012002	Control register
0x81006000	Data I/O register	0x81012004	Timer register
0x81006008	Status register	**0x81013000**	**DAC 1 registers:**
0x81006010	Control register	0x81013000	Data I/O register
0x81006018	Scaler reload register	0x81013002	Control register
0x81006020	Start address register	0x81013004	Timer register
0x81006028	Command register	**0x81014000**	**Pulse counter 0 registers:**
0x81006030	Current address register	0x81014000	Pulse counter register
0x81006038	Debug status register	0x81014004	Pulse counter control register
0x81007000	**UART 1 registers:**	**0x81015000**	**Pulse counter 1 registers:**
0x81007000	Data I/O register	0x81015000	Pulse counter register
0x81007002	Status register	0x81015004	Pulse counter control register
0x81007004	Control register	**0x81016000**	**Pulse counter 2 registers:**
0x81007006	Scaler reload register	0x81016000	Pulse counter register
0x81008000	**UART 2 registers:**	0x81016004	Pulse counter control register
0x81008000	Data I/O register	**0x81017000**	**Pulse counter 3 registers:**
0x81008002	Status register	0x81017000	Pulse counter register
0x81008004	Control register	0x81017004	Pulse counter control register
0x81008006	Scaler reload register	**0x81018000**	**MIL-STD 1553 registers:**
0x81009000	**UART 3 registers:**	0x81018000	Command word FIFO
0x81009000	Data I/O register	0x81018002	Data word FIFO
0x81009002	Status register	0x81018004	MIL-STD 1553 Mode of operation
0x81009004	Control register	0x81018006	Core configuration register
0x81009006	Scaler reload register	0x81018008	MIL bus configuration register
		0x8101800E	MIL 1553 status word
		0x81018010	MIL parity error counter
		0x81018012	MIL Manchester error counter
		0x81018016	FIFO filling levels
		0x8101801C	Core interrupt mask
		0x8101801E	Core status register

Table 12: On-chip Register Address Assignments

6. System Bus Components

6.1. Implementation Register Module (IREG)

The implementation register module contains the following registers:

Register address	Register name
0x80000000	IREG version register
0x80000004	IREG global configuration register
0x80000008	IREG on-chip SRAM control register
0x8000000C	IREG on-chip SRAM status register
0x80000010	IREG PWM pre-scaler register
0x80000014	IREG SPW0 pre-scaler register
0x80000018	IREG SPW1 pre-scaler register

Table 13: IREG registers

6.1.1. Version register

The IREG version register is a read-only 32-bit register with fixed content to the binary equivalent of the number 2021 (0x7E5). It shows the implementation version of the MORAL microcontroller.

Bit	31:0
Name	VERSION
R/W	ro
Reset	0x000007E5

Figure 12: IREG version register (Address Offset: 0x00)

6.1.2. Global configuration register

The IREG global configuration register is a 32-bit register to control specific implementation-dependent features of the MORAL microcontroller. Bits [23:0] are read-write, bits [31:24] read-only.

Bit	31:24	23:0
Name	GBLcfgIn	GBLcfgOut
R/W	ro	rw
Reset	0x00	0x000000

Figure 13: IREG global configuration register (Address Offset: 0x04)

In the current MORAL implementation the bits are used as given in Table 14:

Name	R / W	Description
bit [0]	rw	reserved for swapping cs[0] and cs[1] pads (same as port "swap_cs0_cs1")
bit [1]	rw	reserved (on FPGA: enable EDAC for ext. flash memory)
bit [3:2]	rw	Mode for programming ext. flash at memory bank 0, see chapter 3.3.3
bit [4]	rw	'SEL' port of DAC[0]: '0': DAC output unbuffered on pin daco[0] '1': DAC output buffered on pin dacb[0]
bit [5]	rw	'SEL' port of DAC[1]: '0': DAC output unbuffered on pin daco[1] '1': DAC output buffered on pin dacb[1]

bit [6]	rw	not used
bit [7]	rw	DIFF_SEL port of ADC: '0': 8 single ended inputs, '1': 4 differential inputs The channel number is selected in bits [11:9] of ADC control register 0x81004004.
bit [23:8]	rw	not used
bit [24]	ro	Logic level at *peakrad* ASIC's flash_status input pin (for flash programming)
bit [31:25]	ro	not used

Table 14: IREG global configuration register usage in the "*peakrad*" chip

Application hints the current MORAL "*peakrad"* implementation:

Bits [7:4] are related to ADC and DAC functions / configurations that are not accommodated in the ADC and DAC registers at 0x81004000 and 0x81012000/0x81013000, respectively.

Bits [3:0] are related to special features of the memory interface at banks 0 and 1 (intended for flash and external RAM), in particular needed for flash programming. They are also related to the ports "swap_cs0_cs1" and "cs0_edac_cfi" of the chip.

Setting **port "swap_cs0_cs1"** to logic 'high' swap the memory banks 0 and 1. Bank 0 shall normally be attached to ROM / flash, since the "peaktop" processor starts program execution from address 0 of bank 0. For test purposes (in particular when the flash is not yet programmed or not functional), one can map RAM to the address range near 0 of bank 0, download a program via DebugUART to this memory area and start its execution by releasing "hold" and applying a "reset" pulse.
Bit [0] in the "global configuration register" is reserved for the same purpose, but not implemented.

Programming a CFI compliant flash, in particular its EDAC data, requires special care and effort, since the EDAC data words are normally generated by the Hsiao encoder, but for a CFI compliant flash also programming command words are required that cannot be predictably produced by the Hsiao encoder.

6.1.3. On-chip SRAM control and status registers

The on-chip SRAM control register is primarily used to control the EDAC operation of the on-chip SRAM (see Subsection 6.2). Another function is to control whether SRAM aliasing errors should be signaled or not. Aliasing occurs for example when a doubleword (64-bit wide) is written at the last address of the on-chip SRAM. In this case, the first half of 32 bits will be written at the last address, but the second half of 32 bits will be written at the starting address of the on-chip SRAM.

Bit	31:5	4	3	2	1	0
Name	reserved	EALIAS	CORRECT	DETECT2	DETECT1	ENCODE
R/W	undefined	rw	rw	rw	rw	rw
Reset		1	1	1	1	1

Figure 14: IREG on-chip SRAM control register (Address Offset: 0x04)

Name	Description
ENCODE	Calculate syndrome (0-syndrome zero; 1-calculated syndrome)
DETECT1	Detect single-bit errors in on-chip SRAM (0-do not detect; 1-detect)
DETECT2	Detect double-bit errors in on-chip SRAM (0-do not detect; 1-detect)
CORRECT	Correct single-bit errors in on-chip SRAM (0-do not correct; 1-correct)
EALIAS	Signal aliasing errors (0-do not signal; 1-signal)

Table 15: IREG on-chip SRAM control register description

The IREG on-chip SRAM status register reports the errors that occurred during SRAM operation, i.e., EDAC and aliasing errors.

Bit	31:16	15:8	7:3	2	1	0
Name	ERRCOUNT1	ERRCOUNT2	reserved	EDACERR2	EDACERR1	ALIASERR
R/W	rc	rc	undefined	rc	rc	rc
Reset	0	0		0	0	0

Figure 15: IREG on-chip SRAM status register (Address Offset: 0x0C)

Name	Description
ALIASERR	Aliasing error occurred (0-did not occur; 1-occured)
EDACERR1	Single-bit EDAC error occurred (0-did not occur; 1-occured)
EDACERR2	Double-bit EDAC error occurred (0-did not occur; 1-occured)
ERRCOUNT2	Number of detected double-bit EDAC errors since last read access (saturates at 255)
ERRCOUNT1	Number of detected single-bit EDAC errors since last read access (saturates at 65535)

Table 16: IREG on-chip SRAM status register description

The IREG on-chip SRAM status register has read-clear behaviour. It is automatically cleared to zero after reading the contents of this register.

6.1.4. Peripheral clock pre-scalers

The PWM (Subsection 7.9) and the two SpaceWire components (Subsection 6.6) are clocked through prescaler registers. There are three equivalent prescaler registers for each of the PWM and the SpaceWire components. The address offset for the PWM prescaler is 0x10, while for the two SpaceWire modules the address offsets are 0x14 and 0x18, respectively. Figure 16 and Table 17 describe these registers.

Bit	31:16	15:4	3:2	1	0
Name	reserved	PRESCL	reserved	SEL	EN
R/W	undefined	rw	undefined	rw	rw
Reset		0x0		0	0

Figure 16: PWM and SpaceWire prescaler registers

Name	Description
EN	Enable clock (0-disable; 1-enable)
SEL	Select clock source (0-CLK; 1-PCLK)
PRESCL	12-bit Prescaler reload value

Table 17: PWM and SpaceWire prescaler register Description

If the EN bit is 0, the clock is switched off, i.e., the corresponding component is clock-gated. The SEL bit selects one of the two clocks: either the 50 MHz CLK, or the 200 MHz PCLK (see Subsection 4.2.1). If the PRESCL value is zero, the source frequency is propagated, otherwise the source frequency is prescaled. For example, if PRESCL = 1 then the frequency will be either 25 MHz or 100 MHz (depending on the SEL value), if PRESCL = 2 then the frequency will be 16.67 MHz or 66.67 MHz, for PRESCL = 3 it will be 12.5 MHz or 50 MHz, etc.

6.2. On-chip Static Random Access Memory (On-chip SRAM)

A 32-bit wide on-chip SRAM is mapped in the range 0x90000000 – 0x9007FFFC (512KB). It is intended to reduce the traffic outside the chip improving performance and compactness. Furthermore, the workload of the microcontroller can come from a peripheral device (not necessarily from external memory), and data can be stored in the on-chip SRAM.

6.2.1. On-chip SRAM Error Detection and Correction (EDAC)

Hsiao EDAC is implemented in order to improve the fault tolerance of the on-chip SRAM. The Hsiao code is able to detect a double-error and to detect and correct a single-error. With this code, a 32-bit wide data word is protected by additional 7-bit wide syndrome. Detecting a single- or double-error raises a SYSTEM BUS ERROR exception. A single-error is correctable, while a double-error is only detectable and not correctable. The single-error exception handler may be used to re-write the corrected value back into the memory by software. A double-error exception must be urgently processed by software because it may affect the system behavior or other critical data.

The EDAC operation, i.e., syndrome generation, detection and correction is done by the hardware and is totally transparent to the user/software. That is, although internally 39-bit wide, the user sees a 32-bit wide memory. The on-chip SRAM EDAC is controlled by the on-chip SRAM control register, while the status is reported by the on-chip SRAM status register (see Subsection 6.1.3).

The MORAL controller has the same type of EDAC for the external memory interface in the memory controller (see Subsection 6.3.2).

6.3. External Memory and IO Access

Access to external memory and IO is provided by Memory Controller. The PEAKTOP core does not distinguish between memory and IO addresses, i.e., it is memory-mapped. Usually, the IO is connected in the higher part of the IO address space, i.e., with the MSB of the address equal to logic ‘1’. When assigning the address space, it should be also taken into account that the booting starts at address zero.

6.3.1. Memory Controller (MCTRL)

The memory controller is used for accessing external memories (PROM, EEPROM, and SRAM) and IO devices. It controls the memory interface of the MORAL microcontroller consisting of the signals:

Name	Direction	Pin count	Active level	Description
DATA	in/out	8	n/a	Data bus
EDAC	in/out	5	n/a	EDAC syndrome bus
ADDR	out	29	n/a	Address bus
CS	out	8	0	Chip Select, for each memory/IO banks
WREN	out	1	0	Write enable
OEN	out	1	0	Output enable
READ	out	1	1	Read cycle
EDACEN	in	1	1	Memory controller EDAC enable
BEXCn	in	1	0	Bus exception
BRDYn	in	1	0	Bus ready

Table 18: Memory Controller Pins Descriptions

The eight memory/IO banks are equally distributed in the memory address space:

Start Address	End Address	Chip Select (CS)	Description
0x00000000	0x1FFFFFFF	0	Memory/IO bank 0
0x20000000	0x3FFFFFFF	1	Memory/IO bank 1
0x40000000	0x5FFFFFFF	2	Memory/IO bank 2
0x60000000	0x7FFFFFFF	3	Memory/IO bank 3
0x80000000	0x9FFFFFFF	4	Memory/IO bank 4
0xA0000000	0xBFFFFFFF	5	Memory/IO bank 5
0xC0000000	0xDFFFFFFF	6	Memory/IO bank 6
0xE0000000	0xFFFFFFFF	7	Memory/IO bank 7

Table 19: Memory/IO Banks

Each bank is activated by the corresponding chip select signal CS. The system designer decides which of the banks will be configured as memory or IO banks by setting different parameters like number of read/write cycles.

The on-chip peripherals and on-chip SRAM are mapped in bank 4.

For each of the banks there is an 18-bit register at addresses 0x00 – 0x07, correspondingly which are defined in following figure and table.

Base address of MCTRL registers: 0x8100 1000.

Bit	31:29	28:24	23:21	20:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved	RWS	reserved	WWS	CORREC	DETECT2	DETECT1	ENCODE	LEADOUT	RDLEADI	WRLEADI	BRDYEN	BEXCEN	reserved	LITEND	ALIGNED	SYS	READ	WRITE	EXEC
R/W	ro, rw(s)																			
Reset	0x0	0x1F	0x0	0x1F	1	1	1	1	1	1	1	0	0	0	1	0	0	1	1	1

Figure 17: Memory/IO Bank Registers 0 – 7 (*Address Offsets*: 0x00 – 0x1C)

Bit	Name	Description
0	EXEC	Execute access (0-cannot fetch instructions from this bank; 1-fetch ok)
1	WRITE	Write access (0-cannot write in this bank; 1-write access ok)
2	READ	Read access (0-cannot read in this bank; 1-read access ok)
3	SYS	System access (0-allowed access in both user and system mode; 1-access allowed only in system mode)
4	ALIGNED	Aligned memory access (0-unaligned access allowed; 1-unaligned access not allowed)
5	LITEND	Endianness (0-big endian; 1-little endian access)
6	reserved	Reserved bit, no function
7	BEXCEN	Bus exception enable (0-the bexcn input signal will not be evaluated during access; 1-the bexcn input signal will be evaluated during access)
8	BRDYEN	Bus ready enable (0-the brdyn input signal will not be evaluated during access; 1-the brdyn input signal will be evaluated during access)

Bit	Name	Description
9	WRLEADIN	Lead-in cycle for write accesses (0-no lead-in; 1-lead-in)
10	RDLEADIN	Lead-in cycle for read accesses (0-no lead-in; 1-lead-in)
11	LEADOUT	Lead-out cycle on (non-consecutive) read accesses (e.g., due to slow switch to Hi-Z state on the bus) (0-no lead-out; 1-lead-out)
12	ENCODE	Calculate syndrome (0-syndrome zero; 1-calculated syndrome)
13	DETECT1	Detect single-bit errors in external memory/IO (0-do not detect; 1-detect)
14	DETECT2	Detect double-bit errors in external memory/IO (0-do not detect; 1-detect)
15	CORRECT	Correct single-bit errors in external memory/IO (0-do not correct; 1-correct)
20:16	WWS	Number of write wait states (0,1,2, …, 31)
23:21	reserved	Reserved field, no function
28:24	RWS	Number of read wait states (0,1,2, …, 31)
31:29	reserved	Reserved field, no function

Table 20: Memory/IO Bank Registers 0 – 7 Descriptions

Finally, register 8 at address 0x08 is an error condition register which indicates whether an error occurred during a read/write access. This register is read-only in user and system mode. However, it is reset to zero only by reading it in system mode.

Bit	31:7	6	5	4	3	2	1	0
Name	reserved	OVERRUN	BURSTREQ	UNALIGNED	ACCVIOL	EDACERR2	EDACERR1	BUSEXC
R/W	undefined	ro						
Reset		0x0						

Figure 18: Error Condition Register (*Address Offset*: 0x20)

Bit	Name	Description
0	BUSEXC	Bus exception signaled by the bexcn input line (0-did not occur; 1-occured)
1	EDACERR1	Single-bit EDAC error occurred (0-did not occur; 1-occured)
2	EDACERR2	Double-bit EDAC error occurred (0-did not occur; 1-occured)
3	ACCVIOL	Access violation, affected by exec, write, read and sys (0-did not occur; 1-occured)
4	UNALIGNED	Unaligned access; if aligned access is configured in the corresponding control register (0-7), an error will be signaled on unaligned access (0-did not occur; 1-occured)
5	BURSTREQ	Not applicable/required in the MORAL microcontroller
6	OVERRUN	Not applicable/required in the MORAL microcontroller

Table 21: Error Condition Register Descriptions

The memory controller is connected to the AHB system bus. Thus, an error will set the corresponding error response on the bus which will trigger the “I/D SYSTEM BUS ERROR” exceptions depending on whether instruction or data is accessed (see page 42 in [01]). If an error condition occurdd, no access will take place on the external memory/IO banks.

6.3.2. Memory controller EDAC

The same Hsiao EDAC is used as already described for the on-chip SRAM in Subsection 6.2.1. However, the width of the data bus is here 8 bits (instead of 32) and the width of the syndrome is 5 bits (instead of 7). Here, for each of the memory banks the access can be controlled through the ENCODE, DETECT1, DETECT2 and CORRECT bits of the corresponding Memory/IO Bank Register (see Figure 17 and Table 20). Detected single- and double-bit errors raise the "SYSTEM BUS ERROR" exceptions and can be inspected by reading the Error Condition Register (see Figure 18 and Table 21).

In order for the memory controller EDAC to be active, the external input pin EDACEN should be asserted to 1. Furthermore, the ENCODE, DETECT1, DETECT2 and CORRECT bits have to be set according to the desired operation of the system. The EDACEN external input pin does not affect the on-chip SRAM EDAC but only the memory controller EDAC.

Besides the data bus, the EDAC syndrome is also read/written on 5 external input/output pins (the EDAC syndrome bus). The system designer has to either set a separate memory module (e.g., 8-bit wide), or to combine the EDAC syndrome bus with the data bus and set a single memory module (e.g., 16-bit wide) in which the EDAC syndrome bus will be treated as a higher or lower part of the data bus.

6.3.3. Programming external Flash Memory

Since the *peaktop* processor starts operation after reset at address 0 in memory bank 0, it is normally required that some non-volatile memory, ROM or flash, is attached to bank 0 of the memory controller. Such memory is read in the normal way, but may require special procedures for writing / programming; in particular for CFI-compliant flash when used as EDAC memory. These peculiarities will be discussed in this chapter.

Since the data interface is 8 bit wide and the EDAC interface 5 bit, there are three favorite options for external flash.

Flash Mode	Description
00 bin.	One 8 bit flash for data only, EDAC bits are not connected, EDAC is not usable
01 bin.	One 8 bit flash for EDAC + one 8 bit flash data, EDAC is useable
(10 bin.)	(Special mode when programming EDAC flash, not useable during normal operation)
11 bin.	One 16 bit flash for EDAC (in bits[15:8]) + data (in bits [7:0]), EDAC is useable

Table 22: Flash modes for external flash

In any case, the EDAC interface width will be extended from 5 bit to 8 bit, where the upper bits [7:5] are not used (usually on logic high level).

The flash mode can be set in the "Global Configuration Register" at address 0x8000 0004, see chapter 3.1.2. Bits [3:2] define the "Flash Mode" (default on 00: only 8 bit data flash, no EDAC).

In case of flash mode 01 (two flash chips for EDAC and data, respectively), the ChipEnable of the data flash chip shall be connected to *peakrad* pin cs[0], whereas the ChipEnable of the EDAC flash chip shall be connected to *peakrad* pin cs0_edac_cfi. In flash modes 00 and 11, pin cs0_edac_cfi remains unconnected. All other pins like wren, oen and the data and edac buses are connected in parallel to all relevant chips.

For normal peakrad system operation, i.e. when only reading from the flash, the "FlashMode" bits in the "Global Configuration Register" shall be set according to the real configuration of flash memory chips attached to the memory controller. For flash mode 00, the “edacen” pin (EDAC enable) shall be on logic low, disabling EDAC checks. For flash modes 01 and 11, it should be on logic high. Setting flash mode should be done by software shortly after boot, e.g. when configuring the wait states in the memory controller appropriately.

The procedures for flash programming strongly depend on the kind of flash (e.g. CFI compliant or not) and the flash mode. At least the first programming after system fabrication will be done via the debug interface, e.g. DebugUART. In case that the flash can be programmed / written with a simple write command, no special processing is required. It can be done like writing to RAM (in any of the flash modes), except that one possibly has to wait some time (usually in the milliseconds range) after each byte has been written. Such a procedure applies, e.g. for the Maxwell *Maxwell* 28LV010 chip.

For CFI compliant chips (e.g. *Aeroflex* UT8QNF8M8), however, special programming sequences are required, where all programming bytes must be transferred via DebugUART. There are such sequences for flash programming, for erasing the flash, for reading CFI configuration data and other purposes (chip lock, device ID, etc.)

Flash programming requires sending the (chip dependent) programming sequence (e.g. one data byte 0x40 for *Intel Strata* flash or sequence 0xAA, 0x55, 0xA0 for *Aeroflex* UT8QNF8M8), followed by the address + data byte to be programmed. When all bytes have been programmed, usually a special command is required (data byte 0xFF for *Intel Strata* flash, byte 0xF0 for *Aeroflex* UT8QNF8M8) to return to the normal flash “read-array mode”.

Normally, all these bytes must be sent in separate DebugUART strings (17 bytes each), providing the appropriate address and the command or data byte. This may be very time consuming. In the following we give some hints how to proceed.

FlashMode 00 (only 8 bit data flash, no EDAC flash)

In this mode, first the FlashMode bits in the “Global Configuration Register” shall be set to 00 (write 32-bit word at address 0x8000 0004 via DebugUART). Then, the flash program bytes shall be transferred as sketched above via DebugUART. Finally, the command for returning to flash “read-array mode” must be issued. The FlashMode in the “Global Configuration Register” remains at 00.

FlashMode 11 (16 bit flash for EDAC + data bits, EDAC parity in bits[15:8], data in bits [7:0])

The FlashMode in the “Global Configuration Register” during programming shall be set to 11 and “edacen” pin shall be set to logic high in order to properly generate the Hsiao parity bits in the *peakrad* hardware. Then, the flash program bytes shall be transferred as sketched above via DebugUART. Finally, the command for returning to flash “read-array mode” must be issued.

It may be that the flash status word, that indicates whether a programming or erasure command has finished, cannot be properly read via the DebugUART, since the status word does not comply with the Hsia coding and may be incorrectly decoded. In this case, the flash chip’s status pin shall be connected to the *peakrad*’s flash_status pin, that can be read as bit[31] of the “Global Configuration Register” at address 0x8000 0004 (read one byte from 0x8000 0007 and check for bit[7] of this byte).

FlashMode 01 (8 bit EDAC flash + 8 bit data flash in separate chips)

In this mode, both flash chips must be programmed separately. It is required that the ChipEnable pin of the EDAC flash chip is connected to the cs0_edac_cfi pin of the *peakrad* chip.

For programming the data flash, the FlashMode bits in the “Global Configuration Register” shall be set to 00 (write 32-bit word at address 0x8000 0004 via DebugUART). Then, the flash program bytes shall be transferred in the same way as described for FlashMode 00 via DebugUART. Finally, the command for returning to flash “read-array mode” must be issued.

For programming the EDAC flash, the FlashMode bits in the “Global Configuration Register” must be set to 10. Then, the flash content shall be transferred in the same way, but instead of each data byte, the respective Hsiao parity bits must be sent. They must be calculated by the download software. Bits[7:5] of the byte shall be set to 111 (this is the usual NOR flash content after erasure). Finally, the command for returning to flash “read-array mode” must be issued, and the FlashMode bits in the “Global Configuration Register” shall be set to 01 for normal operation. Pin edacen may be enabled (set to logic high).

Reading the CFI data

For reading CFI data from a CFI capable flash chip, first “CFI query mode” must be activated. This should be a unique command 0x98 for all kinds of chips, perhaps to be written to the specific address 0x555. It must be sent via DebugUART (17 bytes string). After that, the parameters can be read via DebugUART from the appropriate addresses (e.g. the characters ‘Q’, ‘R’, ‘Y’ from addresses 0x20, 0x22, 0x24). Finally, the command for returning to flash “read-array mode” must be issued (byte 0xFF for *Intel Strata* flash, byte 0xF0 for *Aeroflex* UT8QNF8M8).

The FlashMode bits in the “Global Configuration Register” shall be set to 00 when accessing the data flash. For reading CFI data from the EDAC flash, it shall be set to 10 bin.

As a fallback solution when flash programming fails, the *peakrad* chip provides an option to swap memory banks 0 and 1. Provided that bank 0 is flash/ROM and bank 1 is RAM, this allows program download to the RAM, which is then located around the boot address 0 and to start executing the (volatile) software from there. Program download to RAM should be possible without much problems via the debug interface, e.g. DebugUART, whereas flash programming might be more tricky.

Swapping is enabled by setting the *peakrad*’s input port “Swap_cs0_cs1” to logic high. It is only possible to swap banks 0 (supposed to be attached to flash) and bank 1 (supposed to be attached to RAM).

6.4. Interrupt Controller (ICTRL)

The interrupt controller uses 32-bit interrupt request (IRQ) lines (26 hard-wired) for connecting on-chip and off-chip peripherals. Off-chip peripherals with interrupt capabilities can be connected through 4 GPIO pins by enabling the second function of the corresponding GPIO pin. The interrupt priority is fully controlled by software and not dependent on the mapping of IRQ inputs to interrupt sources.

Table 23 shows all IRQ input mappings to available interrupt sources:

ICTRL Inputs	Interrupt Sources
0	System Timer 0
1	System Timer 1
2	User Timer 0
3	User Timer 1
4	User Timer 2
5	User Timer 3
6	ADC
7	UART 0
8	UART 1
9	UART 2
10	UART 3
11	SPW 0

ICTRL Inputs	Interrupt Sources
12	SPW 1
13	SPI 0
14	SPI 1
15	I2C 0
16	I2C 1
17	CAN 0
18	CAN 1
19	DAC 0
20	DAC 1
21	MIL-STD 1553
22	EXT IRQ 0
23	EXT IRQ 1
24	EXT IRQ 2
25	EXT IRQ 3

Table 23: Mapping of IRQ Sources to Peripherals

The IRQ lines 26 to 31 are not hard-wired. However, all IRQ lines 0 to 31 can be also triggered by software.

Registers can be accessed in byte, half-word or word mode. Specifying a mode greater than word sets the error line which invokes the “D SYSTEM BUS ERROR” exception. However, in a single access only one register can be read or written. For example, writing a word that spans two registers (e.g., by specifying word write at an address of the third byte of a register) will effectively write only the first register (i.e., third and fourth byte), while the second register will be unchanged.

The interrupt controller contains five 32-bit registers. Base address is: 0x8030 0000.

Bit	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
Name	ENABLE[31:0]																															
R/W	rw																															
Reset	0x0																															

Figure 19: Interrupt Enable Register (Address Offset: 0x10)

ENABLE register – each bit in this 32-bit register defines whether the corresponding IRQ line is enabled (1) or not enabled (0).

Bit	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
Name	LVLSENS[31:0]																															
R/W	rw																															
Reset	0x0																															

Figure 20: Interrupt LVLSENS Register (Address Offset: 0x08)

LVLSENS register – each bit in this 32-bit register defines whether the corresponding IRQ line is level-sensitive (1) or edge-sensitive (0).

Bit	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
Name	POSTPONE[31:0]																															
R/W	rw																															
Reset	0x0																															

Figure 21: Interrupt POSTPONE Register (Address Offset: 0x0C)

POSTPONE register – each bit in this 32-bit register defines whether the IRQ on the corresponding IRQ line should be postponed (1) or not (0) until the processor is not busy.

Bit	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
Name	PENDACK[31:0]																															
R/W	rw																															
Reset	0x0																															

Figure 22: Interrupt PENDACK Register (Address Offset: 0x00)

PENDACK register – this register signals whether the corresponding IRQ is pending (1) or not (0). Writing a 1 to a bit in this register acknowledges the IRQ to the interrupt controller and clears the corresponding bit in this register by setting it to 0. (However, IRQs may have to be acknowledged also at their source, i.e., in a peripheral register).

Bit	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
Name	SWIRQ[31:0]																															
R/W	rw																															
Reset	0x0																															

Figure 23: Interrupt SWIRQ Register (Address Offset: 0x04)

SWIRQ register – writing 1 to a bit in this register sets the corresponding IRQ as pending if the corresponding IRQ is set to be edge-sensitive in the LVLSENS register. This may be used for software activation of IRQs.

6.5. System Timer and User Timer (TIMER)

The MORAL microcontroller integrates six independent 32-bit timers connected to the AHB system bus. Two of them can be only accessed and operated only in system mode and are called system timers. The other four can be accessed both in system and user mode and are called user timers. The timers are decrementing counters running at the system clock frequency CLK. On underflow each timer generates an interrupt request signal (pulse interrupt). For each timer it is possible to select whether the timer stops on underflow or it restarts with a software-defined reload value. Figure 24 *shows* a block diagram of the described timer.

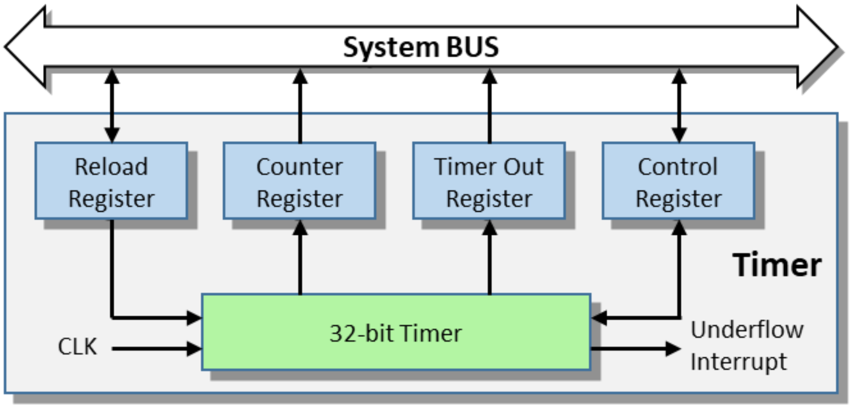


Figure 24: System/User Timer Block Diagram

Base address System timer 0:	0x8040 0000;
Base address System timer 1:	0x8050 0000;
Base address User timer 0:	0x8060 0000;
Base address User timer 1:	0x8070 0000;
Base address User timer 2:	0x8080 0000;
Base address User timer 3:	0x8090 0000;

Bit	31:5	4	3	2	1	0
Name	reserved	IRQEN	CONT	RTO	LOAD	RUN
R/W	undefined	rw				
Reset		0x0				

Figure 25: System/User Timer Control Register (Address Offset: 0x00)

Name	Description
RUN	Run timer (0-stop; 1-run)
LOAD	Load the value of the Reload register in the Timer Counter Register (LOAD is auto-reset to 0)
RTO	Reset Timer Out Register (RTO is auto-reset to 0)
CONT	Continuous operation; (0-stop on underflow; 1-reload and continue running on overflow)
IRQEN	IRQ enable; (0-do not signal IRQ on timer overflow; 1-signal IRQ on timer overflow)

Table 24: System/User Timer Control Register Descriptions

Bit	31:0
Name	Timer reload value
R/W	rw
Reset	0x0

Figure 26: System/User Timer Reload Register (Address Offset: 0x04)

Bit	31:0
Name	Timer value
R/W	ro
Reset	0x0

Figure 27: System/User Timer Counter Register (Address Offset: 0x08)

Bit	31:0
Name	Timer out value (number of underflows)
R/W	rw
Reset	0x0

Figure 28: System/User Timer Out Register (Address Offset: 0x04)

The timer reload register must be initialized before loading/reloading/running the timer. The timer counter register provides the actual counter values and is read-only. The Timer Out register shows the number of timer underflows. However, this register may also be written and can be reset to zero by writing the RTO bit in the Timer Control registers.

When a running counter is disabled and enabled again (by writing the RUN bit to 0 and then to 1), it holds the counter state and is not reset to zero.

6.6. SpaceWire Interface (SPW)

In the MORAL chip implemented is the ESA SPW-RMAP module, supporting the SpaceWire RMAP (remote memory access protocol) function.

Detailed specification see “SPW_RMAP_Specification_Architecture.pdf” in reference [03] provided by EADS Astrium which is a constitutive part of this specification document. The SPW functionality and all required interfaces, e.g. AHB- and APB-interface signals and registers offset addresses are specified in [03].

However, the burst operation of the AHB interface of the SPW is always disabled and cannot be enabled since the AHB bus of the MORAL microcontroller is not fully compliant to the AMBA 2.0 AHB bus. Thus, the Burst Disable Register [BURSTDISREG] (see Section 5 in [03]) is not functional – the burst operation is disabled no matter of the state of the BURST_DIS bit.

The SPW is connected to the system AHB and peripheral APB bus using the CLK clock. However, the transceiver is connected to the PCLK clock which may be pre-scaled (see Subsection 6.1.4).

The APB base addresses of the SpaceWire modules are:

SpaceWire 0: 0x8100 A000
SpaceWire 1: 0x8100 B000

All addresses of the registers are summarized in Table 12.

7. PERIPHERAL BUS DIGITAL COMPONENTS

7.1. Watchdog Timer (WDT)

The MORAL processor implements a 32-bit watchdog timer which can be activated by software. Once activated it is not possible to deactivate the watchdog by software. The only way to deactivate the watchdog timer is a system reset (hard-reset/power-up-reset).

The watchdog interval is configurable in software by setting the WDT register (connected to the peripheral bus). If this register is greater than zero it is decremented on each CLK cycle. The WDT is active as long as the WDT register is greater than zero. When the WDT register reaches zero, a system reset is activated:

- The entire MORAL microcontroller is reset
- The output RESETOUT signal is held active for 16 CLK cycles

The block diagram of the watchdog timer is shown in Figure 29 below.

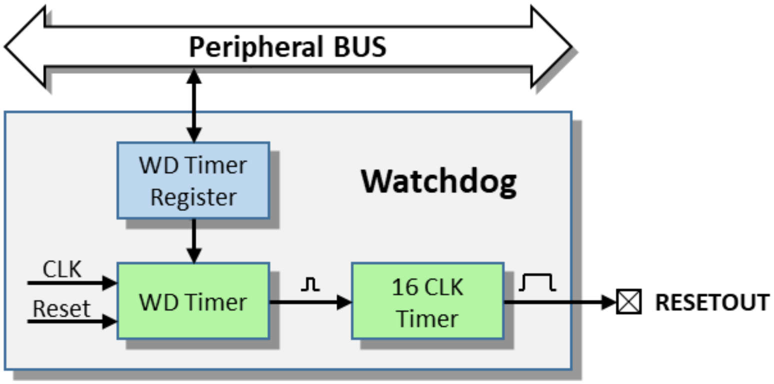


Figure 29: Watchdog Timer Block Diagram

Base address WDT: 0x81003000

Bit	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
Name	WD Timer																															
R/W	rw																															
Reset	0x0																															

Figure 30: Watchdog Timer Register (Address Offset: 0x00)

After external (hard) reset, the WDT register is set to zero and the WDT is inactive.

In many applications, the WDT register is cyclically reloaded with the corresponding value so that it never reaches zero and never generates a reset; however, due to exceptional situation the software routine may not be able to reload the value in the WDT in which case the WDT register reaches zero and a reset is generated.

7.2. General Purpose Input and Output (GPIO)

The MORAL microcontroller integrates a 64 GPIOs with bitwise programmability. The control and data registers are connected to the peripheral bus. The block diagram is presented in Figure 31.

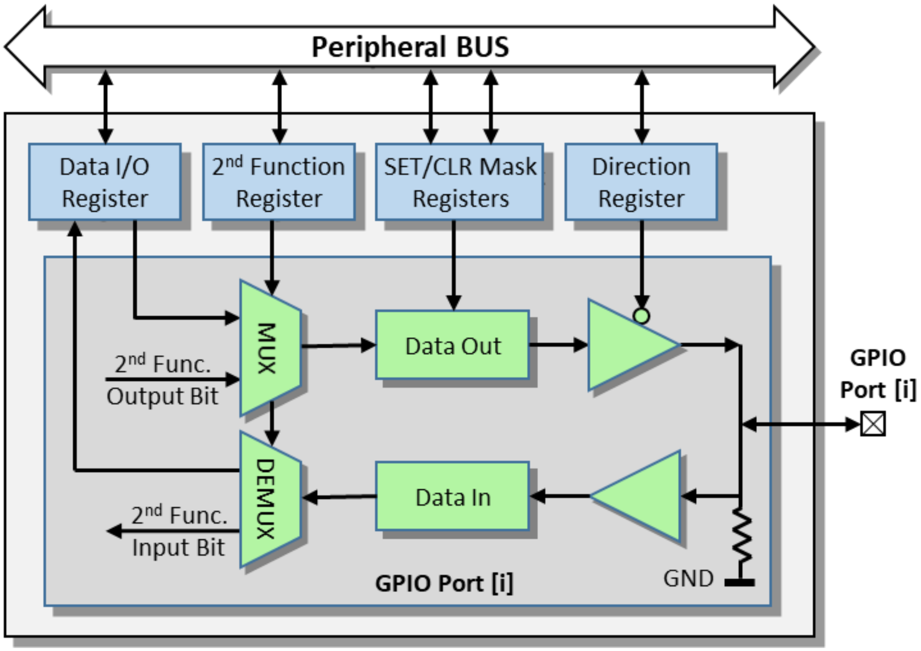


Figure 31: GPIO Block Diagram

Base address GPIO registers: 0x81002000

The block diagram shows how internal bi-directional peripheral devices are connected to GPIO ports. The double-buffered data input register is always readable via the APB bus.

All GPIO registers (DATA, DIR, SETIO, CLRIO, and FUNC) have the same bitwise programmable structure as shown in Figure 32. The GPIO registers contain 64 programmable I/O bits. After reset all GPIO ports are inputs in their basic function (i.e., both DIR and FUNC registers are 0x0).

Bits	63:0
Name	GPIO Register Bits
R/W	rw
Reset	0x0 (DATA register is undefined)

Figure 32: GPIO Registers

GPIO Data I/O Register:	Address Offset: 0x00
GPIO SET Mask Register:	Address Offset: 0x08
GPIO CLR Mask Register:	Address Offset: 0x10
GPIO Direction Register:	Address Offset: 0x18
GPIO 2nd Function Register:	Address Offset: 0x20

Name	Description
DATA	GPIO data register, used for reading/writing the GPIO ports configured as input/output ports.
SETIO/CLRIO	Set and clear bit masks can be used to avoid read-modify-write cycles. A “1” in the set mask will set the related GPIO data bit, whereas a “1” in the clear mask will reset this GPIO data bit to “0”.

Name	Description
DIR	Bit-wise configurable GPIO direction register. A logic ‘0’ defines input ports in which case the associated output drivers are disabled (tri-state). A logic ‘1’ defines output.
FUNC	If set to “1” the corresponding internal I/O data bit is connected to this GPIO port.

Table 25: GPIO Registers Descriptions

The 2nd (special) functions of the GPIO ports can be programmed by writing a logic “1” in the FUNC register for the following alternative I/O usage:

Port	2nd Function	Port	2nd Function	Port	2nd Function	Port	2nd Function
GPIO[0]	IRQ0	GPIO[16]	UART 1 - TXD	GPIO[32]	SPW 1 - DI	GPIO[48]	CAN 0 - TXD
GPIO[1]	IRQ1	GPIO[17]	UART 1 - RXD	GPIO[33]	SPW 1 - SI	GPIO[49]	CAN 0 - RXD
GPIO[2]	IRQ2	GPIO[18]	UART 1 - RTS	GPIO[34]	SPW 1 - DO	GPIO[50]	CAN 1 - TXD
GPIO[3]	IRQ3	GPIO[19]	UART 1 - CTS	GPIO[35]	SPW 1 - SO	GPIO[51]	CAN 1 - RXD
GPIO[4]	PWM 0	GPIO[20]	UART 2 - TXD	GPIO[36]	SPI 0 - SSN	GPIO[52]	PCNT 0
GPIO[5]	PWM 1	GPIO[21]	UART 2 - RXD	GPIO[37]	SPI 0 - SCK	GPIO[53]	PCNT 1
GPIO[6]	PWM 2	GPIO[22]	UART 2 - RTS	GPIO[38]	SPI 0 - MOSI	GPIO[54]	PCNT 2
GPIO[7]	PWM 3	GPIO[23]	UART 2 - CTS	GPIO[39]	SPI 0 - MISO	GPIO[54]	PCNT 3
GPIO[8]	PWM 4	GPIO[24]	UART 3 - TXD	GPIO[40]	SPI 1 - SSN	GPIO[56]	MIL_TXA[0]
GPIO[9]	PWM 5	GPIO[25]	UART 3 - RXD	GPIO[41]	SPI 1 - SCK	GPIO[57]	MIL_TXA[1]
GPIO[10]	PWM 6	GPIO[26]	UART 3 - RTS	GPIO[42]	SPI 1 - MOSI	GPIO[58]	MIL_RXA[0]
GPIO[11]	PWM 7	GPIO[27]	UART 3 - CTS	GPIO[43]	SPI 1 - MISO	GPIO[59]	MIL_RXA[1]
GPIO[12]	UART 0 - TXD	GPIO[28]	SPW 0 - DI	GPIO[44]	I2C 0 - SCL	GPIO[60]	MIL_TXB[0]
GPIO[13]	UART 0 - RXD	GPIO[29]	SPW 0 - SI	GPIO[45]	I2C 0 - SDA	GPIO[61]	MIL_TXB[1]
GPIO[14]	UART 0 - RTS	GPIO[30]	SPW 0 - DO	GPIO[46]	I2C 1 - SCL	GPIO[62]	MIL_RXB[0]
GPIO[15]	UART 0 - CTS	GPIO[31]	SPW 0 - SO	GPIO[47]	I2C 1 - SDA	GPIO[63]	MIL_RXB[1]

Table 26: GPIO 2nd Function Assignments

7.3. CAN Controller (CAN)

The CAN IP functionality is defined as provided by the DCAN user manual [06]. The CAN controller covers the control logic between the off-chip CAN bus transceivers and the APB bus. Two independent CAN controllers are integrated in the MORAL microcontroller. Bit rates of up to 1 Mbit/s are possible. The block diagram is presented in Figure 33.

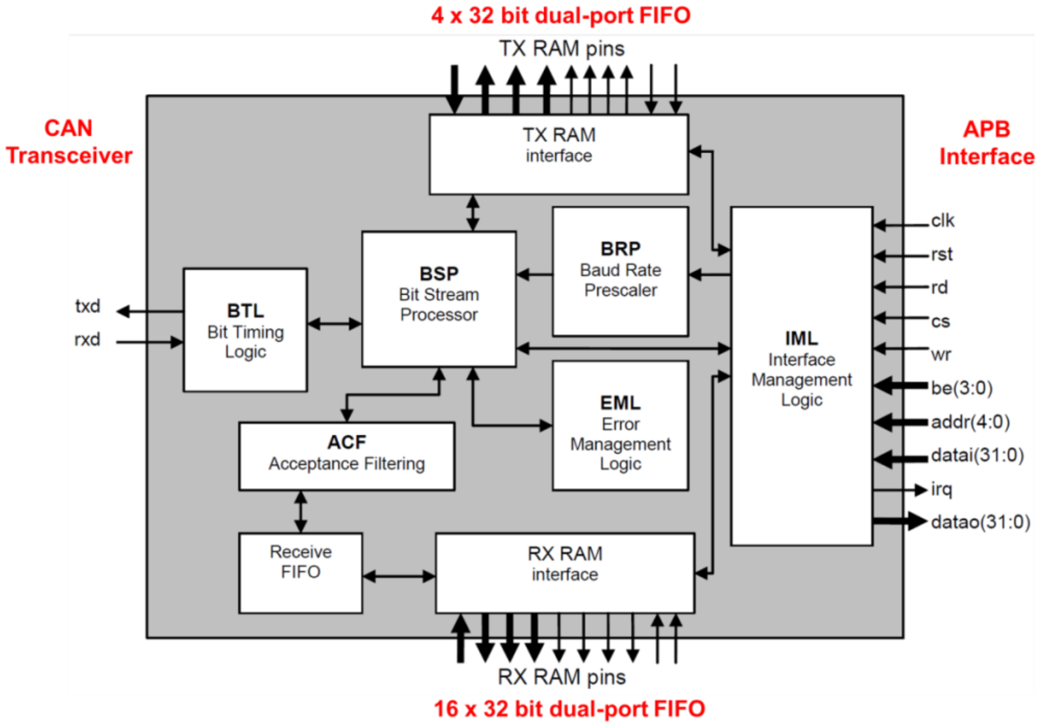


Figure 33: CAN Controller Block Diagram

Base address CAN 0: 0x8101 0000
Base address CAN 1: 0x8101 1000

The CAN Controller is compliant with the CAN Specification Revision 2.0 Part B. The CAN controller features a programmable bit rate to support applications that require a high-speed (up to 1 Mbit/s) or a low-speed CAN interface. A 64-byte receive FIFO and a 16-byte transmit buffer are implemented. The CAN bus serial transmit and receive interface must be connected to an external CAN transceiver. More detailed description of the DCAN functions is available in reference [06] which is a constitutive part of this specification document.

7.4. MIL-STD 1553C

The MIL-STD 1553 component in the MORAL processor implements an interface according to the standard MIL-STD 1553C (US Department of Defense 2018) that is frequently used in civil aircrafts, spacecrafts and military equipment. It is a “Digital time division command / response multiplex data bus”, capable of connecting up to 32 terminals to a bus controller over a two-wire differential serial bus (optionally with redundant cabling) at a raw data rate of 1 Mbit/s. The component can be configured to act as bus controller, remote terminal or bus monitor. It requires a commercial external transceiver chip (bus driver + level shifter) and bus transformer as well as a piece of software to run on the MORAL processor core.

The MIL-STD 1553C serial bus standard establishes requirements for digital, command/response, time division multiplexing (data bus) techniques. It encompasses the data bus line and its interface electronics, and also defines the concept of operation and information flow on the multiplex data bus and the electrical and functional formats to be employed. For a detailed specification see reference [09].

Base address MIL-STD 1553: 0x8101 8000.

Registers are 16 bit wide in a 5 bit address space (up to 16 registers at an address increment of 2, only even addresses can be used). We use in total eleven registers. Here, we describe only the most basic features here.

Bit	15:11	10	9:5	4:0
Name	Remote terminal address	T/R	Sub-address	Word count 1 - 32
R/W	rw			
Reset	undefined			

Figure 34: MIL-1553 Command Word Register (Address Offset: 0x00)

The command word register stores command words that shall be sent by the MIL1553 bus controller to the MIL bus (in transmit direction, i.e. when written from the processor). In receive direction, it stores received command words from the MIL bus in remote terminals and bus monitors or, respectively, received status words in the bus controller. In both directions, there is a FIFO with a capacity of 6 words. For the structure of a command word see the MIL-STD 1553 [09].

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Data															
R/W	rw															
Reset	undefined															

Figure 35: MIL-1553 Data Word Register (Address Offset: 0x02)

The data word register stores data words (including those related to mode commands) that are transmitted to or received from the MIL bus, respectively. In both directions, there is a FIFO with a capacity of 34 words.

Bit		7:6	5	4:0
Name	reserved	Mode	reserved	Remote terminal address
R/W	undefined	rw	undefined	rw
Reset		0x0		0x1F

Figure 36: MIL-1553 Operation Mode Register (Address Offset: 0x04)

The operation mode register is used to set the mode of operation of the MIL core (bus monitor, bus controller or remote terminal). For remote terminals, it also sets its address on the MIL bus.

Name	Description
Mode	**MIL core's mode of operation:** 00: bus monitor (only tracking bus transfers) 01: remote terminal 11: bus monitor (responding to mode commands) 10: bus controller
reserved	for future use.
Addr	**Remote Terminal's** (or bus monitor's in mode 11) **address** on the MIL bus.

Name	Description
	31 is the broadcast address and must not be used.

Table 27: MIL-1553 Operation Mode Register's Description

Bit	15	14	13	12	11:8	7:0
Name	MC31	MC0	reserved	FwdRT	reserved	Presc
R/W	rw	rw	undefined	rw	undefined	rw
Reset	1	1		0		0x32

Figure 37: MIL-1553 Core Configuration Register (Address Offset: 0x06)

The core configuration register defines some basic core features and options. In particular, bits [7:0] must contain the processors clock frequency in MHz in order to generate the correct bit rate on the MIL 1553 bus.

Name	Description
MC31	If high: Support "**Mode control**" operation for sub-address 31 if low: sub-address 31 is treated as normal sub-address for data transfer
MC0	if high: Support "**Mode control**" operation for sub-address 0 if low: sub-address 0 is treated as normal sub-address for data transfer
FwdRT	if high: Send all data words in **RT-to-RT transfer** also to the data FIFO in bus controller if low: Only broadcast data words in RT-to-RT transfers are sent to the data FIFO in bus controller
Presc	Setting of the **clock prescaler** to generate the **1 MHz** bit rate on the MIL bus. The unsigned 8 bit value shall give the processor clock frequency in MHz. It must be an even number larger than 8.

Table 28: MIL-1553 Core Configuration Register's Description

Bit	15	14	13	12	11:10	9	8	7	6:4	3:2	1	0
Name	RXenA	TXinhA	RXenB	TXinhB	reserved	SWo	SWi	Idle	reserved	CtrlMode	reserved	Chnl
R/W	rw	rw	rw	rw	undefined	rw	rw	rw	undefined	rw	undefined	rw
Reset	1	0	1	0		0	0	0		0x0		0

Figure 38: MIL-1553 Bus Configuration Register (Address Offset: 0x08)

The bus configuration register sets properties related to the attached MIL bus, in particular to the transceiver chip. Some functions are specifically designed for the HI-1585 bus transceiver chip from HOLT INTEGRATED CIRCUITS (or equivalent).

Name	Description
RXen A	Control signal Rxen (receiver enable) for channel A of the external transceiver chip
TXinh A	Control signal Txinh (transmitter inhibit) for channel A of the external transceiver chip
RXen B	Control signal Rxen (receiver enable) for channel B of the external transceiver chip

Name	Description
TXinh B	Control signal Txinh (transmitter inhibit) for channel B of the external transceiver chip
SWo	Swap the two lines of MILtx in case of wrong polarity
SWi	Swap the two lines of MILrx in case of wrong polarity
Idle	Logic level on both MILtx lines when transmitter is idle
CtrlMode	Mode how MIL transceiver control signals RXen and Txinh are controlled, see "Docu_MIL1553” Word document
Reserved	for future use
Chnl	When low: transmit and receive on MIL channel A When high: transmit and receive on MIL channel B

Table 29: MIL-1553 Bus Configuration Register’s Description

The MIL status register is relevant for remote terminals only. The MIL status word is defined in clause 4.3.3.5.3 of the MIL-STD 1553. It is regularly sent from the remote terminal to the bus controller. By setting certain bits, the terminal can indicate errors (normally handled by the core in hardware) or service requests etc. For this purpose, the software has to set certain bits like bit[0], bit[2] by writing to register 0x0E. For details, see Word document “Docu-MIL1553.doc“.

Bit	15:0
Name	Number of received MIL words having parity errors
R/W	rc
Reset	0x0

Figure 39: MIL-1553 Parity error counter (Address Offset: 0x10)

The register counts the command / status and data words received from the MIL bus that have parity errors. The counter saturates at 65535. It is cleared by writing any value to the register.

Bit	15:0
Name	Number of received MIL words having Manchester code errors
R/W	rc
Reset	0x0

Figure 40: MIL-1553 Manchester code error counter (Address Offset: 0x12)

The register counts the words received from the MIL bus that have Manchester coding errors. The counter saturates at 65535. It is cleared by writing any value to the register.

Bit	15	14	13:8	7	6	5:0
Name	TXDe	RXDe	RX data FIFO filling level 0 – 34 (=full)	TXCe	RXCe	RX cmd FIFO filling level 0 – 6 (=full)
R/W	ro	ro	ro	ro	ro	ro
Reset	0	0	0x0	0	0	0x0

Figure 41: MIL-1553 FIFO filling level Register (Address Offset: 0x16)

The FIFO filling level registers allows getting the number of words currently stored in both RX FIFO as well as indicators whether TX FIFOs are empty. The TX FIFOs can be cleared be setting the MIL core "Operation Mode" to "00" for a few clock cycles. In this bus monitor mode, all words from both TX FIFOs will read and discarded by the core.

Name	Description
TXDe	When high: TX data FIFO is empty
RXDe	When high: RX data FIFO is empty
RXDlvl	Number of 16-bit words in RX data FIFO, range 0 to 34 = full
TXCe	When high: TX command FIFO is empty
RXCe	When high: RX command/status FIFO is empty
RXClvl	Number of 16-bit words in RX command/status FIFO, range 0 to 6 = full

Table 30: MIL-1553 FIFO filling level Register’s Description

Bit	15	14	13	12	11	10	9	8	7	6:5	4	3	2	1	0
Name	RXCf	RXCe	TXCf	TXCe	RXDf	RXDe	TXDf	TXDe	Sync	reserved	eCmd	RXerr	Tout	nIdle	TXind
R/W	rw	rw	rw	rw	rw	rw	rw	rw	rw	undefined	rw	rw	rw	rw	rw
Reset	0	0	0	0	0	0	0	0	0		0	0	0	0	0

Figure 42: MIL-1553 Interrupt Mask Register (Address Offset: 0x1C)

Bit	15	14	13	12	11	10	9	8	7	6:5	4	3	2	1	0
Name	RXCf	RXCe	TXCf	TXCe	RXDf	RXDe	TXDf	TXDe	Sync	reserved	eCmd	RXerr	Tout	nIdle	TXind
R/W	ro	ro	ro	ro	ro	ro	ro	ro	ro	undefined	ro	ro	ro	ro	ro
Reset	0	0	0	0	0	0	0	0	0		0	0	0	0	0

Figure 43: MIL-1553 Processor Status Register (Address Offset: 0x1E)

The interrupt mask register 0x1C is a read-write register that allows enabling of interrupts to the processor at certain events which set a bit in the processor status register 0x1E. The bit / event assignment is identical for both.

Name	Description
RXCf	When high: RX command/status FIFO is **full**.
RXCe	When high: RX command/status FIFO is **not empty**.
TXCf	When high: TX command/status FIFO is **full**.
TXCe	When high: TX command/status FIFO is **not empty**.
RXDf	When high: RX data FIFO is **full**.
RDCe	When high: RX data FIFO is **not empty**.
TXDf	When high: TX data FIFO is **full**.
TXDe	When high: TX data FIFO is **not empty**.
Sync	A 'high' pulse indicates that mode command “Synchronize” (code 1 or 17) was received

Name	Description
reserved	for future use
eCmd	When high: **Erroneous command** received from processor.
RXerr	When high: **Error** in command or data word received from MIL bus.
Tout	When high: **Timeout** on MIL bus: addressed remote terminal does not respond.
nIdle	When high: Core is **not idle**.
TXind	When high: Core is **transmitting** on MIL bus.

Table 31: MIL-1553 Interrupt Mask and Processor Status Register's Description

7.5. Serial Peripheral Interface (SPI)

The SPI component complies with the quasi-standard as described for example in ref. [02] “SPI_Motorola.pdf”. Figure 44 shows how the SPI module is connected to the internal peripheral bus and the external SPI bus interface.

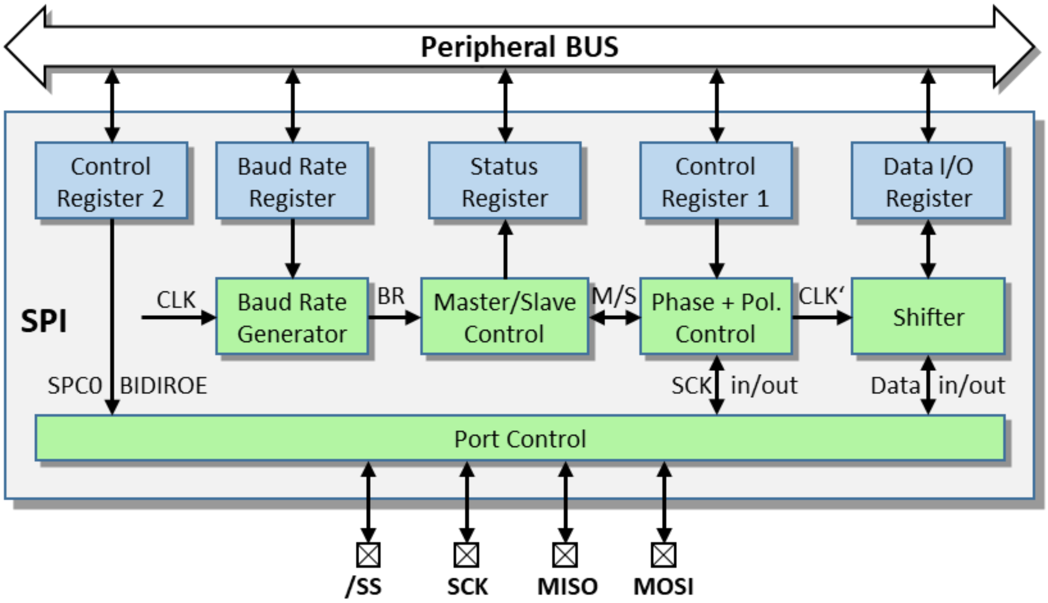


Figure 44: SPI Module Block Diagram

The MORAL processor chip contains two high configurable SPI instances having the base addresses:

Base address SPI 0: 0x8100 C000
Base address SPI 1: 0x8100 D000

The SPI registers are 8-bit wide. There are 5 of them as follows:

Bit	7:0
Name	SPI data to write or read
R/W	rw
Reset	not applicable

Figure 45: SPI Data register (Address Offset: 0x00)

The data register provides access to the TX and RX data FIFOs, each of 4 byte capacity. They shall be used in SPI master and slave modes to store the data to be transmitted or that have been received, respectively. The number of data bytes in the RX FIFO can be read from the status register 4.

Bit	7	6	5	4	3	2	1	0
Name	SPE	MSTR	SPTIE	SPRIE	CPOL	CPHA	SSPOL	LSBFE
R/W	rw	rw	rw	rw	rw	rw	rw	rw
Reset	0	0	0	0	0	0	0	0

Figure 46: SPI Control Register 1 (Address Offset: 0x01)

SPI Control Register 1 defines the basic configuration of the SPI core (off – master - slave, CPOL and CPHA modes, etc.). Any write access to this register cancels any pending SPI transfer and resets the SPI state machine, the status bits and clears both data FIFOs.

Name	Description
LSBFE	**LSB First Enable**: The bit defines whether the first bit of an SPI transfer is bit [7] (MSB, most significant bit, this is default) or bit [0] (LSB, least significant bit) of the data byte. 1 = Data is transferred least significant bit first. 0 = Data is transferred most significant bit first.
SSPOL	**Slave Select Polarity**: This bit defines whether the SPI SlaveSelect (ChipSelect) signal SS (CS) is low or high active. 1 = SS / CS is high active (low when idle). 0 = SS / CS is low active (high when idle)
CPHA	**SPI Clock Phase Bit**: This bit is used to select the SPI clock format. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. 1 = Sampling of data occurs at even edges (2,4,6,...,16) of the SCK clock 0 = Sampling of data occurs at odd edges (1,3,5,...,15) of the SCK clock
CPOL	**SPI Clock Polarity Bit**: This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. 1 = Active-low clocks selected. In idle state SCK is high. 0 = Active-high clocks selected. In idle state SCK is low.
SPRIE	**SPI Receive Interrupt Enable Bit**: This bit enables SPI interrupt requests at the end of each SPI transfer (when SS/CS becomes de-asserted) or when the RX FIFO becomes full. 1 = SPI receive interrupts enabled 0 = SPI receive interrupts disabled
SPTIE	**SPI Transmit Interrupt Enable Bit**: This bit enables SPI interrupt requests at the end of each SPI transfer (when SS/CS becomes de-asserted) or when the TX FIFO becomes empty. 1 = SPI transmit interrupts enabled 0 = SPI transmit interrupts disabled
MSTR	**SPI Master/Slave Mode Select Bit**: This bit selects, if the SPI operates in master or slave mode. 1 = SPI is in Master mode 0 = SPI is in Slave mode
SPE	**SPI System Enable Bit**: This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state, status bits in SPISR register are reset 1 = SPI enabled, port pins are dedicated to SPI functions. 0 = SPI disabled (lower power consumption).

Table 32: SPI Control Register 1 Descriptions

Bit	7:5	4:0
Name	reserved	Word length
R/W	ro	rw
Reset	0x0	0

Figure 47: SPI Control Register 2 (Address Offset: 0x02)

Using SPI Control Register 2 one can select whether SPI data shall be transferred as byte stream of variable length (if bit[4:0] = 0) or in individual fixed-size words having 1 to 31 bit, where the size is given in bit[4:0].

Name	Description
Reserved	future use
Word length	If 0: SPI data is transferred in bytes as long as TX FIFO is not empty when next byte starts If 1-31: SPI word has a fixed length (in bits)

Table 33: SPI Control Register 2 Descriptions

If this register is zero, an SPI transfer starts when a data byte is written into the master's TX FIFO. The data to be transferred at the slave (on MISO) must have been stored in its TX FIFO before. One can store up to 4 bytes in each FIFO. When the last byte is read from the TX FIFO for SPI transfer, an (optional) interrupt is generated that triggers the software to re-fill the FIFO with more data, if more bytes shall be transferred. The time available for refilling is the time required for transferring one byte on SPI, i.e. it depends on the selected SPI clock frequency. The SPI transfer stops (CS line becomes de-asserted) when the TX FIFO at the master is empty in that moment when the next byte of the stream would be requested. Also at the end of the SPI transfer, an interrupt may be generated (e.g. for reading the RX FIFO).

When the SPI Control Register 2 is non-zero, it defines the word length of the SPI transfer (range 1 to 31 bits). Each transfer will have this length, which must be set at the master as well as at the slave. An SPI transfer starts when the first data byte is written into the master's TX FIFO. The data to be transferred at the slave (on MISO) must have been stored in its TX FIFO before. Since the capacity of the FIFO is 4 bytes and the max. SPI word length is 31 bit, all TX bytes fit into the FIFO. The SPI transfer will stop when the required number of bytes has been read.

If the SPI word length is not a multiple of 8 bits, some bits from the FIFO will not be sent or received. For these padding bits, the following rules apply to the last data byte (first bytes are completely sent):

- If data are transferred *MSB first* (bit [0] of Control Register 1 = LSB First Enable = 0):
  At the TX FIFO (master + slave), the valid bits shall be in the higher bits [7:x], lower bits are ignored.
  At the RX FIFO (master + slave), the valid bits are in the lower bits [x:0], higher bits are pad bits.
- If data are transferred LSB first (bit [0] of Control Register 1 = LSB First Enable = 1):
  At the TX FIFO (master + slave), the valid bits shall be in the lower bits [x:0], higher bits are ignored.
  At the RX FIFO (master + slave), the valid bits are in the higher bits [7:x], lower bits are pad bits.

Bit	7:5	4:0
Name	reserved	
R/W	ro	rw
Reset	0x0	0

Figure 48: SPI Control Register 3 (Address Offset: 0x03)

SPI Control Register 3 is reserved for future use.

Name	Description
	future use
	planned for optional 3-wire SPI interfaces (bi-directional MISO/MOSI)

Table 34: SPI Control Register 3 Descriptions

Bit	7	6	5	4:3	2:0
Name	Busy	reserved	TXNE	reserved	RXlvl
R/W	ro	ro	ro	ro	ro
Reset	0	0	0	0	0

Figure 49: SPI Status Register (Address Offset: 0x04)

Name	Description
RXlvl	Filling level of RX data FIFO, 0 (empty) to 4 (full)
TXNE	TX FIFO not empty: 1 = TX FIFO contains data 0 = TX FIFO is empty
SPIbusy	SPI interface busy state: 1 = some data transfer on the SPI 0 = SPI is currently idle

Table 35: SPI Status Register Descriptions

Bit	7:0
Name	SPI clock prescaler ratio
R/W	rw
Reset	24 decimal

Figure 50: SPI Baud rate register (Address Offset: 0x05)

The Baud rate register defines the SPI clock frequency as generated in the SPI master. In slave mode, the register is irrelevant since the slave operates based on the SPI clock coming from its master via the SCK input.

The Baud rate calculates according to the equation:

SPI Baud Rate = System Clock Frequency / (2 * (Clock prescaler + 1))

The following table shows SPI baud rate selection examples based on a 50 MHz system clock:

Baud rate register content	BR Divisor	Baud Rate [kHz]
0	2	25 000
1	4	12 500
2	6	8 333
3	8	6 250
4	10	5 000
10	22	2 273
20	42	1 190
24	50	1 000
50	102	490
100	202	248
200	402	124

Table 36: SPI Baud Rate Selection Examples

Other register address offsets between 0x06 and 0x1F are reserved and not accessible.

7.6. Inter-Integrated Circuit Interface (I²C)

The full I²C-bus specification is available in reference [08] of NXP Semiconductor.

Here are some of the features of the implemented I²C-bus:

- Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCL).
- Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times; masters can operate as master-transmitters or as master-receivers.
- It is a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer.
- Serial, 8-bit oriented, bidirectional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in Fast-mode Plus, or up to 3.4 Mbit/s in the High-speed mode.
- On-chip filtering rejects spikes on the bus data line to preserve data integrity.
- The number of ICs that can be connected to the same bus is limited only by a maximum bus capacitance. More capacitance may be allowed under some conditions.

Data transfers follow the format shown in Figure 51. After the START condition, a slave address is sent. This address is seven bits long followed by an eighth bit which is a data direction bit (R/W) — a ‘zero’ indicates a transmission (WRITE), a ‘one’ indicates a request for data (READ). A data transfer is always terminated by a STOP condition generated by the master.

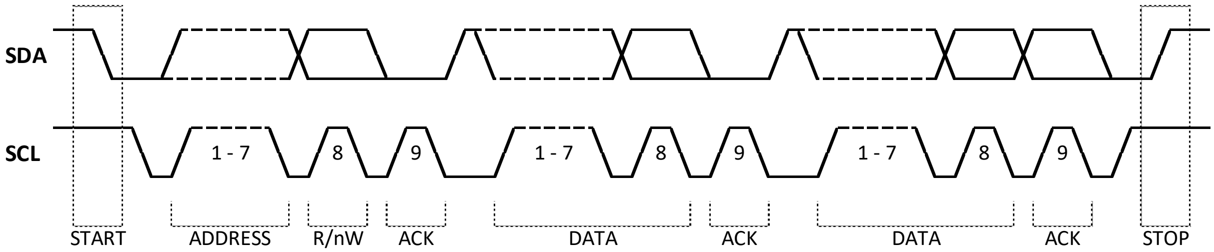


Figure 51: I²C-Bus Data Transfer

The I²C module is compliant to the specification given in [08]. The single exception is that the multi-master mode is not implemented, i.e., clock synchronization and arbitration between two or more masters on the same I²C bus is not implemented. The I²C module can be dynamically configured to be either master or slave.

Base address I²C0: 0x8100 E000
Base address I²C 1: 0x8100 F000

The I²C registers are 8-bit wide. There are 5 of them as follows:

Bit	7:0
Name	I2C data to write or read
R/W	rw
Reset	not applicable

Figure 52: I²C Data register (Address Offset: 0x00)

The data register provides access to the TX and RX data FIFOs, each of 4 byte capacity. They shall be used in I²C master and slave modes to store the data to be transmitted or that have been received, respectively. The number of data bytes in the RX FIFO can be read from the status register 4.

An I²C data transfer is initiated at the master when the software writes the address byte to the TX FIFO. Bit [0] of this byte determines the direction: '0' to write data to the slave, '1' to read data from the slave. Further data bytes shall be written in time to the master's TX FIFO. When writing to the slave, these bytes will be transferred. When reading from the slave, these bytes are dummy. The data received from the I²C bus will be written to the master's RX FIFO. The transfer continues until the TX FIFO is empty when the ACK bit of some byte is being transferred on the I²C bus. In case of a read transfer, the master (who is generating the ACK in this case) will indicate the end of the transfer by sending NACK (SDA = '1' in place of the ACK bit). This allows the slave to detect the end and to stop further sending of data.

In a slave, the data to be read by the master must be put into the TX FIFO in advance, before they are to be sent out. If the TX FIFO is empty in such a moment, random data will be sent. Received bytes are stored in the RX FIFO (capacity 4 byte). An interrupt is optionally generated at the end of each I²C transfer (on the STOP condition), and when either the TX FIFO becomes empty (the last bytes is read out) or the RX FIFO becomes full (the 4th byte is written in).

Bit	7	6	5	4	3	2:0
Name	EN	MSTR	TIE	RIE	PPSCL	Reserved
R/W	rw					
Reset	0x0					

Figure 53: I²C Control Register (Address Offset: 0x01)

Name	Description
PPSCL	**Push-pull driver for SCL line (master only):** Setting this bit lets the I^{2}C master drive the SCL output with a push-pull driver (instead of open-drain) for faster SCL clock signal (steeper low → high edges). 1 = push-pull mode 0 = standard mode (open-drain)
RIE	**I^{2}C Receive Interrupt Enable Bit**: This bit enables I^{2}C receiver interrupt requests. Interrupts are generated when the RX FIFO becomes full and at the end of an I^{2}C transfer. 1/0 = I^{2}C RX interrupts enabled/disabled.
TIE	**I^{2}C Transmit Interrupt Enable**: This bit enables I^{2}C transmitter interrupt requests. Interrupts are generated when the TX FIFO becomes empty and at the end of an I^{2}C transfer. 1/0 = TX interrupt enabled/disabled.
MSTR	**I^{2}C Master/Slave Mode Select Bit**: This bit selects, whether the I^{2}C operates in master or slave mode. Switching the I^{2}C from master to slave or vice versa forces the I^{2}C system into idle state. 1 = I^{2}C is in Master mode 0 = I^{2}C is in Slave mode
EN	**I^{2}C System Enable Bit**: This bit enables the I^{2}C system and dedicates the I^{2}C port pins to I^{2}C system functions. If EN is cleared, I^{2}C is disabled and forced into idle state. 1 = I^{2}C enabled, port pins are dedicated to I^{2}C functions. 0 = I^{2}C disabled (lower power consumption).

Table 37: I^{2}C Control Register Description

Bit	7:1	0
Name	ADDR[6:0]	reserved
R/W	rw	
Reset	0x0	

Figure 54: I^{2}C Address Register (Address Offset: 0x02)

The MORAL microcontroller I^{2}C components use only 7-bit addressing. The address register is used primarily in slave mode, so that the slave may determine whether it is selected for transaction by sensing the I^{2}C bus.

Name	Description
ADDR[6:0]	**Address**: The 7-bit address used for 7-bit addressing (I^{2}C slaves only)
Bit [0]	Future Use

Table 38: I^{2}C Address Register Descriptions

Bit	7	6	5	4	3	2:0
Name	Busy	AddrErr	TXNE	SlaveBusy	DataErr	RXlvl
R/W	ro	ro	ro	rw		ro
Reset	0	0	0	0	0	0

Figure 55: I^{2}C Status Register (Address Offset: 0x04)

The I^{2}C Status Register mainly reports on the filling level of the data FIFOs and on NACK (No Acknowledgement) bits that have possibly been detected on the I^{2}C bus.

Name	Description
RXlvl	Filling level of RX data FIFO, 0 (empty) to 4 (full)
DataErr	NACK has been received after sending a data byte (relevant in master only, in a slave this bit is normally always set since the master announces the end of the I^{2}C transfer with a NACK)
SlaveBusy	This *read-write* bit can be set in a slave to let it send a NACK bit in order to announce to the I^{2}C master that the slave is busy with some operation and cannot process further I^{2}C commands.
TXNE	TX FIFO not empty: 1 = TX FIFO contains data 0 = TX FIFO is empty
AddrErr	NACK has been received after sending the I^{2}C address byte, i.e. no slave responded (master only)
Busy	I^{2}C interface busy state: 1 = some data transfer on the I^{2}C 0 = I^{2}C is currently idle

Table 39: I^{2}C Status Register Descriptions

Bit	7:0
Name	I^{2}C clock prescaler
R/W	rw
Reset	124 decimal

Figure 56: I^{2}C Baud rate register (Address Offset: 0x05)

The Baud rate register defines the I^{2}C clock frequency as generated in the I^{2}C master. In slave mode, the register is irrelevant since the slave operates based on the I^{2}C clock coming from its master via the SCL input.

The Baud rate calculates according to the formula:

I^{2}C Baud rate = System Clock frequency / (4 x (Clock prescaler + 1))

The following table shows I^{2}C baud rate selection examples based on a 50 MHz system clock:

Baud rate register content	BR Divisor	Baud Rate [kHz]
12	50	1 000
31	125	400
124	500	100

Table 40: I^{2}C Baud Rate Selection Examples

Other register address offsets till 0x1F are reserved and not accessible.

7.7. Universal Asynchronous Receiver and Transmitter (UART)

The MORAL microcontroller integrates 4 UART ports. Each UART port consists of a transmitter/receiver block, two FIFO buffers for receiver and transmitter, and a FIFO controller. The FIFO buffers are realized using flip-flops. The block diagram of a UART port is presented in Figure 57. All 4 UART blocks are provided with handshake signals RTS and CTS. The debug UART can be configured to operate as AHB bus masters.

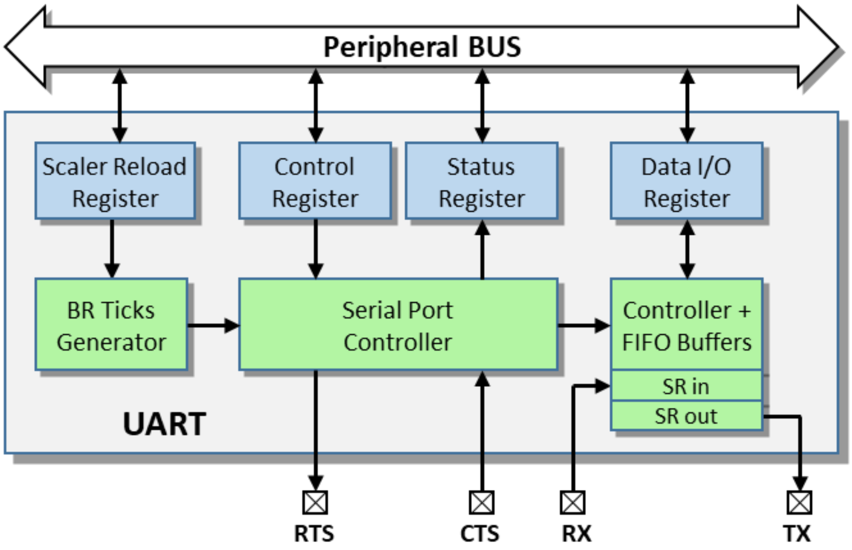


Figure 57: UART Block Diagram

The size of the FIFO buffers is 32 bytes for the receive path and 16 bytes for the transmit path. The FIFO controller manages all accesses to the FIFO. A CPU read access to an empty FIFO as well as a CPU write access to a full FIFO will be ignored and the related error flags set instead. When all data in the TX FIFO are transmitted (FIFO empty) a zero flag is set and an interrupt may be generated. Otherwise, if the incoming UART data are stored in the RX FIFO until it is full, a full flag is set and another interrupt may be generated.

If the CPU is not able to read the data, new UART input data cannot be stored (data loss) end an overflow error flag is set. To avoid huge area and power consumption, the FIFO buffers are not TMR-protected. Therefore software checks of coded data (CRC) should be done. A UART parity bit for hardware check/generation may be configured too. A UART frame consists of one start bit (ST = “low”), 8 data bits, optional parity bit (P), and one or two stop bits (SP1/2 = “high”):

ST	D0	D1	D2	D3	D4	D5	D6	D7	P	SP1	SP2

Figure 58: UART data frame

The UART baud rate is programmable by a 12-bit pre-scaler to generate the UART ticks from the input clock CLK. The UART tick frequency is 8 times the desired baud rate to guarantee the synchronization with start and stop bits. The serial input is shifted with the input clock CLK through an 8-bit shift register where all bits have to have the same value before the new value is taken into account, effectively forming a low-pass filter with a cut-off frequency of 1/8 input clock.

The (low-active) hardware flow control signals (RTSn, CTSn) are available for all UART blocks. The RTSn output signal (request-to-send) is de-asserted “high” as soon the RX FIFO is almost full, i.e. 28 of 32 bytes are full. If the CTSn input signal is de-asserted “high” the UART transmitter stops after an already active byte transmission has been finished.

UART0 is used by the debug unit and used 4 additional registers – see Subsection 4.3.3. All 4 general registers used within each UART module are described in detail below:

Base address UART 0: 0x81006000
Base address UART 1: 0x81007000

Base address UART 2: 0x81008000
Base address UART 3: 0x81009000

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved								D7	D6	D5	D4	D3	D2	D1	D0
R/W	undefined								rw							
Reset									undefined							

Figure 59: UART Data I/O Register (Address Offset: 0x00)

D[7:0] - write access: corresponds to next empty TX FIFO word
- read access: corresponds to the first stored RX FIFO data

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SB	LB	FL	PE	PS	TE	RE	M8	M7	M6	M5	M4	M3	res	M1	M0
R/W	rw															
Reset	0x0															

Figure 60: UART Control Register (Address Offset: 0x04)

Name	Description
M[8:0]	“1” = Interrupt enable bits enabled, corresponding to status/interrupt register bits 8..0
RE	“1” = Receiver enabled, if RE = “0” the corresponding RX FIFO will be cleared.
TE	“1” = Transmitter enabled, if TE = “0” the corresponding TX FIFO will be cleared.
PS	Parity selection (“0” = even, “1” = odd)
PE	“1” = Parity enabled
FL	“1” = Flow control enabled (RTS, CTS)
LB	“1” = Loop back mode enabled
SB	Stop bits (“0” = one stop bit, “1” = 2 stop bits)

Table 41: UART Control Register Descriptions

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved							FE	PE	OV	RF	BR	TF	TH	TS	DR
R/W	undefined							rw			ro					
Reset								0x0						1	1	0

Figure 61: UART Status Register (Address Offset: 0x02)

All status bits may generate interrupts. Interrupts may be bitwise enabled (except TH).

Name	Description
DR	“1” = Data ready for read (new data stored in FIFO), static interrupt if enabled
TS	“1” = Transmitter shift register empty (start/stop-bit not stored in shift register)
TH	“1” = TX FIFO empty, not interrupt capable
TF	“1” = TX FIFO full, if TX FIFO not full and enabled: static interrupt
BR	“1” = Break received (data bits = “0”, stop bit = “0”), automatically reset to “0” when RX = “1”
RF	“1” = RX FIFO full
OV	“1” = RX FIFO overflow (received data discarded), interrupt is deleted by writing “0”

PE	"1" = Parity error detected, interrupt is deleted by writing "0"
FE	"1" = Framing error detected (data bits ≠ "0", stop bit = "0"), interrupt is deleted by writing "0"

Table 42: UART Status Register Descriptions

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved				Scaler Reload Value											
R/W	undefined				rw											
Reset					0											

Figure 62: UART Scaler Reload Register (Address Offset: 0x06)

This register holds the scaler reload value for generating the UART ticks. The scaler is clocked by the input clock CLK and provides the desired baud rate (BR).

7.8. Pulse Counter (PULSE)

The MORAL microcontroller integrates four digital pulse counter inputs. Each pulse counter is 32-bit wide and increments the value upon receiving the external pulse signal. It can be configured whether rising or falling edges of the extern pulse signal are used for counting. This can be re-configured during the counting procedure. Furthermore, it is possible to reset or read the counter value. The block diagram of the Pulse Counter is shown in Figure 63.

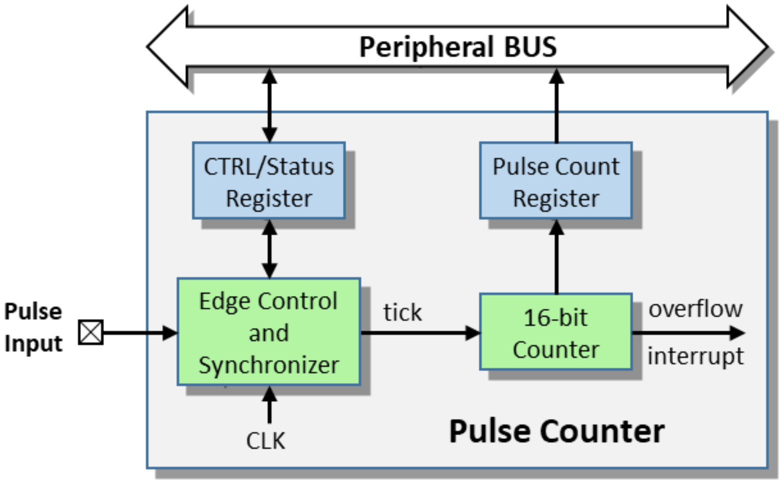


Figure 63: Pulse Counter Block Diagram

Base address pulse counter 0:	0x8101 4000
Base address pulse counter 1:	0x8101 5000
Base address pulse counter 2:	0x8101 6000
Base address pulse counter 3:	0x8100 7000

Both pulse counters are incremental counters whose input signals are internally double-buffered in respect to the input clock CLK. Therefore, the input pulse frequency has to be at least 4x lower than the CLK frequency, i.e., each pulse L/H state should last at least two CLK periods.

Bit	31:0
Name	Pulse Counter Value
R/W	ro
Reset	0x0

Figure 64: Pulse Counter Register (Address Offset: 0x00)

Bit	31:4	3	2	1	0
Name	OVFLW	reserved	CLEAR	NEDGE	EN
R/W	rw	undefined	rw	rw	rw
Reset	0x0		0	0	0

Figure 65: Pulse Counter Control/Status Register (Address Offset: 0x04)

The control/status register for each pulse counter includes:

Name	Description
EN	Enable pulse counter (0-disabled; 1-enabled)
NEDGE	Active edge control (0-count falling edges, 1-count rising edge of the external pulse)
CLEAR	Reset the Pulse Counter Register and the OVFLW field to zero (this CLEAR bit is auto-reset to 0)
OVFLW	Overflow count. (number of overflows of the Pulse Counter Register

Table 43: Pulse Counter Control/Status Register Descriptions

7.9. Pulse Width Modulator (PWM)

The pulse width modulator (PWM) provides 8 channels which may operate independently or complementary in up to four 2-channel pairs.

7.9.1. Independent Operation

Each channel may operate in one of two modes: Single shot mode or continuous mode. Depending on the selected operating mode, the corresponding output pin produces an appropriate signal waveform which is shown in Figure 66 including the period definitions Ta, Tb and Tc.

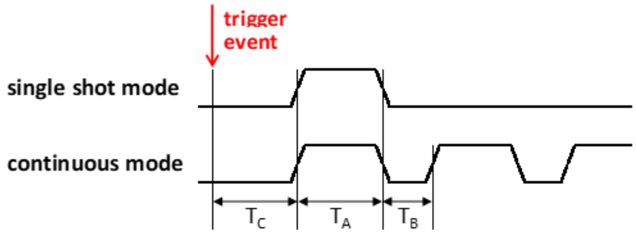


Figure 66: PWM outputs in single-shot mode and continuous mode

When a channel operates in single shot mode, the output generates one pulse only. In continuous mode, the output alternates between high and low levels in pre-defined time intervals. The pulse length in the single shot mode is determined by a 16-bit input parameter (Ta). In the continuous mode, Ta defines the time of the active (start) level, while another 16-bit parameter (Tb) determines the time of the inactive (second) level. The processor triggers a START bit which initiates the modulation process. To get the PWM channels synchronized but shifted in time, a delay time Tc can be configured to define an initial period in inactive state. This initial wait time can be set for each channel individually.

In single shot mode, the output stays in inactive state for time Tc after reception of the trigger signal, and then it goes to the active state and remains active for the duration Ta. Then it falls back to the inactive state waiting for a new trigger event. The PWM module uses the values of Ta and Tc given at the trigger event.

In the continuous mode, the output stays in inactive state for time Tc after reception of the trigger signal, and then it goes to active state and remains active for the duration Ta. Then it falls back to the inactive state and remains inactive for the duration Tb. Then it starts over with the active phase of duration Ta. This modulation process repeats until the operating mode gets changed.

To allow long Ta/Tb time settings, the PWM is driven by a common 12-bit prescaler (see Subsection 6.1.4). Furthermore, each channel has its own individual 12-bit pre-scaler. The settings of the individual the channel pre-scaler value, the continuous/single shot mode, and the idle output state low/high are done in the PRESCLHSC register. The block diagram of a single PWM channel is provided in Figure 67.

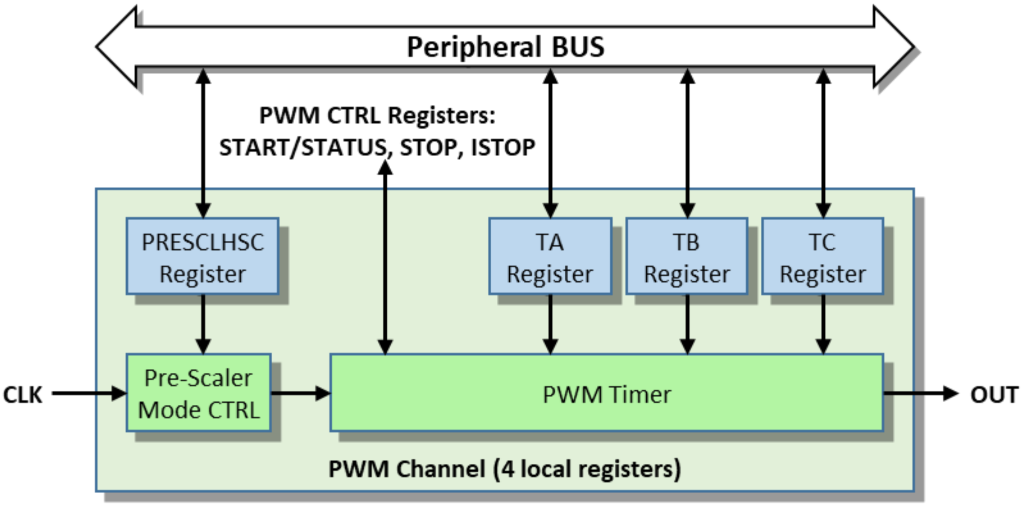


Figure 67: PWM Single Channel Block Diagram

7.9.2. Complementary Operation

The PWM channels may be (independently from one another) arranged in complementary mode of operation. That is, the two PWM outputs in the pair operate in complementary, non-overlapping fashion for the purposes like H-bridge DC/DC and motor control. Figure 68 shows the output waveforms of channels 0 and 1 operating in complementary mode. In the simplest case, the values TA, TB, and TC for both channels are equal, and the LH values (PRSC_LH_SC register, bit [1]) are complementary. Phase shifts at the rising and falling edges may be introduced by slight variations in the TC value, and perhaps also in TA and TB. However, the sum TA + TB must be equal for both PWM channels. Of course, also the prescaler must be the same.

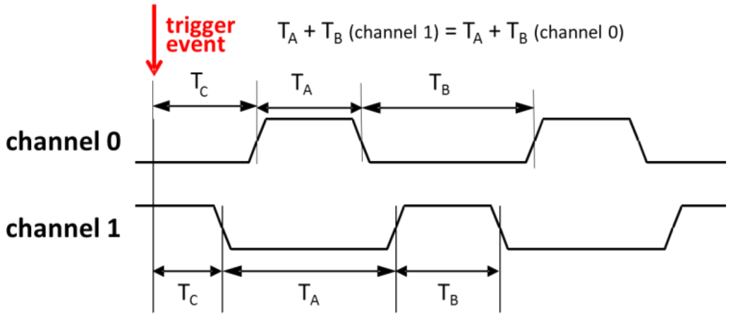


Figure 68: PWM Complementary Channel Operation

7.9.3. PWM Registers

The address offsets of all (4 x 8-bits, 32 x 16-bits) PWM registers is shown below. A 7-bit wide address ADDR[6:0] is used to distinguish the PWM registers.

The base address of PWM registers is: 0x8100 5000.

ADDR			Width (bits)	Access	Register
[6:4]	[3:2]	[1:0]			
0 0 0	0 0	0 0	8 16	wo ro	START STATUS
		0 1	-	-	Future use
		1 0	8	wo	DSTOP
		1 1	8	wo	ISTOP
Channel Number (0…7)	0 1	0 0	16	rw	PRSC_LH_SC
		1 0	16	rw	TA
	1 0	0 0	16	rw	TB
		1 0	16	rw	TC

Table 44: PWM Registers Address Offsets

Table 44 provides the address assignment.

- There is one START, one DSTOP, and one ISTOP register, each 8 bit wide corresponding to the eight channels (bits [15:0] are *don't care* when the register is written).
- When reading from the same addresses, a 16 bit status word is returned. It contains in bit [7:0] a '1' for each channel that is not idle and in bit[15:8] the actual logic output level of each PWM channel.
- There are eight blocks of 16-bit PRSC_LH_SC, TA, TB, and TC registers for each of the eight PWM channels, respectively.

ADDR[6:4] defines the channel number when specifying the address of PRSC_LH_SC, TA, TB or TC.

The PRSC_LH_SC, TA, TB, and TC registers are both writable and readable. The START, DSTOP and ISTOP functions are write-only. When a bit is set on writing to one of these registers, the respective function (start, stop) is triggered for the respective channel. This is possible since all bits in the START, DSTOP and ISTOP registers are auto-reset to 0 once written with 1. When reading from these registers, the 16-bit STATUS is returned. The bit assignment of the START, DSTOP, and ISTOP registers is the same as given in Figure 69.

START/DSTOP/ISTOP registers

Bit	7	6	5	4	3	2	1	0
Name	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
R/W	write-only							
Reset	0x0							

Figure 69: START / DSTOP / ISTOP Registers (Addr. Offset: 0x000 / 0x002 / 0x003)

START register

The start register is 8-bit wide register. Writing a 1 to a bit in the status register triggers the operation of the respective channel, with latching the values of the PRSC, TA, TB, and TC registers. If a channel which is already running is started again, the (possibly new) values of PRSC, TA, TB and TC will be re-loaded and the operation with the new values will be restarted immediately. This register is auto-reset to 0 in the following clock cycle after being written with 1.

DSTOP register

Register structure - see Figure 69 (Address Offset: 0x002).

The 8-bit wide STOP register delayedly stops the operation of the corresponding channel when written with 1. The operation is stopped after finishing completely the ongoing PWM cycle. Then the levels of the PWM channels are restored to the predefined values (bits LH in the PRSC_LH_SC register). Of course, writing a 1 to an already stopped channel does not have any effect. This register is auto-reset to 0 in the following clock cycle after being written with 1.

ISTOP register

Register structure - see Figure 69 (Address Offset: 0x003).

The 8-bit wide ISTOP register immediately stops the operation of the corresponding channel when written with 1. The operation is stopped immediately, not waiting the ongoing PWM cycle. The levels of the PWM channels are immediately restored to the predefined values (bits LH in the PRSC_LH_SC register). Of course, writing a 1 to an already stopped channel does not have any effect. This register is auto-reset to 0 in the following clock cycle after being written with 1.

STATUS register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
	channel's current output level								high when channel is not idle							
R/W	read-only															
Reset	0x0															

Figure 70: STATUS Registers (Addr. Offset: 0x000 / 0x002 / 0x003)

STATUS register

When reading from the addresses 0 to 3, a 16 bit status word is returned. In bit [7:0] it contains a '1' for each PWM channel that is not idle. In bit[15:8] it contains the actual logic output level of each PWM channel.

PRSC_LH_SC registers

Bit	15:4	3:2	1	0
Name	PRESCL	reserved	LH	SC
R/W	rw	undefined	rw	rw
Reset	4		0	0

Figure 71: **PRESCLHSC Registers** (Address Offsets: 0x0N4, where N = channel number 0..7)

There are eight 16-bit wide PRSC_LH_SC registers containing a 12-bit PRESCALER register field for each channel, one LH bit defining the idle state of the channel, and one SC bit defining the single-shot/ continuous mode of operation.

Name	Description
SC	Continuous (0) / Single-shot (1) mode of operation
LH	Low (0) / high (1) idle state of channel
PRESC	12-bit pre-scaler applied on the clock (either external or internal)

Table 45: **PRESCLHSC Registers Descriptions**

TA, TB and TC registers

Bit	15:0
Name	TA/TB/TC
R/W	rw
Reset	9 for TA/TB 0 for TC

Figure 72: **Timer Registers TA, TB, TC (**Address offsets: 0x0N6, 0x0N8, 0x0NA; where N = channel 0..7**)**

For each PWM channel there are 16-bit wide TA, TB and TC registers which define the Ta, Tb, and Tc periods of the PWM waveforms (see Figure 66). Please pay attention that pulse length Ta and Tb will be equal to the register content TA + 1 and TB + 1, respectively. I.e. when TA = 0, the length of the pulse Ta will be 1 unit. In contrast, TC does contain the correct pulse length Tc. If TC = 0, there is no Tc period. The pulse Ta will immediately start.

8. PERIPHERAL BUS ANALOG COMPONENTS

The microcontroller includes 1 Analog-to-Digital converter (ADC) and up to 2 Digital-to-Analog converters (DAC), whose characteristics and operations will be described in the following paragraphs.

8.1. Analog to Digital Converter (ADC)

The Analog-to-Digital Converter is a general-purpose 12-bit SAR block. It is designed to operate in high radiation environments.

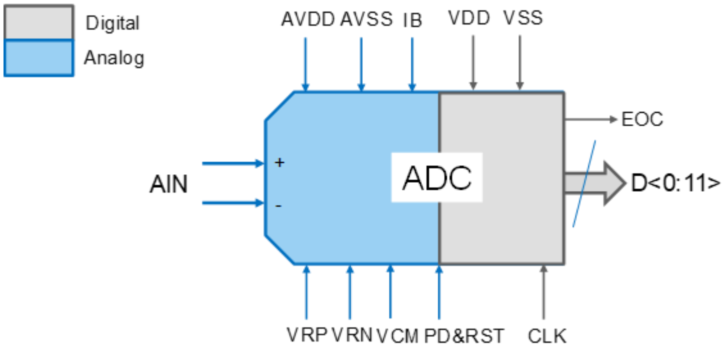


Figure 73: ADC Symbol

The microcontroller hosts one independent instance of the ADC. This ADC is supplied by independent AVDD_ADC and AVSS_ADC sources.

ADC main features summary:

- Resolution: 12-bit
- Operation frequency: 200kS/s (minimum)
- Differential Input
- Power-Down and Reset function

The ADC has an integrated analog multiplexer, MUX, in-front of it, which means that one MUX channel at a time can be measured. The ADC plus MUX can be configured as single-ended (using one input per-channel) or as differential using two input pins per-channels. In the former case, the unused input is connected to the common-mode voltage, VCM.

Separate Analog Supply and Ground for ADC and DAC used. Digital Supply and Ground are in common with overall digital circuitry.

8.1.1. ADC Digital control

The ADC controller can be set to operate as an AHB bus master. However, as a master it can only initiate write cycles at any address (allowed by the MPU) in the memory address space. Furthermore, the ADC controller can be set in a way that the converted data from each of the eight channels be written sequentially in a corresponding circular buffer that resides in memory. Thus, the microcontroller can read the converted ADC data from the circular buffers, process it and send it to the DAC for example. Together with some of the PEAKTOP instructions (i.e., RVB), this mechanism of circular buffers forms the hardware part of the DSP extension of the MORAL microcontroller.

Alternatively, the lastly converted 12-bit ADC data can be read from the ADC DATA register through the APB bus, no matter if the ADC is operating as an AHB master or not.

Bit	31:12	11:0
Name	reserved	DATA
R/W	undefined	ro
Reset		undefined

Figure 74: ADC DATA Register (Address Offset: 0x00)

Bit	31:24	23:12	11:9	8	7	6	5	4	3:2	1	0
Name	reserved	RELOAD	CHSEL	AHBERR	BUFEND	HWRITE	OVFLW	CONV	MMODE	AHBEN	EN
R/W	undefined	rw									
Reset		0x0	0x0	0	0	0	0	0	0x1	0	0

Figure 75: ADC CNTRL Register (Address Offset: 0x04)

Name	Description
EN	ADC enable. If disabled, the analog part is powered down. (0-disable; 1-enable)
AHBEN	AHB mode enable. If enabled, the ADC controller is an AHB master (0-disable; 1-enable)
MMODE	Machine mode (same as PEAKTOP [01]). Since the ADC is 12-bit wide, the most natural machine mode is halfword (16-bits), which is also the minimal mode which can be set.
CONV	Generate IRQ on each converted ADC data
OVFLW	Generate IRQ if data is not read and the NRCHNK counter overflows.
HWRITE	Generate IRQ on successful AHB write as master (in AHB mode only – AHBEN = 1)
BUFEND	Generate IRQ when the buffer end is reached, i.e., NRCHNK reaches max. value (in AHB mode only)
AHBERR	Generate IRQ on AHB write error (in AHB mode only)
CHSEL	Channel selector
RELOAD	Prescaler reload value for the ADC TIMER Register

Table 46: ADC CNTRL Register Description

The ADC digital logic generates an analog clock ACLK which drives not only the analog part of the ADC but also it can be set to drive the analog part of the DAC in order to obtain (when necessary) synchronized ADC and DAC clocks. ACLK is generated by the CLK clock through a 12-bit prescaler, i.e., the ADC TIMER Register:

Bit	31:12	11:0
Name	Reserved	TIMER
R/W	Undefined	rw
Reset		0x0

Figure 76: ADC TIMER Register (Address Offset: 0x08)

On each timer-out of the ADC TIMER Register the state of the ACLK is inverted (which implies 50% duty cycle), and the ADC TIMER Register it is reloaded with the RELOAD value of the ADC Control Register. A suitable value has to be supplied to the RELOAD field (taking into account the 50 MHz CLK) since the maximal operating frequency of the ADC is 3 MHz.

Bit	31:18	17:10	9:7	6	5	4	3:2	1	0
Name	reserved	NRCHNK	NRBYTE	AHBERR	BUFEND	HWRITE	OVFLW	CONV	NOTREAD
R/W	undefined	ro							
Reset		0x0							

Figure 77: ADC STATUS Register (Address Offset: 0x0C)

Name	Description
NOTREAD	ADC enable. If disabled, the analog part is powered down. (0-disable; 1-enable)
CONV	IRQ: converted ADC data
OVFLW	IRQ: data is not read and the NRCHNK counter overflowed.
HWRITE	IRQ: successful AHB write as master
BUFEND	IRQ: buffer end is reached
AHBERR	IRQ: AHB write error
NRBYTE	The current byte within the defined MMODE width
NRCHNK	The current converted data chunk

Table 47: ADC STATUS Register Description

For each of the eight channels the ADC controller has a separate 8-bit field that specifies the depth of the circular buffer in which converted data is placed. Thus, a maximal circular buffer depth of 256 is possible. These 8-bit fields are placed in two 32-bit wide registers NRCHNK30 and NRCHNK74:

Bit	31:24	23:16	15:8	7:0
Name	NRCHNK3/NRCHNK7	NRCHNK2/NRCHNK6	NRCHNK1/NRCHNK5	NRCHNK0/NRCHNK4
R/W	rw			
Reset	0x0			

Figure 78: ADC NRCHNK30/NRCHNK74 Registers (Address Offset: 0x10/0x14)

For each of the eight channels the ADC controller has a separate 32-bit starting address of the circular buffers:

Bit	31:0
Name	STADR0/STADR1/STADR2/STADR3/STADR4/STADR5/STADR6/STADR7
R/W	rw
Reset	0x0

Figure 79: ADC STADR0-STADR7 Registers (Address Offsets: 0x20/0x24/…/0x3C)

On overflow of the circular buffer the ADC starts overwriting at the corresponding STADR address. Therefore, in order not to lose data, the microcontroller should read the converted data before an overflow occurs.

8.2. Digital to Analog Converter (DAC)

The Digital-to-Analog Converter is a general purpose 12-bit resistive-string voltage output block. It is designed to operate in high radiation environments.

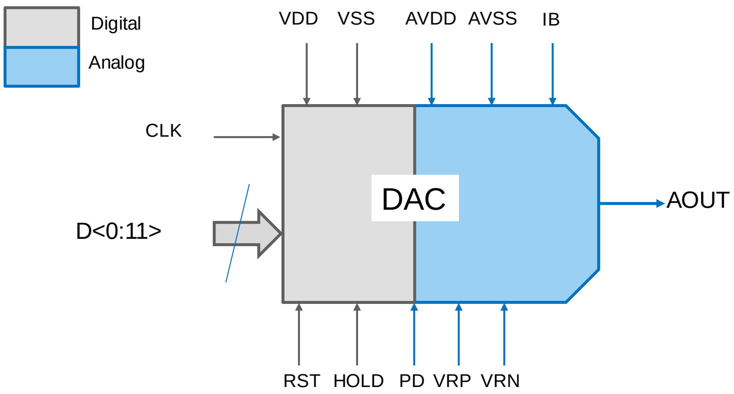


Figure 80: DAC Symbol

The microcontroller hosts two independent instances of the DAC. These DACs are supplied by independent AVDD_DAC and AVSS_DAC sources.

DAC main features summary:

- Resolution: 12-bit
- Operation frequency: 1MS/s (minimum)
- Guaranteed monotonicity
- Hold, Reset and Power-Down functions are available.

8.2.1. DAC Digital Control

The 12-bit DAC data to be converted can be written to the DATA register.

Bit	15:12	11:0
Name	reserved	DATA
R/W	undefined	rw
Reset		undefined

Figure 81: DAC DATA Register (Address Offset: 0x00)

Bit	15:4	3	2	1	0
Name	RELOAD	reserved	IRQEN	ADCCLK	EN
R/W	rw	undefined	rw	rw	rw
Reset	0x0		0	0	0

Figure 82: DAC CNTRL Register (Address Offset: 0x02)

Name	Description
EN	DAC enable. If disabled, the analog part is powered down. (0-disable; 1-enable)
ADCCLK	Analog clock driving the analog DAC part: 0-Let the DAC digital controller itself generate analog clock for the DAC analog part 1-Use the analog clock generated by the ADC digital controller (see Subsection 8.1.1) for the DAC analog part.
IRQEN	Generate IRQ pulse on each DAC conversion
reserved	for future use
RELOAD	Prescaler reload value for the DAC TIMER Register

Table 48: DAC CNTRL Register Description

The DAC digital logic can generate an analog clock ACLK driven by the CLK through a 12-bit prescaler, i.e., the DAC TIMER Register:

Bit	15:12	11:0
Name	reserved	TIMER
R/W	undefined	rw
Reset		0x0

Figure 83: DAC TIMER Register (Address Offset: 0x04)

On each timer-out of the DAC TIMER Register the state of the ACLK is inverted (which implies 50% duty cycle), and the DAC TIMER Register it is reloaded with the RELOAD value of the DAC Control Register. A suitable value has to be supplied to the RELOAD field (taking into account the 50 MHz CLK) since the maximal operating frequency of the DAC is 3 MHz.

Alternatively, the DAC can use the clock generated by the ADC in order to be synchronized with it. Setting the ADCCLK bit in the ADC CNTRL Register to 1 sets the DAC to use the analog clock generated by the ADC control logic (see Subsection 8.1.1).

9. Non-Functional Characteristics

9.1. Technology

The technology is SG13RH (S) process from IHP. For more information about the technology please contact IHP (www.ihp-microelectronics.com).

9.2. Power Supplies and Pin Descriptions

The chip uses two power supplies: core power supply 1.2V and IO power supply 3.3.V.

9.3. Chip IO Pad Characteristics (SG13RH Lib)

The following Figure 84, and Table 49 and Table 50 provides the information about the main IO cell B16M io pad cell B16M and their derivatives B16MPUP with pull-up resistance and B16MPDN with pull-down resistance used in the available IO library related to SG13RH(S), the respective truth table, and corresponding electrical characteristics. Both resistances are approximately 33 kΩ. For more details, the relevant PDK documentation could be referred.

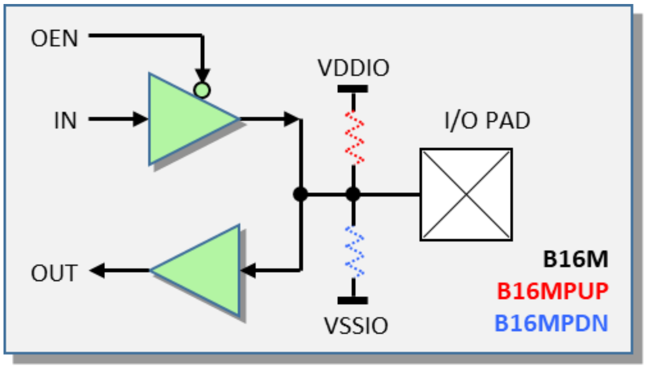


Figure 84: I/O PAD Cells (3 separate pad cells available)

OEN	IN	PAD B16M	PAD B16MPDN	PAD B16MPUP	OUT
0	0	0	0	0	0
0	1	1	1	1	1
1	X	0	0	0	0
1	X	1	1	1	1
1	X		Z -> 0		0
1	X			Z -> 1	1

Table 49: Truth Table B16Mxxx

Parameter	min	typ	max	Unit	Comment
Rise time		370	610	ps	
Fall time		320	520	ps	
Delay IN->PAD			4.61	ns	
Delay PAD->OUT			1.86	ns	
Load capacitance		12		pF	
I_{OL}, I_{OH}		16	19.2	mA	

Table 50: Electrical Characteristics B16M

9.4. *Chip IO Pin Description*

At the time of this document release there are no fully available layout and relevant information to provide the exact pin assignment. This information will be included in upcoming revisions of this datasheet. The pin list is presented in Table 51.

Module Name	Pin Name	I/O	Active State	Type	Package Pins	GPIO Mapped	Comments
SYSTEM	clk	I	n/a		1		System clock (50MHz)
	pclk	I	n/a		1		Peripherals clock (200MHz)
	rst	I	low		1		System reset
MEM CTRL	brdy	I	low	pd	1		Bus ready
	bexc	I	low	pu	1		Bus exception
	edacen	I	high	pd	1		EDAC enable
	edac	I/O	n/a		5		EDAC bus
	data	I/O	n/a		8		Data bus
	addr	O	n/a		29		Address bus
	cs	O	low		8		Chip select
	wren	O	low		1		Write enable
	oen	O	low		1		Output enable
	read	O	high		1		Read cycle
EXECUTION UNIT	nmi	I	high	pd	1		Non-maskable interrupt
	sync	O	high		1		SYNC bit from user control register
	busy	O	high		1		Core busy signal
	err	O	high		1		Core error signal
DEBUG UNIT	hold	I	high	pd	1		Hold execution request signal
WD Timer	wd_rsto	O	low		1		Watchdog timer reset output
SCAN/BIST	testmode	I	high	pd	1		Test mode enable
	scan_en	I	high	pd	1		Scan enable
ADC	adci	I	n/a		8		ADC input channels
	adc_vrp	I	n/a		1		ADC positive reference voltage
	adc_vrn	I	n/a		1		ADC negative reference voltage
	adc_vcm	I	n/a		1		ADC common mode voltage
	adc_ib	I	n/a		1		ADC bias input
DAC	daco	O	n/a		2		DAC data output
	dac_vrp	I	n/a		2		DAC positive reference voltage
	dac_vrn	I	n/a		2		DAC negative reference voltage
	dac_ib	I	n/a		2		DAC bias input
GPIO	gpio	I/O	n/a		64		General purpose IO pins
IRQ	irqi	I	high	gpio		4	IRQ interrupt inputs
PWM	pwmo	O	n/a	gpio		8	PWM output signals
UART	uart_tx	O	n/a	gpio		4	UART transmit signals
	uart_rx	I	n/a	gpio		4	UART receive signals
	uart_ctsn	I	low	gpio		4	UART clear to send
	uart_rtsn	O	low	gpio		4	UART request to send
SPW	spw_si	I	n/a	gpio		2	SpaceWire strobe input
	spw_di	I	n/a	gpio		2	SpaceWire data input

Module Name	Pin Name	I/O	Active State	Type	Package Pins	GPIO Mapped	Comments
	spw_so	O	n/a	gpio		2	SpaceWire strobe output
	spw_do	O	n/a	gpio		2	SpaceWire data output
SPI	spi_ssn	I/O	low	gpio		2	SPI select slave
	spi_sck	I/O	n/a	gpio		2	SPI serial clock
	spi_mosi	I/O	n/a	gpio		2	SPI master out slave in
	spi_miso	I/O	n/a	gpio		2	SPI master in slave out
I2C	i2c_scl	I/O	n/a	gpio		2	I2C serial clock
	i2c_sda	I/O	n/a	gpio		2	I2C serial data
CAN	can_tx	O	n/a	gpio		2	CAN transmit signals
	can_rx	I	n/a	gpio		2	CAN receive signals
PCNT	pcnt	I	n/a	gpio		4	PULSE COUNTER input signal
MIL-1553	mil_txa	O	n/a	gpio		2	MIL-1553 channel A transmit signals
	mil_rxa	I	n/a	gpio		2	MIL-1553 channel A receive signals
	mil_txb	O	n/a	gpio		2	MIL-1553 channel B transmit signals
	mil_rxb	I	n/a	gpio		2	MIL-1553 channel B receive signals
SUPPLY	vddc	I	n/a		13		Core supply
	vssc	I	n/a		13		Core ground
	vddio	I	n/a		13		IO supply
	vssio	I	n/a		12		IO ground
	vdda	I	n/a		3		Analog supply
	vssa	I	n/a		3		Analog ground
Summary					**208**	**64**	

Table 51: Chip IO Pin Count and Descriptions

9.5. Chip Package

The chip will be packaged in a ceramic, top-notch, rad-hard package which is pin-by-pin replaceable by a plastic package. At the current development the following packages are considered (selection to be finalized for the next release of the datasheet):

- NTK CQFP256, Kyocera CERQUAD 208, and Kyocera CQFP208

None of the listed packages are ITAR or EAR-99 restricted.

9.6. Temperature Range

The allowed environment temperature during operation is defined according to available model parameters:

- maximum temperature +125 °C
- minimum temperature –40 °C

9.7. Rad-Hard Parameters

The radiation hardness target of the ASIC chip are:

- Total ionizing dose (TID) 100 krad (Si)
- Single event upset (SEU) for the processor’s digital core >30 MeVcm^2/mg
- Single event latch-up (SEL) >60 MeVcm^2/mg

REFERENCES

[01] PEAKTOP Instruction Set Architecture manual: “peaktop_isa_v1_3_10_5_r210104.pdf”
[02] SPI module datasheet: “SPI_Motorola.pdf”
[03] SpaceWire specification: “SPW_RMAP_Specification_Architecture.pdf”, EADS Astrium
[04] AMBA specification (ARM Limited, Rev 2.0, May 1999)
[05] CAN License Conditions (Robert Bosch GmbH, July 2010)
[06] DCAN Configurable CAN Bus Controller – User Manual (Digital Core Design, 2014)
[07] MORAL Microcontroller Test and Verification Document
[08] NXP Semiconductors - I2C-bus specification and user manual, Rev. 6, April 2014
[09] MIL-STD 1553C.pdf

Revision history

DATE	Revision	CHANGES
10/01/2022	1	Initial release.

Disclosure Statement

The information contained in this document is the property of IHP GMBH – INNOVATIONS FOR HIGH PERFORMANCE MICROELECTRONICS/LEIBNIZ-INSTITUT FUER INNOVATIVE MIKROELEKTRONIK (IHP), REDCAT DEVICES SRL (RCD), SYSGO GmbH (SYSGO), THALES ALENIA SPACE ESPAÑA, SA (TASE), ABSINT ANGEWANDTE INFORMATIK GmbH (ABSINT) and it shall not be reproduced, disclosed, modified or communicated to any third parties without the prior written consent of the abovementioned entities.