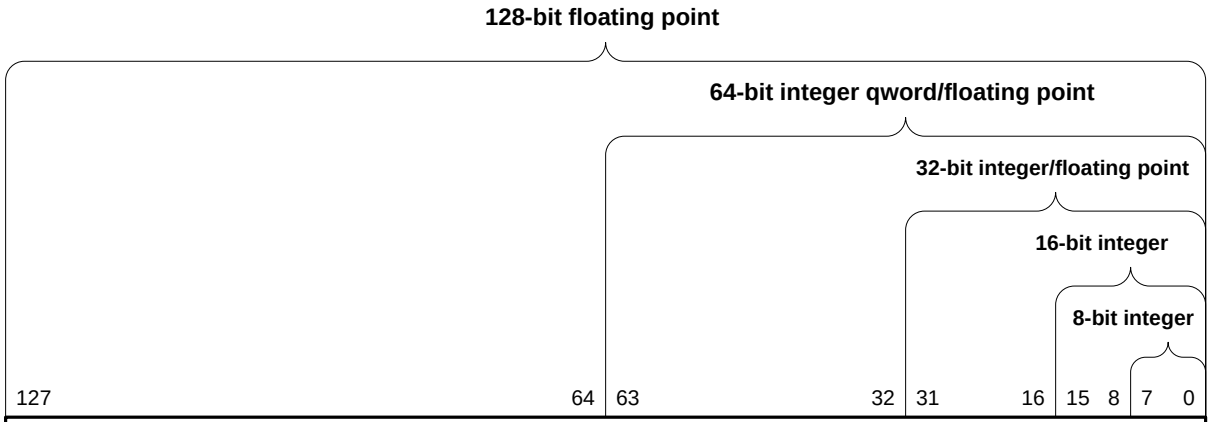
Contents

General purpose registers

Registers 128-bit, designed to store the processed data and do not have any specialized differences among themselves, they can all be used in any instructions.



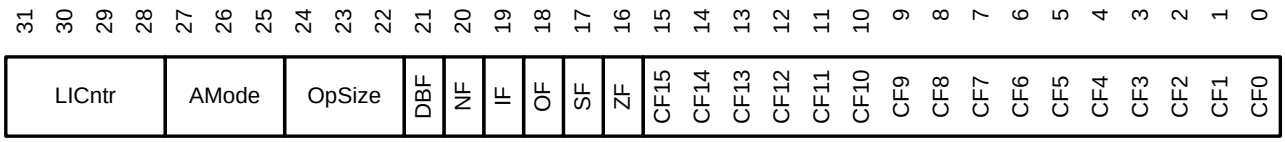
R15
R14
R13
R12
R11
R10
R9
R8
R7
R6
R5
R4
R3
R2
R1
R0

The data of integer formats can be 8-, 16-, 32- and 64-bit. Supports three formats with a floating point - 32, 64 and 128 bits.

Name	***Significand bits***	***Exponent bits***	***Decimal digits***	***Decimal Exp. Max.***
Single prec.	24	8	7.22	38.23
Double prec.	53	11	15.95	307.95
Quadruple prec.	113	15	34.02	4931.77

Flag registers

Each general purpose register has its own register of status flags. So, the core has sixteen flag registers named AFR [15: 0]. The flag register displays the state of the result of arithmetic/logic operations placed in the general purpose register, displays the size of data contained in the general-purpose register, the offset generation mode, and the byte number of the data loaded by LI instruction (see instruction set reference).



CF[15:0] – Carry flags. Used to perform binary-coded decimal correction of the results of addition or subtraction.

ZF. Zero Flag.

SF. Sign Flag.

OF. Overflow Flag.

IF. Infinity Flag.

NF. Not-a-Number Flag.

DBF. Data Bit Flag.

OpSize. Operand Size determines the bit width of the operand located in the general-purpose register. In instructions, there is no explicit indication of the width of the data being processed, except for the instructions for loading data from memory. The size of the operands is determined by the OpSize fields of the registers involved in the operation. The width of the result is chosen from the maximum value of the width of the initial operands. For example, if the operation involves a byte and a 32-bit word, the result will be 32-bit.

OpSize	***Integer***	***Floating point***
000	Byte (8-bit)	Single precision (32-bit)
001	Word (16-bit)	Single precision (32-bit)
010	DWord (32-bit)	Single precision (32-bit)
011	QWord (64-bit)	Double precision (64-bit)
100	Not used, reserved.	Quadruple precision (128-bit)
101	Not used, reserved.	
110		
111	Not-A-Number	

AMode. Address Mode selects the offset generation method in the data transfer commands between general-purpose registers and memory.

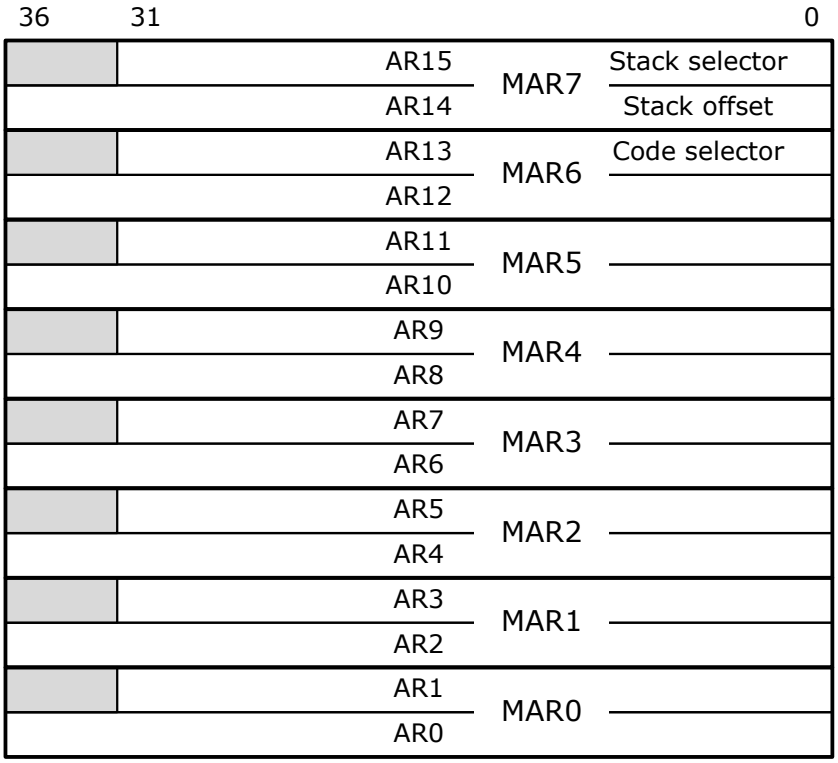
AMode	***Forming an offset and modifying the address register***
000	The contents of the address register completely determine the offset in the object and after the execution of the transaction it does not receive an increment.
001	The content of the address register completely determines the offset in the object and increments by the value from the general register after the transaction is completed.
010	The offset is completely determined by the contents of the general-purpose register.

AMode	*Forming an offset and modifying the address register*
011	The offset in the object is obtained by summing the contents of the address register and the bit [36: 0] of the content of the general-purpose register.
100	The content of the address register completely determines the offset in the object and after the transaction is increased by the number of bytes that made up the data element.
101	The content of the address register completely determines the offset in the object and after the transaction is performed, it decreases by the number of bytes that made up the data element.
110	The contents of the address register, in sum with the bits [36: 0] of the contents of the general-purpose register, determine the offset in the object. After the transaction is completed, the contents of the address register will be increased by the number of bytes that made up the data element.
111	The contents of the address register, in sum with the bits [36: 0] of the contents of the general purpose register, determine the offset in the object. After the transaction is completed, the contents of the address register will be decreased by the number of bytes that made up the data element.

LICntr. LI instruction counter - the LI instruction counter, which determines which byte of the general register will be loaded by the next LI instruction. LICntr is set to 0 by any instruction other than LI, which uses the register as the source of the operand or the result receiver. The LICntr state is stored in the process state object when the running process is interrupted in the middle of the LI command sequence and a transition to another process is performed. When the core returns to the execution of the interrupted process, LICnt will be restored according to the image stored in the process state object.

Address registers

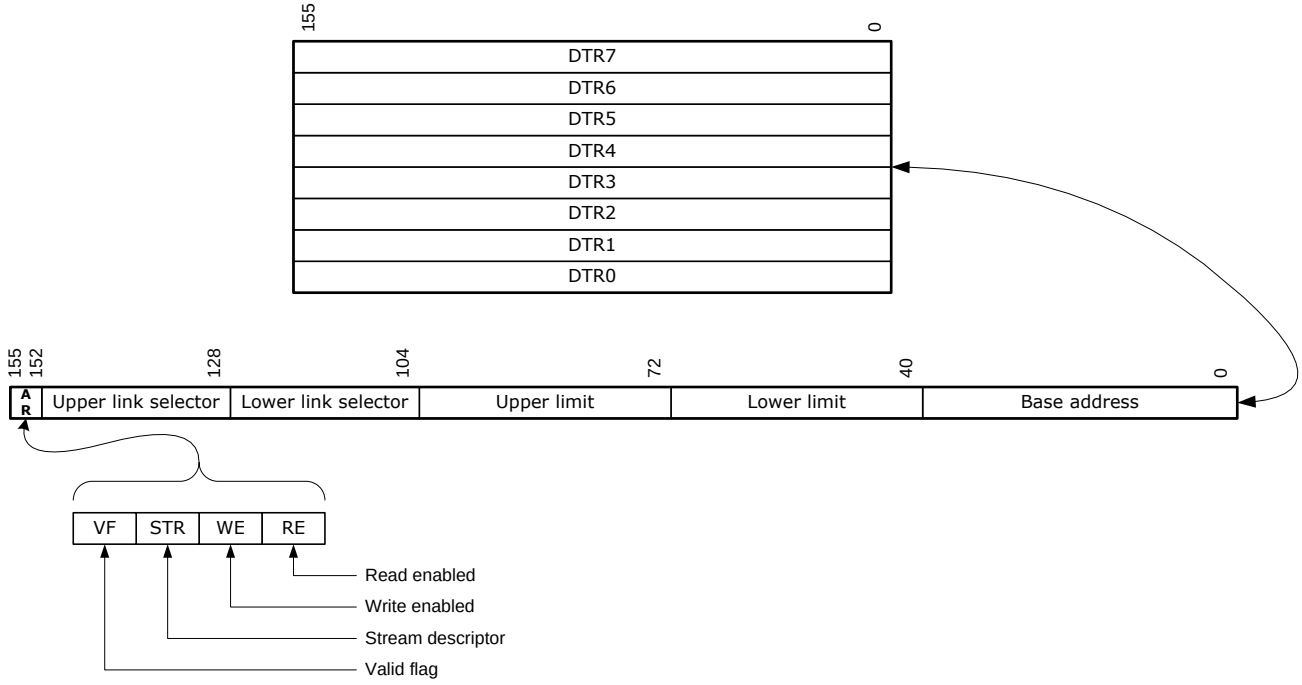
Address registers used to address memory in read/write transactions. The core has 16 address registers AR0-AR15. Address registers grouped into MAR0-MAR7 groups of two registers. Each group holds a 2-component address. One register of the group defines a 37-bit offset, and the second - a 32-bit selector. Registers AR0, AR2, AR4, AR6, AR8, AR10, AR12 and AR14 contain 37-bit offsets. Registers AR1, AR3, AR5, AR7, AR9, AR11, AR13 and AR15 are used for storing object selectors.



Some registers have a special purpose. AR13 from the pair MAR6 is used to store the code object selector. It cannot be modified by program at privilege levels of executable code other than CPL = 0 (CPL is a Current Privilege Level). With CPL <> 0 AR13 is set only when the core context is changed. The AR12 register can be used in the same way as any other register storing the offset. The MAR7 pair is initialized when the process context is loaded into the core. The AR14 pointer modified in the PUSH / POP / CALL / RET instructions when information pushed to the stack or retrieving information from the stack.

Descriptor cache registers

An individual descriptor register is associated with each AR1, AR3, AR5, AR7, AR9, AR11, AR13, and AR15 address register. Descriptor registers are not available for software. In these registers are stored the parameters of the current segment of the object used for memory access control, when the corresponding MAR register is selected for addressing.



Each descriptor register contains:

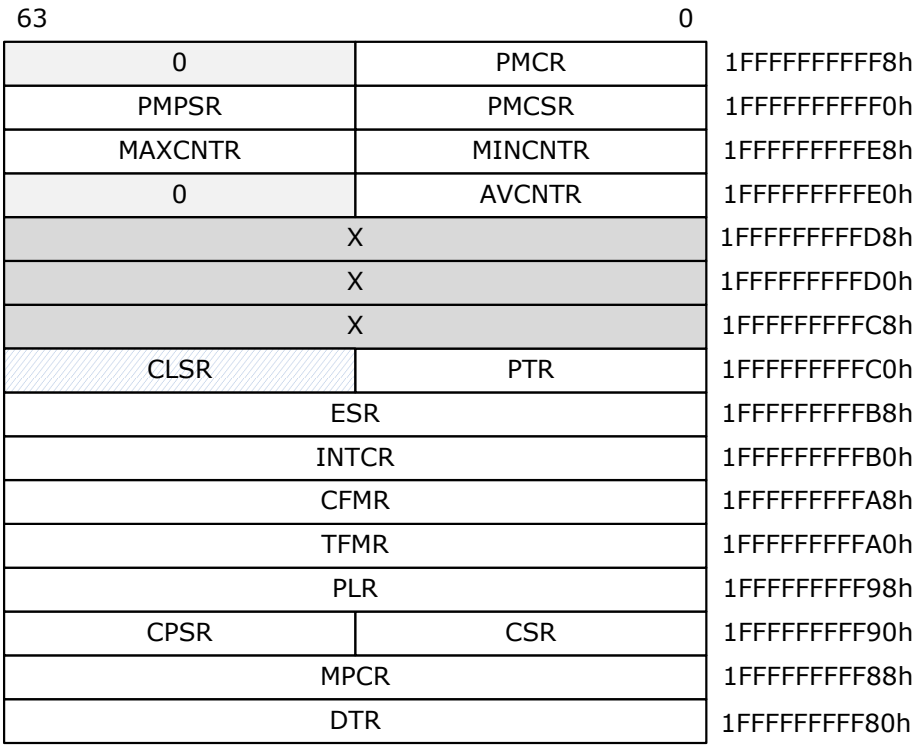
- 40 bits of the base address, the address expressed in 32-byte paragraphs;
- 32 bits of the lower limit of the segment;
- 32 bits of the upper object limit;
- 24 bits of the lower link selector;
- 24 bits of the upper link selector;
- 4 bits of flags that allow or forbid reading/writing of an object, a flag of an object of the type "stream" and a flag of the validity of the descriptor register.

The VF flag is reset to 0 each time a new object selector is written to the corresponding address register. VF is set to 1 after a valid descriptor has been loaded.

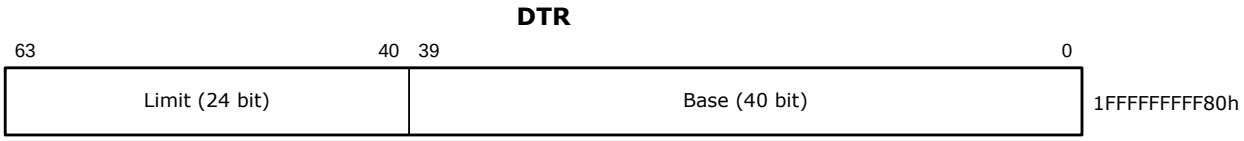
After a system reset, all base addresses except DTR5 are reset. The base address of the DTR5 is set to a value of 0FFFFFFFFCh, which defines the base address of the system register block. All lower limits are set to zero, all upper limits are initialized to the value 0FFFFFFFh - the maximum object size. Communication selectors are reset. Four bits of flags are initialized to 1011b.

System control registers

System registers are addressed as memory locations. They have fixed physical addresses. System registers can be modified only on CPL=0.

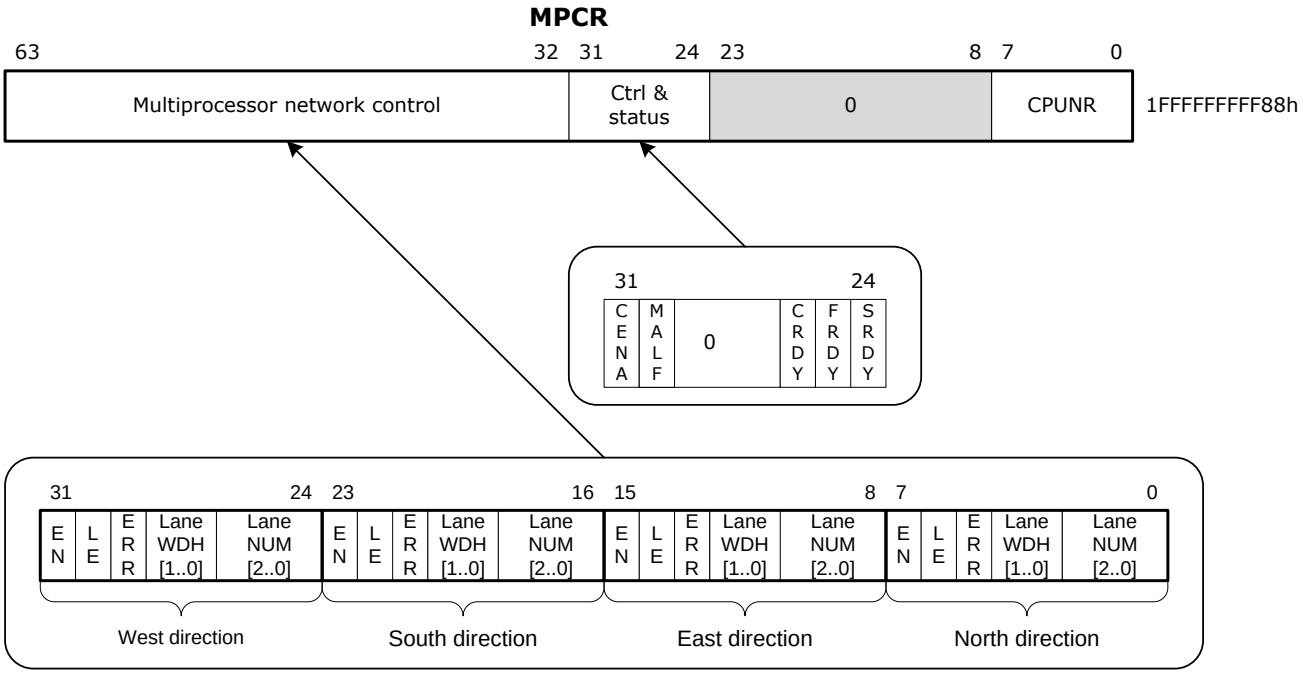


DTR. Descriptor Table Register



The register defines the base address of the descriptor table, represented in 32-bit paragraphs. To get the address expressed in bytes, the 40-bit value must be shifted 5 bits to the left. The 24-bit limit of the table contains the maximum number of object selectors whose descriptors are placed in the table. The DTR can be modified byte by byte.

MPCR. Multiprocessor Control Register



CPUNR – processor number in the network.

SRDY. Stream controller ready. The bit indicates the readiness of the stream controller to operate. The bit is cleared at system reset and is set when the stream controller completes the internal initialization.

FRDY. FPU Ready. The bit indicates that the FPU is ready for use. The bit is cleared during system reset and is set when the FPU completes the internal initialization.

CRDY. Context controller ready. The bit indicates that the context controller is ready for use. The bit is cleared during system reset or when CENA = 0 and is set when the context controller completes the internal initialization.

MALF. Memory Allocation Lock Flag. This bit is used to block the memory allocation system (blocking with MALF = 1). The lock should be used when changing the parameters of the descriptors of free memory blocks in the descriptor table. This is necessary to prevent disruptions in the memory allocation system. Transition of the MALF bit from 1 to 0 enables the operation of the memory allocation system and causes its re-initialization, which is necessary for analyzing the new state of the descriptor table.

CENA. Context controller enable bit. Bit of the work permit / initialization of the context controller. When CENA = 0 (state after reset), the controller is disabled, the start code is executed, the processor is unable to process messages, switch process contexts, and can't manage the memory allocation. The bit is used when the system initialization process occurs and the system tables are not yet ready to fully support the work of high-level functions.

LaneNUM (read only) The field displays the number of physical channels in the corresponding direction of multiprocessor communication.

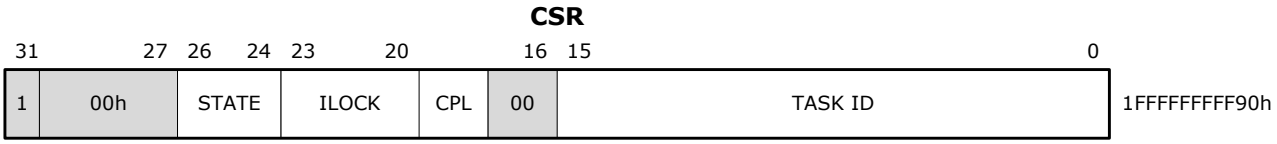
LaneWDH (read only) The field displays the channel width in the corresponding direction of 8-, 16-, 32- or 64-bit.

ERR (read only) A bit of hardware error in the channel. The bit can be cleared by the software.

LE (read only) The presence of the connection by direction. It is set to 1 if at least one of the channels of the north / east / south / west direction has a connection to another processor.

EN Bit enable work direction. If EN = 1, work in the appropriate direction is allowed.

CSR. Core State Register



Task ID. Task ID for the current process. It is used in the work of the memory protection system when checking the accessibility of the process to an object.

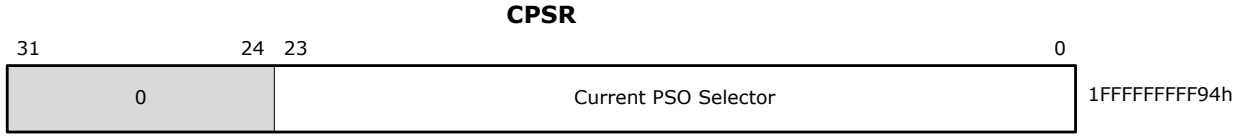
CPL. Current Privilege Level. It is used in the work of the memory protection system when checking the accessibility of the process to an object. The CPL is also used to verify the possibility of transmitting a message from one process to another.

ILOCK. Interrupt Lock bits. ILOCK [0] = 0 blocks the call to the system error handling routine. ILOCK[1]=0 blocks the hardware interrupts. ILOK [2] = 0 blocks hardware switching of processes. ILOCK [3] = 0 blocks the processing of messages.

STATE. Determines the current state of the process.

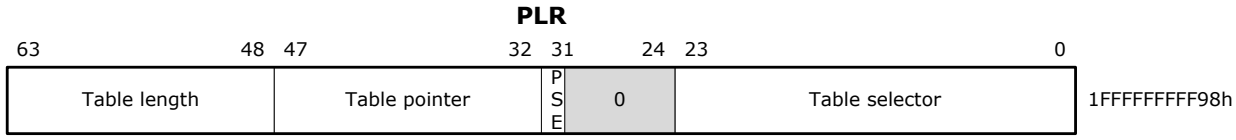
STATE	***Description***
0	The main process code.
1	Sleep state.
2	Exception processing.
3	Interrupt processing.
4	System message processing.
5	Regular message processing.
6	Reserved.
7	Reserved.

CPSR. Current Process state object Selector Register



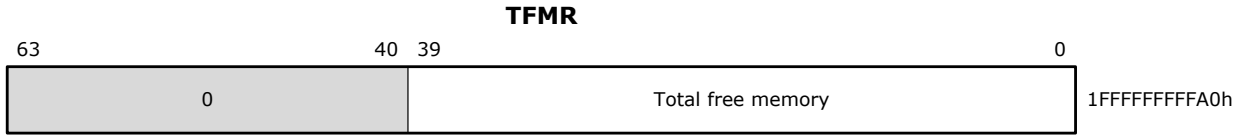
The register contains the PSO selector of the current process.

PLR. Process List Register



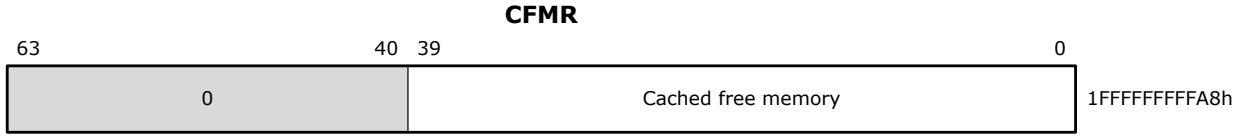
The register is used in the automatic process switching mechanism. Table selector - an object selector in which the table of processes processed in the time-division mode is placed from the zero offset. The PSE bit allows (with PSE = 1) the operation of the automatic process switching mechanism. Table Pointer A 16-bit process table pointer. The pointer is incremented by 1 every time the process activity timer completes the count. The pointer goes to 0 when it reaches the Table Length value. Table Length A 16-bit table limit that determines the number of records in the table.

TFMR. Total Free Memory Register



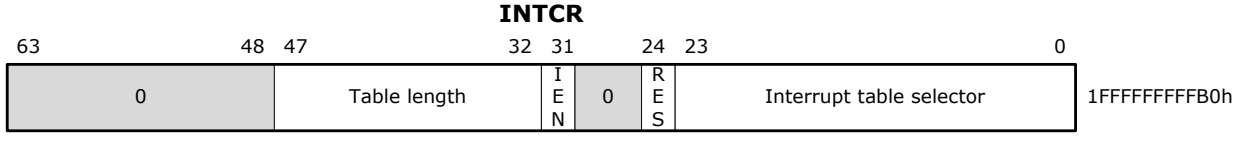
The register contains the results of calculating the value of the total amount of free memory space. This calculation periodically produces a memory manager built into the context controller. The amount of free space is expressed in 32-byte paragraphs.

CFMR. Cached Free Memory Register



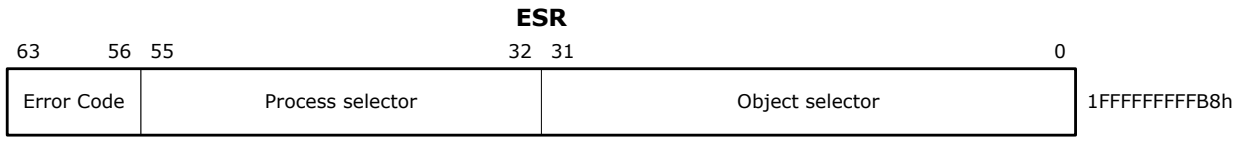
Free memory space can be divided into many separate objects in such a number that references to all these objects will not fit into the cache table of the memory allocation system. In this case, the TFMR and CFMR registers will contain different values. This situation may tell the operating system to start a search procedure and merge adjacent free objects, since the memory allocation system built into the context controller, does not perform such a function.

INTCR. Interrupt Control Register



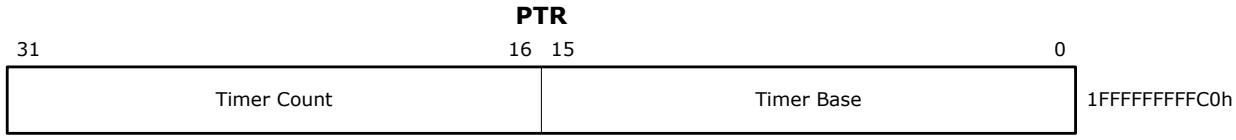
The register contains an object selector in which an interrupt table is placed from the zero offset. RES - interrupt controller reset bit. The IEN interrupt enable bit. Table length - the number of records in the interrupt table.

ESR. Error Status Register



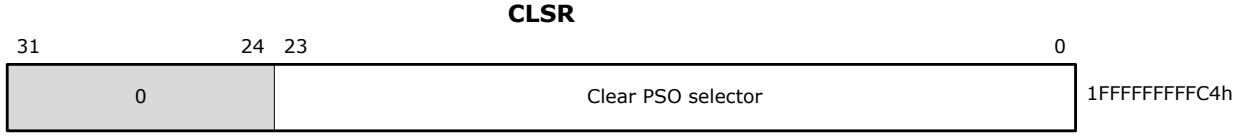
The register displays an object selector that, when accessed, detected a violation of the protection rules, the PSO selector of the process that caused the violation and the error code. The register is the FIFO output when reading which, the next value of the error code is output. The presence of a valid error message can be determined by the error code, if it is not 0, then the value read from the register is valid.

PTR. Process Timer Register



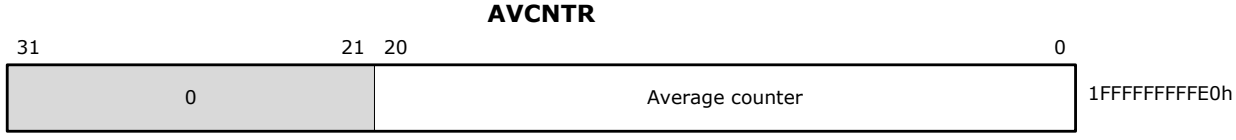
The lower 16 bits display the base value of the process activity counter. Bits [31:16] display the current status of the process counter. A process change is initiated when the current process counter value transition from state 1 to state 0 occurs. Modify only the lower 16 bits of the register, while resetting the counter to a new value.

CLSR. Clear selector register



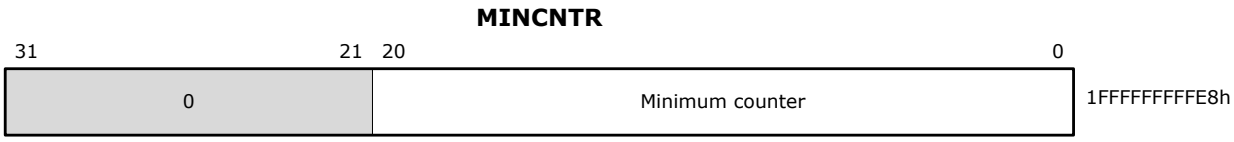
This register is used only in multi-threaded cores and is intended to stop the execution of a process whose PSO selector is written to the CLSR register. In multithreaded cores, situations are possible when a system process removes another process from execution, which is currently executed by a parallel thread. To correctly stop the process, use the CLSR register. Writing to this register causes the process of searching and stopping the specified process in the context controller, and if such a process is detected, its current context is saved in the PSO, and the thread is released to perform any other processes.

AVCNTR. Average counter



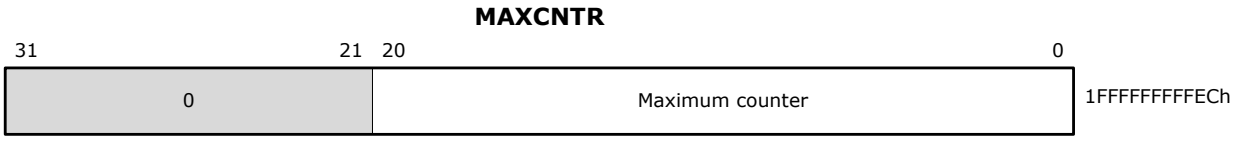
The counter contains the current value of the number of instructions executed over a specified period of time.

MINCNTR. Minimum counter



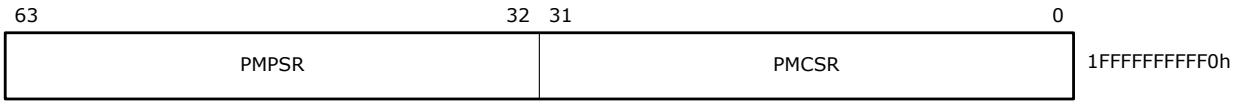
The minimum value of the number of instructions counted for a specified time interval.

MAXCNTR. Maximum counter



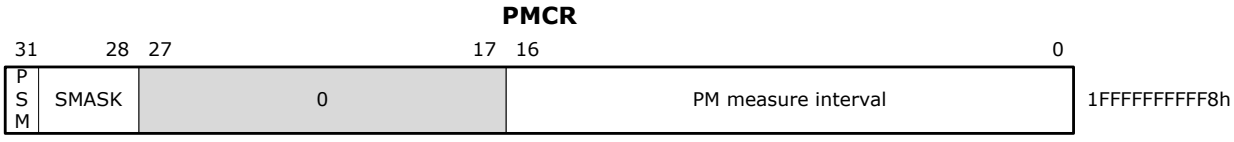
The maximum value of the instructions counter calculated for the specified time interval.

PMCSR, PMPSR. Performance monitor selector registers



Performance monitor code selector register and Performance monitor process selector register. Registers are used to tune the performance monitor to analyze processor performance when executing a specific process and/or a specific code object. If the PMCSR register is 0, then the performance monitor calculates the number of instructions executed regardless of the code object selector. If the PMPSR register is 0, then the performance monitor calculates the number of instructions executed, regardless of the process currently running. The presence of two filtering parameters — the code object selector and the process selector — allows, for example, to evaluate the performance of a separate procedure performed in the context of a specific process.

PMCR. Performance monitor control register



The performance monitor control register contains the value of the measurement interval, expressed in core clock cycles, three bits of the process status mask, allowing you to select at which process state the performance measurement will be performed, as well as the bit allowing the process switching time to be taken into account when measuring performance. With PSM = 1, the performance monitor will take into account the cycles during which context switching is performed. If SMASK is not 0, then performance will be calculated only when the SMASK field is equal to the value of the STATE field of the CSR register.