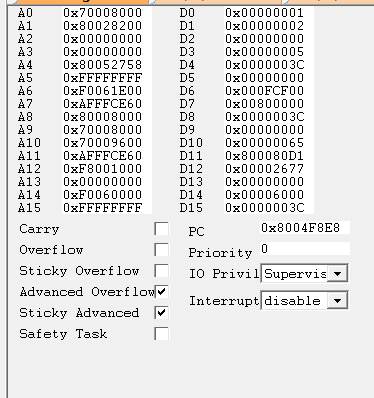
At Main

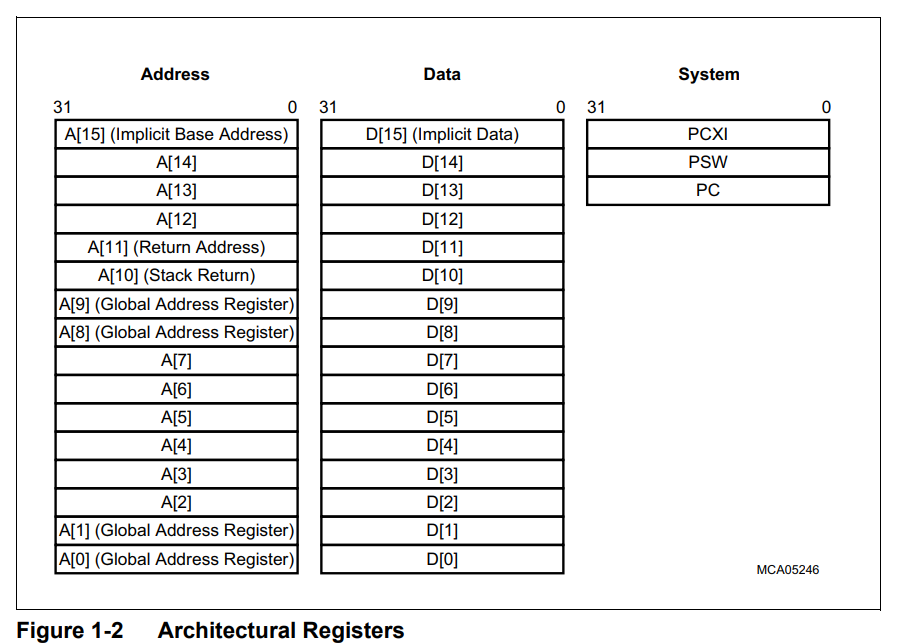


15 u

100 u

1 m

10 m



These are the GPR registers

16 Address – A0 to A15

16 Data – D0 to D15

Two consecutive even-odd data registers can be concatenated to form eight extended-size registers (E[0], E[2], E[4], E[6], E[8], E[10], E[12], and E[14]), in order to support 64-bit values. The address registers (P[0], P[2], P[4], P[6], P[8], P[10], P[12], and P[14]) can be used in the same way.

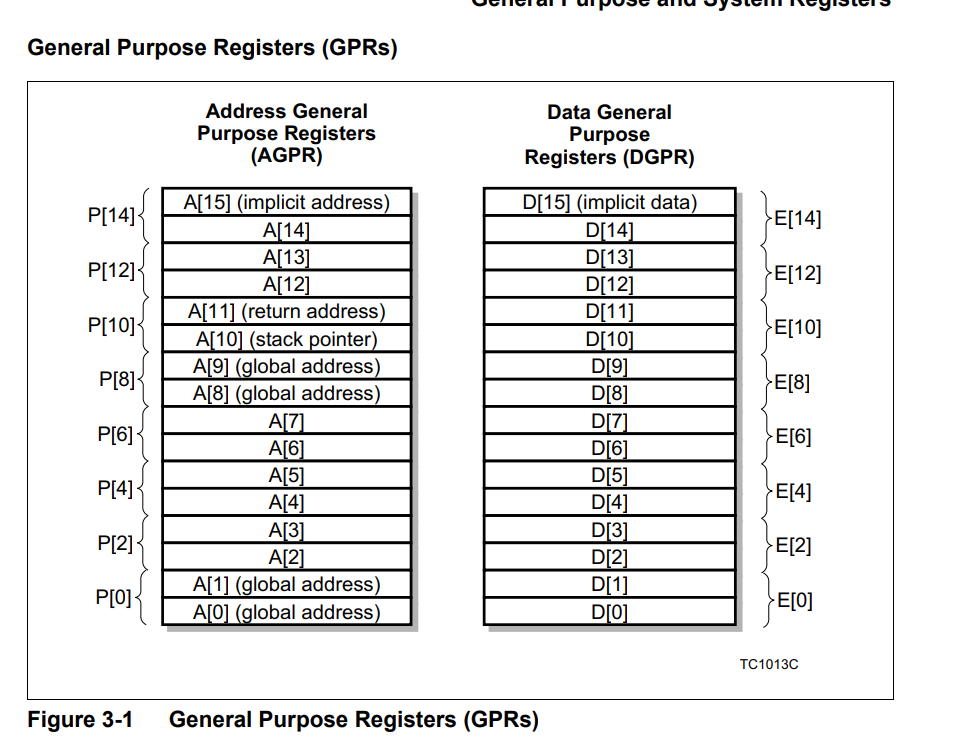
For extended data and extended addressing we use 64 bit by concatenating odd and even registers as mentioned above.

Registers A[0], A[1], A[8], and A[9] are defined as system global registers. Their contents are not saved or restored across calls, traps or interrupts.

Register A[10] is used as the Stack Pointer (SP)

Register A[11] is used to store the Return Address (RA) for calls and linked jumps, and to store the return Program Counter (PC) value for interrupts and traps

While the 32-bit instructions have unlimited use of the GPRs, many 16-bit instructions implicitly use A[15] as their address register and D[15] as their data register. This implicit use eases the encoding of these instructions into 16 bits.



The GPRs are an essential part of a task’s context. When saving or restoring a task’s context to and from memory the context is split into the upper and lower contexts:

• Registers A[2] to A[7] and D[0] to D[7] are part of the **lower context**

• Registers A[10] to A[15] and D[8] to D[15] are part of the **upper context**

**Program State Information Registers**

**Program Counter (PC)**

Holds the address of the instruction that is currently running

The PC should only be written when the core is halted. If the core is not in halt a write will have no effect.

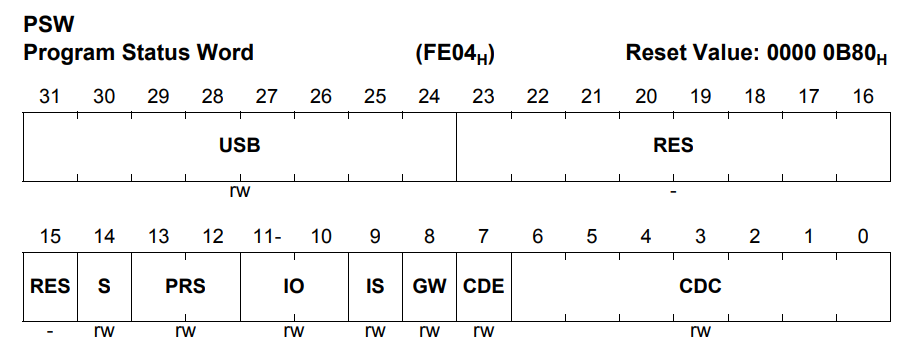
**Program Status Word**

**PSW.Reg = 0x00000980**

**……000000100110000000**

**Here by default the PSW register is given this value in code**

**The bit 11 and 10 indicate Bin (10) ie 2 which denote supervisor mode**



14 S safety task identifier

13 12 PRS Protection register set

Selects the active Data and Code Memory Protection Register Set. The memory protection register values control load, store and instruction fetches within the current process.

00B : Protection Register Set 0

01B : Protection Register Set 1

10B : Protection Register Set 2

11B : Protection Register Set 3

**11 AND 12 Bit (IO)**

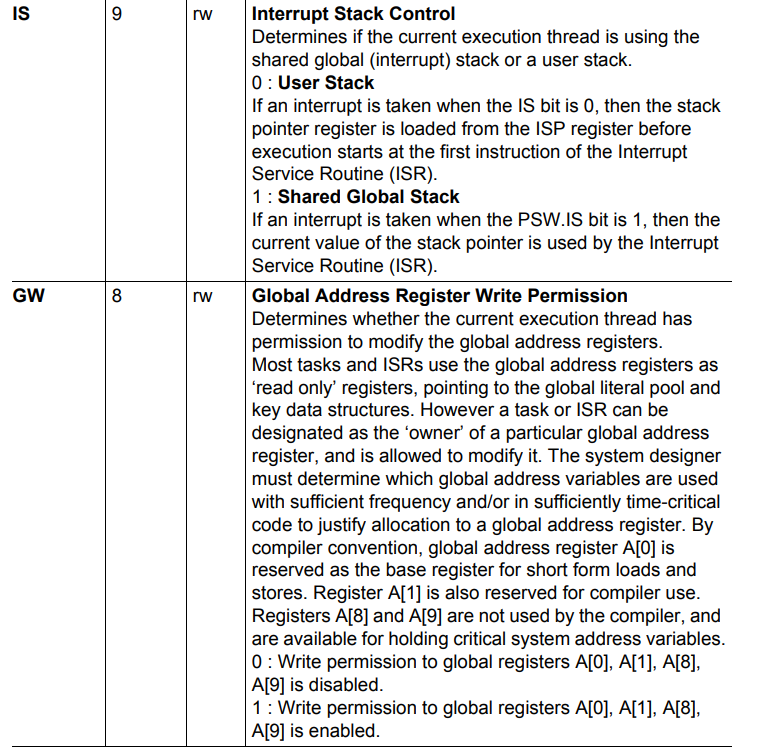
Access Privilege Level Control (I/O Privilege) Determines the access level to special function registers and peripheral devices.

00B : User-0 Mode No peripheral access. Access to memory regions with the peripheral space attribute are prohibited and results in a PSE or MPP trap. This access level is given to tasks that need not directly access peripheral devices. Tasks at this level do not have permission to enable or disable interrupts.

01B : User-1 Mode Regular peripheral access. Enables access to common peripheral devices that are not specially protected, including read/write access to serial I/O ports, read access to timers, and access to most I/O status registers. Tasks at this level may disable interrupts. (The default behavior of this mode may be overridden by the system control register).

10B : Supervisor Mode Enables access to all peripheral devices. It enables read/write access to core registers and protected peripheral devices. Tasks at this level may disable interrupts.

11B : Reserved Value



More bit filed details in Architectural Doc Chap 3 PSW section

**PCXI Previous Context Info contains the link to CSA**

More bit filed details of the register in Architectural Doc Chap 3 PCXI section

**Stack Management**

Stack management in the architecture supports a user stack and an interrupt stack.

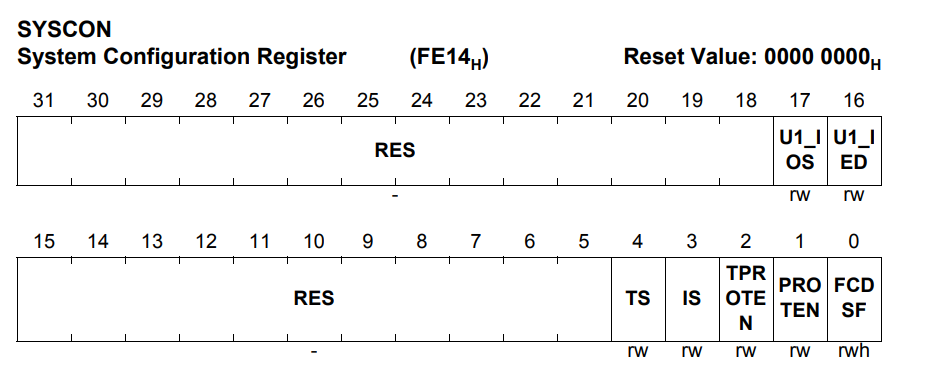
Address register A[10], the Interrupt Stack Pointer (ISP) and a PSW bit are used in the management of the stack.

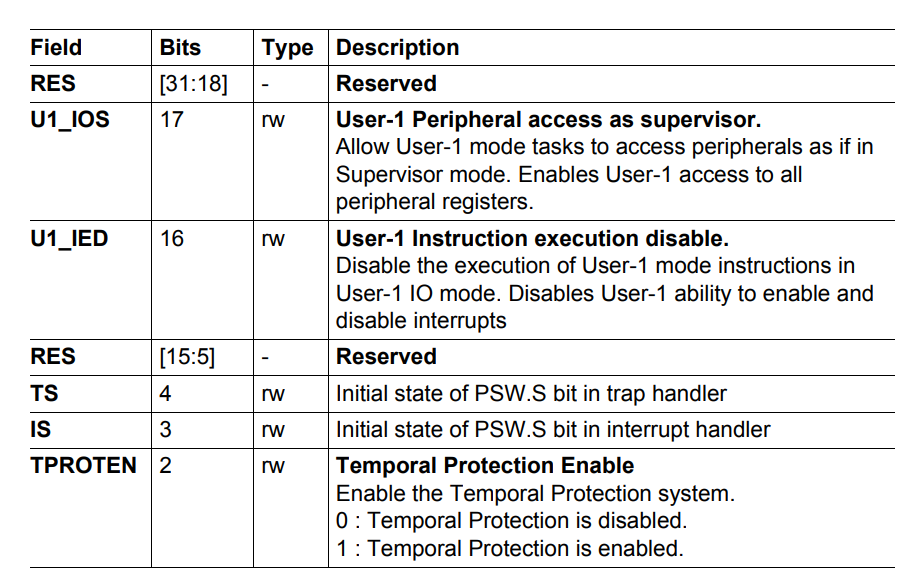
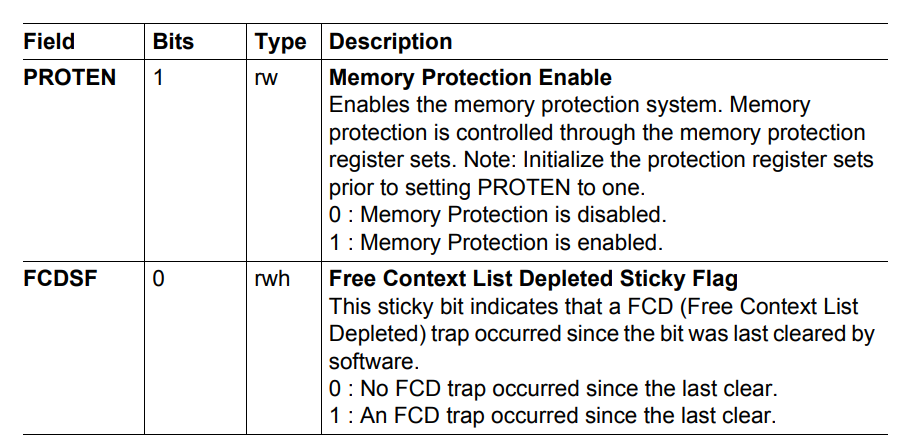
The PSW.IS bit indicates which stack pointer is in effect whether Private (User stack) or Interrupt Stack

When an interrupt is taken and the interrupted task was using its private stack (PSW.IS == 0), the contents are saved with the upper context of the interrupted task and A[10](SP) is loaded with the current contents of the ISP.

When an interrupt or trap is taken and the interrupted task was already using the interrupt stack (PSW.IS == 1), then no pre-loading of A[10](SP) is performed. The Interrupt Service Routine (ISR) continues to use the interrupt stack at the point where the interrupted routine had left it

**System Control Register (SYSCON)**

****

**** ****

Interrupt Control Registers

Two CSFRs support interrupt handling:

• ICR: Interrupt Control Register

• BIV: Base Interrupt Vector Table Pointer

The ICR holds the Current CPU Priority Number (CCPN), the enable/disable bit for the Interrupt System (IE), the Pending Interrupt Priority Number (PIPN), and an implementation specific control for the interrupt arbitration scheme. The BIV register holds the base addresses for the interrupt vector tables. Special instructions control the enabling and disabling of the interrupt system.

**Memory Protection**

Memory Protection Provides control over which regions of memory a task is allowed to access, and what types of access is permitted.

• Range Based The range-based memory protection system is designed for small and low cost applications to provide coarse-grained memory protection for systems that do not require virtual memory.

**Protection Ranges**

A Protection Range is a continuous part of address space for which access permissions may be specified. A Protection Range is defined by the Lower Boundary and the Upper Boundary.

An address belongs to the range if:

• Lower Boundary <= Address < Upper Boundary

There are two groups of Protection Ranges:

• Data Protection Ranges specify data access permissions

• Code Protection Ranges specify instruction fetch permissions