- Where is the Vector table stored (flash or ram)?

====> The static interrupt vector table stores in flash/code memory

====>The dynamic interrupt vector table stores in RAM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Difference between direct function call & ISR generated by swi:

===> The main difference between a function and a software interrupt is what is known as context.

A function runs within the context of your main program.

An interrupt runs within the context of the interrupt handler.

On a simple system this may be no real difference,

and software interrupts may simply be used as a convenient way of providing library routines hard coded in ROM

you don't need to know the address of every routine, only the ID code and the main entry point.

This makes your code more portable.

However,on more complex systems the software interrupt may run in a completely different environment

known as the kernel context.

Normally your application would run in a protected user context which has limited access to resources.

Only when running in the kernel context can you perform the more complicated tasks

indeed some systems even limit which instructions can be executed,

so you need a mechanism to trigger code in the kernel context - and for that an interrupt is used.