

## ISR PREREQUISITES

1. Sufficient stack allocated and stack pointer initialized
  - a. ARM7 (done in *aduc7026.s*) - since user mode and each exception mode has its own stack, our initialization needs to take care of all mode's stack pointers (i.e. R13\_irq/R13\_fiq)
2. Interrupt vector location initialized or ISR located at correct location
  - a. ARM7 (done in *aduc7026.s*) – branch instruction placed at IRQ/FIQ exception address.
3. Configure any peripherals or external hardware used by the ISR.
4. Initialize any global variables used by ISR.
5. Interrupt source configured and enabled.
6. Clear any pending interrupts.
7. Interrupt request from device unmasked
  - a. ADuC7026 - interrupt request routed to FIQ or IRQ mode as desired.
8. Global interrupts enabled
  - a. ARM7 - CPSR I and/or F bit set to 0.

## ISR IMPLEMENTATION

1. Context save
  - a. ISR must not disturb any processor state.
  - b. Many microprocessors will automatically save the necessary processor context on the stack.
  - c. ARM7 - save LR if doing any subroutine calls, save any other registers that are used. (In FIQ, don't need to save R8-R12.)
2. Clear interrupt request from source.
  - a. ARM7 – you must clear the interrupt request at the source.
  - b. In processors with vectored interrupts, the interrupt request is often cleared automatically by the hardware.
3. Allow nesting if desired. (You should have a very good reason for doing this!)
  - a. ARM7 - save SPSR, LR and clear CPSR I/F bits to allow nested interrupts. If nesting FIQ mode interrupts, also need to save R8-R12 if used by the ISR.
4. Perform required processing.
5. Context restore.
  - a. Restore any registers that the ISR saved.
6. Return from interrupt
  - a. Many processors have a special return instruction to exit an ISR – the instruction needs to undo whatever context save the hardware did on entry to the ISR.
  - b. ARM7 – to return from FIQ or IRQ mode, use *SUBS PC, LR, #4*.