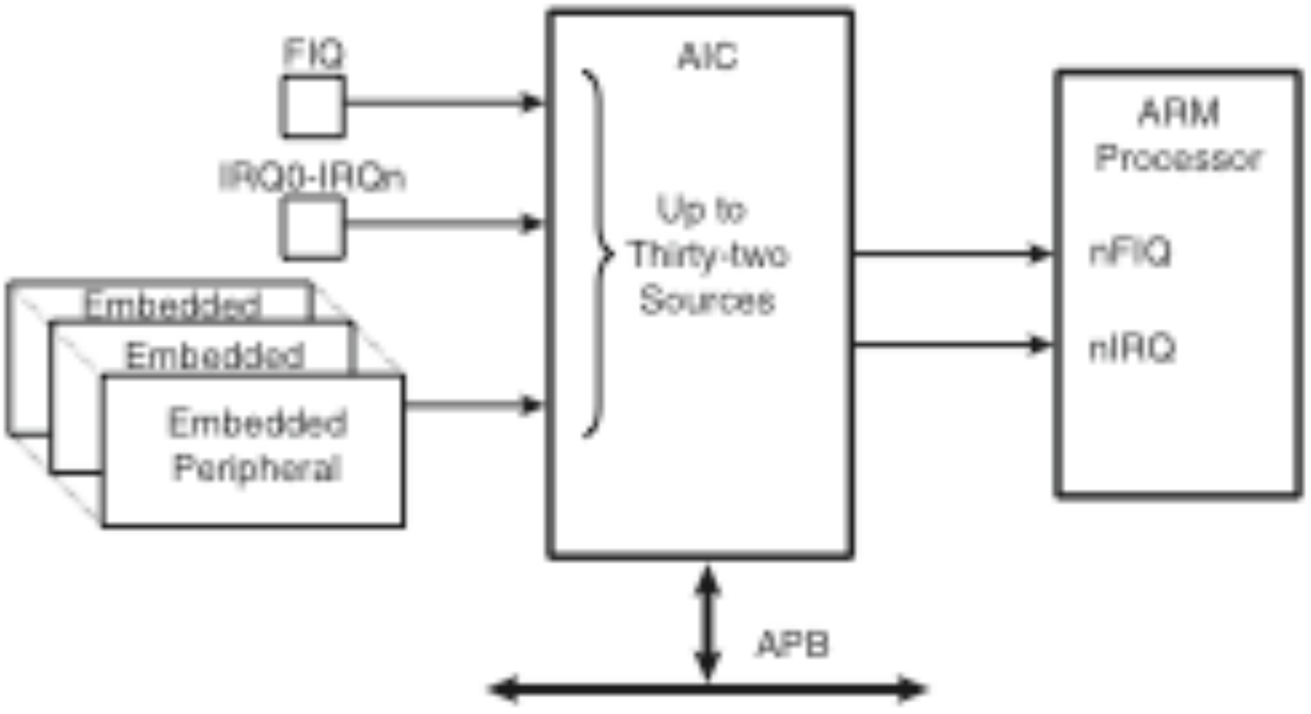
HY428 – Lecture 2
- Advanced Interrupt Controller (AIC)
- ARM Interrupts

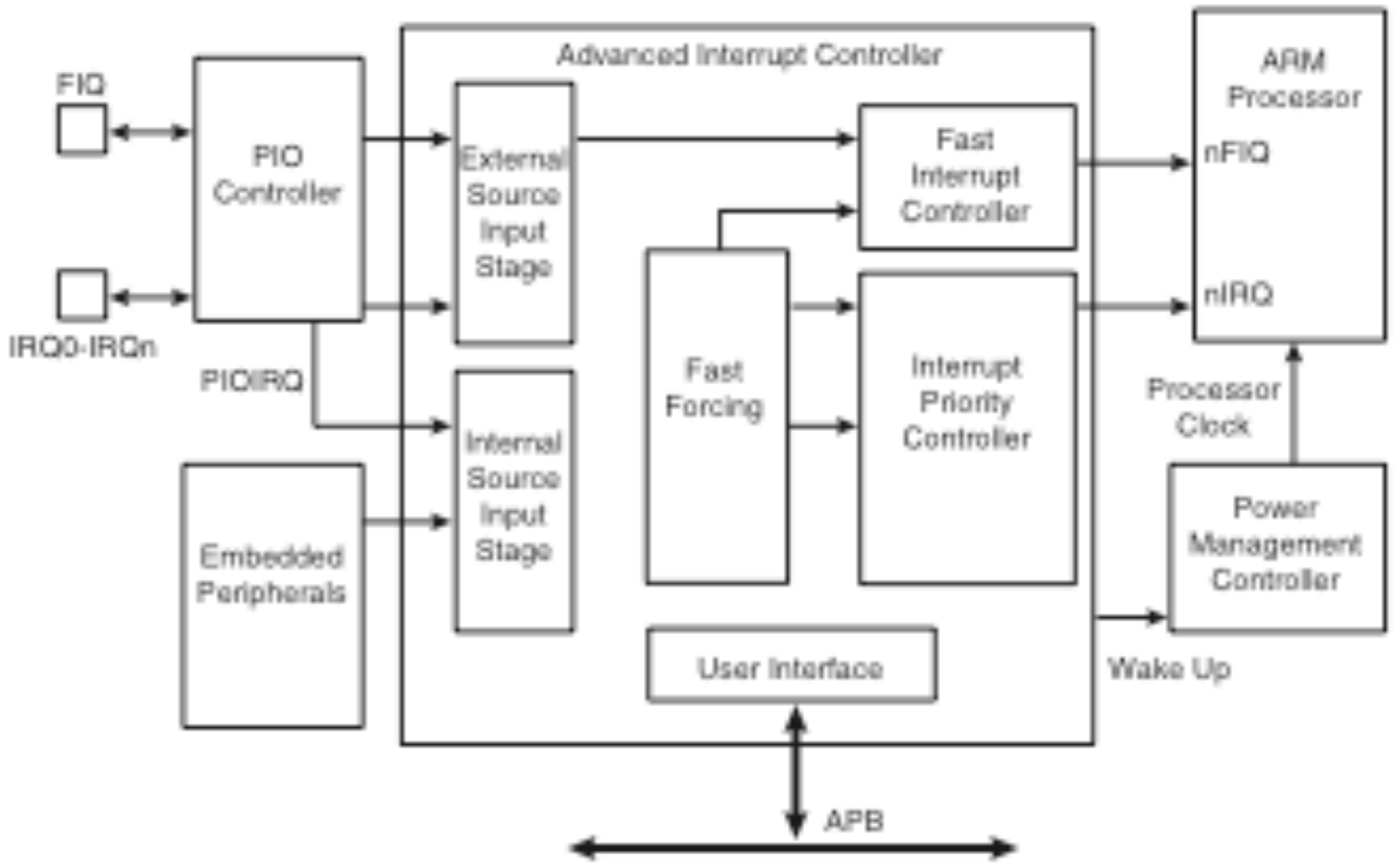
Advanced Interrupt Controller

Figure 24-1. Block Diagram



(Manual: Page 183)

Figure 24-3. AIC Detailed Block Diagram


FIQ
IRQ0-IRQn
PIO Controller
PIOIRQ
Embedded Peripherals
Advanced Interrupt Controller
External Source Input Stage
Internal Source Input Stage
Fast Forcing
Fast Interrupt Controller
Interrupt Priority Controller
User Interface
APB
ARM Processor
nFIQ
nIRQ
Processor Clock
Power Management Controller
Wake Up

(Manual: Page 184)

Table 24-2. Register Mapping

Offset	Register	Name	Access	Reset Value
0000	Source Mode Register 0	AIC_SMR0	Read/Write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read/Write	0x0
---	---	---	---	---
0x7C	Source Mode Register 31	AIC_SMR31	Read/Write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read/Write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read/Write	0x0
---	---	---	---	---
0xFC	Source Vector Register 31	AIC_SVR31	Read/Write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	FIQ Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register[2]	AIC_IPR	Read-only	0x0[1]
0x110	Interrupt Mask Register[2]	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118	Reserved	---	---	---
0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register[2]	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register[2]	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register[2]	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register[2]	AIC_ISCR	Write-only	---
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	---
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read/Write	0x0
0x138	Debug Control Register	AIC_DCR	Read/Write	0x0
0x13C	Reserved	---	---	---
0x140	Fast Forcing Enable Register[2]	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register[2]	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register[2]	AIC_FFSR	Read-only	0x0

Notes: 1. The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.

2. PID2...PID31 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.

24.8.3 AIC Source Mode Register

Register Name: AIC_SMR0..AIC_SMR31

Access Type: Read/Write

Reset Value: 0x0

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
–	SRCTYPE		–	–	PRIOR		

- **PRIOR: Priority Level**

Programs the priority level for all sources except FIQ source (source 0).

The priority level can be between 0 (lowest) and 7 (highest).

The priority level is not used for the FIQ in the related SMR register AIC_SMRx.

- **SRCTYPE: Interrupt Source Type**

The active level or edge is not programmable for the internal interrupt sources.

SRCTYPE		Internal Interrupt Sources	External Interrupt Sources
0	0	High level Sensitive	Low level Sensitive
0	1	Positive edge triggered	Negative edge triggered
1	0	High level Sensitive	High level Sensitive
1	1	Positive edge triggered	Positive edge triggered

24.8.4 AIC Source Vector Register

Register Name: AIC_SVR0..AIC_SVR31

Access Type: Read/Write

Reset Value: 0x0

31	30	29	28	27	26	25	24
VECTOR							

23	22	21	20	19	18	17	16
VECTOR							

15	14	13	12	11	10	9	8
VECTOR							

7	6	5	4	3	2	1	0
VECTOR							

- **VECTOR: Source Vector**

The user may store in these registers the addresses of the corresponding handler for each interrupt source.

24.8.5 AIC Interrupt Vector Register

Register Name: AIC_IVR

Access Type: Read-only

Reset Value: 0x0

31	30	29	28	27	26	25	24
IRQV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
IRQV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IRQV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IRQV							

- **IRQV: Interrupt Vector Register**

The Interrupt Vector Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt.

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the Interrupt Vector Register reads the value stored in AIC_SPU.

24.8.7 AIC Interrupt Status Register

Register Name: AIC_ISR

Access Type: Read-only

Reset Value: 0x0

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
–	–	–			IRQID		

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

24.8.8 AIC Interrupt Pending Register

Register Name:	AIC_IPR
Access Type:	Read-only
Reset Value:	0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Pending**

0 = Corresponding interrupt is not pending.

1 = Corresponding interrupt is pending.

24.8.9 AIC Interrupt Mask Register

Register Name:	AIC_IMR
Access Type:	Read-only
Reset Value:	0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Mask**

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

24.8.10 AIC Core Interrupt Status Register

Register Name: AIC_CISR

Access Type: Read-only

Reset Value: 0x0

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**
–	–	–	–	–	–	NIRQ	NIFQ

- **NFIQ: NFIQ Status**

0 = nFIQ line is deactivated.

1 = nFIQ line is active.

- **NIRQ: NIRQ Status**

0 = nIRQ line is deactivated.

1 = nIRQ line is active.

24.8.11 AIC Interrupt Enable Command Register

Register Name: AIC_IECR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID3: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

24.8.12 AIC Interrupt Disable Command Register

Register Name: AIC_IDCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

24.8.13 AIC Interrupt Clear Command Register

Register Name: AIC_ICCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

24.8.14 AIC Interrupt Set Command Register

Register Name: AIC_ISCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

24.8.15 AIC End of Interrupt Command Register

Register Name: AIC_EOICR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

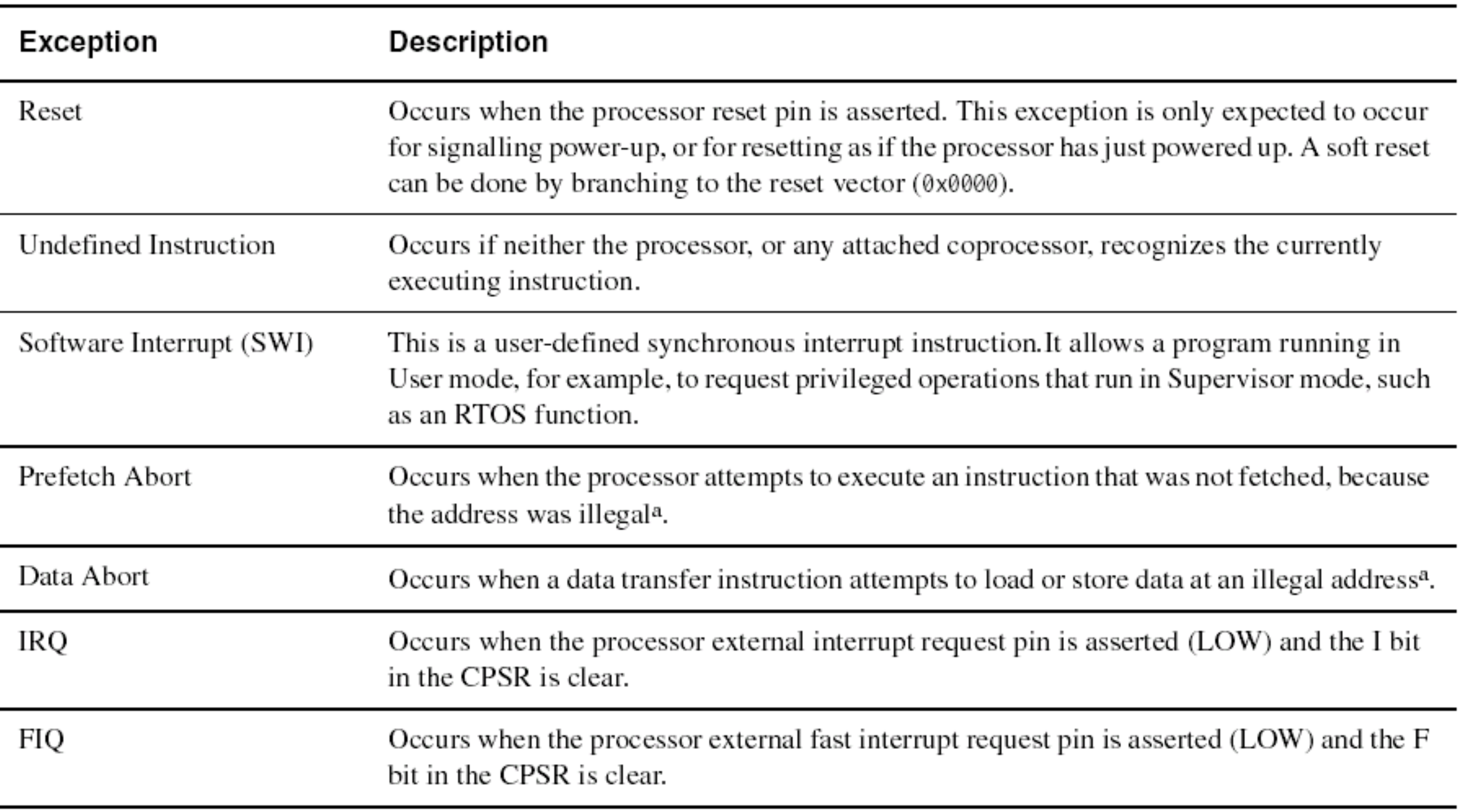
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

The End of Interrupt Command Register is used by the interrupt routine to indicate that the interrupt treatment is complete. Any value can be written because it is only necessary to make a write to this register location to signal the end of interrupt treatment.

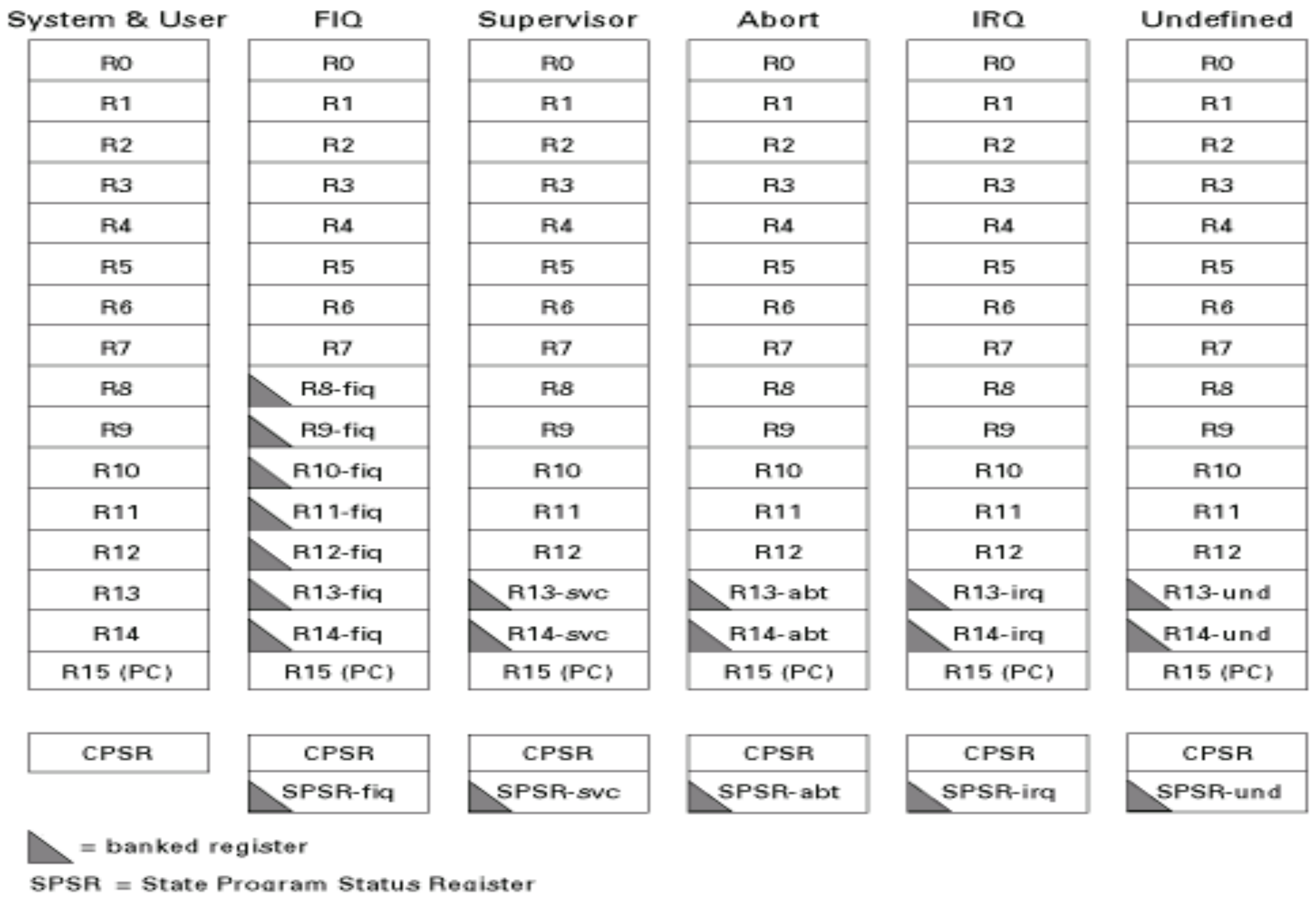
ARM Interrupts

ARM Exceptions



Exception	Description
Reset	Occurs when the processor reset pin is asserted. This exception is only expected to occur for signalling power-up, or for resetting as if the processor has just powered up. A soft reset can be done by branching to the reset vector (`0x0000`).
Undefined Instruction	Occurs if neither the processor, or any attached coprocessor, recognizes the currently executing instruction.
Software Interrupt (SWI)	This is a user-defined synchronous interrupt instruction.It allows a program running in User mode, for example, to request privileged operations that run in Supervisor mode, such as an RTOS function.
Prefetch Abort	Occurs when the processor attempts to execute an instruction that was not fetched, because the address was illegal[a].
Data Abort	Occurs when a data transfer instruction attempts to load or store data at an illegal address[a].
IRQ	Occurs when the processor external interrupt request pin is asserted (LOW) and the I bit in the CPSR is clear.
FIQ	Occurs when the processor external fast interrupt request pin is asserted (LOW) and the F bit in the CPSR is clear.

ARM Registers

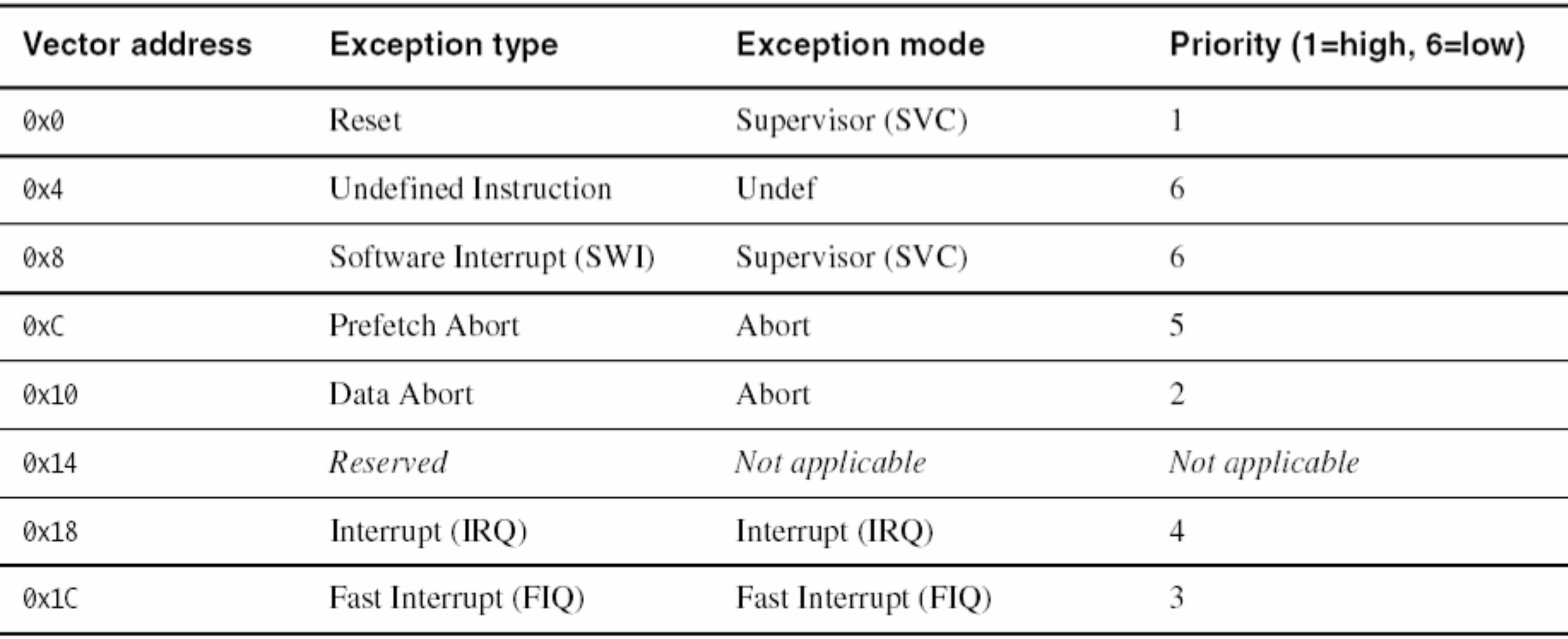


System & User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	◣ R8-fiq	R8	R8	R8	R8
R9	◣ R9-fiq	R9	R9	R9	R9
R10	◣ R10-fiq	R10	R10	R10	R10
R11	◣ R11-fiq	R11	R11	R11	R11
R12	◣ R12-fiq	R12	R12	R12	R12
R13	◣ R13-fiq	◣ R13-svc	◣ R13-abt	◣ R13-irq	◣ R13-und
R14	◣ R14-fiq	◣ R14-svc	◣ R14-abt	◣ R14-irq	◣ R14-und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	◣ SPSR-fiq	◣ SPSR-svc	◣ SPSR-abt	◣ SPSR-irq	◣ SPSR-und

◣ = banked register

SPSR = State Program Status Register

ARM (not AIC!) Priorities



Vector address	Exception type	Exception mode	Priority (1=high, 6=low)
`0x0`	Reset	Supervisor (SVC)	1
`0x4`	Undefined Instruction	Undef	6
`0x8`	Software Interrupt (SWI)	Supervisor (SVC)	6
`0xC`	Prefetch Abort	Abort	5
`0x10`	Data Abort	Abort	2
`0x14`	*Reserved*	*Not applicable*	*Not applicable*
`0x18`	Interrupt (IRQ)	Interrupt (IRQ)	4
`0x1C`	Fast Interrupt (FIQ)	Fast Interrupt (FIQ)	3

ARM Processor Modes (partial)

- *User*: the “normal” program execution mode.
- *IRQ*: used for general-purpose interrupt handling.
- *Supervisor*: a protected mode for the operating system.
  - (there are also Abort, FIQ, SYS, and Undef modes)

The ARM Register Set

- Registers R0-R15 + CPSR (Current Program Status Register)
  - R13: Stack Pointer (by convention)
  - R14: Link Register (hardwired)
  - R15: Program Counter (bits 0:1 ignored, hardwired)

Enabling IRQ in CPSR

31	30	29	28	27	…	8	7	6	5	4	3	2	1	0
N	Z	C	V				I	F		M4	M3	M2	M1	M0

- (Current) Program Status Register
- To disable interrupts, set corresponding “I” bit to 1
- To change the mode of ARM (IRQ, USR, SYS, …) change Mx bits in CPSR using special instructions
  - MSR CPSR/SPSR, r #move register to PSR
  - MRS r, CPSR/SPSR #move PSR to register

IRQ_ENTRY Macro Definition

```
            MACRO
            IRQ_ENTRY $reg
      ;- Adjust and save LR of current mode in current stack
                  sub                 r14, r14, #4
                  stmfd               sp!, {r14}
      ;- Save SPSR and r0 in current stack
                  mrs                 r14, SPSR
                  stmfd               sp!, {r0, r14}
      ;- Read Modify Write the CPSR to Enable the Core Interrupt
      ;- and Switch in SYS Mode ( same LR and stack than USR Mode )
                  mrs                 r14, CPSR
                  bic                 r14, r14, #I_BIT
                  orr                 r14, r14, #ARM_MODE_SYS
                  msr                 CPSR, r14
      ;- Save used registers and LR_usr in the System/User Stack
                  stmfd               sp!, {r1-r3, $reg, r12, r14}
            MEND
```

The parameter “$reg” allow the list of the registers used by the interrupt treatment to be pushed on the SYSTEM/USE stack by using the instruction which pushes R14_User. This list must be the same for the IRQ_EXIT call.

Note that in this application note, all registers defined as “scratched” by APCS (r0, r1, r2, r3, r12) are saved by IRQ_ENTR and restored by IRQ_EXIT.

IRQ_EXIT Macro Definition

```
            MACRO
            IRQ_EXIT    $reg,
      ;- Restore used registers and LR_usr from the System/User Stack
                  ldmfd             sp!, {r1-r3, $reg, r12, r14}
      ;- Read Modify Write the CPSR to disable interrupts
      ;- and to go back in the mode corresponding to the exception
                  mrs               r0, CPSR
                  bic               r0, r0, #ARM_MODE_SYS
                  orr               r0, r0, #I_BIT:OR:ARM_MODE_IRQ
                  msr               CPSR, r0

      ;- Mark the End of Interrupt on the interrupt controller
                  ldr               r0, = AIC_BASE
                  str               r0, [r0, #AIC_EOICR]
      ;- Restore SPSR_irq and r0 from the IRQ stack
                  ldmfd             sp!, {r0, r14}
                  msr               SPSR, r14
      ;- Restore ajusted LR_irq from IRQ stack directly in the PC
                  ldmfd             sp!, {pc}^
            MEND
```

The IRQ Stack

The IRQ stack pointer (R13_irq) must be initialized at the top (upper address) of a reserved space. The size needed for this stack is 12 bytes (3 words for registers r0, r14 and SPSR) per level used in the application. If all levels are used, the stack space must be 96 bytes.

Examine Cstartup.s in pdf

Examine Cstartup_SAM7.c in pdf