

ARM and LPC2292 Operating Modes & Interrupt Handling

From ARM and LPC2292 references.

ARM operating modes

- **User** : unprivileged mode under which most tasks run
- **FIQ** : entered when a high priority (fast) interrupt is raised
- **IRQ** : entered when a low priority (normal) interrupt is raised
- **Supervisor** : entered on reset and when a Software Interrupt instruction (SWI) is executed
- **Abort** : used to handle memory access violations
- **Undef** : used to handle undefined instructions
- **System** : privileged mode using the same registers as user mode

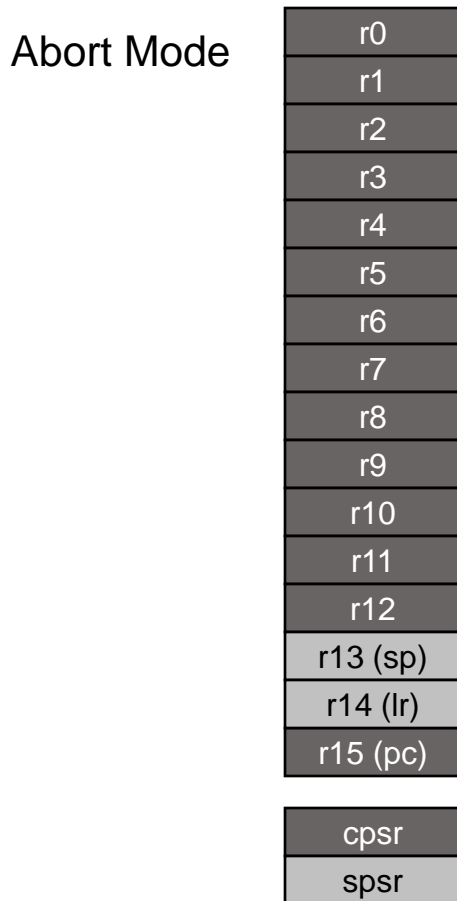
ARM Operating Modes

CPSR[4:0]	Mode	Use	Registers	Vector Address
10000	User	Normal user code	User	0x00000000
10001	FIQ	Fast interrupt	_fiq	0x00000004
10010	IRQ	Standard interrupt	_irq	0x00000008
10011	SVC	Software interrupt	_svc	0x0000000C
10111	Abort	Memory faults	_abt	0x00000010
11011	Undef	Undefined instruction	_und	0x00000018
11111	System	Privileged O/S task	user	0x0000001C

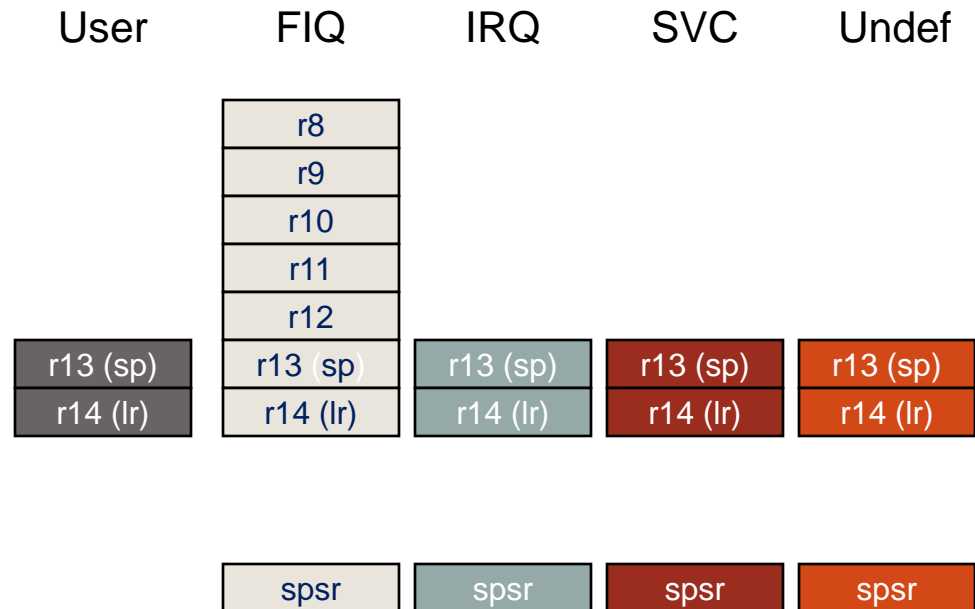
CPSR = Current Processor Status Register

The ARM Register Set

Current Visible Registers



Banked out Registers



ARM interrupts

- ARM7 “core” supports two types of interrupts:
 - **Fast interrupt requests (FIQs)**
 - User registers R0-R7, R15/pc
 - Shadow registers R8_fiq – R14_fiq
 - **Interrupt requests (IRQs)**
 - User registers R0-R12, R15/pc
 - Shadow registers R13_irq, R14_irq
- Interrupt table starts at location 0 and contains subroutine calls to handlers

ARM interrupt vector table

Table 6-1. ARM7 Exception Table

Vector Address	Interrupt Source	Interrupt Type
0x0000 0000	RESET	RESET
0x0000 0004	UNDEFINED INSTRUCTION	Undef' Instr'
0x0000 0008	SWI	S/W Int
0x0000 000C	ABORT (Prefetch)	ABORT
0x0000 0010	ABORT (Data)	ABORT
0x0000 0018	IRQ	Normal Interrupt
0x0000 001C	FIQ	Fast Interrupt

- “Exception” condition changes operating mode
- ARM CPU defines only 7 vector addresses
- Any additional distinction of interrupt sources must be done in software, working with external hardware

LPC2292 – Vector table setup

```
AREA RESET, CODE, READONLY
```

```
; Exception Vectors - Mapped to Address 0 (must use absolute addressing mode )
```

```
Vectors    LDR    PC, Reset_Addr
           LDR    PC, Undef_Addr
           LDR    PC, SWI_Addr
           LDR    PC, PAbt_Addr
           LDR    PC, DAbt_Addr
           NOP                                ; Reserved Vector
;          LDR    PC, IRQ_Addr                ;If not using Vectored Interrupt Controller
           LDR    PC, [PC, #-0x0FF0]         ; Vector from VicVectAddr
           LDR    PC, FIQ_Addr
```

```
;List of handler addresses – loaded by LDR instructions above
```

```
Reset_Addr DCD    Reset_Handler
Undef_Addr DCD    Undef_Handler
SWI_Addr   DCD    SWI_Handler
PAbt_Addr  DCD    PAbt_Handler
DAbt_Addr  DCD    DAbt_Handler
           DCD    0                                ; Reserved Address
IRQ_Addr   DCD    IRQ_Handler
FIQ_Addr   DCD    FIQ_Handler
```

ARM7 Interrupt Exception Summary

Register	FIQ	IRQ
Interrupt Sample	Detect FIQ asserted	Detect FIQ negated, IRQ asserted
Link Register	R14_fiq = nextInst + 4	R14_irq = nextInst + 4
Saved Status Register	SPSR_fiq = CPSR	SPSR_irq = CPSR
Current Processor Status Register	CPSR[M] = 5'b10001, CPSR[T] = 1'b0, CPSR[F] = 1'b1, CPSR[I] = 1'b1	CPSR[M] = 5'b10010, CPSR[T] = 1'b0, CPSR[F] = unaffected, CPSR[I] = 1'b1
Stack Pointer	Activate R13_fiq	Activate R13_irq
Other Banked Registers	Activate R[8-12]_fiq	None
Program Counter	PC = 0x0000_001C	PC = 0x0000_0018

- **CPU actions:**
 - Save PC. Copy CPSR to SPSR (**Saved PSR**)
 - Activate shadow registers
 - Force bits in CPSR to record interrupt.
 - Force PC to vector.
- **Handler responsibilities:**
 - Restore proper PC.
 - Restore CPSR from SPSR.
 - Clear interrupt disable flags.

Supervisor mode

- Use **supervisor mode** to manage the various programs.
 - Provide protective barriers between programs.
 - Avoid memory corruption.
 - Control access to I/O
 - Provide operating system functions
- ARM: Use SWI (software interrupt) instruction to enter supervisor mode, similar to subroutine:
SWI CODE_1
 - Sets PC to 0x08.
 - Argument to SWI is passed to supervisor mode code.
 - Saves CPSR in SPSR.

Using SWI in C programs

- Function prototype for calling from C

```
int __swi(0) add (int i1, int i2);
```

Attribute Function name to use in C program



- Call via: `c = add(a,b);` //treat as normal function
- Function implementation

```
int __SWI_0 (int i1, int i2)  
{ return (i1 + i2); }
```

- Vector table (in startup.s):

```
DCD __SWI_0 ; SWI 0 Function Entry
```

SEE KEIL SWI EXAMPLE

LPC2xxx Vectored Interrupt Controller (VIC)

- 32 interrupt request inputs
- Controller assigns them to 3 categories:
 - **FIQ** (highest priority)
 - **Vectored IRQ** (mid priority)
 - 16 inputs can be assigned vectors
 - **Non-vectored IRQ** (low priority)
 - Share a single vector
- **Only one IRQ signal** (OR vectored & nonvectored)

LPC2xxx VIC operation

- Recognize active interrupt requests:
 - Requests individually masked
 - Requests designated as FIQ, IRQ (vectored & non-vectored)
- Generate one FIQ and/or one IRQ to CPU
- Prioritization and level masking (vectored IRQ):
 - Map IRQ into programmed interrupt level
 - See if active request level $>$ current level
 - Write ISR address of highest priority level into VIC address register
 - Software of IRQ ISR must load VIC address register into PC (CPU vectors through addr. 0x00000018)
- Software can read VIC status register to identify active interrupt requests

LPC2xxx
 vectored
 interrupt
 sources
 (32 “channels”)

Block	Flag(s)	VIC Channel #
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgCommTx	3
TIMER0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4
TIMER1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)	7
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8
I ² C	SI (state change)	9
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	10
SPI1	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	11
PLL	PLL Lock (PLOCK)	12
RTC	Counter Increment (RTCCIF) Alarm (RTCALF)	13
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15
System Control	External Interrupt 2 (EINT2)	16
System Control	External Interrupt 3 (EINT3)	17
A/D	A/D Converter	18

(continued on next)

LPC2xxx vectored interrupt sources

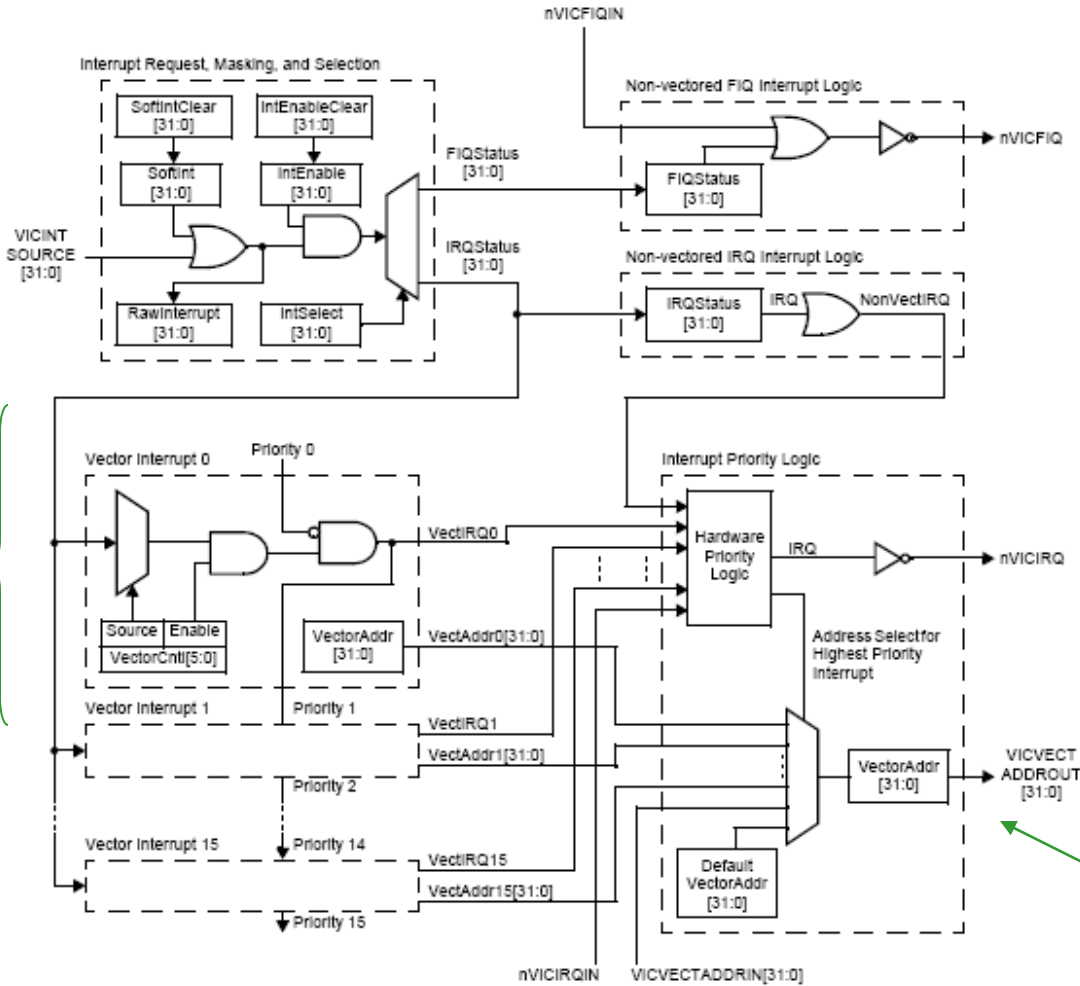
(continued)

Block	Flag(s)	VIC Channel #
CAN	CAN and Acceptance Filter (1 ORed CAN, LUTerr int)	19
	CAN1 Tx	20
	CAN2 Tx	21
	CAN3 Tx (LPC2194/2292/2294 only, otherwise Reserved)	22
	CAN4 Tx (LPC2194/2292/2294 only, otherwise Reserved)	23
	Reserved	24-25
	CAN1 Rx	26
	CAN2 Rx	27
	CAN3 Rx (LPC2194/2292/2294 only, otherwise Reserved)	28
	CAN4 Rx (LPC2194/2292/2294 only, otherwise Reserved)	29
	Reserved	30-31

LPC2xxx vectored interrupt controller (VIC)

Vectored Interrupt Sources

16 of these



To CPU:
FIQ request

IRQ request

Vector address

ISR should read this

LPC2xxx interrupt controller registers

Name	Description	Access	Reset Value*	Address
VICIRQStatus	IRQ Status Register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ.	RO	0	0xFFFF F000
VICFIQStatus	FIQ Status Requests. This register reads out the state of those interrupt requests that are enabled and classified as FIQ.	RO	0	0xFFFF F004
VICRawIntr	Raw Interrupt Status Register. This register reads out the state of the 32 interrupt requests / software interrupts, regardless of enabling or classification.	RO	0	0xFFFF F008
VICIntSelect	Interrupt Select Register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.	R/W	0	0xFFFF F00C
VICIntEnable	Interrupt Enable Register. This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ.	R/W	0	0xFFFF F010
VICIntEnClr	Interrupt Enable Clear Register. This register allows software to clear one or more bits in the Interrupt Enable register.	W	0	0xFFFF F014
VICSoftInt	Software Interrupt Register. The contents of this register are ORed with the 32 interrupt requests from various peripheral functions.	R/W	0	0xFFFF F018
VICSoftIntClear	Software Interrupt Clear Register. This register allows software to clear one or more bits in the Software Interrupt register.	W	0	0xFFFF F01C

LPC2xxx interrupt controller registers

VICProtection	Protection enable register. This register allows limiting access to the VIC registers by software running in privileged mode.	R/W	0	0xFFFF F020
VICVectAddr	Vector Address Register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.	R/W	0	0xFFFF F030
VICDefVectAddr	Default Vector Address Register. This register holds the address of the Interrupt Service routine (ISR) for non-vectorized IRQs.	R/W	0	0xFFFF F034
VICVectAddr0	Vector address 0 register. Vector Address Registers 0-15 hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.	R/W	0	0xFFFF F100
VICVectAddr1	Vector address 1 register	R/W	0	0xFFFF F104

VICVectAddr15	Vector address 15 register	R/W	0	0xFFFF F13C
VICVectCntl0	Vector control 0 register. Vector Control Registers 0-15 each control one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest.	R/W	0	0xFFFF F200
VICVectCntl1	Vector control 1 register	R/W	0	0xFFFF F204

VICVectCntl15	Vector control 15 register	R/W	0	0xFFFF F23C
---------------	----------------------------	-----	---	-------------

Interrupt setup example

Set up UART0 & SPI0 to generate vectored IRQs (UART0 higher level than SPI0), and UART1 and I2C to generate non-vectored IRQs

VICIntSelect = 0x0000 0000

(SPI0, I2C, UART1 and UART0 are IRQ => bit10, bit9, bit7 and bit6=0)

VICIntEnable = 0x0000 06C0

(SPI0, I2C, UART1 and UART0 enabled => bit10, bit9, bit 7 and bit6=1)

VICDefVectAddr = 0x...

(address of routine for servicing non-vectored IRQs (i.e. UART1 and I2C))

VICVectAddr0 = 0x... (address where UART0 IRQ service routine starts)

VICVectAddr1 = 0x... (address where SPI0 IRQ service routine starts)

VICVectCntl0 = 0x0000 0026

(int source with index 6 (UART0) enabled as priority 0 (the highest))

Bit 5 = enable, Bits 4-0 = slot 6

VICVectCntl1 = 0x0000 002A

(int source with index 10 (SPI0) enabled as priority 1)

After any IRQ request (SPI0, I2C, UART0 or UART1), microcontroller will redirect code execution to the address specified at location 0x00000018. For vectored and non-vectored IRQ's the following instruction could be placed at 0x18:

LDR pc,[pc,#-0x0FF0] (load address from VICVectAddr register.)

Returning from interrupt

- R14 (link register) holds return information
- IRQ and FIQ: R14 = next instruction $PC + 4$
 - Return from IRQ/FIQ: `subs pc,r14,#4`
- SWI: PC of next instruction in 14
 - Return from SWI: `movs pc,r14`
- Data abort R14 saves next $PC+8$
 - Return: `subs pc,r14,#8`
- Notes:
 - 's' modifier allows CPSR to be restored
 - Different saved values relates to pipeline architecture