Exception Handling and GPIO

1. Describe how Cortex M4 handle an exception, starting from the instance when the interrupt trigger occurs to the entrance into the ISR and return to interrupted routine. Include the key events, decision points and changes in processor modes.

   - Processor in thread mode when in user routine using process stack
   - Interrupt occur, processor enter into Handler mode. Perform context saving (PC, Processor Status Registers, r0-r3, R12, LR) and fetch interrupt vector from vector table to PC simultaneously.
   - Switch to main stack and execute Exception Handler routine.
   - After executing the ISR, processor returns to interrupted routine.
   - The processor mode and which stack is used after exception return depends on the value of the LR register.
     - 0xFFFFFFF1 Return to handler mode (use main stack)
     - 0xFFFFFFF9 Return to thread mode and use main stack
     - 0xFFFFFFFD Return to thread mode and use process stack

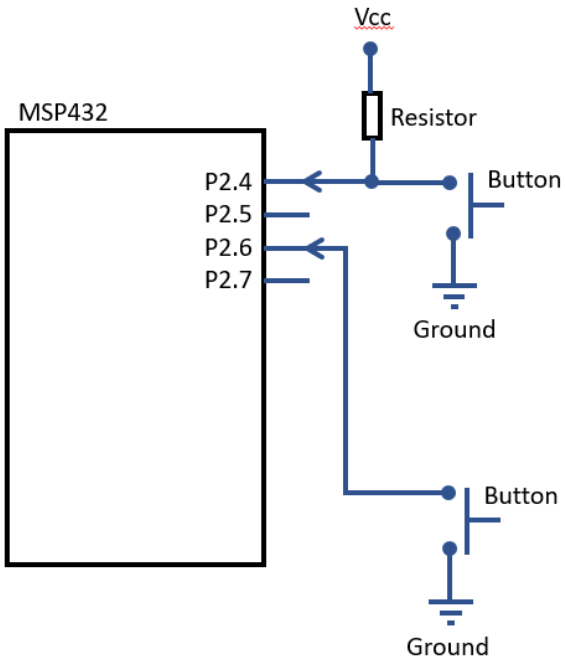
2. What is the optimizations ARM has introduced into ARM Cortex M3/M4 platform where exception handling is concerned? Compare to the older ARM processor such as ARM7/9 platforms.

   - Deterministic interrupt latency (12 cycles)
     - Exceptions handling from exception trigger up to exception handler fetch from vector table are all done in hardware so behavior is deterministic.
     - Compared to classical ARM7/9 processor where the IRQ vector fetch is done in software.
   - Tail Chaining (See Lecture Notes)
   - Late Arrival (See Lecture Notes)

3. The stacking operation done before entrance to exception handler only save r0-r3 and r12 registers. What should the user software do to handle the context saving/restore of other data registers?

- Need to look into the AAPCS (ARM Arch Procedure Call Standard) understand how the various data registers are used between main and sub-routines. Exception handler routines can be viewed as a sub-routine.
- In AAPCS, callee routine i.e. sub-routine is required to save r4-8 and r11, hence the exception service routine user develop should handle the context saving/restore of these registers.
- Use of r9 is platform specific but in general, there is a requirement the content of r9 has to be persistent across routines. Safest way is to not use r9 in exception service routines.
- If the exception service routines are coded in C and compiled with a compiler that is AAPCS compliant, the save/restore of the registers will be done automatically by the compiler.

4. The diagram below shows the schematic diagram of a MSP432 connecting to two user buttons on P2.4 and P2.6. The system is designed such that an interrupt will be triggered when the button is asserted.



Perform the necessary registers configuration for the GPIO and Timer peripherals

- GPIO
  - P2.4 and 2.6 port pins initialization
  - Interrupt configuration for Port 2.
  - Set priority of the interrupt to 2.
  - Only need to initialize the following registers

- For GPIO: PxOUT, PxDIR, PxREN, PxSEL0, PxSEL1, PxIES, PxIE and PxIFG.
- For interrupt: ISER1, ICPR1, IPR9. Why did we choose these registers for Port2?

The necessary documents have been extracted for you and can be found in the appendix of the tutorial.

- GPIO
  - Port 2.4 and P2.6 (input, internal pullup, falling edge triggered interrupt)
  - GPIO Input
    - P2SEL0 &= ~(0x50), P2SEL1 &= ~(0x50), P2DIR &= ~(0x50).
  - Internal Pullup only needed for P2.6. P2.4 has external pullup.
    - P2REN |= (0x40), P2OUT |= 0x40
  - Interrupt Enable, Clear Interrupt Pending Flag, falling edge triggered.
    - P2IE |= (0x50), P2IFG &= ~(0x50), P2IES |= 0x50

- NVIC
  - Port 2 interrupt. Location [36] in NVIC table, i.e. bit 4 of ISER1.
  - NVIC Interrupt enable
    - ISER1 |= (0x10),
  - Clear any pending bit
    - ICPR1 |= (0x10)
  - Set priority to 2 (only upper 3 bits are valid for MSP432)
    - IPR9 = (IPR9 & 0xFFFFFF00) | 0x00000040

Appendix

GPIO

Table 6-67. Port P2 (P2.4 to P2.7) Pin Functions

PIN NAME (P2.x)	x	FUNCTION	CONTROL BITS OR SIGNALS(1)			
			P2DIR.x	P2SEL1.x	P2SEL0.x	P2MAPx
P2.4/PM_TA0.1(2)	4	P2.4 (I/O)	I: 0; O: 1	0	0	X
		TA0.CCI1A	0	0	1	default
		TA0.1	1			
		N/A	0	1	0	X
		DVSS	1			
		N/A	0	1	1	X
		DVSS	1			
P2.5/PM_TA0.2(2)	5	P2.5 (I/O)	I: 0; O: 1	0	0	X
		TA0.CCI2A	0	0	1	default
		TA0.2	1			
		N/A	0	1	0	X
		DVSS	1			
		N/A	0	1	1	X
		DVSS	1			
P2.6/PM_TA0.3(2)	6	P2.6 (I/O)	I: 0; O: 1	0	0	X
		TA0.CCI3A	0	0	1	default
		TA0.3	1			
		N/A	0	1	0	X
		DVSS	1			
		N/A	0	1	1	X
		DVSS	1			
P2.7/PM_TA0.4(2)	7	P2.7 (I/O)	I: 0; O: 1	0	0	X
		TA0.CCI4A	0	0	1	default
		TA0.4	1			
		N/A	0	1	0	X
		DVSS	1			
		N/A	0	1	1	X
		DVSS	1			

(1) X = don't care
(2) Not available on the 64-pin RGC package.

12.4.3 PxOUT Register

Port X Output Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

Figure 12-3. PxOUT Register

7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-6. PxOUT Register Description

Bit	Field	Type	Reset	Description
7-0	PxOUT	RW	Undefined	Port X output. When I/O configured to output mode: 0b = Output is low. 1b = Output is high. When I/O configured to input mode and pullups/pulldowns enabled: 0b = Pulldown selected 1b = Pullup selected

12.4.4 PxDIR Register

Port X Direction Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

Figure 12-4. PxDIR Register

7	6	5	4	3	2	1	0
PxDIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-7. PxDIR Register Description

Bit	Field	Type	Reset	Description
7-0	PxDIR	RW	0h	Port X direction. 0b = Port configured as input 1b = Port configured as output

12.4.5 PxREN Register

Port X Pullup or Pulldown Resistor Enable Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

Figure 12-5. PxREN Register

7	6	5	4	3	2	1	0
PxREN							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-8. PxREN Register Description

Bit	Field	Type	Reset	Description
7-0	PxREN	RW	0h	Port X pullup or pulldown resistor enable. When the port is configured as an input, setting this bit enables or disables the pullup or pulldown. 0b = Pullup or pulldown disabled 1b = Pullup or pulldown enabled

12.4.7 PxSEL0 Register

Port X Function Selection Register 0 (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

Figure 12-7. PxSEL0 Register

7	6	5	4	3	2	1	0
PxSEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-10. PxSEL0 Register Description

Bit	Field	Type	Reset	Description
7-0	PxSEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port X. The values of each bit position in PxSEL1 and PxSEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. See PxSEL1 for the definition of each value.

12.4.8 PxSEL1 Register

Port X Function Selection Register 1 (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

Figure 12-8. PxSEL1 Register

7	6	5	4	3	2	1	0
PxSEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-11. PxSEL1 Register Description

Bit	Field	Type	Reset	Description
7-0	PxSEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port X. The values of each bit position in PxSEL1 and PxSEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

12.4.10 PxIES Register

Port X Interrupt Edge Select Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10)

Figure 12-10. PxIES Register

7	6	5	4	3	2	1	0
PxIES							
rw	rw	rw	rw	rw	rw	rw	rw

Table 12-13. P1IES Register Description

Bit	Field	Type	Reset	Description
7-0	PxIES	RW	Undefined	Port X interrupt edge select 0b = PxIFG flag is set with a low-to-high transition. 1b = PxIFG flag is set with a high-to-low transition.

12.4.11 PxIE Register

Port X Interrupt Enable Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10)

Figure 12-11. PxIE Register

7	6	5	4	3	2	1	0
PxIE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-14. PxIE Register Description

Bit	Field	Type	Reset	Description
7-0	PxIE	RW	0h	Port X interrupt enable 0b = Corresponding port interrupt disabled 1b = Corresponding port interrupt enabled

12.4.12 PxIFG Register

Port X Interrupt Flag Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10)

Figure 12-12. PxIFG Register

7	6	5	4	3	2	1	0
PxIFG							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 12-15. PxIFG Register Description

Bit	Field	Type	Reset	Description
7-0	PxIFG	RW	0h	Port X interrupt flag 0b = No interrupt is pending. 1b = Interrupt is pending.

NVIC

NVIC Table (P118 MSP432 Datasheets)

Table 6-39. NVIC Interrupts (continued)

NVIC INTERRUPT INPUT	SOURCE	FLAGS IN SOURCE
INTISR[4]	FPU_INT (2)	Combined interrupt from flags in the FPSCR (part of Cortex-M4 FPU)
INTISR[5]	FLCTL	Flash Controller interrupt flags
INTISR[6]	COMP_E0	Comparator_E0 interrupt flags
INTISR[7]	COMP_E1	Comparator_E1 interrupt flags
INTISR[8]	Timer_A0	TA0CCTL0.CCIFG
INTISR[9]	Timer_A0	TA0CCTLx.CCIFG (x = 1 to 4), TA0CTL.TAIFG
INTISR[10]	Timer_A1	TA1CCTL0.CCIFG
INTISR[11]	Timer_A1	TA1CCTLx.CCIFG (x = 1 to 4), TA1CTL.TAIFG
INTISR[12]	Timer_A2	TA2CCTL0.CCIFG
INTISR[13]	Timer_A2	TA2CCTLx.CCIFG (x = 1 to 4), TA2CTL.TAIFG
INTISR[14]	Timer_A3	TA3CCTL0.CCIFG
INTISR[15]	Timer_A3	TA3CCTLx.CCIFG (x = 1 to 4), TA3CTL.TAIFG
INTISR[16]	eUSCI_A0	UART or SPI mode TX, RX, and Status Flags
INTISR[17]	eUSCI_A1	UART or SPI mode TX, RX, and Status Flags
INTISR[18]	eUSCI_A2	UART or SPI mode TX, RX, and Status Flags
INTISR[19]	eUSCI_A3	UART or SPI mode TX, RX, and Status Flags
INTISR[20]	eUSCI_B0	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[21]	eUSCI_B1	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[22]	eUSCI_B2	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[23]	eUSCI_B3	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[24]	Precision ADC	IFG[0-31], LO/IN/HI-IFG, RDYIFG, OVIFG, TOVIFG
INTISR[25]	Timer32_INT1	Timer32 interrupt for Timer1
INTISR[26]	Timer32_INT2	Timer32 interrupt for Timer2
INTISR[27]	Timer32_INTC	Timer32 Combined interrupt
INTISR[28]	AES256	AESRDYIFG
INTISR[29]	RTC_C	OFIFG, RDYIFG, TEVIFG, AIFG, RT0PSIFG, RT1PSIFG
INTISR[30]	DMA_ERR	DMA error interrupt
INTISR[31]	DMA_INT3	DMA completion interrupt3
INTISR[32]	DMA_INT2	DMA completion interrupt2
INTISR[33]	DMA_INT1	DMA completion interrupt1
INTISR[34]	DMA_INT0 (3)	DMA completion interrupt0
INTISR[35]	I/O Port P1	P1IFG.x (x = 0 to 7)
INTISR[36]	I/O Port P2	P2IFG.x (x = 0 to 7)
INTISR[37]	I/O Port P3	P3IFG.x (x = 0 to 7)
INTISR[38]	I/O Port P4	P4IFG.x (x = 0 to 7)
INTISR[39]	I/O Port P5	P5IFG.x (x = 0 to 7)
INTISR[40]	I/O Port P6	P6IFG.x (x = 0 to 7)
INTISR[41]	Reserved	
INTISR[42]	Reserved	

2.4.3.2 ISER1 Register (Offset = 104h) [reset = 00000000h]

ISER1 is shown in Figure 2-21 and described in Table 2-27.

Irq 32 to 63 Set Enable Register. Use the Interrupt Set-Enable Registers to enable interrupts and determine which interrupts are currently enabled.

Figure 2-21. ISER1 Register

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
SETENA																															
R/W-0h																															

Table 2-27. ISER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETENA	R/W	0h	Writing 0 to a SETENA bit has no effect, writing 1 to a bit enables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the SETENA fields.

2.4.3.8 ICPR1 Register (Offset = 284h) [reset = 00000000h]

ICPR1 is shown in Figure 2-27 and described in Table 2-33.

Irq 32 to 63 Clear Pending Register. Use the Interrupt Clear-Pending Registers to clear pending interrupts and determine which interrupts are currently pending.

Figure 2-27. ICPR1 Register

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
CLRPEND																															
R/W-0h																															

Table 2-33. ICPR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLRPEND	R/W	0h	Writing 0 to a CLRPEND bit has no effect, writing 1 to a bit clears the corresponding pending interrupt. Reading the bit returns its current state.

2.4.3.20 IPR9 Register (Offset = 424h) [reset = 00000000h]

IPR9 is shown in Figure 2-39 and described in Table 2-45.

Irq 36 to 39 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

Figure 2-39. IPR9 Register

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
PRI_39								PRI_38								PRI_37								PRI_36							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

Table 2-45. IPR9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_39	R/W	0h	Priority of interrupt 39
23-16	PRI_38	R/W	0h	Priority of interrupt 38
15-8	PRI_37	R/W	0h	Priority of interrupt 37
7-0	PRI_36	R/W	0h	Priority of interrupt 36