Topic 10: Exceptions/Interrupts Ch. 6

ECE331

Rev.S11

Outline

- Exceptions
  - concept
- Resets
  - events
  - responses
- Interrupts
  - concept
  - non-maskable
  - maskable
  - HC12 interrupts
- Interrupt response
- Exception priority
- Hardware interfaces

Exceptions

- Normal program flow: beginning to end
  - can only evaluate pre-programmed conditional decisions
  - cannot respond to exceptions in normal program flow

    Example:

- Exceptions = break in normal program flow

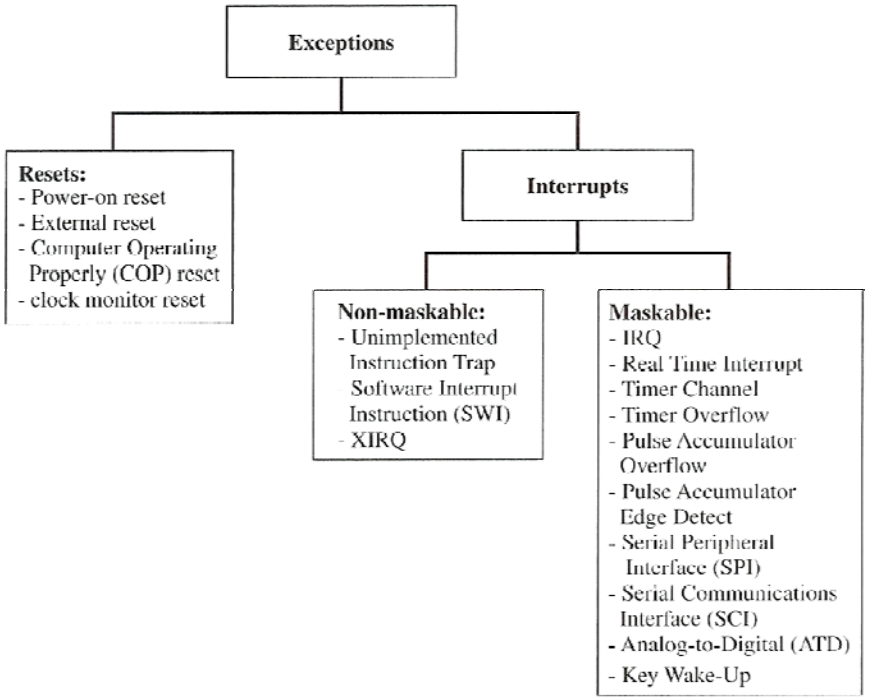
  2 Types
  - Resets:

    Example:

  - Interrupts:

    Example:

HC12/S12 Exception Systems



Resets

- Purpose: return to known state of operation
- Cause: external event or malfunction
- Events that trigger system reset (HC12)
  - Power-on Reset
    - Cause:
    - Action: start program execution with defined configuration
  - Computer Operating Properly (COP) Reset
    - COP is a hardware unit that detects malfunctions in software execution
      - user programmable “time out” duration
    - COP must be enabled; generally disabled while testing
    - Example:
  - Clock Monitor Reset
    - Cause:
  - External Reset
    - Cause:
      - e.g. push button reset, as used in lab

Reset Response

When a reset occurs…

- For external resets
  - program starts over from beginning

- For internal resets (COP & Clock Monitor)
  - specific "reset vector" immediately stored to the PC
    - jump to user code (@ reset vector)
  - reset vector
    - a preset memory address assigned to each exception event
    - contains address of user-defined code to manage exception event
- See "exception vector" tables at end of slides

Interrupts

- Purpose
  - permit MCU to respond to events of higher priority than the normal program

- Cause
  - signal or event causing a break in normal program flow
- Action
  - jump to "interrupt vector" (like reset vector)
  - "interrupt vector" points to specific subroutine for each interrupt event
- Alternative to Interrupts
  - Polling: code continuously monitors for event (e.g., hardware flag) to occur; can't do anything else except wait for event

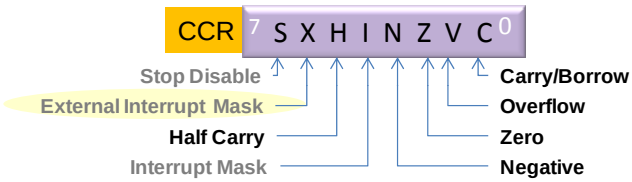
Types of Interrupts

- Maskable
  - interrupt can be enabled/disabled by software programmable hardware settings
    - e.g., configuration registers (register set)
- Non-Maskable
  - interrupt request always honored; can not be turned off by user
  - useful for critical events
    - e.g., power down for batter level low

HC12/S12 Interrupt System

Non-Maskable Interrupts

- Indicator/Control
  - active when CCR bit X is '0'
  - not controllable by user
- Events
  - Unimplemented Instruction Trap
    - for unassigned op-codes
  - Software Interrupt Instruction (SWI)
    - ASM instruction
  - Nonmaskable Interrupt Request ($\overline{\text{XIRQ}}$)
    - active low external pin on MCU



HC12/S12 Interrupt System

- Maskable Interrupts
- Control/Indicator
  - master "mask" (on/off switch): CCR bit I (I = inhibit)
  - @ system reset, I = 1 (off)
  - CLI instruction → I = 0 (turns maskable interrupt system on)
    - **CLI =**
  - SEI insruction → I = 1 (turns maskable interrupt system off)
    - **SEI =**
- Local Masks
  - MCU hardware system interrupts are enabled by local masks
    - e.g., TOI bit in the timer system enables timer interrupts

Master mask **Local masks**

HC12/S12 Maskable Interrupts

- Maskable Interrupt Request ($\overline{IRQ}$)
  - external pin, active low
  - primary external interrupt
    - can be configured for multiple interrupt sources
- Real-Time Interrupt (RTI)
  - periodic interrupt, programmable duration
  - Example:
- Timer Overflow
  - enabled by TOI bit (in configuration register)
- Analog to Digital System
  - create interrupt when A/D conversion complete
- Serial Communication System
  - several maskable interrupts
- Many more…

see complete list in exception vector list (end of slides)

Interrupt Response

What happens when an interrupt occurs?

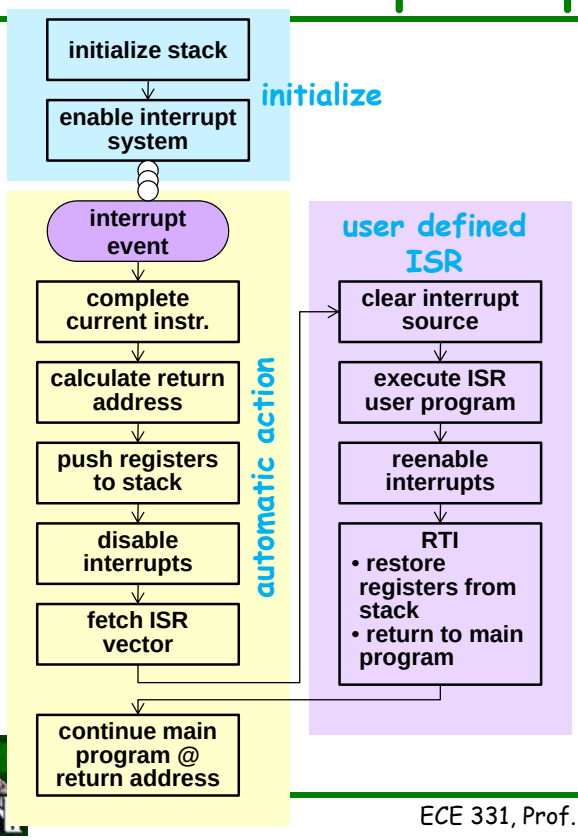
- Interrupt Service Routine (ISR)
  - software code to handle each interrupt source
  - address (location) for each ISR stored at specific memory location called an interrupt vector

    review exception vector list (end of slides)

  - RTI (return from interrupt)
    - ASM instruction to return from interrupt back to main program

- Automated Interrupt Functions
  - performed automatically by hardware (without software)
  - varies with MCU
  - see flowchart on next slide

Interrupt Response Chart



HC12 Auto Interrupt Response

1. Finish current instruction
   - PC set to next instruction
2. Prepare for ISR
   - calculate return address
     - push return address to stack
   - push register values to stack
   - disable interrupts
3. Fetch ISR vector
   - load PC with ISR vector
   - execute ISR
   - return to main (RTI) within ISR

- see stack order on next slide

Exception Priority

What happens if multiple exceptions occur at the same time?

- Exceptions have hard-wired priority (order of importance) that determines order

1. Resets
   1. Power-on reset
   2. Clock monitor reset
   3. COP watchdog reset
2. Non-maskable interrupts (NMI)
   1. Unimplemented instruction trap
   2. SWI
   3. XIRQ
3. Maskable interrupts
   - priority set by interrupt vector map
   - higher addresses = higher priority

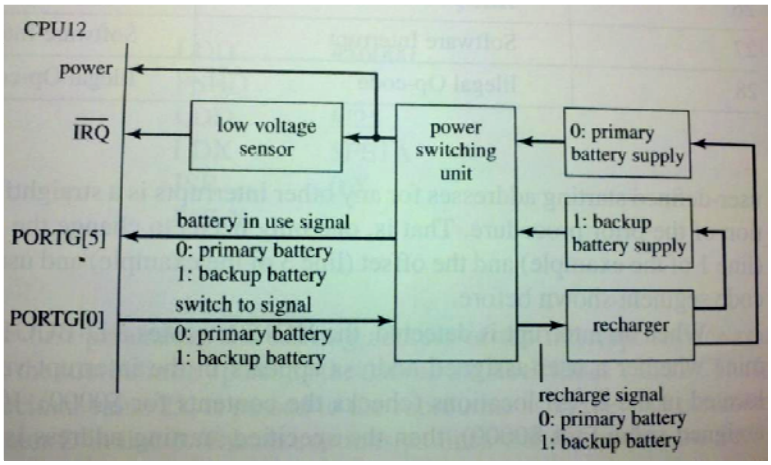
HC12 Stack for Interrupts

top

bottom

Hardware Interfacing

- Example: backup power system; interrupt driven switching between two rechargeable power supplies
  - IRQ'
  - PortG, pin 5,
    - .
  - PortG, pin 0,
    - .



68HCS12 Exception Vector Locations

Vector Address	Interrupt Source	CCR Mask	Local Enable	HPRIO Value to Elevate
$FFFE, $FFFF	Reset	None	None	–
$FFFC, $FFFD	Clock Monitor fail reset	None	PLLCTL (CME, SCME)	–
$FFFA, $FFFB	COP failure reset	None	COP rate select	–
$FFF8, $FFF9	Unimplemented instruction trap	None	None	–
$FFF6, $FFF7	SWI	None	None	–
$FFF4, $FFF5	XIRQ	X-Bit	None	–
$FFF2, $FFF3	IRQ	I-Bit	IRQCR (IRQEN)	$F2
$FFF0, $FFF1	Real Time Interrupt	I-Bit	CRGINT (RTIE)	$F0
$FFEE, $FFEF	Enhanced Capture Timer channel 0	I-Bit	TIE (C0I)	$EE
$FFEC, $FFED	Enhanced Capture Timer channel 1	I-Bit	TIE (C1I)	$EC
$FFEA, $FFEB	Enhanced Capture Timer channel 2	I-Bit	TIE (C2I)	$EA
$FFE8, $FFE9	Enhanced Capture Timer channel 3	I-Bit	TIE (C3I)	$E8
$FFE6, $FFE7	Enhanced Capture Timer channel 4	I-Bit	TIE (C4I)	$E6
$FFE4, $FFE5	Enhanced Capture Timer channel 5	I-Bit	TIE (C5I)	$E4
$FFE2, $FFE3	Enhanced Capture Timer channel 6	I-Bit	TIE (C6I)	$E2
$FFE0, $FFE1	Enhanced Capture Timer channel 7	I-Bit	TIE (C7I)	$E0
$FFDE, $FFDF	Enhanced Capture Timer overflow	I-Bit	TSRC2 (TOF)	$DE
$FFDC, $FFDD	Pulse accumulator A overflow	I-Bit	PACTL (PAOVI)	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I-Bit	PACTL (PAI)	$DA
$FFD8, $FFD9	SPI0	I-Bit	SP0CR1 (SPIE, SPTIE)	$D8
$FFD6, $FFD7	SCI0	I-Bit	SC0CR2 (TIE, TCIE, RIE, ILIE)	$D6
$FFD4, $FFD5	SCI1	I-Bit	SC1CR2 (TIE, TCIE, RIE, ILIE)	$D4
$FFD2, $FFD3	ATD0	I-Bit	ATD0CTL2 (ASCIE)	$D2
$FFD0, $FFD1	ATD1	I-Bit	ATD1CTL2 (ASCIE)	$D0
$FFCE, $FFCF	Port J	I-Bit	PTJIF (PTJIE)	$CE
$FFCC, $FFCD	Port H	I-Bit	PTHIF(PTHIE)	$CC
$FFCA, $FFCB	Modulus Down Counter underflow	I-Bit	MCCTL(MCZI)	$CA

68HCS12 Exception Vector Locations

$FFC8, $FFC9	Pulse Accumulator B Overflow	I-Bit	PBCTL(PBOVI)	$C8
$FFC6, $FFC7	CRG PLL lock	I-Bit	CRGINT(LOCKIE)	$C6
$FFC4, $FFC5	CRG Self Clock Mode	I-Bit	CRGINT (SCMIE)	$C4
$FFC2, $FFC3	BDLC	I-Bit	DLCBCR1(IE)	$C2
$FFC0, $FFC1	IIC Bus	I-Bit	IBCR (IBIE)	$C0
$FFBE, $FFBF	SPI1	I-Bit	SP1CR1 (SPIE, SPTIE)	$BE
$FFBC, $FFBD	SPI2	I-Bit	SP2CR1 (SPIE, SPTIE)	$BC
$FFBA, $FFBB	EEPROM	I-Bit	EECTL(CCIE, CBEIE)	$BA
$FFB8, $FFB9	FLASH	I-Bit	FCTL(CCIE, CBEIE)	$B8
$FFB6, $FFB7	CAN0 wake-up	I-Bit	CAN0RIER (WUPIE)	$B6
$FFB4, $FFB5	CAN0 errors	I-Bit	CAN0RIER (CSCIE, OVRIE)	$B4
$FFB2, $FFB3	CAN0 receive	I-Bit	CAN0RIER (RXFIE)	$B2
$FFB0, $FFB1	CAN0 transmit	I-Bit	CAN0TIER (TXEIE2-TXEIE0)	$B0
$FFAE, $FFAF	CAN1 wake-up	I-Bit	CAN1RIER (WUPIE)	$AE
$FFAC, $FFAD	CAN1 errors	I-Bit	CAN1RIER (CSCIE, OVRIE)	$AC
$FFAA, $FFAB	CAN1 receive	I-Bit	CAN1RIER (RXFIE)	$AA
$FFA8, $FFA9	CAN1 transmit	I-Bit	CAN1TIER (TXEIE2-TXEIE0)	$A8
$FFA6, $FFA7	CAN2 wake-up	I-Bit	CAN2RIER (WUPIE)	$A6
$FFA4, $FFA5	CAN2 errors	I-Bit	CAN2RIER (CSCIE, OVRIE)	$A4
$FFA2, $FFA3	CAN2 receive	I-Bit	CAN2RIER (RXFIE)	$A2
$FFA0, $FFA1	CAN2 transmit	I-Bit	CAN2TIER (TXEIE2-TXEIE0)	$A0
$FF9E, $FF9F	CAN3 wake-up	I-Bit	CAN3RIER (WUPIE)	$9E
$FF9C, $FF9D	CAN3 errors	I-Bit	CAN3RIER (TXEIE2-TXEIE0)	$9C
$FF9A, $FF9B	CAN3 receive	I-Bit	CAN3RIER (RXFIE)	$9A
$FF98, $FF99	CAN3 transmit	I-Bit	CAN3TIER (TXEIE2-TXEIE0)	$98
$FF96, $FF97	CAN4 wake-up	I-Bit	CAN4RIER (WUPIE)	$96
$FF94, $FF95	CAN4 errors	I-Bit	CAN4RIER (CSCIE, OVRIE)	$94
$FF92, $FF93	CAN4 receive	I-Bit	CAN4RIER (RXFIE)	$92
$FF90, $FF91	CAN4 transmit	I-Bit	CAN4TIER (TXEIE2-TXEIE0)	$90
$FF8E, $FF8F	Port P Interrupt	I-Bit	PTPIF (PTPIE)	$8E
$FF8C, $FF8D	PWM Emergency Shutdown	I-Bit	PWMSDN (PWMIE)	$8C
$FF80 to $FF8B	Reserved