Interrupt handler

201624622 홍명신
201724428 김승민
201724626 Laith Tbalieh
201824602 최원준

Contents

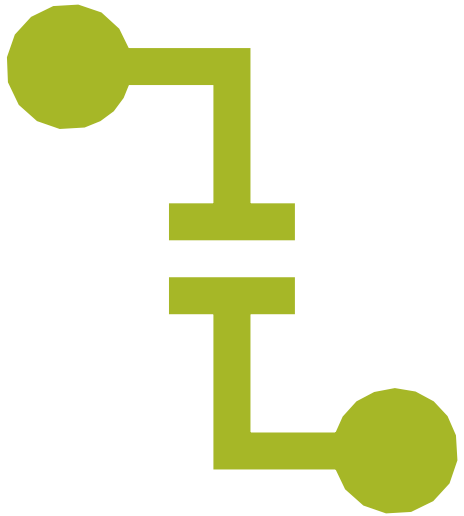
Interrupt Handler란?

The Definition of “Interrupt”

Interrupt: An input signal to the processor which indicates an event that needs immediate attention

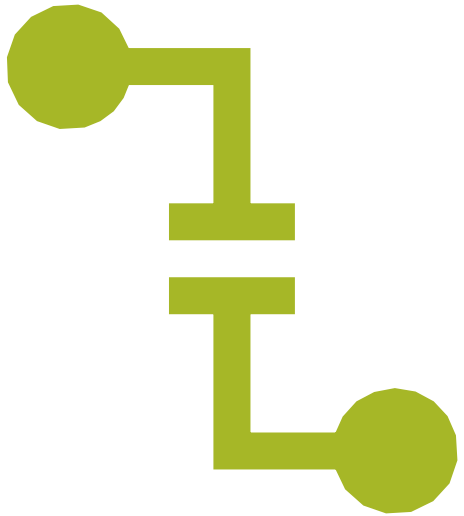
What are the differences between Polling and Interrupt? -> Polling

1. The host repeatedly reads the **busy** bit until that bit becomes clear.
2. The host sets the **write** bit in the **command** register and writes a byte into the **data-out** Register.
3. The host sets the **command-ready** bit.
4. When the controller notices that the **command-ready** bit is set, it sets the **busy** bit.

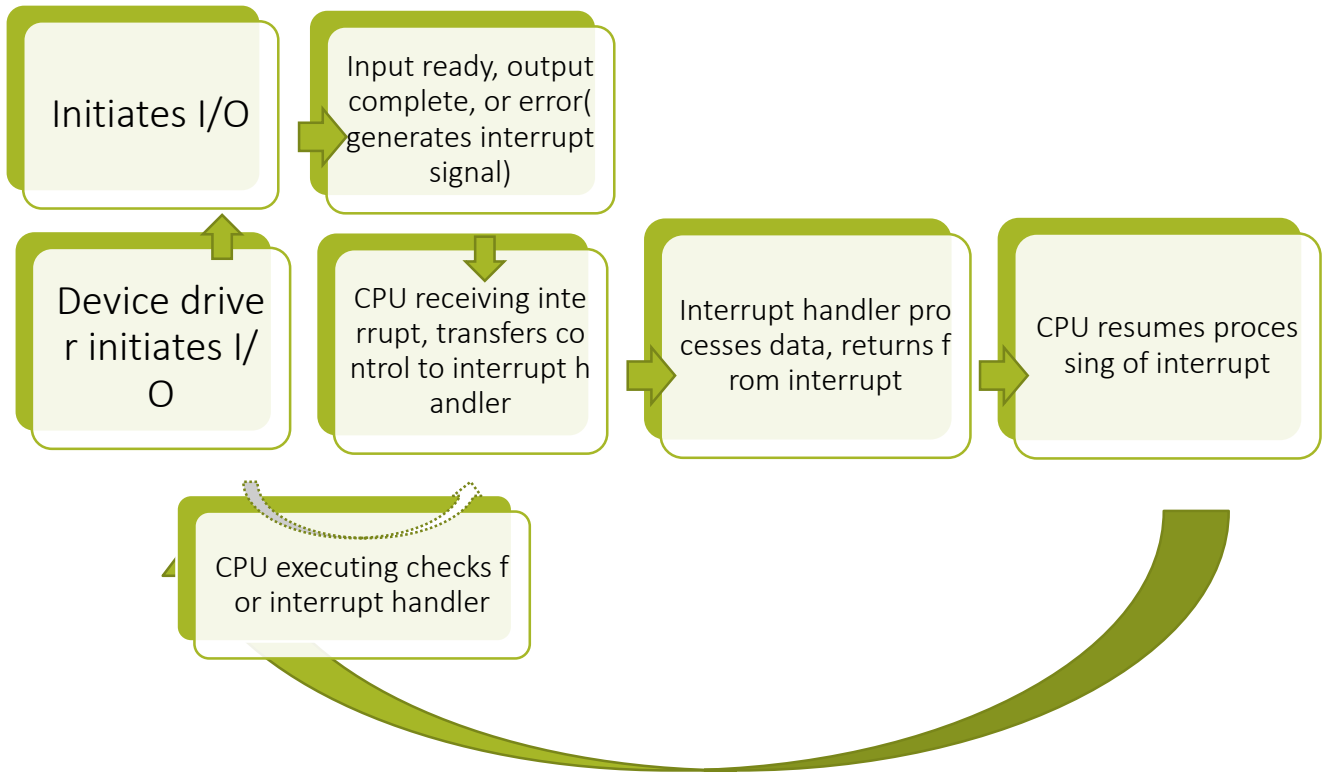


What are the differences between Polling and Interrupt? -> Polling(cont)

5. The controller reads the command register and sees the **`write`** command. It reads the **`data-out`** Register to get the byte, and does the I/O to the device.
6. The controller clears the **`command-ready`** bit, clears the **`error`** bit in the status register to indicate that the device I/O succeeded, and clears the **`busy`** bit to indicate that it is finished.



What are the differences between Polling and Interrupt? -> Interrupt

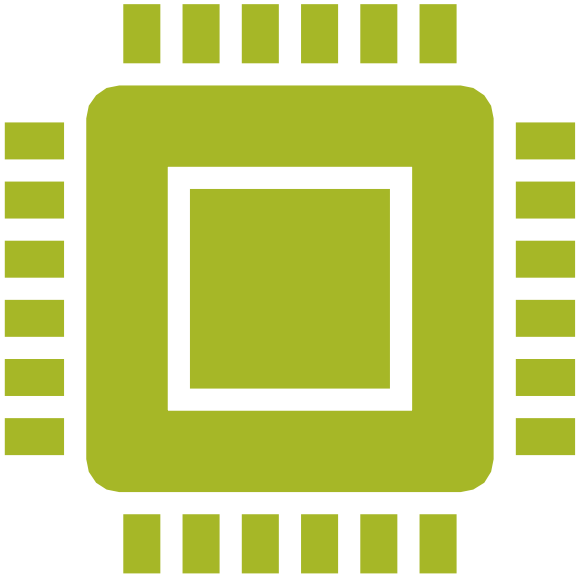


Interrupt Handler

A special block of code associated with a specific “interrupt“ condition.

Initiated by hardware interrupts, software interrupt instructions, or software exceptions.(Some examples in next slide)

We will study hardware interrupts.



Interrupt by H/W vs S/W

《HARDWARE INTERRUPT》

Using virtual-memory paging

System call

《 SOFTWARE INTERRUPT》

Switching to supervisor mode

Interrupt Handler의 구현

Initializing Interrupts using EXTI

Interrupts be prioritized using NVIC

- EXTI: EXTernal Interrupt
- NVIC: Nested Vectored Interrupt Controller
  - ※Vectored: 방향성이 있는

Ex) Joystick, Switch etc.

How can we make an Interrupt handling?

Interrupt Handling 구현-GPIO 선언

```
140 void GPIO_Configure(void){
141     GPIO_InitTypeDef GPIO_InitStructure1,
142
143
144
145
146
147
148 // GPIO Initialize
```

Interrupt Handling 구현-GPIO 포트 설정

```
149     //PD11 USER S1
150     GPIO_InitStructure1.GPIO_Pin = GPIO_Pin_11;
151     GPIO_InitStructure1.GPIO_Speed = GPIO_Speed_50MHz;
152     GPIO_InitStructure1.GPIO_Mode = GPIO_Mode_IPD;
153
154     GPIO_Init(GPIOD,&GPIO_InitStructure1);
155     GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
```

Interrupt Handling 구현-USART 설정

```
178 void USART_Configure(void){
179     USART_InitTypeDef USART_InitStructure;
180 
181     USART_InitStructure.USART_BaudRate = 9600;
182     USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
183     USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
184     USART_InitStructure.USART_Parity = USART_Parity_No;
185     USART_InitStructure.USART_StopBits = USART_StopBits_1;
186     USART_InitStructure.USART_WordLength = USART_WordLength_8b;
187 
188     USART_Init(USART1, &USART_InitStructure);
189     USART_Cmd(USART1, ENABLE);
190     USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
191 }
192 
```

Interrupt Handling 구현-EXTI 설정

```
193
194 void EXTI_Configure(void){
195     EXTI_InitTypeDef EXTI_InitStructure1,EXTI_InitStructure2;
196     EXTI_InitStructure1.EXTI_Line = EXTI_Line11;
197     EXTI_InitStructure1.EXTI_LineCmd = ENABLE;
198     EXTI_InitStructure1.EXTI_Mode = EXTI_Mode_Interrupt;
199     EXTI_InitStructure1.EXTI_Trigger = EXTI_Trigger_Rising;
200     EXTI_Init(&EXTI_InitStructure1);
201
202
203
204
205
206 }
207
```

Interrupt Handling 구현-NVIC 초기 설정

```
207 void NVIC_Configure(void){
208     NVIC_InitTypeDef NVIC_InitStructure1,NVIC_InitStructure2;
209 
210     NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
211     NVIC_InitStructure1.NVIC_IRQChannel = EXTI15_10_IRQn;
212     NVIC_InitStructure1.NVIC_IRQChannelPreemptionPriority = 0x00;
213 
214     NVIC_InitStructure1.NVIC_IRQChannelSubPriority = 0x00;
215     NVIC_InitStructure1.NVIC_IRQChannelCmd = ENABLE;
216     NVIC_Init(&NVIC_InitStructure1);
217 
218 
219 
220 
221 
222 
223 
224 
225 }
```

Interrupt Handling 구현- Handler 설정

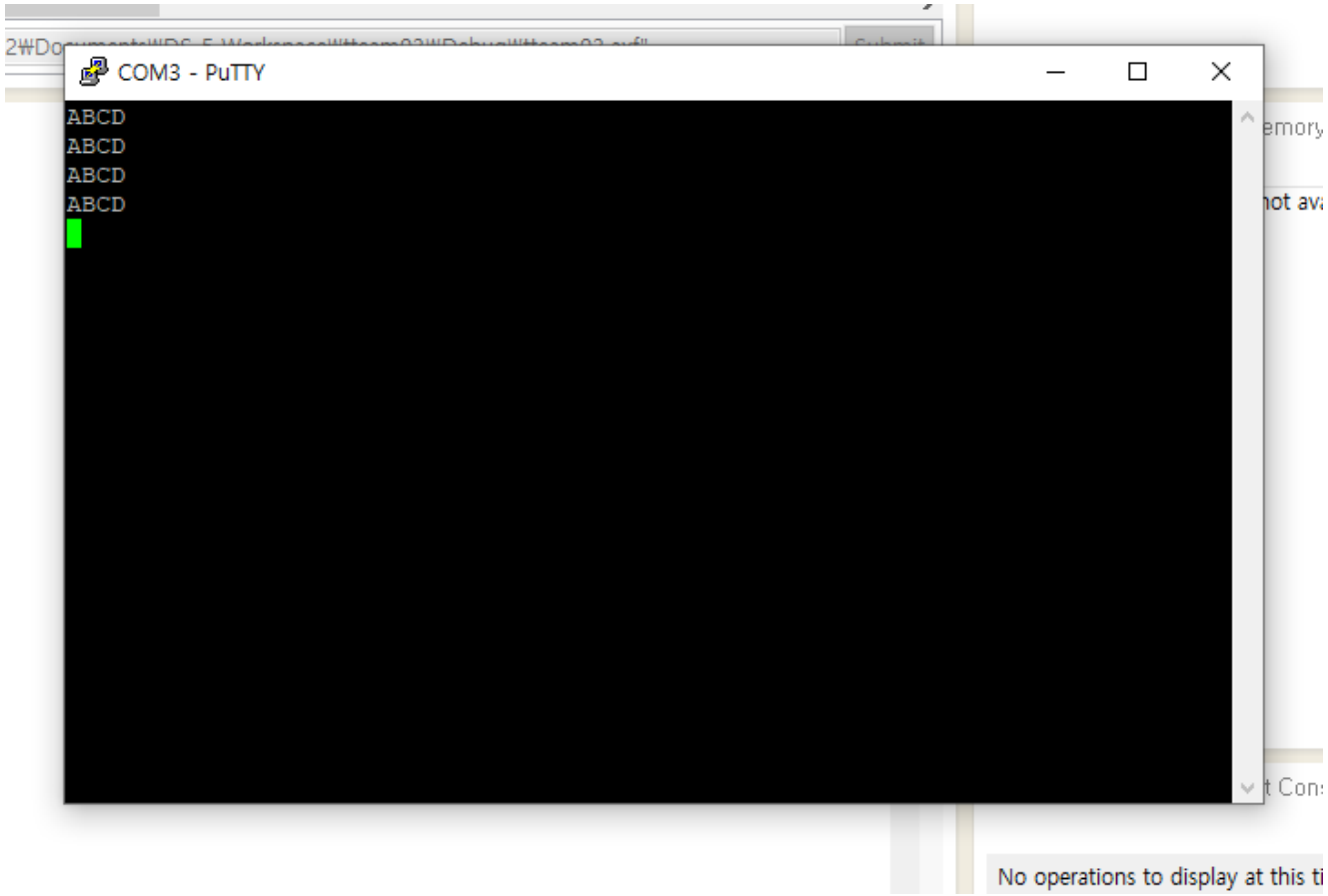
```
227 //sw1 PD11
228 void EXTI15_10_IRQHandler(void){
229     if (EXTI_GetITStatus(EXTI_Line11) != RESET){
230         sw1_state = 1;
231         EXTI_ClearITPendingBit(EXTI_Line11);
232     }
233 }
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248 void delay(){
249     int dummy;
250     for(dummy = 0; dummy < 1000000; dummy++);
251 }
252
```

Interrupt Handling 구현-LED 점멸

```
270 int main(){
271     SystemInit();
272     RCC_Configure();
273     GPIO_Configure();
274     USART_Configure();
275     EXTI_Configure();
276     NVIC_Configure();
277 
278     while(1){
279 
280 
281 
282 
283 
284 
285 
286 
287 
288 
289 
290 
291 
292 
293         }
294 
295         /*Turn off all LEDs*/
296         for(LEDx_index = 0; LEDx_index < 3; LEDx_index++){
297             GPIO_ResetBits(GPIOD,LEDx[LEDx_index]);
298         }
```

Interrupt Handling 구현2-PuTTY String input

```
300             if(sw1_state == 1){
301                 int i;
302                 for (i = 0; i < 6; i++ ){
303                     USART_SendData(USART1, data[i]);
304                     delay();
305                 }
306                 sw1_state = 0;
307 
308             }
309         }
310 }
```



Result II- PuTTY

출처

Page 4 – Page 8: Operating System Concepts 6/e<Abraham Silberschatz>


Q
A