**REPORT**

**| Embedded System**

**| 최종 프로젝트**

**| 컴퓨터정보공학과**

**| 12151476 정보민**

**| 12161558 김혜윤**

**목차**

1. **개요**

* **사용한 기능 및 간단한 설명**

1. **프로젝트 내용 설명**

* **시간내에 불켜기 게임**

1. **프로젝트 실행**

* **실행 과정 스크린샷 및 진행 순서**

1. **소스 코드**

* **변수 설명**
* **함수 설명**
* **IPC**
* **코드 진행**

**개요**

**• 사용한 기능 및 간단한 설명**

최종 프로젝트에서는 CPU에서의 개발과 임베디드 보드에서의 실행을 교차 개발 환경을 사용했고, AVR Studio, Visual Studio를 사용해 MicroC/OS-II를 ATmega128에 Porting 하여 개발하였습니다.

MicroC/OS-II의 주요 기능인 real-time scheduling 및 IPC(Mail Box, Message Queue, Semaphore, Event Flag)를 사용해 우선 task실행 및 task 간의 통신 기능을 사용하였고, ATmega128의 센서 (FND, LED, Buzzer, Switch)와 interrupt(INT4,INT5,Timer 2 Overflow) 기능을 사용하였습니다. FND,LED,Buzzer,Switch,Interrup에 대한 register는 critical section에 진입해 초기값으로 지정해 두고, interrupt 의 발생과 ISR의 실행 위주로 코드가 진행되고, ISR의 결과로 바뀐 변수 값을 확인하는 조건문에 task 간의 통신(IPC)를 진행합니다. 총 4개의 Task를 구현했습니다. 소스코드와 영상을 첨부하였습니다.

**프로젝트 내용 설명**

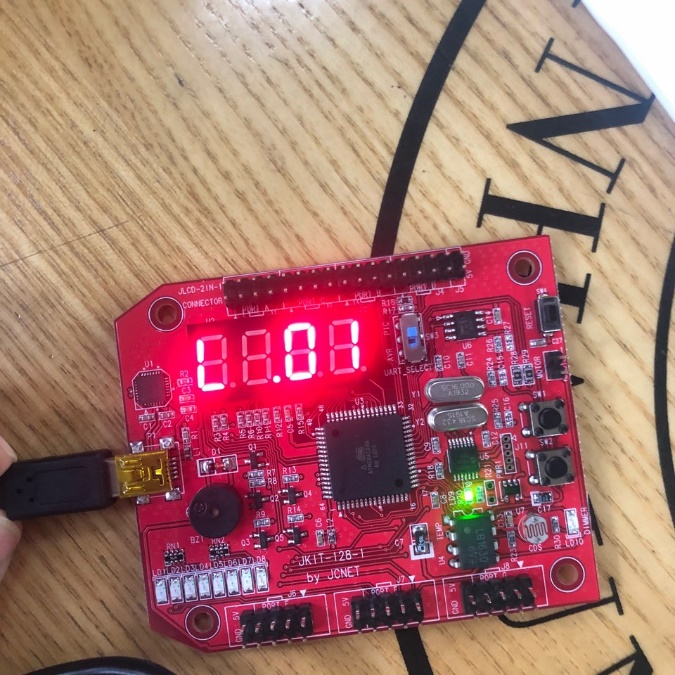
**시간내에 불 켜기 게임**

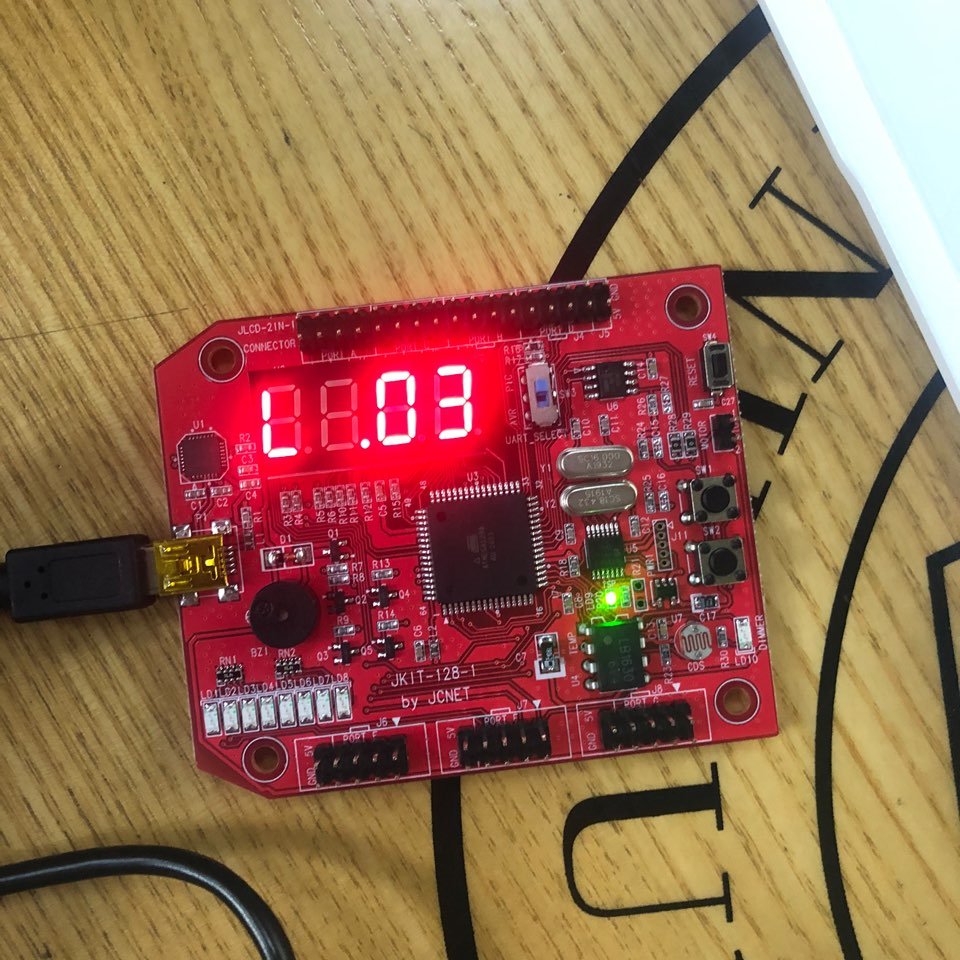
**게임 내용**

1. 광센서로 level 1,2,3 을 설정합니다. FND에 level이 출력됩니다.
2. 게임이 시작되면 LED를 시간내에 모두 켜야합니다.
3. LEVEL 1에서는 150, 2에서는 100, 3에서는 50의 제한시간을 가집니다.
4. 시간내에 완료하면 WIN이라는 글자와 노래가 출력됩니다.
5. 타임내에 완료하지 못하면 LOSE라는 글자가 출력됩니다. 이 경우는 노래가 출력되지않습니다.

**프로젝트 실행**

**실행과정 스크린 샷 및 진행 순서**

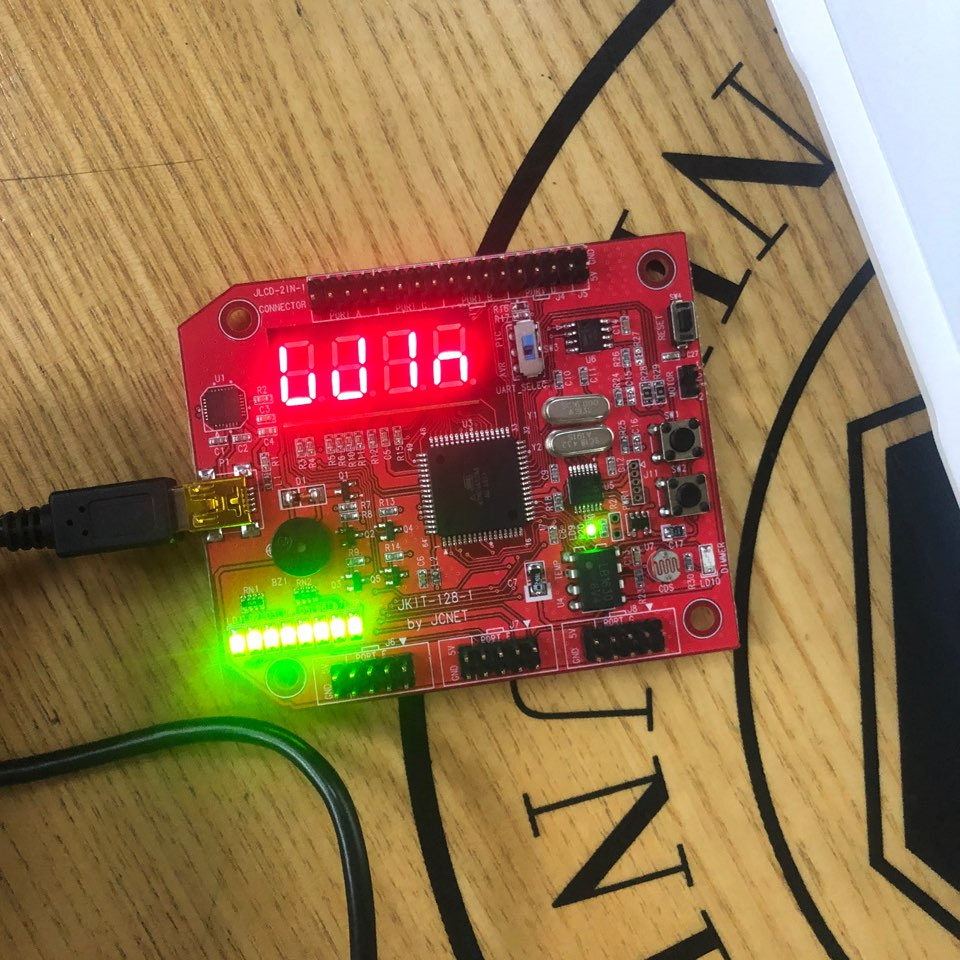
 



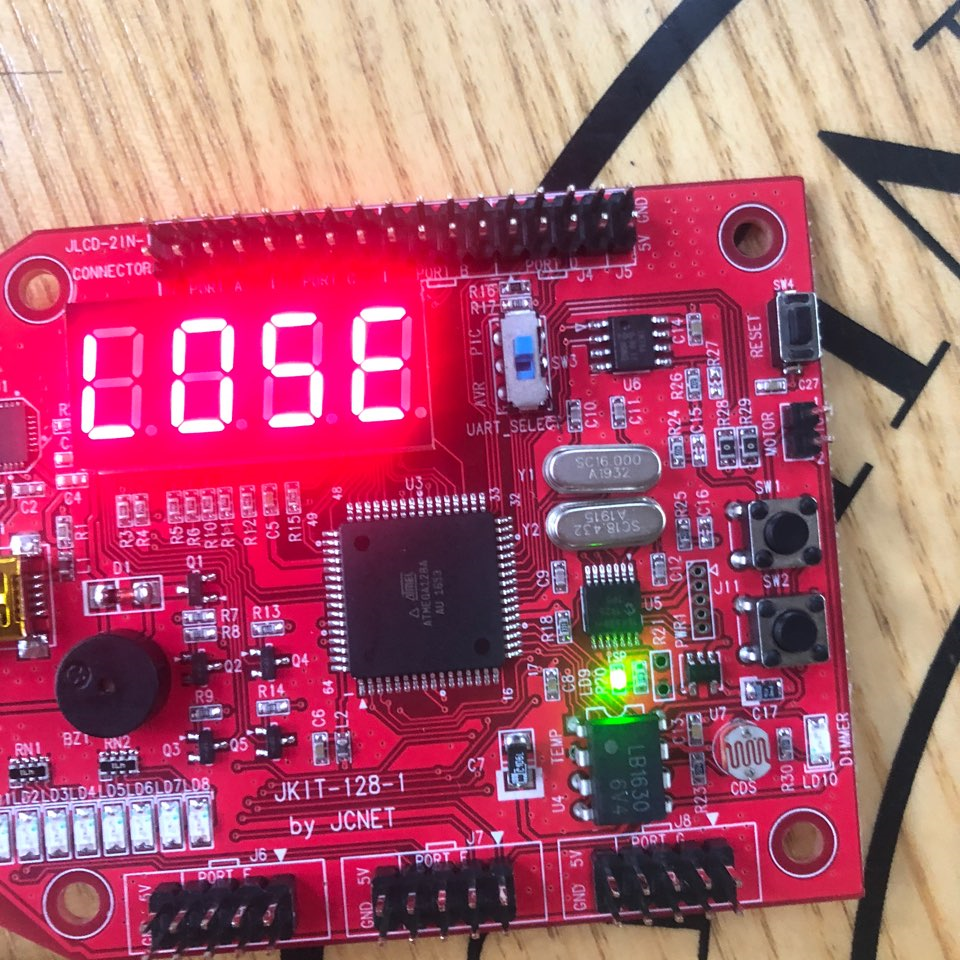
1. 게임 시작 전 LEVEL 설정

Level을 골랐다면, switch 2를 눌러 level pick

1. Switch 1을 눌러 LED를 켭니다.



1. 제한 시간 내에 LED를 모두 켰다면 WIN이라는 글자 출력과 노래가 나옵니다.



1. 제한 시간내에 모두 켜지 못하면 LOSE 라는 글자가 출력됩니다.

**소스 코드**

**변수 설명**

**IPC 관련**

* OS\_EVENT \*Mbox: mail box를 사용하기 위함
* OS\_EVENT \*queue\_to\_led: message box를 사용하기 위함
* OS\_FLAG\_GRP \*l\_grp: event flag를 사용하기 위함
* OS\_FLAG\_GRP \*b\_grp: event flag를 사용하기 위함
* OS\_EVENT \*Sem: Semaphore를 사용하기 위함
* void\* led\_queue[4]: message queue에서 사용하는 배열

**FND 관련**

* unsigned char FND\_DATA[]:FND로 출력할 숫자, 알파벳을 정보가 들어있다.
* volatile int level : level을 저장하는 변수
* char value: 이겼는지 졌는지 여부를 저장하는 변수
* int pend\_time : FND를 출력할 시간
* int stop\_time : 종료 시간을 저장하는 변수
* int on\_cnt : switch 를 누른 횟수

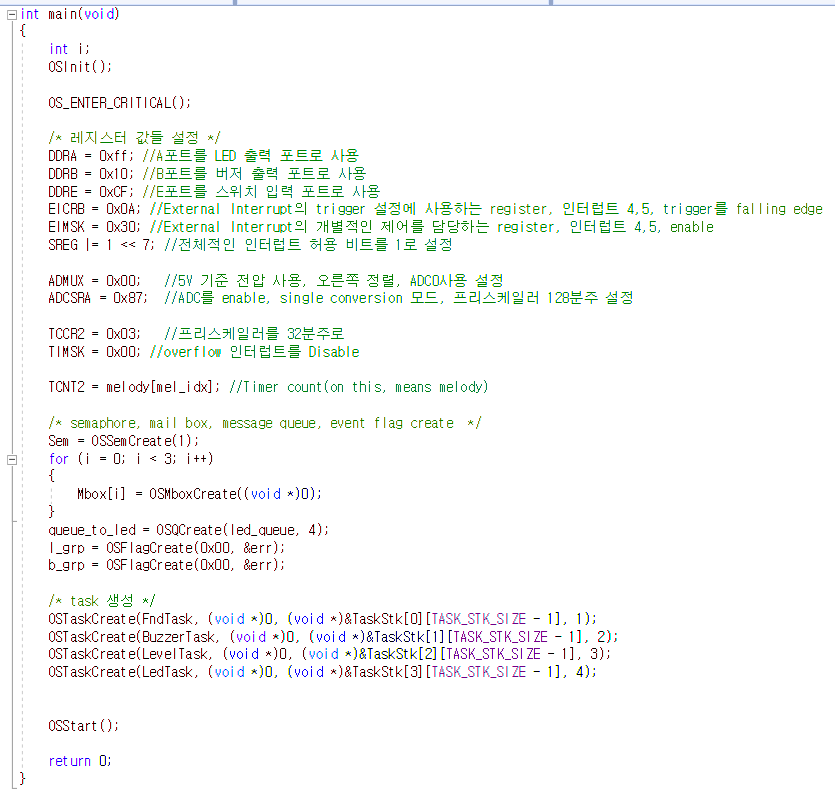
**Buzzer 관련**

* volatile int buzzer\_stat : buzzer의 상태를 저장하는 변수
* volatile int mel\_idx : buzzer에서 melody를 나오게 하는 index
* const unsigned char melody[25] : buzzer에서 출력될 음악이 저장된 변수

**함수 설명**

**메인 및 부수적 함수**

**main**

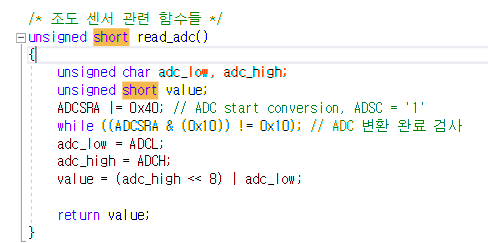


1. critical Section에서 각 register 값 초기화

* DDRA: LED 출력 포트로 사용
* DDRB: Buzzer 출력 포트로 사용
* DDRE: SW1, SW2 사용
* EICRB,EIMSK : interrupt 초기 설정
* SREG: 전체적인 인터럽트 허용
* ADMUX: 5V 기준 전압 사용, 오른쪽 정렬, ADC0사용
* ADCSRA: ADC enable, 단일 변환 모드, 프리스케일러 128분주
* TCNT2 : timer count

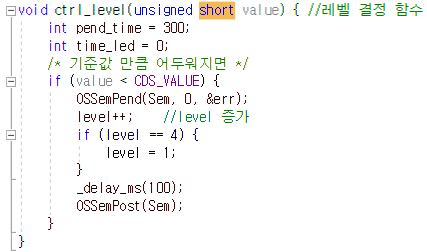
1. 각 IPC 선언 (Mail box, Message Queue, Event flag,Semaphore)
2. Task Create: Fnd, Buzzer, Level, Led 순으로 우선순위 줌
3. OS에게 제어권을 넘겨줌

**unsigned short read\_adc()**



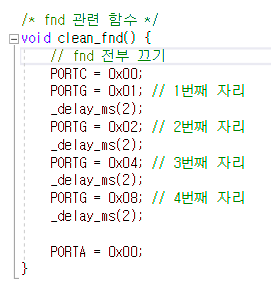
* 아날로그 조도값을 digital 값으로 변환시켜주고 그 값을 return 해준다.

**void ctrl\_level(unsigned short value)**



* Unsigned short read\_adc() 에서 return 한 값을 가져와서 그 값이 기준 값보다 낮으면 level을 증가시켜준다.
* Level이 4가 되면 다시 1로 바꿔준다.

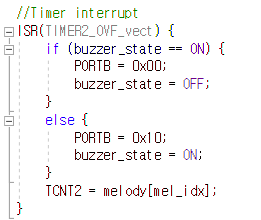
**void clean\_fnd()**



* Fnd를 전부 꺼준다.

**ISR**

**TIMER ISR**

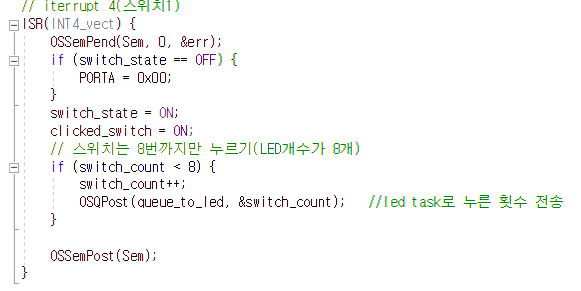


1. 2번 타이머 인터럽트 발생시 buzzer\_state 가 ON이라면 OFF로 ,

OFF라면 ON으로 바꿔준다.

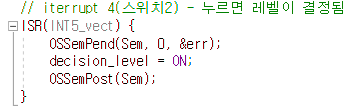
1. TCNT2를 melody[mel\_idx] 로 지정해준다.

**SW1 INT4 ISR**



* SW1이 눌렸을 때, switch\_state가 OFF(level 설정 상태)라면 led를 전부 끄고, switch\_state를 ON(led 상태)로 바꿔주고, clicked\_switch(switch를 눌렀는 지 여부)를 ON으로 해준다. 만약 switch\_count가 8번 미만이라면 swich\_count를 증가 시키고 Message queue로 전송해준다.

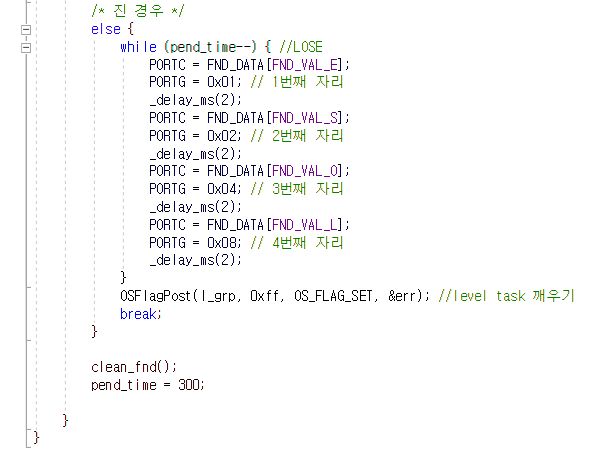
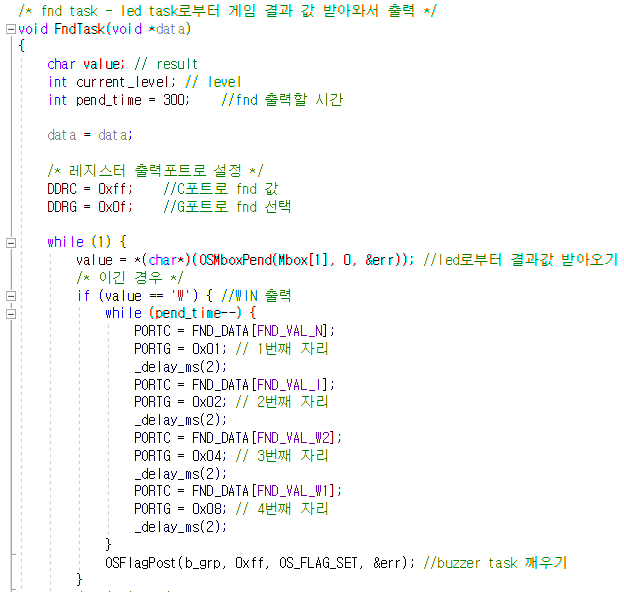
**SW2 INT5 ISR**



* Decision\_level를 ON으로 변경해준다.

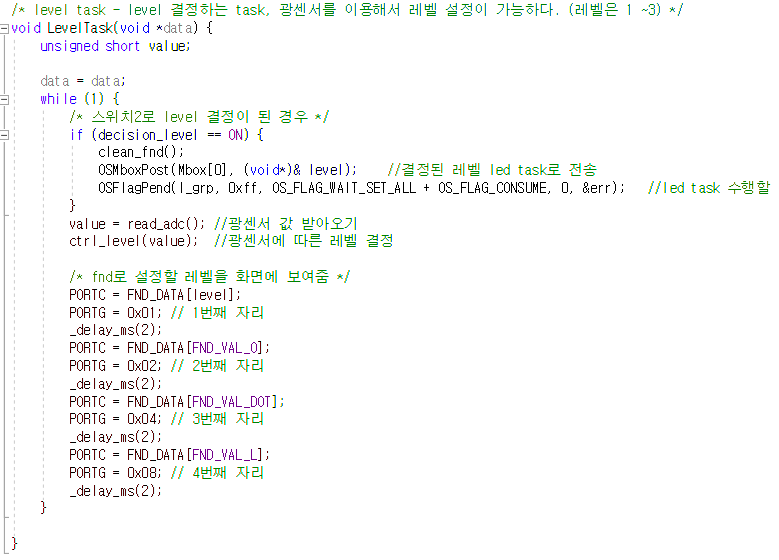
**Task**

**FNDTask**



1. Mbox[1] Mail box에 신호가 오기를 pend한다.
2. 받아온 값이 ‘w’ 라면 win을 출력한다.
3. Pend\_time 동안 fnd를 출력한후, b\_grp event flag에 신호를 준다.
4. 받아온 값이 다른 값이라면 lose 를 출력한다.
5. Pend\_time 동안 fnd를 출력한후 , l\_grp event flag에 신호를 준다.

**LevelTask**



1. SW2\_INT5 ISR에서 decision\_level이 on으로 설정되었다면, clean\_fnd 함수를 호출해준다.
2. 결정된 level을 mbox[0] Mail box로 post한다.
3. L\_grp Event flag에 신호가 올 때 까지 pend한다.
4. Decision\_level이 off라면 value에 adc로 변환된 값을 저장하고, ctrl\_level(value) 함수를 호출한다.
5. FND에 설정된 LEVEL을 출력해준다.

**LedTask**



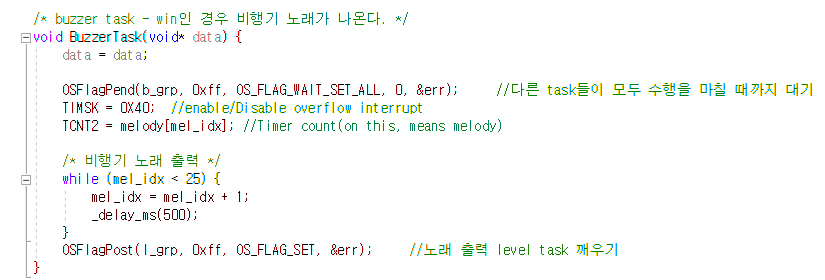
1. Mbox[0] [Mail box로부터 level 값이 올 때 까지 pend 한다.
2. Level에 따라 종료시간을(stop\_time) 계산하고, time\_count를 증가시키다.
3. 만약 time\_count가 stop\_time 보다 크거나 같으면 시간 초과가 발생한 경우임으로 LED를 모두 끄고, result에 L을 넣어주고, result를 mbox[1] mailbox으로 post해준다. Time\_count를 0으로 초기화해준다.
4. 만약 result가 L이 아니고 clicked\_switch가 ON이라면, message queue에서 queue\_to\_led(led task로 누른 횟수) 신호가 올 때 까지 pend 한다.

4-1. message queue 신호를 받으면 clicked\_switch 를 off로 바꾼다.

4-2. 만약 on\_cnt(led task로 누른 횟수) 가 7번이면 (led를 전부 켜서 이긴경우), led를 전부 켜주고 result에 ‘w’ 값을 넣어주고 result를 mbox[1]으로 post해주고, time\_count=0으로 해준다.

4-3. 7이아니라면 아직 게임 중으로 led를 on\_cnt만큼 shift해 켜준다.

**BuzzerTask**



1. b\_grp eventflag 신호가 올때까지 pend한다.
2. TIMSK=0x40 으로 interrupt를 enable해주고, TCNT2=melody[mel\_idx]해주고, 음악을 출력해준다.
3. L\_grp event flag를 post해준다.

**IPC**



**코드 진행**

1. Main 함수에서 각 레지스터 값 초기화, IPC 초기화, task 생성을 한다.
2. os에게 제어권을 넘긴다.
3. FndTask, BuzzerTask의 우선순위가 가장 높지만 두 task는 IPC를 pend 하므로 그 다음 우선순위인 leveltask가 먼저 실행된다.
4. Interrupt 발생을 받아들이며 ISR을 실행하며 volatile 변수가 변경됨에 따라, 이를 확인하는 조건문 내에서 각각 mail box, message queue, event flag로 신호를 보낸다.
5. IPC의 신호를 받은 task가 실행된다.