실험.실습 보고서

(3) 반

조원: (학번:12131489 ,이름:김영훈)

1. 제목

4주차 실습과제 - MicroC/OS-II 스케줄러 이해하기

2. 목적

임베디드 시스템에서 실시간성을 보장하기 위한 선형시간 스케줄링을 할 수 있는 방법에 대해 이해한다. 또한 MicroC/OS-II에서 스케줄링에 필요한 data structure를 이해하고 사용법을 익힌다.

3. 실습에 필요한 기초지식

비트마스크를 사용할 줄 알아야 하고, 스케줄링에 대해 지식이 요구된다. 임베디드에서 real-time system이 왜 필요한지에 대한 이해가 필요하다. 또한 수행시간을 줄이기 위한 코드작성을 위해, 코드에 대한 수행시간을 분석할 줄 알아야한다.

4. 실습 절차, 내용 및 결과

프로그램 실행 시 main 함수부터 시작된다. main 함수에서 TaskStart 함수의 Task를 생성한다. TaskStart 함수에서는 TaskStartCreateTasks 함수를 호출하고 이 함수에서는 task 함수를 Task로 생성한다. task 함수는 4개의 0 ~ 63개의 숫자에서 final 변수보다 작은 최솟값을 꺼내 색을 칠해 화면에 보여주는 역할을 하는 task이다.

숫자 삽입

변수 myRdyGrp 과 배열 myRdyGrp 을 이용하여 어떤 숫자가 들어왔는지 표시를 한다. 랜덤번호가 생성 되었을 때, 총 6 비트 중에 앞 3 비트로 myRdyGrp 의 어느 한 비트를 1 로 만든다. 이 작업을 위해서는 앞 3 비트의 값을 myRdyGrp 에 어떤 비트를 1 로

해야 할지 알기 위해 OSMapTbl 을 이용해야한다. 앞 3 비트를 y 라 할때

myRdyGrp |= OSMapTbl[y]

이렇게 코드를 작성하면 myRdyGrp 에서 y 번째 비트의 값이 1 이 된다.

그리고 myRdyTbl 을 설정해야 하는데, OSMapTbl[y]번째 인덱스이므로 (뒤 3 비트 x)

myRdyTbl[OSMapTbl[y]] |= OSMapTbl[x]

이런 방법으로 들어온 숫자를 ready group&table 에 값이 들어왔음을 표시할 수 있다.

최소값 찾기

OSUnMaptbl 을 이용하면 myRdyTbl 에 있는 값 중에서 선형시간에 최소값을 찾을 수 있다. 최소값이 존재하는 배열의 번호(앞 3 비트 값)

value_y = OSUnMaptbl[OSRdyGrp]

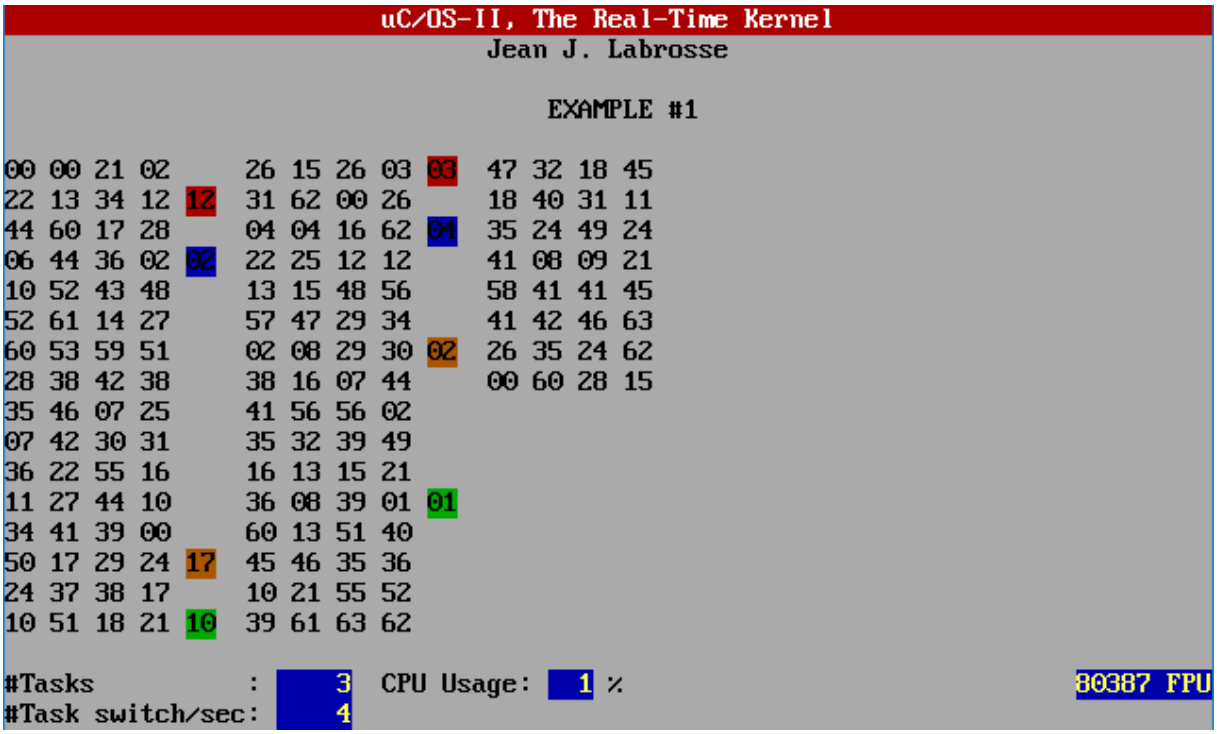
이렇게 추출된 번호로 OSRdyTbl[value_y]의 값을 알아낸다(뒤 3 비트 값)

value_x = OSUnMaptbl[OSRdyTbl[value _y]]

value_x, value_y 를 결합하여 최소값을 만들 수 있다.

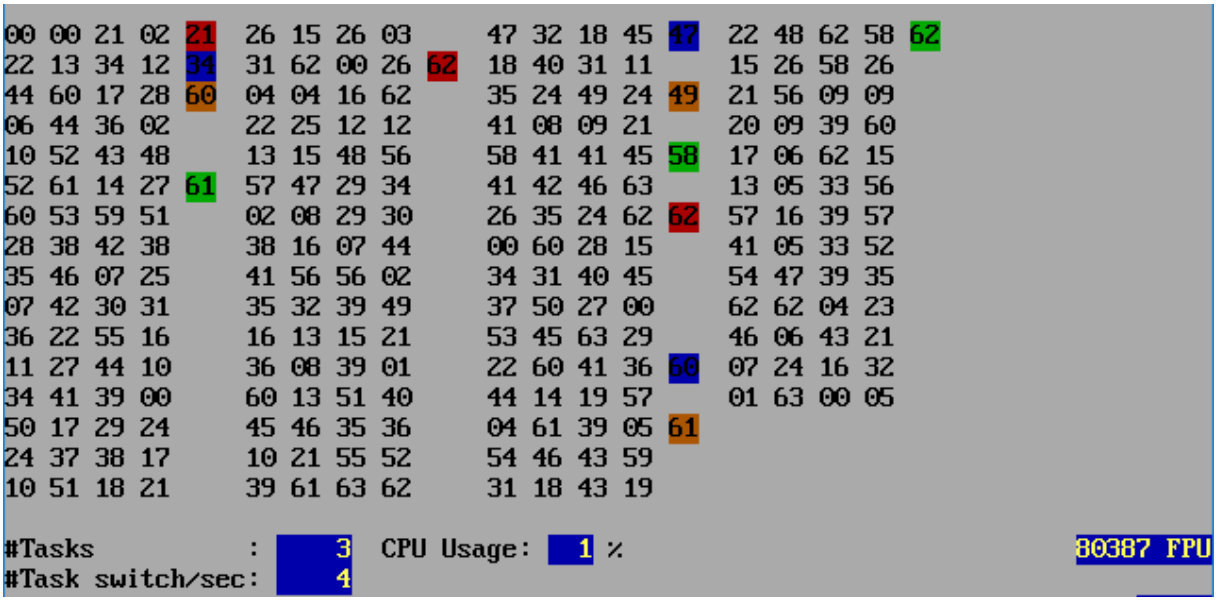
min = (value_y << 3) | value_x

최소값이 0 일 경우 loop_cnt 값을 증가시킨다. loop_cnt 가 4 일 경우 loop 를 종료하고 task 가 종료 된다. 0 이 아니면서 final 보다 min 이 작을 경우, min 을 화면에 띄워주고 색깔을 바꾸고 final 을 min 값으로 갱신한다. 아래 사진은 loop 가 끝나 종료된 모습이다.



optional assignment

원래 소스에서 몇 가지 부분을 수정하여 만들었다. 숫자를 4개 삽입하고 다시 최솟값부터 4개를 꺼낸다. 꺼낼 때 마지막에 꺼낸 숫자는 그 수들 중에서 최대값이 된다. 그 최대값을 final 값과 비교하고 더 클 경우 색깔을 입혀 출력시킨다.



5. 결론

이번 실습에서는 여러 숫자 중에서 최솟값을 찾는 프로그램을 작성하였다. 이것을 잘 활용하면 최솟값을 꺼내는 것을 가장 높은 우선순위를 가지는 task를 실행시키는 것으로 변형할 수 있을 것이다. 또한 들어온 4개의 task 중에서 final보다 우선순위가 높은 task가 있을 때만 context switch를 시키는 것을 적용할 수 있다. 이를 활용하여 실시간으로 우선순위가 높은 task를 먼저 실행시킬 수 있을 것이다.

이번 실습을 통해 real-time-system을 구성하는 방법과 원리를 배웠다. 그리고 어떻게 scheduling하는 것이 좋을지 생각하게 되는 시간을 가질 수 있었다.