**강의명: 실시간 커널**

**실습 번호: 3**

**실습 제목: Timer(타이머)**

**학생 이름: 황귀훈**

**학번: 201710885**

**1. 타이머를 사용한 시계**

**1.1 태스크 함수 TaskStart()**

**INT32U min, sec;**

**Int key;**

**OS\_EVENT \*Sem;**

**OS\_TMR \*Timer;**

**====================================================**

**void TaskStart(void \*pdata)**

**{**

**INT8U err, serr, clock;**

**min = 0;**

**sec = 0;**

**OSSoCInit(); // The first task should call this.**

**OSStatInit();**

**Sem = OSSemCreate(1);**

**Timer = OSTmrCreate(0, 10, OS\_TMR\_OPT\_PERIODIC, timer\_callback, (void \*)0, &clock, &err);**

**OSTaskCreate(DisplayClock, (void\*)0, &DisplayClockStk[TASK\_STK\_SIZE-1], 20);**

**OSTmrStart((OS\_TMR \*)Timer, &err);**

**UARTputgoto(0,0);**

**UARTprintf("uC/OS-II: clock");**

**UARTputgoto(0,13);**

**UARTprintf("Type \'M\'/\'m\' or \'S\'/\'s\' to adjust min and sec.");**

**for(;;)**

**{**

**Key = 1;**

**OSSemPend(Sem, 0, &serr);**

**UARTputgoto(0,1);**

**UARTprintf("Time=%d%d:%d%d, Tasks=%d, CPU=%d%%", min/10, min%10, sec/10, sec%10, OSTaskCtr, OSCPUUsage);**

**OSSemPost(Sem);**

**While(key!=-1){**

**key = UARTgetc\_nb();**

**if(key == 'M'){**

**if(min==59)**

**min=0;**

**else**

**min++;**

**}**

**else if(key == 'm'){**

**if(min==0) min=59;**

**else min--;**

**}**

**else if(key == 'S'){**

**if(sec==59)**

**sec=0;**

**else**

**sec++;**

**}**

**else if(key == 's'){**

**if((sec==0) && (min==0)){**

**min=59;**

**sec=59;**

**}**

**else if((sec==0) && (min!=0)){**

**min--;**

**sec=59;**

**}**

**else{**

**sec--;**

**}**

**}**

**else if((key =='\r') || (key == '\n')){**

**UARTputclear();**

**UARTputgoto(0,0);**

**UARTprintf("uC/OS-II: clock");**

**UARTputgoto(0,13);**

**UARTprintf("Type \'M\'/\'m\' or \'S\'/\'s\' to adjust min and sec.");**

**}**

**}**

**If(min==60) min = 0;**

**OSTimeDly(100);**

**}**

**}**

전체 코드의 처음 부분에 전역 변수로 min, sec, key(=UARTgetc\_nb로 입력을 받기 위한 변수)를 선언했다.

또한, Sem이라는 세마포를 생성하기 위해 OS\_Event type의 변수 Sem을 선언해주고자 OS\_Event \*Sem; 과 같이 작성해주었으며, Timer라는 이름의 타이머를 생성하기 위해 OS\_TMR type의 변수 Timer를 선언해주고자 OS\_TMR \*Timer를 선언했다.

이 프로그램은 2개의 태스크인 TaskStart() 및 DisplayClock()으로 구성된다. 위에 작성한 함수는 TaskStart()함수로 아래 조건을 만족하는 타이머를 생성하고, 생성된 타이머를 시작 시키고, 태스크 DisplayClock()를 생성한 후 화면 위치 (0,0)에 "uC/OS-II: clock"을, 화면 위치 (0,13)에 "Type 'M'/'m' or 'S'/'s' to adjust min and sec."을 출력한 후, 1초에 1회씩 정기적으로 화면 위치 (0,1)에 현재 시계(분:초), task 수, CPU usage를 출력해야 한다. 또한 태스크 TaskStart()는 키보드에서 입력하는 문자 중 아래 문자를 인식하여 해당 작업을 수행한다. ('M': 현재 분을 1 증가, 'm': 현재 분을 1 감소, 'S': 현재 초를 1 증가, 's': 현재 초를 1 감소, Enter 키: 전제 화면 다시 디스플레이, 이외 문자: 그 문자로 시계 출 력) 다음과 같이 프로그래밍 하기 위해 먼저 OSTmrCreate()과 OSTmrStart( )의 동작시 오류가 발생할 경우 오류에 대한 포인터가 필요하므로 확인용 변수로 err과 serr을 선언했다.

OSSemCreate(1)함수를 사용하여 초깃값 1로 세마포를 생성했다. 세마포를 사용하는 이유는 TaskStart()의 UARTprintf()와 DisplayClock()의 UARTprintf()함수를 서로 상호배제 하기 위함이다.

타이머를 생성하기 위해 OSTmrCreate()함수를 사용하여 Timer라는 이름의 타이머를 생성했다. 타이머 생성 함수의 매개변수로 delay, period, option, callback, callback\_arg, pname, perr를 주었고 이때 과제 조건에 맞게 delay=10, period=10, option=OS\_TMR\_OPT\_PERIODIC으로 지정하였다.

또한, 타이머가 만료되면 호출할 callback 함수의 이름인 timer\_callback을 적었으며 타이머가 만료될때 callback함수에 전달되는 인자는 없기 때문에 null포인터로 작성하였다.

이렇게 만든 타이머의 이름을 지정할 수 있는 pname에는 위에서 선언한 &clock를 작성했다.

마지막으로 오류 코드에 대한 포인터로 &err를 작성하는 것으로 타이머를 생성했다.

그리고 OSTaskCreate()함수를 사용하여 DisplayClock 태스크를 생성 하는데 이 때 4개의 인수를 받는다. task: 생성하고자 하는 task 함수이므로 DisplayClock를 작성, pdata: task 변수의 pointer 이므로 딱히 없기 때문에 (void)\* 0으로 작성, ptos: 해당 task stack의 top의 주소이므로 [TASK\_STACK\_SIZE -1]작성, prio: 태스크의 우선순위이다. 이 경우, TaskStart의 priority가 10이었기에 그보다 더 아래인 20으로 설정하였다. 또한 타이머를 동작하기 위해 OSTmrStart()함수를 사용해야한다. 따라서 OSTmrStart((OS\_TMR\*)Timer, &err)를 통해 Timer라는 이름을 가진 타이머를 시작했다. 이때 타이머는 oo:oo 부터 시작하기 때문에 분과 초를 나타내는 전역변수 min, sec는 미리 0으로 초기화 해주었다.

처음 화면 위치 (0,0)에 "uC/OS-II: clock"을, 화면 위치 (0,13)에 "Type 'M'/'m' or 'S'/'s' to adjust min and sec."을 출력하기 위해 UARTputgo()함수를 사용하여 출력해야 할 위치로 커서를 이동한 뒤 UARTprint()함수를 사용하여 주어진 문장을 출력했다.

마지막으로 태스크 TaskStart()는 1초에 1회씩 정기적으로 화면 위치 (0,1)에 현재 시계(분:초), task 수, CPU usage를 출력해야 한다. 따라서 for(;;)를 통해 무한루프를 생성한 뒤 타이머를 출력해서 보여주는 DisplayClock()의 printf문과의 상호배제를 위해 OSSemPend(Sem,0,&serr)를 통해 위에서 선언했던 세마포 Sem를 UARTprintf문 위에서 pend하였다. 그 후 UARTputgo()함수를 사용하여 Time, task 수, CPU를 출력해주는 코드를 작성하였다. TaskStart()의 UARTprintf()함수와 DisplayClock()의 UARTprintf()함수를 상호배제 하는 것 이므로 UARTprint()문을 수행한 뒤 OSSemPost(Sem)를 작성했다. 1초마다 출력해야 하기 때문에 for(;;) 마지막에 OSTimeDly(100)를 사용하였다. (tick은 1/100초이므로 1초를 딜레이하기 위해서는 100을 작성)

또한 키보드에서 입력하는 문자 중 아래 문자를 인식하여 해당 작업을 수행해야 되기 때문에, ('M': 현재 분을 1 증가, 'm': 현재 분을 1 감소, 'S': 현재 초를 1 증가, 's': 현재 초를 1 감소, Enter 키: 전제 화면 다시 디스플레이, 이외 문자: 그 문자로 시계 출 력) 키보드에서 입력하는 값은 UARTgetc\_nb()함수를 사용하여 전역변수 key에 저장하여 if 조건문을 사용해 각각의 키보드 입력에 해당하는 기능을 수행하기위해 프로그램을 작성하였다. 이 때 UARTgetc\_nb()가 한번 수행될 때 마다 버퍼에서 하나를 꺼내기 때문에 쌓여 있는 키입력을 모두 계산하기 위해 while()문을 사용하였다. 이 때 while문의 조건으로 key!=-1을 준 이유는 버퍼에서 모든 c를 꺼내면 -1이 되기 때문에 다 꺼낼 때 까지 while문을 돌린다는 것을 의미한다.

만약, key의 값이 ‘M’이라면 현재 분을 1 증가해야 한다. 따라서 전역 변수로 설정했던 min을 1 증가시키기 위해 min++를 작성했다. 이때 59분 다음은 60분이 아니라 00분이 되도록 하였다. 만약 key에 할당된 값이 문자 ‘M’이 아닌 문자 ‘m’라면 이는 현재 분을 1 감소해야 한다. 따라서 else if 조건문을 사용하여 key가 ‘m’인지 확인하고 해당 key가 ‘m’이라면 min--을 하는 작업이 필요하다. 이때, 예외적으로 min이 0인 상태라면 (min==0)라면 --1이 아니라 59로 설정해야 한다. 따라서, else if 조건문 안에 min이 0인지 아닌지를 확인하기 위한 조건문을 중첩하여 min이 0이라면 min을 59로 할당하고 그렇지 않다면 min--하도록 코드를 작성하였다.

만약 key에 할당된 값이 문자 ‘S’라면 현재 초를 1 증가해야 한다. 따라서 전역 변수로 설정했던 sec를 1 증가시키기 위해 sec++를 작성했다. 이때 59초 다음은 60초가 아니라 00초가 되도록 하였다. 만약 key에 할당된 값이 문자 ‘s’라면 이는 현재 초를 1 감소해야 한다. 이때, 위의 과정과 마찬가지로 추가적인 조건문이 필요하다.

만약, 분과 초가 모두 0일 때, 초를 1 감소할 경우엔 59:59로 출력해야 하므로 min과 sec에 각각 59의 값을 할당했다. 만약, 분은 0이 아니지만 초가 0일 때, 초를 1 감소할 경우엔 min은 1감소하고 sec만 59로 출력해야 하므로 min--, sec에만 59의 값을 할당했다. 마지막으로 위의 조건모두에 해당하지 않는다면 초만 1 감소하면 되기 때문에 sec--와 같이 코드를 작성하였다.

Enter 키를 입력하면 전제 화면을 다시 디스플레이 해야 하는데 ‘\n’은 newline으로 다음줄로 이동하는 것을 의미하고 ‘\r’은 carriage return으로 그 줄 맨 앞으로 이동하는 것을 의미한다. enter키를 사용할 때, 그 다음 줄 맨 앞으로 이동을 해야 하므로 ‘\n’ 와 ‘\r’이 하나만 나오더라도 전체 화면을 다시 디스플레이 하도록 작성해야 하기 때문에 조건문 if를 통해 If ((c == ‘\n’) || (c==‘\r’))의 조건을 먼저 설정하고 코드를 작성하였다.

화면을 다시 디스플레이하기 위해 먼저 화면을 다 지우는 UARTputclear()를 수행하였다.

또한, 커서의 위치를 (0,0)로 변경하기 위해 UARTputgoto(0,0)를 수행했다. 이렇게 clear된 화면에 (0,0)와 (0,13)에 해당 문구들을 UARTprintf문을 통해 출력하도록 작성했다.

**1.2 태스크 함수 DisplayClock()**

**void DisplayClock(void \*pdata)**

**{**

**INT32U m1,m2,s1,s2;**

**INT32U i, j, asc = 79;**

**INT8U serr;**

**for(;;){**

**m1 = min/10, m2 = min%10;**

**s1 = sec/10, s2 = sec%10;**

**if((key != -1) && (key !='M') && (key != 'm') && (key !='S') &&(key != 's'))**

**asc = key;**

**else**

**OSTimeDly(0);**

**OSSemPend(Sem, 0, &serr);**

**UARTputgoto(0,3);**

**for(i=3; i<12; i++){**

**for(j=0; j<8; j++)**

**{**

**if(font\_table[m1+16].font[i-3][j]==0)**

**UARTprintf(" ");**

**else if(font\_table[m1+16].font[i-3][j]==1)**

**UARTprintf("%c", asc);**

**}**

**UARTputgoto(0,i+1);**

**}**

**UARTputgoto(8,3);**

**for(i=3; i<12; i++){**

**for(j=8; j<16; j++)**

**{**

**if(font\_table[m2+16].font[i-3][j-8]==0)**

**UARTprintf(" ");**

**else if(font\_table[m2+16].font[i-3][j-8]==1)**

**UARTprintf("%c", asc);**

**}**

**UARTputgoto(8,i+1);**

**}**

**UARTputgoto(16,3);**

**for(i=3; i<12; i++){**

**for(j=16; j<24; j++)**

**{**

**if(font\_table[26].font[i-3][j-16]==0)**

**UARTprintf(" ");**

**else if(font\_table[26].font[i-3][j-16]==1)**

**UARTprintf("%c", asc);**

**}**

**UARTputgoto(16,i+1);**

**}**

**UARTputgoto(24,3);**

**for(i=3; i<12; i++){**

**for(j=24; j<32; j++)**

**{**

**if(font\_table[s1+16].font[i-3][j-24]==0)**

**UARTprintf(" ");**

**else if(font\_table[s1+16].font[i-3][j-24]==1)**

**UARTprintf("%c", asc);**

**}**

**UARTputgoto(24, i+1);**

**}**

**UARTputgoto(32,3);**

**for(i=3; i<12; i++){**

**for(j=32; j<40; j++)**

**{**

**if(font\_table[s2+16].font[i-3][j-32]==0)**

**UARTprintf(" ");**

**else if(font\_table[s2+16].font[i-3][j-32]==1)**

**UARTprintf("%c", asc);**

**}**

**UARTputgoto(32, i+1);**

**}**

**OSSemPost(Sem);**

**OSTimeDly(100);**

**}**

**}**

함수 시작에 Time = ab:cd와 같이 출력하기 위해 각 자리를 나타내는 변수 m1, m2, s1, s2를 선언하고 의미에 맞게 m1 = min/10, m2 = min%10, s1 = sec/10, s2 = sec%10으로 초기화 했다. 프로그램의 조건으로 첫 출력은 대문자 O로 출력해야 되기 때문에 변수 asc를 대문자 O에 해당하는 ASCII값인 79로 초기화 했다. 아래에서 for문에 사용할 변수로 I,j도 선언 했다.

if문을 통해 TaskStart()에서의 UARTgetc\_nb()함수로 입력 받은 key의 값이 -1(값이 없는 것)과 문자‘M’ ‘m’ ‘S’ ‘s’들이 모두 아닐 경우에만 폰트 출력을 위한 변수 asc에 key값을 할당하도록 설정하였다. (M,m,S,s는 시간을 조정하는 역할로만 사용하기 위해)

또한, 입력이 없을 때 공백이 출력되는데 OSTimeDly(0)를 작성함으로써 이를 방지했다.

마지막으로 디스플레이를 출력하기 전에 세마포를 통해 상호배제를 해야 한다. DisplayClock()의 UARTprintf()함수의 출력과 위에서의 TaskStart()의 UARTprintf()함수의 출력이 서로 겹치면 안되기 떄문에 DisplayClock()의 출력과정이 시작되기 직전에 TaskStart()로부터 받은 Sem을 이용하여 OSSemPend()를 선언했다.

디스플레이에 Time을 출력하는 알고리즘은 다음과 같다.

(i=9행, j=8열)로 설정되어 있는 상태에서 배열의 열을 한 줄 씩 비교하면서 해당 위치의 값이 0 이면 공백으로 출력, 1이면 해당 값을 출력한다. 이때, 첫번째 가로의 비교가 끝나면 UARTputgoto()를 통해 그 다음줄로 커서를 옮겨서(= 행을 한 칸 씩 내려오면서) 비교를 통한 출력을 반복해서 진행한다.

<m1 출력>

먼저, 처음 시작 커서가 과제 조건에서 (0,3)으로 되어있기 때문에 UARTputgoto()를 이용하여 해당위치로 커서를 이동시켰다. 이때, i는 9만큼의 길이를 가지기 때문에 첫번째 for문에서의 조건을 3부터 12(3+9)으로 설정하였다. 이렇게 설정된 for문안에서 열을 한 줄 씩 비교 해야 하기 때문에 j에 대한 for문을 중첩해서 작성하였다. j는 8만큼의 길이를 가지기 때문에 j=0; j<8까지를 조건으로 진행하였다.

이렇게 중첩적으로 설정된 조건문 내에서 만약 font\_table[m1+16].font[i-3][j]가 0이라면 공백을 출력하고 1이라면 UARTprintf()문을 이용하여 UARTgetc\_nb()를 통해 입력 받은 전역변수 key의 값이 할당된 변수 asc를 해당 문자로 출력하도록 하였다.

여기서 조건문에 font\_table[m1+16].font[i-3][j]와 같이 작성한 이유는 다음과 같다. 먼저, m1에 16이 더해진 이유는 fonts.h에서 배열 font\_table[]의 16번째 요소가 숫자인 0이 시작되는 부분이기 때문이다. 0,1,2…9까지의 문자로만 분과 초를 표현할 수 있기 때문에 +16을 더해주는 것이다.

Ex) m1이 0이라면 m1+16=16이기에 16번째 요소의 1,0을 비교하여 1의 부분만 출력

또한, font[i-3][j]의 경우를 살펴보자. 비교를 위해서는 배열의 [0,0]부터 시작해야 하는데 우리가 위에서 i의 조건문을 3부터 시작하였기 때문에 값을 조정해주기 위해 -3을 해주었다.

이와 같은 비교를 통해 font의 ‘1’에 해당하는 부분만 문자로 출력했다면 그 다음 줄로 가기 위한 목적으로 UARTputgoto(0,i+1)를 작성했다.

<m2 출력>

먼저, m1의 디스플레이의 해당 출력 부분은 가로를 기준으로 0~7까지 총 길이 8만큼이다. 따라서 m2의 출력이 시작되는 부분은 (8,3)이기 때문에 UARTputgoto(8,3)을 통해 커서를 해당 위치로 이동시켰다. 이때, i는 9만큼의 길이를 가지기 때문에 첫번째 for문에서의 조건을 3부터 12(3+9)으로 설정하였다. 이렇게 설정된 for문안에서 가로를 한 줄 씩 비교해야 하기 때문에 j에 대한 for문을 중첩해서 작성하였다. j는 8만큼의 길이를 가지기 때문에 j=8; j<16를 조건으로 진행하였다.

이렇게 중첩적으로 설정된 조건문 내에서 만약 font\_table[m1+16].font[i-3][j-8]가 0이라면 공백을 출력하고 1이라면 UARTprintf()함수를 이용하여 UARTgetc\_nb()를 통해 입력 받은 전역변수 key의 값이 할당된 변수 asc를 해당 문자로 출력하도록 하였다.

여기서 font\_table[m2+16].font[i-3][j]의 m2에 16이 더해진 이유는 위의 m1과 마찬가지로 fonts.h에서 배열 font\_table[]의 16번째 요소가 숫자인 0이 시작되는 부분이기 때문이다.

또한, font[i-3][j-8]의 경우를 살펴보자. 비교를 위해서는 배열의 [0,0]부터 시작해야 하는데 우리가 위에서 i의 조건문을 3부터, j의 조건문을 8부터 시작했기 때문에 값을 조정해주기 위해 -3과-8을 해준 것이다. 이와 같은 비교를 통해 ‘1’에 해당하는 부분만 문자로 출력했다면 그 다음 줄로 가기 위한 목적으로 UARTputgoto(8,i+1)를 작성했다.

<: 출력>

먼저, m2의 디스플레이의 해당 출력 부분은 가로를 기준으로 8~15까지 총 길이 8만큼이다. 따라서’:’의 출력이 시작되는 부분은 (16,3)이기 때문에 UARTputgoto(16,3)을 통해 커서를 해당 위치로 이동시켰다. 이때, i는 9만큼의 길이를 가지기 때문에 첫번째 for문에서의 조건을 3부터 12(3+9)으로 설정하였다. 이렇게 설정된 for문안에서 가로를 한 줄 씩 비교해야 하기 때문에 j에 대한 for문을 중첩해서 작성하였다. j는 8만큼의 길이를 가지기 때문에 j=16; j<24 까지를 조건으로 진행하였다. 이렇게 중첩적으로 설정된 조건문 내에서 만약 font\_table[26].font[i-3][j-16]가 0이라면 공백을 출력하고 1이라면 UARTprintf()문을 이용하여 UARTgetc\_nb()를 통해 입력 받은 전역변수 key의 값이 할당된 변수 asc를 해당 문자로 출력하도록 하였다.

여기서 font\_table[26].font[i-3][j-16]의 값이 26인 이유는 fonts.h에서 배열 font\_table[]의 26번째에 출력되어야 할 ‘:’이 존재하기 때문이다. 또한, font[i-3][j-16]의 경우를 살펴보자. 비교를 위해서는 배열[0.0]부터 시작해야 하는데 우리가 위에서 i의 조건문을 3부터, j의 조건문을 16부터 시작했기 때문에 값을 조정해주기 위해 -3과-16을 해준 것이다.

이와 같은 비교를 통해 ‘1’에 해당하는 부분만 문자로 출력했다면 그 다음 줄로 가기 위한 목적으로 UARTputgoto(16,i+1)를 작성했다.

<s1 출력>

먼저, ‘:’의 디스플레이의 해당 출력 부분은 가로를 기준으로 16~23까지 총 길이 8만큼이다. 따라서 s1의 출력이 시작되는 부분은 (24,3)이기 때문에 UARTputgoto(24,3)을 통해 커서를 해당 위치로 이동시켰다. 이때, i는 9만큼의 길이를 가지기 때문에 첫번째 for문에서의 조건을 3부터 12(3+9)으로 설정하였다. 이렇게 설정된 for문안에서 가로를 한 줄 씩 비교해야 하기 때문에 j에 대한 for문을 중첩해서 작성하였다. J는 8만큼의 길이를 가지기 때문에 j=24; j<32 까지를 조건으로 진행하였다. 이렇게 중첩적으로 설정된 조건문 내에서 만약 font\_table[s1+16].font[i-3][j-24]가 0이라면 공백을 출력하고 1이라면 UARTprintf()함수를 이용하여 UARTgetc\_nb()를 통해 입력 받은 전역변수 key의 값이 할당된 변수 asc를 해당 문자로 출력하도록 하였다.

이때, font\_table[s1+16].font[i-3][j-24]의 s1에 16이 더해진 이유는 위의 m1, m2과 마찬가지로 fonts.h에서 배열 font\_table[]의 16번째 요소가 숫자인 0이 시작되는 부분이기 때문이다.

또한, font[i-3][j-24]의 경우를 살펴보자. 비교를 위해서는 배열[0.0]부터 시작해야 하는데 우리가 위에서 i의 조건문을 3부터, j의 조건문을 24부터 시작했기 때문에 값을 조정해주기 위해 -3과 -24을 해준 것이다. 이와 같은 비교를 통해 ‘1’에 해당하는 부분만 문자로 출력했다면 그 다음 줄로 가기 위한 목적으로 UARTputgoto(24,i+1)를 작성했다.

<s2 출력>

먼저, s1의 디스플레이의 해당 출력 부분은 가로를 기준으로 24~31까지 총 길이 8만큼이다. 따라서 s2의 출력이 시작되는 부분은 (32,3)이기 때문에 UARTputgoto(32,3)을 통해 커서를 해당 위치로 이동시켰다. 이때, i는 9만큼의 길이를 가지기 때문에 첫번째 for문에서의 조건을 3부터 12(3+9)으로 설정하였다. 이렇게 설정된 for문안에서 가로를 한 줄 씩 비교해야 하기 때문에 j에 대한 for문을 중첩해서 작성하였다. j는 8만큼의 길이를 가지기 때문에 j=32; j<40 까지를 조건으로 진행하였다. 이렇게 중첩적으로 설정된 조건문 내에서 만약 font\_table[s2+16].font[i-3][j-32]가 0이라면 공백을 출력하고 1이라면 UARTprintf()함수를 이용하여 UARTgetc\_nb()를 통해 입력 받은 전역변수 key의 값이 할당된 변수 asc를 해당 문자로 출력하도록 하였다. 이때, font\_table[s2+16].font[i-3][j-32]의 s2에 16이 더해진 이유는 위의 s1과 마찬가지로 fonts.h에서 배열 font\_table[]의 16번째 요소가 숫자인 0이 시작되는 부분이기 때문이다. 또한, font[i-3][j-32]의 경우를 살펴보자. 비교를 위해서는 배열[0.0]부터 시작해야 하는데 우리가 위에서 i의 조건문을 3부터, j의 조건문을 32부터 시작했기 때문에 값을 조정해주기 위해 -3과 -32을 해준 것이다. 이와 같은 비교를 통해 ‘1’에 해당하는 부분만 문자로 출력했다면 그 다음 줄로 가기 위한 목적으로 UARTputgoto(32,i+1)를 작성했다.

이와 같은 m1 m2 : s1 s2의 출력이 모두 끝났으면 세마포를 다시 놓아주기 위해 OSSemPost(Sem)를 작성하였다. 이때, 1초에 한 번 씩 출력되도록 하는 것이 과제의 조건이므로 OSTimeDly(100)을작성했다. (1tick =1/100초이므로 1초를 나타내주기 위해 100을 작성)

**1.3 타이머 callback 함수**

**void timer\_callback(void \*ptmr, void \*parg)**

**{**

**sec++;**

**if(sec>=60){**

**min++;**

**sec=0;**

**}**

**if(min>=60)**

**min=0;**

**}**

타이머의 callback 함수는 현재 분 및 초를 수정하는 매우 짧은 작업을 수행해야 한다.

callback 함수는 타이머의 period(Ticks)이 0이 되면 실행되는 함수로 이 타이머의 callbak함수는 현재 분 및 초를 수정하는 작업을 진행한다. 타이머 1tick은 =1/10초 이므로 위에서 우리가 타이머의 period를 10으로 지정했다는 것은 사실상 1초의 시간을 의미한다. 따라서 타이머가 시간의 흐름에 따라 동작하기 위해 1초의 흐름이 필요하므로 period가 0이 되는 순간 초(=sec)를 1증가시킨다. 만약, sec가 60이상이라면 순간 분(=min)을 1 증가시키면서 sec를 0으로 만들어주는 작업을 진행한다. 만약, 분(=min)이 60이상이 된다면 min을 0으로 만들어주는 작업을 진행한다.

**1.4 프로그램 수행 초기 화면**

**테이블이(가) 표시된 사진

자동 생성된 설명**

**1.5 프로그램 수행 후 #입력 화면**

**테이블이(가) 표시된 사진

자동 생성된 설명**

**1.6 프로그램 수행 후 'M'/'m'/'S'/'s'입력, 기능 확인**

- 'M': 현재 분을 1 증가하는 기능을 한다.

이때, 분이 59인 상태에서 M의 기능을 구현했을 때는 분이 60이 아니라 00이 되도록 설정하였다.

- 'm': 현재 분을 1 감소하는 기능을 한다.

이때, 분이 0인 상태에서 m의 기능을 구현했을 때는 분을 59로 출력되도록 설정하였다.

- 'S': 현재 초를 1 증가하는 기능을 한다.

이때, 초가 59인 상태에서 S의 기능을 구현했을 때는 초를 60이 아니라 00이 되도록 설정하였다.

- 's': 현재 초를 1 감소하는 기능을 한다.

이 경우, 3가지의 조건을 설정했다.

1. 00:00인 상태라면 59:59로 구현되도록 설정
2. 초만 0에 해당하는 상태라면 분은 1감소시키고 초만 59로 출력하도록 설정
3. 위의 두가지에 해당하지 않는다면 초를 1감소시키도록 설정

키 입력 수행 결과 위의 기능을 올바르게 수행하는 것을 확인했다.

**1.7 화면 출력 확인**

화면에서 고정적으로 출력상태를 유지하는 (0,0)의 uC/OS-II: clock과 (0,13)의 Type 'M'/'m' or 'S'/'s' to adjust min and sec.을 제외한 출력은 무한루프 속에서 1초에 1번씩 출력 되어야한다.

이때, TaskStart()의 UARTprintf()함수와 DisplayClock()의 UARTprintf()함수를 상호배제 하여야 schared resource인 screen에서 출력이 깨지지 않고 정상적으로 나온다. 따라서 상호배제를 위한 목적으로 세마포를 사용하였다.

먼저, 세마포 Sem을 선언하면서 초기값을 1로 설정하였다. (Pend의 경우, Sem값이 양수이면 wait에서 빠져나오면서 Sem의 값을 1 감소시켜준다. 반면에 Sem값이 음수 또는 0일 경우 계속 기다린다) 이렇게 초기화된 세마포 Sem을 이용하여 TaskStart()의 UARTprintf()함수와 위에 OSSemPend(Sem, 0, &serr)를 작성하여 먼저 사용한 후, UARTprintf()함수의 출력이 끝남과 동시에 OSSemPost를 작성함으로써 DisplayClock()이 Sem을 사용할 수 있도록 놓아주었다.

이렇게 TaskStart()로부터 받은 세마포 Sem을 DisplayClock()에서 for문 안의 출력이 시작되는 부분에 OSSemPend(Sem, 0, &serr)를 작성하고 출력이 모두 끝나자마자 OSSemPost(Sem)을 작성하였다.

이와 같은 방식으로 하나의 세마포로 두 테스크를 상호배제 하였기 때문에 화면 출력 시 화면이 깨어지지 않고 잘 나온다.