임베디드 소프트웨어 lab 08 실습보고서

2011104043 장원엽

1. 수업 내용

파이프(두개의 프로세스가 부모, 자식 프로세스일 때), 피포(서로 다른 두 개의 프로세스간 데이터 통신 방법)

메모리 카피 오버헤드가 적어서 성능상 유리하다 = 쉐어드 메모리 모델

메시지 패싱은 커널이 중재하기 때문에 쉐어드 메모리가 좋다.

그러나 메시지 패싱은 메시지를 주고받는 두개의 프로세스간 동기화가 필요하다(커널이 알아서 해준다.)

쉐어드 메모리 방식은 어플리케이션에서 직접 동기화를 해주어야 한다. 세마포어, 뮤텍스 등의 동기화 툴을 이용해서 동기화를 직접 해주어야 한다.

프로세스간 데이터 통신에는 메시지 패싱 모델이 자연스럽다.

파이프나 피포를 사용한다.

스레드일 경우에는 여러 스레드들이 모두 전역변수로 데이터를 공유하기 때문에 쉐어드 메모리를 별도로 만들 필요가 없다. 이 경우 데이터 통신에 쉐어드 메모리를 사용하는데, 이때는 동기화 수단이 필요하다.

마이크로 프로세스는 테스크 모델에 더 가깝기 때문에 쉐어드 모델로 만들 필요가 없다.

메시지 메일 박스는 데이터 하나만 주고받는 경우

메시지 큐는 데이터 여러 개를 주고받는 경우

1. 실습

다음의 프로그램을 위의 두가지 ITC 방법으로 구현: lab8.zip

온도 센서로부터 1초에 한번씩 데이터를 읽는 TemperatureTask (JKit 실습자료 9장 참조)

TemperatureTask로부터 온도를 전달받는 FndTask (두가지 방법)

온도를 FND에 출력하는 FndDisplayTask

#include "includes.h"

#define F\_CPU 16000000UL // CPU frequency = 16 Mhz

#include <avr/io.h>

#include <util/delay.h>

#define TASK\_STK\_SIZE OS\_TASK\_DEF\_STK\_SIZE

#define N\_TASKS 4

#define MBOX 0

#define GVAR 1

#define METHOD MBOX

#define DEBUG 0

OS\_STK TaskStk[N\_TASKS][TASK\_STK\_SIZE];

#if METHOD == MBOX

OS\_EVENT \*Mbox;

#else

OS\_EVENT \*Sem;

volatile INT8U TemperatureValue;

#endif

OS\_EVENT \*Mutex;

volatile INT8U FndNum;

void TemperatureTask(void \*data);

void FndTask(void \*data);

void FndDisplayTask(void \*data);

void BuzzerTask(void \*data);

int main (void)

{

OSInit();

OS\_ENTER\_CRITICAL();

TCCR0=0x07;

TIMSK=\_BV(TOIE0);

TCNT0=256-(CPU\_CLOCK\_HZ/OS\_TICKS\_PER\_SEC/ 1024 );

OS\_EXIT\_CRITICAL();

OSTaskCreate(TemperatureTask, (void \*)0, (void \*)&TaskStk[0][TASK\_STK\_SIZE - 1], 1);

OSTaskCreate(FndTask, (void \*)0, (void \*)&TaskStk[1][TASK\_STK\_SIZE - 1], 3);

OSTaskCreate(FndDisplayTask, (void \*)0, (void \*)&TaskStk[2][TASK\_STK\_SIZE - 1], 4);

OSTaskCreate(BuzzerTask, (void \*)0, (void \*)&TaskStk[3][TASK\_STK\_SIZE - 1], 2);

OSStart();

return 0;

}

void InitI2C()

{

PORTD = 3; // For Pull-up override value

SFIOR &= ~(1 << PUD); // PUD

TWSR = 0; // TWPS0 = 0, TWPS1 = 0

TWBR = 32; // for 100 K Hz bus clock

TWCR = \_BV(TWEA) | \_BV(TWEN); // TWEA = Ack pulse is generated

// TWEN = TWI 동작을 가능하게 한다

}

int ReadTemperature(void)

{

int value;

OS\_ENTER\_CRITICAL();

TWCR = \_BV(TWSTA) | \_BV(TWINT) | \_BV(TWEN);

while(!(TWCR & \_BV(TWINT)));

TWDR = 0x98 + 1; //TEMP\_I2C\_ADDR + 1

TWCR = \_BV(TWINT) | \_BV(TWEN);

while(!(TWCR & \_BV(TWINT)));

TWCR = \_BV(TWINT) | \_BV(TWEN) | \_BV(TWEA);

while(!(TWCR & \_BV(TWINT)));

//온도센서는 16bit 기준으로 값을 가져오므로

//8비트씩 2번을 받아야 한다.

value = TWDR << 8;

TWCR = \_BV(TWINT) | \_BV(TWEN);

while(!(TWCR & \_BV(TWINT)));

value |= TWDR;

TWCR = \_BV(TWINT) | \_BV(TWEN) | \_BV(TWSTO);

value >>= 8;

TIMSK = (value >= 33) ? TIMSK | \_BV(TOIE2): TIMSK & ~\_BV(TOIE2);

OS\_EXIT\_CRITICAL();

return value;

}

void TemperatureTask (void \*data)

{

int value;

INT8U err;

data = data;

#if METHOD == MBOX

Mbox = OSMboxCreate(0);

#else

Sem = OSSemCreate(0);

#endif

Mutex = OSMutexCreate(0, &err);

InitI2C();

DDRA = 0xFF;

while (1) {

value = ReadTemperature();

#if METHOD == MBOX

OSMboxPost(Mbox, &value);

#else

OSMutexPend(Mutex, 0, &err);

TemperatureValue = value;

OSMutexPost(Mutex);

OSSemPost(Sem);

#endif

OSTimeDlyHMSM(0, 0, 1, 0);

}

}

void FndTask (void \*data)

{

INT8U err;

data = data;

while (1) {

#if METHOD == MBOX

FndNum = \*(int \*)OSMboxPend(Mbox, 0, &err);

#else

OSSemPend(Sem, 0, &err);

OSMutexPend(Mutex, 0, &err);

FndNum = TemperatureValue;

OSMutexPost(Mutex);

#endif

#if DEBUG == 1

PORTA = FndNum;

#endif

}

}

void FndDisplayTask (void \*data)

{

unsigned char FND\_DATA[ ]= {

0x3f, // 0

0x06, // 1

0x5b, // 2

0x4f, // 3

0x66, // 4

0x6d, // 5

0x7d, // 6

0x27, // 7

0x7f, // 8

0x6f, // 9

0x77, // A

0x7c, // B

0x39, // C

0x5e, // D

0x79, // E

0x71, // F

0x80, // .

0x40, // -

0x08 // \_

};

unsigned int num0, num1, num2, num3;

INT8U err;

data = data;

DDRC = 0xff;

DDRG = 0x0f;

while(1) {

OSMutexPend(Mutex, 0, &err);

num3 = (FndNum / 1000) % 10;

num2 = (FndNum / 100) % 10;

num1 = (FndNum / 10) % 10;

num0 = FndNum % 10;

OSMutexPost(Mutex);

PORTC = FND\_DATA[num3];

PORTG = 0x08;

\_delay\_ms(2);

PORTC = FND\_DATA[num2];

PORTG = 0x04;

\_delay\_ms(3);

PORTC = FND\_DATA[num1];

PORTG = 0x02;

\_delay\_ms(2);

PORTC = FND\_DATA[num0];

PORTG = 0x01;

\_delay\_ms(3);

}

}

void BuzzerTask(void \*data){

data = data;

int i=0;

DDRB = 0x10; // 포트 B의 bit4 를 입력 상태로 세팅

while(1)

{

if(ReadTemperature() >= 32){

for(i=0; i<100; i++){

PORTB = 0x10; // 1ms 동안 ‘On’ 상태 유지

\_delay\_ms(1);

PORTB = 0x00; // 1ms 동안 ‘Off’ 상태 유지

\_delay\_ms(1);

}

}

OSTimeDly(10);

}

}