Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2006

Se desea desarrollar un sistema de Tiempo Real para la lectura y comprobación de valores de un conjunto de canales analógicos de entrada (A/D), mientras se genera una onda senoidal en un canal analógico de salida (D/A).

Este sistema de TR constará de dos hilos (principal y H2) y una función de activación asíncrona F1. El hilo principal hará las inicializaciones y se queda a la espera de entradas del usuario por teclado. El hilo H2 se activará periódicamente para generar la onda senoidal. La función asíncrona se activará cuando haya nuevos datos en los canales analógicos, y procederá a su procesamiento.

1) *2.5 puntos*

Escribir el programa principal, que recibe 2 parámetros en línea de comandos: el periodo Tm de activación periódica del hilo 2 (en milisegundos) y el nº de canales analógicos a leer. El programa principal, en este orden:

- Vacía el archivo `"salida.txt"`, abriéndolo en modo escritura/texto y cerrándolo inmediatamente.
- Arranca el hilo H2, pasándole como parámetro el periodo Tm de activación periódica (ver doc. en última página).
- Llama a la función de librería de inicialización de conversiones A/D con aviso por función callback:

```
InitAD(int ncanales,int Tm,void (*FnAsincrona)() );
```

- Se queda en un bucle de espera a que el usuario pulse una tecla; si la tecla es 'X' termina la ejecución del programa; si es 'A', pide el valor de la amplitud de la onda; si es 'T', pide el valor del periodo de la onda (las variables amplitud y periodo deben ser accesibles desde el hilo H2); si la tecla es '?', llama a una función que lee las líneas del archivo `"salida.txt"` y escribe en pantalla el nº de líneas que contienen error tipo 1 (o sea, 1 ó 3 - ver pregunta siguiente), error tipo 2 y no error (ok).

2) *2.5 puntos*

Escribir el código de la función de activación asíncrona F1, la cual se activa automáticamente cuando han terminado las conversiones de los canales A/D, o sea, cada Tm milisegundos. Dicha función debe:

- Llamar a la función de librería `LeeAD(float* tabla,int ncanales);` la cual lee sobre la tabla indicada los valores actuales de los `ncanales` solicitados. No se puede suponer en tiempo de compilación el número máximo de canales.
- Calcular la media y desviación típica de los valores leídos de los distintos canales.
- Si la desviación típica supera el valor 1, se pone a 1 una variable para indicar condición de error ; si la media calculada difiere de la media calculada en la activación anterior en un valor superior a 10, dicha variable se pone a 2; si suceden ambas cosas, la variable valdrá 3.
- Añadir al archivo de texto `"salida.txt"` una nueva línea con los valores de media, desviación típica, y el texto "OK" o "ERROR nº" según las condiciones del punto anterior.

Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2006

---

3) *2 puntos*

Escribir el código correspondiente al hilo H2, que debe generar una onda senoidal en un canal de salida analógico con las siguientes características:

- Recibe, como parámetro en la inicialización del hilo (ver doc. en última página), el periodo de generación de la onda Tm.
- Es activada periódicamente mediante un mecanismo síncrono cada Tm milisegundos.
- En cada activación calcula el nuevo valor de la onda senoidal, obteniendo los valores de amplitud A y periodo T introducidos desde el hilo principal, y llama a la función de librería `EscribeDA(float valor);`

4) *1.5 puntos*

Describir y/o implementar cómo se corregiría el código de los hilos principal y H2 para que puedan acceder sincronizadamente a las variables y otros recursos compartidos mediante un mecanismo de semáforos o mútex.

¿Qué mecanismos de sincronización se utilizarían si el hilo principal y el hilo H2 deben ejecutarse en equipos diferentes?

5) *1.5 puntos*

Contestar **razonadamente** a las siguientes preguntas:

a) ¿Si se utilizara un RTOS para el sistema de tiempo real? ¿Qué ventajas e inconvenientes tendría su uso?

b) ¿Cuáles de las siguientes características deberían ser consideradas si el factor velocidad de procesamiento es muy importante para este sistema:

- La velocidad de reloj de la CPU seleccionada.
- La CPU seleccionada dispone de memoria caché.
- La CPU dispone de mecanismo de memoria virtual.
- La CPU seleccionada tiene ALU de coma flotante.

c) ¿Por qué se deben utilizar funciones específicas y no las funciones genéricas fopen(), fscanf(), fprintf() etc. para la lectura y escritura de los canales A/D y D/A?

Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2006

Pase de parámetros a main() desde línea de comandos:

```
int main(int argc,char* argv[])  // argv[1] es una cadena de caracteres para el
                                 // 1er parámetro, argv[2] para el 2°, etc.
```

Conversión de cadena de caracteres a entero:

```
int atoi(char* cadena);
```

Documentación de funciones y estructuras POSIX a utilizar:

```
int pthread_create(pthread_t* id,pthread_attr* attr,void *(*fn)(void*),void* arg);
                   // arg es un puntero al parámetro que se quiere pasar al hilo
                   // usar attr=NULL

int sem_init(sem_t *sem, int pshared, unsigned int value); // Usar pshared=0
int sem_wait(sem_t* sem);
int sem_post(sem_t* sem);
int sem_destroy(sem_t* sem);

int pthread_mutex_init(pthread_mutex_t* mutex, const pthread_mutexattr_t *attr);
                                          // Usar attr=NULL
int pthread_mutex_lock(pthread_mutex_t* mutex);
int pthread_mutex_unlock(pthread_mutex_t* mutex);
int pthread_mutex_destroy(pthread_mutex_t* mutex);

int pthread_cond_init(pthread_cond_t* cond, const pthread_condattr_t *attr);
                                          // Usar attr=NULL
int pthread_cond_wait(pthread_cond_t* cond,pthread_mutex_t* mutex);
int pthread_cond_broadcast(pthread_cond_t* cond);
int pthread_cond_destroy(pthread_cond_t* cond);


int timer_create(clockid_t clockid,struct sigevent* sig,timer_t* id);
                                  // Usar clockid= CLOCK_MONOTONIC y sig=NULL
int timer_settime(timer_t id,int flags,const struct itimerspec* ival,
                  struct itimerspec* oval);   // Usar flags=0, oval=NULL
int sigwait(sigset_t* set,int* sig);    // usar *set=SIGRTALL para el timer
int timer_delete(timer_t id);

struct timespec
{
       time_t tv_sec;      // Segundos
       long tv_nsec;// Nanosegundos
} ;

struct itimerspec
{
       struct timespec it_interval;     // Periodo de temporización
       struct timespec it_value;        // Primera activación
} ;
```