Transformando un microprocesador simple en una máquina de procesamiento de eventos simultáneos organizados por tabla

Ing. Ricardo Andrés Korpys[*], Ing. Sergo A. Garassino [**], Dr. Ing. Javier E. Kolodziej [***]

24 de julio de 2017

Director del proyecto de investigación: Sergio Garassino. Semaforización adaptativa. garassino@fio.unam.edu.ar
Integrante del proyecto de investigación: Ricardo Korpys. korpys@fio.unam.edu.ar
Integrante del proyecto de investigación: Javier Kolodziej. koloj@fio.unam.edu.ar

Resumen

Se propone en este trabajo una evolución en cuanto a una visión secuencial para la resolución de un estadio secuencial de eventos que deben ejecutarse en microrpocesador. Un "intérprete" analiza una tabla de datos estructurados y almacenados en memoria del tipo no volátil (ROM) que permite cambios en la programación del funcionamiento del sistema cambiando sólo una tabla de eventos. Permite una reprogramación del sistema a controlar en fracciones de segundos. Las denominadas acciones iniciales y finales asociadas a estos eventos deben ser de corta duración en relación al denominado "tick" de interrupciones. Con aproximadamente 300 bytes de memoria de programa de un microprocesador típico es posible implementar esta técnica propuesta. La cantidad de memoria volátil (RAM) necesaria depende de la cantidad de eventos a procesar en paralelo del sistema físico propuesto.

Palabras clave: evento, intérprete, microprocesador, reprogramación, eficiencia.

Introducción

Supóngase la escritura del software para un simple microprocesador que deba realizar el manejo de las luces de dos semáforos ubicados en la intersección de dos calles. Un diagama temporal, a modo de ejemplo, que refleja esta situación se muestra en la figura 1. Allí, con el estado "alto" desde el punto de vista digital, se supondrá que una de las luces se enciende, y con el estado "bajo" se apaga.

Casi inmediatamente surge una respuesta a la escritura del software que las manejará, y de una forma absolutamente secuencial. Por ejemplo de la siguiente forma, expresada en un pseudocódigo, como el presentado en el algoritmo 1. Si este control de luces se quiere implementar en un microprocesador que no posee un sistema operativo multitarea, el uso real de la mayor porción del tiempo será en ciclos de esperas, impidiendo la ejecución de otra aplicación que requiera el uso de la CPU, por ejemplo el caso de comunicaciones seriales.

Para explicar el funcionamiento de la propuesta se sugiere analizar la figura 1. Con la primera aparición de la señal denominada "tick" en estado alto, se indica el encendido de la luz llamada Verde 1, Rojo 2 y Peatonal 1, además del apagado del resto de las luces involucradas. A partir de este momento, se arranca una abstracción que se denomina *evento,* que en este caso es el llamado Evento 1, que posee una duración de 6 "tick's".

Para éste caso existirán 4, llamados Evento 1, Evento 2, Evento 3 y Evento 4 que se "ejecutan secuencialmente". Cada evento posee un número único y un tiempo asociado durante el cual se "ejecutará".

[*]El Ing. Ricardo A. Korpys es Profesor Titular en el Departamento de Ingeniería Electrónica de la Facultad de Ingeniería de la Universidad Nacional de Misiones (UNaM), Oberá, Misiones, Argentina. Ex Director del Departamento de Ingeniería Electrónica y ex Director de la Carrera en Ingeniería Electrónica. Profesor responsable de las Asignaturas Física 3, Técnicas Digitales 3 y Proyecto y Diseño Electrónico. Es miembro fundador del Grupo de Investigación y Desarrollo en Ingeniería Electrónica (GID-IE) de esta Universidad. e-mail: korpys@fio.unam.edu.ar . TE: +54-3755-422169 int. 150.

[**]El Ing. Sergio A. Garassino es Profesor Titular en el Departamento de Ingeniería Electrónica de la Facultad de Ingeniería de la Universidad Nacional de Misiones (UNaM), Oberá, Misiones, Argentina. Es Director del Departamento de Ingeniería Electrónica. Profesor responsable de las Asignaturas Electrónica Industrial, entre otras. Fue decano de ésta Facultad de Ingeniería. e-mail: garassino@fio.unam.edu.ar . TE: +54-3755-422169 int. 156.

[***]El Dr. Ing. Javier E. Kolodziej es Profesor Adjunto en el Departamento de Ingeniería Electrónica de la Facultad de Ingeniería de la Universidad Nacional de Misiones (UNaM), Oberá, Misiones, Argentina. Profesor responsable de la Asignatura Física Matemática Aplicada 2. Investigador Asistente del CONICET. e-mail: koloj@fio.unam.edu.ar . TE: +54-3755-422169 int. 152. Dr. en ingeniería, sus temas favoritos incluyen el procesamiento de señales.

Algoritmo 1 Pseudocódigo de software para el manejo de las luces de un semáforo, pensado de una forma secuencial.

encender(Verde 1); apagar(Amarillo 1); apagar(Rojo 1); apagar(Verde 2); apagar(Amarillo 2); encender(Rojo 2); encender(Peatonal 1); apagar(Peatonal 2);
esperar(6 segundos);
apagar(Peatonal 1); encender(Amarillo 1);
esperar(2 segundos);
apagar(Verde 1); apagar(Amarillo 1); encender(Rojo 1); encender(Verde 2); apagar(Amarillo 2); apagar(Rojo 2); encender(Peatonal 2);
esperar(6 segundos); etc. etc. etc.

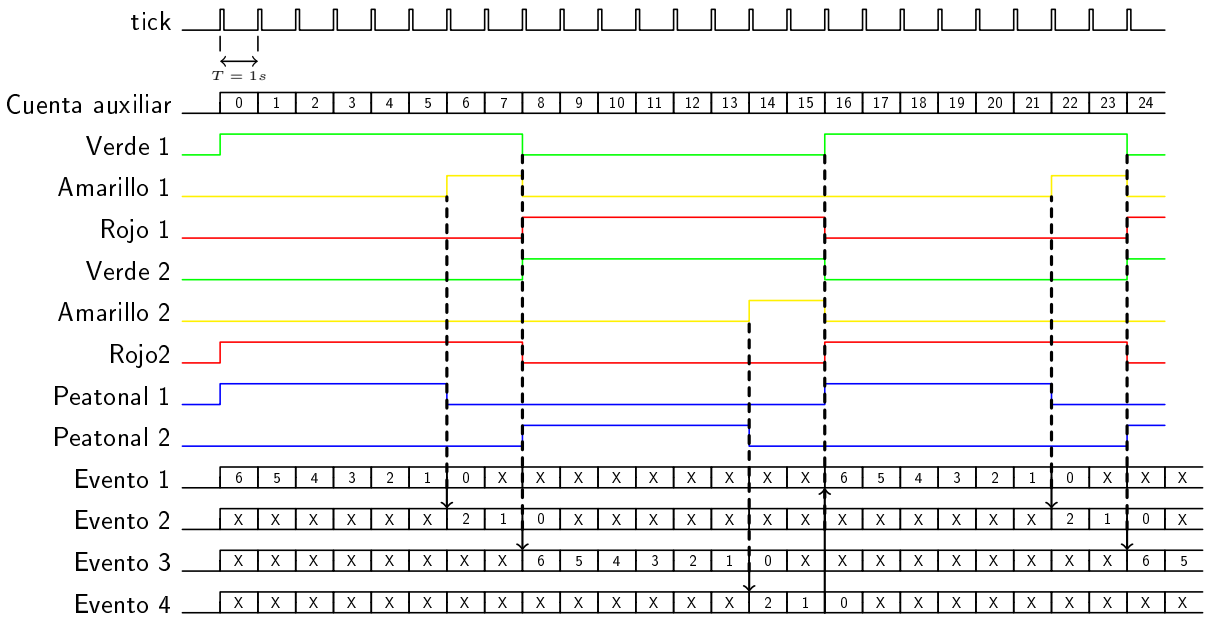

Figura 1: Situación temporal de las luces de semáforos en una intersección de dos calles.

Sucintamente explicado esto funciona de la siguiente forma:

1. En cada ocurrencia de un «tick» se ejecuta una rutina de interrupción, que ejecuta un intérprete y distribuidor.
2. Durante el primer «tick» se carga al Stack de eventos el Evento 1, junto con su duración. Allí se ejecuta una subrutina que ejecuta una *acción inicial* asociada a este evento. Las acciones iniciales implicadas consisten en apagar o encender determinadas luces, de muy corta duracíon.
3. Cuando ocurre el segundo «tick» el scheduler (distribuidor) verifica que hay algún evento en el Stack. Si es así, decrementa el tiempo asociado a este evento (que se encuentra en el Stack). Si este tiempo no es cero, no efectúa ninguna acción, y sale de la rutina de interrupción que provoco el "tick".
4. Mientras el tiempo de algún evento no llegue a cero, simplemente no se efectúa ninguna acción.
5. Como se ve en la figura 1, el tiempo asociado al Evento 1 llega a cero cuando un registro ficticio indicado por la variable Cuenta auxiliar llega al valor de 6 unidades. En este momento, el scheduler ejecuta una subrutina relacionada al evento cuyo tiempo acaba de finalizar, que se denomina *acción final* asociada a este Evento 1. En este caso consiste en encender la luz llamada Amarillo 1 y apagar la Peatonal 1, y ninguna otra acción. El scheduler quita del Stack de eventos el que acaba de llegar a cero, dejando lugar a posibles nuevos eventos. Ahora, la ejecución de un intérprete detecta que existe un *evento final* asociado al que acaba de terminar distinto de cero, que es el Evento 2. Lo carga en el Stack en conjunto con su duración en "tick's" reemplazando al que acabó de finalizar. Ejecuta la *acción inicial* relacionada al Evento 2, que en este caso significa no efectuar ninguna acción «física» sobre el encendido y/o apagado de luces.
6. Cuando ocurre que el tiempo asociado al Evento 2 llega a cero, ejecuta la *acción final* relacionada a este Evento 2, que es apagar las luces Verde 1, Amarillo 1, Rojo 2 y encender las luces Rojo 1, Verde 2 y Peatonal 2 (como puede apreciarse en el cuadro 1). Como este evento llegó a cero lo quita del Stack y lo reemplaza en dicho Stack por el Evento 3, que es su *evento final asociado*. Ejecuta la *acción inicial* del evento recién cargado en el Stack, las cuales no efectúan ninguna acción física (ver cuadro 1).
7. Cuando ocurre que el tiempo asociado al Evento 3 llega a cero, ejecuta la *acción final* relacionada a este Evento 3, que es Encender Amarillo 2 y apagar Peatonal 2 (como puede apreciarse en el cuadro 1). Como este evento llegó a cero lo quita del Stack y lo reemplaza por el Evento 4, que es su *evento final asociado*. Ejecuta la *acción inicial* del evento recién cargado en el Stack, que es el Evento 1, y vuelve a repetirse el ciclo.

Según este simple ejemplo, no existen dos eventos simultáneos en ejecución, porque todos son secuenciales. Pueden presentarse otros ejemplos con eventos simultáneos.

Esto no implica que durante la ejecución de un evento se utilice tiempo de procesamiento de la CPU (Central Processing Unit) del microprocesador. Al contrario, sólo se requiere el uso de la unidad de cálculo de la CPU central cuando un evento dado se "arranca" o cuando se "termina".

En esta propuesta, se utiliza un denominado *Stack de eventos en curso*, que es similar al que se ocupa cuando se utiliza RPN (Reverse Polish Notation). En este Stack se pueden almacenar N eventos que parecerían estar ejecutándose en forma paralela. Se requiere de un temporizador interno que genere una base de tiempos (generalmente regular), que a su vez produzca una interrupción a la CPU que deberá estar habilitada. Durante esta rutina de interrupciones es donde se ejecuta un intérprete y scheduler de eventos que se encuentran en el Stack. El código para su ejecución es relativamente pequeño y ocupa poca memoria. Puede ser almacenado en una ROM. El tiempo de ejecución máximo de esta rutina debe ser mucho más pequeño que el tiempo entre los denominados "tick's".

Desarrollo

Se estructura una porción de la memoria para contener "el programa del semáforo" como se muestra en el cuadro 1. Un "tick" dado por una interrupción de un timer interno del microcontrolador proporciona la base de tiempos para esta propuesta. Puede tener, como ejemplo, un segundo de duración.

El numero de evento 0 se reserva para que el intérprete no recarge ningún evento en el Stack de eventos cuando el triempo asociado llegue a cero.

Una rutina asociada, por ejemplo a la acción inicial del Evento 1, podría escribirse en lenguaje assembler como se muestra en el algoritmo 2, para microcontroladores de la marca NXP.

Nº de evento	Acción inicial	Acción final	Duración del evento	Evento inicial	Evento final
1	Encender Verde 1, Rojo 2 y Peatonal 1. Apagar todas las demás	Apagar Peatonal 1. Encender Amarillo1.	6	0	2
2	No efectar ninguna acción física.	Apagar Verde 1, Amarillo 1, Rojo 2. Encender Verde 2, Rojo 1 y Peatonal 2.	2	0	3
3	No efectuar ninguna acción física.	Encender Amarillo 2. Apagar Peatonal 2.	6	0	4
4	No efectuar ningua acción física.	No efectuar ningua acción física.	2	0	1

Cuadro 1: Desarrollo de la tabla de eventos para el semáforo de ejemplo.

Algoritmo 2 Rutina en lenguaje Assembler de la acción inicial del Evento 1.

```
Dirección de la acción inicial del evento 1:
BSET 0,PORTB ; Instrucción en lenguaje assembler que enciende a la luz Verde 1.
BSET 1,PORTB ; Se enciende la luz llamada Roja 2.
BSET 2,PORTB ; Se enciende la luz llamada Peatonal 1.
BCLR 3,PORTB ; Se apaga la luz llamada Amarillo 1.
BCLR 4,PORTB ; Se apaga la luz llamada Roja 1.
BCLR 5,PORTB ; Se apaga la luz llamada Verde 2.
BCLR 6,PORTB ; Se apaga la luz llamada Amarillo 2.
BCLR 7,PORTB ; Se apaga la luz llamada Peatonal 2.
RTS ; retorna del llamado de una subrutina “superior”.
```

Nº de evento	Dirección donde ejecutar acción inicial del Nº de evento	Dirección donde ejecutar acción final del Nº de evento	Tiempo de duración del Nº de evento	Evento inicial que se carga en el stack de eventos cuando se ejecuta la acción inicial de este evento	Evento inicial que se carga en el stack de eventos cuando se ejecuta la acción inicial de este evento
Evento 1	dirección donde ejecutar acc. inicial del Evento 1	dirección donde ejecutar acc. final del Evento 1	tiempo de duración del evento 1	evento inicial	evento final
Evento 2	dirección donde ejecutar acc. inicial del Evento 2	dirección donde ejecutar acc. final del Evento 2	tiempo de duración del evento 2	evento inicial	evento final
Evento 3	dirección donde ejecutar acc. inicial del Evento 3	dirección donde ejecutar acc. final del Evento 3	tiempo de duración del evento 3	evento inicial	evento final
—-	—-	—-	—-	—-	—-
Evento N	dirección donde ejecutar acc. inicial del Evento N	dirección donde ejecutar acc. final del Evento N	tiempo de duración del evento N	evento inicial	evento final

Cuadro 2: Estructura de los eventos a almacenar en memoria ROM.

Nº de evento / duración	Descripción	
10	Duración en ticks del Evento 3 que restan para decrementarse.	-> SP
3	Número de Evento en el stack en espera de terminarse.	-> SP-1
20	Duración en ticks del Evento 1 que restan para decrementarse.	-> SP-2
1	Número de Evento en el stack en espera de terminarse.	-> SP-3

Cuadro 3: Ejemplo de un Stack de eventos, cuando se encuentra "en ejecución".

Estructura de los eventos a evaluarse en la aplicación completa.

Esta se presenta en el cuadro 2. Esta es la estructura que hay que modificar para cambiar «la forma en que se comporta el semáforo». No requiere substancialmente otro cambio. Por cada evento son necesarias 7 posiciones de memoria ROM. Por ejemplo si en una aplicación específica se requieren de 8 eventos, la cantidad de memoria necesaria es de 56 posiciones.

Estructura del stack de eventos en curso.

Una representacíon gráfica del stack de eventos en curso puede apreciarse en el cuadro 3, para el caso de dos eventos. Esta se organiza en forma dinámica y muy similar a las de calculadoras que utilizan RPM (reverse polish notation). La cantidad de eventos simultáneos que puede manejar una implementación depende de la memoria RAM que posea el procesador utilizado, pero podrían ser 30 eventos simultáneos, como ejemplo. Cada vez que el tiempo de un evento en este Stack llega a cero, el scheduler lo quita de este Stack, dejando lugar a que un nuevo evento ocupe estas posiciones. Al puntero (dirección) que apunta al Stack de eventos se lo denominó SP.

Visto el cuadro 3 luego de 10 "tick's" este se transforma en el cuadro 4, suponiendo que la finalización del Evento 3 invoque al Evento 7.

Implementación y validación de la propuesta

Para la validación esta propuesta, se implementó el control de un semáforo utilizando un microcontrolador de 8 bits modelo MC9S08QG8CBPE [1] con los tiempos y luces indicados en la figura 1. Este posee 8 Kb de memoria de programa, 512 bytes de memoria RAM, 16 pines, un oscilador interno, un timer interno que puede provocar una interrupción cada 1,024 s, entre otras cosas.

Nº de evento / duración	Descripción	
35	Duración en ticks del Evento 7 que restan para decrementarse.	-> SP
7	Número de Evento en el stack en espera de terminarse.	-> SP-1
10	Duración en ticks del Evento 1 que restan para decrementarse.	-> SP-2
1	Número de Evento en el stack en espera de terminarse.	-> SP-3

Cuadro 4: Ejemplo de un Stack de eventos, cuando se termina el Evento 3 y se carga el Evento 7.

En un puerto paralelo de 8 bits de salida llamado Port B, se conectan 8 diodos LED's que poseen los mismos colores que las luminarias de un semáforo real. Se programó el intérprete y scheduler en la zona alta del mapa de memoria, usando lenguaje assembler para codificarlo debido a que debe ser rápido. Este ocupa aproximadamente 300 bytes de los 8 Kb (un 3,7 % del total). Las estructuras que definen los cuatro eventos poseen un tamaño de 4*7= 28 bytes, y el tamaño donde se guardan las subrutinas de las acciones iniciales y finales de cada evento unos 100 bytes.

Al comienzo del programa principal se selecciona al oscilador interno, se configuran las puertas de entrada salida, se programa al denominado RTI (real time interrupt) para que se provoque una interrupción cada 1,024 s , se carga el Evento 1 con su duración en el Stack de eventos, se habilitan las interrupciones generales, y se ejecuta la instrucción:

- STOP , que detiene a la CPU central, colocando al sistema en un muy bajo consumo energético.

Cuando ocurre la siguiente interrupción por parte del RTI, sale de esta condición, ejecuta el intérprete y al scheduler, y vuelve a ejecutar la instrución STOP.

Se alimentó a este prototipo con dos baterías de 1,5 Votios cada una. El mayor consumo de energía esta dado por cada LED que se enciende, de acuerdo al patrón del semáforo de ejemplo. Se instaló un pulsador conectado a un pin del puerto A. Cuando este se mantiene pulsado luego del Reset de la CPU central, los tiempos pasan a ser superiores a los propuestros en el ejemplo, según lo indicado en otra tabla de programación, priorizando una de las calles de una intersección.

Conclusiones

Utilizando esta técnica propuesta es posible minimizar el tamaño del código de programa necesario para implementar una estructura de ejecución de eventos secuenciales. Esta técnica no puede ser considerada como un sistema operativo. Permite reducir el tiempo necesario para alterar y reconfigurar, por ejemplo, la configuración de una serie de semáforos. Como trabajo a futuro, se pretende implementar esta técnica en otra familia de microcontroladores.

Referencias

[1] NXP Semiconductors, MC9S08QG8 and MC9S08QG4 Data Sheet, http://cache.nxp.com/docs/en/data-sheet/MC9S08QG8.pdf , Accedido en junio de 2017.