Manual de usuario de la BT6811

Andrés Prieto-Moreno Torres

10/Feb/2001

1. Introducción

La BT6811 es una tarjeta microcontroladora basada en el MC68hc11 de Motorola. Su objetivo es controlar de forma autónoma cuatro servomecanismos, permitiendo una conexión sencilla de los mismos y facilitando su alimentación.

Para desarrollar esta placa se ha partido del modelo CT6811, anteriormente desarrollado y se han eliminado los componentes que no se van a utilizar. Al final se ha obtenido una placa con unas dimensiones de 48 mm x 58 mm.

Los elementos que se han eliminado con relación a su antecesora han sido: el dispositivo de comunicaciones MAX232, los puertos B, C, Control, el jack de alimentación, los switches de configuración y algunos jumpers de configuración. Por contra, se han introducido los siguientes elementos: Una clema para alimentar los motores, un jumper para poner la alimentación de los motores interna, cuatro conexiones directas para servomecanismos y un puerto mixto, que saca al exterior el puerto C y el E.

Se ha conseguido rutar la placa a una sola cara, logrando así abaratar los costes de fabricación y permitir que la placa se pueda construir artesanalmente. En el caso de soldar a mano se hará por la cara de atrás y la tarjeta no necesitará taladros metalizados. Una consecuencia de esto, es que se han perdido algunas compatibilidades de puertos con relación a la CT6811, pero esto no es importante debido a la nueva función que ha de desarrollar esta placa. En los capítulos siguientes se irán detallando todas las funciones.

2. Modos de funcionamiento

Realmente la BT6811 está pensada para funcionar en modo autónomo, dentro de esta modalidad se pueden distinguir dos tipos de modos.

1. Arrancando en Bootstrap (jumper JP1 puesto): en este caso al alimentar la BT6811 se ejecuta un programa interno del MC6811 que se queda esperando a recibir un programa por el puerto serie. Se puede utilizar esto para volcar programas desde el PC [1], o desde cualquier otro dispositivo serie, como por ejemplo una calculadora HP. Para pasar al modo autónomo desde el *bootstrap* hay que colocar el jumper JP2. Así, cada vez que se haga un reset se le indica al programa *bootstrap* que ejecute el programa grabado en la EEPROM interna. El inconveniente de este método es que al poner el jumper JP2 se pierden las comunicaciones serie asíncronas (SCI) de la BT6811.

2. Arrancando en Single-Chip (jumper JP1 quitado): en este caso al introducir la alimentación en la tarjeta se ejecuta el programa apuntado por el vector de reset. Debido a que este vector se encuentra en la posición $FFFF, será necesario disponer de un microcontrolador modelo E2 (que tiene 2K de EEPROM situados en el tramo $F800-$FFFF). Al grabar el programa en la EEPROM hay que asegurarse de poner el vector de reset apuntando al principio del programa. Con este método no se coloca el jumper JP2 y no se pierden las comunicaciones serie asíncronas.

En ambos casos, se ha pensado en la posibilidad de montar una red de tarjetas BT6811. Para ello se puede utilizar el SPI o el SCI, cuyos pines están en el PUERTO D. La red estará formada por un conjunto de

[1] Utilizando un adaptador especial ya que la BT6811 no lleva MAX y por lo tanto no puede recibir señales del PC.

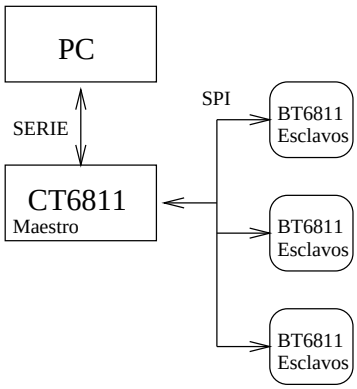


Figura 1: Red de tarjetas BT6811.

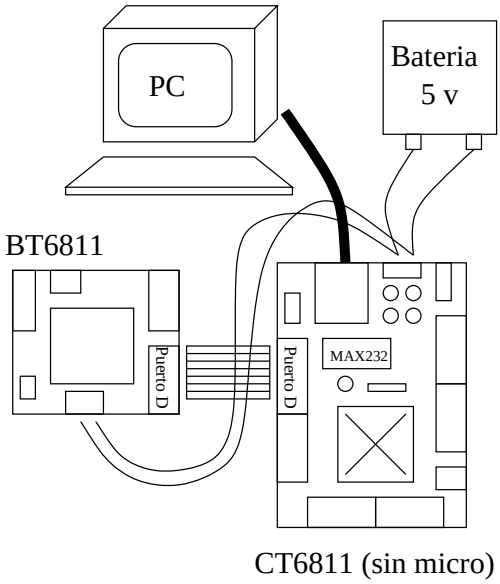


Figura 2: Esquema de conexión de la BT6811 en modo entrenadora.

BT6811 esclavas y un dispositivo maestro, como por ejemplo una CT6811. Con la estructura planteada se puede tener un PC comunicándose con una CT6811 por el puerto serie y ésta a su vez controlando la red de BT6811 por el puerto SPI. Ver figura 1

Otra necesidad diferente es poder cargar o grabar programas desde el PC en el microcontrolador 6811 de la BT6811 sin tener que quitarlo del zócalo. Es decir usar la BT6811 como entrenadora para ejecutar el “downmcu”, el “ctdialog” o el “mcboot”. Como la tarjeta no lleva un MAX232 habrá que proporcionarle uno, las alternativas son dos: o se construye un adaptador o se utiliza una tarjeta CT6811 que ya lo lleva incorporado. En ambos casos la conexión del adaptador se hará por el puerto D de la BT6811.

En este manual se hace referencia al segundo caso, es decir a utilizar la CT6811 como interfaz entre el PC y la placa BT6811. Lo que se hará es conectar los puertos D de ambas tarjetas con un cable de bus estándar, quitar el microcontrolador de la CT6811 y alimentar con 5 voltios ambas tarjetas. El PC se conecta mediante el cable telefónico al conector de la CT6811. A partir de este momento se pueden ejecutar todos los programas citados antes. La única diferencia es que ahora no hay “reset software”, por lo que habrá que pulsar el reset de la BT6811 de forma manual. Salvo este pequeño detalle todo lo demás se mantiene como en la CT6811. Ver figura 2

Es importante que se quite el micro de la CT6811 para poder cargar los programas en la BT6811. Si no se quita aparecerá un conflicto entre los programas de BOOTSTRAP de cada micro. Esto último se

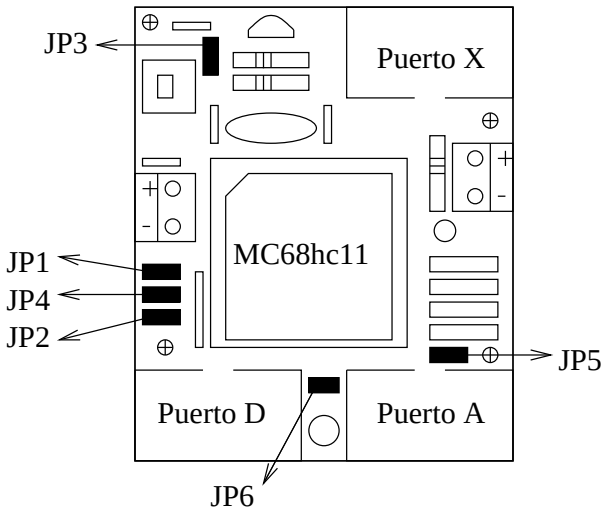


Figura 3: Disposición de los jumpers en la BT6811.

puede generalizar a una red de BT6811, si las tenemos todas conectadas entre sí por un cable de BUS normal y arrancamos todos los micros en BOOTSTRAP para ser cargados desde el adaptador del MAX, resultará que no funcionará nada. Los programas *bootstrap* de cada BT6811 interferirán la línea TX y RX corrompiendo la información. Debido a esto cada BT6811 conectada a una red habrá que reprogramarla aisladamente.

3. Jumpers de configuración de la BT6811

La BT6811 tiene seis jumpers de configuración, se procede a describir la función de cada uno de ellos. Ver figura3

1. JP1 (#B/S): Es el encargado de seleccionar el modo de funcionamiento del microcontrolador. Si el jumper esta puesto se configura el modo “bootstrap”, por el contrario si esta quitado se configura el modo “single-chip”. Para poder utilizar éste último será necesario disponer del vector de reset en EEPROM, como ocurre con los modelos de microcontroladores MC68hc11E2. El jumper estará puesto.

2. JP2 (Go EEPROM): Este jumper tiene sentido cuando el JP1 está conectado, es decir cuando la BT6811 esta configurada en modo “bootstrap”. Si se conecta el JP2, cada vez que se pulse “reset” en la BT6811 se ejecutará el programa que esté grabado en la EEPROM interna. Si por el contrario se deja quitado, al recibir el “reset” se ejecutará el programa “bootstrap” y por lo tanto la placa estará funcionando en modo entrenador, esperando a recibir el programa a ejecutar por el puerto serie.

3. JP3 (LVI): Este jumper esta conectado inicialmente. Su función es proteger el programa grabado en la EEPROM, haciendo un reset del sistema cada vez que la alimentación baje de 4,5v. Por eso si se utilizan aplicaciones que inducen ruido eléctrico en la alimentación, será aconsejable quitar el jumper. Por ejemplo, es el caso de utilizar la placa para mover motores estando configurada para alimentarlos internamente.

4. JP4 (EXP): Su función es permitir sacar por el puerto D la alimentación TTL. Esto es muy útil a la hora de conectar otros periféricos a la BT6811. Inicialmente está quitado, debido a que el periférico que se suele conectar es una CT6811 y ésta no puede recibir ninguna señal de alimentación por dicho puerto.

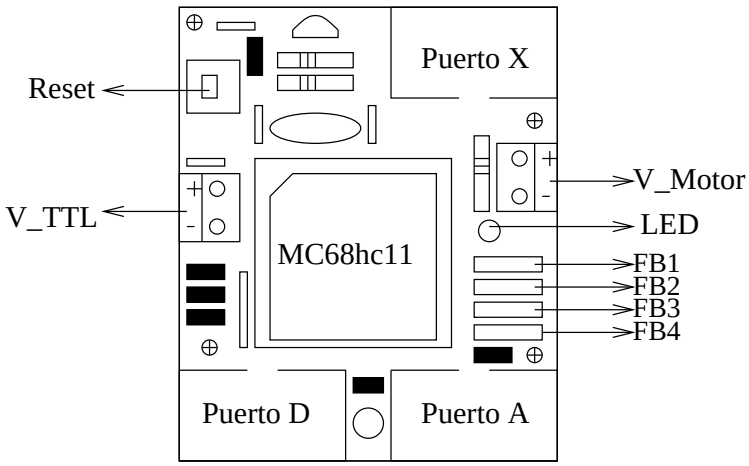


Figura 4: Resto de componentes de la BT6811.

5. JP5 (Vcc ON): Su utilidad es permitir llevar la alimentación TTL a los motores, de manera que no sea necesario conectar una fuente externa (batería, pila, etc.) para poder moverlos. Inicialmente esta quitado, por lo que habrá que utilizar la segunda fuente para poder mover los motores.

6. JP6 (SS): ecto está puesto, su utilidad es facilitar el control en las redes de BT6811 mediante el SPI. Al quitar el jumper JP6, la línea SS (Slave Select) que proviene del maestro no se conecta al SS del esclavo. De esta forma podemos hacer que otros pines del micro maestro se puedan conectar al SS del esclavo, pudiendo activar cada esclavo sin interferir en los demás.

4. Localización del resto de componentes

Ya se han descrito los jumpers de configuración, ahora se describen el resto de componentes de la BT6811. Ver figura 4

- LED: Esta conectado al bit_4 del puerto B. Este puerto esta mapeado en la dirección $1004 del microcontrolador MC68hc11. Si se pone a uno dicho bit, entonces se encenderá el LED.
- Reset: Cada vez que se pulse el “pulsador” se estará haciendo un reset del MC68hc11. El programa en ejecución se parará y volverá a inicializarse el sistema. El modo de arranque será el configurado por medio de los jumpers JP2 y JP1.
- V_TTL: Es la clema por donde se introduce la tensión de alimentación TTL (5v) necesaria para la electrónica.
- V_Motor: Es la clema por donde se introduce la tensión de alimentación de los motores. Si se conecta el jumper JP5 se evita usar esta clema, al permitir que la tensión TTL (la introducida por V_TTL) sea la que alimente los motores.
- Puerto D: Es un bus de expansión, por él van las señales de comunicaciones serie y las del SPI.
- Puerto A: Es un bus de expansión, por él van las señales del puerto A del MC68hc11 junto con V_TTL y GND. Este puerto es totalmente compatible con el de la CT6811.
- Puerto X: Es un bus de expansión mixto. Por él se sacan al exterior las señales del puerto C y del puerto E del MC68hc11. El puerto E es de entrada mientras que el C es bidireccional, hay que tener en cuenta esto para evitar cortocircuitar dos salidas. Por ejemplo si se quiere introducir una señal por el pin PX1, hay que configurar el bit del puerto C correspondiente a PX1 como entrada. Luego se podrá leer el valor bien por el puerto E o por el C. El estado predeterminado del Puerto C y

PUERTO X	Puerto C (bidireccional D)	Puerto E (entada D/A)
PX0	PC0	PE3
PX1	PC1	PE7
PX2	PC2	PE2
PX3	PC3	PE6
PX4	PC4	PE5
PX5	PC5	PE1
PX6	PC6	PE4
PX7	PC7	PE0

Cuadro 1: Correspondencia de bits en el Puerto X.

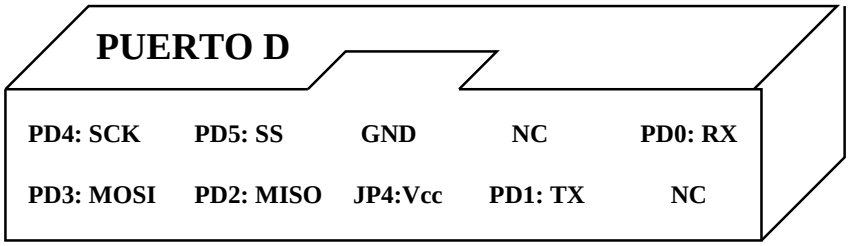
del Puerto E es de entrada, por lo que no habrá ningún conflicto en el momento de arranque del microcontrolador.Ver tabla 1

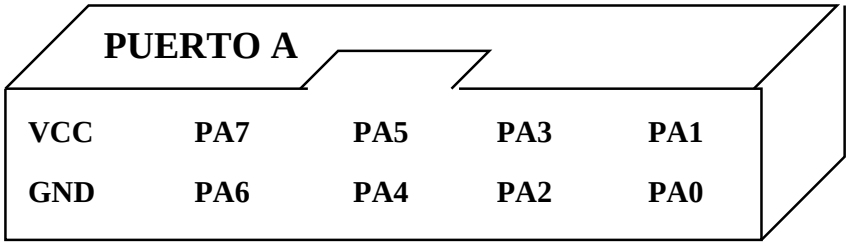
- FB1, FB2, FB3, FB4: Son jumpers triples donde se van a conectar los servomecanismos. El FB1 se controla con el bit_0 del puerto B del MC68hc11, el FB2 con el bit_1, el FB3 con el bit_2 y el FB4 con el bit_3.

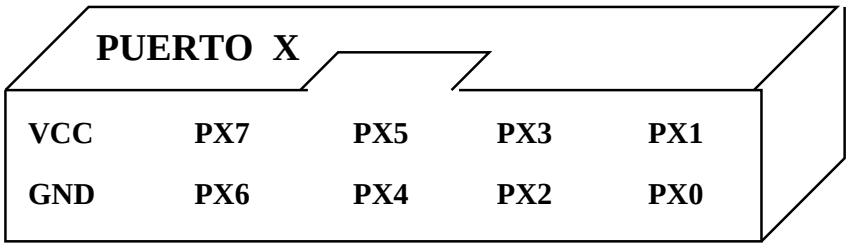
Puerto B	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
$1004	–	–	–	LED	FB4	FB3	FB2	FB1

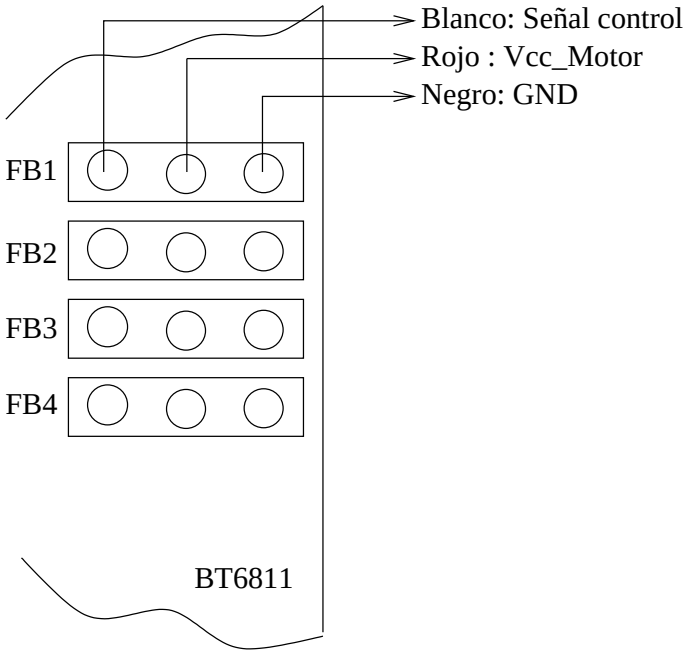
- El resto de componentes son el reloj y sus condensadores asociados, el propio MC68hc11, condensadores de filtrado de alimentación y resistencias de polarización.

5. Pines de los buses de expansión


PUERTO D
PD4: SCK
PD5: SS
GND
NC
PD0: RX
PD3: MOSI
PD2: MISO
JP4:Vcc
PD1: TX
NC


PUERTO A
VCC
PA7
PA5
PA3
PA1
GND
PA6
PA4
PA2
PA0


PUERTO X
VCC
PX7
PX5
PX3
PX1
GND
PX6
PX4
PX2
PX0


Blanco: Señal control
Rojo : Vcc_Motor
Negro: GND
FB1
FB2
FB3
FB4
BT6811

6. Utilización de la BT6811

La BT6811 se puede usar como entrenadora, o como producto final (autónomo), tanto en modo aislado como en red. Su diseño está pensado para los modos autónomos y dentro de ellos el de red. Para usarse como entrenadora se debe disponer de un adaptador serie como el max232, éste será externo y permitirá unir la tarjeta con el puerto serie del PC. Como la CT6811 incorpora uno se puede utilizar ésta como adaptador.

En este apartado se van a describir mediante ejemplos todas las formas de uso. Se empezará por el modo entrenador y se terminará con el modo autónomo en red.

6.1. BT6811 trabajando como entrenadora

La utilidad de usar la BT6811 como entrenadora radica en la posibilidad de programar la tarjeta sin necesidad de sacar el microcontrolador de su zócalo, detalle importante ya que en algunas ocasiones se tiene un difícil acceso a él. Como ya se ha dicho con anterioridad, para poder usar este modo se necesita un adaptador externo que permita conectar el puerto serie de la BT6811 con el puerto serie del PC. En el mercado existen varios integrados que realizan esta función, uno de ellos es el MAX232, que es el que se ha utilizado en la CT6811. Por esta razón se ha diseñado la BT6811 para poder ser conectada al PC usando la CT6811 como adaptador. La única condición es sacar el microcontrolador del zócalo de la CT6811 y unir ambas tarjetas por el puerto D mediante un cable bus estándar de diez hilos. En la figura 2 se describe un esquema de conexión.

Para configurar la BT6811 en modo entrenador hay que colocar los jumpers de la siguiente forma:

1. Jumper JP1 puesto para seleccionar el modo Bootstrap.
2. Jumper JP2 quitado para permitir la ejecución del programa Bootstrap.
3. Los jumpers JP3, JP5 y JP6 no influyen en la configuración del modo entrenador.
4. Jumper JP4 desconectado ya que el modo auxiliar de la BT6811 será la CT6811.

Con la estructura y la configuración planteadas la BT6811 se comporta como una CT6811 pero sin la función de reset software. Esto hace que la mayoría de los procedimientos y ejemplos que se describen en el manual de la CT6811 sean válidos para la BT6811. A pesar de esto se describe a continuación cómo enviar programas a la tarjeta y cómo dejarlos autónomos, pero conviene recordar que se pueden obtener ejemplos más prácticos en el manual de la CT6811.

Para descargar un programa desde el PC se pueden utilizar los programas *downmcu*, *ctload* y *mcboot.* En este caso se va a emplear el *downmcu* por ser el más conocido y disponible para las plataformas MsDos y Linux. Ahora hay que mencionar una diferencia fundamental con respecto al proceso de carga con la entrenadora CT6811. En dicha entrenadora existe la función de reset software que permite que el PC pueda *resetear* al microcontrolador justo antes del proceso de carga. En la BT6811 esa función no existe por lo que el proceso de reset tendrá que ser manual. El método para ello es pulsar el botón de reset de la BT6811 e inmediatamente después pulsar el botón de ENTER del PC para ejecutar el programa *downmcu*. Resumiendo:

1. Teclear en la línea de comandos: downmcu *miprograma* -com1
2. Pulsar el botón de reset de la BT6811
3. Pulsar el botón de ENTER del PC para ejecutar el programa *downmcu*.

Miprograma es el nombre del programa con extensión s19, es decir ya compilado, que se quiere enviar a la BT6811. Si la conexión se ha realizado por el puerto serie dos, habrá que poner como parámetro *-com2*.

Otros programas que pueden utilizarse en modo entrenador son el *mcboot* y el *ctdialog*. El primero consiste en un pequeño terminal que manda por el puerto serie todo lo que recibe del teclado y envía a la pantalla todo lo que recibe por el puerto serie. La forma de utilizarlo es la siguiente:

1. Teclear en la línea de comandos: *mcboot*

2. Pulsar F4 para seleccionar el puerto serie adecuado.

3. Finalmente pulsar F1 para cargar programas en la RAM interna. En el proceso de carga habrá que hacer lo mismo que con el *downmcu*, pulsar el reset de la BT6811 y luego pulsar ENTER para enviar el programa al micro.

Mediante el *mcboot*, una vez cargada la aplicación deseada en la BT6811, se devuelve el control al terminal del *mcboot*. Por eso si el programa enviado a la BT6811 permite recibir carácteres por el puerto serie se le pueden enviar con tan solo pulsar las teclas en el *mcboot*. A su vez si la BT6811 envía datos por el puerto serie estos aparecerán en la pantalla del *mcboot*.

Por último, en el modo entrenador hay otro programa que adquiere mucha importancia a la hora de buscar aplicaciones autónomas. Este programa es el *ctdialog*, que como se acaba de decir a pesar de ejecutarse en modo entrenador es el que permite, entre otras funciones, dejar grabados los programas permanentemente en la EEPROM interna, de forma que se obtienen aplicaciones autónomas del PC.

Para ejecutarlo en MSDOS hay que hacer lo siguiente:

1. downmcu ctserver -com1

2. ctdialog -com1

Si el puerto serie utilizado es el segundo hay que poner -com2 y tener presente siempre que la BT6811 no dispone de reset software por lo que habrá que hacerlo manual al utilizar el *downmcu*. En el *ctdialog* no es necesario ya que opera sobre el servidor *ctserver* previamente cargado en el microcontrolador.

Para ejecutarlo en Linux no hace falta el paso primero, con poner ctdialog -com1 bastará, aunque ahora sí es necesario pulsar el botón de reset de la BT6811 justo antes de pulsar el ENTER. Una vez que se ha cargado el programa aparecerá en pantalla lo que se muestra en la figura 5. A partir de ese momento se puede grabar un programa en la memoria EEPROM[2] mediante la orden siguiente:

1. **ms** 1035 0 (Sólo para el micro E2 y sólo en MSDOS)

2. **eeprom** miprograma

Después de terminar la ejecución de la orden (aparece un mensaje de OK), el microcontrolador de la BT6811 tendrá grabado en su memoria *EEPROM* el programa. Esta memoria no es volátil, permaneciendo inalterado su contenido independientemente del estado de las baterías. Esta característica hace posible las aplicaciones autónomas que se describen en los siguientes apartados.

6.2. BT6811 en modo autónomo aislado

Al decir que la BT6811 esta funcionando en modo autónomo se hace referencia a que el programa de la aplicación ya se encuentra grabado en la EEPROM interna del microcontrolador, por lo que no es necesario utilizar el PC para descargar los programas y probar. Es decir no hace falta utilizar el adaptador que antes había sido necesario incorporar.

¿ Qué aplicaciones se pueden hacer con una BT6811 ?. La respuesta una vez más: depende de la imaginación del lector, pero se pueden citar ejemplos de cómo dotar de movimiento a un robot o a una maqueta. Con una BT6811 se podrían mover alternativamente cuatro motores, ocho luces y tal vez emitir alguna señal acústica de vez en cuando. Se recuerda que el motivo principal del desarrollo de esta tarjeta es el poder controlar hasta cuatro servomecanismos Futaba para la realización de robots autónomos. Además

[2] La memoria EEPROM actua como una EPROM pero que se borra eléctricamente. La particularidad de una EPROM es que sus contenidos se graban electricamente pero se borran con luz ultravioleta, por eso se mantienen independientemente de si hay o no tensión eléctrica. Con la EEPROM se da un paso más ya que se sustituye la luz ultravioleta por unos pulsos de tensión especiales, de forma que el proceso de borrado y escritura se puede realizar sin necesidad de agentes externos y en un tiempo menor. Esta memoria hace que la familia de microcontroladores MC68HC11 haya tenido una aceptación muy grande en la industria, pues de una manera muy fácil permite programar aplicaciones que no necesiten de mecanismos externos para retener en memoria el programa, considerándose así micros autónomos o preprogramados.

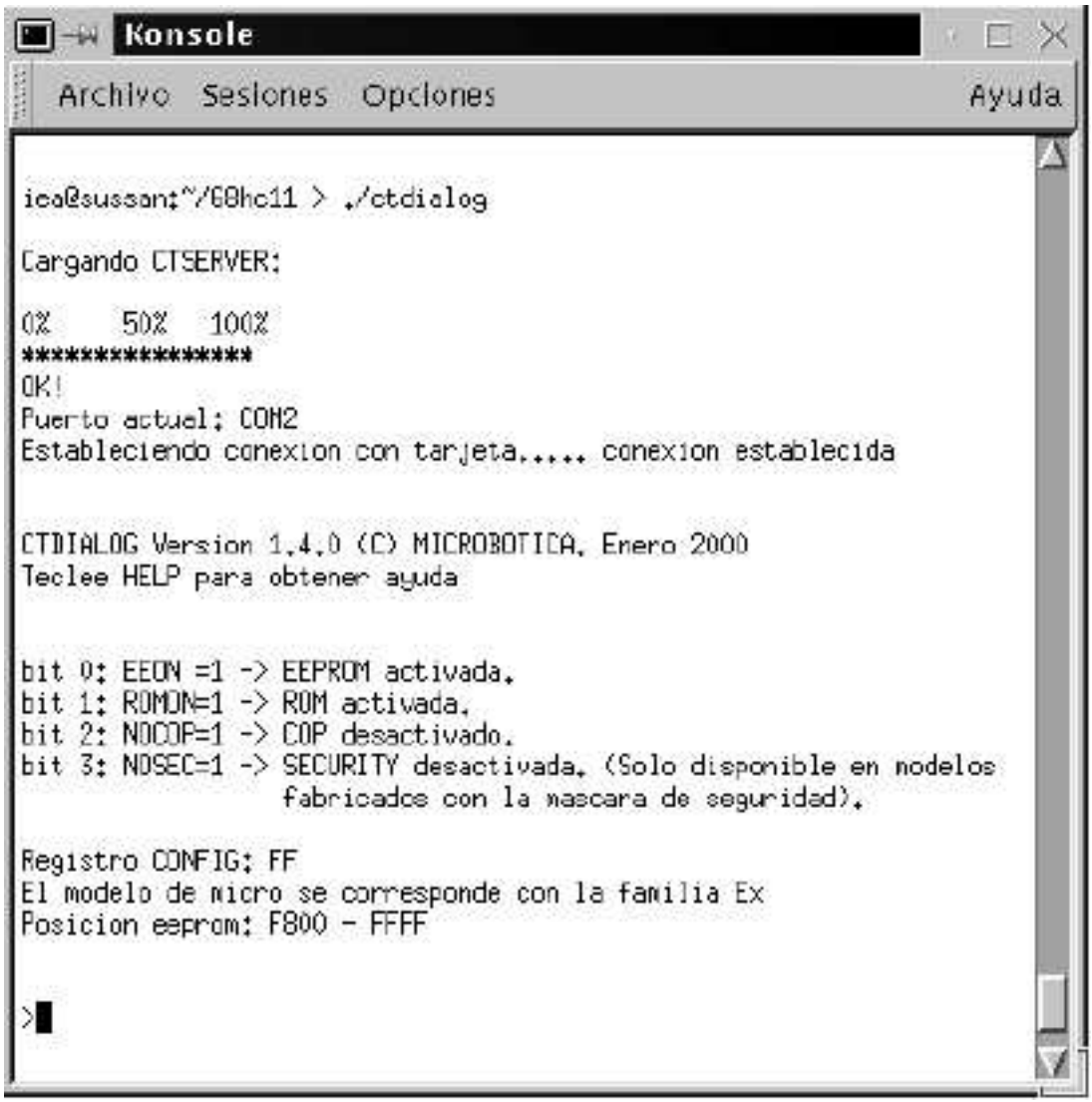


Figura 5: Programa CTDIALOG bajo Linux.

tiene suficiente capacidad para gestionar otros recursos como los anteriormente citados : leds, sonidos, bumpers, etc.

Las ventajas de usar la tarjeta en este modo son:

1. Tiene un tamaño reducido y un coste inferior a la CT6811 ya que sólo utiliza los componentes imprescindibles, el resto los elimina.

2. Ya viene preparada para conectar cuatro servomecanismos Futaba.

El lector puede considerar que estas características no son suficientes para justificar la utilización de la BT6811 en lugar de la CT6811 y en parte lleva razón. Se recuerda que para programarla se necesita una CT6811 para realizar todo lo descrito en el punto anterior. Y si ya se dispone de una CT6811, es más fácil construirse el adaptador para los motores que la propia BT6811. Entonces, ¿ dónde está la verdadera utilidad ?. La respuesta se encuentra en las redes de control distribuido, en donde conceptos como la modularidad, el coste y el tamaño toman bastante importancia. En el apartado siguiente se explicarán ejemplos de esto.

Para configurar la BT6811 en modo autónomo (independientemente de si es aislado o en red) se pueden emplear dos métodos. El primero se usa cuando el microcontrolador que lleva la tarjeta es un MC68HC11A1 y el segundo se emplea cuando el microcontrolador es un MC68HC11E2. La diferencia entre ambos se encuentra en que en el primer caso el microcontrolador tiene situada la EEPROM en la dirección $B600 y la única forma de acceder a ella nada más activar la placa es mediante el modo *bootstrap* y la colocación de un jumper. Por el contrario en el segundo caso el microcontrolador tiene situada la EEPROM en el rango $F800 - $FFFF por lo que el vector de reset puede programarse en ella y por ello se habilita el modo de arranque *single-chip*. Mediante este modo nada más activar la tarjeta se ejecuta el programa indicado por el *vector de reset*.

Configuración del modo autónomo para el microcontrolador 68HC11A1:

1. Jumper JP1 puesto para configurar el modo *bootstrap*.

2. Jumper JP2 puesto para permitir el salto a la EEPROM y ejecutar así el programa grabado en ella.

3. Los demás jumpers no tienen importancia en la selección del modo aunque sí en el resto de funcionalidades.

Configuración del modo autónomo para el microcontrolador 68HC11E2:

1. Jumper JP1 quitado para configurar el modo *single-chip*.

2. Jumper JP2 quitado.

3. Vector de reset grabado en la EEPROM y apuntando al inicio del programa.

4. El resto de jumpers no tienen importancia en la selección del modo aunque sí en el resto de funcionalidades.

Para terminar el apartado se explicará mediante un ejemplo cómo obtener una aplicación autónoma. Lo importante del ejemplo no es la funcionalidad del mismo sino el proceso que se va a seguir para tener el sistema funcionando, sobre todo porque en el apartado siguiente todo lo explicado aquí se dará por aprendido. Además, se va a considerar sólo el segundo caso, es decir cuando el microcontrolador de la BT6811 es un MC68HC11E2.

El ejemplo pretende que en cuanto se enchufe la BT6811, empiece a parpadear su LED. El código del programa sería el siguiente:

```
* Programa que hace parpadear el LED de la BT6811

PORTB  EQU $04
```

```
    ORG $F800  * Micro E2

inicio         * Inicio del programa
    LDX #$1000

bucle
    LDAA PORTB,X
    EORA #$10
    STAA PORTB,X
    BSR  pausa
    BRA  bucle

pausa
    LDY  #$FFFF
sigue
    DEY
    CPY  #$0
    BNE  sigue
    RTS

* Vector de interrupción del reset

    ORG $FFFE
v_reset FDB #inicio
```

Lo importante del programa es ver cómo se ha configurado el vector de reset (últimas dos líneas) apuntando al inicio del programa, de manera que cuando se arranque en *single-chip* se empiece a ejecutar la instrucción situada en la dirección guardada en el vector. Esa instrucción tiene que coincidir con el inicio del programa. Si no se configura correctamente, el programa empezará a ejecutarse por cualquier lado siendo impredecible su comportamiento. En caso de utilizar un microcontrolador MC68HC11A1 este vector no se utiliza, por lo que no hará falta configurarlo y en cambio habrá que considerar que cada vez que se haga un reset y se tenga la placa configurada correctamente se empezará a ejecutar la instrucción del programa que se encuentra en la línea $B600.

El programa anterior hay que compilarlo y mediante el *ctdialog* grabarlo en la EEPROM interna. La forma de realizar esto último se ha explicado en el apartado anterior.

1. Para compilar : as11 miprograma.asm
2. Ejecutar el *ctdialog* : >ctdialog -com1[3]
3. Para grabar : **eeprom** miprograma

Una vez grabado el programa se configura la BT6811 y si todo ha ido bien, cada vez que se pulse el reset o se enchufe la tarjeta el LED empezará a parpadear. A continuación se explicará el modo autónomo en red que, como ya se ha dicho, es con el que se obtiene la máxima utilidad de la BT6811.

6.3. BT6811 en modo autónomo en red

La BT6811 como sistema de control aislado tiene unas ventajas limitadas, pero cuando se utilizan varias tarjetas para formar un sistema distribuido de control éstas aumentan. Ya no sólo es importante el tamaño, sino el precio y la funcionalidad de cada unidad. Suponga por un momento que desea construir

[3]Considerar que se necesita el adaptador, lo del reset software y la diferencia entre el método en linux y msdos. es decir todo lo explicado en 6.6.1.

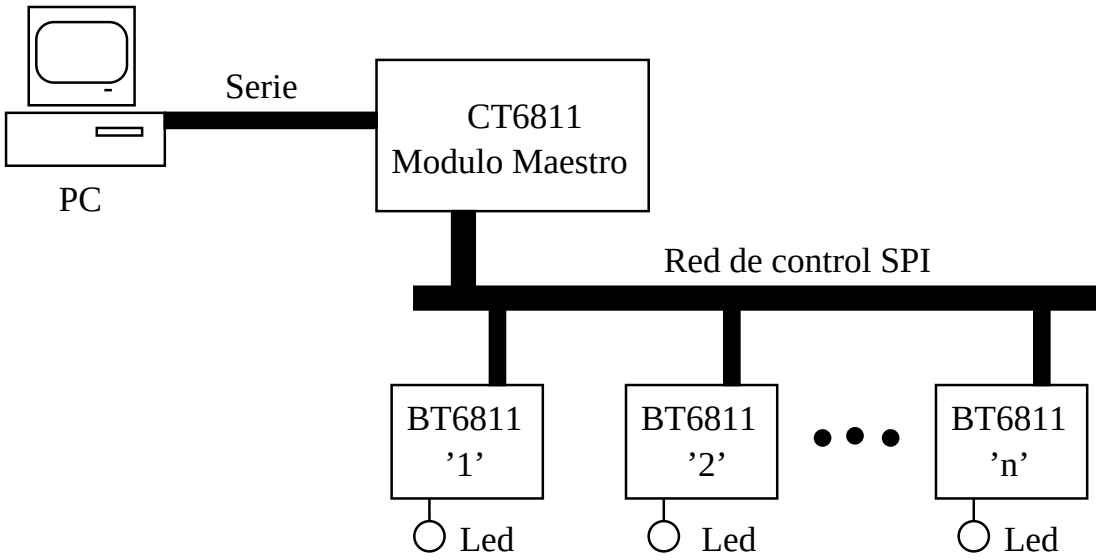


Figura 6: Representación de la red de BT6811.

un robot que tenga cuatro servomecanismos y que sea autónomo. Mediante una BT6811 aislada lo puede hacer, pero ¿qué ocurre si más tarde desea ampliar el robot ?, es decir, ponerle más motores. Es evidente que necesitará ampliar el hardware y lo más probable es que utilice otra BT6811. Ahora bien, el software hay que cambiarlo, ahora hay dos tarjetas en lugar de una: ¿cómo repartir el programa?, ¿ cómo hacerlo en futuras ampliaciones?. Son preguntas que sirven para demostrar la utilidad de la red de control. Si cada vez que se introduce un módulo hay que rectificar todos los programas, algo se está haciendo mal. Además, cuando se construye un robot es muy frecuente que a medida que se van obteniendo buenos resultados se vayan adquiriendo nuevos retos. Y está claro que si se sigue por el camino de modificar todo cada vez que se introduce un nuevo elemento, mal camino se ha escogido.

¿ Cómo se puede evitar lo anterior ?. La respuesta está en la modularidad. La mayoría de los proyectos se pueden subdividir en módulos más o menos independientes y con funcionalidades parecidas o totalmente dispares. De forma que para retocar un módulo no haga falta atacar a los otros. Además si se decidiese a añadir más, lo más seguro es que no se necesite reprogramar todo el sistema, o por lo menos de eso se trata. Todos estos módulos no deben actuar por su cuenta, sino que tienen que guardar un sincronismo entre ellos, deben intercambiar información o por lo menos tener uno maestro que controle a los demás. Sería como construir una plataforma móvil partiendo de bloques ya establecidos, es decir, utilizando un módulo motriz, un módulo sensorial, un módulo actuador y para tener todo comunicado un módulo maestro que gestione los anteriores. Si una vez desarrollado hay que añadir algún nuevo elemento, tan sólo hay que añadirlo a la estructura y modificar el módulo de gestión, los demás deberían permanecer inalterados. Para lograr esto se requiere una buena planificación del proyecto y una descripción buena de los interfaces entre módulos.

Debido a que los ejemplos suelen ser más útiles que las palabras expondremos uno. Aprovechando el LED de la BT6811 se diseñará una red de control que permita desde un módulo maestro encender y apagar cada LED de cada BT6811.

Lo primero que hay que hacer es pensar en la aplicación desde un punto de vista general: el esquema de la figura 6 servirá de ayuda. Se utilizará el PC para mandar las órdenes de encender y apagar el LED de cada BT6811. Habrá un módulo maestro que recibirá por el puerto serie los datos del PC y después los enviará por la red de control. A esta red están conectadas todas las BT6811 que uno quiera y a pesar de que todas ellas recibirán la misma orden, no habrá conflicto. Mediante un identificador cada BT6811 podrá saber si la orden va o no dirigida a ella y en el caso afirmativo procederá a encender o apagar su LED. En este ejemplo todos los módulos van a tener la misma función, pero puede haber otros casos en los que no. En estos, habrá que estudiar bien qué órdenes se van a transmitir por la red. Cuanto mejor se especifiquen éstas, más detallado se tendrá el protocolo final de la red y menos modificaciones se tendrán que hacer a largo plazo.

Identificador de destino	orden
1 Byte	1 Byte
Id	n_orden

Cuadro 2: Especificaciones de la trama de control.

En este ejemplo todos los módulos BT6811 tendrán la misma función: Encender y apagar su LED cuando se lo indique el módulo maestro. Con esta decisión se puede especificar el protocolo de la red o, lo que es lo mismo, el formato de las tramas y el procedimiento para enviarlas por la red.

Cada trama tendrá dos bytes, el primero identificará la BT6811 a la cual va dirigida la orden y el segundo será la propia orden, es decir, encender o apagar el LED, ver la tabla 2. El identificador **Id** será un número entre 1 y 9, y el código de orden, **n_orden,** será el carácter **'a'** para encender el LED y el carácter **'s'** para apagarlo.

Dicho lo anterior se va a proceder a programar el módulo BT6811. Con hacer un programa general bastará y para añadir más módulos tan sólo habrá que ir cambiando el valor de una etiqueta, en concreto la indicada en el programa por **M_ID**. El código del programa se detalla a continuación.

```
* ........................................
* . BTgen. Version 1.0                    .
* ........................................
* . Modulo general de la red de BT6811    .
* ........................................

*****************************************
* Identificador de la BT6811             *
* Incrementar una unidad en cada .nueva BT *
*****************************************

M_ID  equ '1'

*****************************************
* Registros y Puertos del 68hc11         *
*****************************************

* Puertos del 68hc11

PORTB equ $04

* Registros del SPI

PORTD   EQU $08
DDRD    EQU $09
SPCR    EQU $28
SPDR    EQU $2A
SPSR    EQU $29

*****************************************
* Máscaras de acceso                     *
*****************************************

LED  equ $10 ; Bit donde se encuentra el LED
```

```
*****************************************
* ------------------------------------- *
* |  PROGRAMA PRINCIPAL                 | *
* ------------------------------------- *
*****************************************

        ORG $F800    * Programa para un E2

*****************************************
* Inicialización del programa           *
*****************************************

inicializar
        LDS #$FF     * puntero de pila
        LDX #$1000   * acceder a registros

*--- Configuración del SPI (como esclavo)

        LDAA  #$3C    * SS de entrada
        STAA  DDRD,X

        LDAA  #$40    * Colector cerrado
        STAA  SPCR,X  * Modo esclavo el SPI


*****************************************
*     BUCLE PRINCIPAL                   *
*****************************************

inicio
        BSR  recibir_spi
        CMPA #M_ID        * verifica el ID
        BEQ  analizar     * SI->lee orden
        BSR  recibir_spi  * .NO->no hagas nada
        BRA  inicio

analizar BSR  recibir_spi * lee la orden
         CMPA #'a'        * ¿ Es a ?
         BEQ  led_on      * SI -> enciende led
         CMPA #'s'        * ¿ Es b ?
         BEQ  led_off     * SI -> apaga led
         BRA  inicio

*****************************************
* SUBRUTINAS DEL PROGRAMA               *
*****************************************

*.............................
*. Rutina enciende el LED     .
*.............................

led_on
        BSET PORTB,X LED
        BRA  inicio
```

```
*.............................
*. Rutina apaga el LED       .
*.............................

led_off
        BCLR PORTB,X LED
        BRA  inicio

* ........................................
* . Rutina que recibe un dato por el SPI    .
* . La rutina espera hasta recibir el dato  .
* . Entradas:Ninguna                        .
* . Salidas: El acumulador A contiene el dato .
* ........................................

recibir_spi
espera  BRCLR SPSR,X $80 espera  * espera dato
        LDAA  SPDR,X
        RTS

*****************************
* vectores de interrupcion   *
*****************************

         ORG $FFFE          * vector del reset
v_reset  FDB  #inicializar

         END
```

El programa no es complicado: el bucle principal lo único que hace es esperar a recibir el primer byte de cada trama enviada por el bus de control (formado por el SPI). Este byte (**Id**) identifica a la tarjeta a la que se ha enviado la orden, por eso lo siguiente que hace el bucle principal es comparar el byte con su identificador (**M_ID**). Si no coinciden ambos el bucle lee el siguiente byte de la trama, pero no lo utiliza, vuelve al bucle principal a esperar otro principio de trama. Por el contrario si ambos identificadores son iguales, el bucle lee el siguiente byte, lo analiza y en función de su contenido enciende o apaga el LED de la BT6811. Una vez terminado esto vuelve al bucle principal para seguir esperando una nueva trama. Tal vez la parte más complicada es la de recibir los datos por el bus SPI ya que para ello hay que configurarlo de una manera especial, en concreto en los módulos esclavos hay que programarlo con los parámetros siguientes: colector cerrado, señal SS de entrada y modo esclavo. Todo esto se hace en la inicialización del programa. Por último, como el microcontrolador que se está utilizando es el MC68HC11E2 pondremos el vector de reset apuntando al principio del programa para poder arrancar en modo *single-chip*[4].

El lector puede preguntarse cómo se conectan los módulos entre sí, la respuesta es sencilla, o se construye un cable que una los pines MOSI, SS, CLK y GND de todos los módulos incluyendo el maestro, o se construye un cable tipo bus con tantos conectores de salida como módulos tenga la red. Esta última opción es la recomendada en el ejemplo, aunque se podría tomar la precaución de cortar los cables correspondientes al puerto serie (PD1 y PD0) en el enganche al módulo maestro, de esa forma se evitarán conflictos con los puertos serie de los módulos esclavos. En la figura 7 se representa cómo construirse el cable de conexión del bus SPI. Por último, para conectar el PC al módulo esclavo se utilizará el cable serie proporcionado en la CT6811. La alimentación del sistema se introducirá, por ejemplo, por el jack de la CT6811 y

[4] En el apartado 'A.2: modos de funcionamiento' y en el 'A.6.2: funcionamiento en modo autónomo aislado' hay más información sobre el modo de arranque *single-chip*.

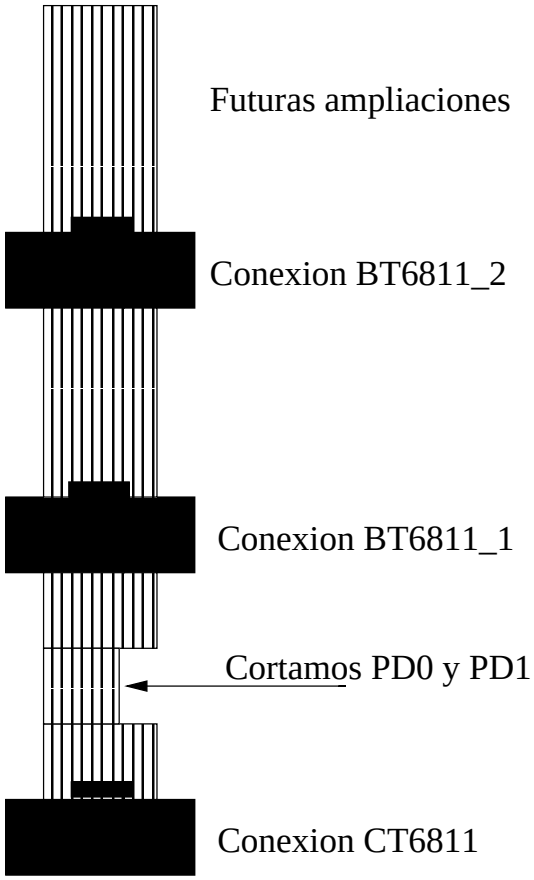


Figura 7: Cable para conectar los distintos módulos entre sí.

se distribuirá por toda la red uniendo con un cable todas las salidas positivas de las clemas V_TTL de cada BT6811 (ver figura 4).

Ya se tiene construida la red y el programa general de los módulos esclavos. Ahora se tiene que grabar éste en cada módulo, para lo que se utiliza toda la teoría explicada en el apartado anterior. Al final se configurará cada BT6811 en modo autónomo aislado, pero con un par de añadidos por utilizar la red SPI, en concreto:

1. Jumper JP1 quitado para configurar el modo *single-chip*.
2. Jumper JP2 quitado.
3. Jumper JP4 quitado ya que en la red se va a colocar una CT6811.
4. Jumper JP6 puesto ya que el control por el SPI va a utilizar la señal SS.
5. El resto de jumpers no tienen importancia en la aplicación.

Al grabar el programa en cada BT6811 hay que acordarse de modificar la etiqueta M_ID que identifica a cada módulo. Se empieza dándole el valor uno al primer módulo y se va incrementando a medida que se van grabando más módulos. En caso de dejar el mismo valor para todos, ocurrirá que todos los LEDs se encenderán o apagarán a la vez.

Ahora queda programar el módulo maestro, que como ya se ha dicho es una CT6811. Como va a estar conectado al PC se puede elegir entre tener grabado el programa en la memoria EEPROM o volcarlo cada vez que se utilice la aplicación. Para facilitar las cosas se hará de esta segunda forma, es decir se dejará configurada en modo entrenador[5]. Se usará el MCBOOT para enviar el programa maestro a la CT6811 y luego tecleando una serie de órdenes se verá como el LED de los módulos esclavos cambia de estado. A continuación se detalla el código del programa maestro.

```
*******************************************
*     Programa para el modulo maestro      *
*******************************************
*  El programa recibe  ordenes por el puerto*
* serie, las interpreta y manda las tramas  *
* adecuadas  por la  red_SPI para  que  le  *
* llegue la orden a la BT6811 adecuada.     *
*******************************************

* --- Puertos de entrada y salida

PORTA   EQU $0

* --- Registros del SCI

BAUD    EQU $2B
SCCR1   EQU $2C
SCCR2   EQU $2D
SCSR    EQU $2E
SCDR    EQU $2F

* --- Registros del SPI

PORTD   EQU $08
```

[5] Para ver la configuración en modo *'entrenador'* recurrir al *'manual de usuario de la CT6811'*. Este se encuentra disponible en la WEB de Microbótica S.L. [10].

```
DDRD    EQU $09
SPCR    EQU $28
SPDR    EQU $2A
SPSR    EQU $29


        ORG $F800  * micro E2

inicio  LDS   #$FF    * pila
        LDX   #$1000  * registros

*---- Configuracion del SCI

wait    BRCLR SCSR,X $40 wait

        LDAA  #$22      * Poner 22 para  7812 baudios
        STAA  BAUD,X    * Poner 30 para  9600 baudios
        LDAA  #$0
        STAA  SCCR1,X   * 8 bits de datos
        LDAA  #$0C      *.No usar interrupciones del SCI
        STAA  SCCR2,X   * Activar receptor y transmisor

*---- Configuracion del SPI (como maestro)

        LDAA  #$3C
        STAA  DDRD,X
        LDAA  #$50      * Colector cerrado
        STAA  SPCR,X    * Activa en modo maestro el SPI

******************************
*     BUCLE PRINCIPAL         *
******************************

bucle_prin
        JSR  leer_car    * leo carácter por el puerto serie
        JSR  enviar_car  * hace eco del caracter
        JSR  enviar_spi  * mando carácter al bus spi
        BRA  bucle_prin

*................................
*. Rutina que recibe un caracter .
*. por el puerto serie           .
*................................

leer_car
        BRCLR SCSR,X $20 leer_car
        LDAA  SCDR,X
        RTS

*................................
*.  Rutina que envia un caracter .
*.  por el puerto serie          .
*................................
```

```
enviar_car
        BRCLR SCSR,X $80 enviar_car
        STAA SCDR,X
        RTS

*.....................................
*. Rutina que envia un dato por el SPI .
*.....................................

enviar_spi
        LDAB  PORTD,X
        ANDB  #$DF
        STAB  PORTD,X    *  activarse al esclavo

        STAA  SPDR,X     * introduzco el dato a mandar
espera  BRCLR SPSR,X $80 espera

        LDAB  PORTD,X
        ORB   #$20
        STAB  PORTD,X    * Desactivamos al esclavo

        RTS

*********************************
*   Vector de interrupcion      *
*********************************

        ORG $FFFE
v_reset FDB #inicio

        END
```

El programa maestro tampoco es difícil de comprender y se podría reducir más, ya que al utilizar el modo entrenador el puerto serie se auto configura, igual que la pila, por lo que se podían haber ahorrado todas esas líneas de configuración. Las líneas de configuración del SPI vuelven a ser claves, aunque ahora se ha configurado como colector cerrado y como maestro.

Después de la configuración se ejecuta el bucle principal que lo único que hace es esperar a recibir datos del PC. En cuanto recibe uno hace eco y lo envía por el bus SPI a los módulos esclavos. Por lo tanto si con el MCBOOT se teclea ’1’ y luego ’a’ se debería encender el LED del módulo esclavo cuyo identificador M_ID es igual a uno. Si se hace lo mismo pero tecleando un ’2’ debería pasar lo mismo pero en el módulo esclavo cuyo M_ID fuera dos. Para apagar el LED de la BT6811 hay que pulsar primero el ’1’ y luego ’s’.

Si una vez terminada la aplicación se quiere añadir un nuevo módulo habrá que hacer lo siguiente:

1. Se incrementa el valor M_ID del programa de los módulos esclavos (BT6811).
2. Se graba dicho programa en el nuevo módulo y se configura en modo autónomo en red.
3. Se conecta el módulo al bus SPI y a la alimentación de 5 voltios.
4. El resto de módulos se dejan como están.

Tal como se dijo al principio del ejemplo al ser un sistema modular no ha hecho falta tocar ninguno de los módulos ya existentes para ampliar el sistema. El lector puede considerar que este ejemplo es muy sencillo, pero en la realidad hay muchos sistemas de control que se adaptan al ejemplo. De todas formas la

BT6811 todavía admite más complejidad en la red. Por ejemplo, en lugar de usar la señal SS común para toda la red se puede independizar (quitar el jumper JP6). En su lugar se conectará una salida del módulo maestro, como puede ser algún pin del puerto B. Haciendo esto se consigue una comunicación de ida y vuelta entre el maestro y el esclavo, de forma que se puede intercambiar información en ambos sentidos. Además, en lugar de ser la trama la que identifica el destino es el propio maestro el que lo hace. En este caso al incluir un nuevo módulo hay que retocar el módulo maestro para que lo reconozca, pero por el contrario, este nuevo módulo puede tener una funcionalidad distinta sin que afecte a la red. Como podrá apreciar el lector las posibilidades son muchas y cuanto más practique más irán surgiendo. Para adquirir experiencia se recomienda leerse el capítulo relacionado con el SPI del libro *Microcontrolador MC68hc11: Fundamentos, recursos y programación* [1].

Referencias

[1] “Microcontrolador MC68hc11: Fundamentos, recursos y programación”. Disponible on-line: http://www.iearobotics.com/proyectos/libro6811/libro6811.html