Cika Electrónica

Nota de Aplicación: CAN-072
Título: **Control de Acceso de muy bajo costo**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	26/02/07	port de CAN-017

Presentamos un control de acceso de muy bajo costo realizado con tarjetas o tags RFID. Utilizamos como lector al módulo GP8F-R2, de muy bajo costo, el que conectamos a un micro de la serie HT48E de Holtek para validar los RFID. Dada la capacidad de memoria EEPROM de estos micros, podemos almacenar hasta 24 RFIDs en memoria con el más pequeño.

Descripción del GP8F-R2

El módulo GP8F-R2 lee tarjetas o tags RFID read-only de 64-bits, código Manchester a 125KHz. Posee una salida para conectar un LED, que permanece encendido y aumenta su intensidad al aproximar un RFID. El ID correspondiente se entrega por un terminal en formato 8 bits serie asincrónico, a 9600 bps, sin paridad, a nivel lógico. El usuario puede optar por conectarlo a algun driver RS-232 para leerlo desde una PC, o conectarlo directamente a un microcontrolador o UART. El formato lógico responde al siguiente esquema, en ASCII:

```
<STX> <DATA (10 bytes)> <CR> <LF> <ETX>
```

STX: ASCII Start of Text (02h)
ETX: ASCII End of Text (03h)
CR: ASCII Carriage Return (0Dh)
LF: ASCII Line Feed (0Ah)

El campo de DATA es una representación en ASCII del ID del RFID, representando 5 bytes binarios (40 bits) de la siguiente forma:

byte	ASCII
C1	43 31

Por ejemplo, el ID `60 22 57 C0 31` se transmite de la siguiente forma:

```
02 36 30 32 32 35 37 43 30 33 31 0D 0A 03
```

Recordemos que este tipo de RFID tags de 64 bits 125 Khz utiliza 9 bits para header, 40 para datos, 14 bits de paridad y uno de stop.

Descripción del sistema de acceso propuesto

Se trata de un simple control de acceso donde al acercar un tag autorizado el sistema responde activando un terminal durante un tiempo determinado. El usuario puede utilizar este terminal para controlar un triac, un relé, o lo que prefiera, que a su vez controlarán un solenoide o pasarán información a otro sistema.

La operatoria del sistema es bastante básica, dado que se trata de una nota de aplicación, sin embargo se provee soporte para incorporar o autorizar nuevos IDs, mediante otro tag previamente autorizado, y una purga o borrado total de la base.

La operatoria propuesta es la siguiente:

1. Al arrancar el sistema, verifica la existencia de IDs en memoria.
2. Si no existen IDs precargados, activa momentáneamente la salida indicando esta situación, y permanece a la espera de un ID. Al ingresarse el primer ID, este pasa a ser el “maestro”, es decir, no se utilizará para ingresar sino para autorizar otros IDs.

3. Si ya existen IDs precargados (al menos uno), el sistema permanece en espera de un ID que conozca, respondiendo ante cualquiera de ellos diferente del “maestro”, pulsando el terminal de activación por un tiempo prefijado.
4. Si el ID reconocido es el “maestro”, y el pin de purga se encuentra en estado lógico 0, se indica esta condición mediante dos pulsaciones de la salida. El sistema queda entonces a la espera de un ID desconocido a autorizar, el cual memorizará, activando la salida. Si se acerca un tag con un ID conocido, se indicará su rechazo mediante dos pulsaciones de la salida y quedando nuevamente a la espera. Puede salirse de este estado acercando nuevamente la tarjeta “maestra”.
5. Si el ID reconocido es el “maestro”, y el pin de purga se encuentra en estado lógico 1, se procede a borrar toda la base de datos, excepto la tarjeta “maestra”. Se indica esta condición mediante tres pulsaciones de la salida.

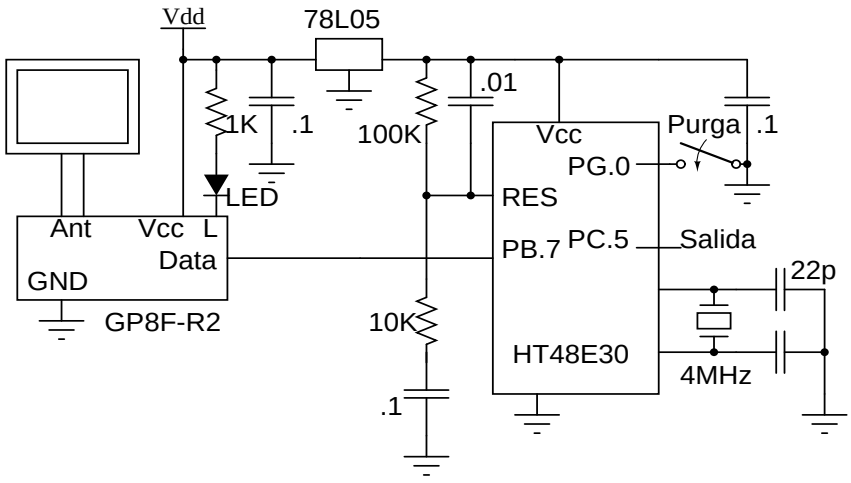
Los puntos más flojos de este sistema, en los que el usuario deberá desarrollar un poco más, son los siguientes:

Si se extravía un ID autorizado, no es posible eliminarlo de forma individual, debiendo purgarse la base de IDs, lo que ocasiona la pérdida de todos los demás, excepto el “maestro”. Implementar un borrado selectivo requiere un gran trabajo sobre la base de datos y alguna interfaz de edición, escapando al alcance de esta nota de aplicación.

La tarjeta o ID “maestra” deberá guardarse en lugar seguro, dado que no puede perderse ni caer en manos de personas no autorizadas. De extraviarse, cosa poco probable si se toman las debidas precauciones, se deberá recargar el firmware. Esto podría resolverse fácilmente mediante un pin extra, que al detectarse activo borre toda la base, incluyendo el ID “maestro”. Sin embargo, preferimos no complicar aún más el desarrollo y dejar pines libres para conectar resonadores o cristales (en caso de tener problemas con el oscilador interno), y una posible comunicación con otro sistema o control de otra salida. Queda a criterio del usuario final la última palabra.

Hardware

Conectamos directamente la salida del módulo GP8F-R2 a un pin del microcontrolador. Necesitamos otra entrada para señalizar la entrada en modo purga, para lo cual utilizamos PG0 y aprovechamos el pull-up interno. Tomamos la salida de PC5. El circuito se alimenta a 12V, según la especificación del GP8F-R2.



Software

Port serie asincrónico

Utilizamos una de las notas de aplicación de Holtek: HA0004e, y dado que el microcontrolador no tiene otra cosa interesante que hacer mientras espera por el ID, realizamos el port serie asincrónico con retardos por software como indica dicha nota. Incluimos los archivos para facilitar el desarrollo

Manejo de la EEPROM

Utilizamos también una de las notas de aplicación de Holtek: HA0086e, que nos reduce el problema a llamar a una subrutina. Incluimos los archivos para facilitar el desarrollo.

La primera vez que se ejecute el programa, deberá tener *00* en la primera posición de EEPROM, para detectar que debe cargar la tarjeta maestra. Para esto, utilizamos el Data EEPROM Editor (*Tools-> Data EEPROM Editor*) y hacemos download del archivo *acceso.mem* que hemos incluido.

Obtención de un ID

Como viéramos al describir el GP8F-R2, los 40 bits del ID son transmitidos como 10 bytes en hex, es decir, dos bytes ASCII representan un byte (8 bits) del ID, ordenados nibble alto-nibble bajo. Por consiguiente, leeremos 10 bytes conteniendo 10 nibbles y los convertiremos y ordenaremos, como se ve en el listado a continuación.

```
id_vars         .section 'DATA'
aux             db ?            ; auxiliar para cálculos
charcnt         db ?            ; cuenta de caracteres
IDbuf           db 5 DUP(?)

NUMIDs          EQU 0           ; # IDs guardadas
IDs             EQU 1           ; dirección donde comienzan las IDs


id_code         .section 'CODE'

getID:          call receive                            ; Espera STX
                sub A,2                                 ; STX ?
                snz Z
                jmp getID                               ; No, loop
                mov A,5
                mov charcnt,A                           ; Caracteres a recibir (2x)
                mov A,OFFSET IDbuf                      ; Inicializa puntero
                mov MP0,A
rxloop:         call receive                            ; Siguiente caracter
                call ASCII2bin                          ; a binario
                mov IAR0,A                              ; en el buffer
                swap IAR0                               ; nibble alto
                call receive                            ; Siguiente caracter
                call ASCII2bin                          ; a binario
                orm A,IAR0                              ; en buffer (nibble bajo)
                inc MP0                                 ; siguiente byte de ID
                sdz charcnt                             ; loop
                jmp rxloop
                call receive                            ; espera CR, descarta
                call receive                            ; espera LF, descarta
                call receive                            ; espera ETX
                sub A,3                                 ; ETX ?
                snz Z
                jmp getID                               ; No, loop
                ret

ASCII2bin:      sub A,041h
                snz C
                add A,7
                add A,10
                ret
```

Operaciones con IDs

Deberemos detectar si el ID recibido corresponde a alguno almacenado en la base, borrar la base, y almacenar un nuevo ID.

Para agregar un ID, por simpleza, dado que no soportamos borrado selectivo de IDs, incrementamos el indicador de cantidad de IDs almacenados y agregamos el nuevo ID al final de la base. De haberse saturado la capacidad de la base ignoramos el ID:

```
addID:          call numID                              ; Obtiene cantidad de IDs en A
                mov aux,A                               ; guarda en aux
                sub A,MAXIDs                            ; Base llena ?
                sz C
                ret                                     ; ignora
                mov A,aux
                inc ACC                                 ; incrementa NUMIDs
```

```
                mov EEDATA,A
                call EWEN                               ; Habilita escritura
                call WRITE                              ; guarda en EEPROM NUMIDs
                clr C                                   ; dirección en EEPROM (5xNUMIDs)
                mov A,aux
                rlc aux                                 ; x2
                rlc aux                                 ; x4
                add A,aux                               ; x5
                mov EEADDR,A
                inc EEADDR                              ; +1 (NUMIDs)
                mov A,5
                mov charcnt,A
                mov A, OFFSET IDbuf                     ; Inicializa puntero
                mov MP0,A
wIDloop:
                mov A,IAR0                              ; obtiene del buffer
                mov EEDATA,A
                call WRITE                              ; guarda en EEPROM
                inc MP0                                 ; siguiente byte en buffer
                inc EEADDR                              ; siguiente byte en EEPROM
                sdz charcnt                             ; loop
                jmp wIDloop
                jmp EWDS                                ; Inhabilita escritura
```

Reconocemos un ID almacenado mediante comparaciones sucesivas, buscando en forma secuencial desde el primero hasta el último:

```
matchID:
                clr EEADDR                              ; obtiene NUMIDs
                call READ                               ; lee EEPROM
                mov A,EEDATA
                mov aux,A                               ; inicializa contador
                inc EEADDR                              ; Apunta a principio de base
mIDl2:          mov A,5                                 ; longitud de ID
                mov charcnt,A
                mov A,OFFSET IDbuf                      ; Inicializa puntero
                mov MP0,A
mIDl1:          call READ                               ; lee EEPROM
                mov A,EEDATA
                sub A,IAR0                              ; compara con buffer
                snz Z                                   ; match ?
                jmp mIDnext                             ; no, siguiente ID
                inc MP0                                 ; sí, byte siguiente en buffer
                inc EEADDR                              ; y en EEPROM
                sdz charcnt                             ; loop
                jmp mIDl1
                mov A,aux                               ; MATCH!: ID# en A y Z reset
                clr Z
                ret
mIDnext:
                mov A,charcnt                           ; apunta a siguiente ID
                addm A,EEADDR
                sdz aux                                 ; hay suficientes ?
                jmp mIDl2
                set Z
                ret                                     ; No match, devuelve Z set
```

Para borrar la base, simplemente hacemos que la cantidad de IDs almacenados sea 1:

```
purgeDB:
                call EWEN                               ; Habilita escritura
                clr EEADDR                              ; apunta a NUMIDs
                mov A,1                                 ; borra todo (excepto 1º ID)
                mov EEDATA,A
                call WRITE                              ; hecho
                jmp EWDS                                ; Inhabilita escritura
```

Control de Acceso

Ya estamos entonces en condiciones de realizar el control de acceso, el cual implementa el protocolo que propusimos al comienzo de este artículo:

```
Purge   equ     PG.0

vars    .section 'DATA'
IDs             db ?                            ; # IDs en base

reset   .section at 0 'CODE'
                ORG 0
                jmp start

                ORG 00Ch
start:          clr PC
                clr PB
                 set PAC                        ; PA0-7 = inputs
                mov a,0F1h                      ; PB1-3 = outputs, PB0,4-7 = inputs
                mov PBC,a
                clr PCC                         ; PC0-7 = outputs
                set PGC                         ; PG0 = input
                mov A,01h
                mov BP,A                        ; bank 1 (EEPROM 'pins')
                mov A,040h
                mov MP1,A                       ; MP1 = EECR address
                call numID                      ; Obtiene # IDs en A
                mov IDs,A
                sz ACC                          ; Z = ninguno, obtiene master
                jmp main
                jmp GetMaster
main:           call getID                      ; Espera un ID
                call matchID                    ; La busca en la base
                sz Z                            ; Match ?
                jmp main                        ; No, loop
                sub A,IDs                       ; Mira si es el Master
                sz Z                            ; Master ?
                jmp Special                     ; Sí, modo Autoriza/purga
open:           call OpenLock                   ; No, abre
                jmp main

Special:
                call pulseLock                  ; Acknowledge (2 pulsos cortos)
                call HSdelay
                call pulseLock
                sz Purge                        ; purga ?
                jmp DOPurge
                call getID                      ; no, espera un ID
                call matchID                    ; lo busca en base
                sz Z                            ; Match ?
                jmp learn                       ; No, lo aprende
                sub A,IDs                       ; Sí, se fija si es el Master
                snz Z                           ; Master ?
                jmp Special                     ; No, ignora
                jmp open                        ; Sí, modo normal (abre)
learn:          call addID                      ; agrega a la base
                call OpenLock                   ; abre (ignora errores)
                jmp start

DOPurge:
                call purgeDB                    ; Purga base
                call HSdelay                    ; Acknowledge (3 pulsos cortos)
                call pulseLock
                jmp start                       ; restart

GetMaster:
                call pulseLock                  ; Acknowledge (1 pulso corto)
                call HSdelay
                call getID                      ; Espera un ID
                call addID                      ; lo guarda
                call pulseLock                  ; Acknowledge (1 pulso corto)
                call HSdelay
                jmp start                       ; restart
```

Las rutinas desarrolladas fueron escritas teniendo en cuenta las especificaciones del fabricante, para 4MHz de clock. La cantidad de calls anidadas se comprobó para un HT48E30. Para otras aplicaciones, se sugiere al interesado revisar que se cumpla la cantidad de calls permitidos por el hardware stack del micro de Holtek utilizado.