Cika Electrónica

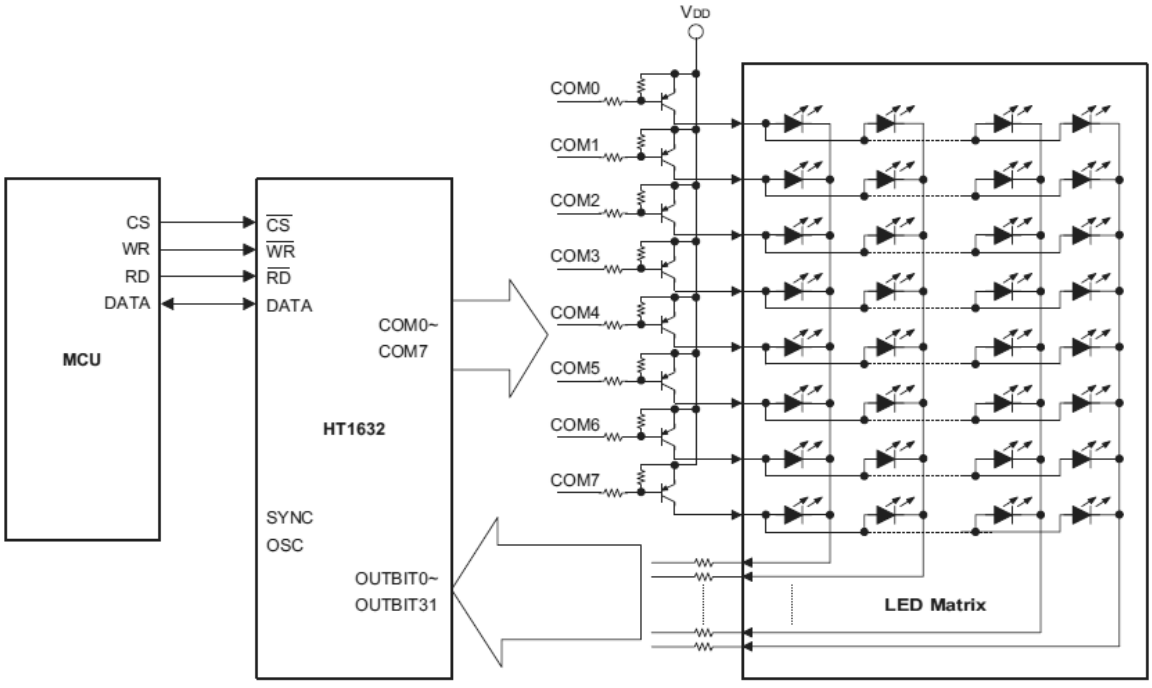
Comentario Técnico: CTC-065
Título: **HT1632, controlador de matriz de LEDs**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	18/02/11	

En este comentario técnico describimos brevemente el funcionamiento del controlador de matriz de LEDs de Holtek, HT1632, y damos un simple ejemplo de utilización en C.

Breve descripción del HT1632

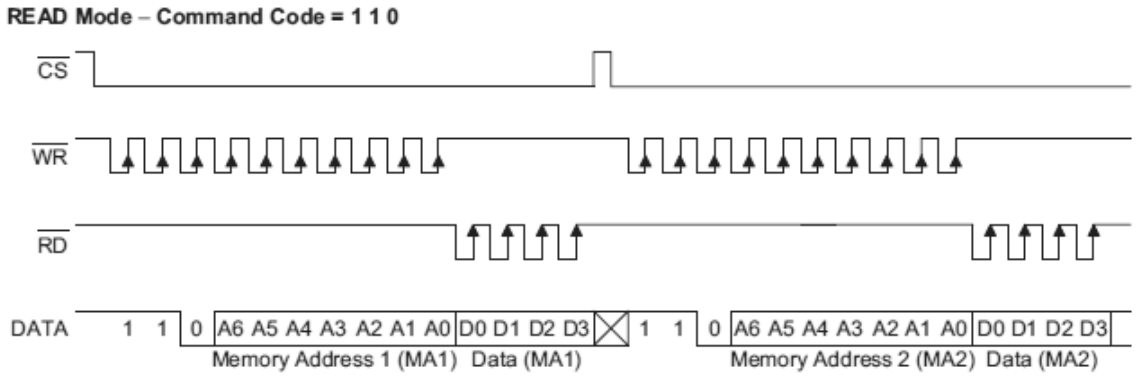
El HT1632 es un chip de 52-pines en formato QFP, capaz de controlar una matriz de LEDs de 32x8 ó de 24x16. Provee un control de intensidad general de 16-niveles, y una memoria interna para representar la matriz de LEDs de 96 palabras de 4-bits. Cada bit de cada palabra corresponde a una de las salidas comunes, mientras que las direcciones se asocian a una salida; por ejemplo, en modo 32x8, la salida 0 tiene dos direcciones de memoria asociadas, una para los comunes 0 a 3 y otra para los comunes 4 a 7.
Un ejemplo de utilización es el siguiente:

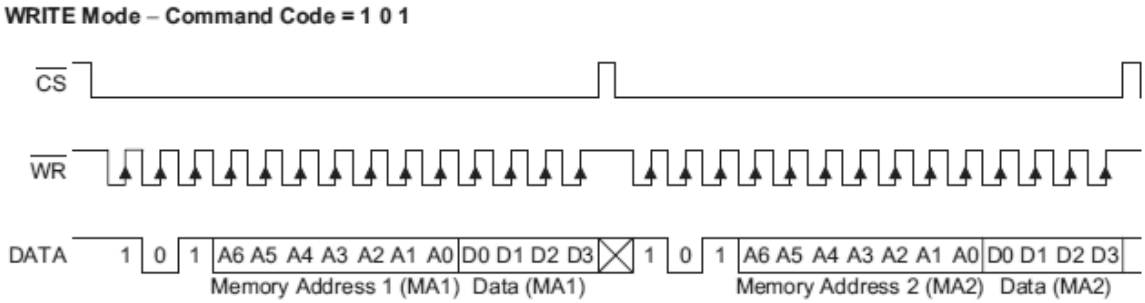


Varios HT1632 pueden conectarse en cascada para controlar una matriz más grande.

Comunicación con el procesador

La comunicación entre el HT1632 y el micro de nuestro agrado se realiza mediante un pin de CS para cada HT1632 y dos relojes para lectura y escritura más una señal de datos, para todo el conjunto. Los datos se presentan en la señal DATA y se indica lectura o escritura (quién controla la señal) mediante el clock correspondiente. La imagen siguiente muestra una lectura y una escritura simples. Para mayor información, consultar la hoja de datos del chip.





El listado siguiente muestra un ejemplo de esta operación. Primero escribimos una serie de rutinas elementales que nos permiten realizar el clocking y la lectura y escritura de varios bits. Las macros en mayúsculas corresponden a operaciones dependientes del micro elegido, y deberán ser escritas por el usuario.

```
void send_cbits(uint8_t cmdbit,uint8_t bits) // envía bits de comando
{
	DATA_ISOUT;			// configura pin como salida
	while(bits--){			// para cada bit
		WR_LO;			// pone clock en estado bajo
		if(cmdbit&0x80)		// si el dato es 1
			DATA_HI;	// pone pin en 1
		else			// si es 0
			DATA_LO;	// pone pin en 0
		WR_HI;			// pone clock en estado alto, indicando bit listo
		cmdbit<<=1;		// siguiente bit
	}
	DATA_ISIN;			// configura pin como entrada
}

uint8_t recv_4dbits(void)			// lee 4 bits de datos
{
uint8_t datbit=0,bits=4;

	while(bits--){				// para cada bit
		RD_LO;				// pone clock en estado bajo
		if(DATA)			// si es 1
			datbit|=(1<<3);		// coloco un 1 en D3
		else				// si es 0
			datbit&=~(1<<3);	// coloco un 0 en D3
		RD_HI;				// pone clock en estado alto, bit leído
		datbit>>=1;			// desplaza a la derecha
	}
	return(datbit);				// retorna con bits en nibble inferior
}

void send_4dbits(uint8_t datbit)		// envía 4 bits de datos
{
uint8_t bits=4;
```

```
        DATA_ISOUT;                             // configura el pin de datos como salida
        while(bits--){                          // para cada bit
                WR_LO;                          // pone clock en estado bajo
                if(datbit&0x01)                 // si es un 1
                        DATA_HI;                // pone un 1
                else                            // si es un 0
                        DATA_LO;                // pone un 0
                WR_HI;                          // pone clock en estado alto, bit listo
                datbit>>=1;                     // siguiente
        }
        DATA_ISIN;                              // configura pin de datos como entrada
}
```

En base a estas rutinas, escribimos funciones que nos permiten acceder a las operaciones elementales del chip:

```
#include "HT1632.h"

uint8_t HT1632_read(uint8_t address)            // lectura de una posición de memoria del chip
{
uint8_t data;

        CS_LO;                                  // chip select en estado bajo
        send_cbits(HT1632_READ,3);              // envía comando LEER
        send_cbits(address<<1,7);               // envía dirección de memoria (7-bits)
        data=recv_4dbits();                     // lee 4 bits
        CS_HI;                                  // chip select inactivo (alto)
        return(data);                           // retorna con el dato leído
}

void HT1632_write(uint8_t address,uint8_t data)      // escribe una posición de memoria
{
        CS_LO;                                  // chip select activo (bajo)
        send_cbits(HT1632_WRITE,3);             // envía comando ESCRIBIR
        send_cbits(address<<1,7);               // envía dirección de memoria (7-bits)
        send_4dbits(data);                      // escribe 4 bits
        CS_HI;                                  // chip select inactivo (alto)
}

void HT1632_command(uint8_t cmd)                // envía comando
{
        CS_LO;                                  // chip select activo (bajo)
        send_cbits(HT1632_CMD,3);               // indica COMANDO
        send_cbits(cmd,9);                      // envía el comando
        CS_HI;                                  // chip select inactivo (alto)
}

void HT1632_readbytes(uint8_t *bytes,uint8_t num)    // lee varios bytes seguidos
{
register uint8_t aux=(uint8_t)bytes;

        CS_LO;                                  // chip select activo (bajo)
        send_cbits(HT1632_READ,3);              // envía comando LEER
        send_cbits(aux<<1,7);                   // envía dirección de memoria
        while(num--)                            // para cada posición de memoria
                *(bytes++)=recv_4dbits();       // lee 4 bits y guarda en buffer, inc ptr
        CS_HI;                                  // chip select inactivo (alto)
}

void HT1632_writebytes(uint8_t *bytes,uint8_t num)   // escribe varios bytes seguidos
{
register uint8_t aux=(uint8_t)bytes;

        CS_LO;                                  // chip select activo (bajo)
        send_cbits(HT1632_WRITE,3);             // envía comando ESCRIBIR
        send_cbits(aux<<1,7);                   // envía dirección de memoria
        while(num--)                            // para cada posición de memoria
                send_4dbits(*(bytes++));        // lee de buffer y escribe 4 bits, inc ptr
        CS_HI;                                  // chip select inactivo (alto)
}
```

Inicialización y utilización

Las funciones a continuación constituyen un ejemplo de inicialización y operación del chip, por ejemplo, para mostrar caracteres de 5x7. En ellas se mantiene un mínimo uso del stack para permitir su utilización en micros con poco o casi nada de él.

```
void init_HT1632(void)
{
	CTRLC;                                  // macro de configuración de pines de control
	DATA_ISIN;                              // pin de datos como entrada
	HT1632_command(HT1632_CMD_SYS_DIS);     // empezamos de cero
	HT1632_command(HT1632_CMD_COM_OPT_N8);       // leer hoja de datos
	HT1632_command(HT1632_CMD_MASTER_MODE);      // ídem
	HT1632_command(HT1632_CMD_CLK_RC);           // ídem
	HT1632_command(HT1632_CMD_SYS_EN);           // ídem
	HT1632_setPWM(15);                      // máximo brillo
	HT1632_command(HT1632_CMD_LED_ON);      // enciende display
}

void HT1632_fill(uint8_t pattern)           // pinta con un patrón (0=borrar)
{
uint8_t i=64;

	CS_LO;                                  // chip select activo (bajo)
	send_cbits(HT1632_WRITE,3);             // envía comando ESCRIBIR
	send_cbits(0,7);                        // dirección de memoria 0
	send_4dbits(pattern);                   // escribe patrón
	while(--i){
		send_4dbits(pattern);
	}
	CS_HI;                                  // chip select inactivo (alto)
}

#include "font5x7.h"

void HT1632_string(uint8_t address,char *string)
{
char c;
uint16_t addr;
uint8_t cnt,aux;

	CS_LO;                                  // chip select activo (bajo)
	send_cbits(HT1632_WRITE,3);             // envía comando ESCRIBIR
	send_cbits(address<<1,7);               // dirección de memoria
	while(c=*(string++)){                   // para cada carácter del string
		addr=(uint16_t)5*(c-' ');       // lo ubica en la tabla
		for(cnt=0;cnt<5;cnt++){         // y lo dibuja
			aux=FONT5x7[addr+cnt];          // no es eficiente, pero es claro
			aux>>=1;
			send_4dbits(aux);       // enviando nibble por nibble
			SWAP(aux);              // que lamentablemente no están en orden
			send_4dbits(aux);       // por cada byte
		}
	}
	CS_HI;                                  // chip select inactivo (alto)
}
```

Por ejemplo, este fragmento de código inicializa el chip y escribe “HOLA!”:

```
#include "HT1632.h"

void main(void)
{
	init_HT1632();
	HT1632_cls();
	HT1632_string(0,"Hola!");
}
```