

DS1307

64 x 8 Serial Real Time Clock Con un ejemplo práctico



DS1307 64 x 8 Serial Real Time Clock

DESCRIPCIÓN.

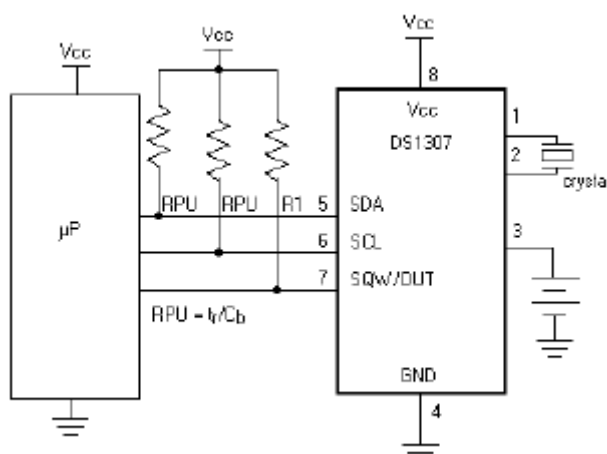
El DS1307 Real-Time-Clock Serie, es un dispositivo de bajo consumo de energía, completo con código binario decimal (BCD), reloj/calendario más 56 bytes de NV SRAM. Dirección y datos son transferidos a través de 2 hilos serie, bus bi-direccional. El reloj/calendario provee información de, segundos, minutos, horas, día, fecha, mes y año. El final de fecha de mes se ajusta automáticamente durante meses menores de 31 días, incluyendo correcciones para el año bisiesto. El reloj funciona en cualquiera formato de 24 horas o en 12 horas con indicador AM/PM. El DS1307 tiene incorporado un circuito de sensor de tensión que detecta fallas de energía y cambia automáticamente al suministro de batería de respaldo.

CARACTERÍSTICAS.

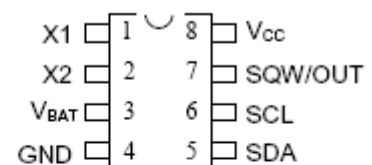
- ° Reloj en tiempo real (RTC) Cuenta segundos, Minutos, horas, fecha del mes, mes, día de la semana, y año con año bisiesto Compensación Válido hasta 2100.
- ° 56-Byte, con respaldo de batería, no volátil (NV) de RAM para almacenamiento de datos
- ° Interface Serie I2C.
- ° Onda-Cuadrada programable de la señal de salida.
- ° Detector Automático Fallo-Energía y Circuito Conmutación.
- ° Consume menos de 500nA en la batería -- Modo de copia de seguridad con el oscilador funcionando.
- ° Rango de temperatura Industrial Opcional: -40 ° C a +85 ° C
- ° Disponible en 8-Pin Plástico DIP o SO
- ° Reconocido Underwriters Laboratory (UL)

El Circuito Típico de funcionamiento y Configuraciones de pines aparecen al final de hoja de datos. Ver niveles de tensión y otras características en el propio DS.

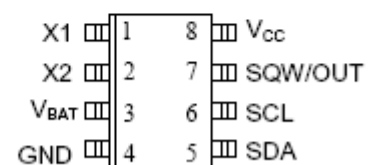
ASIGNACION DE PINES



Circuito Típico.



DS1307 8-Pin DIP (300 mil)



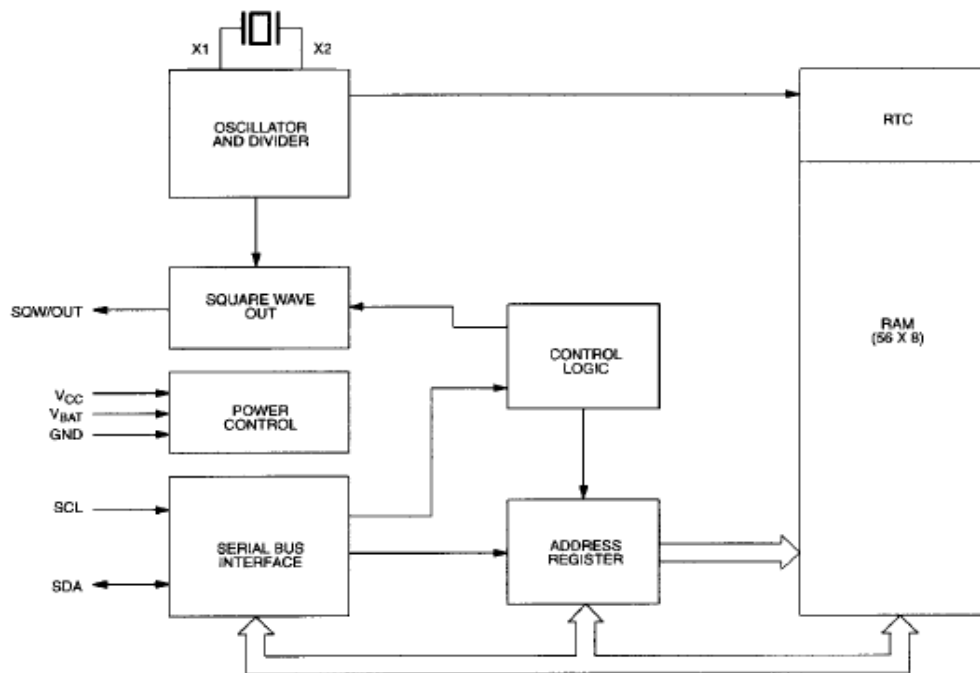
DS1307Z 8-Pin SOIC (150 mil)

Cápsulas.

OPERACIÓN.

El DS1307 funciona como un dispositivo esclavo en el bus serie. El acceso se obtiene mediante la aplicación de una condición de START (Inicio) y la prestación de un código de identificación del dispositivo seguido de una dirección de registro. Se puede acceder a registros posteriores de forma secuencial hasta que es ejecutada una condición STOP. Cuando VCC cae por debajo de $1,25 \times V_{BAT}$ un dispositivo en curso rescinde el acceso y restablece el contador de dirección de dispositivo. En este momento, pueden no ser reconocidas entradas al dispositivo para evitar que se escriban datos erróneos en el dispositivo por fuera de tolerancia del sistema. Cuando VCC cae por debajo de VBAT el dispositivo conmuta a batería de baja corriente modo de seguridad. Tras el encendido, el dispositivo conmuta de la batería a VCC cuando es mayor que $V_{BAT} + 0,2 \text{ V}$ y reconoce las entradas cuando VCC es mayor de $1,25 \times V_{BAT}$. El diagrama de bloques de la Figura 1 muestra los principales elementos del RTC serie.

DS1307 DIAGRAMA BLOQUE. Figura 1



DESCRIPCIÓN DE LA SEÑAL.

VCC, GND - La alimentación DC del dispositivo se ofrece en estos pines. VCC es entrada de +5 V. Cuando se aplican 5V dentro de límites normales, el dispositivo es totalmente accesible y los datos pueden ser escritos y leídos. Cuando una batería de 3V se conecta al dispositivo y VCC es inferior a $1,25 \times V_{BAT}$, se inhiben lectura y escritura. Sin embargo, la función de la hora normal no se ve afectada por la baja tensión de entrada. Como VCC caiga por debajo de VBAT, la RAM y el cronometro se cambian a la fuente de energía externa (nominal 3.0V DC) en VBAT.

VBAT - Entrada de Batería para cualquier célula de litio estándar 3V u otra fuente de energía. El voltaje de la batería debe ser mantenido entre 2,0 V y 3,5 V para su correcto funcionamiento. La tensión nominal de protección de escritura punto de disparo en el cual el acceso al RTC y la memoria RAM de usuario es denegado, es fijado por el circuito interno como nominal $1,25 \times V_{BAT}$. Un batería de litio con 48mAh o mayor mantendrá

copia de seguridad del DS1307 durante más de 10 años en ausencia de energía a 25 ° C. Reconocimiento UL asegura contra inversión de corriente de carga cuando se utiliza junto con un batería de litio.

Ver “Condiciones de accesibilidad” en: <http://www.maxim-ic.com/TechSupport/QA/ntrl.htm>.

SCL (Serial Clock Input) - SCL se utiliza para sincronizar el movimiento de datos en la interfaz serie, requiere una RPA (Resistencia de Polarización a Alto externa).

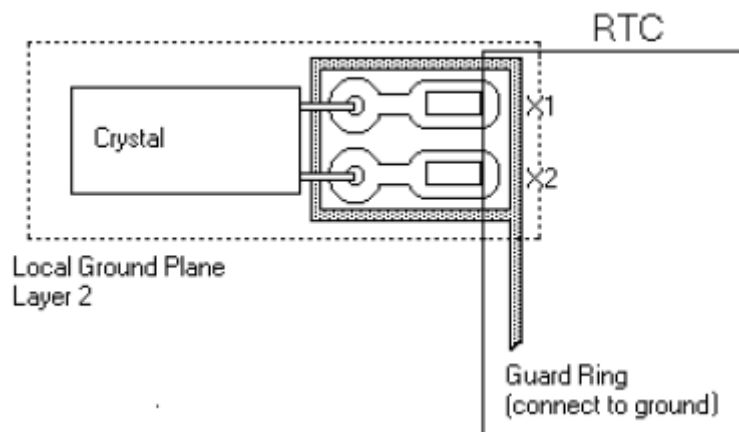
SDA (Serial Data Input/Output) - SDA es el pin entrada/salida para el interfaz 2-hilos serie. El SDA es el pin de drenaje abierto, que requiere una RPA (Resistencia de Polarización a Alto externa).

SQW/OUT (Onda Cuadrada/controlador de Salida) - Cuando se activa, el bit SQWE se establece en 1, el pin SQW/OUT es la salida de una de las cuatro frecuencias de onda cuadrada (1 Hz, 4 kHz, 8 kHz, 32 kHz). El pin SQW/OUT es de drenaje abierto y requiere una RPA (Resistencia de Polarización a Alto externa). SQW/OUT funcionará con cualquiera Vcc o Vbat aplicada.

X1, X2 - Conexiones para un cristal de cuarzo estándar 32.768kHz. El circuito oscilador interno está diseñado para funcionar con un cristal con una capacitancia de carga específica (CL) de 12.5pF.

Para obtener más información sobre la selección de cristal y las consideraciones de diseño de cristal, por favor, consulte Aplicación de Nota 58, “*Crystal Considerations with Dallas Real-Time Clocks.*” El DS1307 también puede ser impulsado por un oscilador externo de 32.768kHz. En esta configuración, el pin X1 está conectado con el oscilador externo de la señal y el pin X2 está flotando.

DISPOSICIÓN RECOMENDADA PARA CRISTAL.



RELOJ DE PRECISIÓN.

La precisión del reloj depende de la exactitud del cristal y la precisión de igualdad entre la carga capacitiva del circuito oscilador y la carga capacitiva para los que el cristal se ha recortado. Se añadirá el error adicional de frecuencia del cristal por la deriva causada por cambios de temperatura. El ruido exterior del circuito, junto al circuito oscilador puede resultar en el reloj corriendo rápido. Ver Nota de aplicación 58, “*Crystal Considerations with Dallas Real-Time Clocks*” para obtener información detallada.

Por favor, revise la Nota de Aplicación 95, “*Interfacing the DS1307 with a 8051-Compatible Microcontroller*” Para obtener información adicional.

RTC Y MAPA DE DIRECCIONES RAM.

El mapa de direcciones para registros del RTC y RAM del DS1307 es mostrado en la Figura 2. Los registros de RTC están situados en localizaciones de dirección 00h a 07h. Los registros RAM están situados en localizaciones de dirección 08h a 3Fh. Durante un acceso multi-byte, cuando el puntero llega a la dirección 3Fh, el fin del espacio de RAM, esto devuelve a la posición 00h, el principio del espacio de reloj.

DS1307 MAPA DE DIRECCIONES. Figura 2.

00H	SECONDS
	MINUTES
	HOURS
	DAY
	DATE
	MONTH
	YEAR
07H	CONTROL
08H - 3FH	RAM - 56 x 8

RELOJ Y CALENDARIO.

La información de tiempo de calendario se obtiene mediante la lectura de los bytes del registro correspondiente. La tabla 2, muestra los registros de RTC. El tiempo y calendario son establecidos o inicializados al escribir los bytes del registro correspondiente. El contenido de los registros de tiempo y calendario están en formato BCD. El registro del día de la semana se incrementa en la medianoche. Los valores que corresponden a los días de la semana son definidos por el usuario, pero debe ser secuencial (es decir, si 1 es igual a domingo, entonces 2 es igual a lunes, y así sucesivamente). Entradas de tiempo y fecha ilógicos causa una operación indeterminada. El Bit 7 del registro 0 es el bit interrupción de reloj alto (CH). Cuando este bit está establecido en 1, el oscilador está desactivado. Cuando se borra a 0, se habilita el oscilador.

Antes de hacer una lectura, se requiere hacer al menos una escritura, para enviar una dirección que pondrá el puntero del registro en el DS1307. En la página 8 de la hoja de datos sobre este tema, es tan rápida que, es fácil perderse. Exactamente dice:

Téngase en cuenta que el estado inicial (power-on) de todos los registros no está definido. Por lo tanto, es importante habilitar el oscilador (bit CH = 0) durante la configuración inicial.

Así pues, en su función de configuración, en primer lugar se ha de hacer un *Wire.send(0x00)* para establecer la dirección de registro en 0, a continuación, establecer el tiempo. Al principio de su función *loop()*, haga un *beginTransmission*, envíe otro 0x00, luego un *endTransmission*. Entonces *requestFrom*, etc., etc., durante el bucle. Entonces debería ser capaz de ver que el reloj hace tictac. Recuerde sin embargo que, usted cada vez está leyendo un byte en formato BCD, seguido por un registro de dirección, los registros posteriores se pueden acceder de forma secuencial.

El DS1307 se puede ejecutar en modo de 12 horas o 24 horas. El bit 6 del registro de las horas se define como bit del modo de seleccionar 12 o 24 horas. Cuando el modo seleccionado es alto, es de 12 horas. En el modo 12 horas, el bit 5 es el bit AM/PM con lógica alta es PM. En modo 24 horas, el bit 5 es el bit, segundas 10 horas (20 - 23 horas).

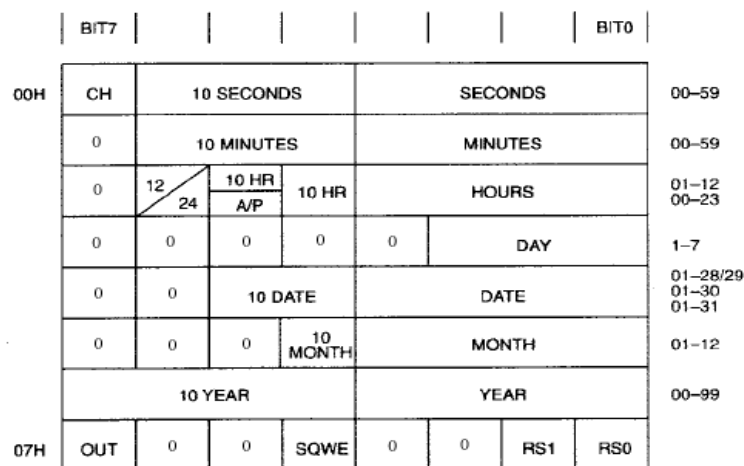
Al leer o escribir los registros de hora y fecha actual se transfiere a un segundo conjunto de registros (buffer), para evitar errores cuando los registros internos se actualizan. Cuando se leen los registros de hora y fecha, los buffers de usuario se sincronizan con los registros internos en cualquier START I²C. La información horaria se lee de estos segundos registros, mientras que el reloj sigue funcionando. Esto elimina la necesidad de volver a leer los registros, en caso de actualización de los registros internos durante una lectura. La cadena de divisores se reinicializa, cada vez que el registro segundos sea escrito. La transferencia de escritura en el I²C se produce con un reconocimiento desde el DS1307. Una vez que la cadena de divisores es reinicializada, para evitar problemas de volcado, los registros de fecha y tiempo restante deben ser escritos dentro de un segundo.

Table 2. Timekeeper Registers

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10' (decenas de Seconds)			Seconds				Seconds	00-59
01H	0	10' (decenas de Minutes)			Minutes				Minutes	00-59
02H	0	12	10'Hour	10'Hour	Hours				Hours	1-12 +AM/PM 00-23
		24	PM/AM							
03H	0	0	0	0	0	DAY			Day	01-07
04H	0	0	10'Date		Date			Date	Date	01-31
05H	0	0	0	10' Month	Month			Month	Month	01-12
06H	10' (decenas de Year)				Year			Year	Year	00-99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H-FFH

0 = Always reads back as 0. 10' = decenas

DS1307 REGISTROS CRONOMETRO Figura 3



REGISTRO DE CONTROL.

En el DS1307 el registro de control se usa para controlar el funcionamiento del pin SQW/OUT.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

Bit 7: OUT (Output control - control de Salida): Este bit controla el nivel de salida del pin SQW/OUT cuando la onda cuadrada de salida se desactiva. Si SQWE = 0, el nivel lógico en el pin SQW/OUT es 1, si OUT = 1 y SQW/OUT es 0 si OUT = 0.

Bit 4: SQWE (Square Wave Enable - Onda Cuadrada Habilitada): Este bit, cuando se establece a lógica 1, habilita la salida del oscilador. La frecuencia de onda cuadrada de salida depende del valor de los bits RS0 y RS1. Con la onda cuadrada de salida establecida a 1Hz, el reloj registra la actualización sobre el borde decreciente de la onda cuadrada.

Bits 1, 0: RS (Rango Seleccionado): Estos bits controlan la frecuencia de onda cuadrada de salida cuando han habilitado la salida de onda cuadrada. La tabla 1 muestra las frecuencias de onda cuadrada que pueden ser seleccionadas con los bits RS.

FRECUENCIA DE SALIDA CUADRADA Tabla 1

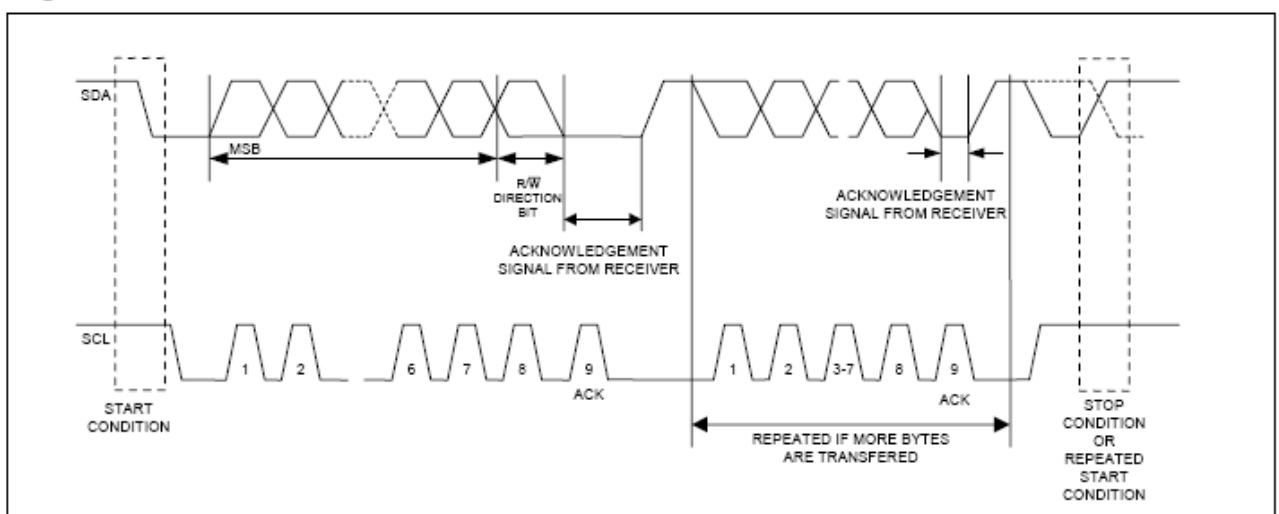
RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

I2C BUS DE DATOS SERIE.

El DS1307 soporta un bus bidireccional de 2 hilos y el protocolo de transmisión de datos. Un dispositivo que envía datos en el bus se define como un transmisor y un dispositivo de recepción de datos, como receptor. El dispositivo que controla el mensaje se llama maestro. Los dispositivos que son controlados por el maestro se denominan esclavos. El bus debe ser controlado por un dispositivo maestro que genera el reloj serie (SCL), controla el de acceso al bus y genera las condiciones de START y STOP. El DS1307 funciona como un esclavo en el bus de 2 hilos. Una configuración típica de buses que utilizan este protocolo de 2 hilos se muestra en la Figura 4.

CONFIGURACION TIPICA BUS I2C Figure 4

Figure 4. Data Transfer on I²C Serial Bus



La transferencia de datos sólo se podrá iniciar cuando el bus no está ocupado.

Durante la transferencia de datos, la línea de datos debe permanecer estable cuando la línea de reloj es ALTA. Los cambios en la línea de datos, mientras la línea de reloj es alta, se interpretan como señales de control.

En consecuencia, las siguientes condiciones de bus han sido definidas:

Bus no ocupado: Ambos datos y líneas de reloj permanecen ALTOS.

Inicio de Transferencia de datos: Un cambio en el estado de la línea de datos, de ALTO a BAJO, mientras el reloj es ALTO, define una condición de START.

Transferencia de datos de Parada: Un cambio en el estado de la línea de datos, de BAJO a ALTO, mientras la línea de reloj es ALTA, define la condición de STOP.

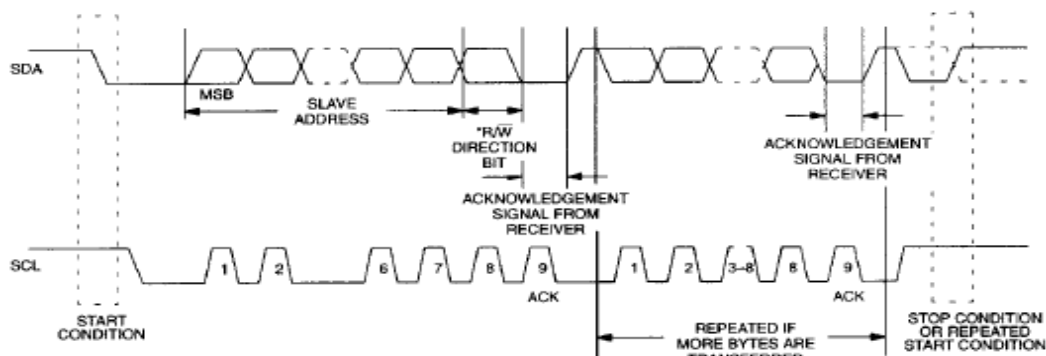
Datos válidos: El estado de la línea de datos representa datos válidos cuando, después de una condición de START, la línea de datos es estable durante del período ALTO de la señal de reloj. Los datos de la línea deben ser cambiados durante el período BAJO de la señal de reloj. Hay un pulso de reloj por bit de datos.

Cada transferencia de datos es iniciada con una condición de START y termina con una condición de STOP. El número de bytes de datos transferidos entre las condiciones de START y STOP no está limitado y se determina por el dispositivo maestro. La información se transfiere byte a byte y cada receptor reconoce con un noveno bit. Dentro de los datos específicos del bus de 2 cables de un modo regular (100 kHz frecuencia de reloj) y de un modo rápido (400kHz frecuencia de reloj) están definidas. El DS1307 funciona en el modo normal (100 kHz) solamente.

Reconocido (acknowledge): Cada dispositivo de recepción, cuando se le habla, está obligado a generar un reconocido después de la de recepción de cada byte. El dispositivo maestro debe generar un pulso de reloj extra que se asocia con este bit reconocido.

Un dispositivo que es reconocido ha de polarizar a masa la línea SDA durante el pulso de reloj reconocido de tal manera que la línea SDA sea estable BAJO durante el período ALTO de reconocido del pulso de reloj relacionado. Desde luego, la configuración y tiempos de espera deben ser tenidos en cuenta. Un maestro debe señalar un final de datos al esclavo no generando un bit reconocido en el último byte que ha sido registrado en el esclavo. En este caso, el esclavo debe dejar la línea de datos ALTA para permitir al maestro generar la condición de PARADA o STOP.

TRANSFERENCIA DE DATOS EN 2-WIRE BUS SERIE Figura 5



Dependiendo del estado del bit de R/W , dos tipos de transferencia de datos son posibles:

1. La transferencia de datos desde un transmisor maestro a un receptor esclavo. El primer byte transmitido por el maestro es la dirección de esclavo. Sigue después una

serie de bytes de datos. El esclavo devuelve un bit reconocido después de cada byte recibido. Los datos se transfieren primero con el bit más significativo (MSB).

2. La transferencia de datos desde un transmisor maestro a un receptor esclavo. El primer byte (dirección del esclavo) es transmitido por el maestro. El esclavo entonces devuelve un bit reconocido. Esto es seguido por el esclavo que transmite un número de bytes de datos. El maestro devuelve un bit reconocido después de todos los bytes recibidos, otro que no sea el último byte. Al final del último byte recibido, un “no reconocido” es devuelto.

El dispositivo maestro genera todos los impulsos de reloj serie y las condiciones de START y STOP. Una transferencia es terminada con una condición de STOP o con una condición de START repetida. Ya que una condición de START repetida es también el comienzo de la siguiente transferencia serie, el bus no será liberado. Los datos se transfieren primero con el bit más significativo (MSB).

El DS1307 puede funcionar en los dos modos siguientes:

1. **El modo de receptor de esclavo (DS1307 modo de escritura):** Datos serie y reloj se reciben a través de SDA y SCL. Después de cada byte recibido un bit de reconocido es transmitido. Las condiciones START y STOP son reconocidos como el comienzo y el final de una transferencia en serie. La dirección de reconocimiento se realiza por el hardware después de la recepción de la dirección de esclavo y bit de dirección (véase la figura 6). El byte de la dirección es el primer byte recibido después de que la condición de START es generada por el maestro. El byte dirección esclavo, contiene la dirección 7-bit de DS1307, que es 1101000, seguida del bit dirección (R/W) que, para la escritura, es un 0. Después de recibir y decodificar el byte dirección esclavo, el DS1307 sacará un reconocido en la línea SDA. Después de que el DS1307 reconoce la dirección esclavo + el bit escribir, el maestro transmite una palabra de dirección al DS1307. Esto establecerá el puntero del registro en el DS1307. El maestro entonces comenzará a transmitir cada byte de datos con el DS1307 reconociendo cada byte recibido. El maestro generará una condición de STOP para terminar la escritura de datos.

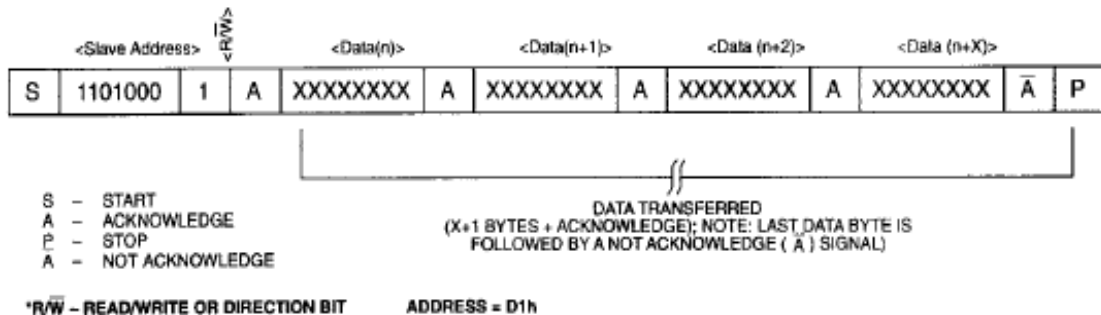
ESCRIBIR DATOS - MODO RECEPTOR ESCLAVO Figura 6



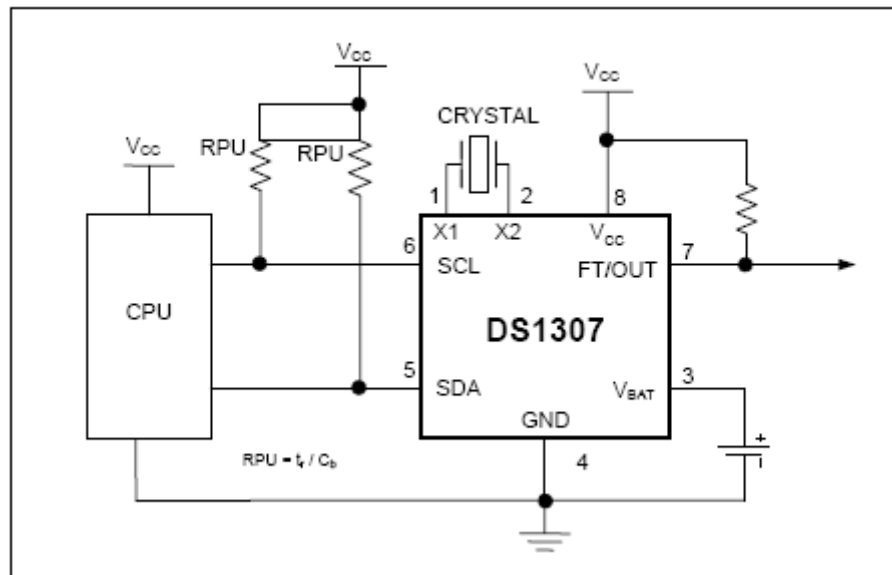
2. **El modo transmisor esclavo (DS1307 modo de lectura):** El primer byte se recibe y se maneja como en el modo receptor esclavo. Sin embargo, en este modo, el bit de *direction indicará que la dirección de transferencia es invertida. Datos en serie se transmiten en el SDA por el DS1307, mientras que el reloj serie es la entrada en SCL. START y STOP son condiciones de reconocido como comienzo y final de una transferencia en serie (véase Figura 7). El byte de dirección es el primer byte recibido después de que la condición de arranque es generada por el maestro. El byte de la dirección contiene la dirección de 7-bit DS1307, que es 1101000, seguido por el bit de * dirección (R/W) que, para una lectura, es un 1. Después de recibir y decodificar la dirección de byte el dispositivo introduce un reconocido en la línea SDA. El DS1307 entonces comienza a transmitir los datos que comienzan con

la dirección de registro indicada por el indicador de registro. Si el puntero de registro no es escrito antes de la iniciación de un modo de leer la primera dirección que es leída, es la última almacenada en el registro puntero. El DS1307 tiene que recibir un “no reconocido” para poner fin a una lectura.

LEER DATOS – MODO TRANSMISOR ESCLAVO Figure 7



TYPICAL OPERATING CIRCUIT



SOLUCIÓN DE PROBLEMAS.

Esta sección es un resumen de las causas más frecuentes de las inexactitudes del reloj en tiempo real. La mayoría de estos problemas se han mencionado anteriormente, pero se repiten aquí como una referencia rápida. En esta sección se ha dividido en tres partes. La primera parte se consideran los factores que causan un reloj de tiempo real a correr muy rápido y la segunda parte tendrá en cuenta los factores que causan un reloj en tiempo real a correr muy lento. El tercera parte se ocupa con los relojes que no se ejecutan.

RELOJES RÁPIDOS.

Los siguientes son los argumentos más comunes que causan un reloj de cristal basado en tiempo real para correr rápido.

1. El acoplamiento de ruido en el cristal de las señales adyacentes: Este problema ha sido ampliamente cubierto por encima.

El acoplamiento de ruido suele provocar en un reloj de tiempo real sea manifiestamente inexacta.

2. **Cristal incorrecto:** Un reloj de tiempo real normalmente correrá rápido, si se utiliza un cristal con una capacidad de carga específica (CL) de más de 6 pF. La gravedad de la falta de precisión depende del valor de CL.

Por ejemplo, utilizar un cristal con un CL, de 12 pF hará que el reloj de tiempo real corra unos 3-4 minutos por mes rápido.

RELOJES LENTOS.

Los siguientes son los escenarios más comunes que causan que un reloj en tiempo real basado en cristal corra lento.

1. **Rebasamiento en tiempo real de pines de entrada del reloj:** Es posible hacer correr despacio un reloj de tiempo real, pasa por detener el oscilador periódicamente. Esto puede ser logrado sin darse cuenta por las señales de entrada ruidosa al reloj de tiempo real. Si una señal de entrada se eleva a una tensión que sea mayor que la caída de diodo ($\sim 0,3$ V) por encima de VDD, el diodo de protección contra descargas electrostáticas ESD para el pin de entrada a la polarización, permitiendo que el sustrato sea inundado con la corriente. Esto, a su vez, detiene el oscilador hasta que el voltaje de la señal de entrada disminuye por debajo de una caída en el diodo por encima de VDD.

Este mecanismo puede hacer que el oscilador se pare con frecuencia, si las señales de entrada son ruidosas. Por lo tanto, debería ser tomado con cuidado para asegurar que no se rebasen las señales de entrada.

Otra situación que es común para que se rebase el problema, es tener una entrada al reloj de tiempo real en 5 voltios cuando el reloj de tiempo real está en el modo de respaldo de batería. Esto puede ser un problema en sistemas que ciertos circuitos se cierran sistemáticamente, pero mantienen otros funcionando. Es muy importante asegurarse de que no hay señales de entrada al reloj de tiempo real que sean mayor que el voltaje de batería cuando el dispositivo está en el modo de respaldo de batería.

2. **Cristal incorrecto:** En un tiempo real, el reloj típicamente correrá lento, si es usado un cristal con una capacitancia de carga específica (CL) menor de 6 pF. La gravedad de la inexactitud depende del valor de CL.

3. **Capacitancia parásita:** La capacitancia parásita entre los pines de cristal puede ralentizar la marcha de un reloj en tiempo real. Por lo tanto el cuidado debe ser tomado diseñando la disposición PCB para asegurar que la capacitancia parásita se mantenga a un mínimo.

4. **Temperatura:** La temperatura adicional de funcionamiento es la pérdida de temperatura del cristal, el cristal oscilará más despacio. Mirar Figuras 3 y 4.

RELOJ NO CORRE.

Los siguientes son los argumentos más comunes que causan que un cristal de reloj basado en tiempo real no correr.

1. El problema más común cuando el reloj no funciona es que el bit CH (Clock Halt - detener reloj) o EOSC (enable oscillator - habilitar oscilador) no se ha establecido o se borra, como se requiere. Muchos RTC Dallas Semiconductor incluyen un circuito que impedirá al oscilador correr cuando el suministro se aplicó por primera vez. Esto

permite a un sistema esperar el envío al cliente, sin llamar la alimentación de la batería de reserva. Cuando el sistema es alimentado por primera vez, el programa debe habilitar el oscilador y pedir al usuario anotar la hora y fecha correctas.

2. Cargas parásitas causadas por la condensación, incompleta retirada de flujo de soldar u otras cargas pueden impedir correr al oscilador.

3. Pines de suministro flotante. Cualquier entrada no utilizada, como Vbat, debe ser conectado a tierra. Si un pin de suministro queda flotando, la comunicación con el RTC puede no funcionar.

Cristales de montaje superficial pueden tener algún pin N/C (no conectado). Asegúrese que los pines correctos del cristal son conectados a pines de X2 y X1. Note que el circuito oscilador sobre Dallas RTCs son de bajo consumo y la señal en los pines del oscilador de entrada puede ser sólo unos pocos cientos de milivoltios pico a pico.

Práctico Reloj con el DS1307.

Poner en práctica este reloj, supone dar un estímulo al lector para que, se ejercite en la puesta en marcha de ejemplos que le lleven a mejorar su aprendizaje.

En este ejercicio, el propio código nos proporcionará la posibilidad de interactuar con el programa cargado en el microprocesador. Aprovecharemos las posibilidades que nos ofrecen Arduino y sus bibliotecas, para acercarnos a la programación de estos micros.

Este es el listado del código:

```

/*
* by <http://www.combustory.com> John Vaughters
* Credit to: Maurice Ribble -
* code RTC Control v.01
*
* Modificado el: 05-09-2010.
* by V. Garcia v.3.1
* <http://www.hispavila.com>
*
* Con este código se puede establecer la fecha y la hora, recuperar la fecha y la hora
* y usar la memoria adicional de un chip DS1307 RTC.
* El programa también pone todo el espacio de memoria suplementaria a 0xff.
* Método de Comunicación Serie con el Arduino que, utiliza un Caracter de texto para
cada
* orden descrita a continuación. También, admite el caracter en minúscula.
*
* Comandos: T(00-59)(00-59)(00-23)(1-7)(01-31)(01-12)(00-99)
* T(sec)(min)(hour)(dayOfWeek)(dayOfMonth)(month)(year) - T pone la fecha del chip
RTC DS1307.
* Ejemplo para poner la fecha y hora: 02-Feb-09 @ 19:57:11 para el día 3 de la semana,
* use la orden - T1157193020209 // T11 57 19 3 02 02 09.
* Q(1-2) :- (Q1) Consulta Memoria (Q2) RTC - Volcado Memoria
* mm:ss:hh:ds:DD:MM:AA
* T20:11:18: 4:15:10:09
*

```

```

* Ahora, puedes leer la hora y día de la fecha con introducir 'l' o 'L'
* Para poner en hora y la fecha del chip RTC DS1307, usa t o T:
* mm:ss:hh:ds:DD:MM:AA
* T20:45:17: 3:06:10:10
* ultimo: T2031162021110
*
* Al aplicar la librería RTCLib, se puede sincronizar la fecha del PC. Al compilar el
* programa y ejecutarlo inmediatamente, se muestra la fecha del PC.
*
* El código funciona bien, con 7634 bytes Arduino 0013
*
* Modificado el: 11-01-2010.
* by <http://www.hispavila.com> V. Garcia v.03.1
*
*/

```

```

#include "Wire.h"
#include <RTCLib.h> //
#define DS1307_I2C_ADDRESS 0x68 // Esta es la direccion I2C

RTC_DS1307 RTC;

// Variables Globales RTC
int command = 0; // Es el comando de carácter, en formato ASCII, enviados desde el puerto serie
int i;
// long previousMillis = 0; // almacenará la última vez que Temp se ha actualizado
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
byte test;

byte SW0 = 4;
byte blinkPin = 13; // activa el LED cada consulta

// Convierte números decimales normales a binario codificado decimal
byte decToBcd(byte val) {
    return ( (val/10*16) + (val%10) ); }

// Convierte binario codificado decimal a números decimales normales
byte bcdToDec(byte val) {
    return ( (val/16*10) + (val%16) ); }

// 1) Establece la fecha y la hora en el ds1307
// 2) Inicia el reloj
// 3) Establece el modo de hora de reloj a 24 horas
// Se supone que está pasando números válidos, probablemente
// tendrá que controlar poner números válidos.
void setDateDs1307() {

// Uso de (byte) tipo de conversión ASCII y matemáticas para alcanzar el resultado.
    second = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48));
    minute = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48));
    hour = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48));

```

```

dayOfWeek = (byte) (Serial.read() - 48);
dayOfMonth = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48));
month = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48));
year = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48));
Wire.beginTransaction(DS1307_I2C_ADDRESS); // Empieza transmisión.
Wire.send(0x00);
Wire.send(decToBcd(second)); // bit 0 a 7 Inicia el reloj
Wire.send(decToBcd(minute));
Wire.send(decToBcd(hour)); // Si quiere 12 horas am/pm tiene que poner
// bit 6 (también tiene que cambiar readDateDs1307)
Wire.send(decToBcd(dayOfWeek));
Wire.send(decToBcd(dayOfMonth));
Wire.send(decToBcd(month));
Wire.send(decToBcd(year));
Wire.endTransmission(); // Termina transmisión
}
// Extrae la fecha y el tiempo del ds1307 e imprime el resultado
void getDateDs1307() {

// Resetea el registro puntero. Es necesario que apunte a 00.
Wire.beginTransaction(DS1307_I2C_ADDRESS);
Wire.send(0x00);
Wire.endTransmission();
Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

// Alguna necesitará enmascarar porque ciertos bits son bits de control
second = bcdToDec(Wire.receive() & 0x7f);
minute = bcdToDec(Wire.receive());
hour = bcdToDec(Wire.receive() & 0x3f); // Tiene que cambiar esto a 80 para 12 hora am/pm.
dayOfWeek = bcdToDec(Wire.receive());
dayOfMonth = bcdToDec(Wire.receive());
month = bcdToDec(Wire.receive());
year = bcdToDec(Wire.receive());

Serial.print("20");
if (year < 10) Serial.print("0"); Serial.print(year, DEC);
Serial.print("-");
// Serial.print(month, DEC);
switch (month) { // pone el nombre en e del mes
  case 1: Serial.print("Ene"); break;
  case 2: Serial.print("Feb"); break;
  case 3: Serial.print("Mar"); break;
  case 4: Serial.print("Abr"); break;
  case 5: Serial.print("May"); break;
  case 6: Serial.print("Jun"); break;
  case 7: Serial.print("Jul"); break;
  case 8: Serial.print("Ago"); break;
  case 9: Serial.print("Sep"); break;
  case 10: Serial.print("Oct"); break;
  case 11: Serial.print("Nov"); break;
  case 12: Serial.print("Dic"); break;
}
}

```

```

    }
    Serial.print("-");
    if (dayOfMonth < 10) Serial.print("0"); Serial.print(dayOfMonth, DEC);

    // Serial.print(" Hoy es:"); // Esto pone nombre del dia de la semana:
    switch (dayOfWeek) {
        case 1: Serial.print(" Lunes"); break;
        case 2: Serial.print(" Martes"); break;
        case 3: Serial.print(" Miercoles"); break;
        case 4: Serial.print(" Jueves"); break;
        case 5: Serial.print(" Viernes"); break;
        case 6: Serial.print(" Sabado"); break;
        case 7: Serial.print(" Domingo"); break;
    }
    Serial.print(" ");
    if (hour < 10) Serial.print("0"); Serial.print(hour, DEC);
    Serial.print(":");
    if (minute < 10) Serial.print("0"); Serial.print(minute, DEC);
    Serial.print(":");
    if (second < 10) Serial.print("0"); Serial.print(second, DEC);
    Serial.println();
}

//
// cambiar el estado de un pin
void toggle(int pinNum) {
    int pinState = digitalRead(pinNum);
    pinState = !pinState;
    digitalWrite(pinNum, pinState);
}

void setup() {
    Wire.begin();
    Serial.begin(57600);
    pinMode(blinkPin, OUTPUT); // para el LED
    digitalWrite(blinkPin, 0);
    // Estas 4 líneas sirven para conectar directamente la placa del RTC al Arduino
    // esto proveerá de alimentación al RTC. Si no se usan, deberían comentarse.
    pinMode(17, OUTPUT); // pone como salida el A3 = digital 17
    pinMode(16, OUTPUT); // pone como salida el A2 = digital 16
    digitalWrite(17, HIGH); // activa las RPA a alto (pull-ups on)
    digitalWrite(16, LOW); // activa las RPB a bajo (pull-ups off)

    pinMode(SW0, INPUT); // para este use un interruptor deslizante
    digitalWrite(SW0, HIGH); // activa las RPA a alto (pull-ups on)

    RTC.begin();

    if (!RTC.isrunning()) {
        Serial.println("RTC is NOT running!"); }
    // La siguiente línea establece al RTC, la fecha y hora de su PC

```

```

// en que este boceto fue compilado, descomentar para poner en hora.
RTC.adjust(DateTime(__DATE__, __TIME__));
}

void loop() {
  if (Serial.available()) // busca caracter en serie y procesa si se encuentra
  {
    command = Serial.read(); // lee entrada serial
    if (command == 84 or command == 116) // Si command = "T" o "t" xa establecer fecha
    {
      setDateDs1307();
      getDateDs1307();
      Serial.println(" ");
    }
    else if (command == 81 or command == 113) //Si command = "Q" o "q" funciones de memoria
    {
      delay(50); // habia un 100
      if (Serial.available())
      {
        command = Serial.read();
        if (command == 49) // Si command = "1" RTC1307 Inicializa Memoria -
        {
          // Todos los datos se establecen en 255 (0xff).
          // Por lo tanto 255 o 0 será un valor no válido.
          //255 será el valor de inicio y 0 será cosiderado un error
          Wire.beginTransmission(DS1307_I2C_ADDRESS);
          // que se produce cuando el RTC está en modo de batería.
          Wire.send(0x08); // Ajusta el registro puntero un poco más allá los registros de fecha/hora.
          for (i = 1; i <= 27; i++)
          { Wire.send(0xff);
            delay(60); // 100. Al poner 50 reduce tiempo respuesta. Ajusta el reloj
          } // las lineas comentadas que siguen, no parecen necesarias. descomentar si procede.
          Wire.endTransmission();
          getDateDs1307();
          Serial.println(": RTC1307 Initialized Memory");
        }
        else if (command == 50) // Si command = "2" Volcado de Memoria RTC1307
        { getDateDs1307();
          Serial.println(": RTC 1307 Dump Begin");
          Wire.beginTransmission(DS1307_I2C_ADDRESS);
          Wire.send(0x00);
          Wire.endTransmission();
          Wire.requestFrom(DS1307_I2C_ADDRESS, 64);
          for (i = 1; i <= 64; i++)
          { test = Wire.receive();
            Serial.print(i);
            Serial.print(":");
            Serial.println(test, DEC);
          }
          Serial.println(" RTC1307 Dump end");
        }
      }
    }
  }
}

```

```

}
// para, leer la hora actual
else if (command == 76 or command ==108) //Si command = L o l
{
  getDateDs1307();
}
else if (command == 65 or command == 97 ) //Si command = a o A, toma la hora del PC
  ahora();
}
command = 0;          // reset command
// La linea siguiente no funciona bien.Tiene que pulsarse W0 para que muestre la hora.
// if (!(digitalRead(SW0))) set_time(); // mantenga pulsado el interruptor para ajustar la hora
  delay(1000);
  toggle(blinkPin); // parpadea led 1Hz.
}

void ahora() {
  Wire.begin();
  RTC.begin();

  // if (! RTC.isrunning()) {
  //   Serial.println("RTC is NOT running!"); }
  // La siguiente línea establece al RTC, la fecha y hora de su PC
  // en que este boceto fue compilado, descomentar para poner en hora.
  RTC.adjust(DateTime(__DATE__, __TIME__));

  Serial.println("RTC está Actualizado.");
  getDateDs1307();
}

void set_time() {
  byte minutes = 0;
  byte hours = 0;
  while (!digitalRead(SW0)) // pulsador de tiempo debe ser liberado para salir
  {
    Wire.beginTransaction(0x68); // activate DS1307
    Wire.send(0); // where to begin
    Wire.endTransmission();
  }
}
//*****The End*****/

```