Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

LABORATORIO DE MICROPROCESADORES
(86.07)

INFORME DE FINAL DEL PROYECTO

Proyecto:	**Detector de Apneas**	
Autores:	Araya, Camilo	96049
	Iglesias, Mariano	96247
Turno de T.P.	Martes 19-22 hs.	
Año y Cuatrimestre	2016	2do
Docente Guía:	Cofman, Fernando; Stola, Gerardo; Salaya, Guido	

Clase de Proyecto		Comentarios
1	04/10/16	-Parcial-
2	11/10/16	Pruebas acelerómetro con biblioteca Sparkfun/Arduino
3	18/10/16	"
4	25/10/16	Bajo consumo, rutina principal, interrupciones
5	01/11/16	Pruebas acelerómetro con Assembler
6	08/11/16	Pruebas I2C con Assembler
7	15/11/16	Pruebas RTC
8	22/11/16	Pruebas Bluetooth, armado final

Observaciones Generales

Calificación Final	
Firma del Docente	
Fecha	

Índice

1. Objetivo del Proyecto

El presente proyecto tiene como objetivo la detección de apneas en bebés recien nacidos o prematuros. El dispositivo es capaz de alertar a los padres cada vez que ocurre un cese en la respiración del bebé. Cabe mencionar que el mismo está dirigido a un uso hogareño, es decir en un cuadro no crítico del paciente, puesto que para ello existen formas de monitorearlo constantemente de modo más preciso dentro de una institución de salud.

En la actualidad, el acceso a dispositivos que detecten estos episodios resulta complejo en el país, dejando como única alternativa importarlos. Esto implica que para acceder a ellos es necesario afrontar grandes sumas de dinero.

El diseño del detector de apneas se focalizó en reducir los costos de los componentes de forma tal que el precio del dispositivo no sea un impedimento para el conjunto de la población.

La temprana detección de estos episodios permite que la persona que las padezca no sufra problemas cardíovasculares en el futuro y es de ayuda si es necesario aplicar maniobras RCP con anticipación.

2. Descripción del Proyecto

El detector de apneas consta de un acelerómetro como sensor principal cuya tarea es detectar el movimiento asociado a la respiración del usuario. El acelerómetro utilizado tiene la particularidad de detectar inactividad, de esta forma, ante cualquier cese de actividad durante una determinada cantidad de tiempo, en la zona toráxica de la persona, el dispositivo es el encargado de avisar al microcontrolador del episodio ocurrido. Se utilizó un parlante que transmite una alarma para despertar y de esta forma, el usuario, el cual no presentó movimientos asociados a la respiración, respire nuevamente. Además, esta alarma permite avisar del suceso a las personas a cargo del bebé, de modo que puedan actuar en consecuencia.

Una vez que se detecta uno de estos episodios, se almacena la fecha del evento. Para ello se utilizó un RTC o *Real Time Clock*, de esta forma, se tiene hora, día, mes y año del evento.

En cuanto a la comunicación de datos se utilizó un módulo bluetooth configurado como esclavo. De este modo, si se requiere información sobre el último evento ocurrido, los datos, almacenados en el microcontrolador, pueden ser envíados a una computadora.

El microcontrolador fue programado de forma de tal de hablitar sus interrupciones externas. Estas se corresponden con la inactividad detectada por el acelerómetro y con la transferencia de datos a traves del módulo *bluetooth*.

Una de las principales características que posee el detector de apneas es que el microcontrolador fue configurado para estar en bajo consumo, de esta forma, varias de las utilidades que posee este son inhabilitadas ya que no son utilizadas al momento de la detección de apneas. Dentro del conjunto de opciones que presenta el modo bajo consumo, se optó por aquella que pueda "despertar" al microcontrolador gracias a interrupciones externas, esta es *Power Down Mode*.

De acuerdo con las características mencionadas anteriormente, el microcontrolador se encuentra en modo bajo consumo todo el tiempo que se encuentra encendido. Sin embargo, esto no sucede con el acelerómetro ya que este se encuentra encendido, monitoreando actividad en la zona toráxica del usuario. Para ello se configuró el acelerómetro de forma tal que, una vez superado un umbral (asociado a la variación de aceleración) de inactividad en un lapso de tiempo, se genera una interrupción. El microcontrolador puede salir del modo bajo consumo por la interrupción generada por el acelerómetro, por inactividad, o porque es necesario el envío de datos a través del módulo bluetooth. Una vez que finalizan las actividades asociadas a las interrupciones, el microcontrolador se configura nuevamente en modo bajo consumo hasta que se genere una nueva interrupción externa. Esto sucede mientras el detector de apneas se encuentre encendido.

La comunicación entre los periféricos como el RTC y el acelerómetro se llevó a cabo gracias al protocolo de comunicación *I2C*, mientras que para la transmisión se utilizó el protocolo *UART*.

En la tabla 1 se presenta un breve resumen de la tarea que realiza cada periférico y el modelo utilizado.

Periférico	Descripción
Acelerómetro: ADXL345	Monitorea la falta de actividad asociada al cese de respiración
Bluetooth : HC-05	Módulo encargado de transmitir los datos
RTC: DS3231	Permite obtener la fecha del episodio

Cuadro 1: Periféricos que conforman el detector de apneas

Tensión	$3.3\,\mathrm{V}$
Consumo energético	Del orden de los mW
Protocolos de comunicación	*I2C* y UART
Rango de mediición del sensor	Ancho de banda $\leq 10\,\mathrm{Hz}$

Cuadro 2: Especificaciones del detector de apneas

En la tabla 2 se observan las características y especificaciones mínimas que presenta el detector de apneas. Cabe aclarar que como el proceso a medir, la respiración, tiene una frecuencia propia del orden del Hertz, se optó por acotar el valor de la frecuencia de muestreo del acelerómetro, lo cual trae ventajas respecto del consumo.

3. Diagrama en Bloques (hardware)

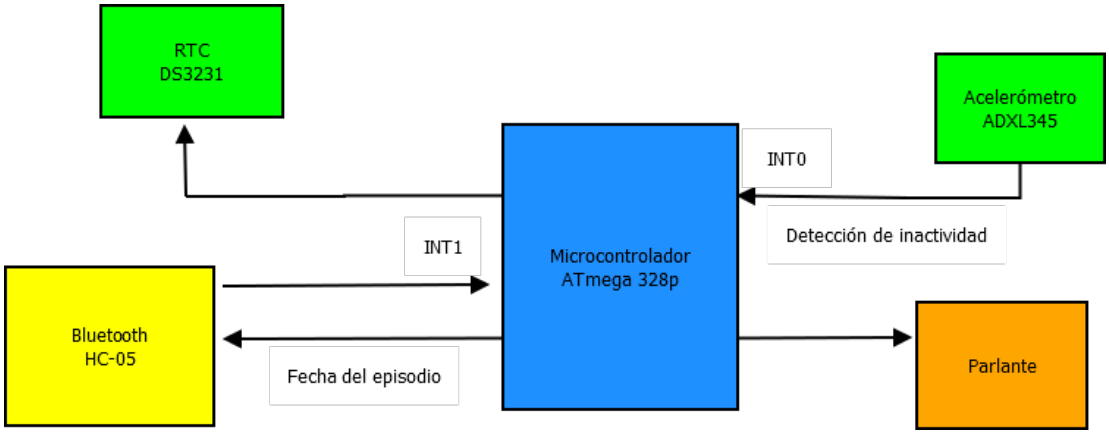


Figura 1: Diagrama de bloques

En la figura 1 se observa el diagrama de bloques que representa el detector de apnea. De acuerdo con la figura, cuando se activa la interrupción **INT0**, el microcontrolador sale del modo bajo consumo porque el acelerómetro no detectó movimiento. Una vez que esto sucede, se almacena la fecha del episodio. Para ello es necesario obtener la fecha que se encuentra dentro del RTC.

La interrupción externa **INT1** está asociada a la transmisión de datos. Cuando esta ocurre, es porque se requiere conocer la fecha del episodio de apnea. Para ello, la fecha almacenada en los registros del microcontrolador, se envía a través del módulo bluetooth.

4. Diagrama de Flujo (firmware)

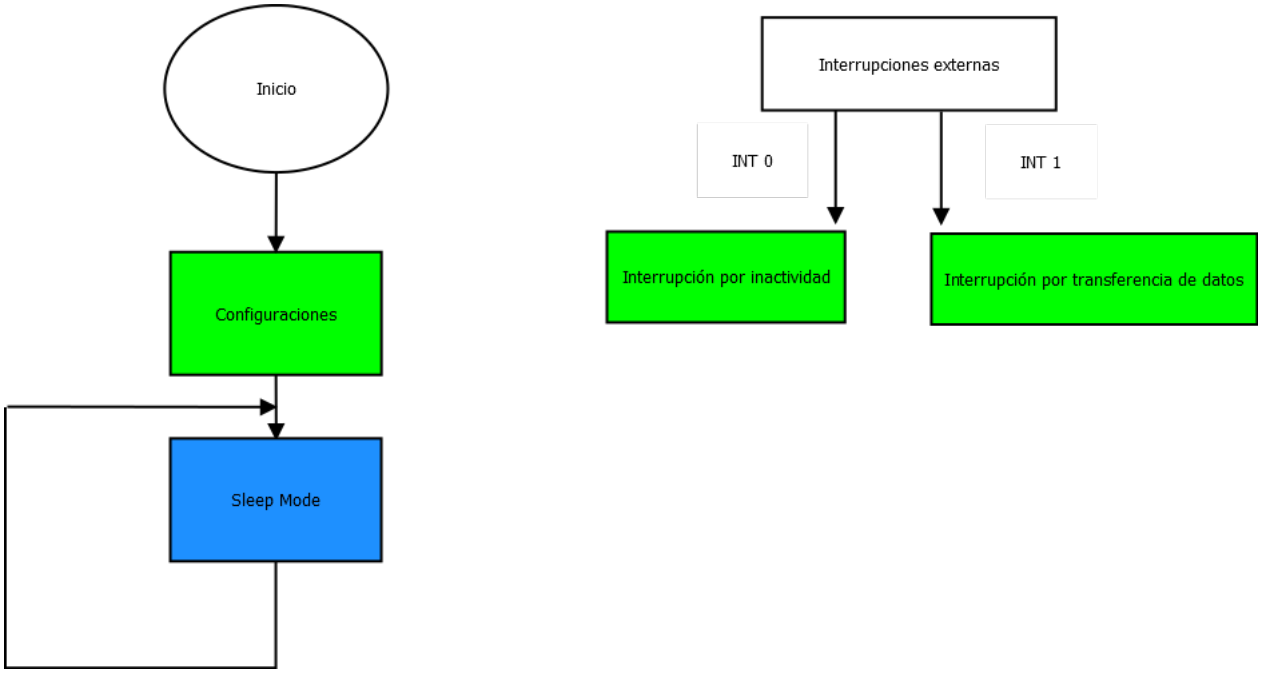


Figura 2: Diagrama de flujos asociado al bloque principal del programa del microcontrolador

En la figura 2, se observa el diagrama de flujo del programa codificado en el microcontrolador. Tal como se puede observar, una vez que se configura, el microcontrolador entra en el modo de bajo consumo. La única forma de que este salga del modo es que se genere una interrupción externa, que puede ser por inactividad (generada por el acelerómetro) o por transferencia de datos (generada por un usuario que necesite información del episodio). Una vez finalizadas las instrucciones asociadas a cada interrupción el microcontrolador nuevamente se encuentra en modo bajo consumo. Este proceso se mantiene hasta que se apague el detector de apneas.

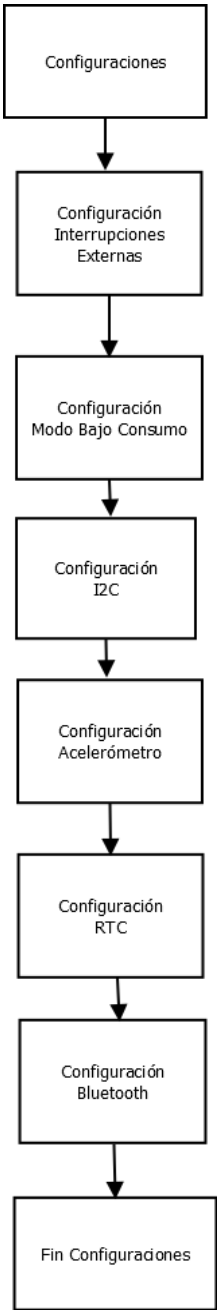


Figura 3: Diagrama de flujos asociado a la configuración del microcontrolador y sus periféricos

En la figura 3 se puede observar el orden de las configuraciones necesarias para poder ejecutar el bloque principal del programa. Es necesario que en primer lugar se habiliten las interupciones externas, al igual que habilitar el bit de interrupción global, para luego habilitar el modo bajo consumo, de forma tal que al ejecutar la instrucción **SLEEP**, el microcontrolador entre en bajo consumo. Luego es necesario configurar el protocolo de comunicación I2C para poder configurar los periféricos como el acelerómetro y el RTC.

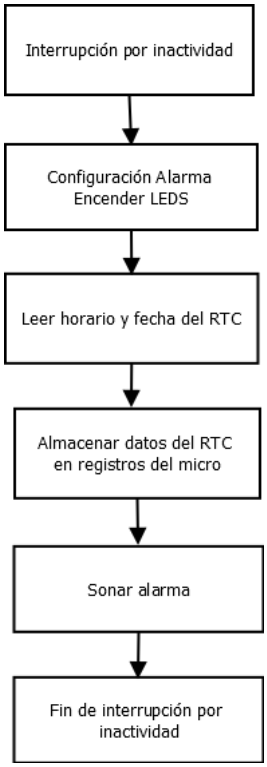


Figura 4: Diagrama de flujos asociado a la interrupción por inactividad

De acuerdo con la figura 4, la interrupción por inactividad, generada por el acelerómetro, debe recurrir al *timestamp* que se encuentra dentro del RTC. Luego, una vez leídos los datos del reloj, se deben almacenar en los registros del microcontrolador. Finalmente, es necesario que a partir de la detección del cese de la respiración, la alarma suene para alertar a los padres del bebé, para que este deje de tener la apnea y vuelva a respirar.

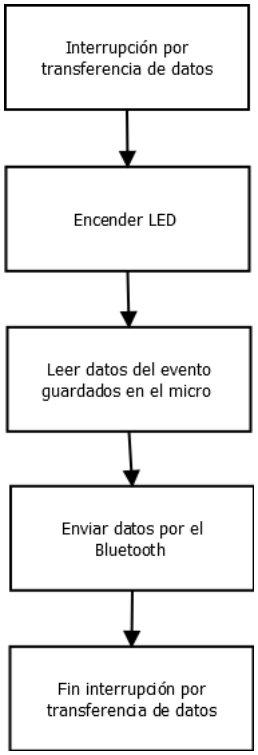


Figura 5: Diagrama de flujos asociado a la transferencia de datos

5. Circuito Esquemático

En la figura 6 se presenta el circuito esquemático del detector de apneas. El mismo tiene como componente principal al microcontrolador *Atmega328P* de *Atmel*, el cual se encuentra sobre la placa de desarrollo *Arduino Nano*. A dicha placa se conectan los periféricos: acelerómetro, RTC y bluetooth, así como también los indicadores de inactividad (LED verde y buzzer), el indicador de transferencia de datos por bluetooth (LED azul) y el pulsador que permite comenzar la transferencia de datos. También se agregó un transistor NPN TIP41C para poder controlar el encendido y apagado del módulo de bluetooth, para reducir el consumo de energía cuando no es necesario. Esto se logra conectando la base del transistor a un pin del microcontrolador.

La razón a la que obedece el uso de la plataforma *Arduino Nano* es que dicha placa posee un tamaño reducido comparable al del microcontrolador en versión DIP, pero con la ventaja de traer embebidos a los reguladores de 3.3 V y 5 V para una tensión de entrada de entre 9 V y 12 V.

En cuanto a componentes adicionales, se debe tener en cuenta a las resistencias de *pull-up* para la conexión I2C, las mismas se recomiendan con un valor entre 1 kΩ y 100 kΩ dependiendo de la capacidad y características del canal, se decidió emplear resistencias 4.7 kΩ. También se agregó un divisor resistivo de forma que los pines de la UART, que tienen valores entre 5 V y 0 V a la salida del microcontrolador, tengan 3.3 V al llegar al módulo bluetooth. Por otra parte, se colocaron resistencias para limitar la corriente por los LEDs indicadores de inactividad y de transmisión de datos bluetooth. Por último, el pulsador conectado a la interrupción causante del inicio de la transmisión bluetooth tiene un circuito RC antirebotes.

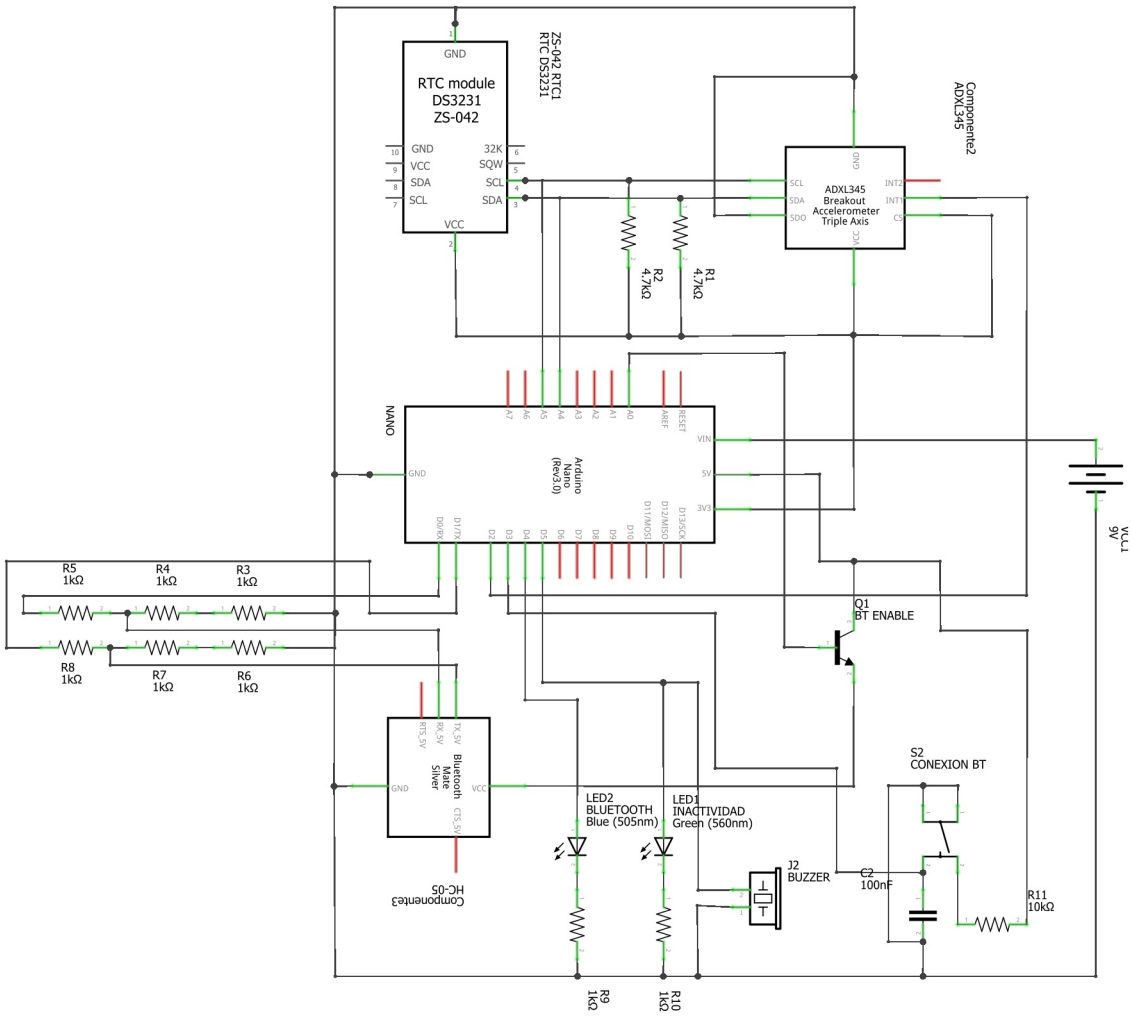


Figura 6: Circuito esquemático del detector de apneas

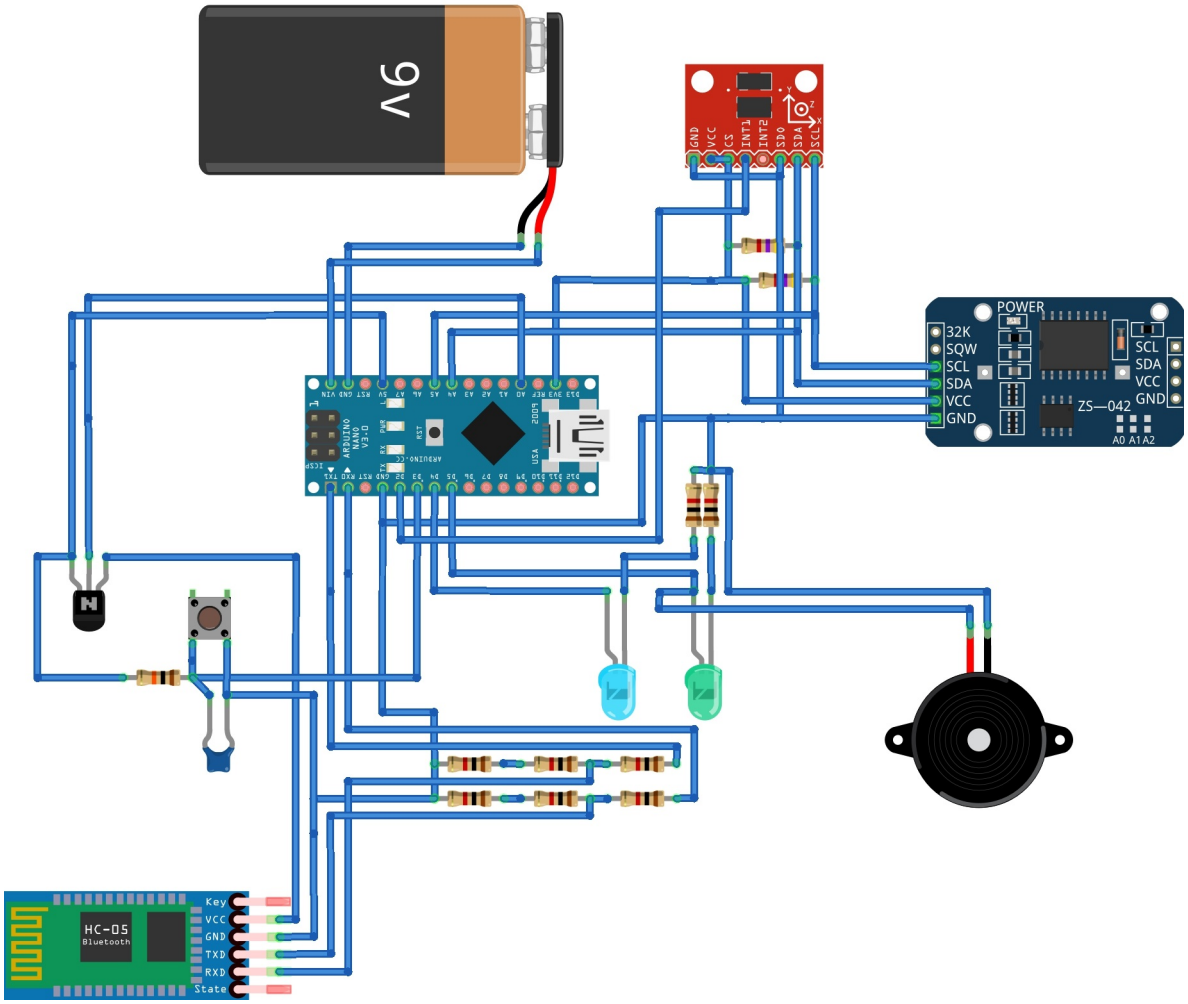


Figura 7: Circuito correspondiente al detector de apneas

6. Listado de Componentes y Costos

Módulo	Precio (AR$)
Bluetooth	90
Acelerómetro	370
RTC	90
Arduino Nano	98
Batería	40
Capacitores, resistencias, etc	10

Cuadro 3: Listado de precios de los módulos utilizados

En la tabla 3 se observan los precios de los módulos utilizados. De acuerdo con los precios mencionados para el proyecto se gastó 698$.

Se estiman 8 Hs. hombre por semana, dando un total de 48 Hs.

7. Resultados

Luego de realizado el proyecto, se efectúa una valoración acerca de los logros realizados en contraste con lo propuesto en el antiproyecto.

En primer lugar, se dejaron de lado los puntos optativos del anteproyecto. El motor vibrador debido a su gran consumo de corriente en un equipo de consumo reducido, y el oxímetro ya que la dificultad del diseño analógico para el sensor comprometía la realización en un cuatrimestre. También merece mención el hecho de que la memoria de los *timestamps* no se realiza en memoria externa, ya que por recomendación de los profesores se evitó el uso de almacenamiento en tarjeta SD.

Fuera de estos puntos, se logró cumplir lo estipulado en el anteproyecto. En la tabla 4 se indican las especificaciones del dispositivo construido.

Tensión	3.3 V/5 V/9 V
Consumo energético con modo sleep (en estado ocioso)	126 mW
Consumo energético sin modo sleep (en estado ocioso)	225 mW
Consumo energético con modo sleep y bluetooth encendido	594 mW
Protocolos de comunicación	*I2C* y UART
Rango de medición del sensor	Ancho de banda = 3.13 Hz
Clock	16 MHz

Cuadro 4: Especificaciones del detector de apneas construido

En cuanto a la autonomía del proyecto, considerando que se lo alimenta con una batería de 9 V, la cual posee un valor de carga de entre 550 mA h y 1200 mA h, y si se usa en promedio 6 horas por noche, se tendría una duración de batería de entre 7 y 14 días, considerando el tipo de batería empleada, respectivamente.

7.1. Problemas presentados durante la realización del proyecto

Por una parte, el protocolo de comunicación I2C, presentó varias dificultades, lo cual generó retrasos en el desarrollo del proyecto. Esta dificultad imposibilitó la configuración de módulos que necesitan de este protocolo, tales como el RTC y el acelerómetro. Para solucionarlo, se buscó información en la hoja de datos del microcontrolador, en la bibliografía y en los apuntes de las clases teóricas, cuyos códigos coincidían entre sí y con el programado en el microcontrolador del presente proyecto. Ante esta problemática, se evaluó la posibilidad de generar un código con partes en lenguaje C y otras en ASM, consultando en las hojas de aplicación de Atmel. Sin embargo, habiendo solucionado y configurado el I2C, las interrupciones externas programadas, dejaron de funcionar, por lo que esta solución fue totalmente descartada cuando se solucionó el problema original en ASM.

El almacenamiento de los datos leídos del RTC en memoria EEPROM presentó varias dificultades las cuales derivaron en optar por no guardar la información temporal del episodio en esta memoria. Como solución ante este problema, se optó por almacenar esta información en los registros del microcontrolador ya que, a pesar de ser memoria volátil, como el detector nunca se apaga y siempre se encuentra en modo bajo consumo, nunca se perdería esta información.

Con respecto al protocolo UART, en los primeros intentos, al intentar transimitir los bytes deseados de forma conjunta, se tenían problemas en la transmisión, es por ello que se verificó el correcto funcionamiento de las rutinas en pequeños bloques de códigos. Una vez que se modificó la frecuencia de clock, se solucionó el problema.

8. Conclusiones

En síntesis, se ha realizado un proyecto que integra temas de la materia para una aplicación particular. El proyecto permitió incentivar la búsqueda de información acerca de los dispositivos

específicos empleados y sobre cómo organizar su funcionamiento en conjunto.

Por otra parte, al haber realizado el proyecto en el lenguaje Assembler, se logró obtener otra visión sobre la programación y el funcionamiento de los sistemas a bajo nivel, permitiendo tener un control más detallado de parámetros como el tiempo de ejecución de rutinas.

En cuanto a las sugerencias finales, se propone para etapas posteriores una realización en base a componentes SMD a modo de disminuir el tamaño del detector de apneas y eliminar partes innecesarias en las placas de los periféricos. También, se debería agregar una memoria externa para almacenar una mayor cantidad de datos. En cuanto a la finalidad del producto, se deberían realizar calibraciones y validaciones con profesionales médicos.

Por último, cabe destacar que para la realización del trabajo se debieron recorrer todas las etapas de un proyecto profesional, lo cual resulta en una ganancia de experiencia frente a una situación problemática de desarrollo.

A. Apéndice

A.1. Código en lenguaje Assembler del proyecto

A continuación se presenta el código utilizado en todo el proyecto

Listing 1: Código proyecto

```
/************************************************************
DIRECTIVAS
************************************************************/
#define F_CPU 16000000UL
.include "m328Pdef.inc"

.def AUX1 = R21
.def TEMP = R17
.def UART_DATA = R18
.def SWITCH = R19
.def SLA_W = R20
.def SLA_R = R16
.def REG_DIR = R22
.def DATA_OUT = R23
.def DATA_IN = R24
.def AUX_I2C = R25

.def SEGUNDOS = R2
.def MINUTOS = R3
.def HORA = R4
.def DIA_SEMANA = R5
.def DIA_MES = R6
.def MES = R7
.def ANIO = R8

.equ CASE_NULL = 0
.equ CASE_INACT = 1
.equ CASE_BT = 2

.equ ORG_MAIN = 0x00
.equ ORG_ISR_INT0_INACTIVITY = 0x02
.equ ORG_ISR_INT1_BLUETOOTH = 0x04

.equ NULL = 0b00000000
.equ BT_ENABLE = 0b00000001
.equ INACT_BT_LEDS = 0b00110000
```

```
.equ TODO_CAMBIO_LOGICO_INT0_INT1 = 0b00000101
.equ INT0_INT1_ENABLE = 0b00000011
.equ POWER_DOWN_MODE_SLEEP_OFF = 0b00000100 ; D3D2D1 = 010 es de
; power down mode y el lsb D0 = 1 es para activar el sleep mode
.equ POWER_DOWN_MODE_SLEEP_ON = 0b00000101
.equ BT_LED_TIME = 10; Duracion LED titilando
.equ INACT_LED_BUZZER_TIME = 200; Duracion LED y buzzer titilando
.equ BT_LED = 0b00010000
.equ INACT_LED = 0b00100000

.equ ACELEROMETRO_SLA_W = 0xA6
; Direccion del esclavo SLA(1010011) + Write(0)
.equ ACELEROMETRO_SLA_R = 0xA7
; Direccion del esclavo SLA(1010011) + Read(1)
.equ THRESH_INACT = 0x25; Umbral de inactividad
.equ THRESH_INACT_VAL = 0b00000001
; 8 bit unsigned. No dejar en 0x00 umbral de inactividad (62.5 mg/LSB)
.equ TIME_INACT = 0x26; Umbral de tiempo de inactividad
.equ TIME_INACT_VAL = 0b00000100; (maximo 255 segundos) (1 sec/LSB)
.equ ACT_INACT_CTL = 0x27; Controla los ejes que intervienen
.equ ACT_INACT_CTL_VAL = 0b00001111
; (D3 en 1=ac coupled)(D2D1D0 = 111 enable en los tres ejes)
.equ BW_RATE = 0x2C; Data rate and power mode control
.equ BW_RATE_VAL = 0b00000110
;(D4=0 sin Low Power)(Rate D3D2D1D0 = 0110 : BW=3.13Hz)
.equ INT_MAP = 0x2F; Interrupt mapping control
.equ INT_MAP_VAL = 0b11110111
; Solo la inactividad conectada al pin INT1, el resto al pin INT2
.equ INT_ENABLE = 0x2E; Interrupt enable control
.equ INT_ENABLE_VAL = 0b00001000
; Se habilita solo la interrupcion por inactividad
.equ POWER_CTL = 0x2D;
.equ POWER_CTL_VAL = 0b00001000
; MEASURE ON, por defecto se prende en modo standby
.equ INT_SOURCE = 0x30; —Muestra la causa de interrupcion, read only

.equ RTC_SLA_W = 0b11010000; Direccion del esclavo + Write(0)
.equ RTC_SLA_R = 0b11010001; Direccion del esclavo + Read(1)
.equ RTC_CTRL_REG = 0x0E; CONFIGURACION REGISTRO DE CONTROL
.equ RTC_CTRL_REG_VAL = 0x00
.equ RTC_SEGUNDOS_REG = 0b00000000; CONFIGURACION SEGUNDOS
.equ RTC_SEGUNDOS_REG_VAL = 0b00000000
.equ RTC_MINUTOS_REG = 0b00000001; CONFIGURACION MINUTOS
.equ RTC_MINUTOS_REG_VAL = 0b00000000
.equ RTC_HORA_REG = 0b00000010; CONFIGURACION HORA
.equ RTC_HORA_REG_VAL = 0b00011001
.equ RTC_DIA_SEMANA_REG = 0b00000011; CONFIGURACION DIA DE LA SEMANA
.equ RTC_DIA_SEMANA_REG_VAL = 0b00000101 ;dia 5 de la semana, viernes
.equ RTC_DIA_MES_REG = 0b00000100; CONFIGURACION DIA DEL MES
.equ RTC_DIA_MES_REG_VAL = 0b00011000
.equ RTC_MES_REG = 0b00000101; CONFIGURACION MES
.equ RTC_MES_REG_VAL = 0b00010001
.equ RTC_ANIO_REG = 0b00000110; CONFIGURACION ANIO
.equ RTC_ANIO_REG_VAL = 0b00010110

.equ UBRR0L_VALUE = 0x67
.equ UBRR0H_VALUE = 0x00

```

```
95
96  /*****************************************************************
97  VECTORES DE INTERRUPCION
98  *****************************************************************/
99
100 .cseg
101 .org ORG_MAIN
102 rjmp main
103 .org ORG_ISR_INT0_INACTIVITY
104 rjmp ISR_INT0_INACTIVITY
105 .org ORG_ISR_INT1_BLUETOOTH
106 rjmp ISR_INT1_BLUETOOTH
107
108
109 /*****************************************************************
110 MAIN
111 *****************************************************************/
112
113 main:
114
115
116 /*****************************************************************
117 CONFIGURACION STACK
118 *****************************************************************/
119
120     LDI TEMP, HIGH(RAMEND)    ; Configura el STACK
121     OUT SPH, TEMP
122     LDI TEMP, LOW(RAMEND)
123     OUT SPL, TEMP
124
125
126 /*****************************************************************
127 CONFIGURACION PUERTOS
128 *****************************************************************/
129
130     LDI TEMP, INACT_BT_LEDS
131     ; LEDs conectados a los bits seteados en dichos puertos
132     OUT DDRD, TEMP
133     LDI TEMP, NULL      ; LEDs apagados
134     OUT PORTD, TEMP
135
136
137 /*****************************************************************
138 CONFIGURACION INTERRUPCIONES EXTERNAS
139 *****************************************************************/
140
141 CONFIG_INT:
142
143     ;EICRA: Configura por flanco o por nivel
144     ldi TEMP, TODO_CAMBIO_LOGICO_INT0_INT1
145     ; Cualquier cambio logico en INT0 e INT1
146     sts EICRA, TEMP          ;
147
148     ;EIMSK: Habilita las interrupciones seleccionadas
149     ldi TEMP, INT0_INT1_ENABLE; Habilita INT0 e INT1
150     out EIMSK, TEMP          ;
151
152     ;EIFR: Tiene que estar seteado junto con el bit de
```

```
153       ;interrupcion global al momento de suceder la interrupcion
154       ldi TEMP, INT0_INT1_ENABLE    ; Para INT0 e INT1
155       out EIFR, TEMP      ;
156
157       SEI
158
159
160  /************************************************************
161  CONFIGURACION BAJO CONSUMO
162  ************************************************************/
163
164  CONFIG_BAJO_CONSUMO:
165       ; Config consumo de energia, PSM o PDM
166       ldi TEMP, POWER_DOWN_MODE_SLEEP_OFF
167       out SMCR, TEMP
168
169
170  /************************************************************
171  INCIALIZACION I2C
172  ************************************************************/
173
174  RCALL I2C_INIT
175
176
177  /************************************************************
178  CONFIGURACION ACELEROMETRO
179  ************************************************************/
180
181  RCALL CONFIG_ACELEROMETRO
182  RCALL CLEAN_INACT
183
184
185  /************************************************************
186  CONFIGURACION RTC
187  ************************************************************/
188
189  RCALL CONFIG_RTC
190
191
192  /************************************************************
193  CONFIGURACION BLUETOOTH
194  ************************************************************/
195
196  RCALL USART_INIT
197
198
199  /************************************************************
200  BLOQUE PRINCIPAL
201  ************************************************************/
202
203  SLEEP_MODE:
204            ldi TEMP, POWER_DOWN_MODE_SLEEP_ON
205            ; Activa el sleep mode
206            out SMCR, TEMP
207            SLEEP
208            ; Entra en modo sleep, al suceder una interrupcion
209            ; se despierta en la instruccion siguiente
210            ldi TEMP, POWER_DOWN_MODE_SLEEP_OFF
```

```
211             ; Desactiva el sleep mode
212             out SMCR, TEMP
213
214 ; En las rutinas de interrupcion se usa SWITCH para almacenar un
215 ;registro indicador del tipo de interrupcion ocurrida
216 ; Es una analogia del switch/case del lenguaje C
217
218             CPI SWITCH, CASE_INACT
219             BREQ ALARMA
220 RETORNO_ALARMA:
221             CPI SWITCH, CASE_BT
222             BREQ BLUETOOTH
223 RETORNO_BLUETOOTH:
224             RJMP SLEEP_MODE
225
226
227 /************************************************************
228 RUTINA ENVIO DE ENVIO DE DATOS POR BLUETOOTH
229 ************************************************************/
230
231 BLUETOOTH:
232 LDI AUX1, BT_LED_TIME
233
234       LDI TEMP, BT_ENABLE       ; Habilita Vcc a BT
235       OUT DDRC, TEMP
236       LDI TEMP, BT_ENABLE
237       OUT PORTC, TEMP
238
239 PARPADEO2:
240             LDI TEMP, BT_LED; Prende LED
241             OUT PORTD, TEMP
242             RCALL RETARDO2
243             LDI TEMP, NULL ; Apaga LED
244             OUT PORTD, TEMP
245             RCALL RETARDO2
246
247             DEC AUX1
248             CPI AUX1, NULL
249             BRNE PARPADEO2
250
251 ; Envia los datos
252             MOV UART_DATA, SEGUNDOS;
253             RCALL USART_TRANSMISION  ;
254
255             MOV UART_DATA, MINUTOS;
256             RCALL USART_TRANSMISION  ;
257
258             MOV UART_DATA, HORA;
259             RCALL USART_TRANSMISION  ;
260
261             MOV UART_DATA, DIA_SEMANA;
262             RCALL USART_TRANSMISION  ;
263
264             MOV UART_DATA, DIA_MES;
265             RCALL USART_TRANSMISION  ;
266
267             MOV UART_DATA, MES;
268             RCALL USART_TRANSMISION  ;
```

```
269
270             MOV UART_DATA, ANIO;
271             RCALL USART_TRANSMISION  ;
272
273
274             RCALL CLEAN_INACT
275             RCALL CONFIG_ACELEROMETRO
276
277             LDI SWITCH, CASE_NULL    ; Se limpia el registro indicador
278             RJMP RETORNO_BLUETOOTH
279
280
281     RETARDO2:
282             LDI R16, 10
283     LOOP12:  LDI R17, 255
284     LOOP22:  LDI R18, 255
285     LOOP32:  DEC R18
286             BRNE LOOP32
287             DEC R17
288             BRNE LOOP22
289             DEC R16
290             BRNE LOOP12
291             RET
292
293     RETARDO3:
294             LDI R16, 5
295     LOOP13:  LDI R17, 255
296     LOOP23:  LDI R18, 255
297     LOOP33: DEC R18
298             BRNE LOOP33
299             DEC R17
300             BRNE LOOP23
301             DEC R16
302             BRNE LOOP13
303             RET
304
305     /*************************************************************
306     RUTINA DE ALARMA
307     *************************************************************/
308
309     ALARMA:
310             LDI AUX1, INACT_LED_BUZZER_TIME
311
312             RCALL PARPADEO
313             RCALL RETARDO3
314             RCALL PARPADEO
315             RCALL RETARDO3
316             RCALL PARPADEO
317             RCALL RETARDO3
318
319             LDI SLA_W, RTC_SLA_W
320             LDI SLA_R, RTC_SLA_R
321
322             LDI REG_DIR, RTC_SEGUNDOS_REG
323             RCALL SINGLE_BYTE_READ
324             MOV SEGUNDOS,DATA_IN
325
326             LDI REG_DIR, RTC_MINUTOS_REG
```

```
327             RCALL SINGLE_BYTE_READ
328             MOV MINUTOS,DATA_IN
329
330             LDI REG_DIR, RTC_HORA_REG
331             RCALL SINGLE_BYTE_READ
332             MOV HORA,DATA_IN
333
334             LDI REG_DIR, RTC_DIA_SEMANA_REG
335             RCALL SINGLE_BYTE_READ
336             MOV DIA_SEMANA,DATA_IN
337
338             LDI REG_DIR, RTC_DIA_MES_REG
339             RCALL SINGLE_BYTE_READ
340             MOV DIA_MES,DATA_IN
341
342             LDI REG_DIR,RTC_MES_REG
343             RCALL SINGLE_BYTE_READ
344             MOV MES,DATA_IN
345
346             LDI REG_DIR, RTC_ANIO_REG
347             RCALL SINGLE_BYTE_READ
348             MOV ANIO,DATA_IN
349
350
351             RCALL CLEAN_INACT
352             RCALL CONFIG_ACELEROMETRO
353
354             LDI SWITCH, CASE_NULL; Se limpia el registro indicador
355             RJMP RETORNO_ALARMA
356
357
358 RETARDO:
359             LDI R16, 1
360 LOOP1:      LDI R17, 50
361 LOOP2:      LDI R18, 50
362 LOOP3:    DEC R18
363             BRNE LOOP3
364             DEC R17
365             BRNE LOOP2
366             DEC R16
367             BRNE LOOP1
368             RET
369
370 PARPADEO:
371       LDI TEMP, INACT_LED ; Prende LED
372       OUT PORTD, TEMP
373       RCALL RETARDO
374       LDI TEMP, NULL ; Apaga LED
375       OUT PORTD, TEMP
376       RCALL RETARDO
377
378       DEC AUX1
379       CPI AUX1, NULL
380       BRNE PARPADEO
381       RET
382
383
384 /*************************************************************
```

```
385 RUTINA DE SERVICIO DE INTERRUPCION POR INACTIVIDAD DEL ACELEROMETRO
386 ************************************************************/
387
388
389 ISR_INT0_INACTIVITY:
390 CLI
391 LDI SWITCH, CASE_INACT
392 SEI
393 RETI
394
395
396
397 /************************************************************
398 RUTINA DE SERVICIO DE INTERRUPCION POR ACTIVACION DEL BLUETOOTH
399 ************************************************************/
400
401
402 ISR_INT1_BLUETOOTH:
403 CLI
404 LDI SWITCH, CASE_BT
405 SEI
406 RETI
407
408
409
410
411 /************************************************************
412 CONFIGURACION ACELEROMETRO
413 ************************************************************/
414
415 CONFIG_ACELEROMETRO:
416
417     LDI SLA_W, ACELEROMETRO_SLA_W
418     LDI SLA_R, ACELEROMETRO_SLA_R
419
420 ; Se comienza la secuencia de seteo del
421 ; acelerometro en los valores necesarios
422
423     LDI REG_DIR, THRESH_INACT
424     LDI DATA_OUT, THRESH_INACT_VAL
425     RCALL MULTIPLE_BYTE_WRITE
426
427     LDI REG_DIR, TIME_INACT
428     LDI DATA_OUT, TIME_INACT_VAL
429     RCALL MULTIPLE_BYTE_WRITE
430
431     LDI REG_DIR, ACT_INACT_CTL
432     LDI DATA_OUT, ACT_INACT_CTL_VAL
433     RCALL MULTIPLE_BYTE_WRITE
434
435     LDI REG_DIR, BW_RATE
436     LDI DATA_OUT, BW_RATE_VAL
437     RCALL MULTIPLE_BYTE_WRITE
438
439     LDI REG_DIR, INT_MAP
440     LDI DATA_OUT, INT_MAP_VAL
441     RCALL MULTIPLE_BYTE_WRITE
442
```

```
443       LDI REG_DIR, INT_ENABLE
444       LDI DATA_OUT, INT_ENABLE_VAL
445       RCALL MULTIPLE_BYTE_WRITE
446
447       LDI REG_DIR, POWER_CTL
448       LDI DATA_OUT, POWER_CTL_VAL
449       RCALL MULTIPLE_BYTE_WRITE
450
451 RET
452
453
454
455 CLEAN_INACT: ; Limpia el flag de inactivity en el acelerometro
456
457       LDI SLA_W, ACELEROMETRO_SLA_W
458       LDI SLA_R, ACELEROMETRO_SLA_R
459       LDI REG_DIR, INT_SOURCE
460       RCALL SINGLE_BYTE_READ
461 RET
462
463
464 /*************************************************************
465 CONFIGURACION RTC
466 *************************************************************/
467
468 CONFIG_RTC:
469
470       LDI SLA_W, RTC_SLA_W
471       LDI SLA_R, RTC_SLA_R
472
473       LDI REG_DIR, RTC_CTRL_REG
474       LDI DATA_OUT, RTC_CTRL_REG_VAL
475       RCALL MULTIPLE_BYTE_WRITE
476
477       LDI REG_DIR, RTC_SEGUNDOS_REG
478       LDI DATA_OUT, RTC_SEGUNDOS_REG_VAL
479       RCALL MULTIPLE_BYTE_WRITE
480
481       LDI REG_DIR, RTC_MINUTOS_REG
482       LDI DATA_OUT, RTC_MINUTOS_REG_VAL
483       RCALL MULTIPLE_BYTE_WRITE
484
485       LDI REG_DIR, RTC_HORA_REG
486       LDI DATA_OUT, RTC_HORA_REG_VAL
487       RCALL MULTIPLE_BYTE_WRITE
488
489       LDI REG_DIR, RTC_DIA_SEMANA_REG
490       LDI DATA_OUT, RTC_DIA_SEMANA_REG_VAL
491       RCALL MULTIPLE_BYTE_WRITE
492
493       LDI REG_DIR, RTC_DIA_MES_REG
494       LDI DATA_OUT, RTC_DIA_MES_REG_VAL
495       RCALL MULTIPLE_BYTE_WRITE
496
497       LDI REG_DIR, RTC_MES_REG
498       LDI DATA_OUT, RTC_MES_REG_VAL
499       RCALL MULTIPLE_BYTE_WRITE
500
```

```
501       LDI REG_DIR, RTC_ANIO_REG
502       LDI DATA_OUT, RTC_ANIO_REG_VAL
503       RCALL MULTIPLE_BYTE_WRITE
504
505   RET
506
507
508   /************************************************************
509   CONFIGURACION I2C
510   ************************************************************/
511
512   I2C_INIT:
513       LDI TEMP, NULL
514       STS TWSR, TEMP       ; Preescaler 1 en TWI Status Reg
515       LDI TEMP, 0xB0       ; 0xB0
516       STS TWBR, TEMP       ; Setea la frecuencia a 43.48 kHz(16 MHz XTAL)
517       LDI TEMP, (1<<TWEN)  ; 0x04 a TEMP (TWEN: Enable bit)
518       STS TWCR, TEMP       ; Habilita el TWI
519       RET
520
521
522   I2C_START:
523       LDI TEMP, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
524       STS TWCR, TEMP       ; Transimitir condicion de START
525
526   WAIT1:
527       LDS TEMP, TWCR       ; Lee el registro
528       SBRS TEMP, TWINT
529       ; Saltea siguiente linea si TWINT es 1 (==operacion finalizada)
530       RJMP WAIT1                ; TWINT esta en 0
531       RET
532
533
534   I2C_WRITE:
535       STS TWDR, AUX_I2C         ; Lleva el byte a TWDR
536       LDI TEMP, (1<<TWINT)|(1<<TWEN) ; Se setean TWINT y TWEN en el TWCR
537       STS TWCR, TEMP       ; Configura TWCR para enviar TWDR
538
539   WAIT2:
540       LDS TEMP, TWCR       ; Lee el registro de control a TEMP
541       SBRS TEMP, TWINT     ; Saltea siguiente linea si TWINT es 1
542       RJMP WAIT2                ; TWINT esta en 0
543       RET
544
545
546   I2C_READ:
547       LDI TEMP, (1<<TWINT)|(1<<TWEN)
548       STS TWCR, TEMP
549
550   WAIT3:
551       LDS TEMP, TWCR
552       SBRS TEMP, TWINT
553       RJMP WAIT3
554       LDS AUX_I2C, TWDR         ; Guarda en AUX_I2C el dato leido
555       RET
556
557
558   I2C_STOP:
```

```
559       LDI TEMP, (1<<TWINT)|(1<<TWSTO)|(1<<TWEN)
560       STS TWCR, TEMP      ; Transmitir condicion de STOP
561       RET
562
563
564  ; Subrutina de escritura de registros de perifericos
565  MULTIPLE_BYTE_WRITE:
566       RCALL I2C_START          ; Transmite la condicion de START
567       MOV AUX_I2C, SLA_W
568       ; Carga la direccion del esclavo + configuracion W
569       RCALL I2C_WRITE          ; Escribe AUX_I2C al bus I2C
570       MOV AUX_I2C, REG_DIR     ; Direccion del registro a escribir
571       RCALL I2C_WRITE          ; Escribe AUX_I2C al bus I2C
572       MOV AUX_I2C, DATA_OUT    ; Dato a transmitir
573       RCALL I2C_WRITE          ; Escribe AUX_I2C al bus I2C
574       RCALL I2C_STOP           ; Transmite la condicion de STOP
575       RET
576
577
578  ; Subrutina de lectura de registros de perifericos
579  SINGLE_BYTE_READ:
580       RCALL I2C_START          ; Transmite la condicion de START
581       MOV AUX_I2C, SLA_W
582       ; Carga la direccion del esclavo + configuracion W
583       RCALL I2C_WRITE          ; Escribe AUX_I2C al bus I2C
584       MOV AUX_I2C, REG_DIR; Direccion del registro a leer
585       RCALL I2C_WRITE          ; Escribe AUX_I2C al bus I2C
586       RCALL I2C_START          ; Retransmite Start
587       MOV AUX_I2C, SLA_R
588       ; Carga la direccion del esclavo + configuracion R
589       RCALL I2C_WRITE
590       LDI AUX_I2C, NULL
591       RCALL I2C_READ
592       MOV DATA_IN, AUX_I2C
593       RCALL I2C_STOP
594       RET
595
596
597  /************************************************************
598  CONFIGURACION BLUETOOTH
599  ************************************************************/
600
601  USART_INIT:
602       ;Baud rate en 9600
603       LDI TEMP, UBRR0L_VALUE
604       STS UBRR0L, TEMP
605       LDI TEMP, UBRR0H_VALUE
606       STS UBRR0H, TEMP
607
608       LDI TEMP, (1<<TXEN0); Habilitacion TX
609       STS UCSR0B, TEMP
610
611       LDI TEMP, (1<<USBS0)|(3<<UCSZ00); 8 bits por dato, 2 bits de stop
612       STS UCSR0C, TEMP
613  RET
614
615
616  USART_TRANSMISION:
```

```
617         LDS   TEMP,UCSR0A
618         ; Chequea que el buffer este listo para la transmision
619         SBRS  TEMP,UDRE0
620         RJMP  USART_TRANSMISION
621         STS   UDR0, UART_DATA
622         ; Envia la informacion que hay en el
623         ;registro UART_DATA al buffer para ser enviada
624 RET
```

A.2. Hojas de datos de los componentes utilizados

A continuación se presentan las hojas de datos de los componentes utilizados para el diseño del dispositivo:

3-Axis, ±2 *g*/±4 *g*/±8 *g*/±16 *g* Digital Accelerometer

ADXL345

FEATURES

Ultralow power: as low as 40 µA in measurement mode and 0.1 µA in standby mode at V_S = 2.5 V (typical)
Power consumption scales automatically with bandwidth
User-selectable resolution
Fixed 10-bit resolution
Full resolution, where resolution increases with *g* range, up to 13-bit resolution at ±16 *g* (maintaining 4 mg/LSB scale factor in all *g* ranges)
Embedded, patent pending FIFO technology minimizes host processor load
Tap/double tap detection
Activity/inactivity monitoring
Free-fall detection
Supply voltage range: 2.0 V to 3.6 V
I/O voltage range: 1.7 V to V_S
SPI (3- and 4-wire) and I^2C digital interfaces
Flexible interrupt modes mappable to either interrupt pin
Measurement ranges selectable via serial command
Bandwidth selectable via serial command
Wide temperature range (−40°C to +85°C)
10,000 *g* shock survival
Pb free/RoHS compliant
Small and thin: 3 mm × 5 mm × 1 mm LGA package

APPLICATIONS

Handsets
Medical instrumentation
Gaming and pointing devices
Industrial instrumentation
Personal navigation devices
Hard disk drive (HDD) protection
Fitness equipment

GENERAL DESCRIPTION

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 *g*. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I^2C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion and if the acceleration on any axis exceeds a user-set level. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm × 5 mm × 1 mm, 14-lead, plastic package.

FUNCTIONAL BLOCK DIAGRAM

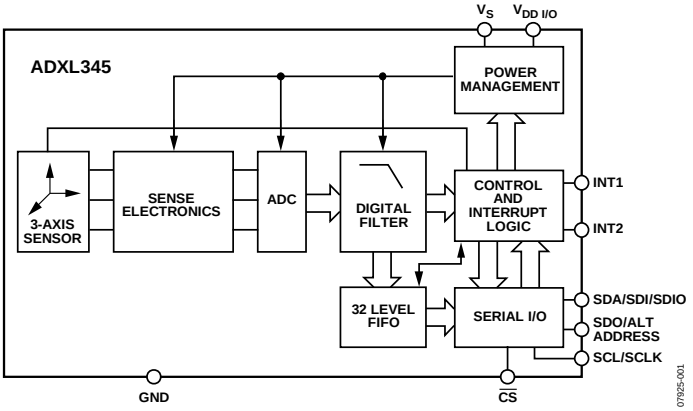


Figure 1.

Rev. 0

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

5/09—Revision 0: Initial Version

SPECIFICATIONS

T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V, acceleration = 0 *g*, C_S = 1 µF tantalum, C_{IO} = 0.1 µF, unless otherwise noted.

Table 1. Specifications[1]

Parameter	Test Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range	User selectable		±2, ±4, ±8, ±16		*g*
Nonlinearity	Percentage of full scale		±0.5		%
Inter-Axis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[2]			±1		%
OUTPUT RESOLUTION	Each axis				
All *g* Ranges	10-bit resolution		10		Bits
±2 *g* Range	Full resolution		10		Bits
±4 *g* Range	Full resolution		11		Bits
±8 *g* Range	Full resolution		12		Bits
±16 *g* Range	Full resolution		13		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	232	256	286	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	3.5	3.9	4.3	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	116	128	143	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	7.0	7.8	8.6	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	58	64	71	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	14.0	15.6	17.2	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	29	32	36	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	28.1	31.2	34.3	mg/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C
0 *g* BIAS LEVEL	Each axis				
0 *g* Output for X_{OUT}, Y_{OUT}		−150	±40	+150	mg
0 *g* Output for Z_{OUT}		−250	±80	+250	mg
0 *g* Offset vs. Temperature for x-, y-Axes			±0.8		mg/°C
0 *g* Offset vs. Temperature for z-Axis			±4.5		mg/°C
NOISE PERFORMANCE					
Noise (x-, y-Axes)	Data rate = 100 Hz for ±2 *g*, 10-bit or full resolution		<1.0		LSB rms
Noise (z-Axis)	Data rate = 100 Hz for ±2 *g*, 10-bit or full resolution		<1.5		LSB rms
OUTPUT DATA RATE AND BANDWIDTH	User selectable				
Measurement Rate[3]		6.25		3200	Hz
SELF-TEST[4]	Data rate ≥ 100 Hz, 2.0 V ≤ V_S ≤ 3.6 V				
Output Change in x-Axis		0.20		2.10	*g*
Output Change in y-Axis		−2.10		−0.20	*g*
Output Change in z-Axis		0.30		3.40	*g*
POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range ($V_{DD\ I/O}$)	V_S ≤ 2.5 V	1.7	1.8	V_S	V
	V_S ≥ 2.5 V	2.0	2.5	V_S	V
Supply Current	Data rate > 100 Hz		145		µA
	Data rate < 10 Hz		40		µA
Standby Mode Leakage Current			0.1	2	µA
Turn-On Time[5]	Data rate = 3200 Hz		1.4		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C
WEIGHT					
Device Weight			20		mg

[1] All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.
[2] Cross-axis sensitivity is defined as coupling between any two axes.
[3] Bandwidth is half the output data rate.
[4] Self-test change is defined as the output (*g*) when the SELF_TEST bit = 1 (in the DATA_FORMAT register) minus the output (*g*) when the SELF_TEST bit = 0 (in the DATA_FORMAT register). Due to device filtering, the output reaches its final value after 4 × τ when enabling or disabling self-test, where τ = 1/(data rate).
[5] Turn-on and wake-up times are determined by the user-defined bandwidth. At a 100 Hz data rate, the turn-on and wake-up times are each approximately 11.1 ms. For other data rates, the turn-on and wake-up times are each approximately τ + 1.1 in milliseconds, where τ = 1/(data rate).

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration	
Any Axis, Unpowered	10,000 *g*
Any Axis, Powered	10,000 *g*
V_S	−0.3 V to +3.6 V
$V_{DD\ I/O}$	−0.3 V to +3.6 V
Digital Pins	−0.3 V to $V_{DD\ I/O}$ + 0.3 V or 3.6 V, whichever is less
All Other Pins	−0.3 V to +3.6 V
Output Short-Circuit Duration (Any Pin to Ground)	Indefinite
Temperature Range	
Powered	−40°C to +105°C
Storage	−40°C to +105°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

THERMAL RESISTANCE

Table 3. Package Characteristics

Package Type	θ_{JA}	θ_{JC}	Device Weight
14-Terminal LGA	150°C/W	85°C/W	20 mg

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

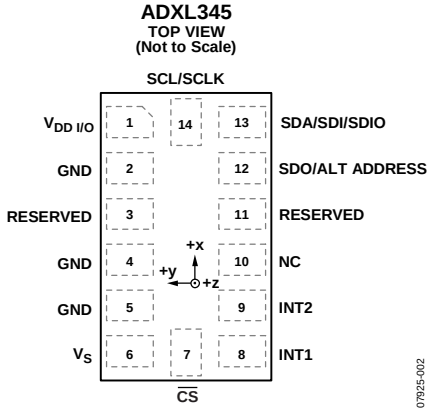


Figure 2. Pin Configuration

Table 4. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	Must be connected to ground.
3	Reserved	Reserved. This pin must be connected to V_S or left open.
4	GND	Must be connected to ground.
5	GND	Must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	Reserved	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output/Alternate I^2C Address Select.
13	SDA/SDI/SDIO	Serial Data (I^2C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock.

THEORY OF OPERATION

The ADXL345 is a complete 3-axis acceleration measurement system with a selectable measurement range of ±2 *g*, ±4 *g*, ±8 *g*, or ±16 *g*. It measures both dynamic acceleration resulting from motion or shock and static acceleration, such as gravity, which allows the device to be used as a tilt sensor.

The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces.

Deflection of the structure is measured using differential capacitors that consist of independent fixed plates and plates attached to the moving mass. Acceleration deflects the beam and unbalances the differential capacitor, resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation is used to determine the magnitude and polarity of the acceleration.

POWER SEQUENCING

Power can be applied to V_S or $V_{DD\ I/O}$ in any sequence without damaging the ADXL345. All possible power-on modes are summarized in Table 5. The interface voltage level is set with the interface supply voltage, $V_{DD\ I/O}$, which must be present to ensure that the ADXL345 does not create a conflict on the communication bus. For single-supply operation, $V_{DD\ I/O}$ can be the same as the main supply, V_S. In a dual-supply application, however, $V_{DD\ I/O}$ can differ from V_S to accommodate the desired interface voltage, as long as V_S is greater than $V_{DD\ I/O}$.

After V_S is applied, the device enters standby mode, where power consumption is minimized and the device waits for $V_{DD\ I/O}$ to be applied and for the command to enter measurement mode to be received. (This command can be initiated by setting the measure bit in the POWER_CTL register (Address 0x2D).) In addition, any register can be written to or read from to configure the part while the device is in standby mode. It is recommended to configure the device in standby mode and then to enable measurement mode. Clearing the measure bit returns the device to the standby mode.

Table 5. Power Sequencing

Condition	V_S	$V_{DD\ I/O}$	Description
Power Off	Off	Off	The device is completely off, but there is a potential for a communication bus conflict.
Bus Disabled	On	Off	The device is on in standby mode, but communication is unavailable and will create a conflict on the communication bus. The duration of this state should be minimized during power-up to prevent a conflict.
Bus Enabled	Off	On	No functions are available, but the device will not create a conflict on the communication bus.
Standby or Measurement	On	On	At power-up, the device is in standby mode, awaiting a command to enter measurement mode, and all sensor functions are off. After the device is instructed to enter measurement mode, all sensor functions are available.

POWER SAVINGS

Power Modes

The ADXL345 automatically modulates its power consumption in proportion to its output data rate, as outlined in Table 6. If additional power savings is desired, a lower power mode is available. In this mode, the internal sampling rate is reduced, allowing for power savings in the 12.5 Hz to 400 Hz data rate range but at the expense of slightly greater noise. To enter lower power mode, set the LOW_POWER bit (Bit 4) in the BW_RATE register (Address 0x2C). The current consumption in low power mode is shown in Table 7 for cases where there is an advantage for using low power mode. The current consumption values shown in Table 6 and Table 7 are for a V_S of 2.5 V. Current scales linearly with V_S.

Table 6. Current Consumption vs. Data Rate (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (μA)
3200	1600	1111	145
1600	800	1110	100
800	400	1101	145
400	200	1100	145
200	100	1011	145
100	50	1010	145
50	25	1001	100
25	12.5	1000	65
12.5	6.25	0111	55
6.25	3.125	0110	40

Table 7. Current Consumption vs. Data Rate, Low Power Mode (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (μA)
400	200	1100	100
200	100	1011	65
100	50	1010	55
50	25	1001	50
25	12.5	1000	40
12.5	6.25	0111	40

Auto Sleep Mode

Additional power can be saved if the ADXL345 automatically switches to sleep mode during periods of inactivity. To enable this feature, set the THRESH_INACT register (Address 0x25) and the TIME_INACT register (Address 0x26) each to a value that signifies inactivity (the appropriate value depends on the application), and then set the AUTO_SLEEP bit and the link bit in the POWER_CTL register (Address 0x2D). Current consumption at the sub-8 Hz data rates used in this mode is typically 40 μA for a V_S of 2.5 V.

Standby Mode

For even lower power operation, standby mode can be used. In standby mode, current consumption is reduced to 0.1 μA (typical). In this mode, no measurements are made. Standby mode is entered by clearing the measure bit (Bit 3) in the POWER_CTL register (Address 0x2D). Placing the device into standby mode preserves the contents of FIFO.

SERIAL COMMUNICATIONS

I²C and SPI digital communications are available. In both cases, the ADXL345 operates as a slave. I²C mode is enabled if the $\overline{CS}$ pin is tied high to $V_{DD\ I/O}$. The $\overline{CS}$ pin should always be tied high to $V_{DD\ I/O}$ or be driven by an external controller because there is no default mode if the $\overline{CS}$ pin is left unconnected. Therefore, not taking these precautions may result in an inability to communicate with the part. In SPI mode, the $\overline{CS}$ pin is controlled by the bus master. In both SPI and I²C modes of operation, data transmitted from the ADXL345 to the master device should be ignored during writes to the ADXL345.

SPI

For SPI, either 3- or 4-wire configuration is possible, as shown in the connection diagrams in Figure 3 and Figure 4. Clearing the SPI bit in the DATA_FORMAT register (Address 0x31) selects 4-wire mode, whereas setting the SPI bit selects 3-wire mode. The maximum SPI clock speed is 5 MHz with 100 pF maximum loading, and the timing scheme follows clock polarity (CPOL) = 1 and clock phase (CPHA) = 1.

$\overline{CS}$ is the serial port enable line and is controlled by the SPI master. This line must go low at the start of a transmission and high at the end of a transmission, as shown in Figure 5. SCLK is the serial port clock and is supplied by the SPI master. It is stopped high when $\overline{CS}$ is high during a period of no transmission. SDI and SDO are the serial data input and output, respectively. Data should be sampled at the rising edge of SCLK.

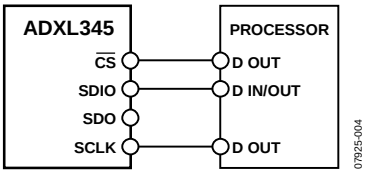


Figure 3. 3-Wire SPI Connection Diagram

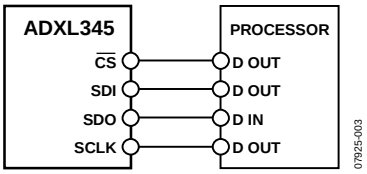


Figure 4. 4-Wire SPI Connection Diagram

To read or write multiple bytes in a single transmission, the multiple-byte bit, located after the R/$\overline{W}$ bit in the first byte transfer (MB in Figure 5 to Figure 7), must be set. After the register addressing and the first byte of data, each subsequent set of clock pulses (eight clock pulses) causes the ADXL345 to point to the next register for a read or write. This shifting continues until the clock pulses cease and $\overline{CS}$ is deasserted. To perform reads or writes on different, nonsequential registers, $\overline{CS}$ must be deasserted between transmissions and the new register must be addressed separately.

The timing diagram for 3-wire SPI reads or writes is shown in Figure 7. The 4-wire equivalents for SPI writes and reads are shown in Figure 5 and Figure 6, respectively.

Table 8. SPI Digital Input/Output Voltage

Parameter	Limit[1]	Unit
Digital Input Voltage		
Low Level Input Voltage (V_{IL})	$0.2 \times V_{DD\ I/O}$	V max
High Level Input Voltage (V_{IH})	$0.8 \times V_{DD\ I/O}$	V min
Digital Output Voltage		
Low Level Output Voltage (V_{OL})	$0.15 \times V_{DD\ I/O}$	V max
High Level Output Voltage (V_{OH})	$0.85 \times V_{DD\ I/O}$	V min

[1] Limits based on characterization results, not production tested.

Table 9. SPI Timing (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)[1]

	Limit[2, 3]			
Parameter	Min	Max	Unit	Description
f_{SCLK}		5	MHz	SPI clock frequency
t_{SCLK}	200		ns	1/(SPI clock frequency) mark-space ratio for the SCLK input is 40/60 to 60/40
t_{DELAY}	10		ns	$\overline{CS}$ falling edge to SCLK falling edge
t_{QUIET}	10		ns	SCLK rising edge to $\overline{CS}$ rising edge
t_{DIS}		100	ns	$\overline{CS}$ rising edge to SDO disabled
$t_{CS,DIS}$	250		ns	$\overline{CS}$ deassertion between SPI communications
t_S	$0.4 \times t_{SCLK}$		ns	SCLK low pulse width (space)
t_M	$0.4 \times t_{SCLK}$		ns	SCLK high pulse width (mark)
t_{SDO}		95	ns	SCLK falling edge to SDO transition
t_{SETUP}	10		ns	SDI valid before SCLK rising edge
t_{HOLD}	10		ns	SDI valid after SCLK rising edge

[1] The $\overline{CS}$, SCLK, SDI, and SDO pins are not internally pulled up or down; they must be driven for proper operation.
[2] Limits based on characterization results, characterized with f_{SCLK} = 5 MHz and bus load capacitance of 100 pF; not production tested.
[3] The timing values are measured corresponding to the input thresholds (V_{IL} and V_{IH}) given in Table 8.

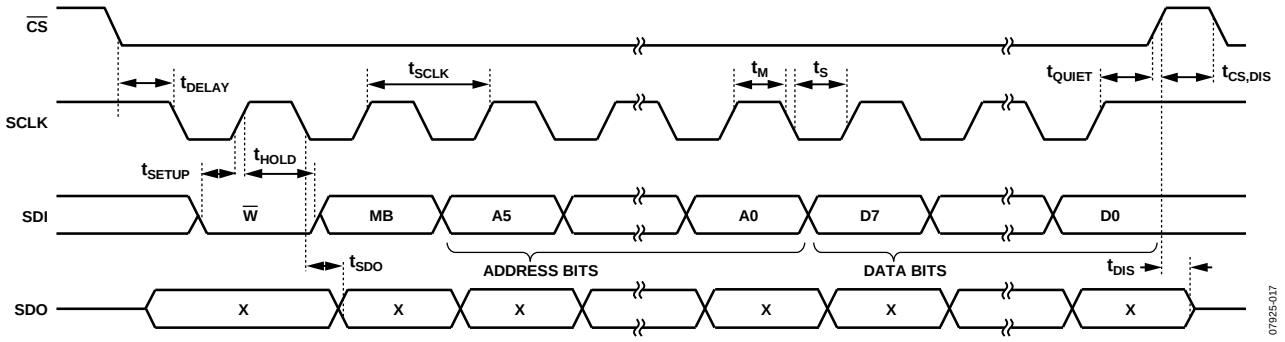


Figure 5. SPI 4-Wire Write

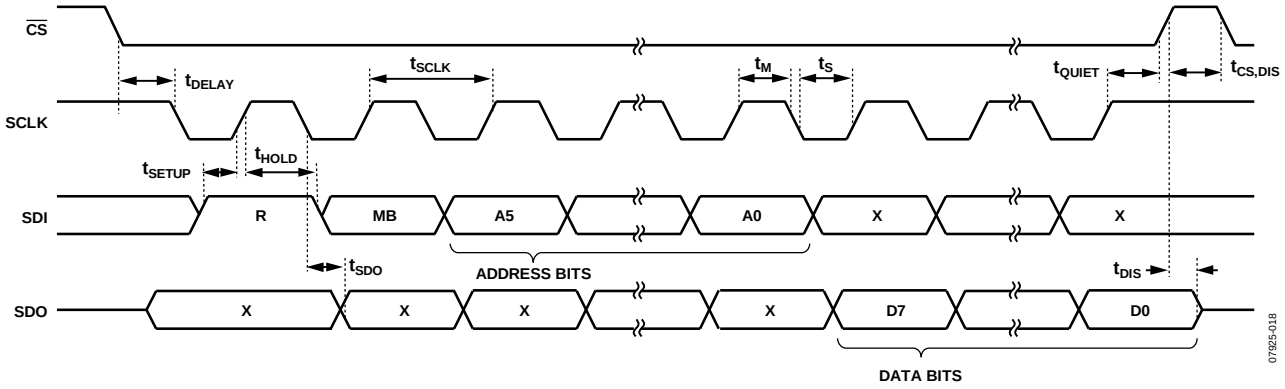
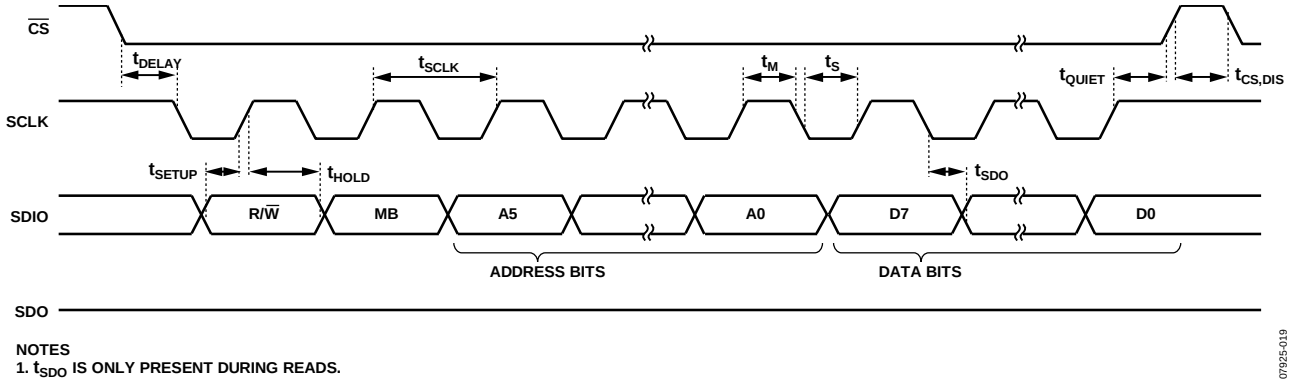


Figure 6. SPI 4-Wire Read



NOTES
1. t_{SDO} IS ONLY PRESENT DURING READS.

Figure 7. SPI 3-Wire Read/Write

I²C

With $\overline{CS}$ tied high to $V_{DD\ I/O}$, the ADXL345 is in I²C mode, requiring a simple 2-wire connection as shown in Figure 8. The ADXL345 conforms to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, available from NXP Semiconductor. It supports standard (100 kHz) and fast (400 kHz) data transfer modes if the timing parameters given in Table 11 and Figure 10 are met. Single- or multiple-byte reads/writes are supported, as shown in Figure 9. With the SDO/ALT ADDRESS pin high, the 7-bit I²C address for the device is 0x1D, followed by the R/$\overline{W}$ bit. This translates to 0x3A for a write and 0x3B for a read. An alternate I²C address of 0x53 (followed by the R/$\overline{W}$ bit) can be chosen by grounding the SDO/ALT ADDRESS pin (Pin 12). This translates to 0xA6 for a write and 0xA7 for a read.

If other devices are connected to the same I²C bus, the nominal operating voltage level of these other devices cannot exceed $V_{DD\ I/O}$ by more than 0.3 V. External pull-up resistors, R_P, are necessary for proper I²C operation. Refer to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, when selecting pull-up resistor values to ensure proper operation.

Table 10. I²C Digital Input/Output Voltage

Parameter	Limit[1]	Unit
Digital Input Voltage		
Low Level Input Voltage (V_{IL})	$0.25 \times V_{DD\ I/O}$	V max
High Level Input Voltage (V_{IH})	$0.75 \times V_{DD\ I/O}$	V min
Digital Output Voltage		
Low Level Output Voltage (V_{OL})[2]	$0.2 \times V_{DD\ I/O}$	V max

[1] Limits based on characterization results; not production tested.
[2] The limit given is only for $V_{DD\ I/O} < 2$ V. When $V_{DD\ I/O} > 2$ V, the limit is 0.4 V max.

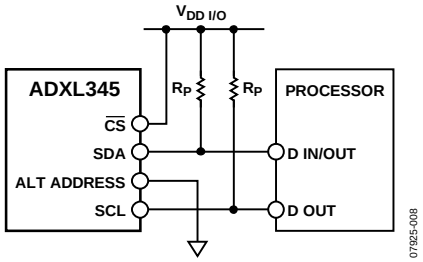


Figure 8. I²C Connection Diagram (Address 0x53)

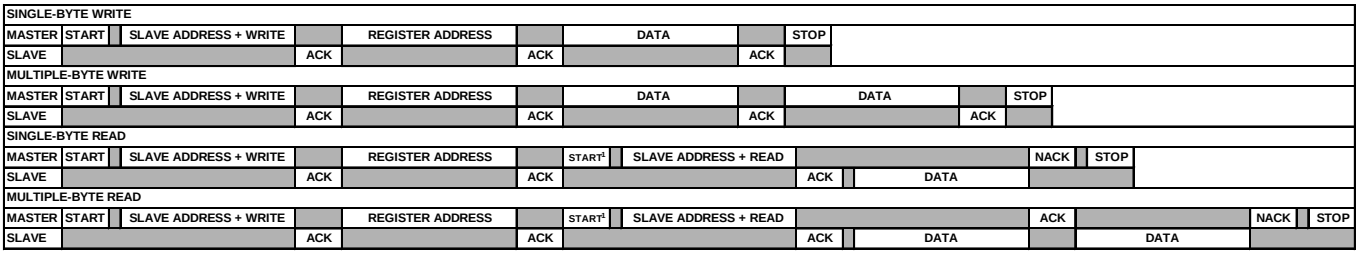


Figure 9. I²C Device Addressing

Table 11. I²C Timing (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

	Limit[1, 2]			
Parameter	**Min**	**Max**	**Unit**	**Description**
f_{SCL}		400	kHz	SCL clock frequency
t_1	2.5		µs	SCL cycle time
t_2	0.6		µs	t_{HIGH}, SCL high time
t_3	1.3		µs	t_{LOW}, SCL low time
t_4	0.6		µs	$t_{HD, STA}$, start/repeated start condition hold time
t_5	350		ns	$t_{SU, DAT}$, data setup time
t_6[3, 4, 5, 6]	0	0.65	µs	$t_{HD, DAT}$, data hold time
t_7	0.6		µs	$t_{SU, STA}$, setup time for repeated start
t_8	0.6		µs	$t_{SU, STO}$, stop condition setup time
t_9	1.3		µs	t_{BUF}, bus-free time between a stop condition and a start condition
t_{10}		300	ns	t_R, rise time of both SCL and SDA when receiving
	0		ns	t_R, rise time of both SCL and SDA when receiving or transmitting
t_{11}		250	ns	t_F, fall time of SDA when receiving
		300	ns	t_F, fall time of both SCL and SDA when transmitting
	20 + 0.1 C_b[7]		ns	t_F, fall time of both SCL and SDA when transmitting or receiveing
C_b		400	pF	Capacitive load for each bus line

[1] Limits based on characterization results, with f_{SCL} = 400 kHz and a 3 mA sink current; not production tested.
[2] All values referred to the V_{IH} and the V_{IL} levels given in Table 10.
[3] t_6 is the data hold time that is measured from the falling edge of SCL. It applies to data in transmission and acknowledge times.
[4] A transmitting device must internally provide an output hold time of at least 300 ns for the SDA signal (with respect to $V_{IH(min)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.
[5] The maximum t_6 value must be met only if the device does not stretch the low period (t_3) of the SCL signal.
[6] The maximum value for t_6 is a function of the clock low time (t_3), the clock rise time (t_{10}), and the minimum data setup time ($t_{5(min)}$). This value is calculated as $t_{6(max)} = t_3 - t_{10} - t_{5(min)}$.
[7] C_b is the total capacitance of one bus line in picofarads.

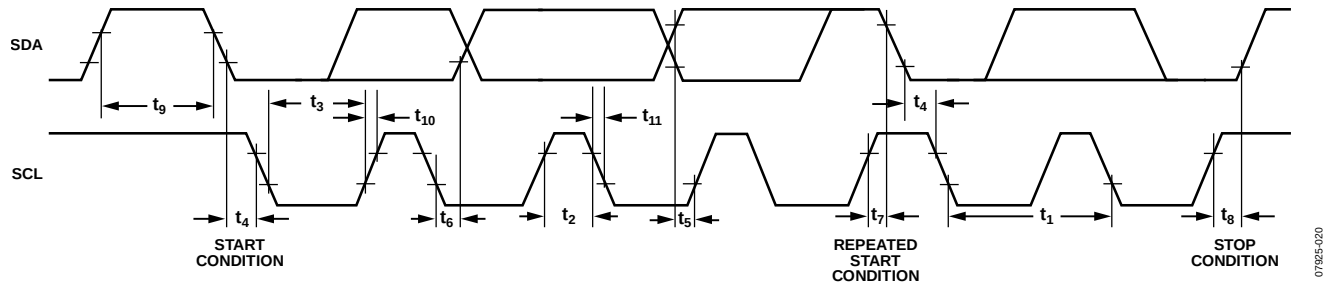


Figure 10. I²C Timing Diagram

INTERRUPTS

The ADXL345 provides two output pins for driving interrupts: INT1 and INT2. Each interrupt function is described in detail in this section. All functions can be used simultaneously, with the only limiting feature being that some functions may need to share interrupt pins. Interrupts are enabled by setting the appropriate bit in the INT_ENABLE register (Address 0x2E) and are mapped to either the INT1 or INT2 pin based on the contents of the INT_MAP register (Address 0x2F). It is recommended that interrupt bits be configured with the interrupts disabled, preventing interrupts from being accidentally triggered during configuration. This can be done by writing a value of 0x00 to the INT_ENABLE register. Clearing interrupts is performed either by reading the data registers (Address 0x32 to Address 0x37) until the interrupt condition is no longer valid for the data-related interrupts or by reading the INT_SOURCE register (Address 0x30) for the remaining interrupts. This section describes the interrupts that can be set in the INT_ENABLE register and monitored in the INT_SOURCE register.

DATA_READY

The DATA_READY bit is set when new data is available and is cleared when no new data is available.

SINGLE_TAP

The SINGLE_TAP bit is set when a single acceleration event that is greater than the value in the THRESH_TAP register (Address 0x1D) occurs for less time than is specified in the DUR register (Address 0x21).

DOUBLE_TAP

The DOUBLE_TAP bit is set when two acceleration events that are greater than the value in the THRESH_TAP register (Address 0x1D) occur for less time than is specified in the DUR register (Address 0x21), with the second tap starting after the time specified by the latent register (Address 0x22) but within the time specified in the window register (Address 0x23). See the Tap Detection section for more details.

Activity

The activity bit is set when acceleration greater than the value stored in the THRESH_ACT register (Address 0x24) is experienced.

Inactivity

The inactivity bit is set when acceleration of less than the value stored in the THRESH_INACT register (Address 0x25) is experienced for more time than is specified in the TIME_INACT register (Address 0x26). The maximum value for TIME_INACT is 255 sec.

FREE_FALL

The FREE_FALL bit is set when acceleration of less than the value stored in the THRESH_FF register (Address 0x28) is experienced for more time than is specified in the TIME_FF register (Address 0x29). The FREE_FALL interrupt differs from the inactivity interrupt as follows: all axes always participate, the timer period is much smaller (1.28 sec maximum), and the mode of operation is always dc-coupled.

Watermark

The watermark bit is set when the number of samples in FIFO equals the value stored in the samples bits (Register FIFO_CTL, Address 0x38). The watermark bit is cleared automatically when FIFO is read, and the content returns to a value below the value stored in the samples bits.

Overrun

The overrun bit is set when new data replaces unread data. The precise operation of the overrun function depends on the FIFO mode. In bypass mode, the overrun bit is set when new data replaces unread data in the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). In all other modes, the overrun bit is set when FIFO is filled. The overrun bit is automatically cleared when the contents of FIFO are read.

FIFO

The ADXL345 contains patent pending technology for an embedded 32-level FIFO that can be used to minimize host processor burden. This buffer has four modes: bypass, FIFO, stream, and trigger (see Table 19). Each mode is selected by the settings of the FIFO_MODE bits in the FIFO_CTL register (Address 0x38).

Bypass Mode

In bypass mode, FIFO is not operational and, therefore, remains empty.

FIFO Mode

In FIFO mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples until it is full (32 samples from measurements of the x-, y-, and z-axes) and then stops collecting data. After FIFO stops collecting data, the device continues to operate; therefore, features such as tap detection can be used after FIFO is full. The watermark interrupt continues to occur until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Stream Mode

In stream mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples and holds the latest 32 samples from measurements of the x-, y-, and z-axes, discarding older data as new data arrives. The watermark interrupt continues occurring until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Trigger Mode

In trigger mode, FIFO accumulates samples, holding the latest 32 samples from measurements of the x-, y-, and z-axes. After a trigger event occurs and an interrupt is sent to the INT1 or INT2 pin (determined by the trigger bit in the FIFO_CTL register), FIFO keeps the last n samples (where n is the value specified by the samples bits in the FIFO_CTL register) and then operates in FIFO mode, collecting new samples only when FIFO is not full. A delay of at least 5 μs should be present between the trigger event occurring and the start of reading data from the FIFO to allow the FIFO to discard and retain the necessary samples. Additional trigger events cannot be recognized until the trigger mode is reset. To reset the trigger mode, set the device to bypass mode and then set the device back to trigger mode. Note that the FIFO data should be read first because placing the device into bypass mode clears FIFO.

Retrieving Data from FIFO

The FIFO data is read through the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). When the FIFO is in FIFO, stream, or trigger mode, reads to the DATAX, DATAY, and DATAZ registers read data stored in the FIFO. Each time data is read from the FIFO, the oldest x-, y-, and z-axes data are placed into the DATAX, DATAY and DATAZ registers.

If a single-byte read operation is performed, the remaining bytes of data for the current FIFO sample are lost. Therefore, all axes of interest should be read in a burst (or multiple-byte) read operation. To ensure that the FIFO has completely popped (that is, that new data has completely moved into the DATAX, DATAY, and DATAZ registers), there must be at least 5 μs between the end of reading the data registers and the start of a new read of the FIFO or a read of the FIFO_STATUS register (Address 0x39). The end of reading a data register is signified by the transition from Register 0x37 to Register 0x38 or by the $\overline{CS}$ pin going high.

For SPI operation at 1.6 MHz or less, the register addressing portion of the transmission is a sufficient delay to ensure that the FIFO has completely popped. For SPI operation greater than 1.6 MHz, it is necessary to deassert the $\overline{CS}$ pin to ensure a total delay of 5 μs; otherwise, the delay will not be sufficient. The total delay necessary for 5 MHz operation is at most 3.4 μs. This is not a concern when using I^2C mode because the communication rate is low enough to ensure a sufficient delay between FIFO reads.

SELF-TEST

The ADXL345 incorporates a self-test feature that effectively tests its mechanical and electronic systems simultaneously. When the self-test function is enabled (via the SELF_TEST bit in the DATA_FORMAT register, Address 0x31), an electrostatic force is exerted on the mechanical sensor. This electrostatic force moves the mechanical sensing element in the same manner as acceleration, and it is additive to the acceleration experienced by the device. This added electrostatic force results in an output change in the x-, y-, and z-axes. Because the electrostatic force is proportional to V_S^2, the output change varies with V_S. The self-test feature of the ADXL345 also exhibits a bimodal behavior that depends on which phase of the clock self-test is enabled. However, the limits shown in Table 1 and Table 12 to Table 15 are valid for all potential self-test values across the entire allowable voltage range. Use of the self-test feature at data rates less than 100 Hz may yield values outside these limits. Therefore, the part should be placed into a data rate of 100 Hz or greater when using self-test.

Table 12. Self-Test Output in LSB for ±2 *g*, Full Resolution

Axis	Min	Max	Unit
X	50	540	LSB
Y	−540	−50	LSB
Z	75	875	LSB

Table 13. Self-Test Output in LSB for ±4 *g*, 10-Bit Resolution

Axis	Min	Max	Unit
X	25	270	LSB
Y	−270	−25	LSB
Z	38	438	LSB

Table 14. Self-Test Output in LSB for ±8 *g*, 10-Bit Resolution

Axis	Min	Max	Unit
X	12	135	LSB
Y	−135	−12	LSB
Z	19	219	LSB

Table 15. Self-Test Output in LSB for ±16 *g*, 10-Bit Resolution

Axis	Min	Max	Unit
X	6	67	LSB
Y	−67	−6	LSB
Z	10	110	LSB

REGISTER MAP

Table 16. Register Map

Address					
Hex	**Dec**	**Name**	**Type**	**Reset Value**	**Description**
0x00	0	DEVID	R	11100101	Device ID.
0x01 to 0x01C	1 to 28	Reserved			Reserved. Do not access.
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold.
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset.
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset.
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset.
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration.
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency.
0x23	35	Window	R/$\overline{W}$	00000000	Tap window.
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold.
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold.
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time.
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection.
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold.
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time.
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for tap/double tap.
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of tap/double tap.
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control.
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control.
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control.
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control.
0x30	48	INT_SOURCE	R	00000010	Source of interrupts.
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control.
0x32	50	DATAX0	R	00000000	X-Axis Data 0.
0x33	51	DATAX1	R	00000000	X-Axis Data 1.
0x34	52	DATAY0	R	00000000	Y-Axis Data 0.
0x35	53	DATAY1	R	00000000	Y-Axis Data 1.
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0.
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1.
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control.
0x39	57	FIFO_STATUS	R	00000000	FIFO status.

REGISTER DEFINITIONS

Register 0x00—DEVID (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	1	0	1

The DEVID register holds a fixed device ID code of 0xE5 (345 octal).

Register 0x1D—THRESH_TAP (Read/Write)

The THRESH_TAP register is eight bits and holds the threshold value for tap interrupts. The data format is unsigned, so the magnitude of the tap event is compared with the value in THRESH_TAP. The scale factor is 62.5 mg/LSB (that is, 0xFF = +16 *g*). A value of 0 may result in undesirable behavior if tap/double tap interrupts are enabled.

Register 0x1E, Register 0x1F, Register 0x20—OFSX, OFSY, OFSZ (Read/Write)

The OFSX, OFSY, and OFSZ registers are each eight bits and offer user-set offset adjustments in twos complement format with a scale factor of 15.6 mg/LSB (that is, 0x7F = +2 *g*).

Register 0x21—DUR (Read/Write)

The DUR register is eight bits and contains an unsigned time value representing the maximum time that an event must be above the THRESH_TAP threshold to qualify as a tap event. The scale factor is 625 µs/LSB. A value of 0 disables the tap/double tap functions.

Register 0x22—Latent (Read/Write)

The latent register is eight bits and contains an unsigned time value representing the wait time from the detection of a tap event to the start of the time window (defined by the window register) during which a possible second tap event can be detected. The scale factor is 1.25 ms/LSB. A value of 0 disables the double tap function.

Register 0x23—Window (Read/Write)

The window register is eight bits and contains an unsigned time value representing the amount of time after the expiration of the latency time (determined by the latent register) during which a second valid tap can begin. The scale factor is 1.25 ms/LSB. A value of 0 disables the double tap function.

Register 0x24—THRESH_ACT (Read/Write)

The THRESH_ACT register is eight bits and holds the threshold value for detecting activity. The data format is unsigned, so the magnitude of the activity event is compared with the value in the THRESH_ACT register. The scale factor is 62.5 mg/LSB. A value of 0 may result in undesirable behavior if the activity interrupt is enabled.

Register 0x25—THRESH_INACT (Read/Write)

The THRESH_INACT register is eight bits and holds the threshold value for detecting inactivity. The data format is unsigned, so the magnitude of the inactivity event is compared with the value in the THRESH_INACT register. The scale factor is 62.5 mg/LSB. A value of 0 m*g* may result in undesirable behavior if the inactivity interrupt is enabled.

Register 0x26—TIME_INACT (Read/Write)

The TIME_INACT register is eight bits and contains an unsigned time value representing the amount of time that acceleration must be less than the value in the THRESH_INACT register for inactivity to be declared. The scale factor is 1 sec/LSB. Unlike the other interrupt functions, which use unfiltered data (see the Threshold section), the inactivity function uses filtered output data. At least one output sample must be generated for the inactivity interrupt to be triggered. This results in the function appearing unresponsive if the TIME_INACT register is set to a value less than the time constant of the output data rate. A value of 0 results in an interrupt when the output data is less than the value in the THRESH_INACT register.

Register 0x27—ACT_INACT_CTL (Read/Write)

D7	D6	D5	D4
ACT ac/dc	ACT_X enable	ACT_Y enable	ACT_Z enable
D3	**D2**	**D1**	**D0**
INACT ac/dc	INACT_X enable	INACT_Y enable	INACT_Z enable

ACT AC/DC and INACT AC/DC Bits

A setting of 0 selects dc-coupled operation, and a setting of 1 enables ac-coupled operation. In dc-coupled operation, the current acceleration magnitude is compared directly with THRESH_ACT and THRESH_INACT to determine whether activity or inactivity is detected.

In ac-coupled operation for activity detection, the acceleration value at the start of activity detection is taken as a reference value. New samples of acceleration are then compared to this reference value, and if the magnitude of the difference exceeds the THRESH_ACT value, the device triggers an activity interrupt.

Similarly, in ac-coupled operation for inactivity detection, a reference value is used for comparison and is updated whenever the device exceeds the inactivity threshold. After the reference value is selected, the device compares the magnitude of the difference between the reference value and the current acceleration with THRESH_INACT. If the difference is less than the value in THRESH_INACT for the time in TIME_INACT, the device is considered inactive and the inactivity interrupt is triggered.

ACT_x Enable Bits and INACT_x Enable Bits

A setting of 1 enables x-, y-, or z-axis participation in detecting activity or inactivity. A setting of 0 excludes the selected axis from participation. If all axes are excluded, the function is disabled.

Register 0x28—THRESH_FF (Read/Write)

The THRESH_FF register is eight bits and holds the threshold value, in unsigned format, for free-fall detection. The root-sum-square (RSS) value of all axes is calculated and compared with the value in THRESH_FF to determine if a free-fall event occurred. The scale factor is 62.5 mg/LSB. Note that a value of 0 m*g* may result in undesirable behavior if the free-fall interrupt is enabled. Values between 300 m*g* and 600 m*g* (0x05 to 0x09) are recommended.

Register 0x29—TIME_FF (Read/Write)

The TIME_FF register is eight bits and stores an unsigned time value representing the minimum time that the RSS value of all axes must be less than THRESH_FF to generate a free-fall interrupt. The scale factor is 5 ms/LSB. A value of 0 may result in undesirable behavior if the free-fall interrupt is enabled. Values between 100 ms and 350 ms (0x14 to 0x46) are recommended.

Register 0x2A—TAP_AXES (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	Suppress	TAP_X enable	TAP_Y enable	TAP_Z enable

Suppress Bit

Setting the suppress bit suppresses double tap detection if acceleration greater than the value in THRESH_TAP is present between taps. See the Tap Detection section for more details.

TAP_x Enable Bits

A setting of 1 in the TAP_X enable, TAP_Y enable, or TAP_Z enable bit enables x-, y-, or z-axis participation in tap detection. A setting of 0 excludes the selected axis from participation in tap detection.

Register 0x2B—ACT_TAP_STATUS (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
0	ACT_X source	ACT_Y source	ACT_Z source	Asleep	TAP_X source	TAP_Y source	TAP_Z source

ACT_x Source and TAP_x Source Bits

These bits indicate the first axis involved in a tap or activity event. A setting of 1 corresponds to involvement in the event, and a setting of 0 corresponds to no involvement. When new data is available, these bits are not cleared but are overwritten by the new data. The ACT_TAP_STATUS register should be read before clearing the interrupt. Disabling an axis from participation clears the corresponding source bit when the next activity or tap/double tap event occurs.

Asleep Bit

A setting of 1 in the asleep bit indicates that the part is asleep, and a setting of 0 indicates that the part is not asleep. See the Register 0x2D—POWER_CTL (Read/Write) section for more information on autosleep mode.

Register 0x2C—BW_RATE (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	LOW_POWER	Rate			

LOW_POWER Bit

A setting of 0 in the LOW_POWER bit selects normal operation, and a setting of 1 selects reduced power operation, which has somewhat higher noise (see the Power Modes section for details).

Rate Bits

These bits select the device bandwidth and output data rate (see Table 6 and Table 7 for details). The default value is 0x0A, which translates to a 100 Hz output data rate. An output data rate should be selected that is appropriate for the communication protocol and frequency selected. Selecting too high of an output data rate with a low communication speed results in samples being discarded.

Register 0x2D—POWER_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Link Bit

A setting of 1 in the link bit with both the activity and inactivity functions enabled delays the start of the activity function until inactivity is detected. After activity is detected, inactivity detection begins, preventing the detection of activity. This bit serially links the activity and inactivity functions. When this bit is set to 0, the inactivity and activity functions are concurrent. Additional information can be found in the Link Mode section.

When clearing the link bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the link bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

AUTO_SLEEP Bit

If the link bit is set, a setting of 1 in the AUTO_SLEEP bit sets the ADXL345 to switch to sleep mode when inactivity is detected (that is, when acceleration has been below the THRESH_INACT value for at least the time indicated by TIME_INACT). A setting of 0 disables automatic switching to sleep mode. See the description of the sleep bit in this section for more information.

When clearing the AUTO_SLEEP bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the AUTO_SLEEP bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

Measure Bit

A setting of 0 in the measure bit places the part into standby mode, and a setting of 1 places the part into measurement mode. The ADXL345 powers up in standby mode with minimum power consumption.

Sleep Bit

A setting of 0 in the sleep bit puts the part into the normal mode of operation, and a setting of 1 places the part into sleep mode. Sleep mode suppresses DATA_READY, stops transmission of data to FIFO, and switches the sampling rate to one specified by the wakeup bits. In sleep mode, only the activity function can be used.

When clearing the sleep bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the sleep bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

Wakeup Bits

These bits control the frequency of readings in sleep mode as described in Table 17.

Table 17. Frequency of Readings in Sleep Mode

Setting		
D1	**D0**	**Frequency (Hz)**
0	0	8
0	1	4
1	0	2
1	1	1

Register 0x2E—INT_ENABLE (Read/Write)

D7 DATA_READY	**D6** SINGLE_TAP	**D5** DOUBLE_TAP	**D4** Activity
D3 Inactivity	**D2** FREE_FALL	**D1** Watermark	**D0** Overrun

Setting bits in this register to a value of 1 enables their respective functions to generate interrupts, whereas a value of 0 prevents the functions from generating interrupts. The DATA_READY, watermark, and overrun bits enable only the interrupt output; the functions are always enabled. It is recommended that interrupts be configured before enabling their outputs.

Register 0x2F—INT_MAP (Read/Write)

D7 DATA_READY	**D6** SINGLE_TAP	**D5** DOUBLE_TAP	**D4** Activity
D3 Inactivity	**D2** FREE_FALL	**D1** Watermark	**D0** Overrun

Any bits set to 0 in this register send their respective interrupts to the INT1 pin, whereas bits set to 1 send their respective interrupts to the INT2 pin. All selected interrupts for a given pin are OR'ed.

Register 0x30—INT_SOURCE (Read Only)

D7 DATA_READY	**D6** SINGLE_TAP	**D5** DOUBLE_TAP	**D4** Activity
D3 Inactivity	**D2** FREE_FALL	**D1** Watermark	**D0** Overrun

Bits set to 1 in this register indicate that their respective functions have triggered an event, whereas a value of 0 indicates that the corresponding event has not occurred. The DATA_READY, watermark, and overrun bits are always set if the corresponding events occur, regardless of the INT_ENABLE register settings, and are cleared by reading data from the DATAX, DATAY, and DATAZ registers. The DATA_READY and watermark bits may require multiple reads, as indicated in the FIFO mode descriptions in the FIFO section. Other bits, and the corresponding interrupts, are cleared by reading the INT_SOURCE register.

Register 0x31—DATA_FORMAT (Read/Write)

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

The DATA_FORMAT register controls the presentation of data to Register 0x32 through Register 0x37. All data, except that for the ±16 *g* range, must be clipped to avoid rollover.

SELF_TEST Bit

A setting of 1 in the SELF_TEST bit applies a self-test force to the sensor, causing a shift in the output data. A value of 0 disables the self-test force.

SPI Bit

A value of 1 in the SPI bit sets the device to 3-wire SPI mode, and a value of 0 sets the device to 4-wire SPI mode.

INT_INVERT Bit

A value of 0 in the INT_INVERT bit sets the interrupts to active high, and a value of 1 sets the interrupts to active low.

FULL_RES Bit

When this bit is set to a value of 1, the device is in full resolution mode, where the output resolution increases with the *g* range set by the range bits to maintain a 4 mg/LSB scale factor. When the FULL_RES bit is set to 0, the device is in 10-bit mode, and the range bits determine the maximum *g* range and scale factor.

Justify Bit

A setting of 1 in the justify bit selects left (MSB) justified mode, and a setting of 0 selects right justified mode with sign extension.

Range Bits

These bits set the *g* range as described in Table 18.

Table 18. *g* Range Setting

Setting		
D1	**D0**	***g* Range**
0	0	±2 *g*
0	1	±4 *g*
1	0	±8 *g*
1	1	±16 *g*

Register 0x32 to Register 0x37—DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0, DATAZ1 (Read Only)

These six bytes (Register 0x32 to Register 0x37) are eight bits each and hold the output data for each axis. Register 0x32 and Register 0x33 hold the output data for the x-axis, Register 0x34 and Register 0x35 hold the output data for the y-axis, and Register 0x36 and Register 0x37 hold the output data for the z-axis. The output data is twos complement, with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z. The DATA_FORMAT register (Address 0x31) controls the format of the data. It is recommended that a multiple-byte read of all registers be performed to prevent a change in data between reads of sequential registers.

Register 0x38—FIFO_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_MODE		Trigger	Samples				

FIFO_MODE Bits

These bits set the FIFO mode, as described in Table 19.

Table 19. FIFO Modes

Setting			
D7	D6	Mode	Function
0	0	Bypass	FIFO is bypassed.
0	1	FIFO	FIFO collects up to 32 values and then stops collecting data, collecting new data only when FIFO is not full.
1	0	Stream	FIFO holds the last 32 data values. When FIFO is full, the oldest data is overwritten with newer data.
1	1	Trigger	When triggered by the trigger bit, FIFO holds the last data samples before the trigger event and then continues to collect data until full. New data is collected only when FIFO is not full.

Trigger Bit

A value of 0 in the trigger bit links the trigger event of trigger mode to INT1, and a value of 1 links the trigger event to INT2.

Samples Bits

The function of these bits depends on the FIFO mode selected (see Table 20). Entering a value of 0 in the samples bits immediately sets the watermark status bit in the INT_SOURCE register, regardless of which FIFO mode is selected. Undesirable operation may occur if a value of 0 is used for the samples bits when trigger mode is used.

Table 20. Samples Bits Functions

FIFO Mode	Samples Bits Function
Bypass	None.
FIFO	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Stream	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Trigger	Specifies how many FIFO samples are retained in the FIFO buffer before a trigger event.

0x39—FIFO_STATUS (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_TRIG	0	Entries					

FIFO_TRIG Bit

A 1 in the FIFO_TRIG bit corresponds to a trigger event occurring, and a 0 means that a FIFO trigger event has not occurred.

Entries Bits

These bits report how many data values are stored in FIFO. Access to collect the data from FIFO is provided through the DATAX, DATAY, and DATAZ registers. FIFO reads must be done in burst or multiple-byte mode because each FIFO level is cleared after any read (single- or multiple-byte) of FIFO. FIFO stores a maximum of 32 entries, which equates to a maximum of 33 entries available at any given time because an additional entry is available at the output filter of the device.

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

A 1 µF tantalum capacitor (C_S) at V_S and a 0.1 µF ceramic capacitor (C_{IO}) at $V_{DD\ I/O}$ placed close to the ADXL345 supply pins is used for testing and is recommended to adequately decouple the accelerometer from noise on the power supply. If additional decoupling is necessary, a resistor or ferrite bead, no larger than 100 Ω, in series with V_S may be helpful. Additionally, increasing the bypass capacitance on V_S to a 10 µF tantalum capacitor in parallel with a 0.1 µF ceramic capacitor may also improve noise.

Care should be taken to ensure that the connection from the ADXL345 ground to the power supply ground has low impedance because noise transmitted through ground has an effect similar to noise transmitted through V_S. It is recommended that V_S and $V_{DD\ I/O}$ be separate supplies to minimize digital clocking noise on the V_S supply. If this is not possible, additional filtering of the supplies as previously mentioned may be necessary.

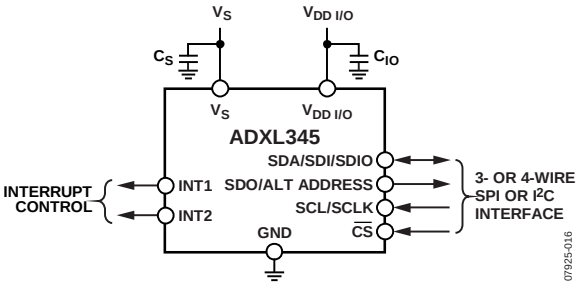


Figure 11. Application Diagram

MECHANICAL CONSIDERATIONS FOR MOUNTING

The ADXL345 should be mounted on the PCB in a location close to a hard mounting point of the PCB to the case. Mounting the ADXL345 at an unsupported PCB location, as shown in Figure 12, may result in large, apparent measurement errors due to undampened PCB vibration. Locating the accelerometer near a hard mounting point ensures that any PCB vibration at the accelerometer is above the accelerometer's mechanical sensor resonant frequency and, therefore, effectively invisible to the accelerometer.

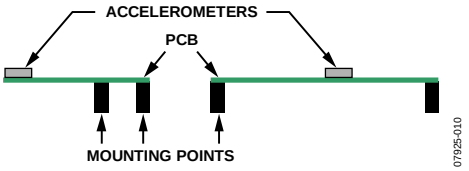


Figure 12. Incorrectly Placed Accelerometers

TAP DETECTION

The tap interrupt function is capable of detecting either single or double taps. The following parameters are shown in Figure 13 for a valid single and valid double tap event:

- The tap detection threshold is defined by the THRESH_TAP register (Address 0x1D).
- The maximum tap duration time is defined by the DUR register (Address 0x21).
- The tap latency time is defined by the latent register (Address 0x22) and is the waiting period from the end of the first tap until the start of the time window, when a second tap can be detected, which is determined by the value in the window register (Address 0x23).
- The interval after the latency time (set by the latent register) is defined by the window register. Although a second tap must begin after the latency time has expired, it need not finish before the end of the time defined by the window register.

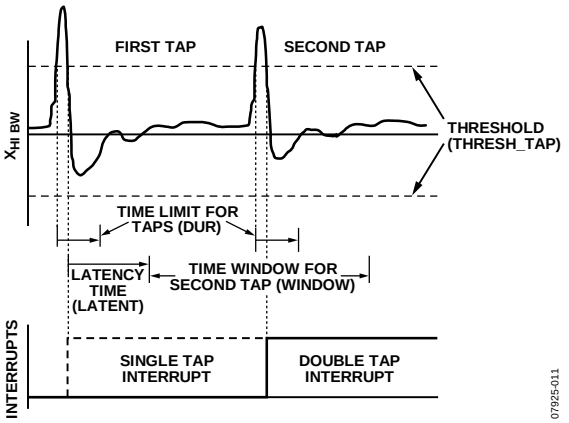


Figure 13. Tap Interrupt Function with Valid Single and Double Taps

If only the single tap function is in use, the single tap interrupt is triggered when the acceleration goes below the threshold, as long as DUR has not been exceeded. If both single and double tap functions are in use, the single tap interrupt is triggered when the double tap event has been either validated or invalidated.

Several events can occur to invalidate the second tap of a double tap event. First, if the suppress bit in the TAP_AXES register (Address 0x2A) is set, any acceleration spike above the threshold during the latency time (set by the latent register) invalidates the double tap detection, as shown in Figure 14.

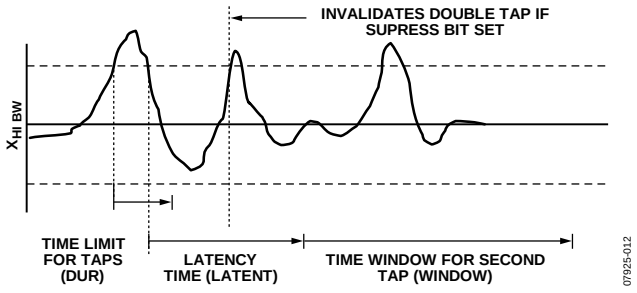


Figure 14. Double Tap Event Invalid Due to High g Event When the Suppress Bit Is Set

A double tap event can also be invalidated if acceleration above the threshold is detected at the start of the time window for the second tap (set by the window register). This results in an invalid double tap at the start of this window, as shown in Figure 15. Additionally, a double tap event can be invalidated if an accel-

eration exceeds the time limit for taps (set by the DUR register), resulting in an invalid double tap at the end of the DUR time limit for the second tap event, also shown in Figure 15.

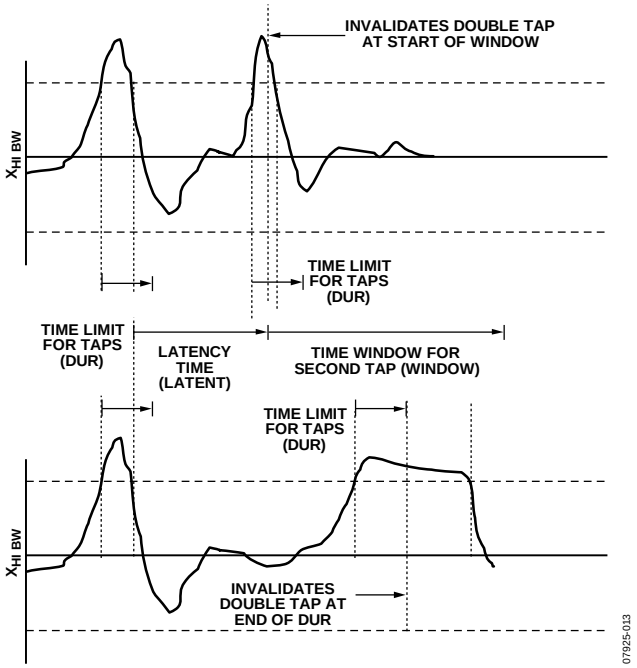


Figure 15. Tap Interrupt Function with Invalid Double Taps

Single taps, double taps, or both can be detected by setting the respective bits in the INT_ENABLE register (Address 0x2E). Control over participation of each of the three axes in single tap/double tap detection is exerted by setting the appropriate bits in the TAP_AXES register (Address 0x2A). For the double tap function to operate, both the latent and window registers must be set to a nonzero value.

Every mechanical system has somewhat different single tap/double tap responses based on the mechanical characteristics of the system. Therefore, some experimentation with values for the latent, window, and THRESH_TAP registers is required. In general, a good starting point is to set the latent register to a value greater than 0x10, to set the window register to a value greater than 0x10, and to set the THRESH_TAP register to be greater than 3 *g*. Setting a very low value in the latent, window, or THRESH_TAP register may result in an unpredictable response due to the accelerometer picking up echoes of the tap inputs.

After a tap interrupt has been received, the first axis to exceed the THRESH_TAP level is reported in the ACT_TAP_STATUS register (Address 0x2B). This register is never cleared, but is overwritten with new data.

THRESHOLD

The lower output data rates are achieved by decimating a common sampling frequency inside the device. The activity, free-fall, and single tap/double tap detection functions are performed using unfiltered data. Since the output data is filtered, the high frequency and high *g* data that is used to determine activity, free-fall, and single tap/double tap events may not be present if the output of the accelerometer is examined. This may result in trigger events being detected when acceleration does not appear to trigger an event because the unfiltered data may have exceeded a threshold or remained below a threshold for a certain period of time while the filtered output data has not exceeded such a threshold.

LINK MODE

The function of the link bit is to reduce the number of activity interrupts that the processor must service by setting the device to look for activity only after inactivity. For proper operation of this feature, the processor must still respond to the activity and inactivity interrupts by reading the INT_SOURCE register (Address 0x30) and, therefore, clearing the interrupts. If an activity interrupt is not cleared, the part cannot go into autosleep mode. The asleep bit in the ACT_TAP_STATUS register (Address 0x2B) indicates if the part is asleep.

SLEEP MODE VS. LOW POWER MODE

In applications where a low data rate is sufficient and low power consumption is desired, it is recommended that the low power mode be used in conjunction with the FIFO. The sleep mode, while offering a low data rate and low average current consumption, suppresses the DATA_READY interrupt, preventing the accelerometer from sending an interrupt signal to the host processor when data is ready to be collected. In this application, setting the part into low power mode (by setting the LOW_POWER bit in the BW_RATE register) and enabling the FIFO in FIFO mode to collect a large value of samples reduces the power consumption of the ADXL345 and allows the host processor to go to sleep while the FIFO is filling up.

USING SELF-TEST

The self-test change is defined as the difference between the acceleration output of an axis with self-test enabled and the acceleration output of the same axis with self-test disabled (see Endnote 4 of Table 1). This definition assumes that the sensor does not move between these two measurements, because if the sensor moves, a non–self-test related shift corrupts the test.

Proper configuration of the ADXL345 is also necessary for an accurate self-test measurement. The part should be set with a data rate greater than or equal to 100 Hz. This is done by ensuring that a value greater than or equal to 0x0A is written into the rate bits (Bit D3 through Bit D0) in the BW_RATE register (Address 0x2C). It is also recommended that the part be set to full-resolution, 16 *g* mode to ensure that there is sufficient dynamic range for the entire self-test shift. This is done by setting Bit D3 of the DATA_FORMAT register (Address 0x31) and writing a value of 0x03 to the range bits (Bit D1 and Bit D0) of the DATA_FORMAT register (Address 0x31). This results in a high dynamic range for measurement and a 3.9 mg/LSB scale factor.

After the part is configured for accurate self-test measurement, several samples of x-, y-, and z-axis acceleration data should be retrieved from the sensor and averaged together. The number of

samples averaged is a choice of the system designer, but a recommended starting point is 0.1 sec worth of data, which corresponds to 10 samples at 100 Hz data rate. The averaged values should be stored and labeled appropriately as the self-test disabled data, that is, X_{ST_OFF}, Y_{ST_OFF}, and Z_{ST_OFF}.

Next, self-test should be enabled by setting Bit D7 of the DATA_FORMAT register (Address 0x31). The output needs some time (about four samples) to settle after enabling self-test. After allowing the output to settle, several samples of the x-, y-, and z-axis acceleration data should be taken again and averaged. It is recommended that the same number of samples be taken for this average as was previously taken. These averaged values should again be stored and labeled appropriately as the value with self-test enabled, that is, X_{ST_ON}, Y_{ST_ON}, and Z_{ST_ON}. Self-test can then be disabled by clearing Bit D7 of the DATA_FORMAT register (Address 0x31).

With the stored values for self-test enabled and disabled, the self-test change is as follows:

$$X_{ST} = X_{ST\_ON} - X_{ST\_OFF}$$

$$Y_{ST} = Y_{ST\_ON} - Y_{ST\_OFF}$$

$$Z_{ST} = Z_{ST\_ON} - Z_{ST\_OFF}$$

Because the measured output for each axis is expressed in LSBs, X_{ST}, Y_{ST}, and Z_{ST} are also expressed in LSBs. These values can be converted to *g*'s of acceleration by multiplying each value by the 3.9 mg/LSB scale factor, if configured for full-resolution, 16 *g* mode. Additionally, Table 12 through Table 15 correspond to the self-test range converted to LSBs and can be compared with the measured self-test change. If the part was placed into full-resolution, 16 *g* mode, the values listed in Table 12 should be used. Although the fixed 10-bit mode or a range other than 16 *g* can be used, a different set of values, as indicated in Table 13 through Table 15, would need to be used. Using a range below 8 *g* may result in insufficient dynamic range and should be considered when selecting the range of operation for measuring self-test. In addition, note that the range in Table 1 and the values in Table 12 through Table 15 take into account all possible supply voltages, V_S, and no additional conversion due to V_S is necessary.

If the self-test change is within the valid range, the test is considered successful. Generally, a part is considered to pass if the minimum magnitude of change is achieved. However, a part that changes by more than the maximum magnitude is not necessarily a failure.

AXES OF ACCELERATION SENSITIVITY

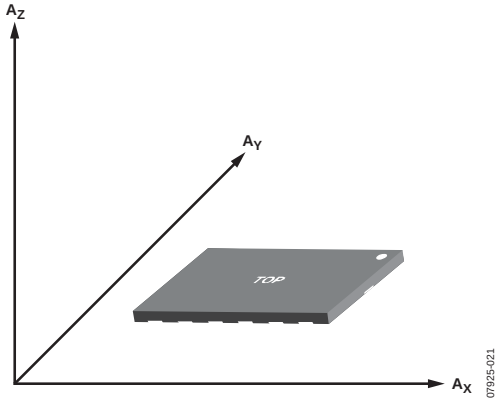


Figure 16. Axes of Acceleration Sensitivity (Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis)

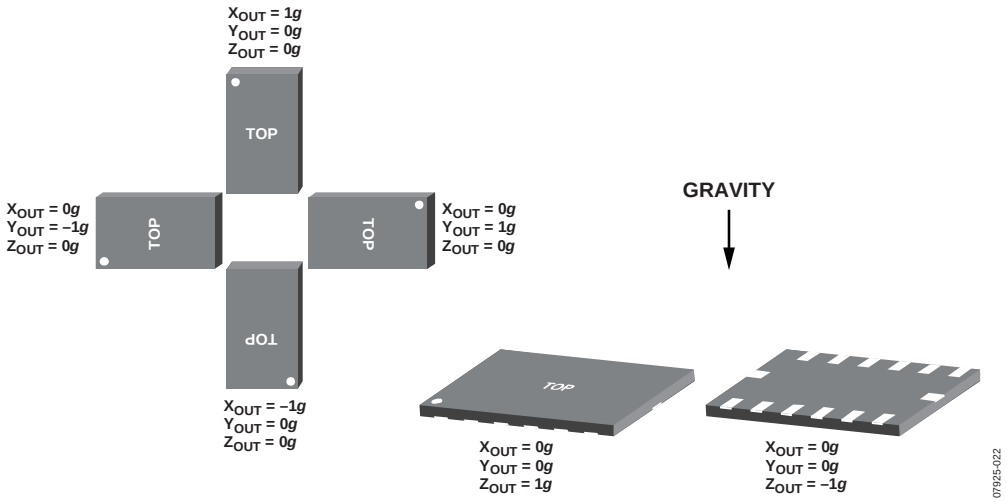


Figure 17. Output Response vs. Orientation to Gravity

LAYOUT AND DESIGN RECOMMENDATIONS

Figure 18 shows the recommended printed wiring board land pattern. Figure 19 and Table 21 provide details about the recommended soldering profile.

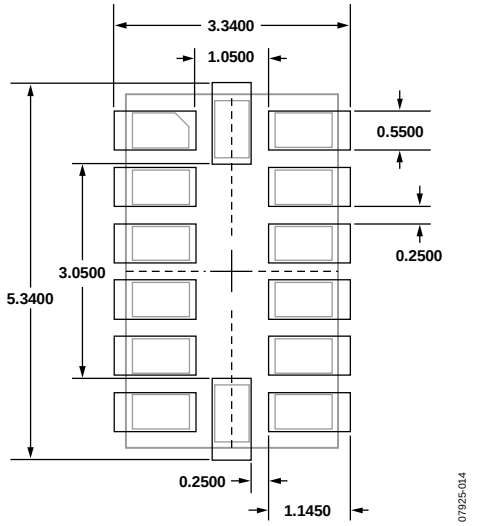


Figure 18. Recommended Printed Wiring Board Land Pattern (Dimensions shown in millimeters)

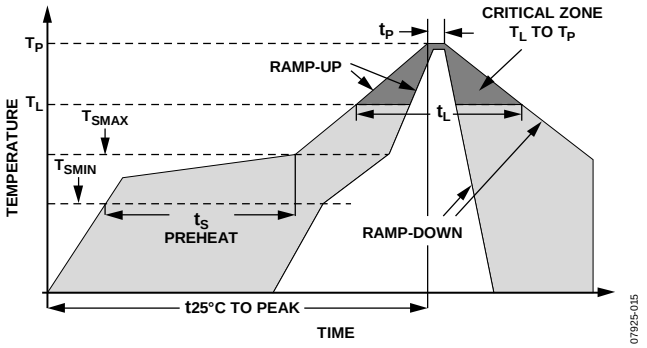


Figure 19. Recommended Soldering Profile

Table 21. Recommended Soldering Profile[1, 2]

	Condition	
Profile Feature	**Sn63/Pb37**	**Pb-Free**
Average Ramp Rate from Liquid Temperature (T_L) to Peak Temperature (T_P)	3°C/sec max	3°C/sec max
Preheat		
Minimum Temperature (T_{SMIN})	100°C	150°C
Maximum Temperature (T_{SMAX})	150°C	200°C
Time from T_{SMIN} to T_{SMAX} (t_S)	60 sec to 120 sec	60 sec to 180 sec
T_{SMAX} to T_L Ramp-Up Rate	3°C/sec max	3°C/sec max
Liquid Temperature (T_L)	183°C	217°C
Time Maintained Above T_L (t_L)	60 sec to 150 sec	60 sec to 150 sec
Peak Temperature (T_P)	240 + 0/−5°C	260 + 0/−5°C
Time of Actual T_P − 5°C (t_P)	10 sec to 30 sec	20 sec to 40 sec
Ramp-Down Rate	6°C/sec max	6°C/sec max
Time 25°C to Peak Temperature	6 minutes max	8 minutes max

[1] Based on JEDEC Standard J-STD-020D.1.
[2] For best results, the soldering profile should be in accordance with the recommendations of the manufacturer of the solder paste used.

OUTLINE DIMENSIONS

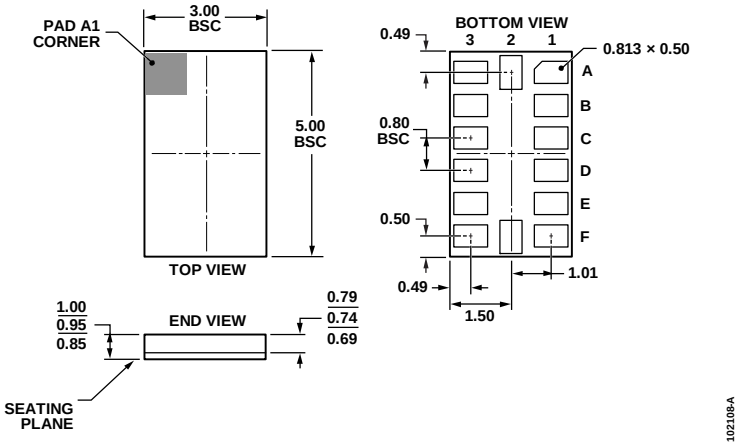


Figure 20. 14-Terminal Land Grid Array [LGA]
(CC-14-1)
Solder Terminations Finish Is Au over Ni
(Dimensions shown in millimeters)

ORDERING GUIDE

Model	Measurement Range (*g*)	Specified Voltage (V)	Temperature Range	Package Description	Package Option
ADXL345BCCZ[1]	±2, ±4, ±8, ±16	2.5	−40°C to +85°C	14-Terminal Land Grid Array [LGA]	CC-14-1
ADXL345BCCZ-RL[1]	±2, ±4, ±8, ±16	2.5	−40°C to +85°C	14-Terminal Land Grid Array [LGA]	CC-14-1
ADXL345BCCZ-RL7[1]	±2, ±4, ±8, ±16	2.5	−40°C to +85°C	14- Terminal Land Grid Array [LGA]	CC-14-1
EVAL-ADXL345Z[1]				Evaluation Board	
EVAL-ADXL345Z-M[1]				Analog Devices Inertial Sensor Evaluation System, Includes ADXL345 Satellite	
EVAL-ADXL345Z-S[1]				ADXL345 Satellite, Standalone	

[1] Z = RoHS Compliant Part.

Rev 2; 6/05

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

General Description

The DS3231 is a low-cost, extremely accurate I²C real-time clock (RTC) with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device as well as reduces the piece-part count in a manufacturing line. The DS3231 is available in commercial and industrial temperature ranges, and is offered in a 16-pin, 300-mil SO package.

The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-of-day alarms and a programmable square-wave output are provided. Address and data are transferred serially through an I²C bidirectional bus.

A precision temperature-compensated voltage reference and comparator circuit monitors the status of V_{CC} to detect power failures, to provide a reset output, and to automatically switch to the backup supply when necessary. Additionally, the $\overline{RST}$ pin is monitored as a pushbutton input for generating a reset externally.

Applications

Servers	Utility Power Meters
Telematics	GPS

Pin Configuration appears at end of data sheet.

Features

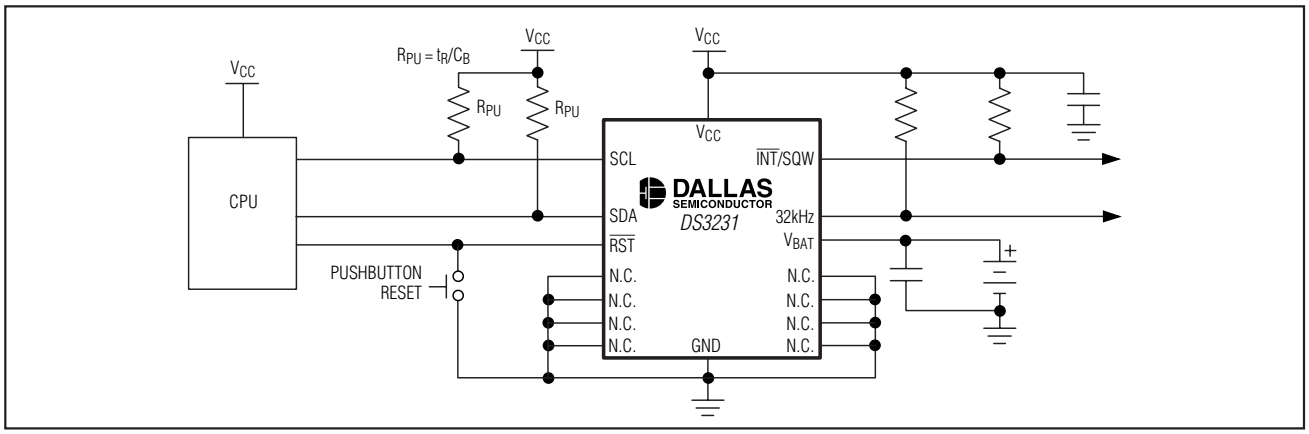
- **Accuracy ±2ppm from 0°C to +40°C**
- **Accuracy ±3.5ppm from -40°C to +85°C**
- **Battery Backup Input for Continuous Timekeeping**
- **Operating Temperature Ranges**
  Commercial: 0°C to +70°C
  Industrial: -40°C to +85°C
- **Low-Power Consumption**
- **Real-Time Clock Counts Seconds, Minutes, Hours, Day, Date, Month, and Year with Leap Year Compensation Valid Up to 2100**
- **Two Time-of-Day Alarms**
- **Programmable Square-Wave Output**
- **Fast (400kHz) I²C Interface**
- **3.3V Operation**
- **Digital Temp Sensor Output: ±3°C Accuracy**
- **Register for Aging Trim**
- **$\overline{RST}$ Input/Output**
- **UL Recognized**

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK
DS3231S	0°C to +70°C	16 SO	DS3231
DS3231SN	-40°C to +85°C	16 SO	DS3231N
DS3231S+	0°C to +70°C	16 SO	DS3231+
DS3231SN+	-40°C to +85°C	16 SO	DS3231N+

+Denotes lead-free

Typical Operating Circuit



Purchase of I²C components from Maxim Integrated Products, Inc., or one of its sublicensed Associated Companies, conveys a license under the Philips I²C Patent Rights to use these components in an I²C system, provided that the system conforms to the I²C Standard Specification as defined by Philips.

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

ABSOLUTE MAXIMUM RATINGS

Voltage Range on V_{CC}, V_{BAT}, 32kHz, SCL, SDA, $\overline{RST}$, $\overline{INT}$/SQW Relative to Ground .....-0.3V to +6.0V
Operating Temperature Range (noncondensing) .....-40°C to +85°C
Junction Temperature .....+125°C
Storage Temperature Range .....-40°C to +85°C
Lead Temperature (Soldering, 10s).....+260°C/10s
Soldering Temperature.....See the *Handling, PC Board Layout, and Assembly* section.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		2.3	3.3	5.5	V
	V_{BAT}		2.3	3.0	5.5	V
Logic 1 Input SDA, SCL	V_{IH}		0.7 x V_{CC}		V_{CC} + 0.3	V
Logic 0 Input SDA, SCL	V_{IL}		-0.3		+0.3 x V_{CC}	V
Pullup Voltage (SDA, SCL, 32kHz, $\overline{INT}$/SQW)	V_{PU}	V_{CC} = 0V			5.5V	V

ELECTRICAL CHARACTERISTICS

(V_{CC} = 2.3V to 5.5V, V_{CC} > V_{BAT}, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Typical values are at V_{CC} = 3.3V, V_{BAT} = 3.0V, and T_A = +25°C, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Active Supply Current	I_{CCA}	(Notes 3, 4)	V_{CC} = 3.63V			200	µA
			V_{CC} = 5.5V			300	
Standby Supply Current	I_{CCS}	I²C bus inactive, 32kHz output on, SQW output off (Note 4)	V_{CC} = 3.63V			110	µA
			V_{CC} = 5.5V			170	
Temperature Conversion Current	$I_{CCSCONV}$	I²C bus inactive, 32kHz output on, SQW output off	V_{CC} = 3.63V			575	µA
			V_{CC} = 5.5V			650	
Power-Fail Voltage	V_{PF}			2.45	2.575	2.70	V
Logic 0 Output, 32kHz, $\overline{INT}$/SQW, SDA	V_{OL}	I_{OL} = 3mA				0.4	V
Logic 0 Output, $\overline{RST}$	V_{OL}	I_{OL} = 1mA				0.4	V
Output Leakage Current 32kHz, $\overline{INT}$/SQW, SDA	I_{LO}	Output high impedance		-1	0	+1	µA
Input Leakage SCL	I_{LI}			-1		+1	µA
$\overline{RST}$ Pin I/O Leakage	I_{OL}	$\overline{RST}$ high impedance (Note 5)		-200		+10	µA
V_{BAT} Leakage Current (V_{CC} Active)	I_{BATLKG}				25	100	nA

ELECTRICAL CHARACTERISTICS (continued)

(V_{CC} = 2.3V to 5.5V, V_{CC} > V_{BAT}, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Typical values are at V_{CC} = 3.3V, V_{BAT} = 3.0V, and T_A = +25°C, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Output Frequency	f_{OUT}	V_{CC} = 3.3V or V_{BAT} = 3.3V			32.768		kHz
Frequency Stability vs. Temperature (Commercial)	$\Delta f/f_{OUT}$	V_{CC} = 3.3V or V_{BAT} = 3.3V, aging offset = 00h	0°C to +40°C			±2	ppm
			>40°C to +70°C			±3.5	
Frequency Stability vs. Temperature (Industrial)	$\Delta f/f_{OUT}$	V_{CC} = 3.3V or V_{BAT} = 3.3V, aging offset = 00h	-40°C to <0°C			±3.5	ppm
			0°C to +40°C			±2	
			>40°C to +85°C			±3.5	
Frequency Stability vs. Voltage	$\Delta f/V$				1		ppm/V
Trim Register Frequency Sensitivity per LSB	$\Delta f/LSB$	Specified at:	-40°C		0.7		ppm
			+25°C		0.1		
			+70°C		0.4		
			+85°C		0.8		
Temperature Accuracy	Temp	V_{CC} = 3.3V or V_{BAT} = 3.3V		-3		+3	°C
Crystal Aging	$\Delta f/f_0$	After reflow, not production tested	First year		±1.0		ppm
			0–10 years		±5.0		

ELECTRICAL CHARACTERISTICS

(**V_{CC} = 0V, V_{BAT} = 2.3V to 5.5V**, T_A = T_{MIN} to T_{MAX}, unless otherw.ise noted.) (Note 1)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Active Battery Current	I_{BATA}	$\overline{EOSC}$ = 0, BBSQW = 0, SCL = 400kHz (Note 4)	V_{BAT} = 3.63V			70	µA
			V_{BAT} = 5.5V			150	
Timekeeping Battery Current	I_{BATT}	$\overline{EOSC}$ = 0, BBSQW = 0, EN32kHz = 1, SCL = SDA = 0V or SCL = SDA = V_{BAT} (Note 4)	V_{BAT} = 3.63V		0.84	3.0	µA
			V_{BAT} = 5.5V		1.0	3.5	
Temperature Conversion Current	I_{BATTC}	$\overline{EOSC}$ = 0, BBSQW = 0, SCL = SDA = 0V or SCL = SDA = V_{BAT}	V_{BAT} = 3.63V			575	µA
			V_{BAT} = 5.5V			650	
Data-Retention Current	I_{BATTDR}	$\overline{EOSC}$ = 1, SCL = SDA = 0V, +25°C				100	nA

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

AC ELECTRICAL CHARACTERISTICS

(V_{CC} = $V_{CC(MIN)}$ to $V_{CC(MAX)}$ or V_{BAT} = $V_{BAT(MIN)}$ to $V_{BAT(MAX)}$, V_{BAT} > V_{CC}, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Note 1)

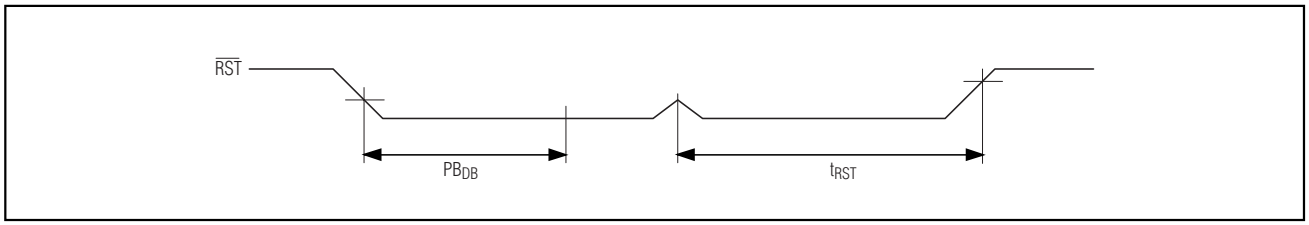
PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL Clock Frequency	f_{SCL}	Fast mode	100		400	kHz
		Standard mode	0		100	
Bus Free Time Between STOP and START Conditions	t_{BUF}	Fast mode	1.3			µs
		Standard mode	4.7			
Hold Time (Repeated) START Condition (Note 6)	$t_{HD:STA}$	Fast mode	0.6			µs
		Standard mode	4.0			
Low Period of SCL Clock	t_{LOW}	Fast mode	1.3			µs
		Standard mode	4.7			
High Period of SCL Clock	t_{HIGH}	Fast mode	0.6			µs
		Standard mode	4.0			
Data Hold Time (Notes 7, 8)	$t_{HD:DAT}$	Fast mode	0		0.9	µs
		Standard mode	0		0.9	
Data Setup Time (Note 9)	$t_{SU:DAT}$	Fast mode	100			ns
		Standard mode	250			
Start Setup Time	$t_{SU:STA}$	Fast mode	0.6			µs
		Standard mode	4.7			
Rise Time of Both SDA and SCL Signals (Note 10)	t_R	Fast mode	20 + 0.1C_B		300	ns
		Standard mode			1000	
Fall Time of Both SDA and SCL Signals (Note 10)	t_F	Fast mode	20 + 0.1C_B		300	ns
		Standard mode			300	
Setup Time for STOP Condition	$t_{SU:STO}$	Fast mode	0.6			µs
		Standard mode	4.7			
Capacitive Load for Each Bus Line (Note 10)	C_B				400	pF
Capacitance for SDA, SCL	$C_{I/O}$			10		pF
Pulse Width of Spikes That Must Be Suppressed by the Input Filter	t_{SP}			30		ns
Pushbutton Debounce	PB_{DB}			250		ms
Reset Active Time	t_{RST}			250		ms
Oscillator Stop Flag (OSF) Delay	t_{OSF}	(Note 11)		100		ms
Temperature Conversion Time	t_{CONV}			125	200	ms

POWER-SWITCH CHARACTERISTICS

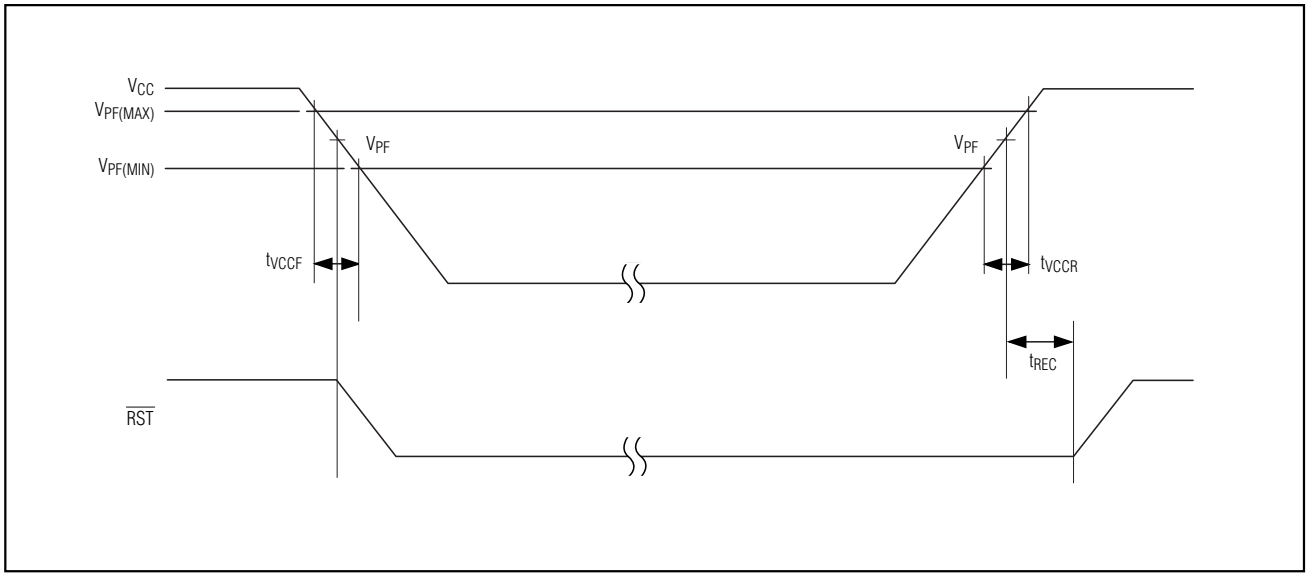
(T_A = T_{MIN} to T_{MAX})

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{CC} Fall Time; $V_{PF(MAX)}$ to $V_{PF(MIN)}$	t_{VCCF}		300			µs
V_{CC} Rise Time; $V_{PF(MIN)}$ to $V_{PF(MAX)}$	t_{VCCR}		0			µs
Recovery at Power-Up	t_{REC}	(Note 12)		250	300	ms

Pushbutton Reset Timing

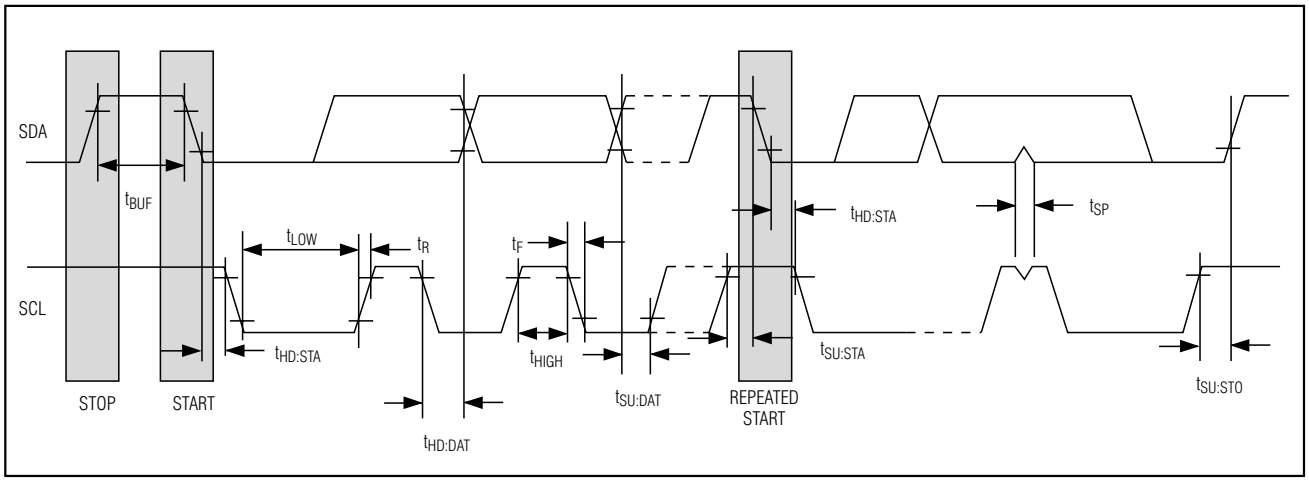


Power-Switch Timing



Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

Data Transfer on I²C Serial Bus

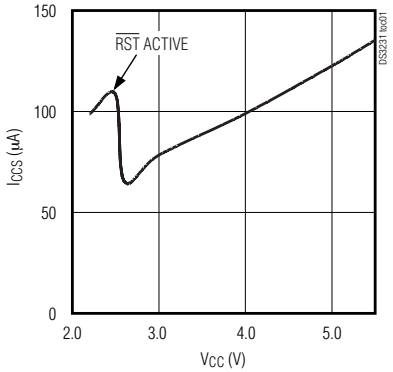


Note 1: Limits at -40°C are guaranteed by design and not production tested.
Note 2: All voltages are referenced to ground.
Note 3: I_{CCA}—SCL clocking at max frequency = 400kHz.
Note 4: Current is the averaged input current, which includes the temperature conversion current.
Note 5: The $\overline{RST}$ pin has an internal 50kΩ (nominal) pullup resistor to V_{CC}.
Note 6: After this period, the first clock pulse is generated.
Note 7: A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the $V_{IH(MIN)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.
Note 8: The maximum $t_{HD:DAT}$ needs only to be met if the device does not stretch the low period (t_{LOW}) of the SCL signal.
Note 9: A fast-mode device can be used in a standard-mode system, but the requirement $t_{SU:DAT} \geq 250ns$ must then be met. This is automatically the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line $t_{R(MAX)} + t_{SU:DAT} = 1000 + 250 = 1250ns$ before the SCL line is released.
Note 10: C_B—total capacitance of one bus line in pF.
Note 11: The parameter t_{OSF} is the period of time the oscillator must be stopped for the OSF flag to be set over the voltage range of $0.0V \leq V_{CC} \leq V_{CC(MAX)}$ and $2.3V \leq V_{BAT} \leq 3.4V$.
Note 12: This delay applies only if the oscillator is enabled and running. If the $\overline{EOSC}$ bit is a 1, the startup time of the oscillator is added to this delay.

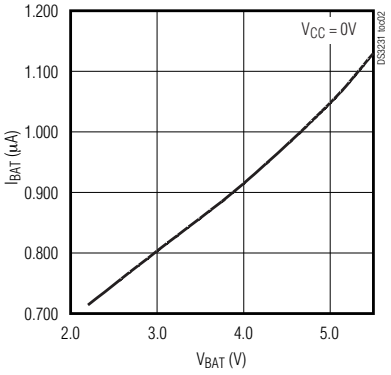
Typical Operating Characteristics

(V_{CC} = +3.3V, T_A = +25°C, unless otherwise noted.)

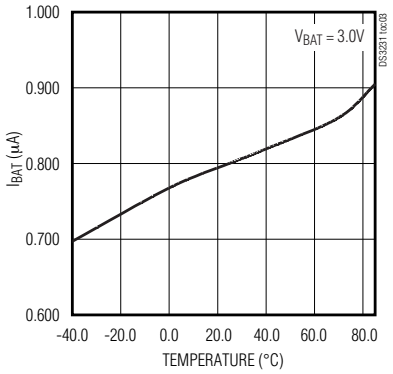
STANDBY SUPPLY CURRENT vs. SUPPLY VOLTAGE



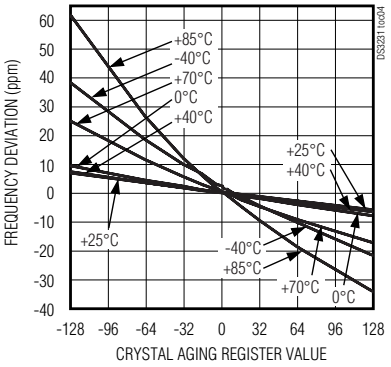
SUPPLY CURRENT vs. SUPPLY VOLTAGE



SUPPLY CURRENT vs. TEMPERATURE

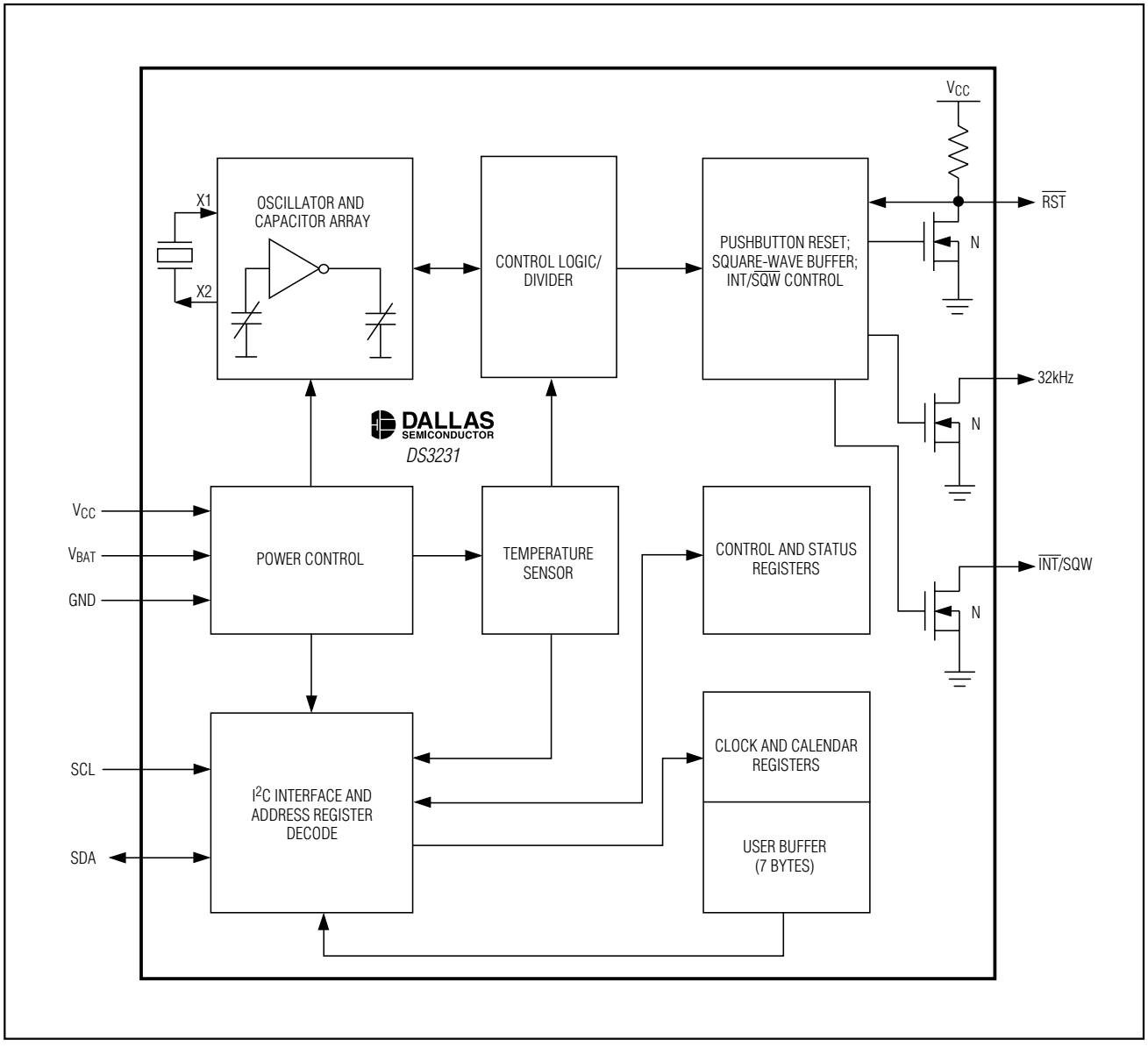


FREQUENCY DEVIATION vs. TEMPERATURE vs. AGING VALUE



Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

Block Diagram



Pin Description

PIN	NAME	FUNCTION
1	32kHz	32kHz Output. This open-drain pin requires an external pullup resistor. It may be left open if not used.
2	V_{CC}	DC Power Pin for Primary Power Supply. This pin should be decoupled using a 0.1µF to 1.0µF capacitor. If not used, connect to ground.
3	$\overline{INT}$/SQW	Active-Low Interrupt or Square-Wave Output. This open-drain pin requires an external pullup resistor. It may be left open if not used. This multifunction pin is determined by the state of the INTCN bit in the Control Register (0Eh). When INTCN is set to logic 0, this pin outputs a square wave and its frequency is determined by RS2 and RS1 bits. When INTCN is set to logic 1, then a match between the timekeeping registers and either of the alarm registers activates the $\overline{INT}$/SQW pin (if the alarm is enabled). Because the INTCN bit is set to logic 1 when power is first applied, the pin defaults to an interrupt output with alarms disabled.
4	$\overline{RST}$	Active-Low Reset. This pin is an open-drain input/output. It indicates the status of V_{CC} relative to the V_{PF} specification. As V_{CC} falls below V_{PF}, the $\overline{RST}$ pin is driven low. When V_{CC} exceeds V_{PF}, for t_{RST}, the $\overline{RST}$ pin is driven high impedance. The active-low, open-drain output is combined with a debounced pushbutton input function. This pin can be activated by a pushbutton reset request. It has an internal 50kΩ nominal value pullup resistor to V_{CC}. No external pullup resistors should be connected. If the crystal oscillator is disabled, the startup time of the oscillator is added to the t_{RST} delay.
5–12	N.C.	No Connection. Must be connected to ground.
13	GND	Ground
14	V_{BAT}	Backup Power-Supply Input. This pin should be decoupled using a 0.1µF to 1.0µF low-leakage capacitor. If the I²C interface is inactive whenever the device is powered by the V_{BAT} input, the decoupling capacitor is not required. If V_{BAT} is not used, connect to ground. UL recognized to ensure against reverse charging when used with a lithium battery. Go to www.maxim-ic.com/qa/info/ul.
15	SDA	Serial Data Input/Output. This pin is the data input/output for the I²C serial interface. This open-drain pin requires an external pullup resistor.
16	SCL	Serial Clock Input. This pin is the clock input for the I²C serial interface and is used to synchronize data movement on the serial interface.

Detailed Description

The DS3231 is a serial RTC driven by a temperature-compensated 32kHz crystal oscillator. The TCXO provides a stable and accurate reference clock, and maintains the RTC to within ±2 minutes per year accuracy from -40°C to +85°C. The TCXO frequency output is available at the 32kHz pin. The RTC is a low-power clock/calendar with two programmable time-of-day alarms and a programmable square-wave output. The $\overline{INT}$/SQW provides either an interrupt signal due to alarm conditions or a square-wave output. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an $\overline{AM}$/PM indicator. The internal registers are accessible though an I²C bus interface.

A temperature-compensated voltage reference and comparator circuit monitors the level of V_{CC} to detect power failures and to automatically switch to the backup supply when necessary. The $\overline{RST}$ pin provides an external pushbutton function and acts as an indicator of a power-fail event.

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

Operation

The block diagram shows the main elements of the DS3231. The eight blocks can be grouped into four functional groups: TCXO, power control, pushbutton function, and RTC. Their operations are described separately in the following sections.

32kHz TCXO

The temperature sensor, oscillator, and control logic form the TCXO. The controller reads the output of the on-chip temperature sensor and uses a lookup table to determine the capacitance required, adds the aging correction in AGE register, and then sets the capacitance selection registers. New values, including changes to the AGE register, are loaded only when a change in the temperature value occurs, or when a user-initiated temperature conversion is completed. The temperature is read on initial application of V_{CC} and once every 64 seconds afterwards.

Power Control

This function is provided by a temperature-compensated voltage reference and a comparator circuit that monitors the V_{CC} level. When V_{CC} is greater than V_{PF}, the part is powered by V_{CC}. When V_{CC} is less than V_{PF} but greater than V_{BAT}, the DS3231 is powered by V_{CC}. If V_{CC} is less than V_{PF} and is less than V_{BAT}, the device is powered by V_{BAT}. See Table 1.

Table 1. Power Control

SUPPLY CONDITION	POWERED BY
$V_{CC} < V_{PF}$, $V_{CC} < V_{BAT}$	V_{BAT}
$V_{CC} < V_{PF}$, $V_{CC} > V_{BAT}$	V_{CC}
$V_{CC} > V_{PF}$, $V_{CC} < V_{BAT}$	V_{CC}
$V_{CC} > V_{PF}$, $V_{CC} > V_{BAT}$	V_{CC}

To preserve the battery, the first time V_{BAT} is applied to the device, the oscillator will not start up until V_{CC} is applied, or until a valid I²C address is written to the part. Typical oscillator startup time is less than one second. Approximately 2 seconds after V_{CC} is applied, or a valid I²C address is written, the device makes a temperature measurement and applies the calculated correction to the oscillator. Once the oscillator is running, it continues to run as long as a valid power source is available (V_{CC} or V_{BAT}), and the device continues to measure the temperature and correct the oscillator frequency every 64 seconds.

Pushbutton Reset Function

The DS3231 provides for a pushbutton switch to be connected to the $\overline{RST}$ output pin. When the DS3231 is not in a reset cycle, it continuously monitors the $\overline{RST}$ signal for a low going edge. If an edge transition is detected, the DS3231 debounces the switch by pulling the $\overline{RST}$ low. After the internal timer has expired (PB_{DB}), the DS3231 continues to monitor the $\overline{RST}$ line. If the line is still low, the DS3231 continuously monitors the line looking for a rising edge. Upon detecting release, the DS3231 forces the $\overline{RST}$ pin low and holds it low for t_{RST}.

The same pin, $\overline{RST}$, is used to indicate a power-fail condition. When V_{CC} is lower than V_{PF}, an internal power-fail signal is generated, which forces the $\overline{RST}$ pin low. When V_{CC} returns to a level above V_{PF}, the $\overline{RST}$ pin is held low for approximately 250ms (t_{REC}) to allow the power supply to stabilize. If the oscillator is not running (see the *Power Control* section) when V_{CC} is applied, t_{REC} is bypassed and $\overline{RST}$ immediately goes high.

Real-Time Clock

With the clock source from the TCXO, the RTC provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an $\overline{AM}$/PM indicator.

The clock provides two programmable time-of-day alarms and a programmable square-wave output. The $\overline{INT}$/SQW pin either generates an interrupt due to alarm condition or outputs a square-wave signal and the selection is controlled by the bit INTCN.

Address Map

Figure 1 shows the address map for the DS3231 timekeeping registers. During a multibyte access, when the address pointer reaches the end of the register space (12h), it wraps around to location 00h. On an I²C START or address pointer incrementing to location 00h, the current time is transferred to a second set of registers. The time information is read from these secondary registers, while the clock may continue to run. This eliminates the need to reread the registers in case the main registers update during a read.

I²C Interface

The I²C interface is accessible whenever either V_{CC} or V_{BAT} is at a valid level. If a microcontroller connected to the DS3231 resets because of a loss of V_{CC} or other event, it is possible that the microcontroller and DS3231 I²C communications could become unsynchronized, e.g., the microcontroller resets while reading data from the DS3231. When the microcontroller resets, the

Figure 1. Timekeeping Registers

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00H	0	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12/$\overline{24}$	$\overline{AM}$/PM 10 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03H	0	0	0	0	0	Day			Day	1–7
04H	0	0	10 Date		Date				Date	00–31
05H	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06H	10 Year				Year				Year	00–99
07H	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08H	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09H	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 10 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0AH	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0BH	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0CH	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 10 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0DH	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0EH	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0FH	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10H	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11H	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12H	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Note: *Unless otherwise specified, the registers' state is not defined when power is first applied.*

DS3231 I²C interface may be placed into a known state by toggling SCL until SDA is observed to be at a high level. At that point the microcontroller should pull SDA low while SCL is high, generating a START condition.

Clock and Calendar

The time and calendar information is obtained by reading the appropriate register bytes. Figure 1 illustrates the RTC registers. The time and calendar data are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the binary-coded decimal (BCD) format. The DS3231 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the $\overline{AM}$/PM bit with logic-high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20–23 hours). The century bit (bit 7 of the month register) is toggled when the years register overflows from 99 to 00.

The day-of-week register increments at midnight. Values that correspond to the day of week are user-defined but must be sequential (i.e., if 1 equals Sunday, then 2 equals Monday, and so on). Illogical time and date entries result in undefined operation.

When reading or writing the time and date registers, secondary (user) buffers are used to prevent errors when the internal registers update. When reading the time and date registers, the user buffers are synchronized to the internal registers on any START and when the register pointer rolls over to zero. The time information is read from these secondary registers, while the clock continues to run. This eliminates the need to reread the registers in case the main registers update during a read.

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

The countdown chain is reset whenever the seconds register is written. Write transfers occur on the acknowledge from the DS3231. Once the countdown chain is reset, to avoid rollover issues the remaining time and date registers must be written within 1 second. The 1Hz square-wave output, if enabled, transitions high 500ms after the seconds data transfer, provided the oscillator is already running.

Alarms

The DS3231 contains two time-of-day/date alarms. Alarm 1 can be set by writing to registers 07h to 0Ah. Alarm 2 can be set by writing to registers 0Bh to 0Dh. The alarms can be programmed (by the alarm enable and INTCN bits of the control register) to activate the $\overline{INT}$/SQW output on an alarm match condition. Bit 7 of each of the time-of-day/date alarm registers are mask bits (Table 2). When all the mask bits for each alarm are logic 0, an alarm only occurs when the values in the timekeeping registers match the corresponding values stored in the time-of-day/date alarm registers. The alarms can also be programmed to repeat every second, minute, hour, day, or date. Table 2 shows the possible settings. Configurations not listed in the table will result in illogical operation.

The DY/$\overline{DT}$ bits (bit 6 of the alarm day/date registers) control whether the alarm value stored in bits 0 to 5 of that register reflects the day of the week or the date of the month. If DY/$\overline{DT}$ is written to logic 0, the alarm will be the result of a match with date of the month. If DY/$\overline{DT}$ is written to logic 1, the alarm will be the result of a match with day of the week.

When the RTC register values match alarm register settings, the corresponding Alarm Flag 'A1F' or 'A2F' bit is set to logic 1. If the corresponding Alarm Interrupt Enable 'A1IE' or 'A2IE' is also set to logic 1 and the INTCN bit is set to logic 1, the alarm condition will activate the $\overline{INT}$/SQW signal. The match is tested on the once-per-second update of the time and date registers.

Table 2. Alarm Mask Bits

DY/$\overline{DT}$	ALARM 1 REGISTER MASK BITS (BIT 7)				ALARM RATE
	A1M4	A1M3	A1M2	A1M1	
X	1	1	1	1	Alarm once per second
X	1	1	1	0	Alarm when seconds match
X	1	1	0	0	Alarm when minutes and seconds match
X	1	0	0	0	Alarm when hours, minutes, and seconds match
0	0	0	0	0	Alarm when date, hours, minutes, and seconds match
1	0	0	0	0	Alarm when day, hours, minutes, and seconds match

DY/$\overline{DT}$	ALARM 2 REGISTER MASK BITS (BIT 7)			ALARM RATE
	A2M4	A2M3	A2M2	
X	1	1	1	Alarm once per minute (00 seconds of every minute)
X	1	1	0	Alarm when minutes match
X	1	0	0	Alarm when hours and minutes match
0	0	0	0	Alarm when date, hours, and minutes match
1	0	0	0	Alarm when day, hours, and minutes match

Control Register (0Eh)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE

Special-Purpose Registers

The DS3231 has two additional registers (control and status) that control the real-time clock, alarms, and square-wave output.

Control Register (0Eh)

Bit 7: Enable Oscillator ($\overline{EOSC}$). When set to logic 0, the oscillator is started. When set to logic 1, the oscillator is stopped when the DS3231 switches to V_{BAT}. This bit is clear (logic 0) when power is first applied. When the DS3231 is powered by V_{CC}, the oscillator is always on regardless of the status of the $\overline{EOSC}$ bit.

Bit 6: Battery-Backed Square-Wave Enable (BBSQW). When set to logic 1 and the DS3231 is being powered by the V_{BAT} pin, this bit enables the square-wave output when V_{CC} is absent. When BBSQW is logic 0, the $\overline{INT}$/SQW pin goes high impedance when V_{CC} falls below the power-fail trip point. This bit is disabled (logic 0) when power is first applied.

Bit 5: Convert Temperature (CONV). Setting this bit to 1 forces the temperature sensor to convert the temperature into digital code and execute the TCXO algorithm to update the capacitance array to the oscillator. This can only happen during the idle period. The status bit, BSY, prevents the bit from being set when BSY = 1. The user should check the status bit BSY before forcing the controller to start a new TCXO execution. A user-initiated temperature conversion does not affect the internal 64-second update cycle.

A user-initiated temperature conversion does not affect the BSY bit for approximately 2ms. The CONV bit remains at a 1 from the time it is written until the conversion is finished, at which time both CONV and BSY go to 0. The CONV bit should be used when monitoring the status of a user-initiated conversion.

Bits 4 and 3: Rate Select (RS2 and RS1). These bits control the frequency of the square-wave output when the square wave has been enabled. The following table shows the square-wave frequencies that can be selected with the RS bits. These bits are both set to logic 1 (8.192kHz) when power is first applied.

SQUARE-WAVE OUTPUT FREQUENCY

RS2	RS1	SQUARE-WAVE OUTPUT FREQUENCY
0	0	1Hz
0	1	1.024kHz
1	0	4.096kHz
1	1	8.192kHz

Bit 2: Interrupt Control (INTCN). This bit controls the $\overline{INT}$/SQW signal. When the INTCN bit is set to logic 0, a square wave is output on the $\overline{INT}$/SQW pin. When the INTCN bit is set to logic 1, then a match between the timekeeping registers and either of the alarm registers activates the $\overline{INT}$/SQW (if the alarm is also enabled). The corresponding alarm flag is always set regardless of the state of the INTCN bit. The INTCN bit is set to logic 1 when power is first applied.

Bit 1: Alarm 2 Interrupt Enable (A2IE). When set to logic 1, this bit permits the alarm 2 flag (A2F) bit in the status register to assert $\overline{INT}$/SQW (when INTCN = 1). When the A2IE bit is set to logic 0 or INTCN is set to logic 0, the A2F bit does not initiate an interrupt signal. The A2IE bit is disabled (logic 0) when power is first applied.

Bit 0: Alarm 1 Interrupt Enable (A1IE). When set to logic 1, this bit permits the alarm 1 flag (A1F) bit in the status register to assert $\overline{INT}$/SQW (when INTCN = 1). When the A1IE bit is set to logic 0 or INTCN is set to logic 0, the A1F bit does not initiate the $\overline{INT}$/SQW signal. The A1IE bit is disabled (logic 0) when power is first applied.

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

Status Register (0Fh)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OSF	0	0	0	EN32kHz	BSY	A2F	A1F

Status Register (0Fh)

Bit 7: Oscillator Stop Flag (OSF). A logic 1 in this bit indicates that the oscillator either is stopped or was stopped for some period and may be used to judge the validity of the timekeeping data. This bit is set to logic 1 any time that the oscillator stops. The following are examples of conditions that can cause the OSF bit to be set:

1) The first time power is applied.
2) The voltages present on both V_{CC} and V_{BAT} are insufficient to support oscillation.
3) The $\overline{EOSC}$ bit is turned off in battery-backed mode.
4) External influences on the crystal (i.e., noise, leakage, etc.).

This bit remains at logic 1 until written to logic 0.

Bit 3: Enable 32kHz Output (EN32kHz). This bit indicates the status of the 32kHz pin. When set to logic 1, the 32kHz pin is enabled and outputs a 32.768kHz square-wave signal. When set to logic 0, the 32kHz pin goes to a high-impedance state. The initial power-up state of this bit is logic 1, and a 32.768kHz square-wave signal appears at the 32kHz pin after a power source is applied to the DS3231 (if the oscillator is running).

Bit 2: Busy (BSY). This bit indicates the device is busy executing TCXO functions. It goes to logic 1 when the conversion signal to the temperature sensor is asserted and then is cleared when the device is in the 1-minute idle state.

Bit 1: Alarm 2 Flag (A2F). A logic 1 in the alarm 2 flag bit indicates that the time matched the alarm 2 registers. If the A2IE bit is logic 1 and the INTCN bit is set to logic 1, the $\overline{INT}$/SQW pin is also asserted. A2F is cleared when written to logic 0. This bit can only be written to logic 0. Attempting to write to logic 1 leaves the value unchanged.

Bit 0: Alarm 1 Flag (A1F). A logic 1 in the alarm 1 flag bit indicates that the time matched the alarm 1 registers. If the A1IE bit is logic 1 and the INTCN bit is set to logic 1, the $\overline{INT}$/SQW pin is also asserted. A1F is cleared when written to logic 0. This bit can only be written to logic 0. Attempting to write to logic 1 leaves the value unchanged.

Crystal Aging

The crystal aging offset register provides an 8-bit code to add to the codes in the capacitance array registers. The code is encoded in two's complement. One LSB represents one small capacitor to be switched in or out of the capacitance array at the crystal pins. The offset register is added to the capacitance array register under the following conditions: during a normal temperature conversion, if the temperature changes from the previous conversion, or during a manual user conversion (setting the CONV bit). To see the effects of the aging register on the 32kHz output frequency immediately, a manual conversion should be started after each aging register change.

Positive aging values add capacitance to the array, slowing the oscillator frequency. Negative values remove capacitance from the array, increasing the oscillator frequency.

The change in ppm per LSB is different at different temperatures. The frequency vs. temperature curve is shifted by the values used in this register. At +25°C, one LSB typically provides about 0.1ppm change in frequency.

Crystal Aging Offset (10h)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Sign	Data	Data	Data	Data	Data	Data	Data

Temperature Register (Upper Byte) (11h)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Sign	Data	Data	Data	Data	Data	Data	Data

Temperature Register (Lower Byte) (12h)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Data	Data	0	0	0	0	0	0

Temperature Registers (11h–12h)

Temperature is represented as a 10-bit code with a resolution of +0.25°C and is accessible at location 11h and 12h. The temperature is encoded in two's complement format. The upper 8 bits are at location 11h and the lower 2 bits are in the upper nibble at location 12h. Upon power reset, the registers are set to a default temperature of 0°C and the controller starts a temperature conversion. New temperature readings are stored in this register.

I²C Serial Data Bus

The DS3231 supports a bidirectional I²C bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data is defined as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are slaves. The bus must be controlled by a master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS3231 operates as a slave on the I²C bus. Connections to the bus are made through the SCL input and open-drain SDA I/O lines. Within the bus specifications, a standard mode (100kHz maximum clock rate) and a fast mode (400kHz maximum clock rate) are defined. The DS3231 works in both modes.

The following bus protocol has been defined (Figure 2):

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is high. Changes in the data line while the clock line is high are interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain high.

Start data transfer: A change in the state of the data line from high to low, while the clock line is high, defines a START condition.

Stop data transfer: A change in the state of the data line from low to high, while the clock line is high, defines a STOP condition.

Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the high period of the clock signal. The data on the line must be changed during the low period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between the START and the STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse, which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable low during the high period of the acknowledge-related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line high to enable the master to generate the STOP condition.

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

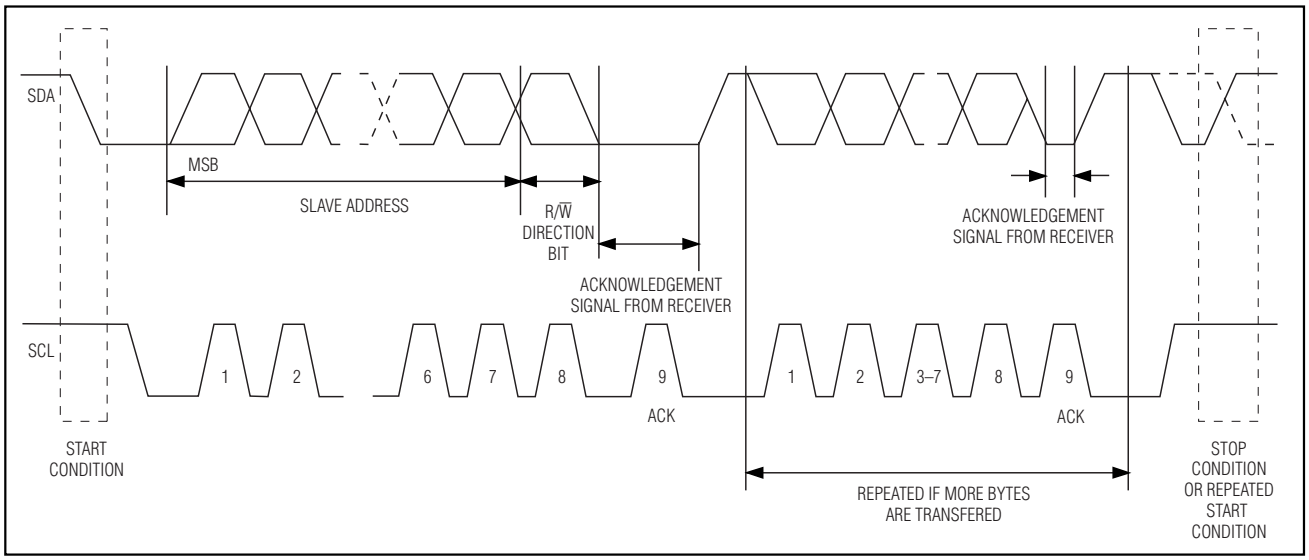


Figure 2. I²C Data Transfer Overview

Figures 3 and 4 detail how data transfer is accomplished on the I²C bus. Depending upon the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows a number of data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a not acknowledge is returned.

The master device generates all the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

The DS3231 can operate in the following two modes:

Slave receiver mode (DS3231 write mode): Serial data and clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer.

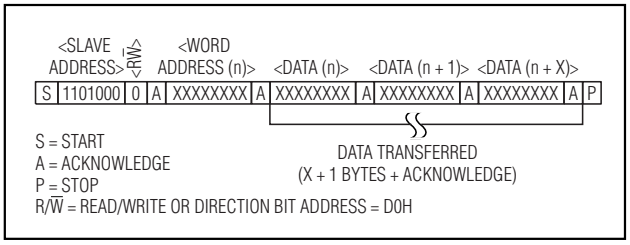


Figure 3. Slave Receiver Mode (Write Mode)

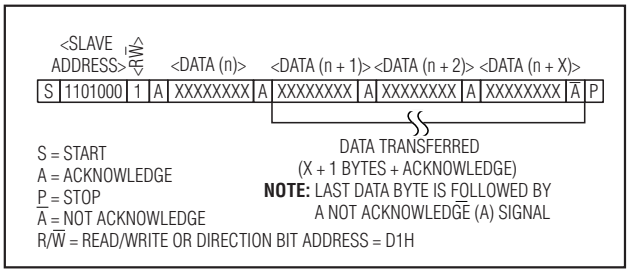


Figure 4. Slave Transmitter Mode (Read Mode)

Address recognition is performed by hardware after reception of the slave address and direction bit. The slave address byte is the first byte received after the master generates the START condition. The slave address byte contains the 7-bit DS3231 address, which is 1101000, followed by the direction bit (R/$\overline{W}$), which is 0 for a write. After receiving and decoding the slave address byte, the DS3231 outputs an

acknowledge on SDA. After the DS3231 acknowledges the slave address + write bit, the master transmits a word address to the DS3231. This sets the register pointer on the DS3231, with the DS3231 acknowledging the transfer. The master may then transmit zero or more bytes of data, with the DS3231 acknowledging each byte received. The register pointer increments after each data byte is transferred. The master generates a STOP condition to terminate the data write.

Slave transmitter mode (DS3231 read mode): The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit indicates that the transfer direction is reversed. Serial data is transmitted on SDA by the DS3231 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit. The slave address byte is the first byte received after the master generates a START condition. The slave address byte contains the 7-bit DS3231 address, which is 1101000, followed by the direction bit (R/$\overline{W}$), which is 1 for a read. After receiving and decoding the slave address byte, the DS3231 outputs an acknowledge on SDA. The DS3231 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode, the first address that is read is the last one stored in the register pointer. The DS3231 must receive a not acknowledge to end a read.

Handling, PC Board Layout, and Assembly

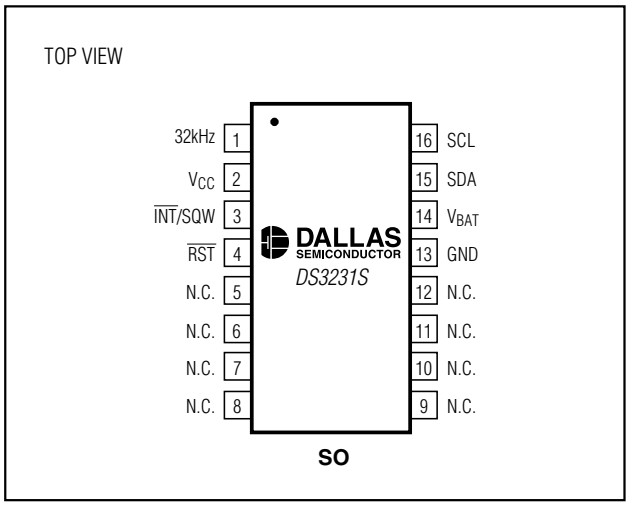
The DS3231 package contains a quartz tuning-fork crystal. Pick-and-place equipment can be used, but precautions should be taken to ensure that excessive shocks are avoided. Ultrasonic cleaning should be avoided to prevent damage to the crystal.

Avoid running signal traces under the package, unless a ground plane is placed between the package and the signal line. All N.C. (no connect) pins must be connected to ground.

Moisture-sensitive packages are shipped from the factory dry packed. Handling instructions listed on the package label must be followed to prevent damage during reflow. See IPC/JEDEC J-STD-020 standard for moisture-sensitive device (MSD) classifications and reflow profiles. Exposure to reflow is limited to 2 times maximum.

Extremely Accurate I²C-Integrated RTC/TCXO/Crystal

Pin Configuration



Chip Information

TRANSISTOR COUNT: 33,000

SUBSTRATE CONNECTED TO GROUND

PROCESS: CMOS

Thermal Information

Theta-JA: +73°C/W

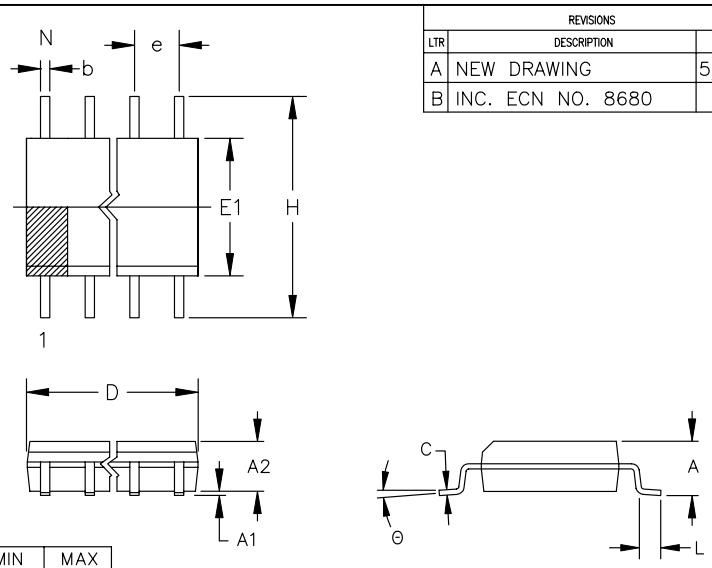
Theta-JC: +23°C/W

Package Information

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information, go to **www.maxim-ic.com/DallasPackInfo**).

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
A	NEW DRAWING	5/18	J.W.
B	INC. ECN NO. 8680		

56-G4009-001.EPS



LTR		MIN	MAX
A	IN.	0.094	0.105
	MM	2.39	2.67
A1	IN.	0.004	0.012
	MM	0.102	0.30
A2	IN.	0.089	0.095
	MM	2.26	2.41
b	IN.	0.013	0.020
	MM	0.33	0.51
C	IN.	0.009	0.013
	MM	0.229	0.33
e	IN.	.050 BSC	
	MM	1.27 BSC	
E1	IN.	0.290	0.300
	MM	7.37	7.62
H	IN.	0.398	0.416
	MM	10.11	10.57
L	IN.	0.016	0.040
	MM	0.40	1.02
Θ		0°	8°

		16 PIN		18 PIN		20 PIN		24 PIN		28 PIN	
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX
D	IN.	0.398	0.412	0.448	0.462	0.498	0.511	0.598	0.612	0.698	0.712
	MM	10.11	10.46	11.38	11.73	12.65	12.99	15.19	15.54	17.73	18.08

THE CHAMFER ON THE BODY IS OPTIONAL.
IF IT IS NOT PRESENT, A TERMINAL 1 IDENTIFIER MUST BE POSITIONED SO THAT 1/2 OR MORE OF IT'S AREA IS CONTAINED IN THE HATCHED ZONE.

SIGNATURE		DATE
DOC. CONTROL:	J.WILKINS	5/94
ENGR. MGR:	B.W.MCARTY	5/94
MFG. ENGR:	C.M.SELLS	5/94
CHECKED BY:	C.M.SELLS	5/94
DRAWN BY:	M.W.C.	5/94

DALLAS SEMICONDUCTOR MAXIM

TITLE: PACKAGE OUTLINE .300" SOIC 16,18,20,24&28 LD.

SIZE	FSCM NO	PART NO.	REV
A		56-G4009-001	B

DO NOT SCALE DWG. | SCALE N/A | SHEET 1 OF 1

DATASHEET
BLUETOOTH TO SERIAL PORT MODULE
HC05

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature). It has the
footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

www.electronica60norte.com
electronica60norte@hotmail.com

Specifications

Hardware features

- Typical -80dBm sensitivity.
- Up to +4dBm RF transmit power.
- Low Power 1.8V Operation, 3.3 to 5 V I/O.
- PIO control.
- UART interface with programmable baud rate.
- With integrated antenna.
- With edge connector.

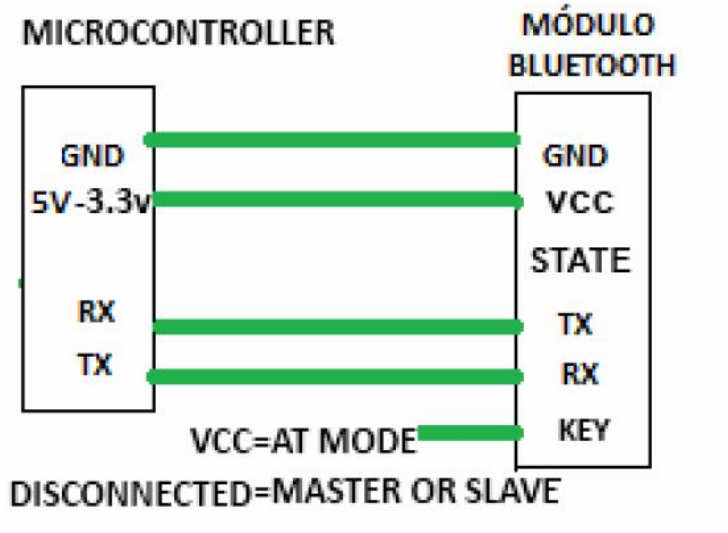
Software features

- Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No parity.
- PIO9 and PIO8 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing **PINCODE:"1234"** as default.
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

www.electronica60norte.com
electronica60norte@hotmail.com

Pin out configuration

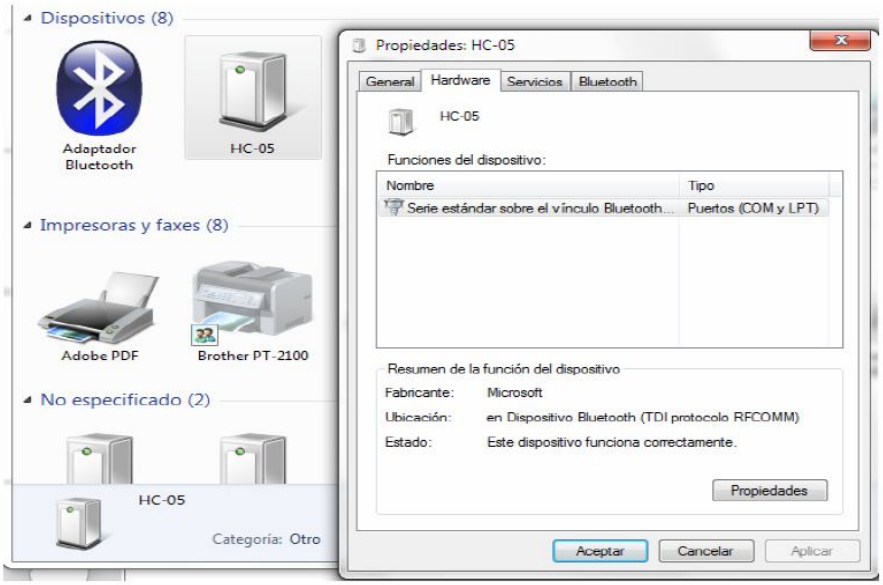
Typical Application Circuit



www.electronica60norte.com
electronica60norte@hotmail.com

After connect the Bluetooth module, scan for new devices from the PC and you will find the module with the device name “HC-05”, after that, click to connect, if some message appears asking about “Pairing code” just put **“1234”** as default code.

BLUE LED = ACTIVE (Blinking 500ms period inactive connection, change 1seg with active connection)



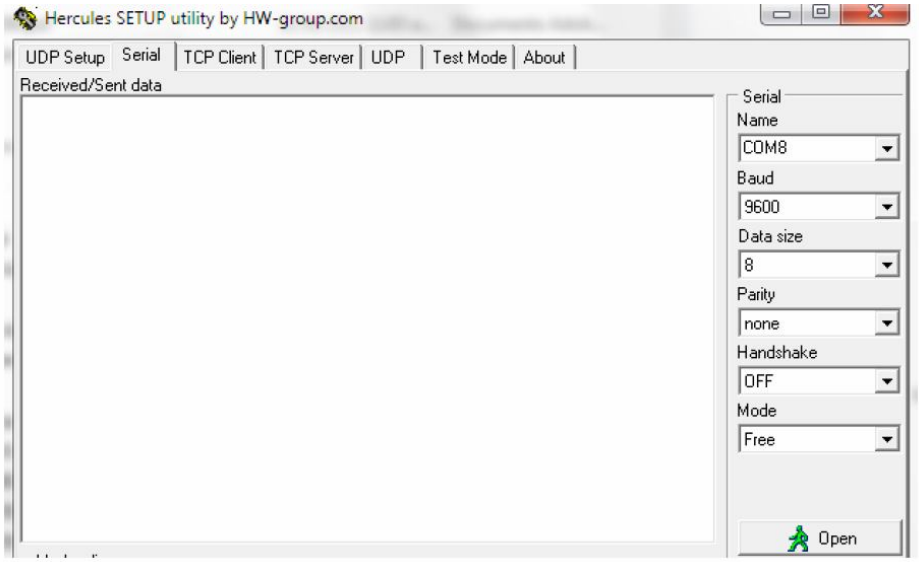
Open a serial terminal and select the serial COM x port number that assigned Windows to Bluetooth Module.

Configure the serial terminal with these parameters:

- Baud rate: 9600.
- Data bits:8.
- Stop bit:1.
- Parity: No parity.

www.electronica60norte.com
electronica60norte@hotmail.com

Open connection and you will be ready to send and receive data from module Bluetooth like Serial Port COM



AT COMMANDS

How to get to AT COMMAND mode

1: Connect KEY pin to VCC.
2: Supply power to module. Then the module will enter into AT MODE. In this mode you have to use baud rate at 38400. In this way, user should change the baud rate for SLAVE AND MASTER mode.

How to set this module as "Master - Host" role

1: Input high level to KEY.
2: Supply power to the module. And the module will enter to AT COMMAND.
3: Set the parameters of the hyper terminal or the other serial tools (baud rate: 38400, data bit:8, stop bit:1, no parity bit, no Flow Control).
4: Sent the characters "AT+ROLE=1\r\n" through serial, then receive the characters "OK\r\n". Here, "\r\n" is the CRLF.
5: Sent the characters "AT+CMODE=1\r\n" through serial, then receive the characters "OK\r\n". Here, "\r\n" is the CRLF.
6: Default factory password passkey is: 1243, this must be the same in the Bluetooth slave module if you want to pair it.
To read passkey use this command: "AT+PSWD?".
To Reset the password command sent the characters "AT+PSWD=XXXX".
The password must be 4-bits.

7: Leave free KEY, and supply power to the module again. Then this module will become master role and search the other module (slave role) automatically to build the connection (baud rate:9600, data bit:8, stop bit:1, no parity bit, no Flow Control).

How to set this module be the "Slave - Device" role

1: Input high level to KEY.
2: Supply power to the module. And the module will enter to AT COMMAND.
3: Set the parameters of the super terminal or the other serial tools (baud rate:
38400, data bit:8, stop bit:1, no parity bit, no Flow Control).
4: Sent the characters "AT+ROLE=0\r\n" through serial, then receive the characters "OK\r\n". Here, "\r\n" is the CRLF.
5: Sent the characters "AT+CMODE=0\r\n" through serial, then receive the characters "OK\r\n". Here, "\r\n" is the CRLF.
6: Default factory password passkey is: 1243, this must be the same in the Bluetooth master module if you want to pair it.
To read passkey sent the characters "AT+PSWD?".
To Reset the password command sent the characters "AT+PSWD=XXXX".
The password must be 4-bits.
7: Leave free KEY, and supply power to the module again. Then this module will become slave role and wait to be discover it by the other module (master role) automatically to build the connection (baud rate:38400, data bit:8, stop bit:1, no parity bit, no Flow Control).

How to get to the standard communication mode

1: Leave free KEY, don't connect it to VDD neither GND.
2: Supply power to the module. Then the module will enter to communication mode. It can be used for pairing.

Notes

(1) HC-05's command should end up with "\r\n". It means when you finish programming, you should add terminator ("ENTER" or "0x0d 0x0a") to the program.
(2) The most common commands for HC-05 are: AT+ROLE (set master–slave), AT+CMODE(set address pairing) , AT+PSWD (set password).
If you want the master module has the function of remembering slave module,the most simply way is: First, set AT+CMODE=1. Make the master module pair with the slave module. Second, set AT+CMODE=0. Then the master module just can make pair with that specified slave module.

A.3. Bibliografia utilizada

A continuación se detallan los libros y hojas de datos y de aplicación que fueron consultadas en este proyecto

Referencias

[1] Hoja de datos del microcontrolador ATmega328P. Disponible en: http://www.atmel.com/Images/Atmel-42397-8-bit-AVR-Microcontroller-ATmega328PB_Datasheet.pdf

[2] Set de instrucciones de los microcontroladores de 8 bits ATmel AVR. Disponible en: http://campus.fi.uba.ar/mod/resource/view.php?id=56142

[3] Muhammad Ali Mazidi, Sarmad Naimi, Sepehr Naimi.The avr microcontroller and embeddeed system.Using assembly and C, first edition. Pearson Custom Electronics Technology

A.4. Notas de aplicación de Atmel utilizadas

AVR034: Mixing C and Assembly Code with IAR Embedded Workbench for AVR

8-bit AVR® Microcontroller

Application Note

Features

- **Passing Variables Between C and Assembly Code Functions**
- **Calling Assembly Code Functions from C**
- **Calling C Functions from Assembly Code**
- **Writing Interrupt Functions in Assembly Code**
- **Accessing Global Variables in Assembly Code**

This application note describes how to use C to control the program flow and main program and assembly modules to control time critical I/O functions.

Introduction

This application note describes how to set up and use the IAR C-compiler for the AVR controller in projects including both C and Assembly code. By mixing C and Assembly designers can combine the powerful C language instructions with the effective hardware-near assembly code instructions.

Table 1. The Pluses and Minuses of C and Assembly

Assembly	C
+ Full control of Resource Usage	+ Efficient code in larger applications
+ Compact/fast code in small applications	+ Structured code
- Inefficient code in larger applications	+ Easy to maintain
- Cryptic code	+ Portable
- Hard to maintain	- Limited control of Resource Usage
- Non-portable	- Larger/slower code in small applications

This information in this document is based on the calling conventions of the IAR ICCA90 compiler. Later versions of the IAR compiler may have a different calling convention. However, the basics are the similar, but refer to the Compiler's Reference Guide for the latest information about calling conventions.

Rev. 1234B–AVR–04/03

Passing Variables Between C and Assembly Code Functions

When the IAR C-compiler is used for the AVR the Register File is segmented as shown in Figure 1.

Scratch Registers are not preserved across functions calls. Local registers are preserved across function calls. The Y Register (R28:R29) is used as Data Stack Pointer to SRAM. The Scratch Registers are used to passing parameters and return values between functions.

When a function is called the parameters to be passed to the function is placed in the Register File Registers R16-R23. When a function is returning a value this value is placed in the Register File Registers R16-R19, depending on the size of the parameters and the returned value.

Table 2 shows example placement of parameter when calling a function:

Figure 1. Segments in the Register File

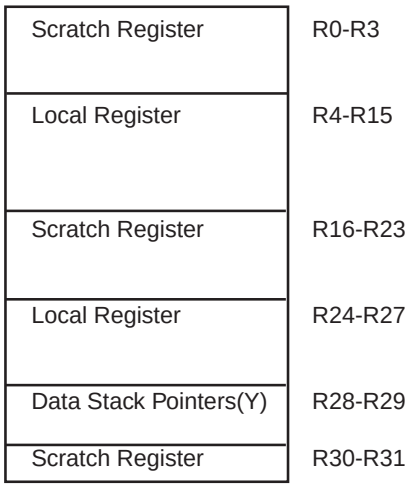


Table 2. Placement and Parameters to C-functions

Function	Parameter 1 Registers	Parameter 2 Registers
func (char ,char)	R16	R20
func (char ,int)	R16	R20, R21
func (int ,long)	R16 ,R17	R20, R21, R22, R23
func (long ,long)	R16, R17, R18, R19	R20, R21, R22, R23

For complete reference of the supported data types and corresponding sizes, see the IAR AT90S Users Guide, Data Representation section.

Example C function call:

```
int get_port(unsigned char temp, int num)
```

When calling this C function the one byte parameter ***temp*** is placed in R16, the two byte parameter ***num*** is placed in R20:R21. The function returns a two byte value which is placed in R16:R17 after return from the function.

If a function is called with more than two parameters the first two parameters are passed to the function as shown above, the remaining parameters are passed to the function on the Data Stack. If a function is called with a ***struct*** or ***union*** as parameter a pointer to the structure is passed on to the function on the Data Stack.

If a function need to use any local registers it first pushes the registers on the Data Stack. Then return value from the function is placed at adresses R16-R19 depending on the size of the returned value.

Example 1

Calling Assembly Code Functions from a C Program

- with no parameters and no return value

Example C Code for Calling Assembly Code Function

```
#include "io8515.h"
extern void get_port(void);/* Function prototype for asm function */
void main(void)
{
  DDRD = 0x00;/* Initialization of the I/O ports*/
  DDRB = 0xFF;
  while(1)/* Infinite loop*/
{
    get_port();/* Call the assembler function   */
  }
}
```

The Called Assembly Code Function

```
NAME get_port
    #include "io8515.h"     ; The #include file must be within the module
    PUBLIC get_port         ; Declare symbols to be exported to C function
    RSEG CODE               ; This code is relocatable, RSEG

get_port;                   ; Label, start execution here
    in R16,PIND             ; Read in the pind value
    swap R16                ; Swap the upper and lower nibble
    out PORTB,R16           ; Output the data to the port register
    ret                     ; Return to the main function
END
```

Calling Assembly Code Functions from a C Function

-passing parameters and returning values.

This example C function is calling an assembler function. The one byte ***mask*** is passed as a parameter to the assembly function, ***mask*** is placed in R16 before the function call. The assembly function is returning a value in R16 to the C variable ***value***.

```
#include "io8515.h"
char get_port(char mask);       /*Function prototype for asm function */
void C_task main(void)
{
    DDRB=0xFF
    while(1)                    /* Infinite loop*/
    {
        char value, temp;       /* Decalre local variables*/
        temp = 0x0F;
        value = get_port(temp); /* Call the assembler function */
        if(value==0x01)
        {
         /* Do something if value is 0x01        */
    PORTB=~(PORTB);             /* Invert value on Port B  */
        }
    }
}
```

The Called Assembly Code Function

```
NAME get_port
    #include "io8515.h"   ; The #include file must be within the module
    PUBLIC get_port       ; Symbols to be exported to C function

RSEG CODE                 ; This code is relocatable, RSEG
get_port:                 ; Label, start execution here
      in   R17,PIND       ; Read in the pinb value
      eor  R16,R17        ; XOR value with mask(in R16) from main()
      swap R16            ; Swap the upper and lower nibble
      rol  R16            ; Rotate R16 to the left
      brcc ret0           ; Jump if the carry flag is cleared
      ldi  r16,0x01       ; Load 1 into R16, return value
      ret                 ; Return
ret0: clr  R16            ; Load 0 into R16, return value
ret                       ; Return
END
```

Calling C Functions from Assembly Code

Assuming that the assembly function calls the standard C library routine rand() to get a random number to output to the port. The rand() routine returns an integer value(16 bits). This example writes only the lower byte/8bits to a port.

```
NAME get_port
    #include "io8515.h"     ; The #include file must be within the module
    EXTERN rand, max_val    ; External symbols used in the function
    PUBLIC get_port         ; Symbols to be exported to C function

    RSEG CODE               ; This code is relocatable, RSEG

get_port:                   ; Label, start execution here
    clr     R16             ; Clear R16
    sbis    PIND,0          ; Test if PIND0 is 0
    rcall   rand            ; Call RAND() if PIND0 = 0
    out     PORTB,R16       ; Output random value to PORTB
    lds     R17,max_val     ; Load the global variable max_val
    cp      R17,R16         ; Check if number higher than max_val
    brlt    nostore         ; Skip if not
    sts     max_val,R16     ; Store the new number if it is higher
nostore:
    ret                     ; Return
END
```

Writing Interrupt Functions in Assembly.

Interrupt functions can be written in assembly. Interrupt functions can not have any parameters nor returning any value. Because an interrupt can occur anywhere in the program execution it needs to store all used registers on the stack.

Care must be taken when assembler code is placed at the interrupt vector adresses to avoid problems with the interrupt functions in C.

Example Code Placed at Interrupt Vector

```
NAME EXT_INT1
#include "io8515.h"

extern c_int1
COMMON INTVEC(1)            ; Code in interrupt vector segment
ORG INT1_vect               ; Place code at interrupt vector
    RJMP   c_int1           ; Jump to assembler interrupt function
ENDMOD

                          ;The interrupt vector code performs a jump to the
function c_int1:

NAME c_int1
    #include "io8515.h"
PUBLIC c_int1               ; Symbols to be exported to C function

    RSEG CODE               ; This code is relocatable, RSEG
c_int1:
    st     -Y,R16           ; Push used registers on stack
    in     R16,SREG         ; Read status register
    st     -Y,R16           ; Push Status register
```

```
    in      R16,PIND          ; Load in value from port D

    com     R16               ; Invert it

    out     PORTB,R16         ; Output inverted value to port B

    ld      R16,Y+            ; Pop status register
    out     SREG,R16          ; Store status register
    ld      R16,Y+            ; Pop Register R16
    reti
END
```

Accessing Global Variables in Assembly

The main program introduces a global variable called **max_val**. To access this variable in assembly the variable must be declared as **EXTERN max_val**. To access the variable the assembly function uses LDS (Load Direct from SRAM) and STS (STore Direct to SRAM) intructions.

```
#include "io8515.h"

char max_val;
void get_port(void);        /* Function prototype for assembler function */

void C_task main(void)
{
    DDRB = 0xFF;            /* Set port B as output */
    while(1)                /* Infinite loop */
    {
        get_port();         /* Call assembly code function */
    }
}
```

```
NAME get_port
    #include "io8515.h"     ; The #include file must be within the module
    EXTERN rand, max_val    ; External symbols used in the function
    PUBLIC get_port         ; Symbols to be exported to C function

    RSEG CODE               ; This code is relocatable, RSEG

get_port:                   ; Label, start execution here
    clr     R16             ; Clear R16
    sbis    PIND,0          ; Test if PIND0 is 0
    rcall   rand            ; Call RAND() if PIND0 = 0
    out     PORTB,R16       ; Output random value to PORTB
    lds     R17,max_val     ; Load the global variable max_val
    cp      R17,R16         ; Check if number higher than max_val
    brlt    nostore         ; Skip if not
    sts     max_val,R16     ; Store the new number if it is higher
nostore:
    ret                     ; Return
END
```

References

IAR Systems AT09S USER GUIDE

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

1234B–AVR–04/03 0M

AVR 300: Software TWI Master Interface

8-bit AVR® Microcontroller

Application Note

Features

- **Uses No Interrupts**
- **Supports Normal and Fast Mode**
- **Supports Both 7-bit and 10-bit Addressing**
- **Supports the Entire AVR Microcontroller Family**

Introduction

The need for a simple and cost effective inter-IC bus for use in consumer, telecommunications and industrial electronics, led to the developing of the TWI bus. Today the TWI bus is implemented in a large number of peripheral and microcontrollers, making it a good choice in low speed applications.

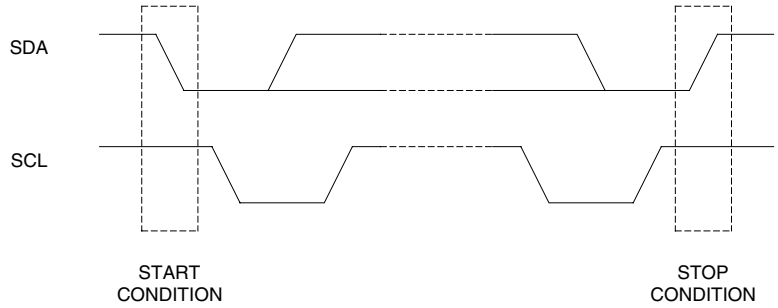
The AVR microcontroller family does not have dedicated hardware for TWI operation, but because of the flexible I/O and high processing speed, an efficient software TWI single master interface, can easily be implemented.

Theory of Operation

The TWI bus is a Two-wire synchronous serial interface consisting of one data (SDA) and one clock (SCL) line. By using open drain/collector outputs, the TWI bus supports any fabrication process (CMOS, bipolar and more).

The TWI bus is a multi-master bus where one or more devices, capable of taking control of the bus, can be connected. When there is only one master connected to the bus, this does not need to support handling of bus contentions and inter master access (a master accessing another master). Only master devices can drive both the SCL and SDA lines while a slave device is only allowed to issue data on the SDA line.

Figure 1. START and STOP Conditions

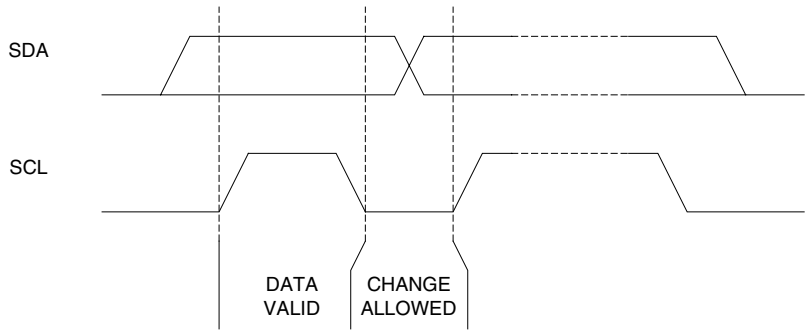


Rev. 0954B–AVR–05/02

Data transfer is always initiated by a Bus Master device. A **high** to **low** transition on the SDA line while SCL is high is defined to be a START condition (or a repeated start condition). A START condition is always followed by the (unique) 7-bit slave address and then by a data direction bit. The Slave device addressed now acknowledges to the Master by holding SDA low for one clock cycle. If the Master does not receives any acknowledge, the transfer is terminated. Depending of the data direction bit, the Master or Slave now transmits 8-bit of data on the SDA line. The receiving device then acknowledges the data. Multiple bytes can be transferred in one direction before a repeated START or a STOP condition is issued by the Master. The transfer is terminated when the Master issues a STOP condition. A STOP condition is defined by a **low** to **high** transition on the SDA line while the SCL is high.

If a Slave device cannot handle incoming data until it has performed some other function, it can hold SCL low to force the master into a Wait State.

Figure 2. Bit Transfer on the TWI Bus



Change of data on the SDA line is only allowed during the low period of SCL as shown in Figure 2. This is a direct consequence of the definition of the START and STOP conditions. A more detailed description and timing specifications, can be found in [1].

Transferring Data

All transfer on the bus is byte sized. Each byte is followed by an acknowledge bit set by the Receiver. The slave address byte contains both a 7-bit address and a read/write bit.

Figure 3. Byte Transfer Formats

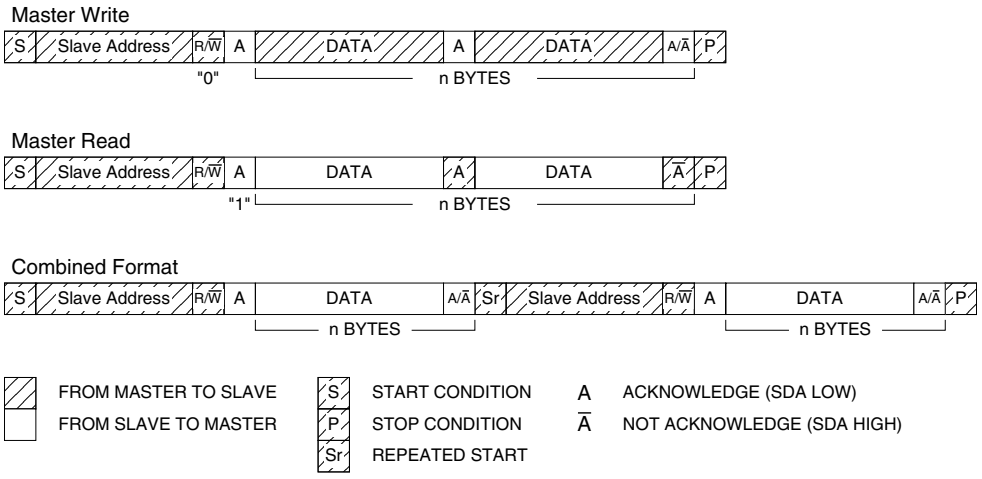


Figure 3 shows the valid data transfer formats. In the combined format, multiple data can be sent in any direction (to the same Slave device). A change in data direction is done by using a *repeated* START condition. Note that a Master read operation must be terminated by not acknowledging the last byte read.

Connection

Both TWI lines (SDA and SCL) are bi-directional therefore outputs must be of an open-drain or an open-collector type. Each line must be connected to the supply voltage via a pull-up resistor. A line is then logic high when none of the connected devices drives the line, and logic low if one or more is drives the line low.

Figure 4. Physical Connection to the TWI Bus

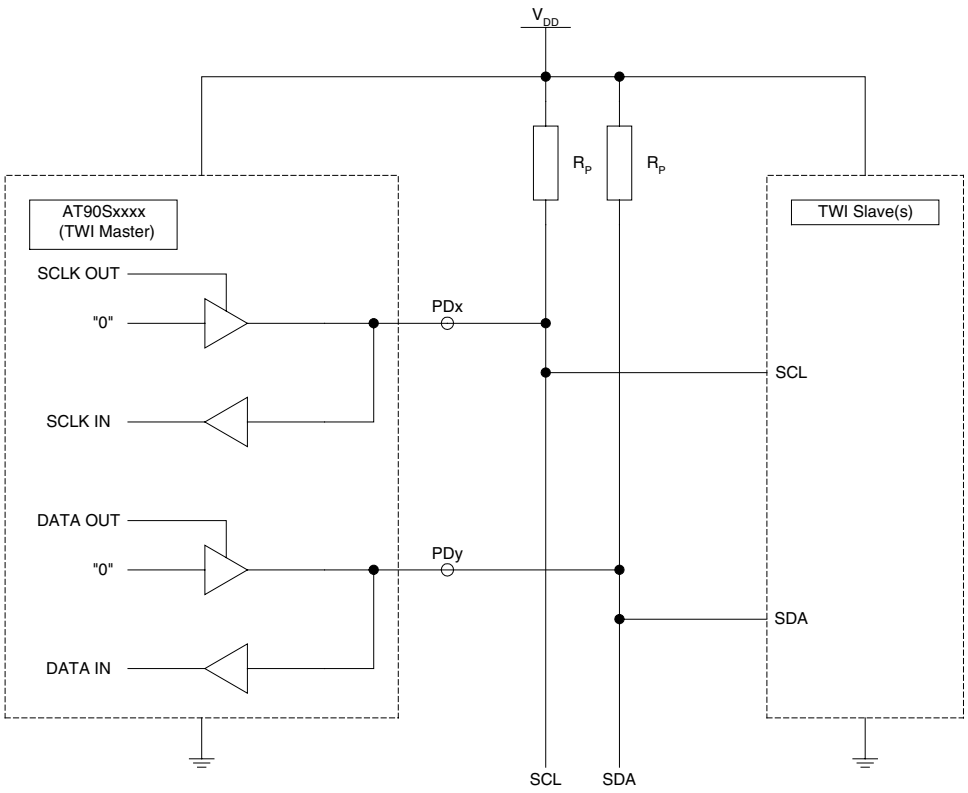


Figure 4 shows how to connect the AVR microcontroller to the TWI bus. The value of R_P depends on V_{DD} and the bus capacitance (typically 4.7 k).

Implementation

The only resources used by the TWI Master routines presented in this application, is the two pins for SCL and SDA on port D. Since the TWI bus is synchronous, the duty cycle and the period time to the Serial Clock Line (SCL) is not critical. Therefore it is not necessary to “fine-tune” the routines which would cause to an increase of program size.

There are two types of delays used in this implementation: quarter period and half period delays. For TWI in Normal mode (100 kHz), these delays must be $t_{quarter} > 2.5$ µs and $t_{half} > 5.0$ µs. For TWI in Fast mode (400 kHz) the parameters are $t_{quarter} > 0.6$ µs and $t_{half} > 1.3$ µs.

There is a large number of possible implementations of the delay loop. All the implementations depends on the MCU clock frequency. It is not possible to make a generalized version which is efficient for all clock speeds.

The following steps show how to choose the most efficient implementation of the delays:

1. Calculate the required number of clock cycles both delays:
   $n = t * f_{osc}$; $t = t_{quarter}$ and t_{half}, f_{osc} is MCU clock.
2. Use n to chose one of the following methods for both delays.

n	Delay method
< 1	Remove all calls to the delay routine
1 < n < 2	Replace all calls to the delay routine with one “nop” instruction
2 < n < 3	Replace all calls to the delay routine with an “rjmp 1” instruction
2 < n < 7	The delay routine should consist of one “ret” instruction only

> 7	Use the following routine :
	`ldi TWIdelay, 1+ (n-7)/3`
	`loop: dec TWIdelay`
	`brne loop`
	`ret`
	`(this routine is used in the program code!)`

TWI Subroutines

“TWI_init”

Initializes SCL and SDA lines. The SCLP and SDAP constants located top of the program code, chooses the pin number on port D. It is possible to use any pins on any port by changing the program code if required.

All the port D initialization can be put in this subroutine to reduce code size.

“TWI_start”

Generate start condition and sends slave address. All data transfer must start with this subroutine. When a transfer is done the “TWI_end” must be called. *When the bus is free (after “TWI”_end is called) all registers are free for other usage.*

Parameter	Value
Code Size	3
Execution Cycles	N/A
Register Usage	• Low Registers :None • High Registers :3 • Global Registers :1

Register	Input	Internal	Output
r16		“TWIdelay” – Delay Loop Counter	
r17		“TWIdata” – Transmit buffer	
r18	“TWIadr” – Slave address and transfer direction (global)		

"TWI_rep_start"

Generate repeated start condition and sends slave address. A repeated START can only be given after a byte has been read or written.

Parameter	Value
Code Size	5
Execution Cycles	N/A
Register Usage	• Low Registers :None • High Registers :3 • Global Registers :1

Register	Input	Internal	Output
r16		"TWIdelay" – Delay Loop Counter	
r17		"TWIdata" – Transmit buffer	
r18	"TWIadr" – Slave address and transfer direction (global)		

"TWI_write"

Writes data (one byte) to the TWI bus. This function is also used for sending the address.

Parameter	Value
Code Size	16
Execution Cycles	N/A
Register Usage	• Low Registers :None • High Registers :2 • Global Registers :None

Register	Input	Internal	Output
r16		"TWIdelay" – Delay Loop Counter	
r17	"TWIdata" – Data to be written		

"TWI_get_ack"

Get slave acknowledge response. The reason for separate this subroutine from the "TWI_write" routine is to get a more readable program code.

Parameter	Value
Code Size	11
Execution Cycles	N/A
Register Usage	• Low Registers :None • High Registers :1 • Global Registers :None

Register	Input	Internal	Output
r16		"TWIdelay" – Delay Loop Counter	

Figure 5. “TWI_start”, “TWI_rep_start”, “TWI_write”, and “TWI_get_ack” Flow Chart

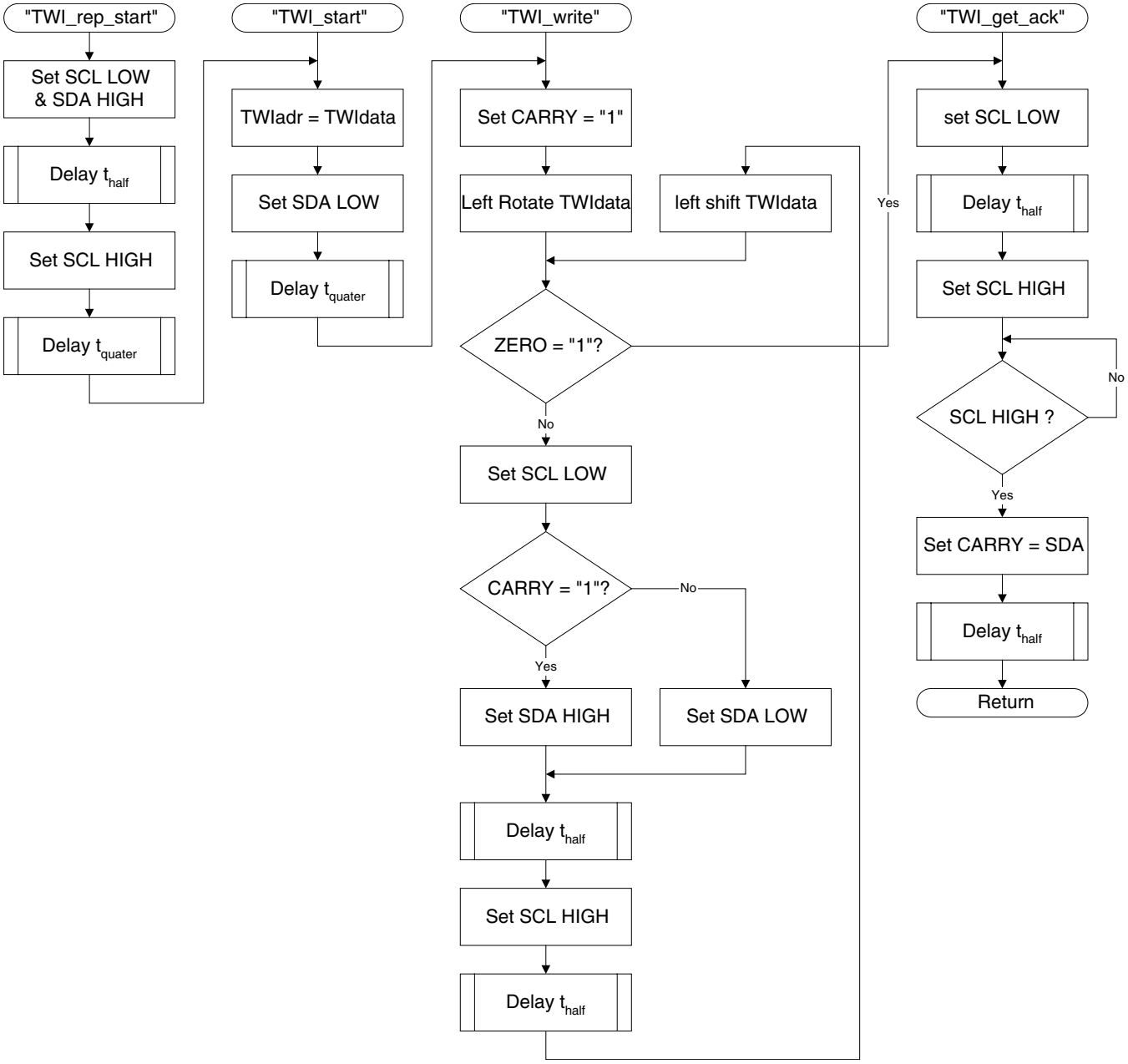


Figure 5 shows the flow chart for “TWI_start”, “TWI_rep_start”, “TWI_write”, and “TWI_get_ack”. These subroutines shares program code to reduce size.

"TWI_read"

Reads data (one byte) from the TWI bus.

Parameter	Value
Code Size	11
Execution Cycles	N/A
Register Usage	• Low Registers :None • High Registers :3 • Global Registers :1

Register	Input	Internal	Output
r16		"TWIdelay" – Delay Loop Counter	
r17			"TWIdata" – Received data
r19		"TWIstat" – Store acknowledge bit (global)	

"TWI_put_ack"

Put an acknowledge bit depending on carry flag is set or not. Separating this subroutine from the "TWI_read" routine is convenient for the user if a acknowledge is based on the result of the read operation.

Parameter	Value
Code Size	12
Execution Cycles	N/A
Register Usage	• Low Registers :None • High Registers :2 • Global Registers :1

Register	Input	Internal	Output
r16		"TWIdelay" – Delay Loop Counter	
r19		"TWIstat" – Acknowledge bit (global)	

Figure 6. “TWI_read” and “TWI_put_ack” Flow Chart

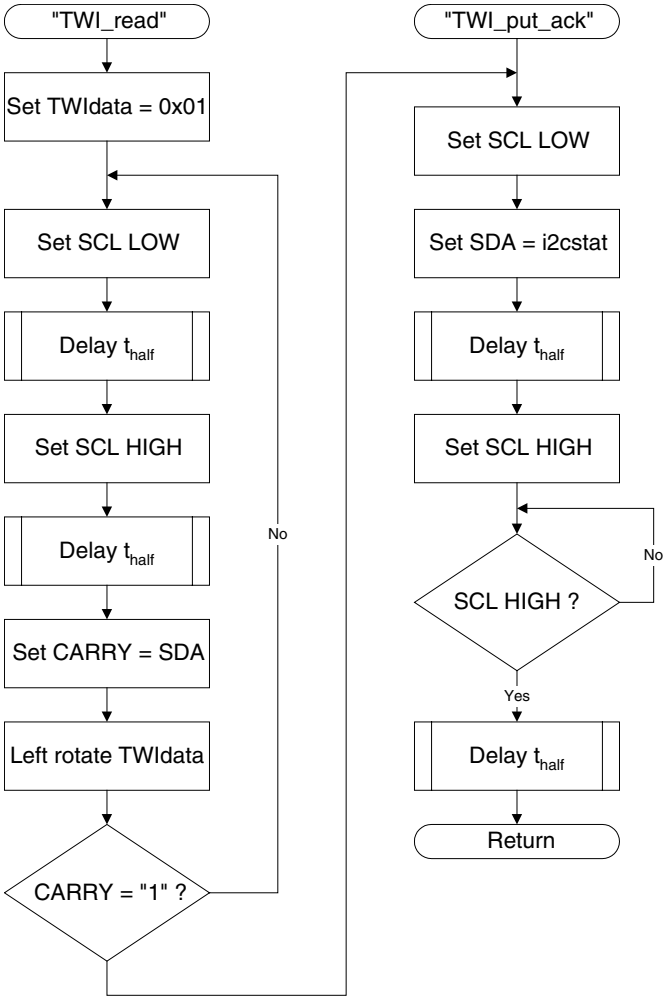


Figure 6 shows the flow chart for “TWI_read” and “TWI_put_ack”. These subroutines shares program code to reduce size.

“TWI_stop”

Generate stop condition. When a transfer is done the “TWI_end” must be called. When the bus is free (after “TWI”_end is called) all registers are free for use.

Parameter	Value
Code Size	8
Execution Cycles	N/A
Register Usage	• Low Registers :None • High Registers :1 • Global Registers :None

Register	Input	Internal	Output
r16		“TWIdelay” – Delay Loop Counter	

Figure 7. “TWI_stop” Flow Chart

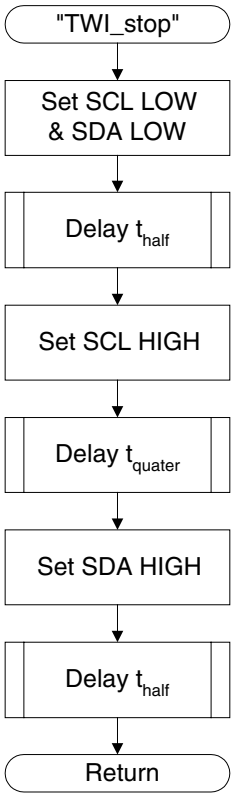


Figure 7 shows the flow chart for “TWI_stop”.

“TWI_do_transfer”

The “TWI_do_transfer” routine is implemented for convenience only. It uses the direction bit from the last address byte send, to decide whether to call the “TWI_read” or the “TWI_write” routine.

Tips and Warnings

The main loop in the program shows an example of reading and writing data to a 256-byte SRAM. This is a simple demonstration of how to use the TWI routines. Typically the reading and writing of SRAM will be implemented as functions calls, but since there is a large variety of slave implementations and ways of accessing them, the making this type of function calls is left for the user.

Warning! Do not change the order of the TWI routines. Most routines expects to be followed by a another specific TWI routine to work correctly.

Conclusion

This application note shows how to implement a master TWI interface on any AVR microcontroller device. This by using a minimum of resources. Since no interrupts are used in the implementation, these are free for other applications. It is also possible to use the TWI interface inside interrupts.

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

0954B–AVR–05/02 0M

AVR305: Half Duplex Compact Software UART

Features

- **32 Words of Code, Only**
- **Handles Baud Rates of up to 38.4 kbps with a 1 MHz XTAL**
- **Runs on Any AVR Device**
- **Only Two Port Pins Required**
- **Does Not Use Any Timer**

1 Introduction

In many applications utilizing serial communication, the software handling communication does not have to be performed as a background task. This application note describes how to implement polled software UART capable of handling speeds up to 614,400 bps on an AT90S1200. This implementation is intended for applications where small code is preferred. All bit delays are software delays, so no timer is required. The controller cannot perform any other tasks while receiving or transmitting, but when the operation is complete, the controller is free to continue program execution.

8-bit AVR® Microcontrollers

Application Note

Rev. 0952C-AVR-0905

2 Theory of Operation

In asynchronous serial communication, no clock information is transferred. The data is transferred serially, one bit at a time. In Idle state, the line is held at logical “1”. When data is to be transferred, the first bit is a so-called start bit, often given the symbol S. The start bit is a “0”, causing a “1” to “0” transition at the line. This can be detected by the receiver, signaling that data is coming. The following bits transferred are the data bits, with the LSB first. Then, one or more stop bits (P) are transferred. The stop bit is a logical "1", putting the line back to idle state before a new start bit followed by a data byte can be transferred. There must be at least one stop bit, so that a "1" to "0" transition can be detected by the receiver, indicating a new start bit. The *frame* shown in Figure 2-1 has got eight data bits and one stop bit. Sometimes, a parity bit is included between the last data bit and the stop bit, and there can be several stop bits.

Figure 2-1. Frame Format

The Receiver must sample the data line in the middle of every bit in order to receive the data properly. The bit length has to be equal for all bits, so that the Receiver knows when to sample the line. The Receiver is synchronized with the Transmitter by the falling edge of the start bit. The Receiver must have an internal timer in order to sample the bits at the right time.

The bit length must be the same for both transmitter and receiver, and some standard speeds are defined in bits per second, *bps*.

3 Implementation

3.1 Bit Length Delays - *UART_delay*

The delay between the bits are generated by calling a delay subroutine twice (as the subroutine generates a half-bit delay, see receiving data). If very short delays are required (when transmitting and receiving at very high speed), the delay must be implemented inside the *putchar* and *getchar* routines. The required delay length in clock cycles can be calculated using the following equation:

$$c = \frac{f_{CLCL}}{baud\ rate}$$

where c is the bit length in clock cycles and f_{CLCL} is the crystal frequency.

Both *putchar* and *getchar* use nine CPU cycles to send or receive a bit. Hence, a delay of *c* - 9 cycles must be generated between each bit. The *rcall* and *ret* instructions require a total of seven cycles. If the subroutine is to be called twice to generate the required delay, the delay has to be *d* cycles:

$$d = \frac{c-9}{2} - 7$$

If the delay is generated as shown below, the total execution time is 3·b cycles plus seven cycles for *rcall* and *ret*.

```
                    rcall       UART_delay
UART_delay:         ldi         temp,b
UART_delay1:        dec         temp
                    brne        UART_delay1
                    ret
```

The value *b* is found by the equation:

$$b = \frac{\frac{c-9}{2} - 7}{3} = \frac{c-23}{6}$$

The actual delay generated, calling delay twice is

$$d = (3 \times b + 7) \times 2 + 9 = 6 \times b + 23$$

From this, the minimum and maximum delays are d_{min} = 29 and d_{max} = 1,559 cycles. The *c* and *b* values for some bit rates and frequencies are shown in Table 3-1.

Table 3-1. "UART_delay" Subroutine Performance Figures

Parameter	Value
Code Size	4 words
Execution Cycles	Min: 7 cycles Max: 772 cycles (including *ret*)
Register Usage	Low Registers : None High registers : None Global : 1

Table 3-2. "UART_delay" Register Usage

Register	Input	Internal	Output
R17	-	"temp" – count variable	-

3.2 Transmitting Data - *putchar*

The *putchar* subroutine transmits the byte stored in the register *Txbyte*. The data bits are shifted into the Carry bit. The easiest way to generate stop bits is to let the zeros shifted into the transmitted byte be interpreted as ones. If the data byte is inverted before it is shifted, a "0" in Carry must give a "1" on the line, and a "0" in Carry gives a "1" on the line. When 0's are shifted into the data byte, they are handled as 1's. This way, any number of stop bits can be generated by just repeating the transmittal loop. The start bit is generated by setting the carry bit before data are shifted out.

Table 3-3. "putchar" Subroutine Performance Figures

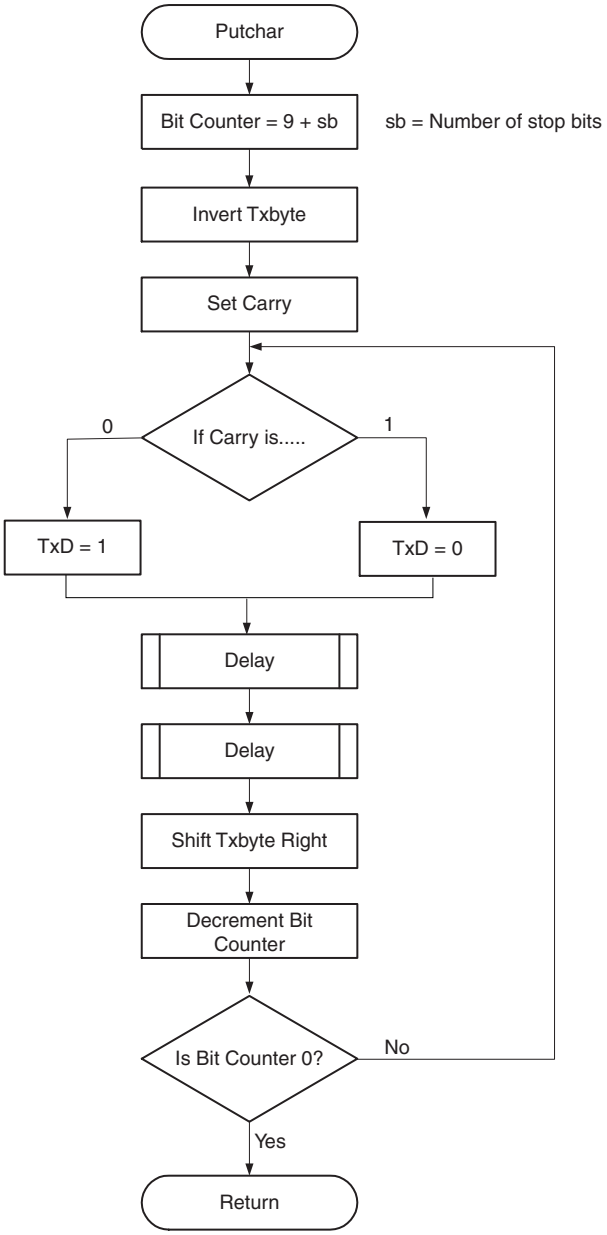
Parameter	Value
Code Size	14 words
Execution Cycles	Depends on bit rate
Register Usage	Low Registers : None High registers : None Global : 2

Table 3-4. "putchar" Register Usage

Register	Input	Internal	Output
R16	-	"bitcnt" – counts the number of bits transferred	-
R18	"Txbyte – the byte to send	-	-

The algorithm for transmitting data is shown in Figure 3-1

Figure 3-1. *putchar* Subroutine



3.3 Receiving Data - *getchar*

First, the routine waits for a logical "0" (not a transition). When the start bit is detected, a 1.5 bit delay is generated. This is performed by calling the delay subroutine three times. Sampling then starts at 1-bit intervals. The Carry is set or cleared according to the logic value at the RxD pin. If less than eight data bits are received, the Carry is shifted into the receive byte. If not, the routine returns with the received byte in *Rxbyte*.

The routine waits for one bit length after the last data bit and terminates in the middle of the stop bit. This is done to prevent detection of a false start bit if the routine is called again immediately after a complete reception.

Table 3-5. "getchar" Subroutine Performance Figures

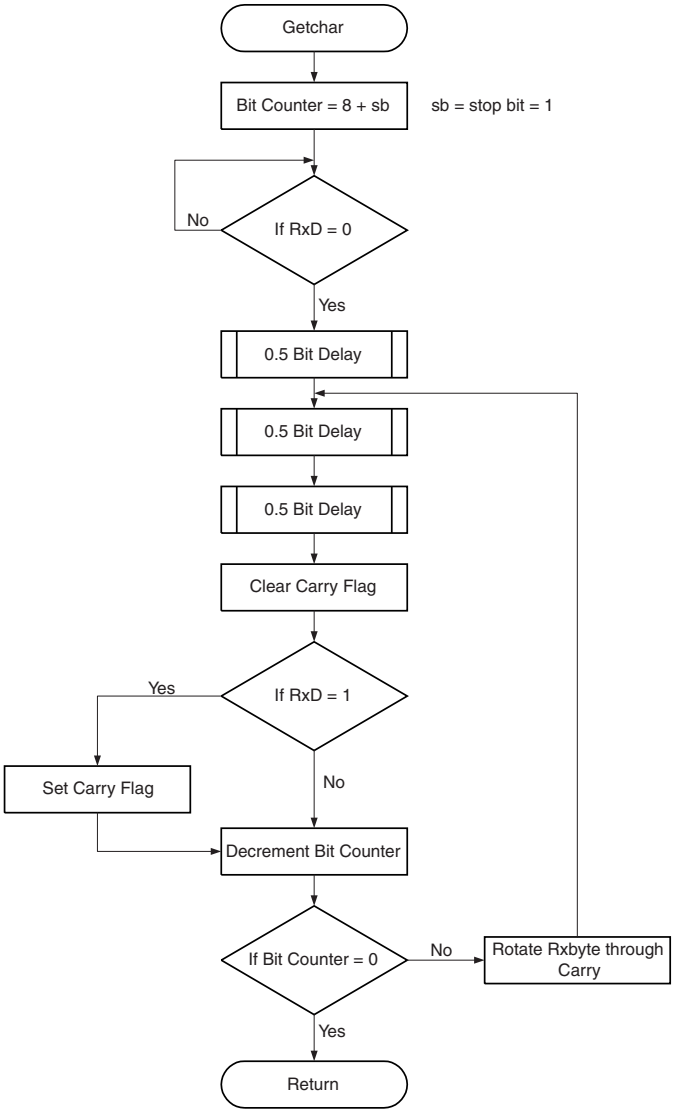
Parameter	Value
Code Size	14 words
Execution Cycles	Waits until byte received
Register Usage	Low Registers : None High registers : None Global : 2

Table 3-6. "getchar" Register Usage

Register	Input	Internal	Output
R16	-	"bitcnt" – counts the number of bits received	-
R18	-	-	"Rxbyte" – the received byte

The algorithm for receiving data is shown in Figure 3-2.

Figure 3-2. *getchar* Subroutine



4 Example Program

The example program receives a character with *getchar* and echoes it back with *putchar*.

Table 4-1. Overall Performance Figures

Parameter	Value
Code Size	32 words - UART routines only 40 words - Complete application note
Register Usage	Low Registers : None High registers : 4 Global : None
Interrupt Usage	None
Peripheral Usage	Port D pin 0 and 1 (any two pins can be used)

Table 4-2. Baud Rate Table

1 MHz

BaudRate	Cycles required	b-value	Error %
2400	417	66	0.6
4800	208	31	0.3
9600	104	14	2.7
14400	69	8	2.2
19200	52	5	1.8
28800	35	2	0.8
57600	-	-	-
115200	-	-	-

1.8432 MHz

BaudRate	Cycles required	b-value	Error %
2400	768	124	0.1
4800	384	60	0.3
9600	192	28	0.5
14400	128	18	2.3
19200	96	12	1.0
28800	64	7	1.6
57600	32	2	(1) 9.4
115200	-	-	-

2 MHz

BaudRate	Cycles required	b-value	Error %
2400	833	135	0.0
4800	417	66	0.6
9600	208	31	0.3
14400	139	19	1.4
19200	104	14	2.7
28800	69	8	2.2
57600	35	2	0.8
115200	-	-	-

2.4576 MHz

BaudRate	Cycles required	b-value	Error %
2400	1024	167	0.1
4800	512	82	0.6
9600	256	39	0.4
14400	171	25	1.4
19200	128	18	2.3
28800	85	10	2.7
57600	43	3	(1) 3.9
115200	-	-	-

3.276 MHz

BaudRate	Cycles required	b-value	Error %
2400	1365	224	0.1
4800	683	110	0.0
9600	341	53	0.1
14400	228	34	0.2
19200	171	25	1.4
28800	114	15	0.7
57600	57	6	(1) 3.7
115200	28	1	2.0

3.6864 Mhz

BaudRate	Cycles required	b-value	Error %
2400	1536	252	0.1
4800	768	124	0.1
9600	384	60	0.3
14400	256	39	0.4
19200	192	28	0.5
28800	128	18	2.3
57600	64	7	1.6
115200	32	2	(1) 9.4

4 MHz

BaudRate	Cycles required	b-value	Error %
2400	1667	(2) 274	0.0
4800	833	135	0.0
9600	417	66	0.6
14400	278	42	1.0
19200	208	31	0.3
28800	139	19	1.4
57600	69	8	2.2
115200	35	2	0.8

4.608 MHz

BaudRate	Cycles required	b-value	Error %
2400	1920	(2) 316	0.1
4800	960	156	0.1
9600	480	76	0.2
14400	320	50	0.9
19200	240	36	0.4
28800	160	23	0.6
57600	80	10	(1) 3.8
115200	40	3	2.5

7.3728 MHz

BaudRate	Cycles required	b-value	Error %
2400	3072	(1) 508	0.0
4800	1536	252	0.1
9600	768	124	0.1
14400	512	82	0.6
19200	384	60	0.3
28800	256	39	0.4
57600	128	18	2.3
115200	64	7	1.6

8 MHz

BaudRate	Cycles required	b-value	Error %
2400	3333	(2) 552	0.0
4800	1667	(2) 274	0.0
9600	833	135	0.0
14400	556	89	0.3
19200	417	66	0.6
28800	278	42	1.0
57600	139	19	1.4
115200	69	8	2.2

9.216 MHz

BaudRate	Cycles required	b-value	Error %
2400	3840	(2) 636	0.0
4800	1920	(2) 316	0.1
9600	960	156	0.1
14400	640	103	0.2
19200	480	76	0.2
28800	320	50	0.9
57600	160	23	0.6
115200	80	10	(1) 3.8

11.059 MHz

BaudRate	Cycles required	b-value	Error %
2400	4608	(2) 764	0.0
4800	2304	(2) 380	0.0
9600	1152	188	0.1
14400	768	124	0.1
19200	576	92	0.2
28800	384	60	0.3
57600	192	28	0.5
115200	96	12	1.0

14.746 MHz

BaudRate	Cycles required	b-value	Error %
2400	6144	(2) 1020	0.0
4800	3072	(2) 508	0.0
9600	1536	(2) 252	0.1
14400	1024	167	0.1
19200	768	124	0.1
28800	512	82	0.6
57600	256	39	0.4
115200	128	18	2.3

16 MHz

BaudRate	Cycles required	b-value	Error %
2400	6667	(2) 1107	0.0
4800	3333	(2) 552	0.0
9600	1667	(2) 274	0.0
14400	1111	181	0.2
19200	833	135	0.0
28800	556	89	0.3
57600	278	42	1.0
115200	139	19	1.4

Notes:
1. Baud rates over 3 % are likely to cause communication errors
2. The provided delay routine will not give a valid result for b-values larger than 255 when using 8-bits registers.

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

0952C-AVR-0905