AN74875

Designing with Serial I²C nvSRAM

Author: Shivendra Singh
Associated Project: Yes
Associated Part Family: CY14xxxxI, CY14xxxxJ
Software Version: PSoC® Creator™ 3.0 or above
Related Application Notes: AN61546, AN43593

AN74875 provides design guidelines and example circuits for inter-integrated circuit (I²C) nvSRAM device. The I²C nvSRAM is a high-performance nonvolatile serial interface memory that offers zero cycle delay write operation and infinite SRAM write endurance. The I²C nvSRAM is a slave I²C device and requires an I²C master controller to access it in a system. An associated library component for PSoC 3 is also provided as an example project.

Contents

Introduction

Cypress nvSRAM integrates a SRAM cell and a nonvolatile memory cell into a single nvSRAM cell. In the normal mode of operation, all reads and writes happen directly from and to the SRAM portion of the nvSRAM. This provides faster write and read access compared to any existing nonvolatile memory technology such as EEPROM and flash. In the event of system power loss, data from the SRAM is transferred to its nonvolatile cell automatically by using energy stored in a small capacitor connected to the V_{CAP} pin. During the subsequent power-on cycle, data from the nonvolatile cell is recalled automatically in the SRAM array and available to the user. A capacitor connected to the V_{CAP} pin is charged by the nvSRAM during the normal operation.

The nvSRAM specifies one million endurance cycles for nonvolatile cells. In nvSRAM, endurance cycle is consumed only when data transfer happens from the SRAM cells to nonvolatile cells during the STORE operation. The nonvolatile store operation in the nvSRAM is initiated either automatically when the device power drops below a predefined threshold level (V_{SWITCH}), or on demand either by writing a specific command in the command register (0xAA), or through the hardware pin ($\overline{HSB}$) by toggling it to LOW. The command register in the nvSRAM is defined in control register space. The control registers are addressed through a dedicated I²C slave ID. Detailed description on the nvSRAM addressing has been provided in Addressing in I2C nvSRAM section described later in this application note.

The nvSRAM initiates nonvolatile Store only when the system power failure is detected and new data written in SRAM is required to be moved safely into nonvolatile cells. Hence the total non volatile cell endurance count in nvSRAM equates to the total number of non volatile Store cycles not the SRAM write cycles.

There are many data logging applications, which require instant saving of runtime critical information in the event of power loss. This critical information includes controller run time states, scratch pad data, parameter settings, and the other environment variables measured by the controller. The I²C nvSRAM can ideally fit into such data logging applications due to its fast nonvolatile write speed. The I²C master controller can log hundreds of bytes of data in tens of microseconds in nvSRAM whereas it takes tens of milliseconds to write the same amount of data in EEPROM or flash memory. The I²C nvSRAM is offered in industry standard 8-pin SOIC and 16-pin SOIC packages.

This application note describes about the I²C nvSRAM configuration, example circuits for different package options, method of determining suitable pull-up resistor values for the I²C bus, data byte format for the I²C communication in nvSRAM, and I²C addressing scheme to access the Memory, Real Time Clock (RTC), and Control functions of the nvSRAM. For other details on the I²C nvSRAM, refer the specific device datasheet.

A PSoC 3 nvRAM I²C library component is attached along with this application note as an associated project.

I²C nvSRAM Configurations

The I²C nvSRAM supports the highest I²C data transfer rate up to 3.4 Mbits/s (I²C clock frequency at 3.4 MHz) along with the support of all the other lower frequency accesses as defined in I²C-bus standard spec.

- **Standard-mode (Sm)** - Bit rate up to100 Kbit/s
- **Fast-mode (Fm)** - Bit rate up to 400 Kbit/s
- **Fast-mode Plus (Fm+)** - Bit rate up to 1 Mbit/s
- **High-speed mode (Hs)** - Bit rate up to 3.4 Mbit/s.

All the above four bus modes are offered in all device configurations (see Table 1) and do not require any special setting in the device.

Applicability of I²C-Bus Protocol Features

Table 1 summarizes all the mandatory and optional features of standard I²C-slave bus specifications. The I²C nvSRAM supports all mandatory features of a standard I²C slave device.

Table 1. Applicability of I²C-bus protocol

Feature	I²C Spec Standards	I²C nvSRAM
START condition	Mandatory	√
STOP condition	Mandatory	√
Acknowledge	Mandatory	√
7-bit slave address	Mandatory	√
10-bit slave address	Optional	Not Offered
Clock stretching	Optional	Not Required
General call address	Optional	Not Offered
Device ID	Optional	Not Offered
Software Reset	Optional	Not Offered

I²C nvSRAM Device Options

Cypress supports I²C nvSRAM in different configurations and package options as shown in Table 2.

Table 2. I²C nvSRAM Configurations

nvSRAM Part Number	Status	Operating Voltage (Typ)	Package	WP Pin	V_{CAP} Pin / AutoStore	$\overline{(HSB)}$ Pin / HW Store	A0 Pins	Number of Devices Per I²C Bus	RTC
CY14CXXXJ	NRND	2.5 V	8 SOIC	Yes	No/No	No/No	Yes [Note 1]	4 or 8 [Note 1]	NO
CY14BXXXJ	NRND	3 V							
CY14EXXXJ	NRND	5 V							
CY14CXXXJ	NRND	2.5 V	8 SOIC	Yes	Yes/Yes	No/No	No	4	NO
CY14BXXXJ	NRND	3 V							
CY14EXXXJ	NRND	5 V							
CY14CXXXJ	NRND	2.5 V	16 SOIC	Yes	Yes/Yes	Yes	Yes [Note 1]	4 or 8 [Note 1]	NO
CY14BXXXJ	NRND	3 V							
CY14EXXXJ	NRND	5 V							
CY14CXXXI	Contact Cypress	2.5 V	16 SOIC	Yes	Yes/Yes	Yes	Yes [Note 1]	4 or 8 [Note 1]	YES
CY14BXXXI	In Production	3 V							
CY14EXXXI	In Production	5 V							

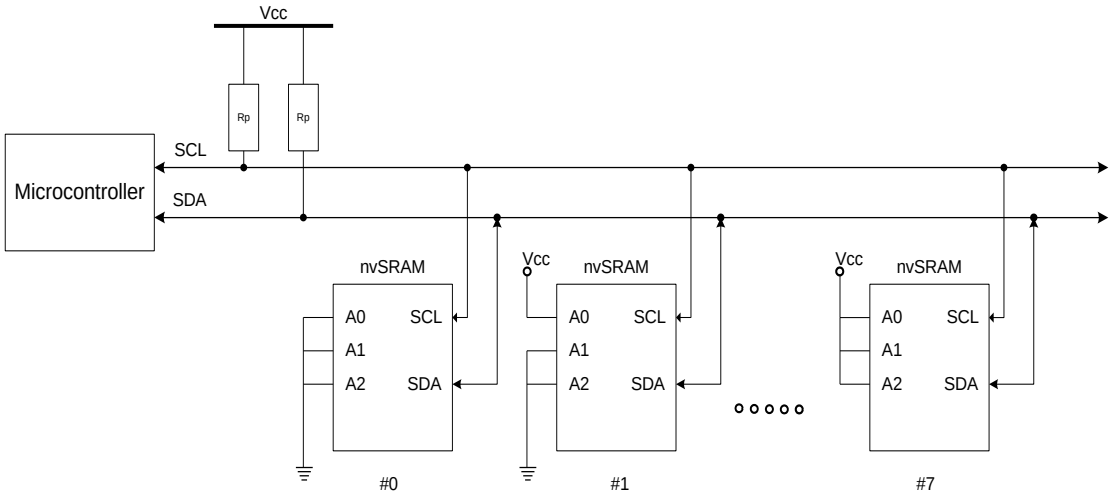
NRND – Not Recommended for new designs

Note 1 The least significant slave address bit space (A0) is internally used in 1-Mbit nvSRAM devices; therefore it is not available in 1-Mbit density options. The A0 pin is available in all 512-Kbit and lower density options except J2 parts. Without A0 pin, the I²C nvSRAM is limited to maximum of four devices per I²C bus.

I²C nvSRAM Device Connections

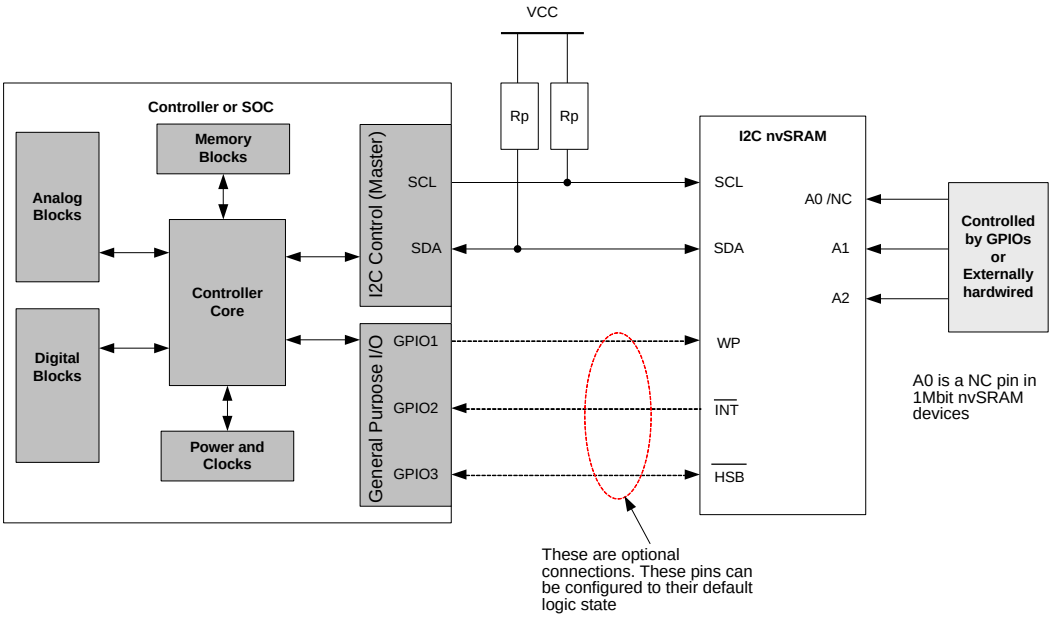
A typical I²C single master-multi slave configuration is shown in Figure 1. The I²C master device can be any microcontroller or a programmable device, which should be capable of generating I²C master protocols, whereas the slave devices can be any standard I²C slave device. In Figure 1 example, the I²C nvSRAM is taken as an I²C slave. Since 512-Kbit and lower density I²C nvSRAMs support three slave addressing bits in a few package options, it is possible to connect up to eight devices on the same I²C bus. A unique slave ID is assigned to each slave device by configuring slave select address lines (A2, A1, A0) in eight different combinations. In a package configuration where A0 is not available, it is possible to connect only up to four slave devices sharing the same bus by configuring slave select address pins A2 and A1.

Figure 1. Typical I²C Master Slave Configuration



A typical system level configuration of the I²C nvSRAM device is illustrated in Figure 2. For microcontrollers that do not have a dedicated I²C bus, general purpose I/O ports may be used for SCL and SDA by bit banging.

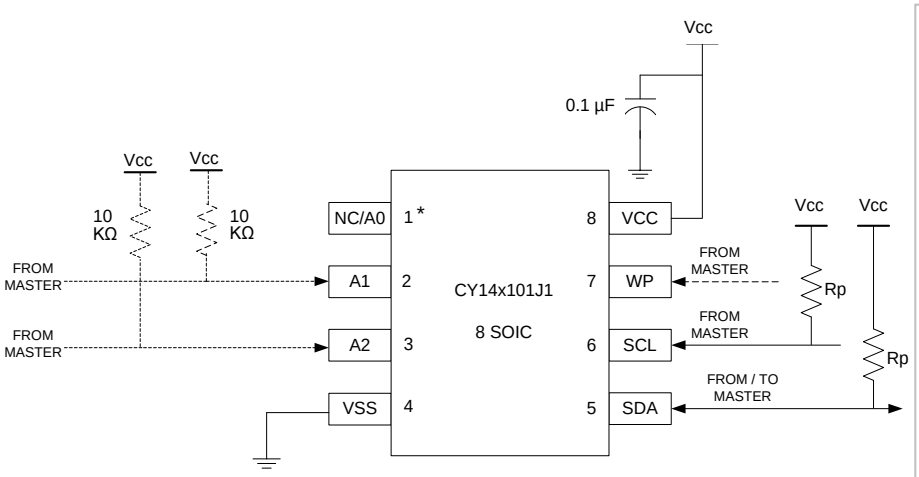
Figure 2. Typical I²C nvSRAM Connection



Sample Circuits

The following figures (Figure 3 to 5) show the detailed schematic connections for 1-Mbit I²C nvSRAMs. The hardware connections between an I²C master and the nvSRAM slave will remain the same for all lower density (512-Kbit and below) parts.

Figure 3. 8 Pin SOIC 1-Mbit I²C nvSRAM Interface (No V_{CAP})



All optional connections are shown as dotted lines.

Pull-up resistors on A2 and A1 will set the slave addresses bits A2 and A1 to '1'. If system requires configuring either of the address pin as '0' then remove the pull-up resistor and leave the pin floating. It will be pulled to LOW internally by a weak pull-down resistor.

Figure 4. 8 Pin SOIC 1-Mbit I²C nvSRAM Interface (with V_{CAP})

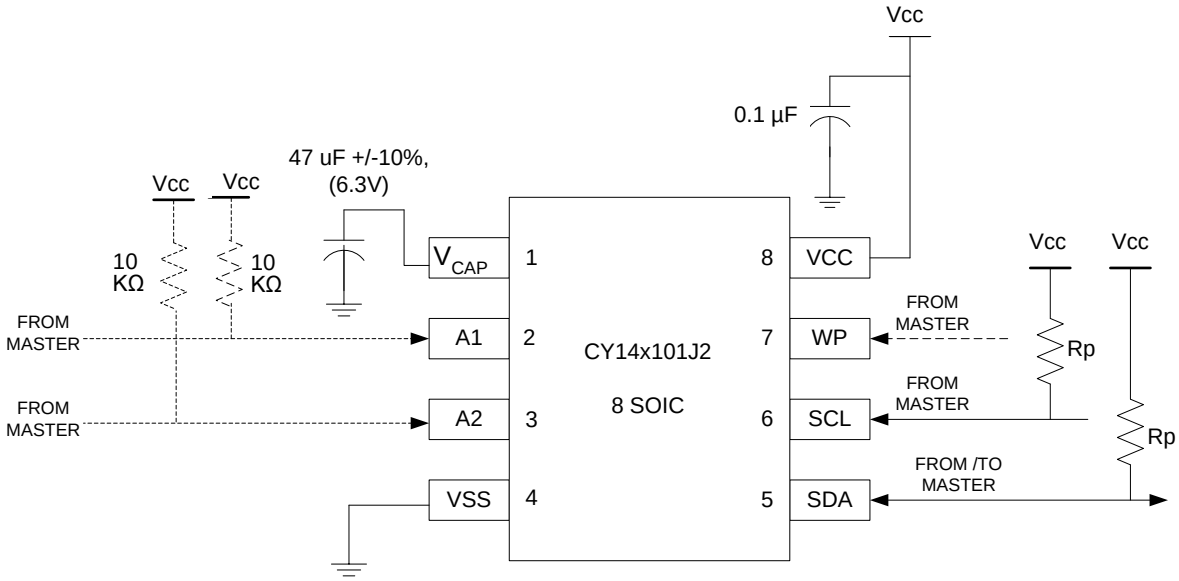
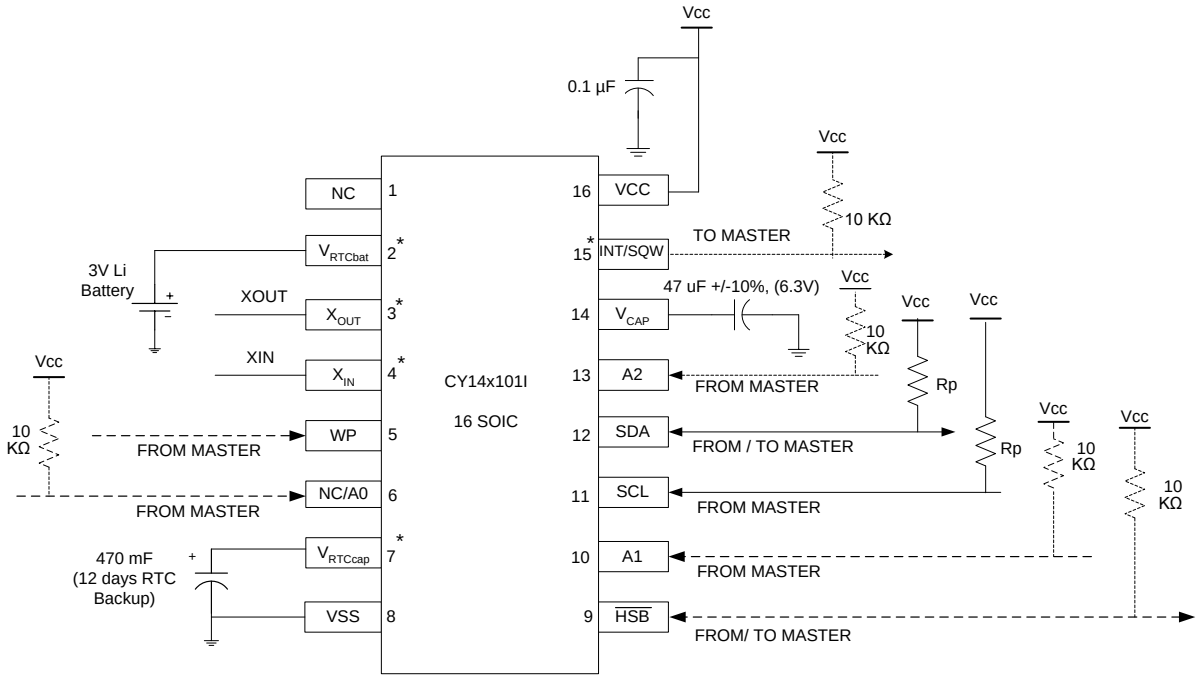


Figure 5. 16 Pin SOIC 1-Mbit (RTC) I²C nvSRAM Interface



Pin marked with '*' are specific to the RTC nvSRAM part. These pins become no connect (NC) pin in non RTC parts.

Pin 6 of CY14x101I (Figure 5)and Pin 1 of CY14x101J1 (Figure 3) are NC pins for 1 Mbit density. It is enabled as Slave Address bit 0 (A0) for 512 Kbit and lower density parts.

Connect either a battery on the V_{RTCbat} or a super capacitor on the V_{RTCcap} pin.

Determining I²C Pull-up Resistor Values

The I²C bus transmits data and clock on SDA and SCL lines. The SDA and SCL lines are open-drain (also known as open-collector in the TTL family) output driver, that means I²C master and slave devices can only drive these lines to logic LOW or leave them open. The termination resistor (Rp) pulls the line HIGH to the V_{CC} if no I²C device on the same bus is pulling it LOW. The open drain driver configuration is required to support some special I²C features such as multi-master configuration and clock stretching by slave. The clock stretching is an optional feature of I²C standard and not supported in I²C nvSRAM, therefore the I²C clock signal is an input (only) signal in I²C nvSRAM.

Together with the total bus capacitance (Cb), the termination resistor (Rp) affects the timing behavior of the signals on SDA and SCL. While I²C device pulls down the line with open drain drivers, the pull-up resistor Rp is responsible to get the signal back to HIGH level in a specified time. The value of pull Rp depends on multiple electrical parameters such as operating voltage (V_{CC}), output LOW logic level (V_{OL}) spec of the device; sink current (I_{OL}) spec, total bus load (Cb) and timing parameters such as rise time (t_R) spec.

The following sections describe the methods of determining the pull resistor value for I²C bus in a given system configuration.

Determining Rp (Max)

Consider the input threshold of CMOS logic level as V_{IH} = 0.7 V_{CC} (Min) and V_{IL} = 0.3 V_{CC} (Max) for the purposes of RC time constant calculation.

Then V(t) = V_{CC} (1 − e−t / RC), where t is the time since the charging started and RC is the time constant.

V (t1) = 0.3 × V_{CC} = V_{CC} (1 − e−t1 / RC) then:

$$t1 = 0.3566749 \times RC$$ Equation 1

V (t2) = 0.7 × V_{CC} = V_{CC} (1 − e−t2 / RC); then:

$$t2 = 1.2039729 \times RC$$ Equation 2

The total rise time (T) is the time it takes to charge the bus capacitance voltage level from V_{IL} to V_{IH}:

T = t2 − t1

= 1.2039729 × RC - 0.3566749 × RC

$= \mathbf{0.8473 \times RC}$ Equation 3

Equation 3 is used to determine the maximum limit for the pull-up resistor value to connect to the I²C line. Table 3 shows maximum Rp as a function of bus capacitance for all timing modes. For each mode, the Rp (max) is a function of the rise time minimum (t_R) and the estimated bus capacitance (Cb):

$$\mathbf{Rp\ (Max)} = \frac{\mathbf{tr}}{(\mathbf{0.8473 * Cb})}$$ Equation 4

The bus capacitance (Cb) is the total capacitance of wire, connections and pins.

Determining Rp (Min)

The operating voltage and the sink current (I_{OL}) limit the pull-up resistor minimum value, Rp (min). The value of Rp (Min) as a function of V_{CC} and I_{OL} are calculated using the Equation 5.

$$\mathbf{Rp\ (min)} = \frac{\mathbf{Vcc - VOL\ (max)}}{\mathbf{IOL}}$$ Equation 5

The value of Rp must be selected within specified min and max range.

$$\mathbf{Rp\ (Min) \le Rp \le Rp(Max)}$$ Equation 6

Low power designs should prefer using a value toward the higher limit of the range in order to limit the current consumption.

Table 3 provides the list of values of Rp (Min, Max) for a given bus load condition and operating voltage. Values not appearing in Table 3 can be obtained from the equations 4 and 5 for calculating Rp (Max) and Rp (Min).

Shaded region in the Table 3 indicates that Rp (Min) exceeds the Rp (Max) value for a few bus loads (Cb) under a given operating voltage condition. Since the Rp (Min) can never exceed Rp (Max) value, this will put a limit on the maximum capacitive load (Cb) to be used on the I²C bus.

For example: If a 3 V part is configured to operate at minimum V_{CC} supply (V_{CC} = 2.7 V) then the system must not exceed the following load (in picofarad) on the SCL and SDA lines when operating in the following bus modes:

Sm = Cb ≤ 550 pF; 0.77 kΩ ≤ Rp ≤ 2.15 kΩ

Fm = Cb ≤ 450 pF; 0.77 kΩ ≤ Rp ≤ 0.79 kΩ

Fm+ = Cb ≤ 150 pF; 0.77 kΩ ≤ Rp ≤ 0.94 kΩ

Hs = Cb ≤ 100 pF; 0.77 kΩ ≤ Rp ≤0.94 kΩ

Similarly, the max bus load (Cb) and I²C pull-up resistor (Rp) value for other operating voltages and operating frequencies can be obtained from Table 3 and Figure 6.

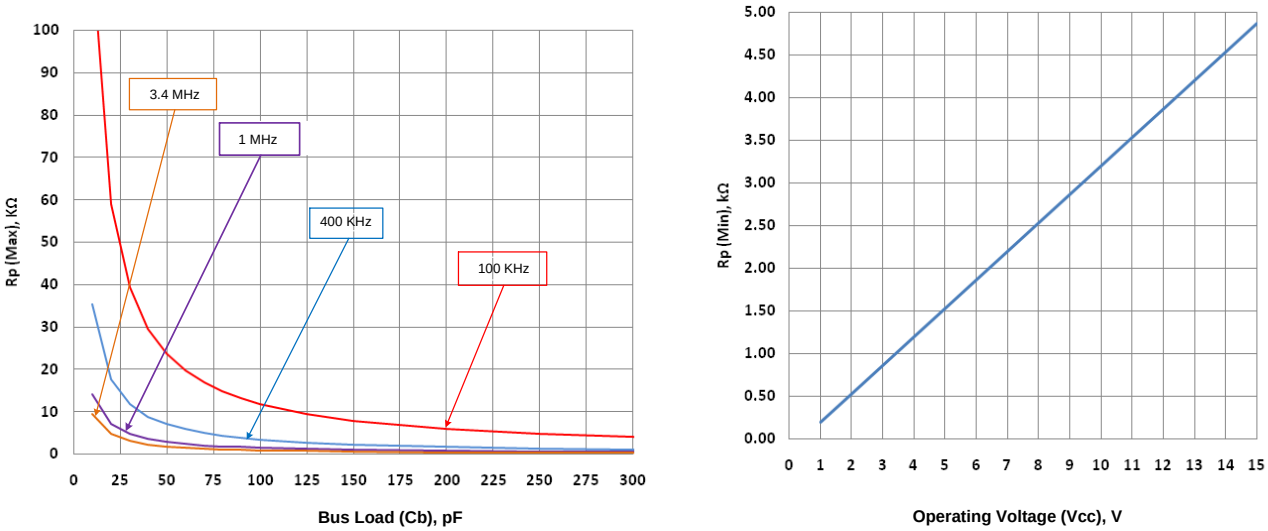
Table 3. Rp (Min, Max) Values for Different Bus Loads and Operating Voltages

Cb (pf)	Rp(Min) (kΩ)	Rp(Max) (kΩ)			
	2.45V	100 KHz	400 KHz	1 MHz	3.4 MHz
10	0.68	118.02	35.41	14.16	9.44
20	0.68	59.01	17.70	7.08	4.72
30	0.68	39.34	11.80	4.72	3.15
40	0.68	29.51	8.85	3.54	2.36
50	0.68	23.60	7.08	2.83	1.89
60	0.68	19.67	5.90	2.36	1.57
70	0.68	16.86	5.06	2.02	1.35
80	0.68	14.75	4.43	1.77	1.18
90	0.68	13.11	3.93	1.57	1.05
100	0.68	11.80	3.54	1.42	0.94
125	0.68	9.44	2.83	1.13	0.76
150	0.68	7.87	2.36	0.94	0.63
200	0.68	5.90	1.77	0.71	0.47
250	0.68	4.72	1.42	0.57	0.38
300	0.68	3.93	1.18	0.47	0.31
350	0.68	3.37	1.01	0.40	0.27
400	0.68	2.95	0.89	0.35	0.24
450	0.68	2.62	0.79	0.31	0.21
500	0.68	2.36	0.71	0.28	0.19
550	0.68	2.15	0.64	0.26	0.17

Cb (pf)	Rp(Min) (kΩ)	Rp(Max) (kΩ)			
	2.7V	100 KHz	400 KHz	1 MHz	3.4 MHz
10	0.77	118.02	35.41	14.16	9.44
20	0.77	59.01	17.70	7.08	4.72
30	0.77	39.34	11.80	4.72	3.15
40	0.77	29.51	8.85	3.54	2.36
50	0.77	23.60	7.08	2.83	1.89
60	0.77	19.67	5.90	2.36	1.57
70	0.77	16.86	5.06	2.02	1.35
80	0.77	14.75	4.43	1.77	1.18
90	0.77	13.11	3.93	1.57	1.05
100	0.77	11.80	3.54	1.42	0.94
125	0.77	9.44	2.83	1.13	0.76
150	0.77	7.87	2.36	0.94	0.63
200	0.77	5.90	1.77	0.71	0.47
250	0.77	4.72	1.42	0.57	0.38
300	0.77	3.93	1.18	0.47	0.31
350	0.77	3.37	1.01	0.40	0.27
400	0.77	2.95	0.89	0.35	0.24
450	0.77	2.62	0.79	0.31	0.21
500	0.77	2.36	0.71	0.28	0.19
550	0.77	2.15	0.64	0.26	0.17

Cb (pf)	Rp(Min) (kΩ)	Rp(Max) (kΩ)			
	4.5V	100 KHz	400 KHz	1 MHz	3.4 MHz
10	1.37	118.02	35.41	14.16	9.44
20	1.37	59.01	17.70	7.08	4.72
30	1.37	39.34	11.80	4.72	3.15
40	1.37	29.51	8.85	3.54	2.36
50	1.37	23.60	7.08	2.83	1.89
60	1.37	19.67	5.90	2.36	1.57
70	1.37	16.86	5.06	2.02	1.35
80	1.37	14.75	4.43	1.77	1.18
90	1.37	13.11	3.93	1.57	1.05
100	1.37	11.80	3.54	1.42	0.94
125	1.37	9.44	2.83	1.13	0.76
150	1.37	7.87	2.36	0.94	0.63
200	1.37	5.90	1.77	0.71	0.47
250	1.37	4.72	1.42	0.57	0.38
300	1.37	3.93	1.18	0.47	0.31
350	1.37	3.37	1.01	0.40	0.27
400	1.37	2.95	0.89	0.35	0.24
450	1.37	2.62	0.79	0.31	0.21
500	1.37	2.36	0.71	0.28	0.19
550	1.37	2.15	0.64	0.26	0.17

Figure 6. Rp (Min, Max) Values for Different Bus Loads and Operating Voltages



Control Input Pin Configuration

The I²C nvSRAM has many control pins that are input pins and they should be properly biased to fixed logic state (HIGH or LOW) for the proper operation of the device. If an input control pin is left floating without biasing it o appropriate logic levels (either HIGH or LOW) then it is possible that the floating pin may settle to some intermediate metabstable state, which can make device behavior random. Therefore, all unused input pins that do not have any internal pull-up or pull-down option should always be tied to a proper logic level externally by using pull-up or pull-down resistor. A resistor of value between 1 kΩ -10 kΩ can be used for this purpose.

WP Pin:

The WP pin is an active high pin and protects entire memory and all registers from write operations. When this pin is HIGH, all memory and register writes are prohibited and address counter is not incremented. The I²C nvSRAM provides an internal pull-down resistor on this pin. Therefore, this pin can be left floating (no connect) if write protect functionality is not used. If this pin is connected to a controller I/O for external control then an external pull-up resistor is recommended to avoid any undesired triggering due to noise on this line. A resistor of value between 1 kΩ -10 kΩ can be used for this purpose.

A2, A1, A0 Pins:

These are slave address pins and are used to configure the different slave addresses for different slave devices in multi slave configuration. These pins are internally pulled to LOW and hence can be left floating (not connected) if not used. To configure to logic HIGH state, these pins should either be connected to external pull-up resistor or these can be directly connected to the V_{CC} power supply. A resistor of value between 1 kΩ -10 kΩ can be used as a pull-up resistor. In a few configurations where system requires changing the slave address dynamically, these address pins should be connected to the controller I/Os for configuring the slave select address pins (A2, A1, A0) on the fly and access the device.

$\overline{HSB}$ Pin:

The $\overline{HSB}$ pin is a bidirectional pin on the nvSRAM. As an output, it provides nvSRAM ready or busy status during normal operation. When device is powering up or a nonvolatile Store cycle is in progress, the $\overline{HSB}$ pin is pulled to LOW by the device indicating its busy status. When the $\overline{HSB}$ pin is in HIGH state, it indicates that the device is ready for normal write or read operations. As an input pin, the $\overline{HSB}$ pin is used to initiate hardware STORE externally by pulling it to LOW by the controller. This pin can be left floating if not connected to any GPIO. The I²C nvSRAM provides an internal weak pull-up resistor on the $\overline{HSB}$ pin to keep it HIGH during normal operation. If this pin is connected to a controller I/O for external control then an external pull-up resistor is recommended to avoid any undesired triggering due to noise on this line. A resistor of value between 1 kΩ -10 kΩ can be used for this purpose.

V_{CAP}:

A capacitor connected on the V_{CAP} pin supplies power to the nvSRAM for transferring data SRAM nonvolatile elements in case of power loss. During normal operation, the device draws current from V_{CC} to charge the capacitor on V_{CAP}. Stored charge on the V_{CAP} is used by the nvSRAM device to perform a single STORE operation. If the voltage on the V_{CC} pin drops below V_{SWITCH}, the device automatically isolates the V_{CAP} pin from V_{CC} and STORE operation is initiated using stored charge on the V_{CAP}.

It is a must to connect an appropriate value capacitor on the V_{CAP} pin for a successful AutoStore operation. The capacitor value selected should fall within the range prescribed in the device datasheet. An improper selection of capacitor may lead to malfunctioning of the device. See application note, *AN43593 - Storage Capacitor Options for Cypress nvSRAM*, for more details on capacitor selection guidelines for nvSRAM products.

RTC Part Specific Pin Configuration

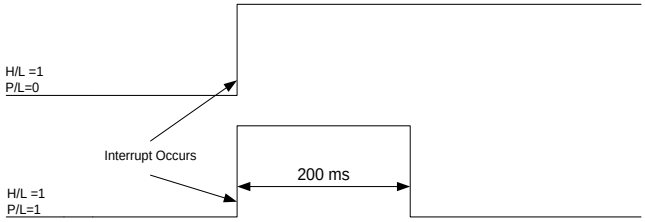
The RTC feature requires following additional pins on the package. These pins should be configured appropriately for correct RTC function.

INT Pin: This is an output pin in the RTC parts. The RTC nvSRAM offers different functionalities such as alarm, watchdog timer, calibration clock output, and square wave generator. The INT output is multiplexed to bring out the status/ output of functionalities, depending upon the RTC register setting and their priority defined in the nvSRAM. The INT pin is a configurable driver output. The output mode of INT pin is configured by setting 'H/L' bit in the Interrupt Status/Control register (0x06) in the I²C slave device. When the H/L bit is set to '1', the INT output is configured as active HIGH and the drive mode is push pull. When the H/L bit is set to '0', the INT output drive is configured as active LOW open drain output and thus requires an external pull-up resistor to drive the output to a logic HIGH state when not driven by the device. The INT pin must be pulled to the V_{CC} by using an external pull-up resistor of value between 1 kΩ – 10 kΩ when using INT in active low mode (H/L bit is set to '0').

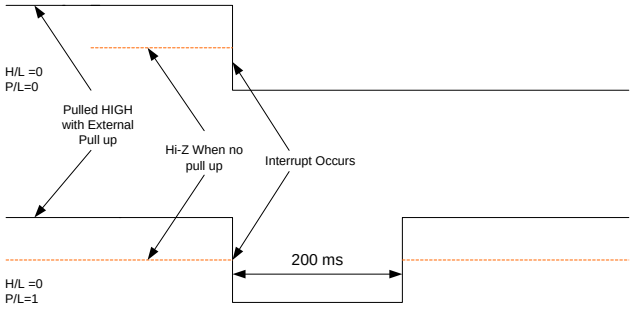
The INT pin behavior for different H/L and P/L settings: The H/L bit setting determines the status of INT pin output as HIGH or LOW when interrupt occurs. Similarly, the P/L settings determine the pulse or level for the INT pin. The I²C nvSRAM interrupt pin (INT) behavior is shown in Figure 7.

Figure 7. INT Pin Behavior (RTC)

H/L set to '1'



H/L set to '0'



V_{RTCbat} and V_{RTCcap} Pins: These pins are used to provide the backup power supply to the RTC circuitry to keep the oscillator clock running when the system power supply (V_{CC}) is down. To backup the RTC oscillation during power down, either connect V_{RTCbat} to a non-rechargeable or connect a super capacitor on the V_{RTCcap} pin. If not used, these pins should be left floating.

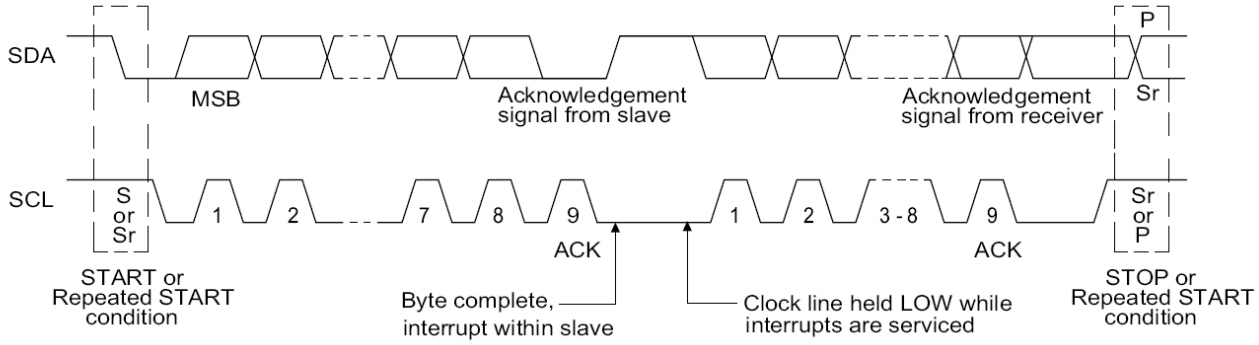
Note The V_{RTCcap} pin cannot be shorted to the V_{SS} directly because this pin is used to charge the super capacitor connected to it during the normal operation. Hence, connecting the V_{RTCcap} pin directly to the ground (V_{SS}) may draw excessive current from the nvSRAM.

For nvSRAM RTC design guidelines and best practices, see application note, *AN61546 - Non Volatile Static Random Access Memory (nvSRAM) Real Time Clock (RTC) Design Guidelines and Best Practice*.

I²C nvSRAM Operation

The I²C nvSRAM access is always in byte format and every byte put on the SDA line must be 8 bits long. The number of bytes that can be transmitted per transfer is unrestricted; therefore it supports burst mode writes and reads. Each byte has to be followed by an Acknowledge (A) bit. Data is transferred with the most significant bit (MSb) first and the least significant bit (LSb) last in each byte transfer. Figure 8 shows the I²C nvSRAM data transfer.

Figure 8. I²C nvSRAM Data Transfer



The I²C nvSRAM data transfers follow the format shown in Figure 9. After the START condition (S), a slave address is sent. This address is 7 bits long followed by 8th bit, which is a data direction bit (R/$\overline{W}$). If bit (R/$\overline{W}$) is set to '0' it indicates a transmission (WRITE), if this bit (R/$\overline{W}$) is set '1' it indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) instead and address the slave again or communicate with the other slave devices without generating a STOP condition. All standard I²C modes except the high-speed mode will follow the data format as described in Figure 9.

Figure 9. I²C Data Byte Format (Sm, Fm, Fm+)

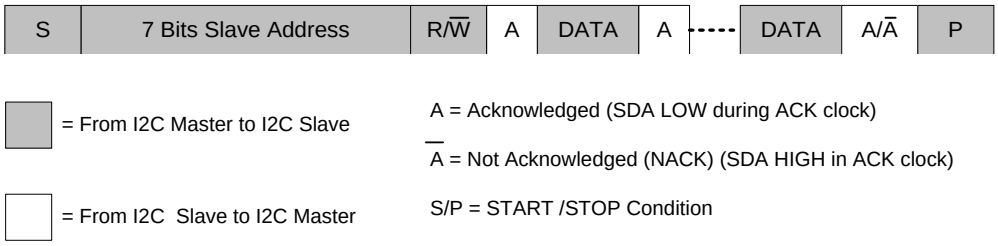


Figure 10. Data Byte Format (Sm, Fm, Fm+) – Write Operation

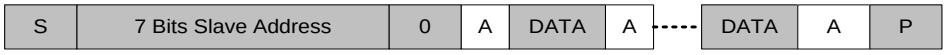
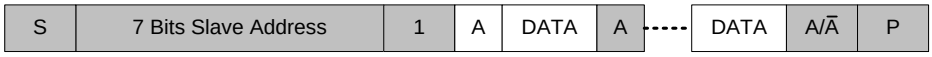


Figure 11. Data Byte Format (SM, FM, Fm+) - Read Operation



High-Speed Mode (Hs-mode) Operation

In Hs-mode the nvSRAM can transfer data at bit rates up to 3.4 Mbit/s. After the START condition (S) is generated, an 8-bit master code (0000 1XXXb) is sent for which nvSRAM sends the NACK ($\overline{A}$) but put the data interface in Hs-mode for all subsequent operations. The device exits Hs-mode only after following the STOP (P) condition. After putting the slave in Hs-mode, the I²C master transmit 7 bits slave address followed by 8th bit, which is a data direction bit (R/$\overline{W}$). If bit (R/$\overline{W}$) is set to '0' it indicates a transmission (WRITE), if this bit (R/$\overline{W}$) is set '1' it indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus in Hs-mode, it can generate a repeated START condition (Sr) and address the slave without generating a STOP condition.

Figure 12. I²C Data Byte Format (Hs)

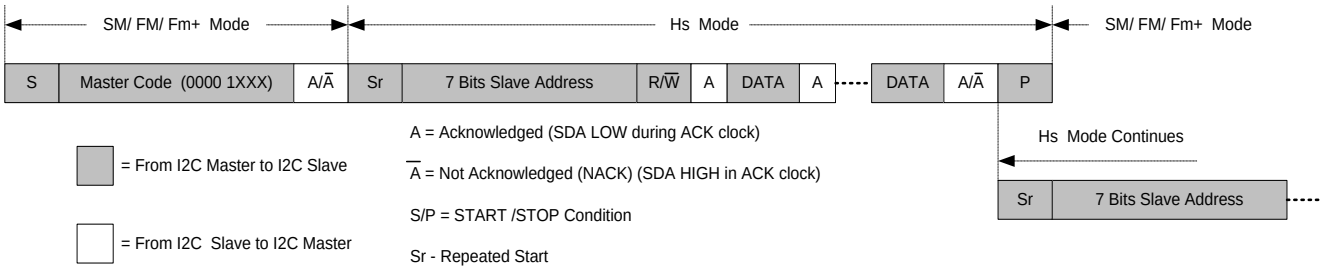


Figure 13. I²C Data Byte Format (Hs) – Write Operation

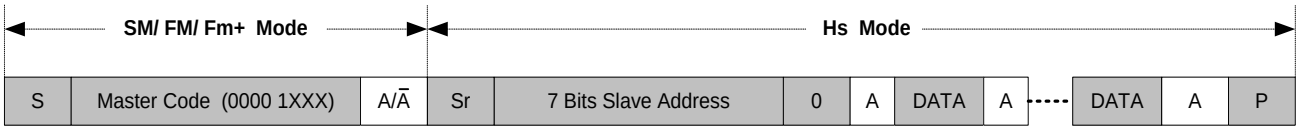
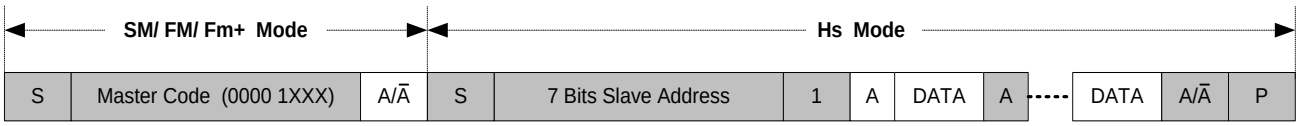


Figure 14. I²C Data Byte Format (Hs) – Read Operation



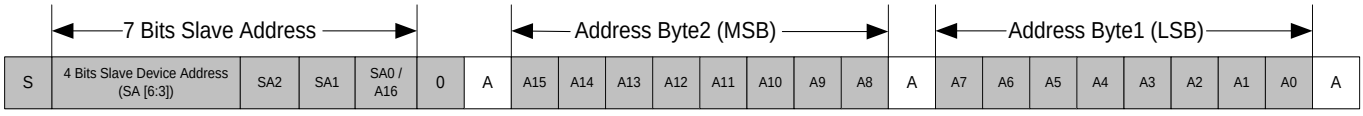
Addressing in I²C nvSRAM

An I²C master controller communicates with the I²C nvSRAM slave on byte-by-byte basis and always transmits the most significant bit in the first clock cycle and the least significant bit in the 8th clock cycle during a byte transmission. This holds good for all I²C communication including command, address, and data bytes. Similarly, when an I²C nvSRAM transmits the data byte during read operation, it always transmits the most significant bit first and the least significant bit last.

Figure 15 shows an example of address bits being transmitted over the I²C bus.

The 7 bits long slave address is represented by acronym "SA [6:0]" to differentiate it from memory address bits, which use acronym "A [16:0]". Henceforth all the follow on sections will show slave address bits as SA [x].

Figure 15. Address Bits Transmission in I²C nvSRAM



Slave Device Address

The I²C nvSRAM slave supports 7 bits slave addressing SA [6:0] of which four most significant address bits SA [6:3] are fixed in the device and not alterable by the user. The remaining three least significant address bits SA [2:0] are configurable through external address pins (A2, A1, A0) provided on the device. The I²C nvSRAM offers three different functions as data Memory, RTC function, and other Controls in a single device. The I²C nvSRAM assigns three unique slave IDs by fixing upper 4 bits (see Table 4) of the slave address SA [6:3] to allow an I²C master to access these functions.

Table 4. Slave Device Address

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	nvSRAM Function Select
1	0	1	0	Device select ID		A16/ SA0/ X	R/$\overline{W}$	Selects Memory
1	1	0	1	Device select ID		X	R/$\overline{W}$	Selects RTC Registers
0	0	1	1	Device select ID		X	R/$\overline{W}$	Selects Control Registers

CY14X101I Slave Devices

- Memory, 128 K × 8
- RTC Registers, 16 × 8
- Control Registers
  - Memory Control Register, 1 × 8
  - Serial Number, 8 × 8
  - Device ID, 4 × 8
  - Command Register, 1 × 8

If any other slave ID on the bus matches with the upper four slave address bit SA [6:3], then the user must configure lower slave address bits SA [2:0] differently so that it each slave device sharing the same system bus has been assigned an unique slave ID.

Figure 16. Slave Device Address Select

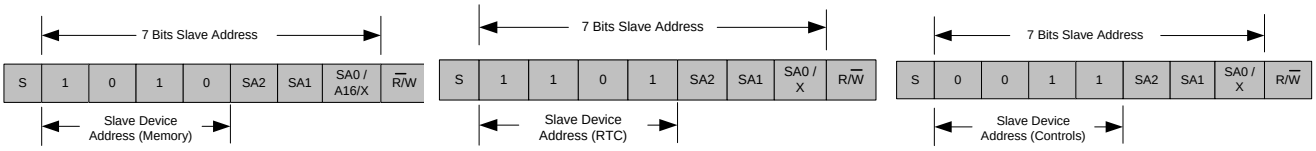


Table 5. I²C nvSRAM addressing for SRAM Write and Read

Density	Slave Address Byte								Address Byte2 (MSB)								Address Byte1 (LSB)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
64 Kbit	Slave Device Address			SA2	SA1	(SA0) Note3	R/$\overline{W}$	(X) Note2	(X) Note2	(X) Note2	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
256 Kbit	Slave Device Address			SA2	SA1	(SA0) Note3	R/$\overline{W}$	(X) Note2	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
512 Kbit	Slave Device Address			SA2	SA1	(SA0) Note3	R/$\overline{W}$	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1 Mbit	Slave Device Address			SA2	SA1	A16	R/$\overline{W}$	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	

Note 2 Unused bits of the most significant address byte (MSB) are don't care bits and nvSRAM ignores them. However, the best practice is to set the unused address bit locations to '0' in the firmware. This approach makes it easy in upgrading the firmware while moving to a higher density option in future.

Note 3 In some nvSRAM device configurations only two address pins (A2, A1) are provided either due to unavailability of sufficient pins on the package or A0 address bit is used internally. The I²C nvSRAM with 1-Mbit density requires 17 address bits A [16:0] to map its entire memory location. Therefore, the slave address space SA0 is used to transmit the A16 address bit in these parts and allows only 2 bits SA [2:1] to configure the slave address externally. In lower density devices (512 Kbit and below) where A0 is not available on the package due to shortage of pins, this bit becomes don't care ('X') internally. The I²C slave device with slave address bit A0 as don't care will acknowledge for two slave addresses (for A0=0 and A0=1) sent by the I²C master.

I²C nvSRAM Access

All I²C nvSRAM functions including standard (memory writes and reads) and special (NV Operations, Device ID, and Serial Number) are accessed through standard I²C write and read protocols. Figure 17 to Figure 21 shows a simplified flow diagram explaining I²C nvSRAM write and read operations. The device datasheet should be referred for detailed description on each I²C nvSRAM functions and their implementation details.

Figure 17. Simplified Flow Diagram for I²C nvSRAM Data Memory Write

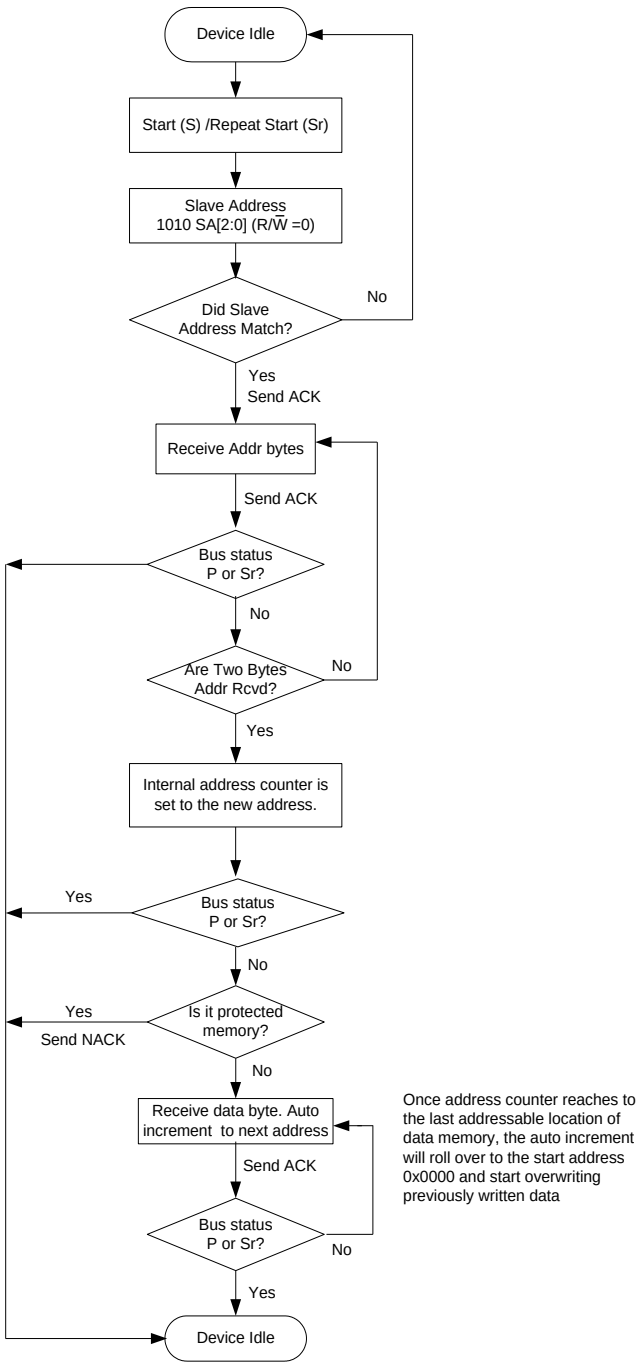


Figure 18. Simplified Flow Diagram for I²C nvSRAM RTC and Control Registers Write

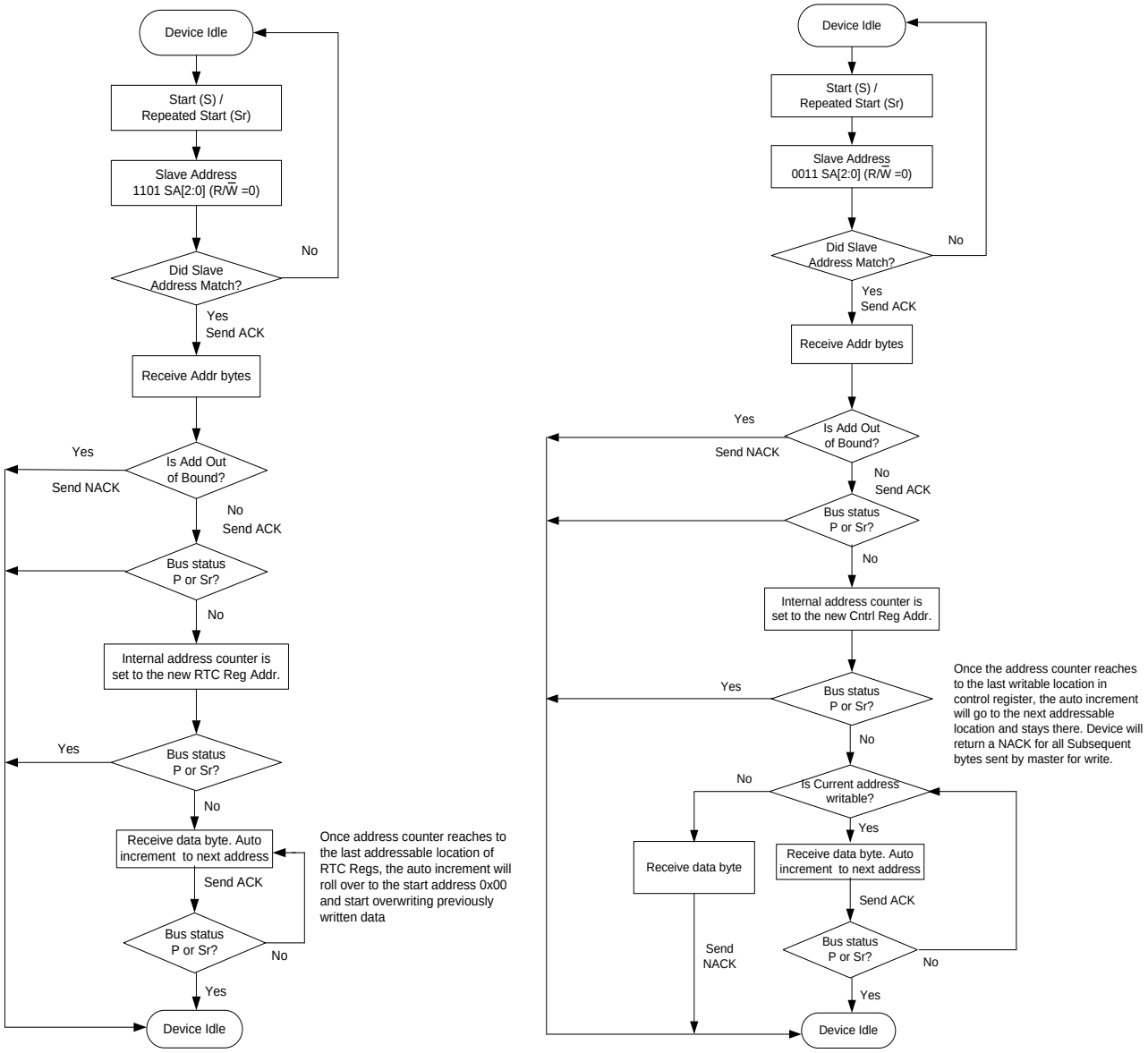


Figure 19. Simplified Flow Diagram for I²C nvSRAM Data Memory Current Address Read

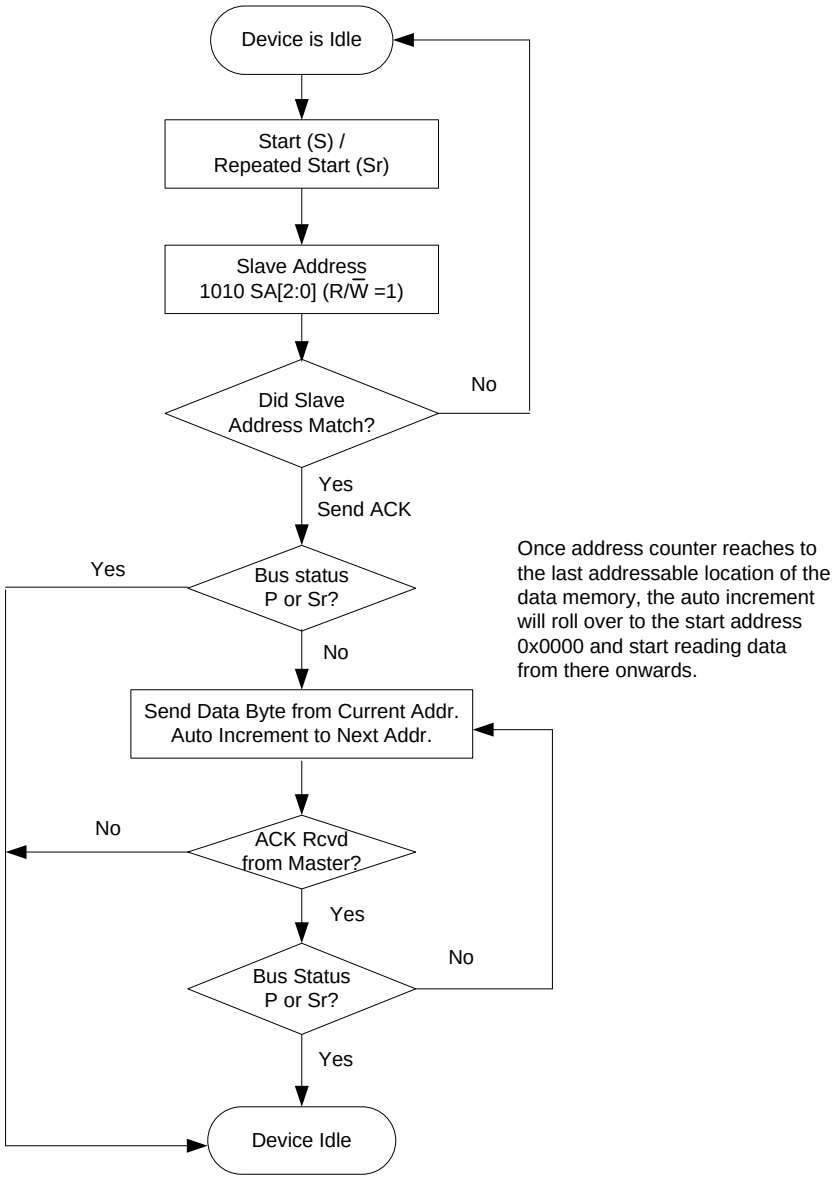
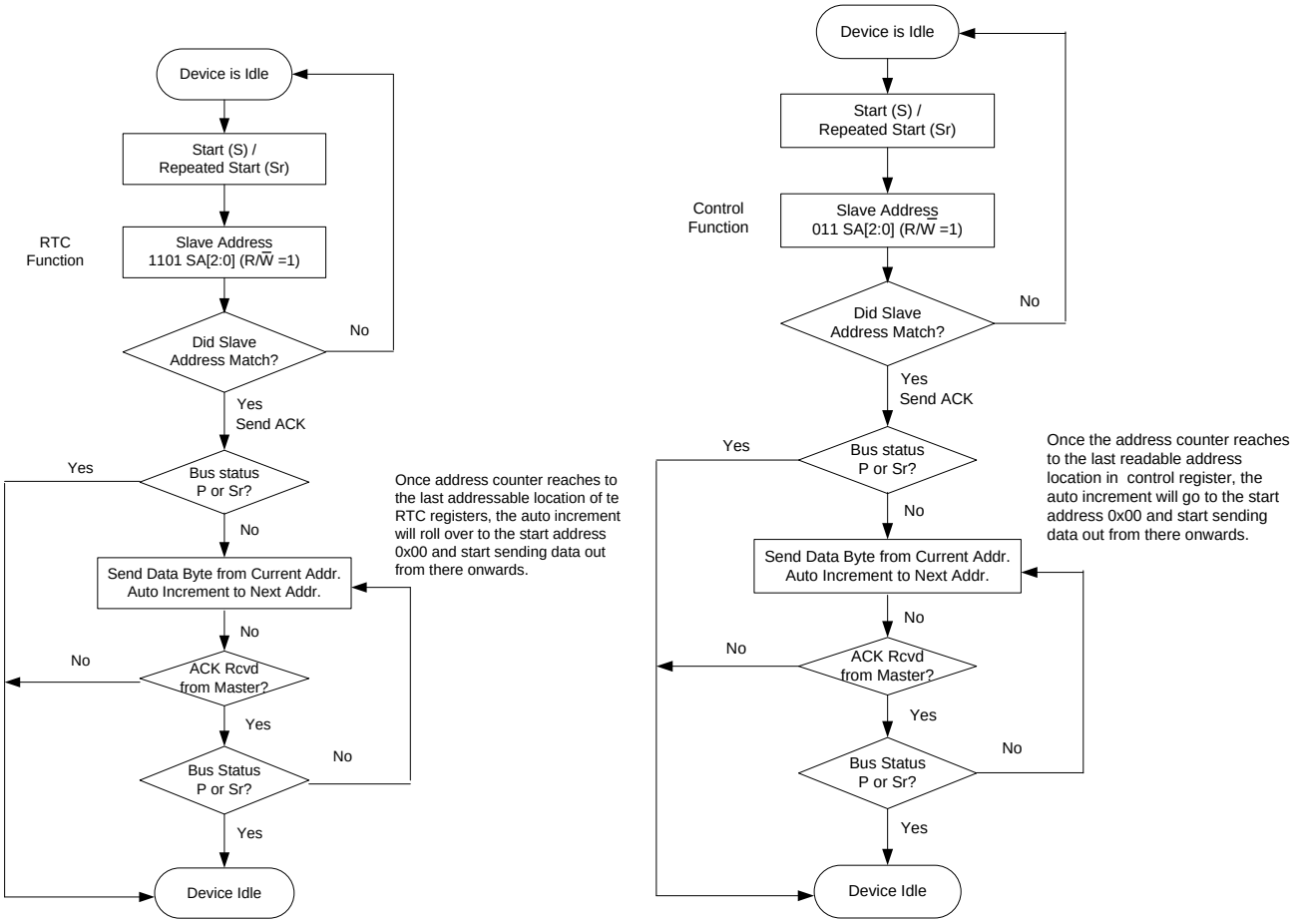


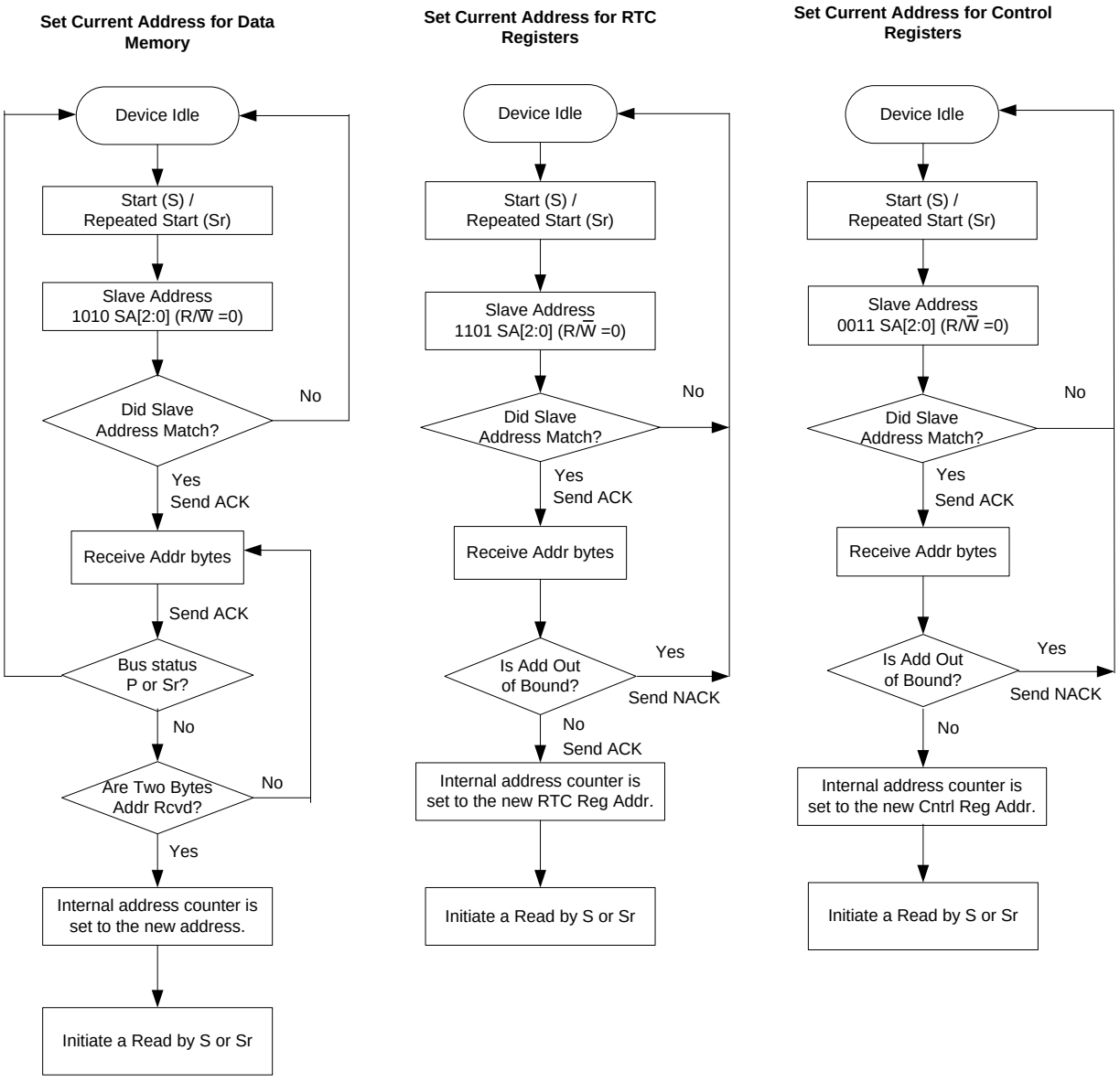
Figure 20. Simplified Flow Diagram for I²C nvSRAM RTC and Control Register Current Address Read



The flow charts as shown in Figure 19 and

Figure 20 are for the current location read from data memory, RTC register and control register. The current location is the address in the address counter at the time of exiting the previous write or read operations. In case if the user needs to read from a different location then the user must set the address counter with a new address by performing a write cycle as shown in Figure 21.

Figure 21. Simplified Flow Diagram for I²C nvSRAM Data Memory, RTC, and Control Register Random Address Read



Summary

Cypress I²C nvSRAM supports the standard I²C access protocols similar to any other nonvolatile I²C memory products. This makes the nvSRAM compatible to all I²C master controllers and reduces the system development cycle time. This application note demonstrates how to configure the I²C nvSRAM in an application with the help of schematics and timing diagrams.

Appendix A (Pseudo Code Example)

I²C Write

```
/*Sm, Fm, Fm+ Mode*/

void I2C_Write_nvSRAM(BYTE slave_Addr,BYTE Addr_MSB, BYTE Addr_LSB, BYTE *Data, int
n_Byte)
{
    int i=0;
    BYTE txBuffer[2];

      txBuffer[0]=Addr_MSB; //Copy I2C slave address in local buffer
       txBuffer[1]=Addr_LSB;
      I2CHW_ClrWrStatus(); //Clear the status register of I2C master
      I2CHW_fSendStart(slave_Addr, I2CHW_WRITE); //Returns a non zero if slave device
ACKs
       while(!I2CHW_bReadI2CStatus() & I2CHW_WR_COMPLETE); //Wait till all bits are
transmitted

    for (i=0;i<n_Byte; i++){
            I2CHW_fWrite( Data[i]); //Master transmit data bytes
             while (!I2CHW_bReadI2CStatus() & I2CHW_WR_COMPLETE);
           }
    I2CHW_SendStop (); // Master sends S/Sr to terminate write
}
```

```
/*Hs Mode*/

void I2C_Write_HSMODE_nvSRAM(BYTE slave_Addr,BYTE Addr_MSB, BYTE Addr_LSB, BYTE *Data,
int n_Byte)
{
    int i=0;
    BYTE txBuffer[2];

       txBuffer[0]=Addr_MSB;
       txBuffer[1]=Addr_LSB; //Copy I2C slave address in local buffer
      I2CHW_ClrWrStatus(); //Clear the status register of I2C master
    //0x00001xxx is a HS mode address hence. (Read/Write also don't care).HS mode command
byte can be set anything from 0x08 to 0x0F
          I2CHW_fSendStart( 0x04, I2CHW_WRITE); //No ACK from any slave.
          while(!I2CHW_bReadI2CStatus() & I2CHW_WR_COMPLETE);//Wait till all bits are
transmitted
          I2CHW_fSendRepeatStart(slave_Addr, I2CHW_WRITE); //Send repeat start with slave
ID to access a slave in HS mode.
         while(!I2CHW_bReadI2CStatus() & I2CHW_WR_COMPLETE);

    for (i=0;i<n_Byte; i++){
            I2CHW_fWrite( Data[i]);
             while(!I2CHW_bReadI2CStatus() & I2CHW_WR_COMPLETE);
          }
      I2CHW_SendStop (); //Master sends S/Sr to terminate write
}
```

I²C Read

```
/*Sm, Fm, Fm+ Mode*/
void I2C_Read_nvSRAM(BYTE slave_Addr, int n_Byte)
{
 int i=0;
 BYTE dataRD;

  I2CHW_ClrWrStatus();//Clear the status register of I2C master
  I2CHW_fSendStart( slave_Addr, I2CHW_READ);
  while(!I2CHW_bReadI2CStatus() & I2CHW_RD_COMPLETE);

 for(i=0;i<n_Byte; i++)  {
  if(i==(n_Byte-1)) {
    dataRD =I2CHW_bRead (I2CHW_NAKslave); //Master sends NACK for the last read to terminate
the Read
     while(!I2CHW_bReadI2CStatus() & I2CHW_RD_COMPLETE); //Wait till all bits Rcvd
        }
  else   {
           dataRD =I2CHW_bRead (I2CHW_ACKslave);
           while(!I2CHW_bReadI2CStatus() & I2CHW_RD_COMPLETE);
         }
     }
 I2CHW_SendStop (); //Master sends S/Sr to terminate Read
}
```

```
/*Hs Mode*/

void I2C_Read_HSMODE_nvSRAM(BYTE slave_Addr, int n_Byte)
{
 int i=0;
 BYTE dataRD;

  I2CHW_ClrWrStatus();//Clear the status register of I2C master
  I2CHW_fSendStart( 0x04, I2CHW_READ); //0x0000 1xxx is a HS mode address hence slave addr
can be 0x0X. No ACK from any slave.
  while(!I2CHW_bReadI2CStatus() & I2CHW_RD_COMPLETE);//Wait till all bits received
  I2CHW_fSendRepeatStart( slave_Addr, I2CHW_READ); //Send repeat start with slave ID to
access a slave in the HS mode.
  while(!I2CHW_bReadI2CStatus() & I2CHW_RD_COMPLETE);

  for(i=0;i<n_Byte; i++){
      if(i==(n_Byte-1)) {
          dataRD =I2CHW_bRead (I2CHW_NAKslave);
          while(!I2CHW_bReadI2CStatus() & I2CHW_RD_COMPLETE);
        }
      else {
           dataRD =I2CHW_bRead (I2CHW_ACKslave);
           while(!I2CHW_bReadI2CStatus() & I2CHW_RD_COMPLETE);
         }
     }
 I2CHW_SendStop //Master sends S/Sr to terminate Read
}
```

Document History

Document Title: Designing with Serial I²C nvSRAM - AN74875

Document Number: 001-74875

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3471498	ZSK	12/29/2011	New document
*A	3466105	ZSK	01/05/2012	Removed incorrect document
*B	3524014	ZSK	02/13/2012	Reactivated spec
*C	3724929	ZSK	08/27/2012	Attached PSoC 3 based I²C nvSRAM component example project
*D	3918328	ZSK	03/01/2013	No update to the App note contents Changed the PSoC 3 component library name from “nvSRAM_I2C” to “nvRAM_I2C” Added APIs to access the RTC registers in the PSoC 3 example project Made an enhancement in the PSoC 3 example project to add user select options for memory density, RTC/non RTC, and nvSRAM /FRAM
*E	4234992	ZSK	01/08/2014	Added Software Version as “PSoC® Creator™ 3.0 or above”. Updated Abstract. Updated I²C nvSRAM Configurations: Updated I²C nvSRAM Device Options: Updated Table 2. Updated Determining I2C Pull-up Resistor Values: Updated Figure 6. Completing Sunset Review. Updated in new template.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com