

AN74875

使用串行 I²C nvSRAM 进行设计

作者: Shivendra Singh

相关项目: 有

相关器件系列: CY14xxxxl、CY14xxxxJ

软件版本: PSoC[®] Creator™ 3.0 或更高版本

相关应用笔记: AN61546、AN43593

AN74875 提供了内部集成电路 (I²C) nvSRAM 器件的设计指南及示例。I²C nvSRAM 是一个高性能的非易失性串行接口存储器, 它无需周期延迟写操作, 并能提供无限次 SRAM 写循环。I²C nvSRAM 是一个从设备 I²C 器件。在系统中要通过 I²C 主设备控制器访问它。此外, 还提供了 PSoC 3 相关库组件的示例项目。

目录

简介	1
I ² C nvSRAM 配置	2
I ² C 总线协议特性的可用性	2
I ² C nvSRAM 器件选项	2
I ² C nvSRAM 器件连接	3
确定 I ² C 上拉电阻值	5
控制输入引脚配置	7
RTC 器件的特定引脚配置	8
I ² C nvSRAM 操作	9
高速模式 (Hs 模式) 操作	10
在 I ² C nvSRAM 中进行寻址	10
I ² C nvSRAM 访问	12
总结	16
附录 A (伪代码示例)	17
I ² C 写入	17
I ² C 读取	18
全球销售和 design 支持	20

简介

赛普拉斯 nvSRAM 将一个 SRAM 单元和一个非易失性存储器单元集成到一个单一 nvSRAM 单元内。在正常运行模式下, 可以直接读取和写入 nvSRAM 中的 SRAM 部分。它比其他现有的非易失性存储器技术 (如 EEPROM 和闪存) 的读写访问速度更快。系统掉电时, 通过使用存储在较小电容的能量 (该电容与器件的 V_{CAP} 引脚相连), 将 SRAM 中的数据自动传输到非易失性单元内。在下次通电时, 非易失性单元中的数据被自动回读到 SRAM 阵列内, 以供给用户使用。在正常运行模式下, 与 V_{CAP} 引脚相连的电容由 nvSRAM 充电。

nvSRAM 中的非易失性单元能够进行一百万次擦写周期。如果在 nvSRAM 中进行存储操作, 仅在 SRAM 单元中的数据被传输到非易失性单元内时, 才执行 nvSRAM 擦写周期。在下面情况下, nvSRAM 中的非易失性存储操作被启动: 当器件的电源降至预定义的阈值电平 (V_{SWITCH}) 以下时, 存储操作被自动启动; 向指令寄存器 (0xAA) 写入特定指令; 将硬件引脚 (HSB) 置于低电平。在控制寄存器空间内定义 nvSRAM 的指令寄存器。通过一个专用的 I²C 从设备 ID 寻址控制寄存器。有关 nvSRAM 寻址的详细信息, 请参考本应用笔记中在 I²C nvSRAM 中进行寻址一节。

仅在检测到系统掉电事件, 并要将 SRAM 中刚写入的数据安全地传输到非易失性单元时, nvSRAM 才启动非易失性存储操作。因此, nvSRAM 中的非易失性单元擦写计数总和等于非易失性存储周期, 并不等于 SRAM 写入周期。

许多数据记录应用要求在掉电时立即存储运行时的关键信息。此关键信息包括控制器运行时的状态、暂存器数据、参数设置以及由控制器测量的其他环境向量。凭借快速非易失性写入特性, I²C nvSRAM 特别适合该类数据记录应用。在 nvSRAM 中, I²C 主设备控制器可以在几十微秒内记录数百个数据字节, 而在 EEPROM 或闪存存储器中, 则需要花费几十毫秒的时间写入同样大小的数据。行业标准的 8 引脚 SOIC 和 16 引脚 SOIC 均提供 I²C nvSRAM。

本应用笔记介绍了 I²C nvSRAM 配置、各种封装选项的示例电路、确定 I²C 总线合适的上拉电阻值的方法、nvSRAM 中 I²C 通信的数据字节格式，以及用于访问 nvSRAM 存储器、实时时钟（RTC）和控制函数的 I²C 寻址结构。更多关于 I²C nvSRAM 的详细信息，请参考特定的器件数据手册。

本应用笔记中还包含了一个相关项目—PSoc 3 nvRAM I²C 库组件。

I²C nvSRAM 配置

I²C nvSRAM 支持的 I²C 数据传输速率可达 3.4 Mbits/s（I²C 时钟周期为 3.4 MHz），同时支持其他所有较低频率的访问（如 I²C 总线标准规范所定义）。

- **标准模式（Sm）** — 比特率可达 100 Kbit/s
- **快速模式（Fm）** — 比特率可达 400 Kbit/s
- **增强型快速模式（Fm+）** — 比特率可达 1 Mbit/s
- **高速模式（Hs）** — 比特率可达 3.4 Mbit/s。

所有器件配置都提供了上述四种总线模式（请查看表 1），因此无需器件的任何特殊设置。

表 2. I²C nvSRAM 配置

nvSRAM 器件型号	状态	工作电压 (典型值)	封装	WP 引脚	V _{CAP} 引脚/ 自动存储	(HSB)引脚/ HW 存储	A0 引脚	每个 I ² C 总线上 器件的数量	RTC
CY14CXXXJ	NRND	2.5 V	8 SOIC	有	无/无	无/无	有 ^{注意 1}	4 或 8 ^{注意 1}	无
CY14BXXXJ	NRND	3 V							
CY14EXXXJ	NRND	5 V							
CY14CXXXJ	NRND	2.5 V	8 SOIC	有	有/有	无/无	无	4	无
CY14BXXXJ	NRND	3 V							
CY14EXXXJ	NRND	5 V							
CY14CXXXJ	NRND	2.5 V	16 SOIC	有	有/有	有	有 ^{注意 1}	4 或 8 ^{注意 1}	无
CY14BXXXJ	NRND	3 V							
CY14EXXXJ	NRND	5 V							
CY14CXXXI	联系赛普拉斯	2.5 V	16 SOIC	有	有/有	有	有 ^{注意 1}	4 或 8 ^{注意 1}	有
CY14BXXXI	正在生产	3 V							
CY14EXXXI	正在生产	5 V							

不建议用于新设计

注意 1: 在内部 1 Mbit nvSRAM 器件中使用了最低有效从器件地址空间（A0），因此，在 1 Mbit 密度的选项中，该空间不可用。在所有 512 Kbit 和容量更低的选项（J2 器件除外）中均能够使用 A0 引脚。如果不使用 A0 引脚，则 I²C nvSRAM 的每个 I²C 总线最多只能使用于四个器件。

I²C 总线协议特性的可用性

表 1 总结了标准 I²C 从设备总线规范的所有必要和可选特性。I²C nvSRAM 支持标准 I²C 从器件的所有必要特性。

表 1. I²C 总线协议的可用性

特性	I ² C 规范标准	I ² C nvSRAM
启动条件	必要	√
停止条件	必要	√
确认	必要	√
7 位从设备地址。	必要	√
10 位从设备地址	可选	不提供
时钟延展	可选	无需
通用调用地址	可选	不提供
器件 ID	可选	不提供
软件复位	可选	不提供

I²C nvSRAM 器件选项

赛普拉斯在各种配置和封装选项中支持 I²C nvSRAM，具体如表 2 所示。

I²C nvSRAM 器件连接

图 1 显示的是一个 I²C 单主设备-多从设备的典型配置情况。所有微控制器或可编程的器件都能够作为 I²C 主设备，并能够生成 I²C 主设备协议。同时，任何标准的 I²C 从设备都能作为从设备使用。在图 1 示例中，I²C nvSRAM 作为 I²C 从设备使用。由于某些封装选项中的 512 Kbit 和容量更低的 I²C nvSRAM 支持三个从设备寻址位，因此，可以在一个

I²C 总线上连接八个器件。通过八种不同的组合方式配置从设备选择地址行（A2、A1、A0），可以为每一个从器件分配唯一的从设备 ID。在不使用 A0 的封装配置中，通过配置从设备选择地址引脚 A2 和 A1，在同一个总线上最多只能连接四个从器件。

图 1. 典型的 I²C 主-从设备配置

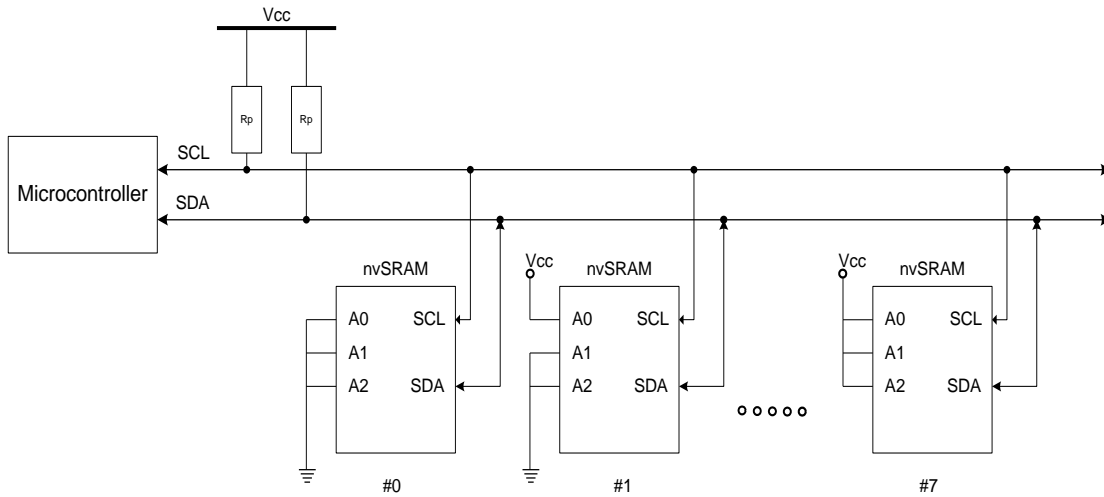
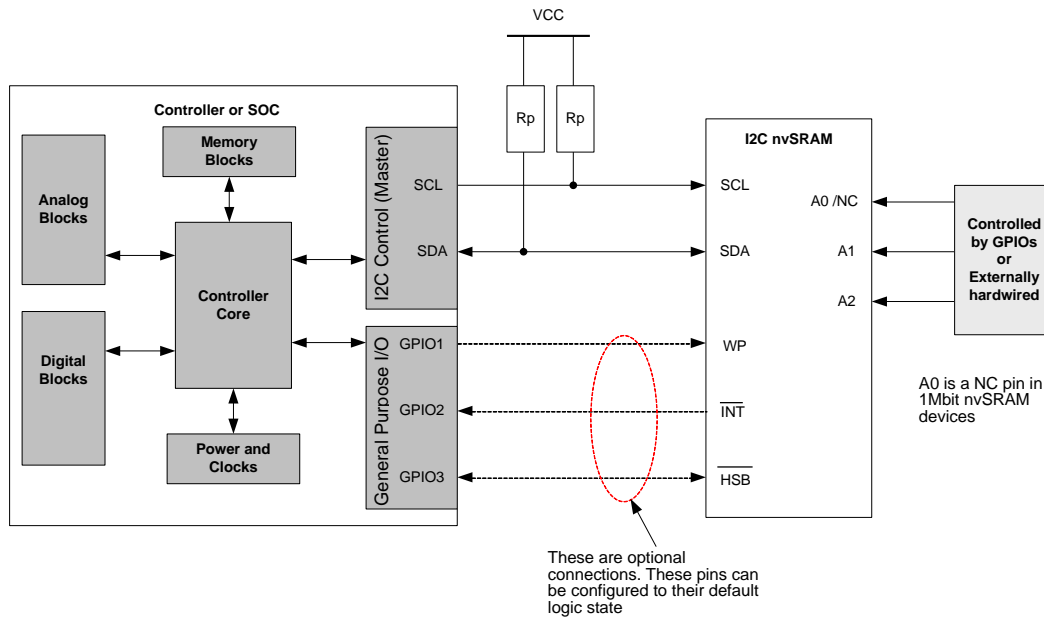


图 2 显示了 I²C nvSRAM 器件的典型系统级配置。对于没有专用 I²C 总线的微控制器，通过“bit banging”技术，可将通用 I/O 端口使用于 SCL 和 SDA。

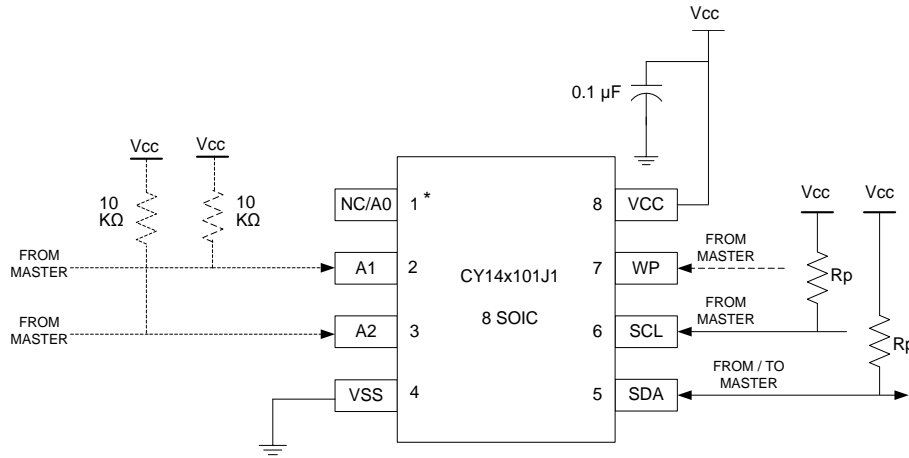
图 2. 典型的 I²C nvSRAM 连接。



样例电路

下图（图 3 至图 5）显示的是 1 Mbit I²C nvSRAM 的详细原理图连接。在所有容量较低的器件（512 Kbit 和容量更低）中，I²C 主设备和 nvSRAM 从设备间的硬件连接保持不变。

图 3. 8 引脚 SOIC 1 Mbit I²C nvSRAM 接口（无 V_{CAP}）



所有可选连接都用虚线显示。

A2 和 A1 上的上拉电阻将从设备地址位 A2 和 A1 设置为 1。如果系统要求将某个地址引脚配置为 0，那么，要移除上拉电阻，并将该引脚设置为未连接状态。通过一个弱下拉电阻可内部将其置于低电平。

图 4. 8 引脚 SOIC 1 Mbit I²C nvSRAM 接口（带有 V_{CAP}）

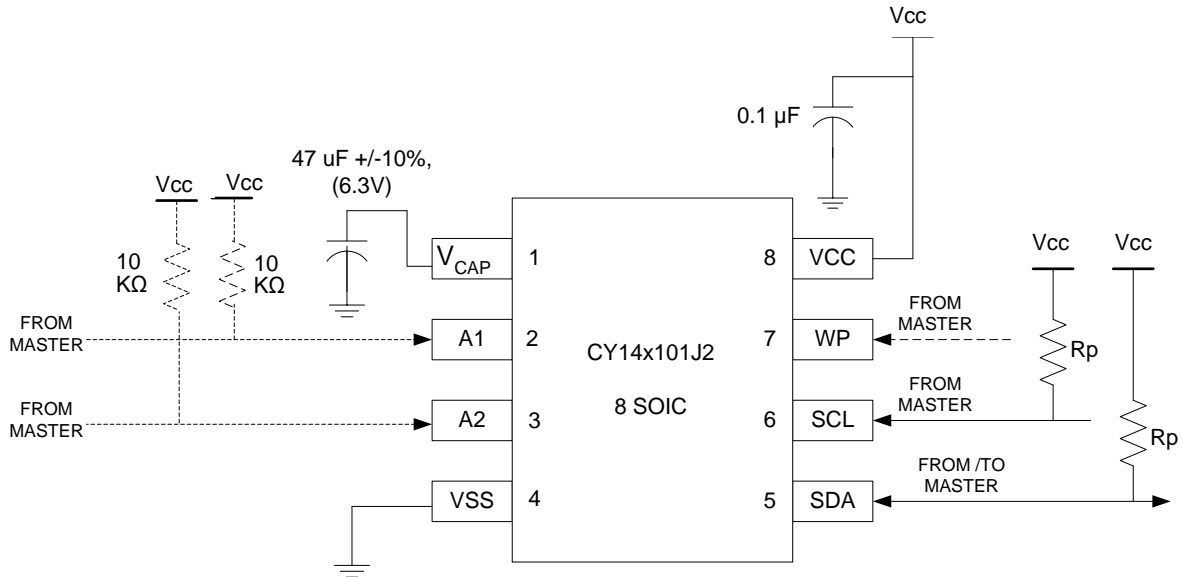
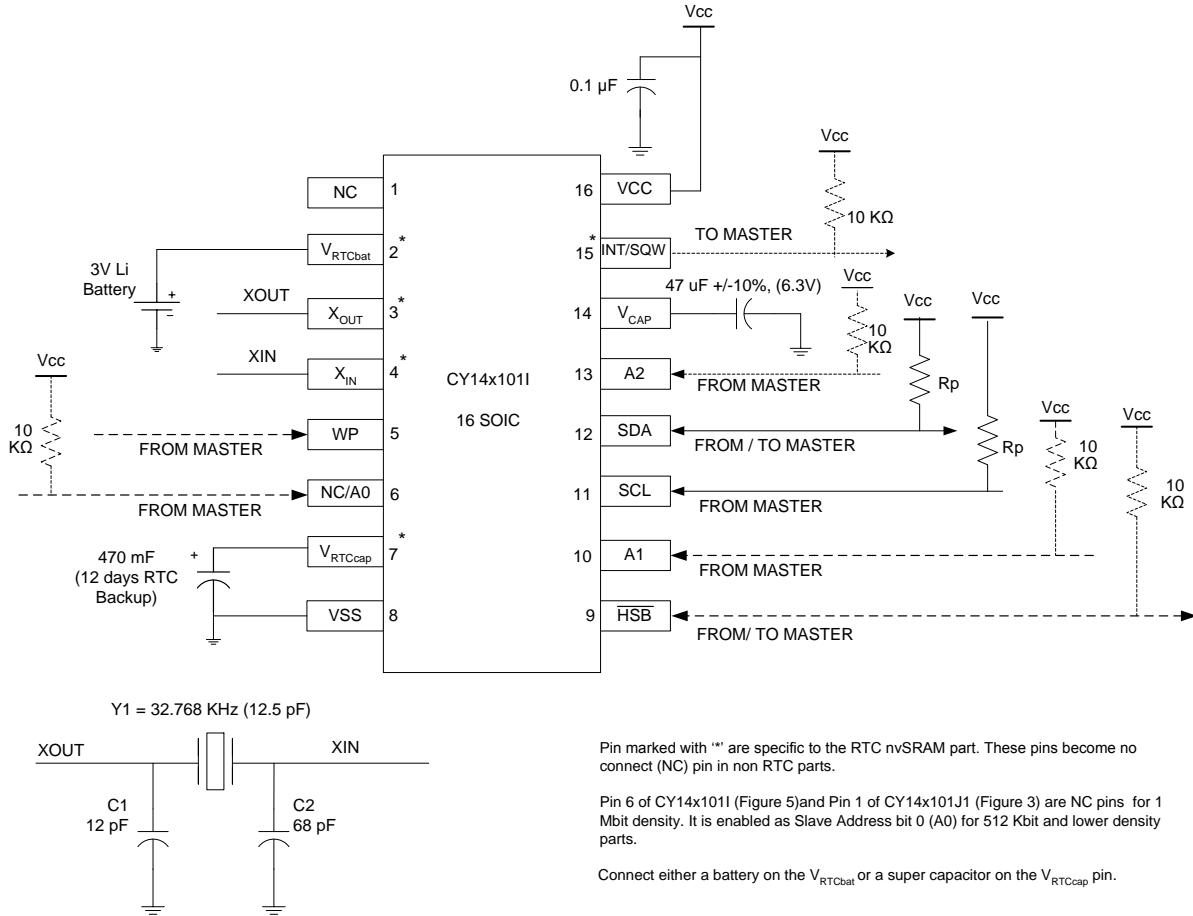


图 5. 16 引脚 SOIC 1 Mbit (RTC) I²C nvSRAM 接口


确定 I²C 上拉电阻值

I²C 总线在 SDA 和 SCL 线上传输数据和时钟。SDA 和 SCL 线都是开漏（即为 TTL 系列中的集电极开路）输出驱动器。因此，I²C 主设备和从器件只能将这两条线驱动为逻辑低电平，或将它们置于开路状态。如果在同一个总线上，没有任何 I²C 器件将该线置于低电平，终端电阻（R_p）会将它上拉到 V_{CC} 电压。开漏驱动器配置支持 I²C 的某些特殊特性，如多主设备配置和从设备的时钟延展。时钟延展是 I²C 标准的一个可选特性，I²C nvSRAM 并不支持该特性，因此，I²C 时钟信号只能作为 I²C nvSRAM 的输入信号。

除了总线电容总和（C_b）外，终端电阻（R_p）也影响 SDA 和 SCL 上信号的时序状况。当 I²C 器件通过开漏驱动器下拉该行时，上拉电阻会在指定的时间内将信号置于高电平。R_p 的值取决于多个电气参数，如：器件的工作电压（V_{CC}）和输出逻辑低电平（V_{OL}）规范，灌电流（I_{OL}）规范、总线负载总和（C_b）和时序参数，如（t_R）规范。

下节介绍的是在已给系统配置中如何确定 I²C 总线的电阻值。

确定 R_p（最大值）

为计算 RC 时间常量，假定 CMOS 逻辑电平的阈值为 V_{IH} = 0.7 V_{CC}（最小值）和 V_{IL} = 0.3 V_{CC}（最大值）。

$V(t) = V_{CC} (1 - e^{-t / RC})$ ，其中“t”是指自从开始充电且 RC 为时间常量的时长。

$V(t_1) = 0.3 \times V_{CC} = V_{CC} (1 - e^{-t_1 / RC})$ ，那么：

$$t_1 = 0.3566749 \times RC \quad \text{公式 1}$$

$V(t_2) = 0.7 \times V_{CC} = V_{CC} (1 - e^{-t_2 / RC})$ ：

$$t_2 = 1.2039729 \times RC \quad \text{公式 2}$$

总上升时间（T）是指给总线电容电压充电，使之从 V_{IL} 增加到 V_{IH} 所需要的时间。

$$T = t_2 - t_1$$

$$= 1.2039729 \times RC - 0.3566749 \times RC$$

$$= 0.8473 \times RC \quad \text{公式 3}$$

公式 3 用来确定连接至 I²C 线时上拉电阻的上限值。表 3 显示的是在各种时序模式下作为总线电容函数的最大 R_p 。在每种模式下， R_p (max) 是最小上升时间 (t_r) 和估计总线电容 (C_b) 的函数：

$$R_p(\text{Max}) = \frac{t_r}{(0.8473 \cdot C_b)} \quad \text{公式 4}$$

总线电容 (C_b) 是线路、连接和引脚的电容总和。

确定 R_p (最小值)

工作电压和灌电流 (I_{OL}) 闲置时上拉电阻的最小值， R_p (min)。 R_p (Min) 的值是 V_{CC} 和 I_{OL} 的函数，如公式 5：

$$R_p(\text{min}) = \frac{V_{CC} - V_{OL}(\text{max})}{I_{OL}} \quad \text{公式 5}$$

选择的 R_p 值必须在指定的上限和下限间。

$$R_p(\text{Min}) \leq R_p \leq R_p(\text{Max}) \quad \text{公式 6}$$

低功耗设计应优先使用该范围内较高的值，以限制电流消耗。

表 3 显示的是在已给的总线负载条件下和工作电压范围内 R_p (最小、最大) 的值。 可从公式 4 和 5 得到表 3 中尚未显示的数值，用以计算 R_p (Max) 和 R_p (Min)。

表 3 中的阴影区域表示在已给的工作电压条件下，并对于某些总线负载 (C_b)， R_p (最小值) 超过了 R_p (最大值)。由于 R_p (Min) 不能超过 R_p (Max)，因此在 I²C 总线上使用的最大电容负载将受限制。

例如，如果将某个 3 V 的器件配置为最小 V_{CC} 供电 ($V_{CC} = 2.7 \text{ V}$)，那么，运行于下述总线模式时，系统不能超过 SCL 和 SDA 线上的以下负载（单位为皮法）：

$$S_m = C_b \leq 550 \text{ pF}; 0.77 \text{ k}\Omega \leq R_p \leq 2.15 \text{ k}\Omega$$

$$F_m = C_b \leq 450 \text{ pF}; 0.77 \text{ k}\Omega \leq R_p \leq 0.79 \text{ k}\Omega$$

$$F_m+ = C_b \leq 150 \text{ pF}; 0.77 \text{ k}\Omega \leq R_p \leq 0.94 \text{ k}\Omega$$

$$H_s = C_b \leq 100 \text{ pF}; 0.77 \text{ k}\Omega \leq R_p \leq 0.94 \text{ k}\Omega$$

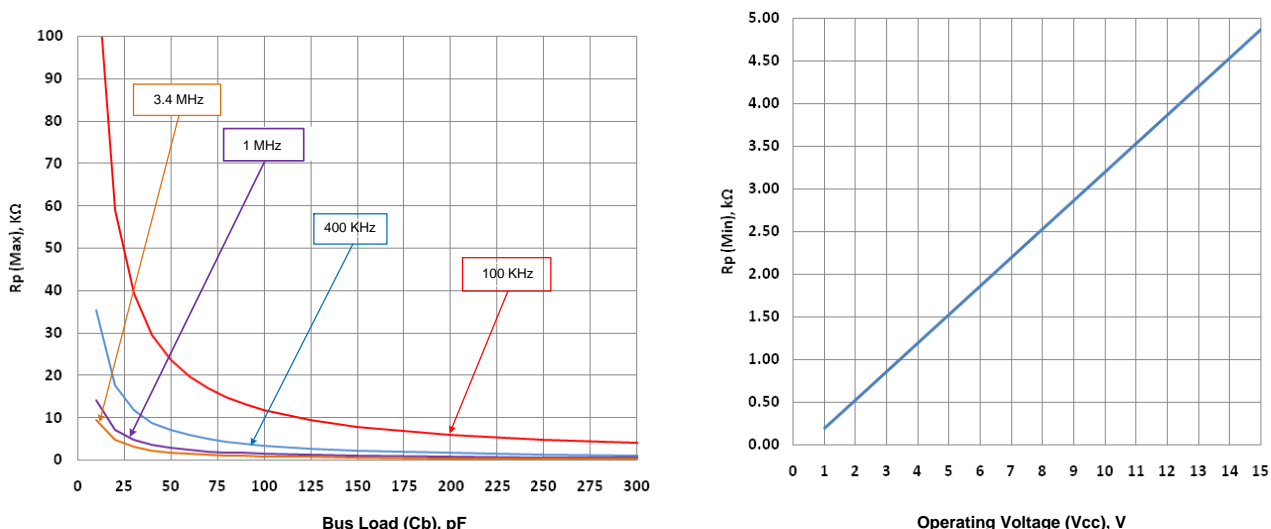
同样也可从表 3 和图 6 获取其他工作电压和工作频率条件下的最大总线负载 (C_b) 和 I²C 上拉电阻 (R_p) 值。

表 3. 各种总线负载和工作电压条件下的 R_p 的（最小、最大）值

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

图 6. 各种总线负载和工作电压条件下的 R_p（最小、最大）值


控制输入引脚配置

I²C nvSRAM 有很多控制输入引脚。为使器件正常运行，需要将这些引脚正确设置为固定的逻辑状态（高电平或低电平）。如果不将某个输入控制引脚设置为合适的逻辑电平（高电平或低电平），而让它进入悬空状态，则该悬空引脚可能被置于中间的亚稳定状态，因而不能确保器件的行为。因此，必须将所有没有内部上拉或下拉选项的未使用输入引脚连接至由上拉或下拉电阻外部设置的合适的逻辑电平。可以使用的电阻值范围为 1 kΩ 至 10 kΩ。

WP 引脚：

WP 引脚为高电平有效引脚，从写操作中保护整个存储器和所有的寄存器。当该引脚处于高电平时，所有的存储器和寄存器写入都被禁止，且地址计数器不会增加。I²C nvSRAM 为该引脚提供了一个内部下拉电阻。因此，如果未使用写保护功能，则可将该引脚置于悬空状态（未连接）。如果将该引脚连接到一个控制器 I/O 以进行外部控制，则建议使用一个外部上拉电阻，用以防止由该线上的噪声引起的意外触发。外部上拉电阻可使用值范围为 1 kΩ 至 10 kΩ 的电阻。

A2、A1、A0 引脚：

它们都是从设备地址引脚，用于在多个从设备配置中配置不同的从器件上的各个从设备地址。这些引脚被内部设置为低电平，因此，如果未使用的话，可将它们置于悬空状态（尚未连接）。为将这些引脚配置为逻辑高电平状态，要将它们连接至外部上拉电阻，或直接将其连接至 V_{CC} 电源。可以使用值范围为 1 kΩ 至 10 kΩ 的上拉电阻。在系统要求动态更改从设备地址的某些配置中，要求将这些地址引脚连接至控制器 I/O，以随时配置从设备选择地址引脚（A2、A1、A0）并访问器件。

HSB 引脚：

HSB 引脚是 nvSRAM 中的双向引脚。作为输出引脚时，它指示在正常运行时 nvSRAM 的就绪或忙碌状态。当器件通电或正在执行某个非易失性存储周期时，器件会将 HSB 引脚置于低电平状态，以指示忙碌状态。当 HSB 引脚处于高电平状态时，则表示器件准备好执行普通的读和写操作。作为输入引脚时，可通过控制器将 HSB 置于低电平，用以外部启动硬件存储。如果不与任何 GPIO 相连，该引脚会悬空。I²C nvSRAM 在 HSB 引脚上提供一个内部弱上拉电阻，以在正常运行时保持该引脚为高电平状态。如果将该引脚连接至一个控制器 I/O 以进行外部控制，建议使用一个外部上拉电阻，以阻止由该线上的噪声引起的意外触发。可以使用值范围为 1 kΩ 至 10 kΩ 的电阻。

V_{CAP}：

掉电时，V_{CAP} 引脚上所连接的电容给 nvSRAM 供电，用以传输 SRAM 非易失性单元内的数据。在正常操作时，器件从 V_{CC} 接收电流，以给 V_{CAP} 上的电容充电。nvSRAM 器件使用 V_{CAP} 上存储的电荷执行单一存储操作。如果 V_{CC} 引脚上的电压降至 V_{SWITCH} 以下，器件自动将 V_{CAP} 引脚和 V_{CC} 隔离开，并同时使用 V_{CAP} 上存储的电荷启动存储操作。

为成功执行自动存储操作，必须在 V_{CAP} 引脚上连接合适的电容。应该在器件数据手册中所规定的范围内选择电容值。如果错误选择电容值，将会导致器件故障。欲了解有关 nvSRAM 产品的电容选择指南的信息，请参考应用笔记 *赛普拉斯 nvSRAM 的存储电容选项 — AN43593*。

RTC 器件的特定引脚配置

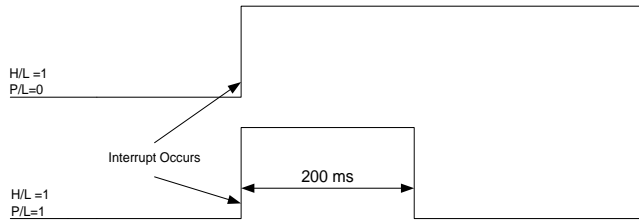
在封装中，RTC 特性要求以下各额外引脚。要准确配置这些引脚，使 RTC 正常工作。

INT 引脚：该引脚是 RTC 器件中的输出引脚。RTC nvSRAM 提供了各种功能，如警报、看门狗定时器、校准时钟输出以及方波发生器。根据 RTC 寄存器的设置和在 nvSRAM 中定义的功能优先级，可复用 INT 输出以指示各个功能的状态/输出。INT 引脚是一个可配置的驱动器输出。在 I²C 从器件中，通过设置中断状态/控制寄存器 (0x06) 中的 ‘H/L’ 位，可以配置 INT 引脚的输出模式。当将 H/L 位设置为 ‘1’ 时，INT 输出被配置为高电平有效，且驱动模式为推挽式。当将 H/L 位设置为 ‘0’ 时，INT 输出驱动被配置为低电平有效的开漏输出。因此，在器件不驱动该输出时，要求使用一个外部上拉电阻将它置为逻辑高电平状态。当使用处于低电平有效模式的 INT (H/L 位被设置为 ‘0’) 时，必须使用一个外部上拉电阻 (值范围为 1 kΩ 至 10 kΩ) 将 INT 上拉到 V_{CC} 电平。

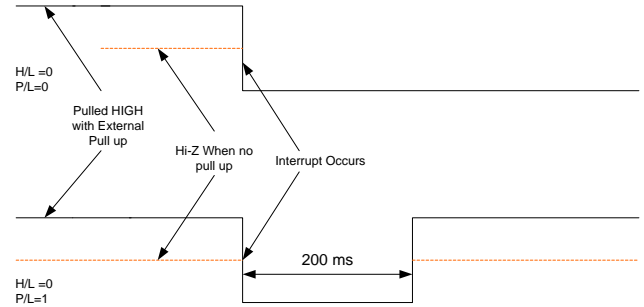
根据 H/L 和 P/L 的不同设置，INT 引脚行为如下：当发生中断时，H/L 位的设置确定 INT 引脚输出处于高电平还是低电平状态。同样，P/L 的设置确定 INT 引脚是脉冲还是电平。图 7 显示的是 I²C nvSRAM 中断引脚 (INT) 的行为。

图 7. INT 引脚行为 (RTC)

H/L 被设置为 ‘1’



H/L 被设置为 ‘0’



V_{RTCbat} 和 V_{RTCcap} 引脚：这些引脚用于给 RTC 电路提供备份电源，以确保在系统电源 (V_{CC}) 断电时振荡器时钟持续运行。为在断电时对 RTC 振荡器进行电源备份，则要将 V_{RTCbat} 连接至某个非充电器件，或将它连接至 V_{RTCcap} 引脚上的一个超级电容。如果不被使用，这些引脚将悬空。

注意：V_{RTCcap} 引脚不能与 V_{SS} 直接短接，因为在正常运行模式下，该引脚用于充电与它相连的超级电容。因此，V_{RTCcap} 引脚直接接地 (V_{SS}) 时，nvSRAM 会消耗极大电流。

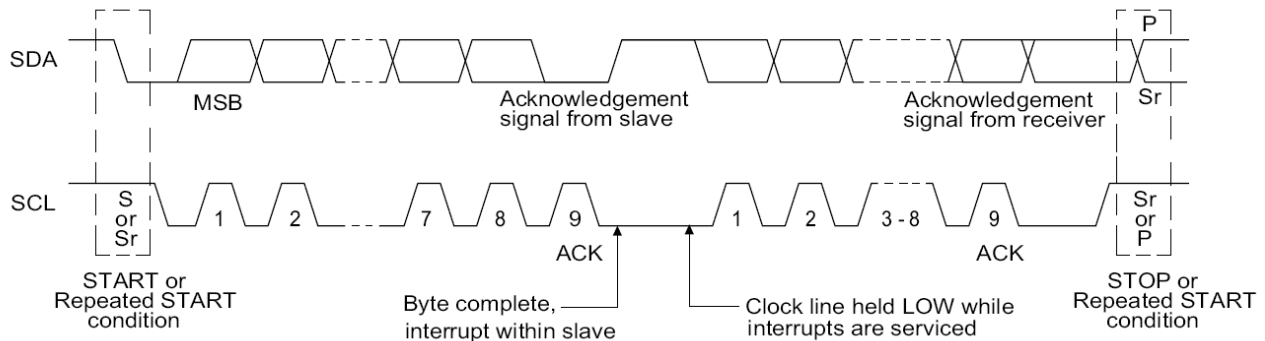
欲了解有关 nvSRAM RTC 设计指南和最佳实践的详细信息，请参考应用笔记 *非易失性静态随机存取存储器 (nvSRAM) 实时时钟 (RTC) 的设计指南和最佳实践* — [AN61546](#)。

I²C nvSRAM 操作

必须始终以字节格式访问 I²C nvSRAM，并且放置在 SDA 线上的每个字节的长度都必须为 8 位。每次传输能发送无

限制的字节数量；因此，该器件支持连续读和写操作。每个字节后面都需要一个确认 (A) 位。传输每个字节时，要求先传输最高有效位 (MSb)，最后传输最低有效位 (LSb)。图 8 显示了 I²C nvSRAM 的数据传输。

图 8. I²C nvSRAM 数据传输



I²C nvSRAM 按照图 9 中显示的格式传输数据。在启动条件 (S) 发生后，将发送一个从器件地址。该地址的长度为 7 位，其后面是数据方向位 (R/W) (第 8 位)。如果位 (R/W) 被设置为 '0'，它表示传输 (写入)；如果该 (R/W) 位被设置为 '1'，它表示一个数据请求 (读取)。

数据传输总是以一个由主设备生成的停止条件作为终止标志。然而，如果某个主设备仍希望在总线上进行通信，则它可生成一个重复的启动条件 (Sr)，并再次寻址从器件，或者与其他从器件进行通信，而不生成停止条件。所有的标准 I²C 模式 (高速模式除外) 都遵循图 9 中描述的数据格式。

图 9. I²C 数据字节格式 (Sm、Fm、Fm+)

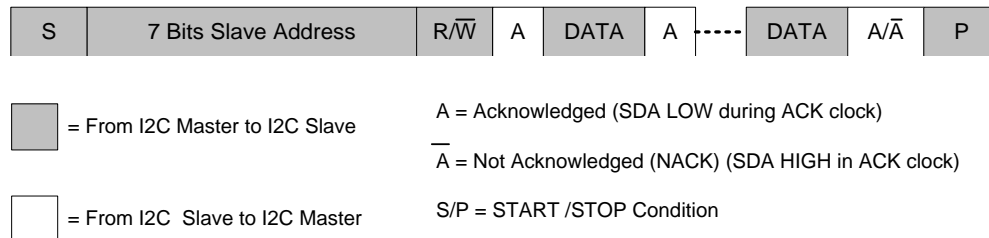
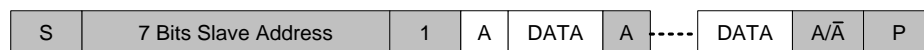


图 10. 数据字节格式 (Sm、Fm、Fm+) — 写入操作



图 11. 数据字节格式 (SM、FM、Fm+) — 读取操作



高速模式 (Hs 模式) 操作

在 Hs 模式下，nvSRAM 的数据传输比特率可达 3.4 Mbit/s。生成启动条件 (S) 后，将发送一个 8 位的主设备代码 (0000 1XXXb)，nvSRAM 随之会发送一个 NACK (\bar{A})，但在所有后续操作中，它都会将数据接口置于 Hs 模式。仅在出现停止 (P) 条件时，器件才会退出 Hs 模式。将从设备置于 Hs 模式后，I²C 主设备将发送 7 位从设备地址，之

后便是数据方向位 (R/\bar{W}) (第 8 位)。如果位 (R/\bar{W}) 被设置为 ‘0’，它表示传输 (写入)，如果该 (R/\bar{W}) 位被设置为 ‘1’，它表示一个数据请求 (读取)。数据传输总是以一个由主设备生成的停止条件作为终止标志。然而，在高速模式下，如果主设备仍希望在总线上进行通信，则它可生成一个重复的启动条件 (Sr)，并对从设备进行寻址，而不生成停止条件。

图 12. I²C 数据字节格式 (Hs)

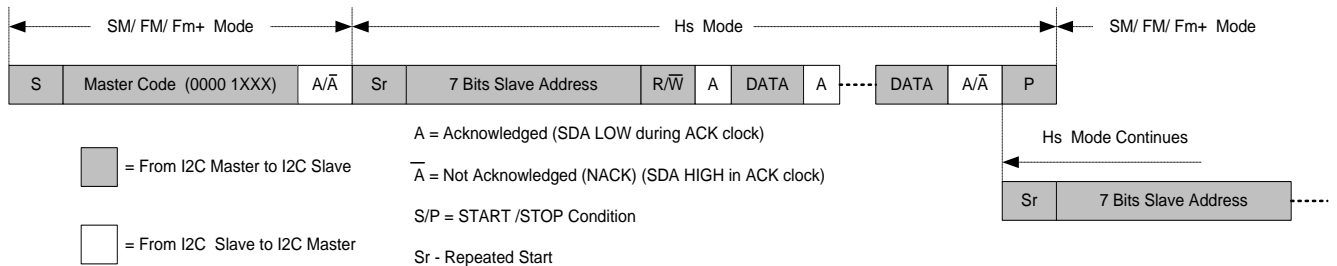


图 13. I²C 数据字节格式 (Hs) —写入操作

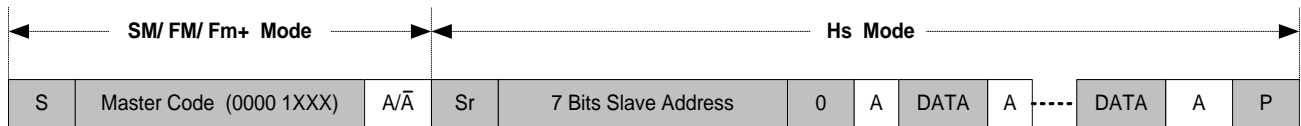
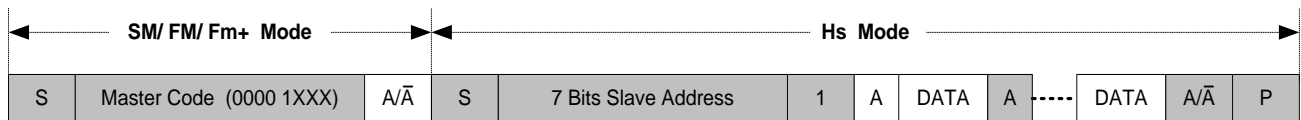


图 14. I²C 数据字节格式 (Hs) —读取操作



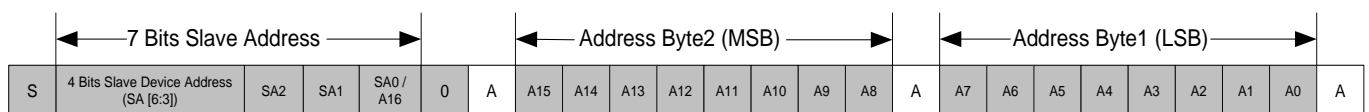
在 I²C nvSRAM 中进行寻址

I²C 主设备控制器以字节格式与 I²C nvSRAM 从设备通行。在传输某个字节时，它始终在第一个时钟周期内传输最高有效位，并在第 8 个时钟周期内发送最低有效位。这种方法适用于所有 I²C 通信，包括指令、地址和数据字节。类似的，当 I²C nvSRAM 在读取操作期间传输数据字节时，它始终先传输最高有效位，最后才传输最低有效位。

图 15 显示的是通过 I²C 总线传输的地址位的示例。

将长度为 7 位的从设备地址表示为 “SA [6:0]”，这样可以同存储器地址位 (表示为 “A [16:0]”) 相区分。在后续内容中，将使用 SA [x] 表示从设备地址位。

图 15. I²C nvSRAM 中的地址位传输



从器件地址

I²C nvSRAM 从设备支持 7 位从设备寻址 SA [6:0]，其中四个最高有效地址位 SA [6:3] 是器件的固定地址位，并且用户不能对其进行修改。通过器件所提供的外部地址引脚（A2、A1、A0），可以配置剩下的三个最低有效地址位 SA [2:0]。I²C nvSRAM 在一个单一的器件内集成了三种不

同的功能，即数据存储器、RTC 功能和其他控制功能。通过修改高 4 位（请查看表 4），I²C nvSRAM 可以分配从设备地址 SA [6:3] 的三个专用 ID，从而可允许 I²C 主设备访问这些功能。

表 4. 从器件地址

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/W	Selects Memory
1	1	0	1	Device select ID		X	R/W	Selects RTC Registers
0	0	1	1	Device select ID		X	R/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

如果总线上的其他任何从设备 ID 与四个从设备的高位 SA [6:3] 相匹配，则用户必须以不同的数值配置从设备地址的低位 SA [2:0]，以便使共用同一个系统总线的所有从器件均有唯一的从设备 ID。

图 16. 从器件地址选择

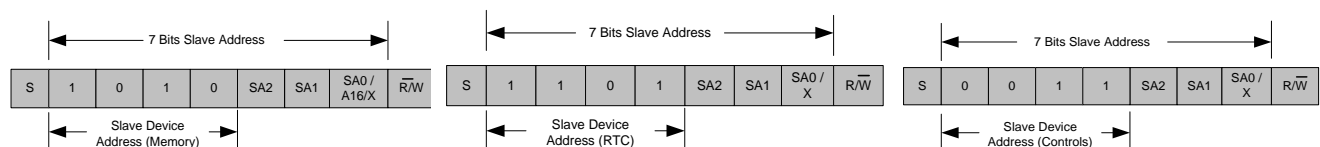


表 5. 对 SRAM 进行读取和写入时，I²C nvSRAM 的寻址情况

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/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/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/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/W	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

注意 2： 最高有效地址字节（MSB）中未使用的位无需关注，并且 nvSRAM 将忽略它们。然而，在固件中，最好将这些未使用的地址位的位置设置为 ‘0’。这样，在移动到更高密度的选项时，可以轻松实现固件升级。

注意 3： 在某些 nvSRAM 器件配置中，由于封装中的引脚资源不足或由于内部使用了 A0 地址位，因此这些器件仅提供了两个地址引脚（A2 和 A1）。大小为 1 Mbit 的 I²C nvSRAM 要求 17 地址位 A [16:0]，用以映射到它的整个存储器位置。因此，在这些器件中，从设备地址空间 SA0 用于发送 A16 地址位，并且只能使用两个位 SA [2:1] 外部配置从设备地址。在容量更小的器件（512 Kbit 或更小）中，由于封装中的引脚不足，所以 A0 不可用。因而，在内部无需关注（‘X’）该位。带无需关注的从设备地址位 A0 的 I²C 从器件将应答由 I²C 主设备发送的两个从设备地址（在 A0=0 和 A0=1 时）。

I²C nvSRAM 访问

通过标准的 I²C 写入和读取协议，可以访问所有的 I²C nvSRAM 功能，包括标准功能（读取和写入存储器）以及特殊功能（NV 操作、器件 ID 和序列号）。图 17 至图 21

显示的是对 I²C nvSRAM 进行的读和写操作的简化流程图。有关每一个 I²C nvSRAM 功能的说明及其实现的详细信息，请参阅器件的数据手册。

图 17. 写入到 I²C nvSRAM 数据存储器的简化流程图

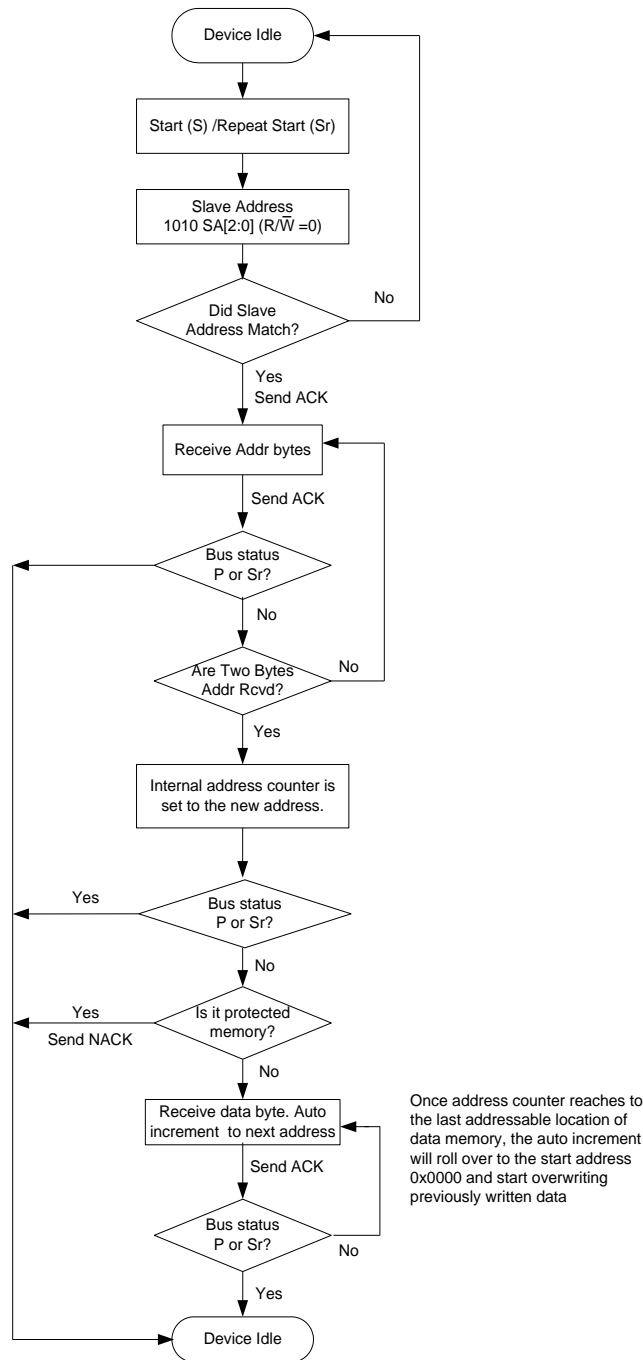


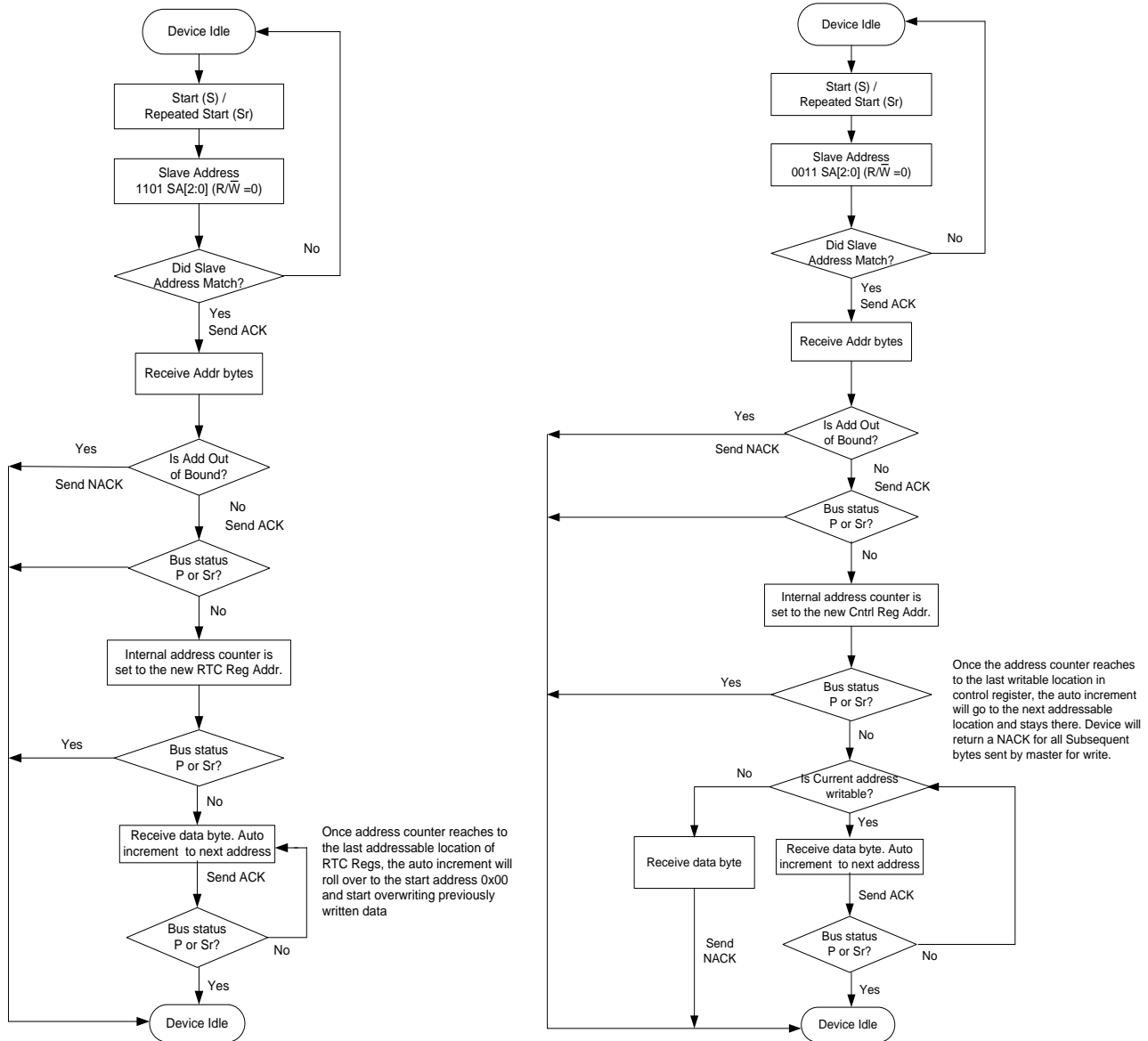
图 18. 写入到 I²C nvSRAM RTC 和控制寄存器的简化流程图


图 19. 读取 I²C nvSRAM 数据存储器的当前地址的简化流程图

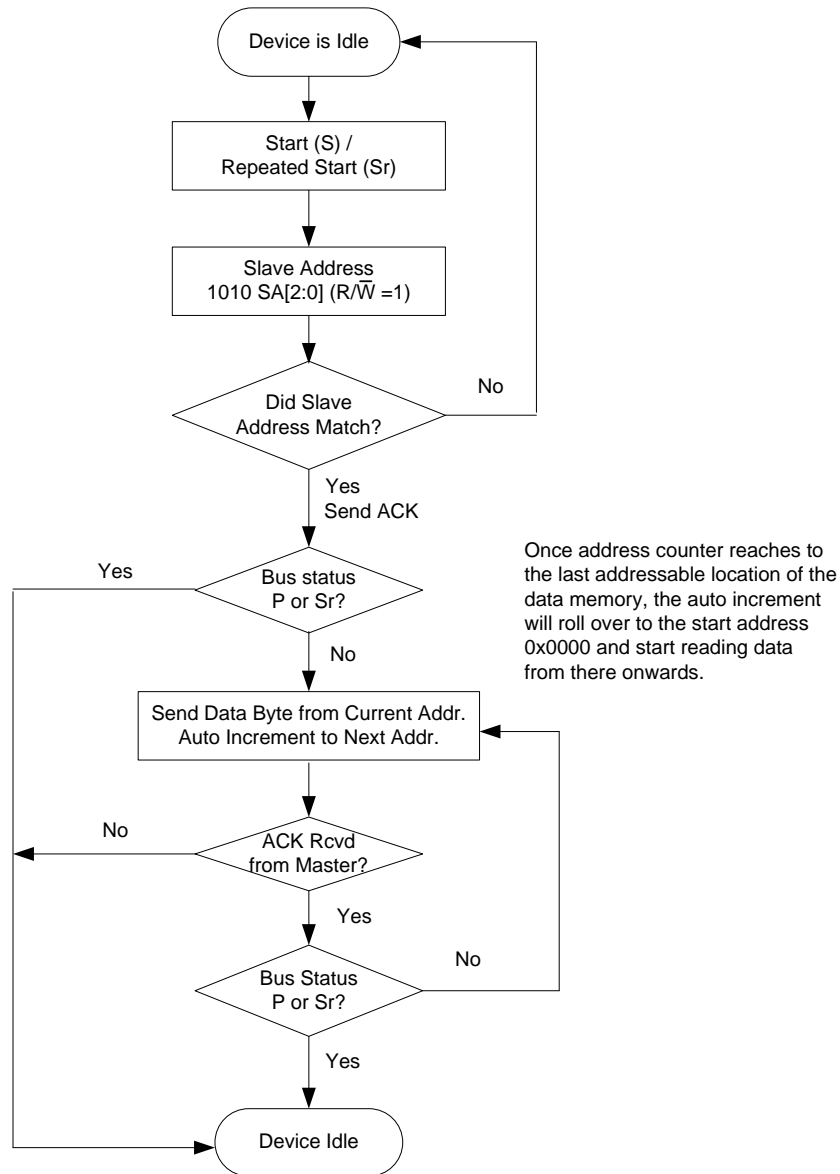


图 20. 读取 I²C nvSRAM RTC 和控制存储器的当前地址的简化流程图

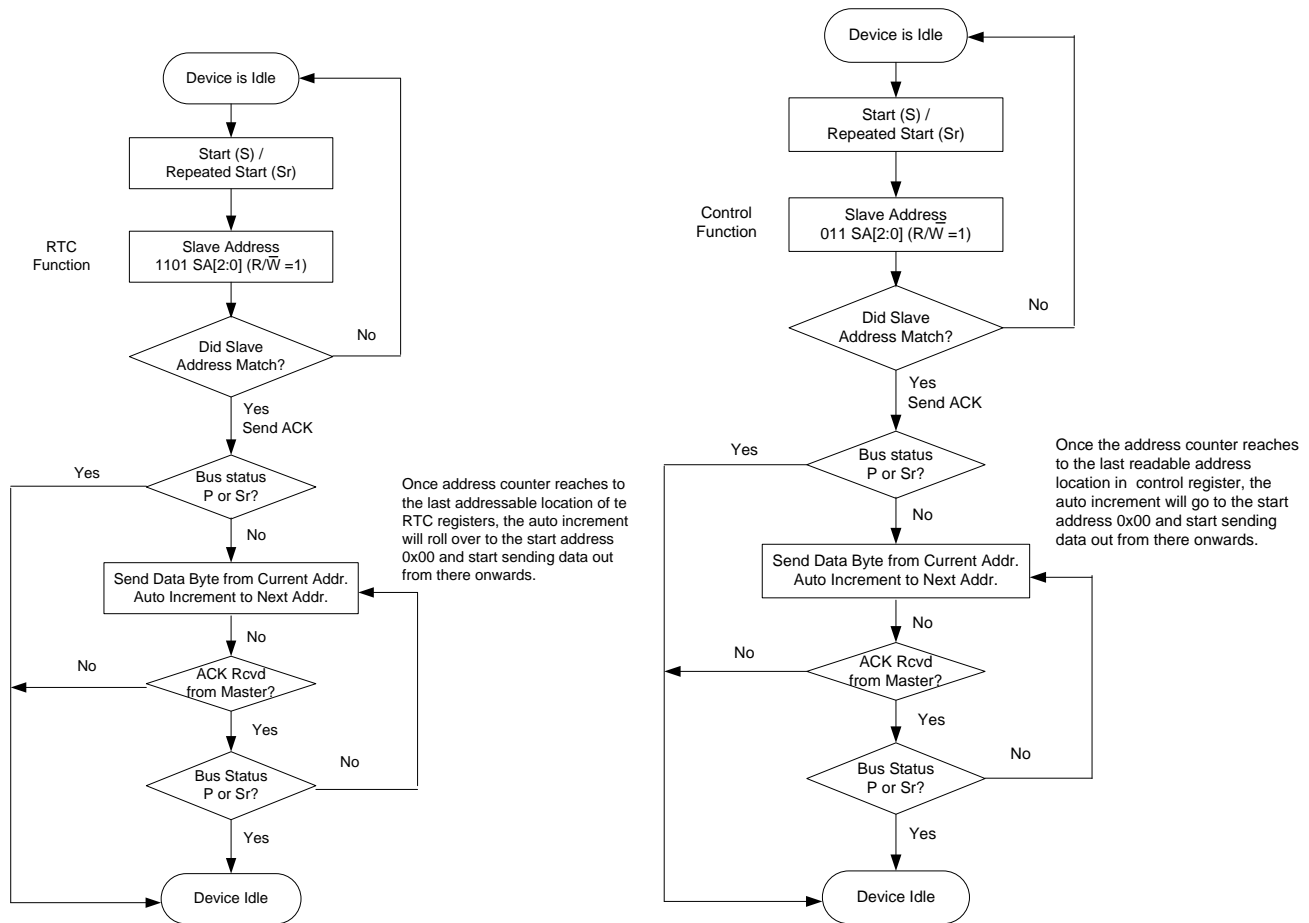
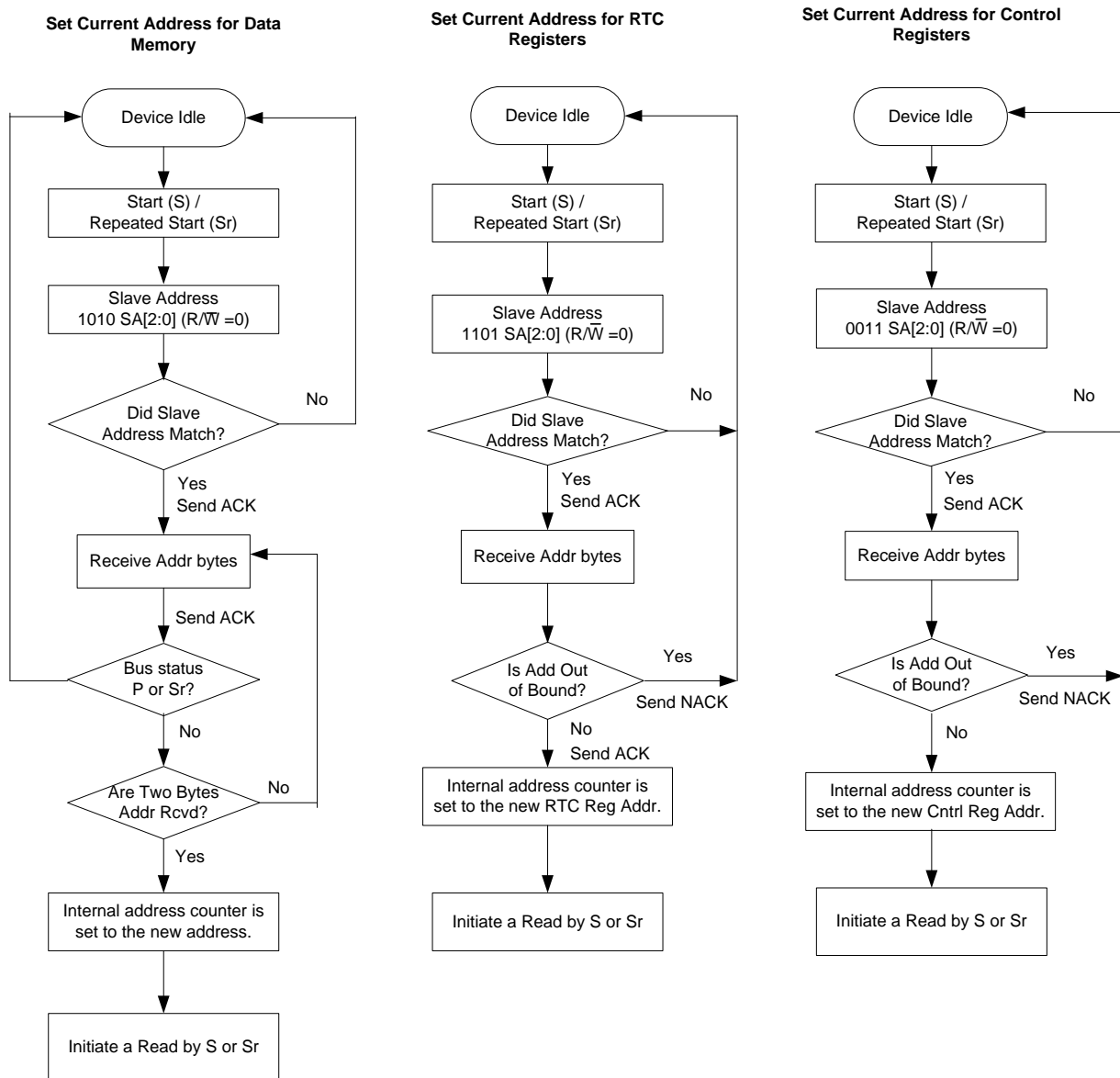


图 19 至图 20 显示的是读取数据存储、RTC 寄存器以及控制寄存器的当前位置的流程图。当前位置是指退出前一个读或写操作时位于地址计数器的地址。如果用户需要读取其他位置，那么必须通过执行一个写周期在地址计数器中设置新地址，如图 21 所示。

图 21. 读取 I²C nvSRAM 数据存储、RTC 和控制寄存器随机地址的简化流程图


总结

与其他所有非易失性 I²C 存储器产品相同，赛普拉斯 I²C nvSRAM 也支持标准的 I²C 访问协议。这样能使 nvSRAM 与所有 I²C 主控制器相兼容，并缩短系统开发的时间。本应用笔记介绍的是如何使用原理图和时序图在应用中配置 I²C nvSRAM。

附录 A（伪代码示例）

I²C 写入

```

/*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 读取

```

/*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
}

```

文档修订记录

文档标题：使用串行 I²C nvSRAM 进行设计—AN74875

文档编号：001-92139

修订版	ECN	原始变更	提交日期	变更说明
**	4345916	LISZ	05/16/2014	本文档版本号为 Rev**，译自英文版 001-74875 Rev*E。

全球销售和设计支持

赛普拉斯公司拥有一个由办事处、解决方案中心、工厂代表和经销商组成的全球性网络。要找到离您最近的办事处，请访问[赛普拉斯所在地](#)。

产品

汽车	cypress.com/go/automotive
时钟与缓冲区	cypress.com/go/clocks
接口	cypress.com/go/interface
照明和电源控制	cypress.com/go/powerpsoc cypress.com/go/plc
存储器	cypress.com/go/memory
光学导航传感器	cypress.com/go/ons
PSoC	cypress.com/go/psoc
触摸感应	cypress.com/go/touch
USB 控制器	cypress.com/go/usb
无线/射频	cypress.com/go/wireless

PSoC®解决方案

psoc.cypress.com/solutions
PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

赛普拉斯开发者社区

[社区](#) | [论坛](#) | [博客](#) | [视频](#) | [培训](#)

技术支持

cypress.com/go/support

此处引用的所有其他商标或注册商标归其各自所有者所有。



赛普拉斯半导体
198 Champion Court
San Jose, CA 95134-1709

电话 : 408-943-2600
传真 : 408-943-4730
网站 : www.cypress.com

©赛普拉斯半导体公司，2011-2014。此处，所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品内嵌的电路外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不根据专利或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯产品不保证能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于合理预计会发生运行异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯将不批准将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

该源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定用途外，未经赛普拉斯的明确书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对该材料提供任何类型的明示或暗示保证，包括（但不限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不另行通知的情况下对此处所述材料进行更改的权利。赛普拉斯不在此处所述之任何产品或电路的应用或使用承担任何责任。对于合理预计可能发生运转异常和故障，并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用受适用的赛普拉斯软件许可协议限制并完全按照此协议使用。