Analog Engineer's Circuit

使用智能 DAC 的高侧电流源 LED 偏置电路

Data Converters

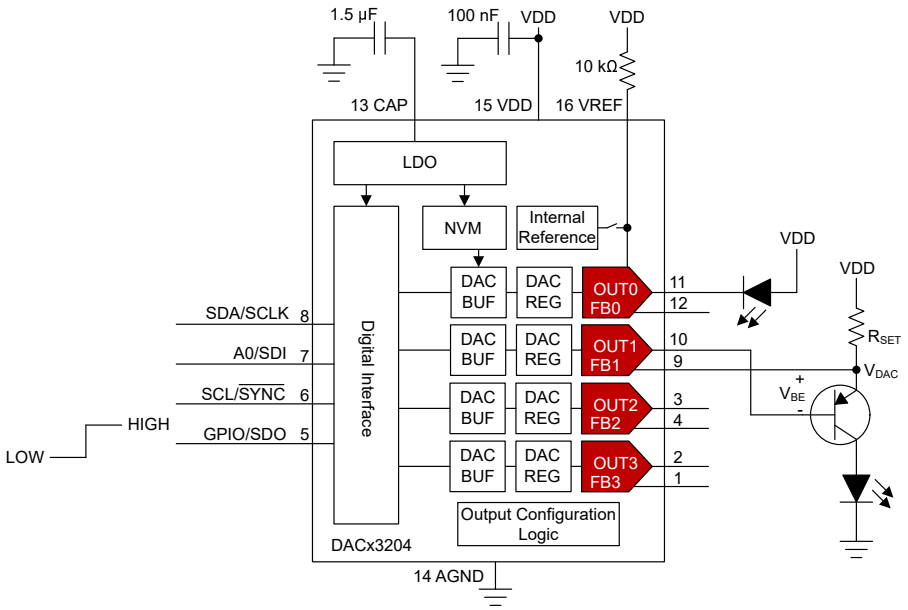
设计目标

主要输入参数	主要输出信号	推荐器件
编程 DAC 代码 0x000 至 0xFFF、GPI 触发器的 SPI 或 I^2C 通信	0A 至 250μA 和 0mA 至 20mA LED 电流	DAC43204（8 位）、 DAC53204（10 位）、 DAC63204（12 位）

目标： *使用智能 DAC 偏置具有高侧电流源的 LED。*

设计说明

本设计采用四通道缓冲电压或电流输出智能 DAC（比如 DAC43204、DAC53204 或 DAC63204 (DACx3204)）来偏置发光二极管 (LED)。在仅需数毫安电流的 LED 偏置应用中，智能 DAC 可配置为电压输出模式，并与双极结型晶体管 (BJT) 连接形成强制感测配置，如电路原理图中 DACx3204 的通道 1 所示。DAC 设置 PNP 型 BJTx（比如 2N2905）的集电极电流，并通过改变基极电压来控制流过 LED 的电流量。LED 连接在 BJT 集电极和接地端之间。DAC 可用于电流输出模式，使用高达 250μA 的电流直接驱动 LED，适合低电流 LED 偏置应用，如电路原理图中 DACx3204 的通道 0 所示。使用 BJT 配置时，DACx3204 的反馈引脚 (V_{FB}) 可补偿基极-发射极电压 (V_{BE}) 压降和 BJT 漂移。DACx3204 具有通用输入输出 (GPIO) 引脚，可在两个电流值之间切换 LED，或开关 LED。可使用智能 DAC 的非易失性存储器 (NVM) 保存所有寄存器设置，这意味着可在无处理器时使用器件，即使在下电上电后也是如此。该电路可用于条形码扫描仪、条形码读取器、点钞机、POS 打印机、光学模块和电器照明等应用。



设计注意事项

1. DACx3204 带自动检测型 I2C、PMBus™ 或 SPI 的 12 位、10 位和 8 位四路电压和电流输出智能 DAC 数据表建议：将 100nF 去耦电容器用于 VDD 引脚，将 1.5μF 或更高的旁路电容器用于 CAP 引脚。CAP 引脚连接到内部低压降稳压器 (LDO)。将这些电容器靠近器件引脚放置。
2. 未使用外部基准时，数据表建议将 VREF 引脚经上拉电阻器连接到 VDD。
3. 此示例电路显示了控制 LED 电流的两种方法。可通过 R_{SET} 电阻器并改变 DACx3204 输出的外部 PNP 型 BJT 的基极电压来设置电流，或者使用 DACx3204 的电流输出模式来设置 LED 电流。
   a. 如需通过外部 BJT 调节 LED 电流，请选择 R_{SET} 电阻器并改变 DAC 输出的基极电压。R_{SET} 的计算公式为：

      $$R_{SET} = \frac{V_{SET}}{I_{LED}}$$

      如果所选 V_{SET} 电压范围为 0V 至 1V，且所需 LED 电流范围为 0mA 到 20mA，则 R_{SET} 的计算公式为：

      $$R_{SET} = \frac{1\,V}{20\,mA)} = 50\,\Omega$$

      10 位 DAC53204 的 DAC 代码的计算公式为：

      $$Code = \frac{V_{DAC}}{V_{REF}} \times 1024$$

      V_{DAC} 的计算公式为：

      $$V_{DAC} = VDD - V_{SET}$$

      如果以 5V VDD 为基准，则高低 DAC 代码分别为：

      $$Code = \frac{5\,V - 0}{5\,V} \times 1024 = 1024\,d$$

      $$Code = \frac{5\,V - 1V}{5\,V} \times 1024 = 819.2\,d$$

      向下舍入后为 1023d 和 819d，得到的高低值分别为 4.995V 和 3.999V。这种配置可补偿温度、集电极电流和 BJT 老化导致的 V_{BE} 压降。与 MOSFET 的典型栅源电压 (V_{GS}) 压降相比，BJT 的 V_{BE} 压降更小。
   b. DAC 可用于电流输出模式，使用高达 250μA 的电流直接驱动 LED。如果所选范围为 ±250μA，则 DAC 代码的计算公式为：

      $$Code = \frac{(I_{DAC} - I_{MIN}) \times 256}{I_{MAX} - I_{MIN}}$$

      高低 DAC53204 代码分别为：

      $$Code = \frac{(0\,\mu A + 250\,\mu A) \times 256}{250\,\mu A + 250\,\mu A} = 128\,d$$

      $$Code = \frac{(-250\,\mu A + 250\,\mu A) \times 256}{250\,\mu A + 250\,\mu A} = 0\,d$$

4. 如果高低 DAC 代码分别存储在 MARGIN-HIGH 和 MARGIN-LOW DAC 寄存器中，则可对这两个值之间的压摆率进行编程。转换时间由 DAC-X-FUNC-CONFIG 寄存器中 SLEW-RATE 和 CODE-STEP 字段的设置值确定。转换时间的计算公式为：

$$SleW\ Time = \frac{(MARGIN\_HIGH\_CODE\ -\ MARGIN\_LOW\_CODE + 1)}{CODE\_STEP} \times SLEW\_RATE$$

如果 CODE-STEP 设置为 1LSB，SLEW-RATE 设置为 4μs/步进，则电压配置的转换时间为：

$$SleW\ Time = \frac{(1023 - 819 + 1)}{1} \times 4\ \mu s = 0.82\ ms$$

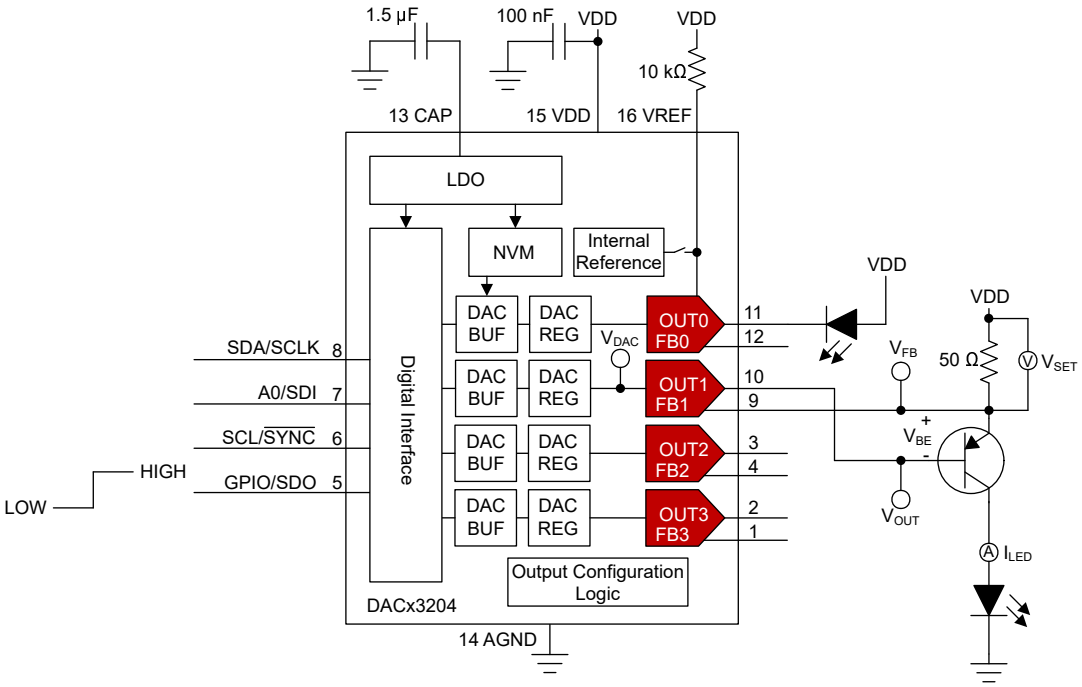
电流输出配置的转换时间为：

$$SleW\ Time = \frac{(128 - 0 + 1)}{1} \times 4\ \mu s = 0.516\ ms$$

5. DACx3204 的 GPIO 引脚可用于根据 DAC-X-FUNC-CONFIG 寄存器中设置的转换时间设置在裕量高和裕量低输出值之间切换。GPI 上的高电平会触发输出转换为高裕量值。GPI 上的低电平会触发输出转换为低裕量值。寄存器设置部分介绍了启用 GPIO 来实现该功能的寄存器设置。
6. 根据寄存器设置部分所述的初始寄存器设置，可使用 I^2C 或 SPI 对 DACx3204 进行编程。可将初始寄存器设置保存至 NVM，方法是将 1 写入 COMMON-TRIGGER 寄存器的 NVM-PROG 字段。对 NVM 进行编程后，器件将在重置或下电上电之后加载具有 NVM 所存储值的所有寄存器。

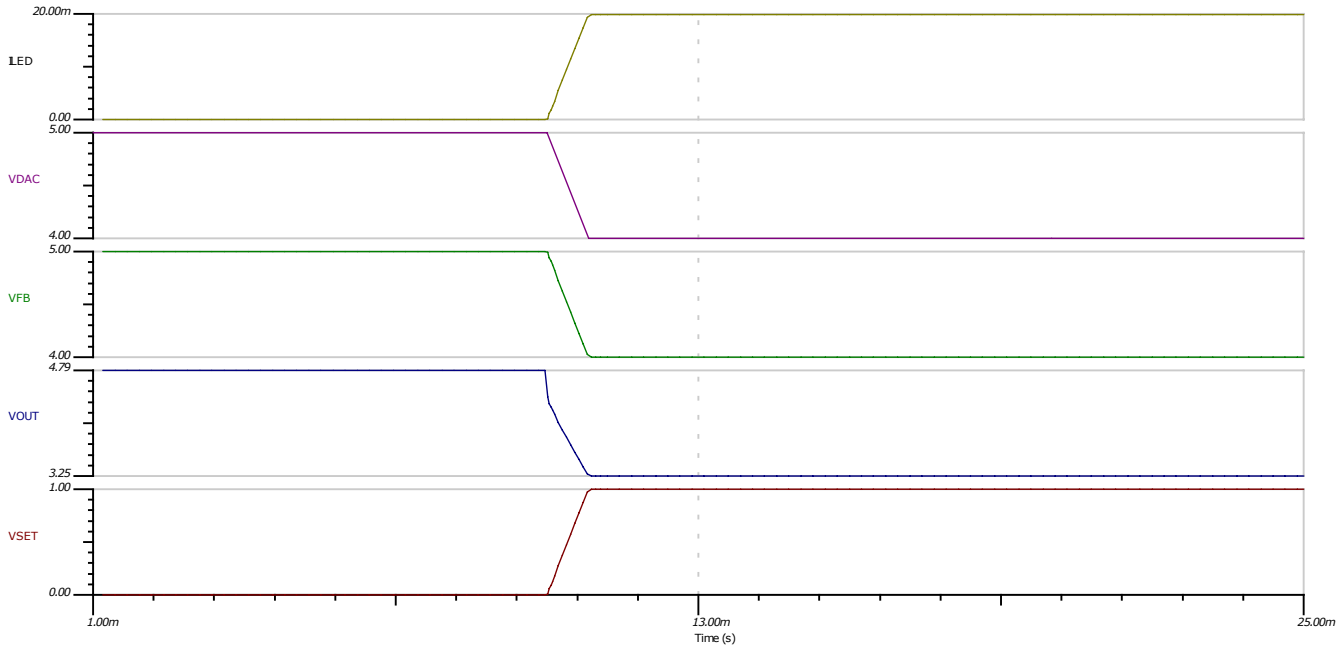
设计仿真

该原理图用于如下 DAC53204 仿真。



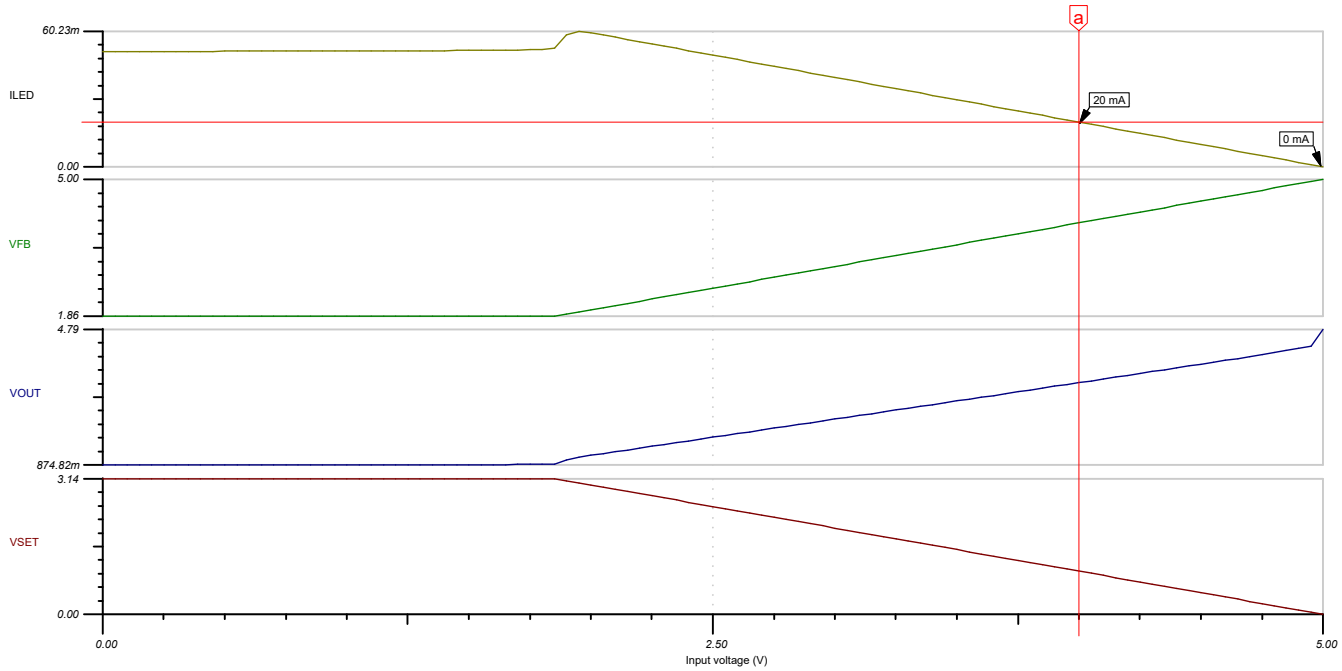
瞬态仿真结果

该仿真展示了 DAC 从裕量高到裕量低代码转换时的 LED 电流。



直流传输仿真结果

该仿真显示了 LED 电流与 DAC 输出电压间的关系。为使典型 PNP 型 BJT 导通，基极电压需比发射极电压低 0.7V。在该电路中，基极电压超过 900mV 时，所选 2N2905 晶体管开始导通。



寄存器设置

电压输出配置的寄存器设置

寄存器地址	寄存器名称	设置	说明
0x01	DAC-0-MARGIN-HIGH	0xFFC0	[15:4] 0xFFC：左对齐 10 位数据更新 MARGIN-HIGH 代码
			[3:0] 0x0：不用考虑
0x02	DAC-0-MARGIN-LOW	0xCCC0	[15:4] 0xCCC：左对齐 10 位数据更新 MARGIN-LOW 代码
			[3:0] 0x0：不用考虑
0x06	DAC-0-FUNC-CONFIG	0x0001	[15] 0b0：写入 0b1，将 DAC-0 清除设置设为中标度
			[14] 0b0：写入 0b1，通过 LDAC 触发器更新 DAC-0
			[13] 0b0：写入 0b1，使 DAC-0 通过广播命令更新
			[12:11] 0b00：为函数发生器选择相位
			[10:8] 0b000：选择函数发生器生成的波形
			[7] 0b0：写入 0b1，启用对数转换
			[6:4] 0b000：选择 1 LSB 的 代码步进
			[3:0] 0b001：选择 4μs/步进的压摆率
0x1F	COMMON-CONFIG	0x0FF9	[15] 0b0：写入 0b1，将窗口比较器输出设置为锁存输出
			[14] 0b0：写入 0b1，锁定器件。将 0b0101 写入 COMMON-TRIGGER 寄存器的 DEV-UNLOCK 字段，以进行解锁
			[13] 0b0：写入 0b1，在地址 0x01 处设置故障转储读取使能
			[12] 0b0：写入 0b1，启用内部基准
			[11:10] 0b11：将 VOUT3 断电
			[9] 0b1：将 IOUT3 断电
			[8:7] 0b11：将 VOUT2 断电
			[6] 0b1：将 IOUT2 断电
			[5:4] 0b11：将 VOUT1 断电
			[3] 0b1：将 IOUT1 断电
			[2:1] 0b00：将 VOUT0 上电
			[0] 0b1：将 IOUT0 断电
0x20	COMMON-TRIGGER	0x0002	[15:12] 0b0000：写入 0b0101，解锁器件
			[11:8] 0b0000：写入 0b1010，触发 POR 复位
			[7] 0b0：如 DAC-X-FUNC-CONFIG 寄存器中相应 SYNC-CONFIG-X 位为 1，则写入 0b1，触发 LDAC 运行。
			[6] 0b0：写入 0b1，基于 DAC-X-FUNC-CONFIG 寄存器中相应 CLR-SEL-X 位，将 DAC 寄存器和输出设置为零代码或中间代码
			[5] 0b0：不用考虑
			[4] 0b0：写入 0b1，触发故障转储序列
			[3] 0b0：写入 0b1，触发 PROTECT 功能
			[2] 0b0：写入 0b1，读取 NVM 的一行进行故障转储
			[1] 0b1：写入 0b1，将适用寄存器设置存储到 NVM
			[0] 0b0：写入 0b1，使用现有 NVM 设置重新加载适用寄存器

电压输出配置的寄存器设置 (continued)

寄存器地址	寄存器名称	设置	说明
0x24	GPIO-CONFIG	0x0035	[15] 0b0：写入 0b1，在 GPI 上启用干扰滤波器
			[14] 0b0：不用考虑
			[13] 0b0：写入 0b1，在 GPIO 引脚上启用输出模式
			[12:9] 0b0000：映射到 GPIO 的 STATUS 功能设置作为输出
			[8:5] 0b0001：确定受特定于通道的 GPI 功能影响的通道
			[4:1] 0b1010：选择 GPI，触发裕量高/低
			[0] 0b1：启用 GPIO 引脚的输入模式

电流输出配置的寄存器设置

寄存器地址	寄存器名称	设置	说明
0x01	DAC-0-MARGIN-HIGH	0x8000	[15:4] 0x800：左对齐 8 位数据更新 MARGIN-HIGH 代码
			[3:0] 0x0：不用考虑
0x02	DAC-0-MARGIN-LOW	0x0000	[15:4] 0x000：左对齐 8 位数据更新 MARGIN-LOW 代码
			[3:0] 0x0：不用考虑
0x06	DAC-0-FUNC-CONFIG	0x0001	[15] 0b0：写入 0b1，将 DAC-0 清除设置设为中标度
			[14] 0b0：写入 0b1，通过 LDAC 触发器更新 DAC-0
			[13] 0b0：写入 0b1，使 DAC-0 通过广播命令更新
			[12:11] 0b00：为函数发生器选择相位
			[10:8] 0b000：选择函数发生器生成的波形
			[7] 0b0：写入 0b1，启用对数转换
			[6:4] 0b000：选择 1 LSB 的 代码步进
			[3:0] 0b001：选择 4µs/步进的压摆率
0x1F	COMMON-CONFIG	0x0FFE	[15] 0b0：写入 0b1，将窗口比较器输出设置为锁存输出
			[14] 0b0：写入 0b1，锁定器件。将 0b0101 写入 COMMON-TRIGGER 寄存器的 DEV-UNLOCK 字段，以进行解锁
			[13] 0b0：写入 0b1，在地址 0x01 处设置故障转储读取使能
			[12] 0b0：写入 0b1，启用内部基准
			[11:10] 0b11：将 VOUT3 断电
			[9] 0b1：将 IOUT3 断电
			[8:7] 0b11：将 VOUT2 断电
			[6] 0b1：将 IOUT2 断电
			[5:4] 0b11：将 VOUT1 断电
			[3] 0b1：将 IOUT1 断电
			[2:1] 0b11：将 VOUT0 断电
			[0] 0b0：将 IOUT0 上电

电流输出配置的寄存器设置 (continued)

寄存器地址	寄存器名称	设置	说明
0x20	COMMON-TRIGGER	0x0002	[15:12] 0b0000：写入 0b0101，解锁器件
			[11:8] 0b0000：写入 0b1010，触发 POR 复位
			[7] 0b0：如 DAC-X-FUNC-CONFIG 寄存器中相应 SYNC-CONFIG-X 位为 1，则写入 0b1，触发 LDAC 运行。
			[6] 0b0：写入 0b1，基于 DAC-X-FUNC-CONFIG 寄存器中相应 CLR-SEL-X 位，将 DAC 寄存器和输出设置为零代码或中间代码
			[5] 0b0：不用考虑
			[4] 0b0：写入 0b1，触发故障转储序列
			[3] 0b0：写入 0b1，触发 PROTECT 功能
			[2] 0b0：写入 0b1，读取 NVM 的一行进行故障转储
			[1] 0b1：写入 0b1，将适用寄存器设置存储到 NVM
			[0] 0b0：写入 0b1，使用现有 NVM 设置重新加载适用寄存器
0x24	GPIO-CONFIG	0x0035	[15] 0b0：写入 0b1，在 GPI 上启用干扰滤波器
			[14] 0b0：不用考虑
			[13] 0b0：写入 0b1，在 GPIO 引脚上启用输出模式
			[12:9] 0b0000：映射到 GPIO 的 STATUS 功能设置作为输出
			[8:5] 0b0001：确定受特定于通道的 GPI 功能影响的通道
			[4:1] 0b1010：选择 GPI，触发裕量高/低
			[0] 0b1：启用 GPIO 引脚的输入模式

伪代码示例

下面所示为将初始寄存器值编程到 DAC53204 的 NVM 的伪代码序列。此处给出的值基于在设计注意事项中所做的设计选择。

电压输出配置的伪代码示例

```
1:   //SYNTAX: WRITE <REGISTER NAME (Hex code)>, <MSB DATA>, <LSB DATA>
2:  //Configure GPI for margin high/low trigger
3:  WRITE GPIO-CONFIG(0x24), 0x00, 0x35
4:  //With 16-bit left alignment 0x3FF becomes 0xFFC0
5:  WRITE DAC-0-MARGIN-HIGH(0x01), 0xFF, 0xC0
6:  //With 16-bit left alignment 0x333 becomes 0xCCC0
7:  WRITE DAC-0-MARGIN-LOW(0x02), 0xCC, 0xC0
8:  //Set the CODE-SETP to 1 LSB and SLEW-RATE to 4 µs/step
9:  WRITE DAC-0-FUNC-CONFIG(0x06), 0x00, 0x01
10: //Power-up voltage output on channel 0, internal reference disabled
11: WRITE GENERAL_CONFIG(0x1F), 0x0F, 0xF9
12: //Save settings to NVM
13: WRITE COMMON-TRIGGER(0x20), 0x00, 0x02
```

电流输出配置的伪代码示例

```
1:   //SYNTAX: WRITE <REGISTER NAME (Hex code)>, <MSB DATA>, <LSB DATA>
2:  //Configure GPI for deep-sleep trigger and enable deep-sleep function
3:  WRITE GPIO-CONFIG(0x24), 0x00, 0x35
4:  //With 16-bit left alignment 0x80 becomes 0x8000
5:  WRITE DAC-0-MARGIN-HIGH(0x01), 0x80, 0x00
6:  //Write DAC0 margin low code
7:  WRITE DAC-0-MARGIN-LOW(0x02), 0x00, 0x00
8:  //Set the CODE-SETP to 1 LSB and SLEW-RATE to 4 µs/step
9:  WRITE DAC-0-FUNC-CONFIG(0x06), 0x00, 0x01
10: //Power-up current output on channel 0, internal reference disabled
11: WRITE GENERAL_CONFIG(0x1F), 0x0F, 0xFE
12: //Save settings to NVM
13: WRITE COMMON-TRIGGER(0x20), 0x00, 0x02
```

设计中采用的器件

器件	关键特性	链接
DAC43204	具有 I2C、SPI 和 PWM 的 4 通道 8 位 VOUT 和 IOUT 智能 DAC	DAC43204
DAC53204	具有 I2C、SPI 和 PWM 的 4 通道 10 位 VOUT 和 IOUT 智能 DAC	DAC53204
DAC63204	具有 I2C、SPI 和 PWM 的 4 通道 12 位 VOUT 和 IOUT 智能 DAC	DAC63204

使用参数搜索工具查找其他可能的器件。

设计参考资料

有关 TI 综合电路库的信息，请参阅*模拟工程师电路手册*。

附加资源

- 德州仪器 (TI)，DAC63204 评估模块
- 德州仪器 (TI)，DAC63204 EVM 用户指南
- 德州仪器 (TI)，高精度实验室 - DAC

如需 TI 工程师的直接支持，请登陆 E2E 社区：

e2e.ti.com

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265