



Nima Eskandari and Veena Kamath

摘要

链接器命令文件在嵌入式程序中起着重要作用，因为它们指定了将代码段和数据段分配到目标存储器中的位置。没有此文件，链接器不知道目标存储器配置以及如何正确分配这些段。对于 C2000™ 实时控制器，您必须通过查看器件特定数据表和技术参考手册来了解器件存储器。C2000Ware SDK 中提供示例 C2000 链接器命令文件，但对于任何给定应用，您可能必须修改 C2000Ware SDK 中可用的模板链接器命令文件才能满足您的应用需求。这要求您了解编写链接器命令文件时可用的语法和选项。C2000 链接器 CMD 工具通过提供直观的 GUI 和自动代码生成，简化了创建应用特定链接器命令文件的任务。

内容

1 引言	2
2 C2000 链接器命令工具 - GUI 配置	2
2.1 存储器组合.....	5
2.2 存储器段.....	6
2.3 CLA 段.....	7
3 C2000 链接器命令工具 - 代码生成	8
3.1 device_cmd.cmd 文件.....	8
3.2 支持文件.....	9
4 跨器件系列迁移	12
5 总结	14
6 参考文献	15

插图清单

图 2-1. 链接器 CMD 工具 - GUI 概述.....	3
图 2-2. 链接器 CMD 工具 - 生成的文件.....	3
图 2-3. 链接器 CMD 工具 - 全局参数.....	4
图 2-4. 链接器 CMD 工具 - CMD 实例配置.....	4
图 2-5. 链接器 CMD 工具 - 存储器组合.....	5
图 2-6. 链接器 CMD 工具 - 存储器段.....	6
图 2-7. 链接器 CMD 工具 - 加载存储器.....	6
图 2-8. 链接器 CMD 工具 - 用户定义的段.....	7
图 2-9. 链接器 CMD 工具 - CLA 段.....	7
图 3-1. 生成的文件 - CMD 文件.....	8
图 3-2. 生成的文件 - CMD 文件段.....	9
图 3-3. 生成的文件 - CMD File Diff.....	9
图 3-4. 生成的文件 - 复制表.....	10
图 3-5. 生成的文件 - C 文件.....	10
图 3-6. 生成的文件 - OPT 文件.....	11
图 3-7. 生成的文件 - Genlibs 文件.....	11
图 4-1. 器件迁移 - SWITCH.....	12
图 4-2. 器件迁移 - 文件已更改.....	13
图 4-3. 器件迁移 - 文件更改.....	14

商标

C2000™ and Code Composer Studio™ are trademarks of Texas Instruments.

所有商标均为其各自所有者的财产。

1 引言

链接器命令文件用于应用程序编译过程的链接阶段，在此阶段，链接器会组合目标文件并将片段分配到目标系统的已配置内存中。链接器命令文件是一个 ASCII 文件，它使用两个链接器指令 **MEMORY** 和 **SECTIONS** 将片段分配到内存的特定区域中。**MEMORY** 指令定义目标内存配置。**SECTIONS** 指令控制如何构建和分配片段。此外，链接器命令文件还可以包含输入文件名和链接器选项。

[链接器命令文件入门](#)页面介绍了链接器命令文件的基本信息，重点阐述了 **MEMORY** 和 **SECTIONS** 指令。

对于新用户而言，从头开始创建新的链接器命令文件会很困难，甚至编辑现有的链接器命令文件模板都不容易。用户必须了解链接器命令文件的结构及其特定器件的存储器结构。

C2000 链接器 **CMD** 工具通过提供以下特性，显著简化了新建链接器命令文件或编辑现有链接器命令文件的任务：

- 基于 **GUI** 的直观界面，展示了所有可用的自定义选项
- 执行错误检查以帮助避免犯错
- 自动生成 **CMD** 文件
- 自动修改 **Code Composer Studio™** 工程属性
- 自动生成用于初始化存储器段的附加 **C** 源文件和头文件
- 展示所选器件系列的可用存储器

使用 **C2000** 链接器 **CMD** 工具可以加快新用户和高级用户的软件开发速度。

2 C2000 链接器命令工具 - GUI 配置

C2000 链接器命令工具是一款基于 **SysConfig** 的产品，可无缝集成到 **C2000** 系统配置工具中。

更多有关 **C2000** 系统配置工具的信息，请访问：

视频系列：

- [7.1 C2000™ SysConfig：概述](#)
- [7.2 C2000™ SysConfig：入门](#)
- [7.3 C2000™ SysConfig：PinMux](#)
- [7.4 C2000™ SysConfig：板级支持](#)
- [7.5 C2000™ SysConfig：示例演练](#)
- [7.6 C2000™ SysConfig：在 10 分钟内迁移 C2000 器件](#)

C2000 SysConfig 的优势：

- [利用 SysConfig 并借助 C2000™ 实时 MCU 加速开发](#)

应用报告 - **C2000 SysConfig** 分步使用指南：

- [C2000 SysConfig](#)

软件入门指南：

- https://software-dl.ti.com/C2000/docs/software_guide/c2000_sysconfig.html

开发人员必须针对他们的给定器件和封装启动 **C2000 SysConfig** 工具，才能使用 **C2000** 链接器 **CMD** 工具。[C2000 SysConfig](#) 将引导您完成在 **Code Composer Studio** 工程上下文以及 **SysConfig** 独立工具中启动 **C2000 SysConfig** 工具所需的步骤。

C2000 链接器命令工具如图 2-1 所示。

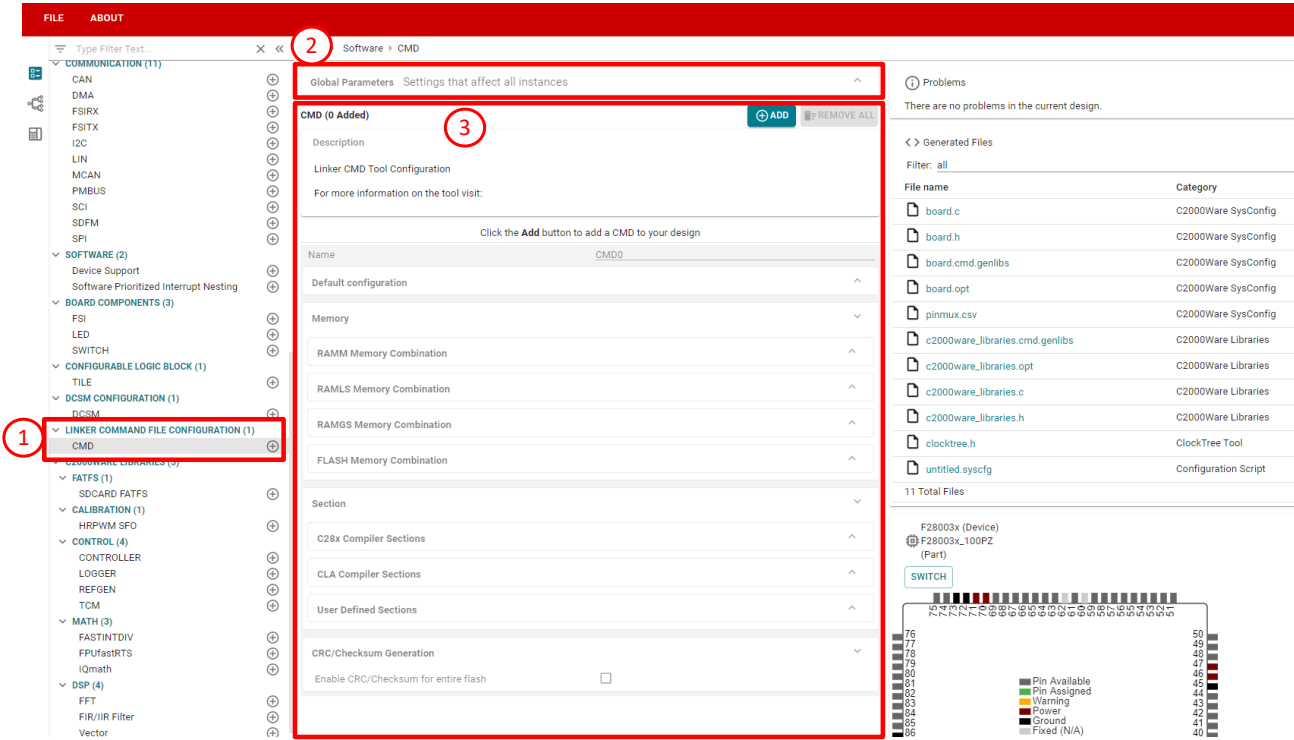


图 2-1. 链接器 CMD 工具 - GUI 概述

1. 链接器 CMD 模块
2. 影响添加到设计中的所有 CMD 模块实例的全局设置
3. 设计中 CMD 模块的实例

添加 CMD 模块后，该工具会生成其他文件。

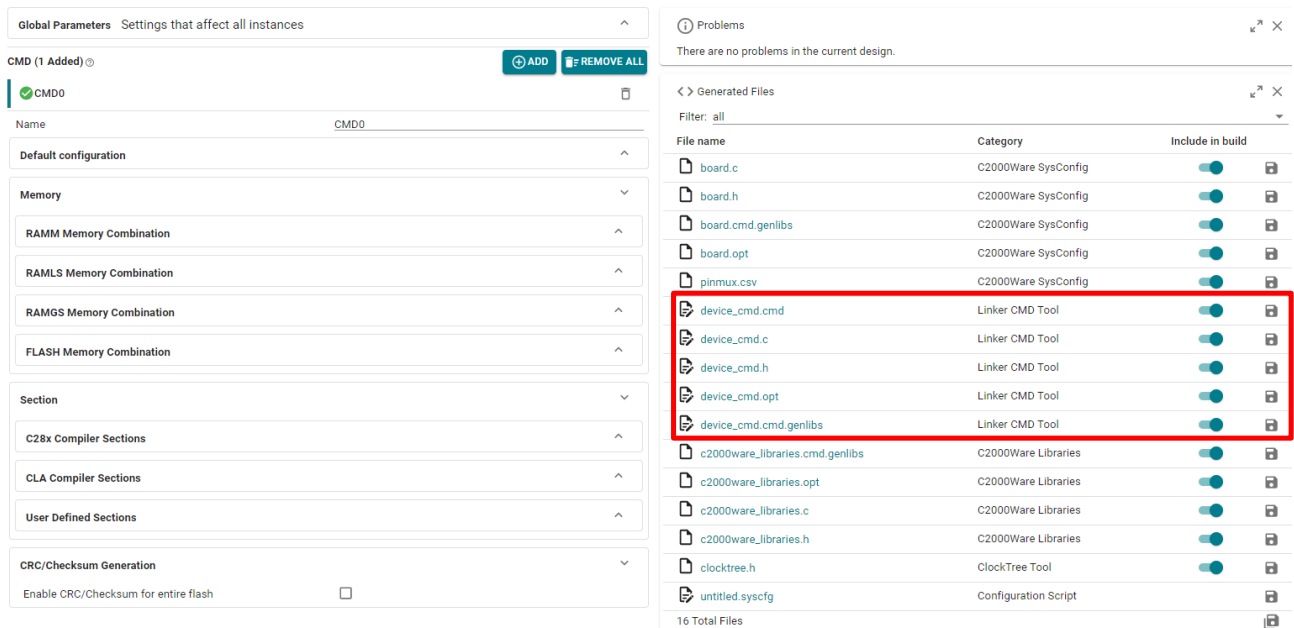


图 2-2. 链接器 CMD 工具 - 生成的文件

请注意，您可以添加多个 **CMD** 模块。CMD 模块的所有不同实例均可保存在 **syscfg** 文件中。您可以决定哪个 **CMD** 模块处于活动状态，方法是在**全局参数**中选择该模块。

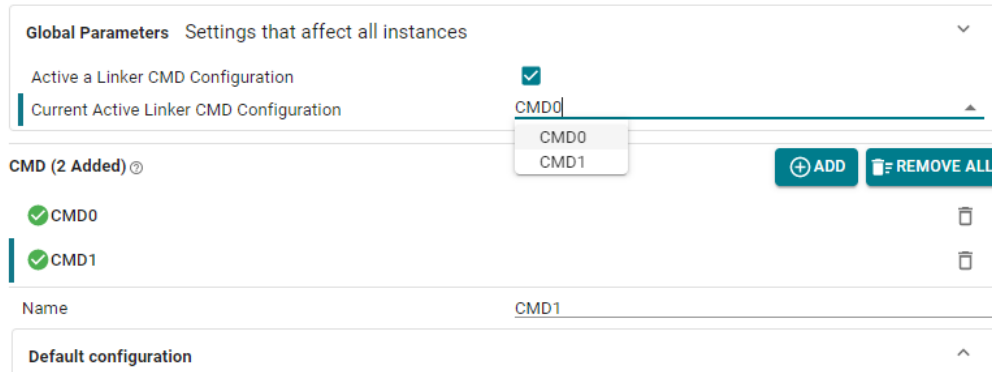


图 2-3. 链接器 **CMD** 工具 - 全局参数

CMD 模块的每个实例都包含以下条目：

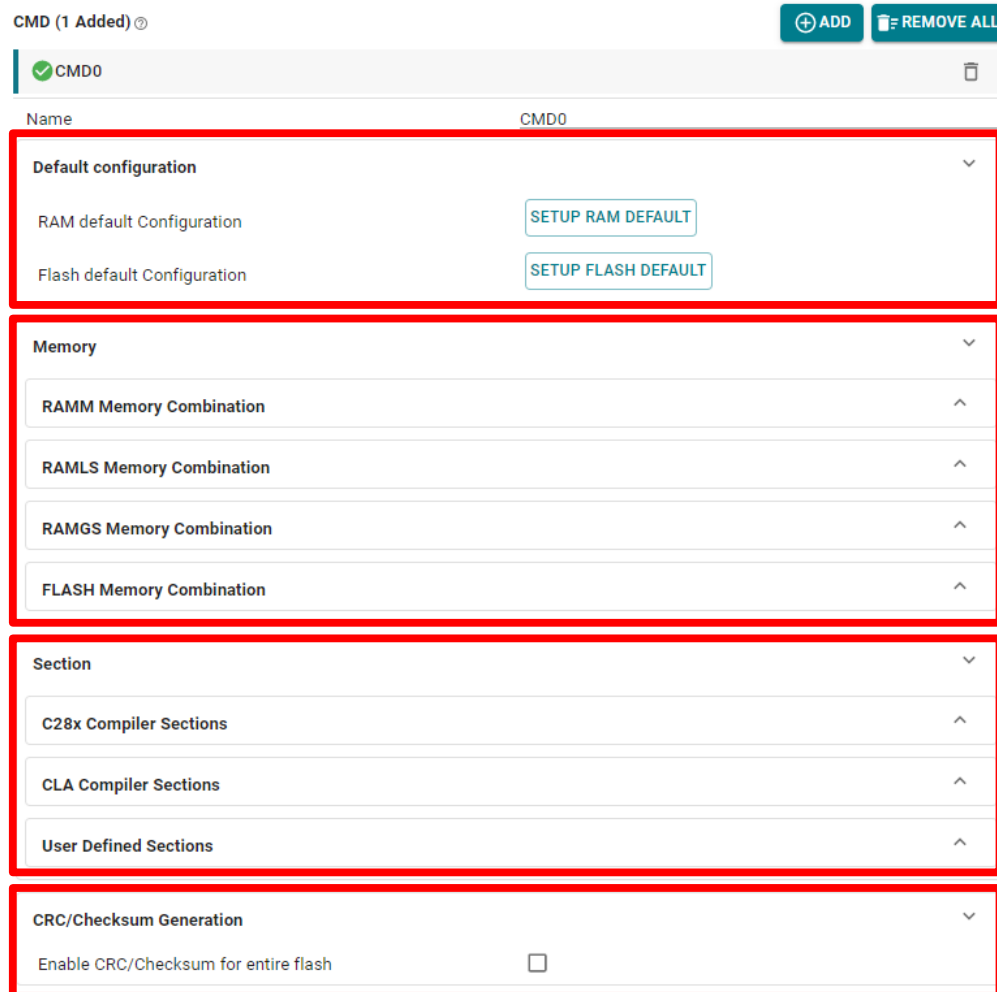


图 2-4. 链接器 **CMD** 工具 - **CMD** 实例配置

- 默认配置：将实例配置为针对该器件提供的默认设置
- 存储器：按存储器类型分组，组合存储块以创建更大的存储器组
- 章节：根据用途分配器件存储器
- CRC/校验和生成：为整个闪存生成 CRC/校验和

2.1 存储器组合

您可以将同一类型的不同存储块组合起来并为新存储块命名，如图 2-5 所示。

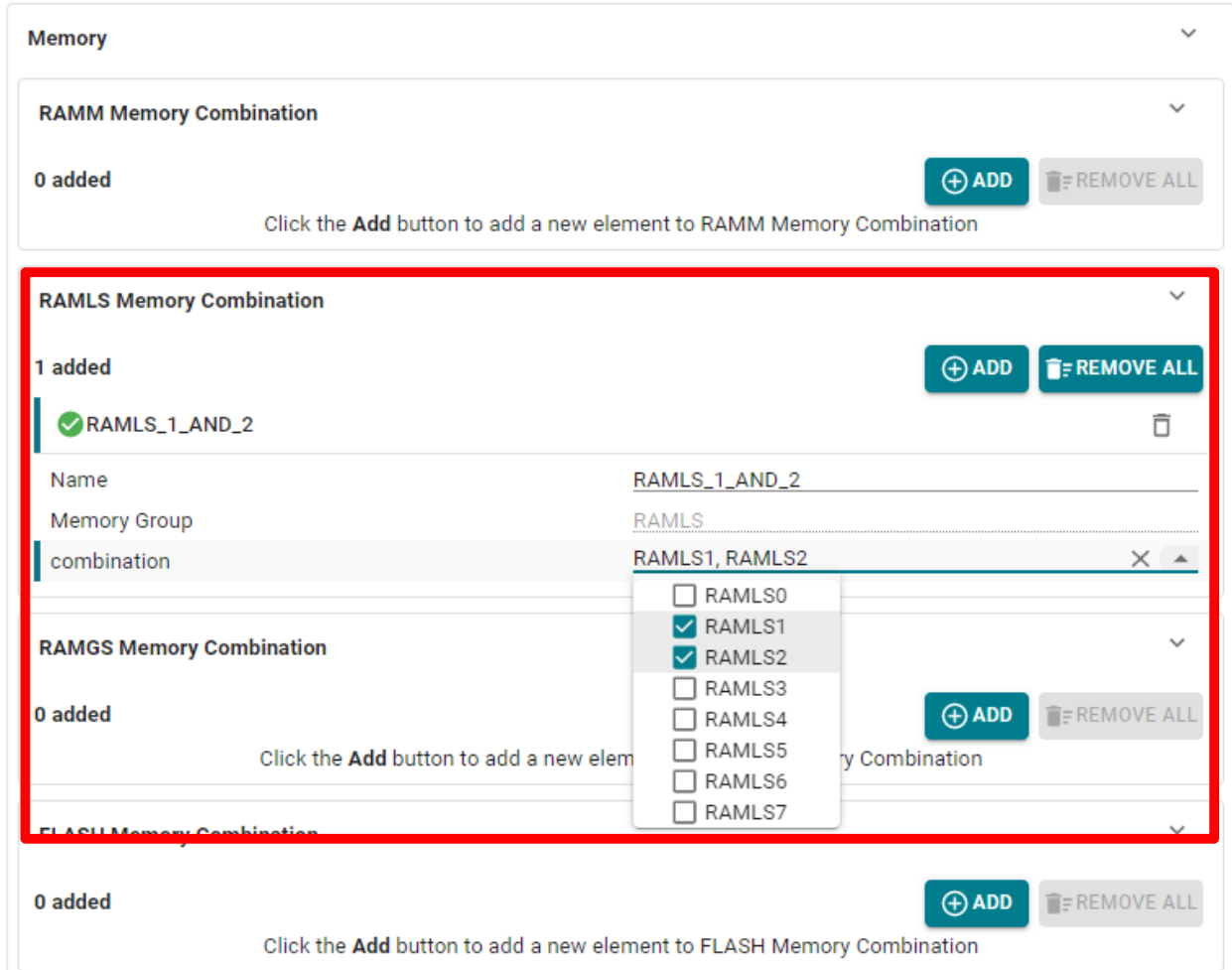


图 2-5. 链接器 CMD 工具 - 存储器组合

该工具不允许组合不连续的器件存储器。

2.2 存储器段

这些段分成几组，具体取决于它们是属于 C28x、CLA 还是自定义的“用户定义”段。



图 2-6. 链接器 CMD 工具 - 存储器段

每个段都分配了一个存储块。可用的选项包括器件存储块和组合存储块。

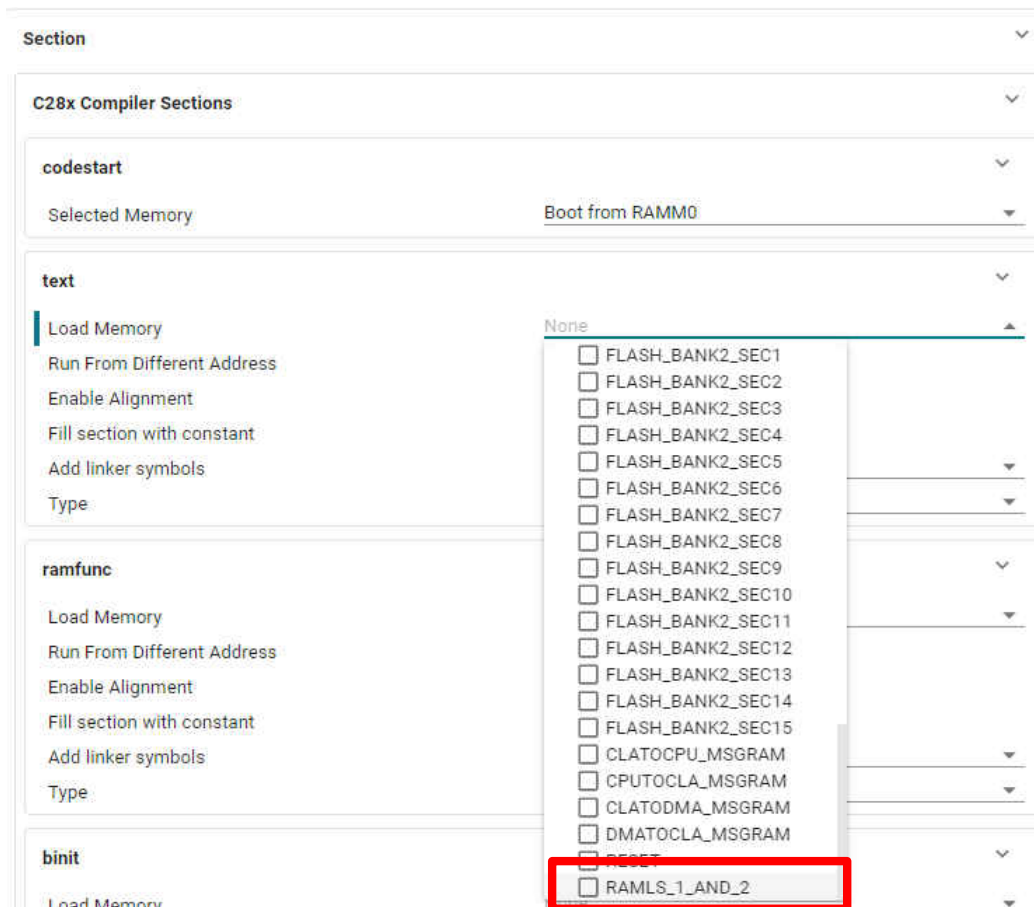


图 2-7. 链接器 CMD 工具 - 加载存储器

如果器件有一个额外的 CLA 内核，则会显示 CLA 段供您配置。

可以根据需要添加和命名用户定义的段，以满足应用需求。

User Defined Sections ▼

1 added
+ ADD
 - REMOVE ALL

✔ userSection0 🗑️

Name	userSection0
Section Name	customUserSection0
Library Name	
Obj Name	
Load Memory	RAMLS0 ▼
Run From Different Address	<input type="checkbox"/>
Enable Alignment	<input type="checkbox"/>
Fill section with constant	<input type="checkbox"/>
Add linker symbols	None ▼
Type	None ▼

图 2-8. 链接器 CMD 工具 - 用户定义的段

2.3 CLA 段

链接器 CMD 工具还支持具有 CLA 支持的器件上的 CLA 段。

CLA Compiler Sections ▼

cla1Prog ▼

Load Memory	None ▼
Run From Different Address	<input type="checkbox"/>
Enable Alignment	<input type="checkbox"/>
Fill section with constant	<input type="checkbox"/>
Add linker symbols	None ▼
Type	None ▼

claConst ▼

Load Memory	None ▼
Run From Different Address	<input type="checkbox"/>
Enable Alignment	<input type="checkbox"/>
Fill section with constant	<input type="checkbox"/>
Add linker symbols	None ▼
Type	None ▼

claScratchpad ▼

Load Memory	None ▼
Enable Alignment	<input type="checkbox"/>
Fill section with constant	<input type="checkbox"/>
Add linker symbols	None ▼
Type	None ▼

bssCla ▼

Load Memory	None ▼
Enable Alignment	<input type="checkbox"/>
Fill section with constant	<input type="checkbox"/>
Add linker symbols	None ▼
Type	None ▼

图 2-9. 链接器 CMD 工具 - CLA 段

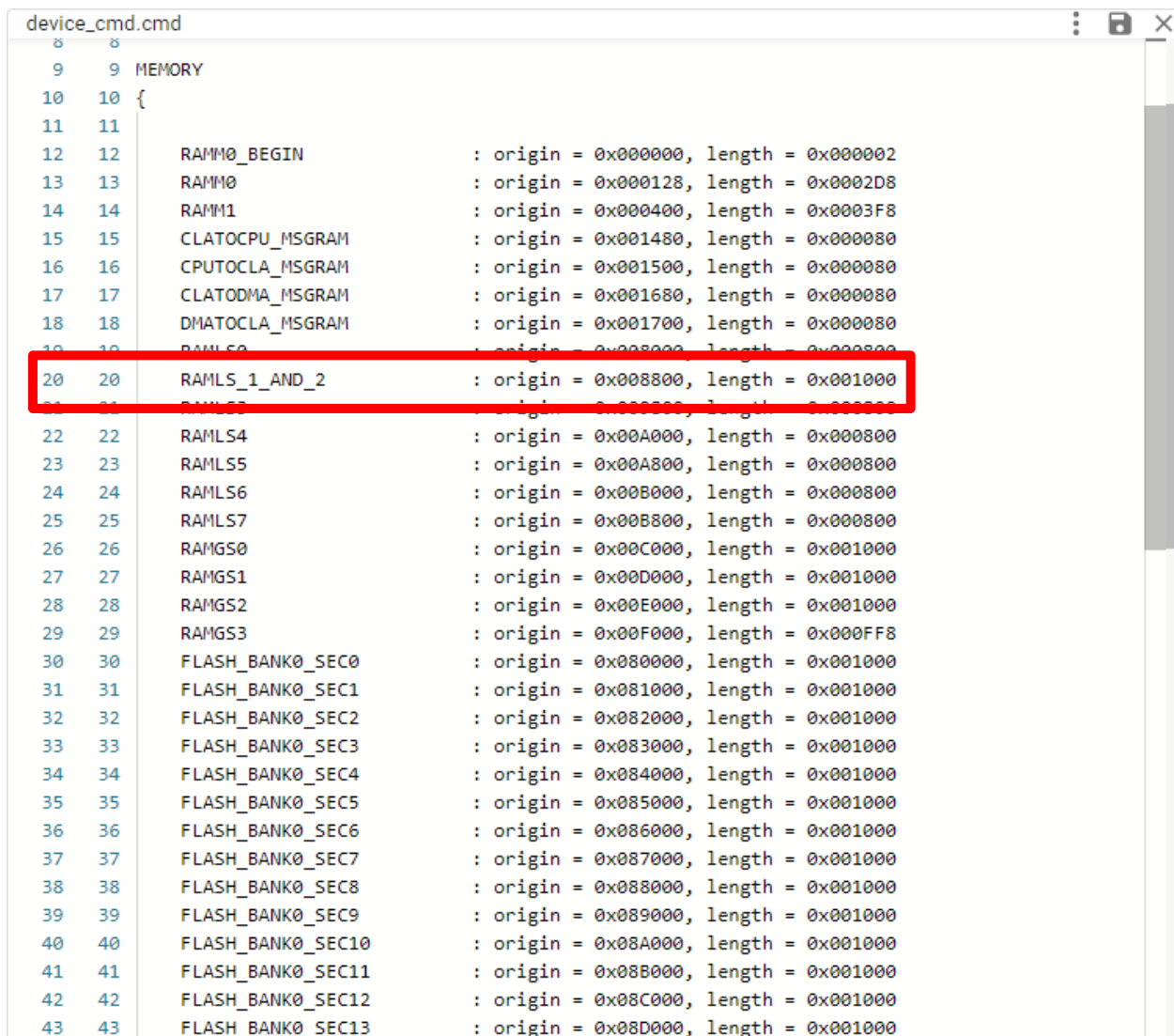
3 C2000 链接器命令工具 - 代码生成

链接器 CMD 工具生成一系列文件：

- **device_cmd.cmd** : 这是主文件，其中包含基于 GUI 中所选选项的链接器命令文件条目。
- **device_cmd.c** : 该文件包含特定存储器段的初始化代码，这些段需要从与其加载地址不同的地址运行。
- **device_cmd.h** : 与 device_cmd.c 文件密切相关的头源文件。
- **device_cmd.opt** : 在 Code Composer Studio 工程的上下文中，OPT 文件中的条目是 CMD 工具所需的编译器选项。这些选项会自动附加到工程中。
- **device_cmd.cmd.genlibs** : 在 Code Composer Studio 工程的上下文中，GENLIBS 文件中的条目是 CMD 工具所需的链接器选项。如果链接器在工程属性中引用该文件，这些选项会自动附加到工程中。

3.1 device_cmd.cmd 文件

device_cmd.cmd 文件包含链接器 cmd 条目。



```

device_cmd.cmd
9 9 MEMORY
10 10 {
11 11
12 12     RAMM0_BEGIN           : origin = 0x000000, length = 0x000002
13 13     RAMM0                 : origin = 0x000128, length = 0x0002D8
14 14     RAMM1                 : origin = 0x000400, length = 0x0003F8
15 15     CLATOCPU_MSGRAM      : origin = 0x001480, length = 0x000080
16 16     CPUTOCLA_MSGRAM      : origin = 0x001500, length = 0x000080
17 17     CLATODMA_MSGRAM     : origin = 0x001680, length = 0x000080
18 18     DMATODCLA_MSGRAM    : origin = 0x001700, length = 0x000080
19 19     RAMLS0               : origin = 0x008000, length = 0x000800
20 20     RAMLS1_AND_2        : origin = 0x008800, length = 0x001000
21 21     RAMLS2               : origin = 0x008900, length = 0x000800
22 22     RAMLS4               : origin = 0x00A000, length = 0x000800
23 23     RAMLS5               : origin = 0x00A800, length = 0x000800
24 24     RAMLS6               : origin = 0x00B000, length = 0x000800
25 25     RAMLS7               : origin = 0x00B800, length = 0x000800
26 26     RAMGS0               : origin = 0x00C000, length = 0x001000
27 27     RAMGS1               : origin = 0x00D000, length = 0x001000
28 28     RAMGS2               : origin = 0x00E000, length = 0x001000
29 29     RAMGS3               : origin = 0x00F000, length = 0x000FF8
30 30     FLASH_BANK0_SEC0    : origin = 0x080000, length = 0x001000
31 31     FLASH_BANK0_SEC1    : origin = 0x081000, length = 0x001000
32 32     FLASH_BANK0_SEC2    : origin = 0x082000, length = 0x001000
33 33     FLASH_BANK0_SEC3    : origin = 0x083000, length = 0x001000
34 34     FLASH_BANK0_SEC4    : origin = 0x084000, length = 0x001000
35 35     FLASH_BANK0_SEC5    : origin = 0x085000, length = 0x001000
36 36     FLASH_BANK0_SEC6    : origin = 0x086000, length = 0x001000
37 37     FLASH_BANK0_SEC7    : origin = 0x087000, length = 0x001000
38 38     FLASH_BANK0_SEC8    : origin = 0x088000, length = 0x001000
39 39     FLASH_BANK0_SEC9    : origin = 0x089000, length = 0x001000
40 40     FLASH_BANK0_SEC10   : origin = 0x08A000, length = 0x001000
41 41     FLASH_BANK0_SEC11   : origin = 0x08B000, length = 0x001000
42 42     FLASH_BANK0_SEC12   : origin = 0x08C000, length = 0x001000
43 43     FLASH_BANK0_SEC13   : origin = 0x08D000, length = 0x001000
    
```

图 3-1. 生成的文件 - CMD 文件

.cmd 文件的“存储器”部分中的条目包括存储器组合块。该文件显示存储器组合的大小和来源。

仅当特定段选择了有效的“加载存储器”条目时，才会显示 .cmd 文件的“段”部分中的条目。


```

device_cmd.cmd
01 01
82 82 SECTIONS
83 83 {
84 84     //
85 85     // C28x Sections
86 86     //
87 87     .reset           : > RESET, TYPE = DSECT /* not used, */
88 88     codestart        : > 0x000000
89 89
90+  //
91+  // User Sections
92+  //
93+  userSection0 { *(customUserSection0) } > RAMLS0
94+
90 95 }
91 96
92 97 #endif
93 98
94 99 /*
95 100 //=====
96 101 // End of file.
97 102 //=====
98 103 */
99 104

```

图 3-2. 生成的文件 - CMD 文件段

3.2 支持文件

附加的 `device_cmd.c`、`device_cmd.h`、`device_cmd.opt` 和 `device_cmd.cmd.genlibs` 是该工具生成的支持文件。

代码生成包括一款 LIVE DIFF 工具，该工具展示了 GUI 中的更改如何导致生成的代码发生变化。

如果您决定必须从不同的地址加载并运行段，则链接器 `cmd` 文件中生成的条目格式会自动更改。

图 3-3. 生成的文件 - CMD File Diff

停用将复制表置于 BINIT 段中后，`.c` 和 `.h` 文件也会更新，并且系统会自动生成所需的初始化代码。

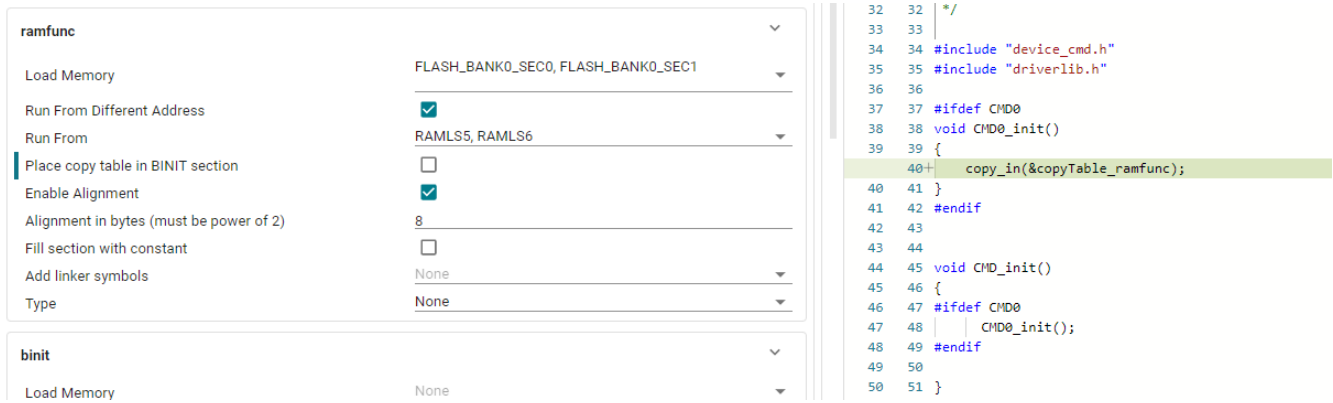


图 3-4. 生成的文件 - 复制表

您需要在应用代码中调用 **CMD_init** 函数来初始化此类区段。C2000 器件的 **device.c** 包含一个 **Device_init** 函数，如果需要，可用来调用 **CMD_init** 函数。

```

device.c
51 51 #ifdef CMDTOOL
52 52 #include "device_cmd.h"
53 53 #endif
54 54
55 55 //*****
56 56 //
57 57 // Function to initialize the device. Primarily initializes system control to a
58 58 // known state by disabling the watchdog, setting up the SYSCLKOUT frequency,
59 59 // and enabling the clocks to the peripherals.
60 60 // The function also configures the GPIO pins 22 and 23 in digital mode.
61 61 // To configure these pins as analog pins, use the function GPIO_setAnalogMode
62 62 //
63 63 //*****
64 64 void Device_init(void)
65 65 {
66 66     //
67 67     // Disable the watchdog
68 68     //
69 69     SysCtl_disableWatchdog();
70 70 #ifdef CMDTOOL
71 71     CMD_init();
72 72 #endif
73 73
74 74 #ifdef _FLASH
75 75 #ifndef CMDTOOL

```

图 3-5. 生成的文件 - C 文件

device_cmd.opt 和 **device_cmd.cmd.genlibs** 自动设置 Code Composer Studio 工程属性。

OPT 文件为 **CMDTOOL** 和活动 CMD 模块实例名称创建预定义符号。

```

device_cmd.opt
1 1 /*
2 2 * ===== device_cmd.opt =====
3 3 * Project options needed for this application's configuration
4 4 *
5 5 * NOTE, this feature requires software components configured in your
6 6 * system to correctly indicate their project properties
7 7 * needed for your specific configuration. If you find
8 8 * errors, please report them on TI's E2E forums
9 9 * (https://e2e.ti.com/) so they can be addressed in a future
10 10 * release.
11 11 *
12 12 * This file allows one to portably link applications that use SysConfig
13 13 * _without_ having to make changes to build rules when moving to a new
14 14 * device OR when upgrading to a new version of a SysConfig enabled
15 15 * product.
16 16 *
17 17 * DO NOT EDIT - This file is generated by the SysConfig tool for the
18 18 * TI C/C++ toolchain
19 19 */
20 20 -DCMDTOOL
21 21 -DCMD0
  
```

图 3-6. 生成的文件 - OPT 文件

这些预定义符号用于应用 C 代码：**device.c** 和 **device.h** 文件。

device_cmd.cmd.genlibs 文件遵循类似的路径。

```

device_cmd.cmd.genlibs
1 1 /*
2 2 * ===== device_cmd.cmd.genlibs =====
3 3 * Project options needed for this application's configuration
4 4 *
5 5 * NOTE, this feature requires software components configured in your
6 6 * system to correctly indicate their project properties
7 7 * needed for your specific configuration. If you find
8 8 * errors, please report them on TI's E2E forums
9 9 * (https://e2e.ti.com/) so they can be addressed in a future
10 10 * release.
11 11 *
12 12 * This file allows one to portably link applications that use SysConfig
13 13 * _without_ having to make changes to build rules when moving to a new
14 14 * device OR when upgrading to a new version of a SysConfig enabled
15 15 * product.
16 16 *
17 17 * DO NOT EDIT - This file is generated by the SysConfig tool for the
18 18 * TI C/C++ toolchain
19 19 */
20 20 --define=CMDTOOL
21 21 --define=CMD0
22 22
  
```

图 3-7. 生成的文件 - Genlibs 文件

4 跨器件系列迁移

使用 C2000 SysConfig 工具的用户可以利用 **SWITCH** 按钮将其设计从一个器件系列迁移到另一个器件系列。

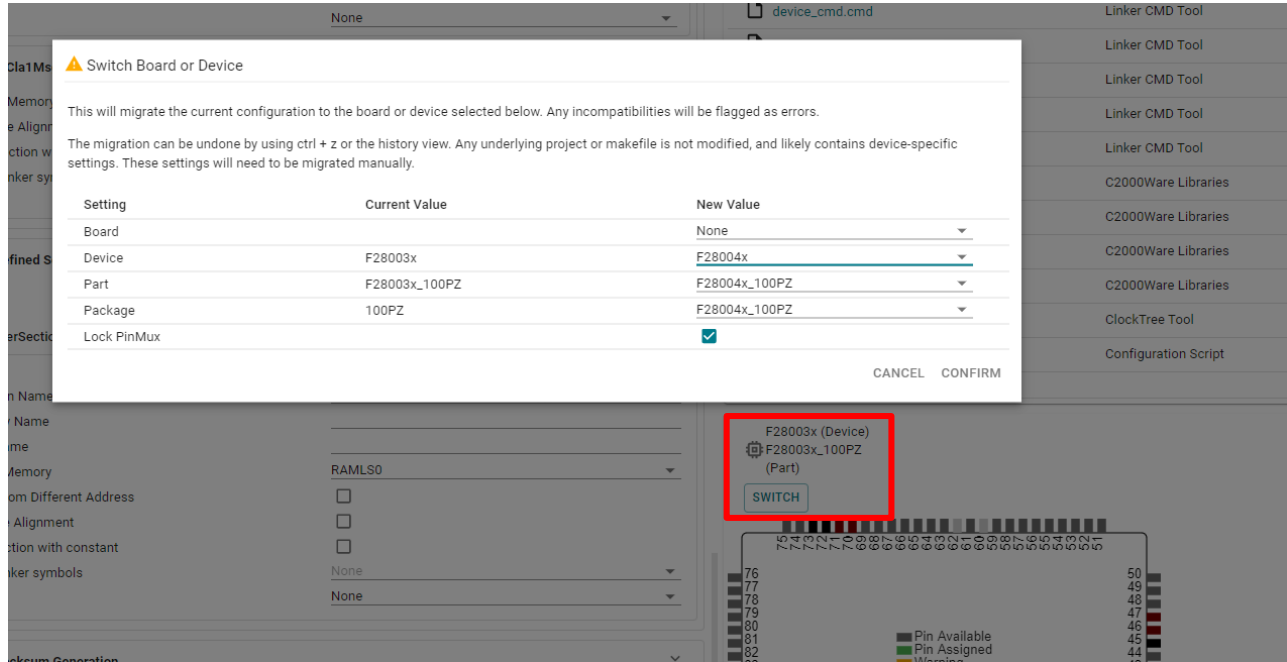


图 4-1. 器件迁移 - SWITCH

迁移完成后，系统将标识 SysConfig 工具生成的所有修改后的文件。















































<> Generated Files				
Filter: all				
File name	Category		Include in build	
  board.c	C2000Ware SysConfig	<input checked="" type="checkbox"/>		
 board.h	C2000Ware SysConfig	<input checked="" type="checkbox"/>		
  board.cmd.genlibs	C2000Ware SysConfig	<input checked="" type="checkbox"/>		
  board.opt	C2000Ware SysConfig	<input checked="" type="checkbox"/>		
  pinmux.csv	C2000Ware SysConfig	<input checked="" type="checkbox"/>		
  device.c	C2000Ware SysConfig	<input checked="" type="checkbox"/>		
  device.h	C2000Ware SysConfig	<input checked="" type="checkbox"/>		
  device_cmd.cmd	Linker CMD Tool	<input checked="" type="checkbox"/>		
 device_cmd.c	Linker CMD Tool	<input checked="" type="checkbox"/>		
 device_cmd.h	Linker CMD Tool	<input checked="" type="checkbox"/>		
 device_cmd.opt	Linker CMD Tool	<input checked="" type="checkbox"/>		
 device_cmd.cmd.genlibs	Linker CMD Tool	<input checked="" type="checkbox"/>		
 c2000ware_libraries.cmd.genlibs	C2000Ware Libraries	<input checked="" type="checkbox"/>		
 c2000ware_libraries.opt	C2000Ware Libraries	<input checked="" type="checkbox"/>		
 c2000ware_libraries.c	C2000Ware Libraries	<input checked="" type="checkbox"/>		
 c2000ware_libraries.h	C2000Ware Libraries	<input checked="" type="checkbox"/>		
  clocktree.h	ClockTree Tool	<input checked="" type="checkbox"/>		
  untitled.syscfg	Configuration Script	<input type="checkbox"/>		
18 Total Files				

图 4-2. 器件迁移 - 文件已更改

每个文件还标识所生成代码中的更改。

```

device_cmd.cmd
6 6 #define CMD0
7 7 #ifdef CMD0
8 8
9 9 MEMORY
10 10 {
11 11
12 12     RAMM0_BEGIN           : origin = 0x000000, length = 0x000002
13 13     RAMM0                 : origin = 0x000128, length = 0x0002D8
13+ 13+     RAMM0                 : origin = 0x0000F6, length = 0x00030A
14 14     RAMM1                 : origin = 0x000400, length = 0x0003F8
15 15     CLATOCPU_MSGRAM      : origin = 0x001480, length = 0x000080
16 16     CPUTOCLA_MSGRAM      : origin = 0x001500, length = 0x000080
17 17     CLATODMA_MSGRAM      : origin = 0x001680, length = 0x000080
18 18     DMATOCCLA_MSGRAM     : origin = 0x001700, length = 0x000080
19 17     RAMLS0                : origin = 0x008000, length = 0x000800
20 18     RAMLS_1_AND_2        : origin = 0x008800, length = 0x001000
21 19     RAMLS3                : origin = 0x009800, length = 0x000800
22 20     RAMLS4                : origin = 0x00A000, length = 0x000800
23 21     RAMLS5                : origin = 0x00A800, length = 0x000800
24 22     RAMLS6                : origin = 0x00B000, length = 0x000800
25 23     RAMLS7                : origin = 0x00B800, length = 0x000800
26 24+     RAMGS0               : origin = 0x00C000, length = 0x001000
27 25+     RAMGS1               : origin = 0x00D000, length = 0x001000
28 26+     RAMGS2               : origin = 0x00E000, length = 0x001000
29 27+     RAMGS3               : origin = 0x00F000, length = 0x000FF8
24+ 24+     RAMGS0               : origin = 0x00C000, length = 0x002000
25+ 25+     RAMGS1               : origin = 0x00E000, length = 0x002000
26+ 26+     RAMGS2               : origin = 0x010000, length = 0x002000
27+ 27+     RAMGS3               : origin = 0x012000, length = 0x001FF8

```

图 4-3. 器件迁移 - 文件更改

文件差异指示由于迁移而发生的所有更改。

5 总结

C2000 链接器 CMD 工具是一款直观的图形用户界面工具，可针对给定应用配置器件存储器。该工具可通过提供错误检查、自动工程设置、自动代码生成和器件系列迁移支持，显著简化用户的软件开发流程。

6 参考文献

视频系列：

- [7.1 C2000™ SysConfig：概述](#)
- [7.2 C2000™ SysConfig：入门](#)
- [7.3 C2000™ SysConfig：PinMux](#)
- [7.4 C2000™ SysConfig：板级支持](#)
- [7.5 C2000™ SysConfig：示例演练](#)
- [7.6 C2000™ SysConfig：在 10 分钟内迁移 C2000 器件](#)

C2000 SysConfig 的优势：

- 德州仪器 (TI)：[利用 SysConfig 并借助 C2000™ 实时 MCU 加速开发](#)

应用手册 - C2000 SysConfig 分步使用指南：

- 德州仪器 (TI)：[C2000 SysConfig](#)

软件入门指南：

- https://software-dl.ti.com/C2000/docs/software_guide/c2000_sysconfig.html
- https://software-dl.ti.com/ccs/esd/documents/sdto_cgt_Linkers-Command-File-Primer.html

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

Copyright © 2023，德州仪器 (TI) 公司