

AT32微控制器上移植GUIX

前言

本应用笔记主要演示GUIX移植到AT32 MCU的过程和方法。

本应用笔记在ThreadX OS基础上进行讲解，因此建议读者先看AT32的如下应用指南：
[AN0079_AT32_MCU_On_ThreadX_OS](#)。

注：本应用笔记对应的代码是基于雅特力提供的V2.x.x 板级支持包（BSP）而开发，对于其他版本BSP，需要注意使用上的区别。

支持型号列表：

支持型号	AT32F4 系列
------	-----------

目录

1	ThreadX 在 MDK 移植.....	5
1.1	软件资源准备.....	5
1.2	MDK 源码工程配置.....	5
1.3	GUIX Studio 生成应用文件.....	8
1.4	应用代码解析.....	12
2	示例快速使用.....	15
2.1.1	硬件资源.....	15
2.1.2	软件资源.....	15
2.1.3	demo 使用.....	15
3	文档版本历史.....	17

表目录

表 1. 文档版本历史 17

图目录

图 1. GUIX common/ports/ports 文件夹.....	5
图 2. MDK5 导入 GUIX C 文件.....	6
图 3. MDK5 导入 app 文件.....	7
图 4. MDK5 添加 GUIX 头文件路径.....	8
图 5. GUIX Studio 安装包下载.....	8
图 6. GUIX Studio create new project.....	9
图 7. GUIX Studio config project.....	9
图 8. GUIX Studio 加载 test 控件.....	10
图 9. GUIX Studio 配置 Startup.....	10
图 10. GUIX Studio 仿真.....	11
图 11. GUIX Studio 生成 C 源文件.....	11
图 12. GUIX Studio 生成 C 源文件.....	12
图 13 串口助手打印信息.....	16
图 14 串口助手打印信息.....	16

1 ThreadX 在 MDK 移植

1.1 软件资源准备

移植前需要提前准备好的软件资源有：

- AT32 AN0079 应用指南：雅特力论坛或雅特力官网下载
- GUIX 源码和 GUIX Studio 工具
Github 获取地址：<https://github.com/azure-rtos/guix/releases>
- MDK5.30 及以上版本：Keil 官方下载

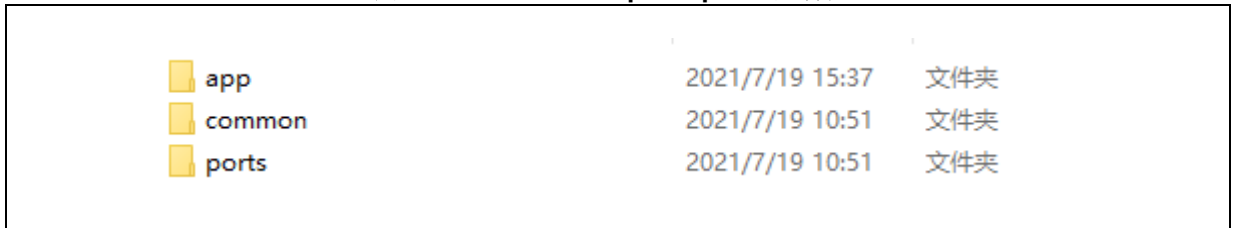
1.2 MDK 源码工程配置

STEP 1 准备 Thread OS 工程。

对于 AT32 MCU，可直接使用 AN0079 工程。

STEP 2 拷贝 GUIX 包中的 common、ports、app 文件夹到 GUIX 路径，同时建立 app 文件夹用于存放 guix 应用文件和移植文件

图 1. GUIX common/ports/ports 文件夹



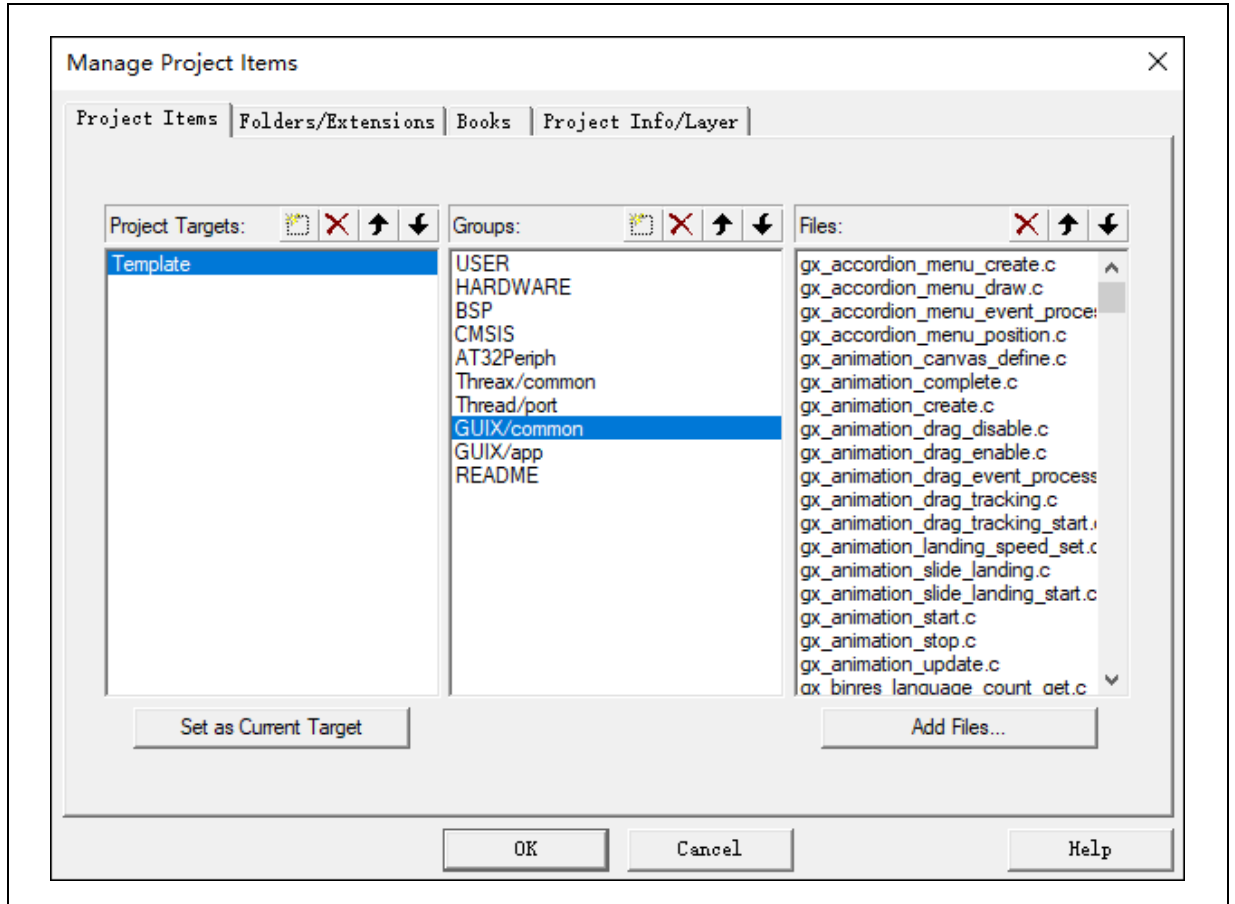
app	2021/7/19 15:37	文件夹
common	2021/7/19 10:51	文件夹
ports	2021/7/19 10:51	文件夹

其中

- common/ports 文件夹内容来自于 GUIX 包
- app 文件夹中 window_demo_resources.c、window_demo_resources.h、window_demo_specifications.c 和 window_demo_specifications.h 由 GUIX Studio 工具自动生成，后面章节会有介绍。
- app 文件夹中 guix_task.c、guix_task.h、guix_touch_task.c 和 guix_touch_task.h 为 GUIX 的移植文件。

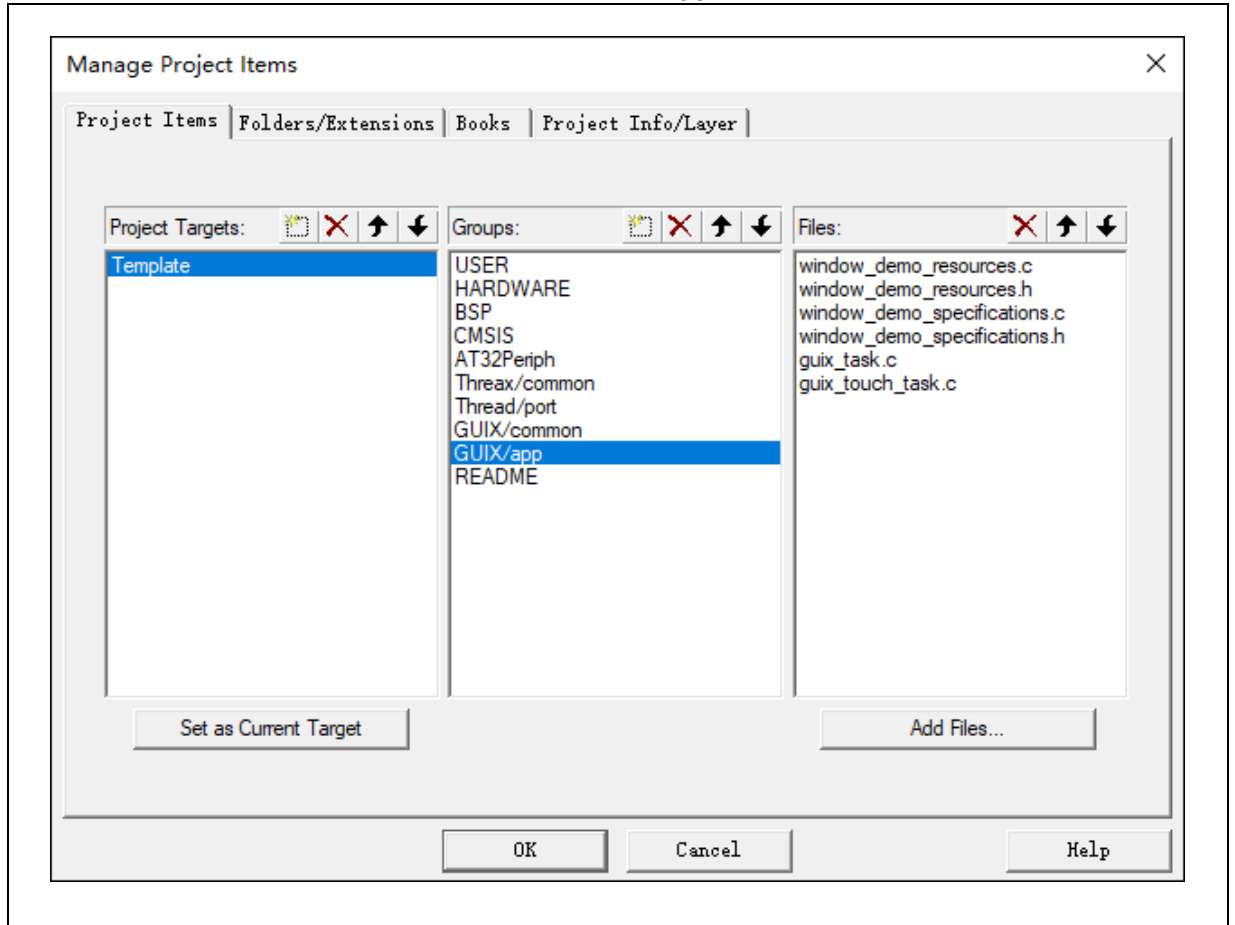
STEP 3 将 GUIX C 文件导入 MDK 工程

图 2. MDK5 导入 GUIX C 文件



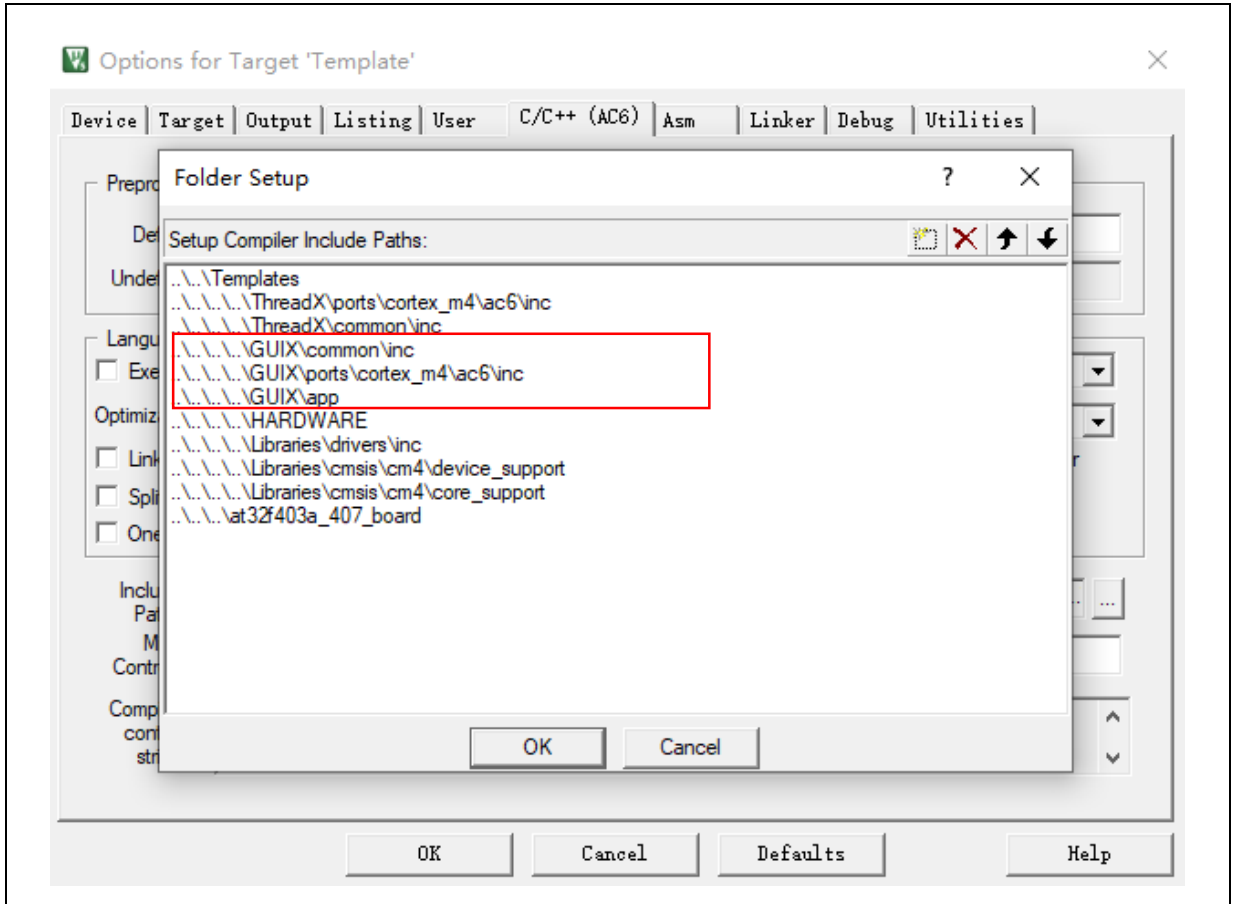
STEP 4 将 GUIX 应用程序 C 文件导入 MDK 工程
app 文件夹中 window_demo_resources.c、window_demo_resources.h、
window_demo_specifications.c 和 window_demo_specifications.h 由 GUIX Studio 工具自动生成，
后面 GUIX Studio 章节会有介绍。

图 3. MDK5 导入 app 文件



STEP 5 GUIX 相关头文件地址添加

图 4. MDK5 添加 GUIX 头文件路径



1.3 GUIX Studio 生成应用文件

GUIX studio 的使用较为简单，按如下操作即可配置生成最基本的应用代码

STEP1 下载安装并安装最新版 GUIX Studio

GUIX Studio 官方下载地址：<https://github.com/azure-rtos/guix/releases>

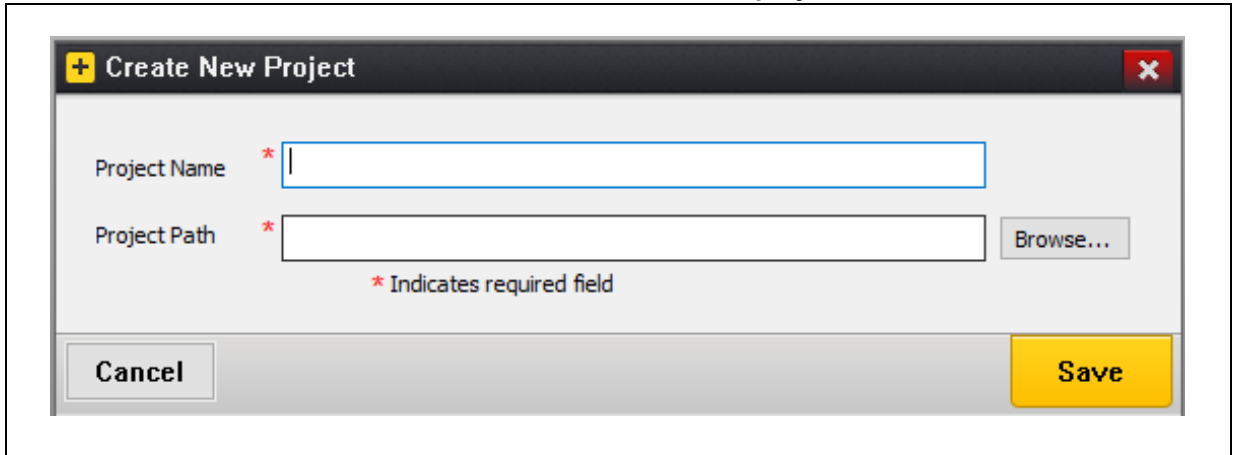
图 5. GUIX Studio 安装包下载



STEP2 配置 GUIX Studio

Project -> new project -> 输入工程名和目标路径

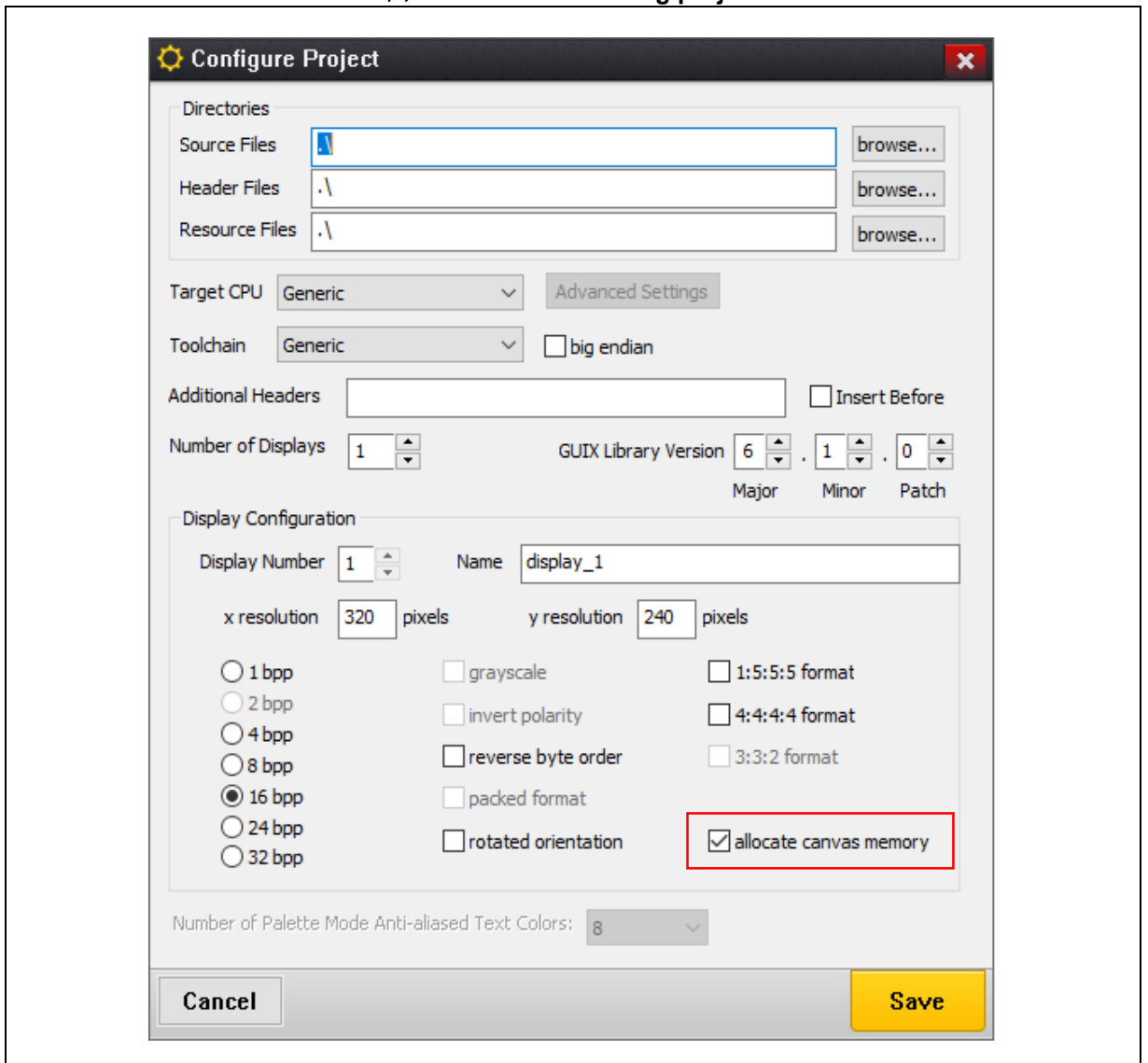
图 6. GUIX Studio create new project



STEP3 配置 GUIX Studio config project

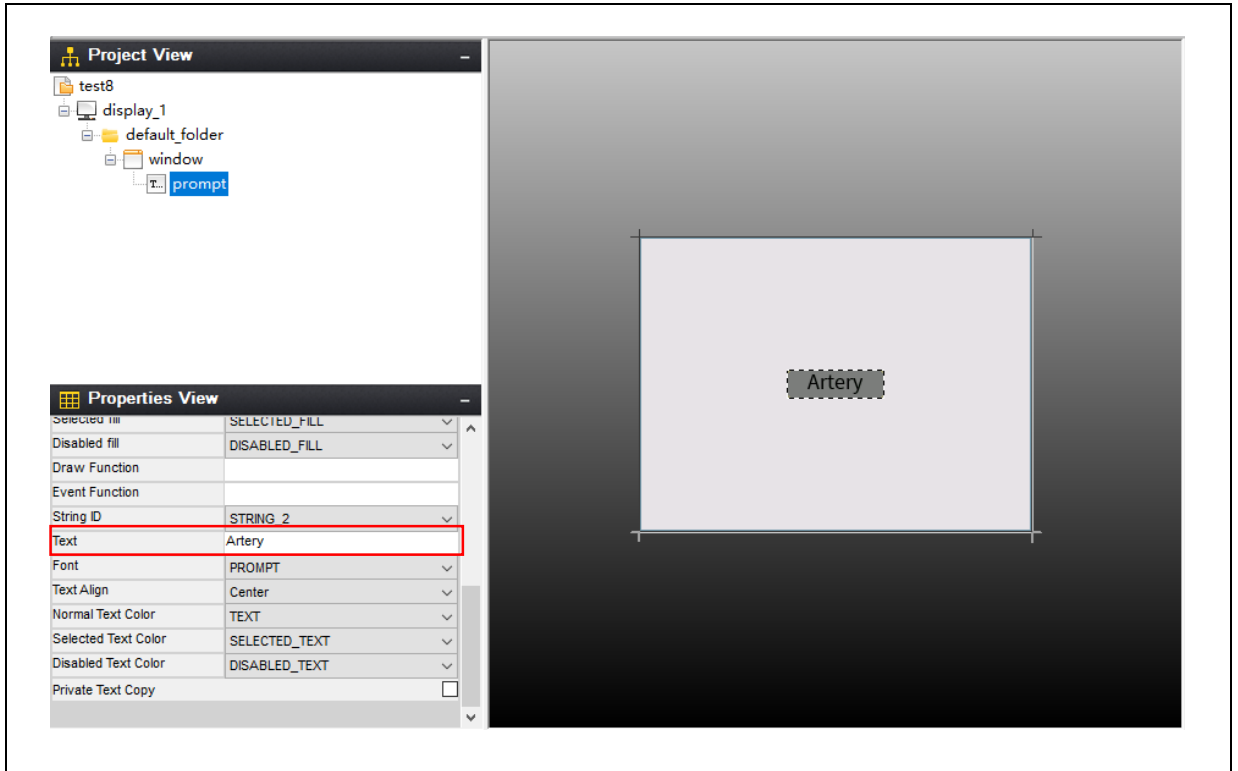
其中 allocate canvas memory 选项勾选表示使用与 LCD 同等大小的缓存作为画布缓存。

图 7. GUIX Studio config project



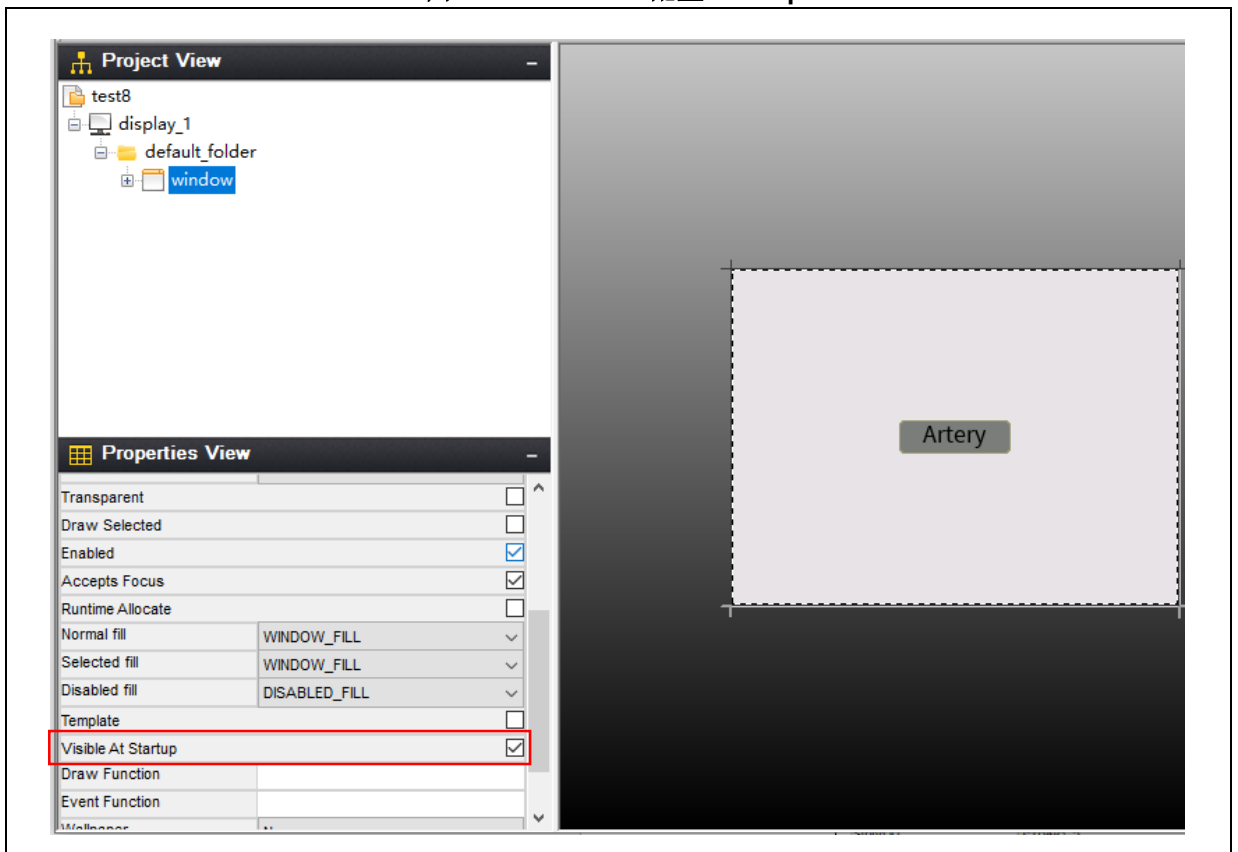
STEP4 插入 GUIX 桌面 test 控件，命名为 Artery。

图 8. GUIX Studio 加载 test 控件



STEP5 GUIXwindow 窗口配置为起始画面 Startup。
该配置勾选后才可以进行仿真。

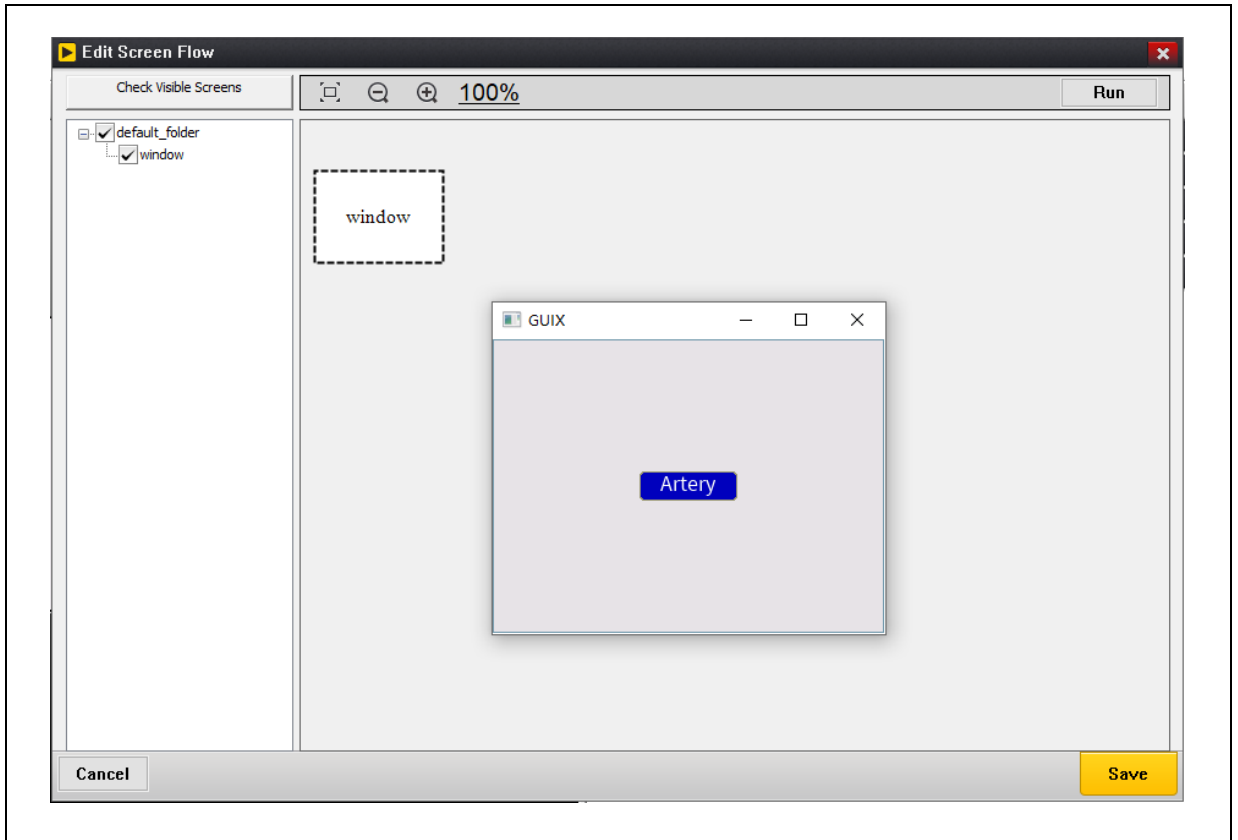
图 9. GUIX Studio 配置 Startup



STEP6 进入仿真

点击 **Configure** 菜单，选择 **Screen Flow** 选项，选择 **window** 界面，点击 **run** 进行仿真。

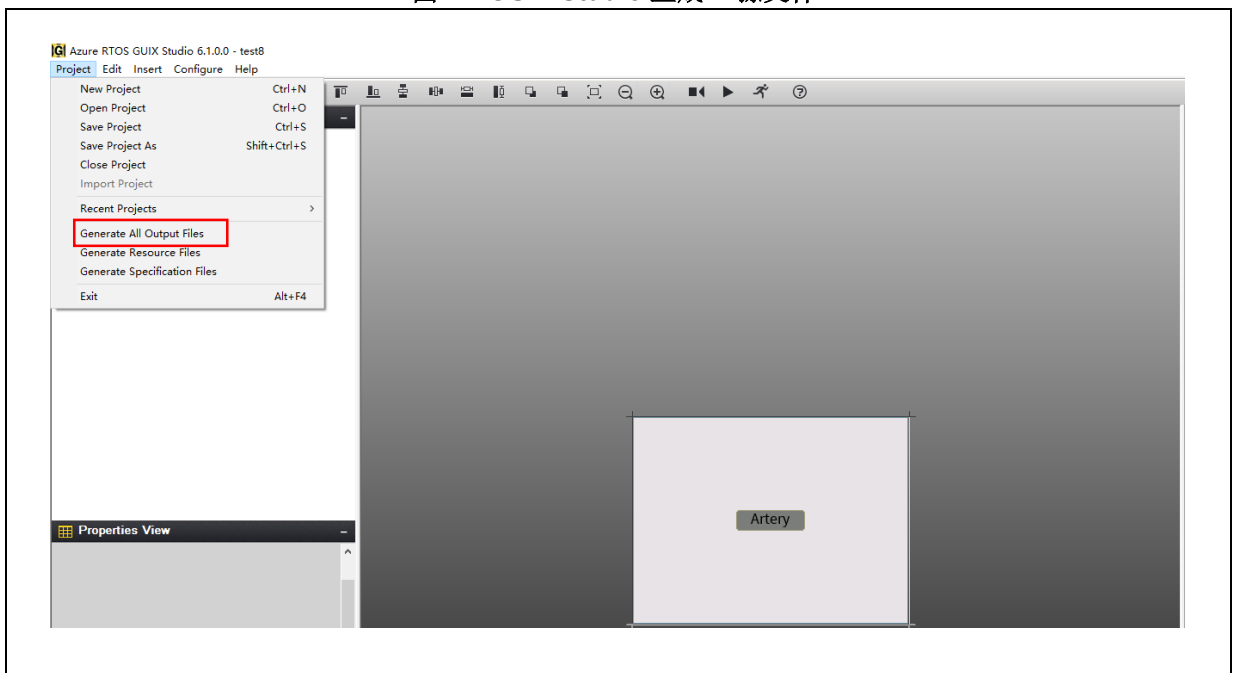
图 10. GUIX Studio 仿真



STEP6 生成 C 源文件

点击 **Project** 菜单，选择 **Generate All Output Files** 生成 C 应用源文件。

图 11. GUIX Studio 生成 C 源文件



STEP7 同步画布缓存名称

添加前文生成的 C 源文件到 app 文件夹，并注意与 DMA 源地址的缓存名称要一致。

图 12. GUIX Studio 生成 C 源文件

```

18 static GX_WIDGET *gx_studio_nested_widget_create(GX_BYTE *control, GX_CONST GX_STUDIO_WIDGET *definition, GX_WIDGET *parent);
19 WINDOW_1_CONTROL_BLOCK window_1;
20 WINDOW_CONTROL_BLOCK window;
21 GX_DISPLAY display_1_control_block;
22 GX_WINDOW_ROOT display_1_root_window;
23 GX_CANVAS display_1_canvas_control_block;
24 ULONG display_1_canvas_memory[38400];
25 extern GX_CONST GX_THEME *display_1_theme_table[];
26 extern GX_CONST GX_STRING *display_1_language_table[];
27
28 GX_STUDIO_DISPLAY_INFO window_demo_display_table[1] =
29 {
30 {
31 "display_1",
32 "display_1_canvas",
33 display_1_theme_table,
34 display_1_language_table,
35 DISPLAY_1_THEME_TABLE_SIZE,
36 DISPLAY_1_LANGUAGE_TABLE_SIZE,
37 DISPLAY_1_STRING_TABLE_SIZE,
38 240, /* x resolution */
39 320, /* y resolution */
40 &display_1_control_block,
41 &display_1_canvas_control_block,
42 &display_1_root_window,
43 display_1_canvas_memory, /* canvas memory area */
44 183600 /* canvas memory size in bytes */
45 }
46 };

```

```

18 /*
19 *****
20 * 函数名: at32_monochrome_buffer_toggle
21 * 功能说明: XMC 8080屏绘制，使用MDA方式直接做整个屏的重绘
22 * 形参: 无
23 * 返回值: 无
24 *****
25 */
26 static void at32_monochrome_buffer_toggle(GX_CANVAS *canvas, GX_RECTANGLE *dirty)
27 {
28 /* 防止警告 */
29 (void) canvas;
30 (void) dirty;
31
32 LCD_SetBlock(0,0,240-1,320-1);
33 DMA1_CHANNEL1->ctrl  = ~(uint16_t)1;
34 DMA1_CHANNEL1->paddr  = (uint32_t)display_1_canvas_memory;
35 DMA1_CHANNEL1->dtcnt  = 38400;
36 DMA1_CHANNEL1->ctrl  |= (uint16_t)1;
37 while(dma_flag_get(DMA1_FDT1_FLAG) !=SET);
38 dma_flag_clear(DMA1_FDT1_FLAG);
39 }

```

STEP8 在 MCU 端编译调试

编译 GUIX 工程文件，并下载到 AT32 MCU 上观察效果。

1.4 应用代码解析

main 函数介绍:

```

int main(void)
{
    GPIO_InitType GPIO_InitStructure;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);

    /*initialize Delay Function*/
    //初始化 delay 函数，使用 systick 作为时钟源。
    //主要用于 LCD 和触摸屏初始化，在 OS 启动后请勿调用 Delay 函数，
    //因为 OS 也是使用的 systick 作为时钟源。
    Delay_init();
}

```

```

//初始化 LED 和按键
AT32_Board_Init();

//初始化 LED 和按键
UART_Print_Init(115200);
printf("init ok\r\n");

//初始化 LCD 触摸屏
TOUCH_PIN_Init();
//初始化 LCD
LCD_init();
LCD_Clear(WHITE);

/* Enter the ThreadX kernel. */
tx_kernel_enter( );

for(;;)
{

}

int main(void)
{

//初始化系统时钟为 240 MHz
system_clock_config();

//初始化 LED 和按键
at32_board_init();

//初始化 USART1 (PA9) 用作打印
usart1_init(115200);
printf("init ok\r\n");

//初始化触摸屏
TOUCH_PIN_Init();

//初始化 LCD
LCD_init();
LCD_Clear(WHITE);

/* Enter the ThreadX kernel. */
tx_kernel_enter( );

for(;;)
{

}

}

```

■ 相关任务介绍

```

//按键任务, 检测到 USER key 按下后, LED3 toggle, 并且会有串口打印信息
AppTaskUserIF

//串口打印任务, 每个 1000 ms 打印 cnt++
AppTaskCOM

//guix 显示任务
AppTaskGUI

//触摸检测任务
touch_thread_entry

```

■ `buffer_toggle` 函数介绍:

该函数主要用于更新画布，采用的是 DMA 方式，DMA 传送宽度为 word (32 bit)，硬件自动将一次传送分解成两次 16bit 宽度按传送。

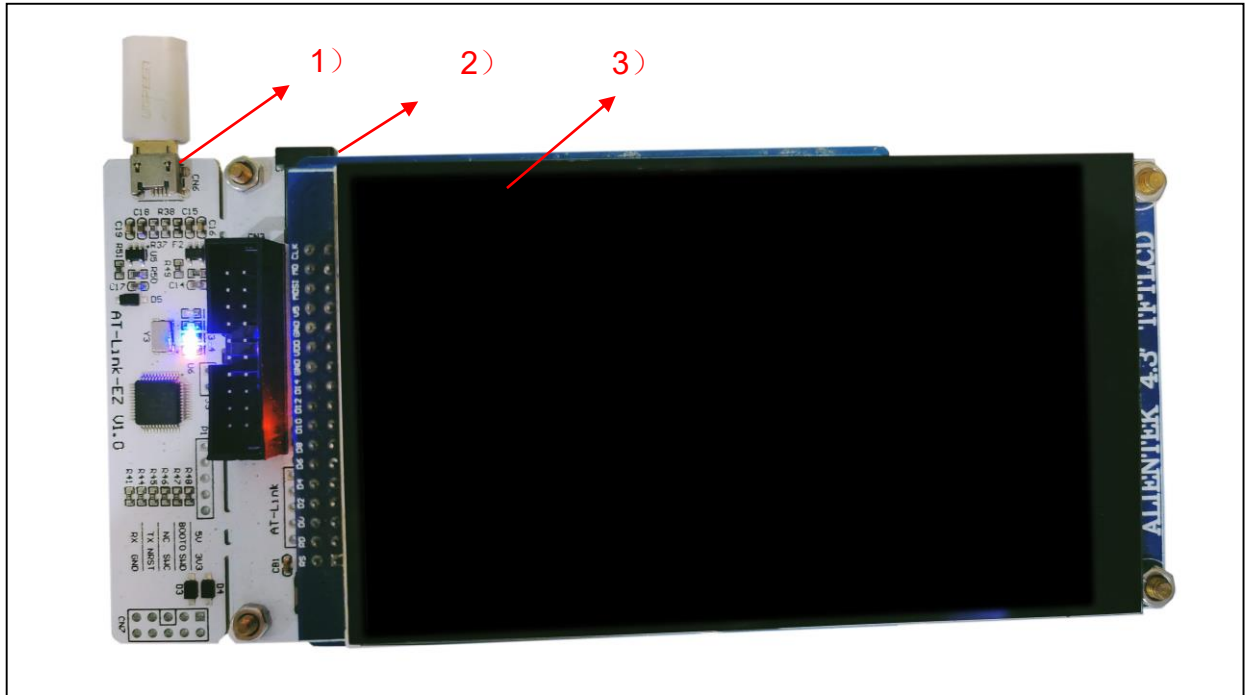
```
static void at32_monochrome_buffer_toggle(GX_CANVAS *canvas, GX_RECTANGLE *dirty)
```

2 示例快速使用

2.1.1 硬件资源

- 1) AT Link虚拟串口
- 2) AT-START-F403A V1.0 实验板
- 3) 8080触摸屏及转接板

图 11. AT-START-F403A V1.0 实验板



注：文档中是基于AT32F403A的硬件条件为例，demo源代码还包括AT32其他型号，请编译烧录在相应AT-START开发板运行即可。

2.1.2 软件资源

- 1) SourceCode
 - AT32F4xx_GUIX_V2.0.0.zip
- 2) Doc
 - AN0079_AT32_MCU_On_GUIX_ZH_V2.0.0

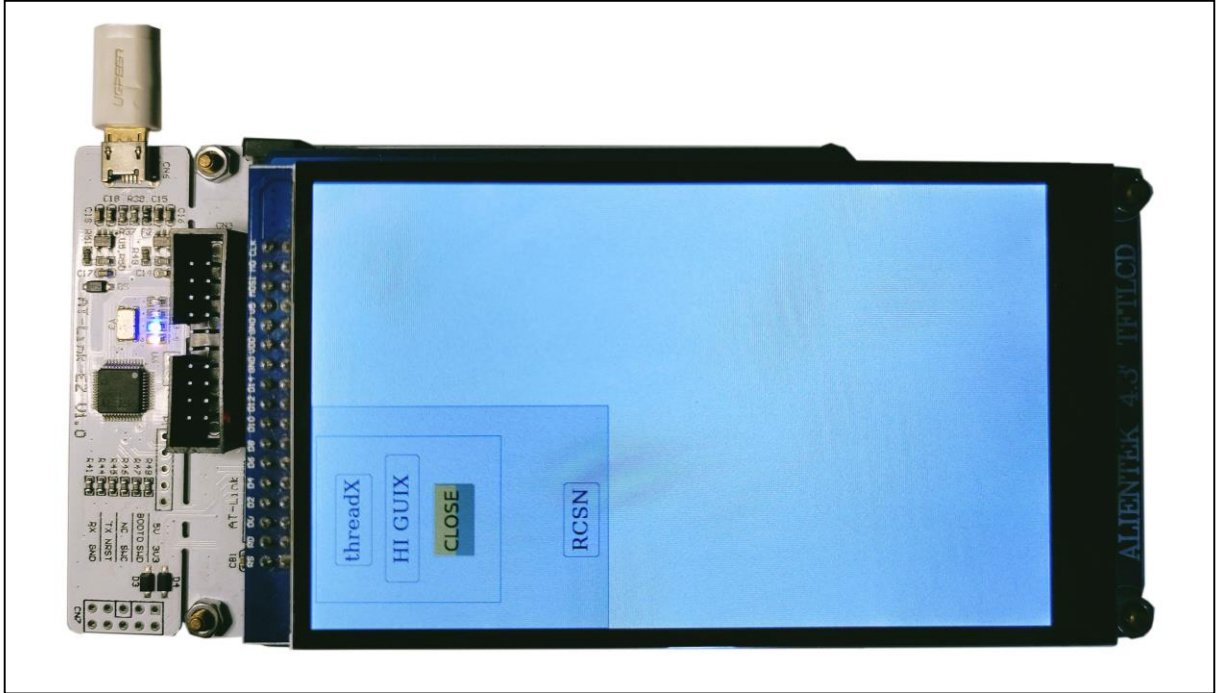
注：所有project都是基于keil 5而建立，若用户需要在其他编译环境上使用，请参考AT32xxx_Firmware_Library_V2.x.x\project\at_start_xxx\templates中各种编译环境（例如IAR6/7,keil 4/5）进行简单修改即可。

2.1.3 demo 使用

- 1) 通过USB下载线连接AT-START，并在串口助手找到AT Link虚拟串口
- 2) 通过ICP tool将AT32 MCU的224 K SRAM开启（默认96 K）
- 3) 打开AT32F4xx_GUIX工程源程序，编译后下载到实验板
- 4) 观察LED3/LED4闪烁状态和串口助手打印信息
- 5) 观察LCD显示画面并触摸UI显示按钮

LCD显示画面如下图

图 13 串口助手打印信息



串口助手打印信息，Task Com表示printf任务，Task Button表示按键任务检测到user 按键输入。

图 14 串口助手打印信息



视频效果展示

<https://b23.tv/liDto6>

3 文档版本历史

表 1. 文档版本历史

日期	版本	变更
2022.12.03	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损失的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独立负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。

© 2022 雅特力科技 保留所有权利