ZS110A-硬件外设示例

发布 ***1.1.0***

2019 年 **03** 月 **04** 日

目录

CHAPTER 1

文档介绍

1.1 文档目的

介绍硬件外设示例的通用模板以及多个硬件外设示例。

1.2 术语说明

表 1.1: 术语说明

术语	说明
ZEPHYR	为所有资源受限设备，构建了针对低功耗、小型内存微处理器设备而进行优化的物联网嵌入式小型、可扩展的实时操作系统（RTOS）

1.3 参考文档

- http://docs.zephyrproject.org/

1.4 版本历史

表 1.2: 版本历史

日期	版本	注释	作者
2018-08-22	1.0	初始版本	ZS110A 项目组
2019-03-04	1.1	增加 gpio 按键和 uart pipe 示例	ZS110A 项目组

CHAPTER 2

硬件外设示例概述

2.1 概述

硬件外设示例运行在 zephyr 多任务环境下。为了调试方便，zephyr 的串口打印功能默认打开了。

示例都是基于模板开发的。模板已经配置好了 zephyr 环境和串口打印相关的内核模块和驱动。

示例开发只需要在模板基础上添加被测硬件模块的驱动、应用代码，以及一些简单的配置就可以了。

2.2 示例模板

下面以 pwm 示例来介绍示例模板。

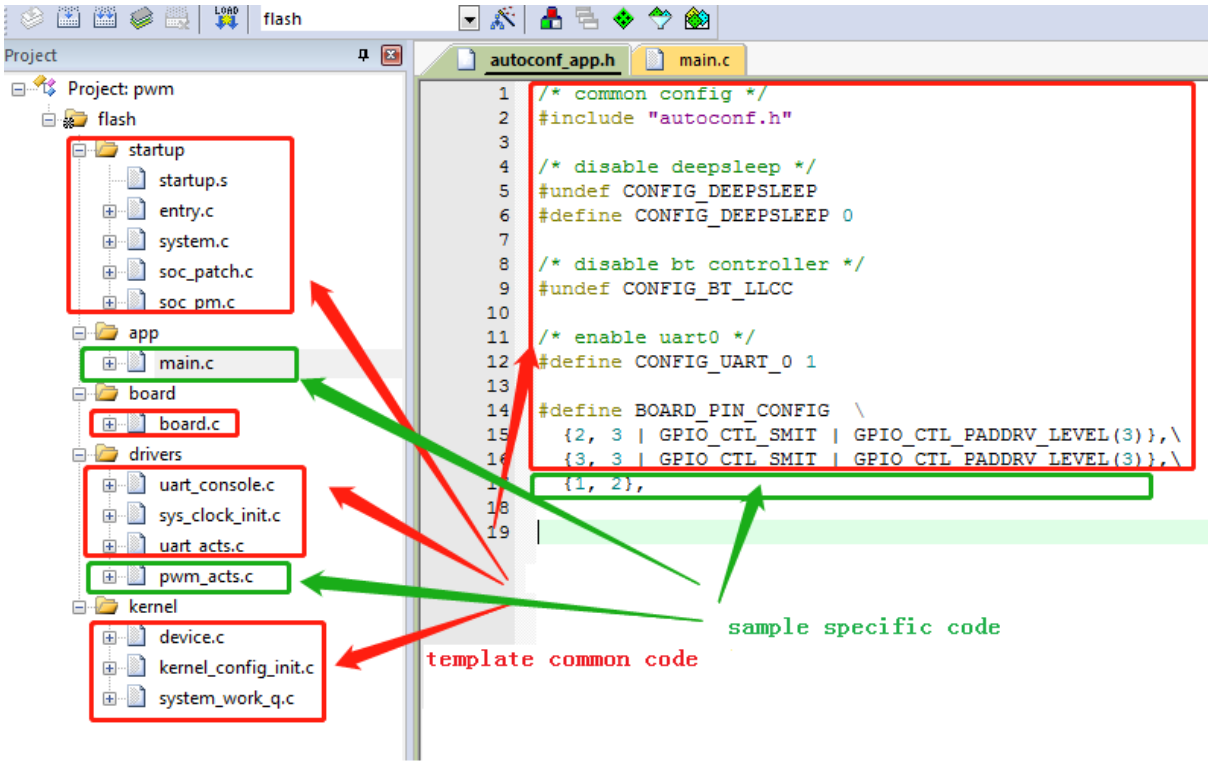

图 2.1: pwm 示例工程

红框部分为模板，示例不要修改。绿框部分为硬件外设示例相关部分。

硬件外设示例

1. 驱动: pwm_acts.c
2. 应用示例: main.c
3. 配置：autoconf_app.h

配置文件 autoconf_app.h 中包含模板定义部分和硬件外设相关部分。

1. 模板定义了 3 个配置，示例不要修改：

- 系统不进入 deepsleep 模式

```
#undef CONFIG_DEEPSLEEP
#define CONFIG_DEEPSLEEP 0
```

- 不打开蓝牙功能

```
#undef CONFIG_BT_LLCC
```

- 使用 uart0 作为打印串口

```
/* enable uart0 */
#define CONFIG_UART_0 1

#define BOARD_PIN_CONFIG        \
```

(continues on next page)

(续上页)

```
        {2, 3 | pwm_CTL_SMIT | pwm_CTL_PADDRV_
↪LEVEL(3)},\
        {3, 3 | pwm_CTL_SMIT | pwm_CTL_PADDRV_
↪LEVEL(3)},\
```

2. 与 pwm 示例硬件相关配置：

```
#define BOARD_PIN_CONFIG        \
        {1, 2},
```

CHAPTER 3

gpio 示例

GPIO1 配置为输出,GPIO0 配置为输入，两个 GPIO 短接。GPIO1 不断切换输出高低电平，GPIO0 接收电平并与 GPIO 输出的电平做比较。如果电平相同，则在串口打印“Consistent GPIO read/write value”；否则打印“Inconsistent GPIO read/write value”。

3.1 设置

源码目录

- \samples\peripheral\gpio

硬件设置

- GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口
- 用杜邦线连接 GPIO0 和 GPIO1

3.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

- 波特率: 115200
- 数据位：8
- 奇偶校验：None

- 停止位：1
- 流控：None

3. 按 reset 键后观察串口打印

   串口会持续打印“Consistent GPIO read/write value”。

```
GPIO testing
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
Consistent GPIO read/write value
```

CHAPTER 4

pwm 示例

示例中 GPIO1 配置为 PWM 功能, 与 LED1 相连。通过调整 PWM 的占空比来改变 led 的闪烁频率。

4.1 设置

源码目录

- \samples\peripheral\pwm

硬件设置

- 用杜邦线连接 GPIO1(PWM1) 和 LED1

4.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 按 reset 键后观察 LED1

   LED1 会从慢闪变为快闪（0.5Hz->1Hz->2Hz->4Hz->8Hz->16Hz），然后再变为慢闪（16Hz->8Hz->4Hz->2Hz->1Hz->0.5Hz）。反复如此一致循环。

CHAPTER 5

dma 示例

通过 dma0 将 ram 中数据由源地址搬到目的地址，并比较源地址和目的地址的数据。

5.1 设置

源码目录

- \samples\peripheral\dma

硬件设置

- GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口

5.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

- 波特率: 115200
- 数据位：8
- 奇偶校验：None
- 停止位：1
- 流控：None

3. 按 reset 键后观察串口打印

- 源地址和目的地址的数据完全相同，串口打印“compare pass”。

```
dma testing
test_task: chan 0
Preparing DMA Controller: Chan_ID=0
Starting the transfer
DMA half transfer done
DMA transfer done
rxdata: It is harder to be kind than to be wise
compare pass
```

- 源地址和目的地址的数据存在差异，串口打印“compare error”。

CHAPTER 6

nvram 示例

示例中示范了 nvram 驱动 4 个接口的使用方法

- nvram_config_set
- nvram_config_set_factory
- nvram_config_get_factory
- nvram_config_get

6.1 设置

源码目录

- \samples\peripheral\nvram

硬件设置

- GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口

6.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

- 波特率: 115200

- 数据位：8
- 奇偶校验：None
- 停止位：1
- 流控：None

3. 按 reset 键后观察串口打印

   串口打印如下:

```
NVRAM testing
region Factory Config: base addr 0x3f0000 total size␣
↪0x4000
region offs 0x100, size 0x100, data size 0x1c, age 0x2
[ 0] config addr 0x3f0100 size 0x1c data size 0x7
                 config name: fcfg1
                 config data:
00000000: 66 64 61 74 61 31 00                          ␣
↪     fdata1.
region User Config: base addr 0x3f4000 total size 0xc000
region offs 0x100, size 0x100, data size 0x1c, age 0x2
[ 0] config addr 0x3f4100 size 0x1c data size 0x7
                 config name: ucfg1
                 config data:
00000000: 75 64 61 74 61 31 00                          ␣
↪     udata1.
data_len 7, fcfg1: fdata1
data_len 7, fcfg1: fdata1
data_len 7, ucfg1: udata1
region Factory Config: base addr 0x3f0000 total size␣
↪0x4000
region offs 0x200, size 0x100, data size 0x1c, age 0x3
[ 0] config addr 0x3f0200 size 0x1c data size 0x7
                 config name: fcfg1
                 config data:
00000000: 66 64 61 74 61 31 00                          ␣
↪     fdata1.
region User Config: base addr 0x3f4000 total size 0xc000
region offs 0x200, size 0x100, data size 0x1c, age 0x3
[ 0] config addr 0x3f4200 size 0x1c data size 0x7
                 config name: ucfg1
                 config data:
00000000: 75 64 61 74 61 31 00                          ␣
↪     udata1.
```

CHAPTER 7

i2c 示例

示例中通过 i2c 读写 eeprom 来示范 i2c 操作方法。示例中流程如下：

- 先写入 32byte 到 eeprom
- k_sleep(10) 确保写操作完成
- 从 eeprom 读出 32byte，并与写入值比较。

7.1 设置

源码目录

- \samples\peripheral\i2c

硬件设置

- GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口
- GPIO8(I2C0_SCL)、GPIO9(I2C0_SDA)、VCC、GND 分别连到 EEPROM 子板上对应的 pin 脚

7.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

- 波特率: 115200
- 数据位：8
- 奇偶校验：None
- 停止位：1
- 流控：None

3. 按 reset 键后观察串口打印 读写数据完全正确时串口打印“E2PROM test pass!”。如果出错会有相应的打印提示。

CHAPTER 8

mxkeypad 示例

示例中示范了如何使能 mxkeypad 成为 input 设备，以及如何注册 input 设备的 notify 接口

8.1 设置

源码目录

- \samples\peripheral\mxkeypad

硬件设置

- GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口
- GPIO10(KEY1)、GPIO11(KEY2)、GPIO8(KEY6)、GPIO9(KEY7) 分别连接到 KEY0/KEY1/KEY2/KEY3

8.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

- 波特率: 115200
- 数据位：8
- 奇偶校验：None

- 停止位：1
- 流控：None

3. 按 reset 键后观察串口打印 6 个 key，按不同 key 串口会打印不同的键值。

```
Test martrix keypad driver
Wait for martrix keypad key pressed
mxkeypad input value
        type 1, code 7, value 1
mxkeypad input value
        type 1, code 7, value 1
mxkeypad input value
        type 1, code 7, value 0
mxkeypad input value
        type 1, code 9, value 1
mxkeypad input value
        type 1, code 9, value 1
mxkeypad input value
        type 1, code 9, value 0
mxkeypad input value
        type 1, code 6, value 1
mxkeypad input value
        type 1, code 6, value 1
mxkeypad input value
        type 1, code 6, value 0
mxkeypad input value
        type 1, code 2, value 1
mxkeypad input value
        type 1, code 2, value 1
mxkeypad input value
        type 1, code 2, value 0
mxkeypad input value
        type 1, code 3, value 1
mxkeypad input value
        type 1, code 3, value 1
mxkeypad input value
        type 1, code 3, value 0
mxkeypad input value
        type 1, code 8, value 1
mxkeypad input value
        type 1, code 8, value 1
mxkeypad input value
        type 1, code 8, value 0
```

CHAPTER 9

adckey 示例

示例中示范了如何使能 adckey 成为 input 设备，以及如何注册 input 设备的 notify 接口

9.1 设置

源码目录

- \samples\peripheral\adckey

硬件设置

- GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口
- GPIO1 连接到 AD_KEY

9.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

- 波特率: 115200
- 数据位：8
- 奇偶校验：None

- 停止位：1
- 流控：None

3. 按 reset 键后观察串口打印 3 个 adckey，按不同 key 串口会打印不同的键值。

```
Test adc keypad driver
Wait for adc keypad key pressed
adckey input value
        type 1, code 163, value 1
adckey input value
        type 1, code 163, value 0
adckey input value
        type 1, code 165, value 1
adckey input value
        type 1, code 165, value 0
adckey input value
        type 1, code 174, value 1
adckey input value
        type 1, code 174, value 0
```

CHAPTER 10

button 示例

示例中示范了 gpio 用作按键的使用方法

10.1 设置

源码目录

- \samples\peripheral\button

硬件设置

- GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口
- GPIO7 连接到 KEY0

10.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

- 波特率: 115200
- 数据位：8
- 奇偶校验：None
- 停止位：1

- 流控：None

3. 按 reset 键后观察串口打印

   按下按键 KEY7，串口会出现如下打印:

```
button test
pin=7 Button pressed at 54451630
pin=7 Button pressed at 76449812
pin=7 Button pressed at 89106967
pin=7 Button pressed at 99299354
pin=7 Button pressed at 103368394
pin=7 Button pressed at 108253831
pin=7 Button pressed at 112788719
pin=7 Button pressed at 116798099
pin=7 Button pressed at 120734317
pin=7 Button pressed at 125611915
pin=7 Button pressed at 133481859
pin=7 Button pressed at 140830516
```

CHAPTER 11

uart pipe 示例

示例中示范了 uart pipe 的使用方法，使用到了两个串口: UART0, UART1。UART0 用作数据收发，UART1 用作打印。UART0 每收到 20 个字节的数据会对其进行回显。

11.1 设置

源码目录

- \samples\peripheral\uart_pipe

硬件设置

- 通信口: GPIO2(UART0_RX)、GPIO3(UART0_TX)、VCC、GND 分别连接到 PC 的串口
- Debug 口: GPIO5(UART1_RX)、GPIO4(UART1_TX)、VCC、GND 分别连接到 PC 的另一个串口

11.2 测试

测试步骤如下：

1. 编译并烧写程序

2. 打开 pc 端串口调试助手，选择 Debug 口对应的 com 口，配置如下：

- 波特率: 115200
- 数据位：8
- 奇偶校验：None

- 停止位：1
- 流控：None

3. 打开 pc 端串口调试助手，选择通信口对应的 com 口，配置如下：
   - 波特率: 115200
   - 数据位：8
   - 奇偶校验：None
   - 停止位：1
   - 流控：None
4. 按 reset 键后观察串口打印

   通过步骤 3 中的串口调试助手向 UART0 发送数据，如字符串”12345678901234567890”，将会出现如下打印：

```
uart pipe test
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
received data: 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36␣
↪37 38 39 30
```

List of Figures

List of Tables