API 接口说明

变更记录

变更时间	变更内容	版本号	提交变更者
2019.3.11	初版	0.0.1	Ender.Wigg
2019.3.15	支持多FPGA固化，增加lib配置宏	0.0.2	Ender.Wigg
2019.3.20	将CAN中断函数放置到hhd32f10x_it.c	0.0.3	Ender.Wigg
2019.3.25	增加切换网口说明	0.0.4	Ender.Wigg

基本信息描述

基本参数

系统主频	**60MHZ**
网络连接速度	**100M**
系统调度定时器中断周期	**10ms**
CAN波特率	**1Mbps**

可用外设

- GPIO
- MAC
- CAN
- SPI

##系统初始化

系统基本功能初始化

```
1  void base_init(void)
```

描述：完成系统运行环境的构建，具体包括，相关外设的初始化，根据GPIO 引脚分配表初始化相关引脚，设置系统调度定时器中断等功能。在 main 函数中**必须**首先调用该函数。

初始化的外设如下：

GPIO（根据`pins_table`进行初始化）

MAC

SPI

SystemTick

构建运行环境：

LWIP 协议栈初始化

创建 FPGA 远程调试 网络服务器

创建 FPGA 配置文件固化 网络服务器

参数：void

返回值： void

GPIO

GPIO 初始化

GPIO初始化采用 pin描述结构，即期望使用的GPIO引脚需要在 `pins_table` pin描述结构中进行描述，库会自动的对该引脚进行初始化, pin描述接口需要借助宏`__HHD_PIN`来完成填充。例如：

```
1     __HHD_PIN(C,    7,    0,    INPUT,
   GPIO_Mode_DEF)  //ARM_FPGA1_IO1
```

原型：

```
#define __HHD_PIN(gpio, gpio_index, af, dir,mode)\
            {RCC_APB2Periph_GPIO##gpio,\
            GPIO##gpio,\
            PIN##gpio_index,\
            &IOCON->P##gpio##gpio_index,\
            af,\
            dir,\
            mode\
            }
```

参数：

gpio：引脚所在的PORT,[A,B,C,D,E,F]

gpio_index：引脚在组内的序号[0,15]

af：引脚服用功能，一般作为GPIO af 填0

dir：引脚方向，输入INPUT，输出 OUTPUT

mode：此处填默认值 GPIO_Mode_DEF

例如：将PD10 配置为输出

```
__HHD_PIN(D,    10,    0,    OUTPUT,
GPIO_Mode_DEF)
```

GPIO引脚控制

```
void GPIO_WriteBit(HHD32F_GPIO_TypeDef* port, uint16_t
GPIO_Pin, BitAction BitVal)
```

描述：对指定的引脚控制输出高低电频

参数：

port, 该GPIO所在的组，[GPIOA, GPIOB, GPIOC, GPIOD, GPIOE, GPIOF]

GPIO_Pin, 引脚组内编号，[0,15]

BitVal，输出电平，输出低电平Bit_RESET，输出高电平 Bit_SET 其中 枚举 BitAction定义如下：

```
typedef enum
{ Bit_RESET = 0,
  Bit_SET
}BitAction;
```

返回值：void

SPI

使用引脚

针对采集板

ARM 引脚名称	SPI 功能	FPGA 引脚名称
PC8	CS	ARM_FPGA1_IO1
PC10	CLK	ARM_FPGA1_IO2
PC11	MISO	ARM_FPGA1_IO3
PC12	MOSI	ARM_FPGA1_IO4

针对4个FPGA板

ARM 引脚名称	SPI 功能	FPGA 引脚名称
PC7	CS	ARM_FPGA1_IO1
PC8	CLK	ARM_FPGA1_IO2
PC10	MISO	ARM_FPGA1_IO3
PD11	MOSI	ARM_FPGA1_IO4

SPI 接口初始化

```
1  void SPI_To_FPGA_Init(void)
```

描述：函数用于初始化用于与FPGA通信的SPI接口

参数：void

返回值：void

SPI 连续写据到FPGA

```
1  int SPI_To_FPGA_Wirte(uint8_t fpga, uint8_t addr,
   uint8_t *data, int len)
```

描述：将data指向的数据的前len个字节写到指定的FPGA，addr 为写入到FPGA的起始地址，地址会自动向后递增

参数：

fpga，选中的FPGA，[0，3]

addr ，起始地址，[0, 0x3F]

data, 待写入的数据指针

len，数据长度，[0, 0x40]

返回值：写入到FPGA的数据长度

SPI 从 FPGA读数据

```
int SPI_To_FPGA_Read(uint8_t fpga, uint8_t addr,
uint8_t *data, int len)
```

描述：从选中的FPGA的addr地址开始，读出len个字节存在data所指向的位置

参数：

fpga，选中FPGA，[0,3]

addr ，起始地址，[0, 0x3F]

data, 数据缓存

len，期望读出的数据长度，[0, 0x40]

返回值：从FPGA读出数据的长度，该长度可能小于 len

CAN

CAN接口初始化

```
void can_init(HHD32F1_CAN_TypeDef *can, EN_BAUD baud,
uint32_t filterID, uint32_t mask);
```

描述：初始化指定的CAN接口

参数：

can ，can接口，即期望使用的can接口，可指定为如下两个宏中的任意一个：

`CAN1`,`CAN2`

baud，接口波特率，当前支持1Mbps和500Kbps,通过如下两个枚举选择：

`CAN_BAUD_500K`,`CAN_BAUD_1M`

filterID，设置过滤ID，取值范围[0, 0x3FFFFFFF]

mask，设置过滤ID的掩码，如果bit置为0，则ID中的该bit会严格比较；为1则该为不经进行比较

返回值：void

CAN发送一帧数据

```
int CAN_Transmit(HHD32F1_CAN_TypeDef *can, CanTxMsg *TxMessage)
```

描述：从指定的can接口发送一帧数据

参数：can ，can接口，即使用的can接口，可指定为如下两个宏中的任意一个：

`CAN1`,`CAN2`

TxMessage，待发送的数据和帧描述信息，CanTxMsg定义如下：

```
typedef struct
{
  uint32_t StdId;  /*!< 标准帧ID，取值范围[0,7FF].*/

  uint32_t ExtId;  /*!< 扩展帧ID，取值范围[0,0x1FFFFFFF].*/
  uint8_t IDE;     /*!< 帧类型，标准帧 CAN_Id_Standard
                                扩展帧 CAN_Id_Extended */
  uint8_t RTR;     /*!< 传输消息类型   数据 CAN_RTR_Data
                                      远程帧 CAN_RTR_Remote*/
  uint8_t DLC;     /*!< 该帧所携带的数据的长度，取值范围[0,8] */
  uint8_t Data[8]; /*!< 数据负载 */
} CanTxMsg;
```

返回值：发送状态`CAN_TxStatus_Failed`发送失败

`CAN_TxStatus_Ok` 发送成功

CAN接收一帧数据

```
int CAN_Receive(HHD32F1_CAN_TypeDef *can, CanRxMsg
*RxMessage)
```

描述：从指定的can接口接收一帧数据

参数：can ，can接口，即使用的can接口，可指定为如下两个宏中的任意一个：

`CAN1`,`CAN2`

RxMessage，接收的数据和帧描述信息，CanRxMsg定义如下：

```
typedef struct
{
  uint32_t StdId;  /*!< 标准帧ID，取值范围[0,7FF].*/

  uint32_t ExtId;  /*!< 扩展帧ID，取值范围
[0,0x1FFFFFFF].*/
  uint8_t IDE;     /*!< 帧类型，标准帧 CAN_Id_Standard
                              扩展帧 CAN_Id_Extended
*/
  uint8_t RTR;     /*!< 传输消息类型   数据 CAN_RTR_Data
                                    远程帧
CAN_RTR_Remote*/
  uint8_t DLC;     /*!< 该帧所携带的数据的长度，取值范围
[0,8] */
  uint8_t Data[8]; /*!< 数据负载 */
  uint8_t FMI;     /*!< 该参数未使用*/
} CanRxMsg;
```

返回值：发送状态 `CAN_TxStatus_Ok` 发送成功

CAN接收中断

```
void CAN1_IRQHandler(void)
```

描述：CAN 接收中断处理函数，该函数定义在`hhd32f10x_it.c`

库的配置参数

库的运行相关参数在`HHD1705_lib.h`中定义，包括可选的系统主频，是否启用外设，和功能,以及导出一部分用户可能使用到的API接口。通过修改在文件中的宏，可以轻松编译功能不同库。其中，关于宏的启用和关闭有如下约定：

首字母为x的宏，表示未定义该宏

例如： `#define xCOMPILE_TO_LIB` 等价于 `#undef xCOMPILE_TO_LIB`

```
/*****************************************************
*****************************************************
  * Company: Hiwafer Technology Co., Ltd.
  *****************************************************
*****************************************************
  * 文件名称: HHD1705_lib.h
  * 功能说明:
  * 版    本: V1.0
  * 作    者: EnderWigg
  * 日    期: 2019.3.12
  *
  * 该文件用于配置库文件相关功能
  * 约定:“x”标记表示取消该宏的定义，例如 不启用CAN1, xCFG_USING_CAN1
*****************************************************
*****************************************************/
#ifndef __HHD1705_LIB_H__
#define __HHD1705_LIB_H__
#include "hhd32f10x_conf.h"


#define xCOMPILE_TO_LIB     /* 如果需要将该工程编译成库，必须使能该宏
                               如果编译为直接烧写，则不需要定义该宏*/


#define MII_MODE            /* MII mode  (走调试网口) */
#define xMII_TO_SGMII       /* MII to SGMII 功能 (走交换机)*/
#define xBUG_GMII_TO_SGMII  /* 是否监视交换机状态，不是必须要启用*/
#define xETH_100M           /* 确定网口速度，如果定义该宏，网口速度将被配置为100M，否则为10M*/
#define CFG_SYS_60MHZ       /* 指定系统主频 60MHz*/
#define xCFG_SYS_50MHZ      /* 指定系统主频 50MHz*/

#define CFG_USING_CAN1      /* 使用CAN1接口*/
#define CFG_USING_NET       /* 使用网络*/
#define CFG_USING_SPI       /* 使用SPI接口*/
// SPI 引脚选择
#define xCFG_SELECT_SPI_IO_0    /* 使用 PC9, PC10, PC11,  PC12*/
#define CFG_SELECT_SPI_IO_1     /* 使用 PC7, PC8, PC10,  PC11,  该模式支持访问4片FPGA*/
```

```
35
36  // 设备默认IP
37  #define IP_0            192
38  #define IP_1            168
39  #define IP_2            2
40  #define IP_3            198
41  // FPGA 固化功能 默认选择的FPGA
42
43  #ifdef CFG_SELECT_SPI_IO_1
44      #define FPGA0           0
45      #define FPGA1           1
46      #define FPGA2           2
47      #define FPGA3           3
48  #else
49  #define FPGA0           0
50  #define FPGA1           0
51  #define FPGA2           0
52  #define FPGA3           0
53  #endif
54
55  #define DEF_ACCESS_FPGA FPGA0
56
57  /*****************************************依赖条件检
    查
    ************************************************/
58
59  #ifdef MII_TO_SGMII       //使用MII to SGMII 必须启用
    MII
60      #ifndef MII_MODE
61          #define MII_MODE
62      #endif
63  #endif
64
65  #ifdef CFG_USING_NET
66      #ifdef CFG_USING_SPI
67          #define CFG_USING_LOAD          // 使用FPGA
    远程固化网络服务器,  (该功能需要依赖 网络和SPI)
68      #endif
69      #define CFG_USING_XVC               // 使用FPGA
    远程调试网络服务器
70  #endif
71
72  #ifdef CFG_USING_CAN1
73      #define CFG_USING_CAN1_WRITE_BACK   //MCU将通过
    CAN1回显收到的数据
74  #endif
75
```

```
76  ////////////////////////////////////////////////////////////////////////////////
    ///////
77  //                                                  API
    //
78  ////////////////////////////////////////////////////////////////////////////////
    ///////
79
80
81  /*******************************************************************************
    *******
82  *
83  *系统基础功能初始化
84  *
85  *******************************************************************************
    *******/
86  void base_init(void);
87
88  /*******************************************************************************
    *******
89  *
90  *   基于SysTick的延时
91  *
92  *******************************************************************************
    ******/
93  void Delay(uint32_t nCount);
94
95  /*******************************************************************************
    *******
96  *
97  *   获取SysTick的当前值
98  *
99  *******************************************************************************
    ******/
100
101   uint32_t Get_Tick(void);
102
103 ////////////////////////////////////////////////CAN//
    ////////////////////////////////////////////////////////////////////////////////
    ///////
104
```

```
105 /*************************************************
    *************************************************
    *******
106 *
107 *CAN 接口初始化
108 *
109 *************************************************
    *************************************************
    *******/
110 void can_init(HHD32F1_CAN_TypeDef *can, EN_BAUD
    baud, uint32_t filterID, uint32_t mask);
111
112 /*************************************************
    *************************************************
    *******
113 *
114 * 通过can 接口发送一帧数据
115 *
116 *************************************************
    *************************************************
    *******/
117 int CAN_Transmit(HHD32F1_CAN_TypeDef *can, CanTxMsg
    *TxMessage);
118
119 /*************************************************
    *************************************************
    *******
120 *
121 * 通过can 接口接收一帧数据
122 *
123 * 其中 FIFONumber 参数未使用
124 *************************************************
    *************************************************
    *******/
125 int CAN_Receive(HHD32F1_CAN_TypeDef *can, uint8_t
    FIFONumber, CanRxMsg* RxMessage);
126
127 ////////////////////////////////////////////SPI
    ////////////////////////////////////////////////
    ///////
128
129 /*************************************************
    *************************************************
    *******
130 *
131 *   初始化 SPI接口
132 *
```

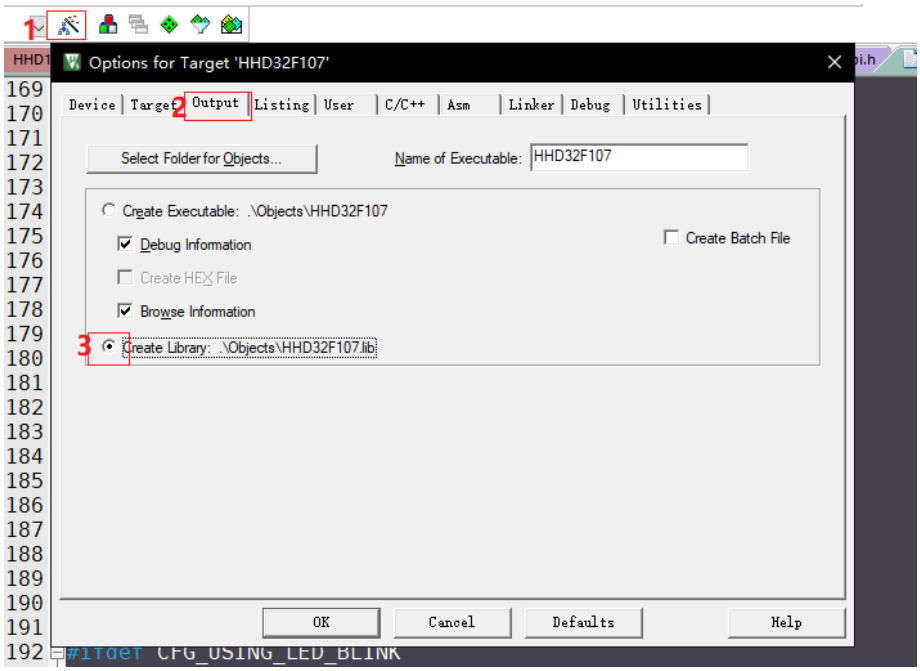
```
133 ************************************************
    ************************************************
    *******/
134 void spi_Init(void);
135
136
137 /************************************************
    ************************************************
    *******
138 *
139 *    写数据到FPGA
140 *
141 ************************************************
    ************************************************
    *******/
142 int SPI_To_FPGA_Wirte(uint8_t fpga, uint8_t addr,
    uint8_t *data, int len);
143
144 /************************************************
    ************************************************
    *******
145 *
146 *    连续从FPGA读回数据
147 *
148 ************************************************
    ************************************************
    *******/
149 int SPI_To_FPGA_Read(uint8_t fpga, uint8_t addr,
    uint8_t *data, int len);
150
151 #endif
```

编译

编译成库

1、在HHD1705_lib.h中，确保 `COMPILE_TO_LIB` 被定义

2、如下图操作，选择编译成库



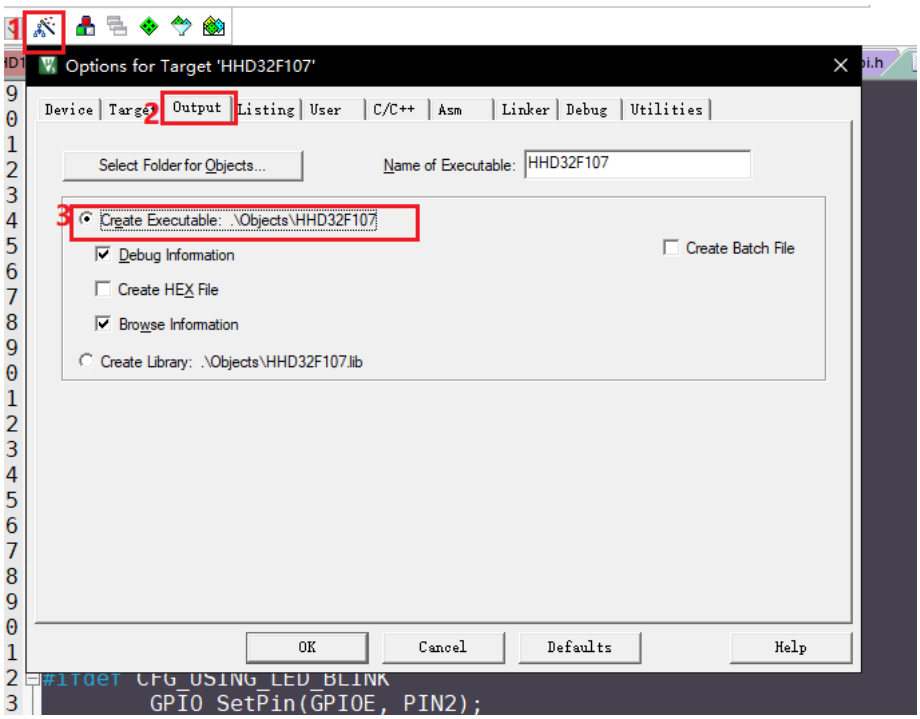
3、使整个工程全部编译

4、生成的文件在`\Project\Objects`,文件名为：**HHD32F107.lib**

编译成直接烧写文件

1、在HHD1705_lib.h中，确保 `COMPILE_TO_LIB` **没有**被定义

2、如下图操作，选择编译成直接烧写文件：



调试网口与背板交换机的切换

`HHD1705_Full_Lib`工程支持两种网络路径：

- 调试网口
- 背板交换机

两种历经通过一个函数参数来控制，函数定义在如下位置：

文件路径： HHD1705_Full_Lib\User\main.c

在函数`void peripheral_init(void)`中

```
94  /*******************************************************************************
95  void peripheral_init(void)
96  {
97      SysTick_Config(SystemCoreClock /100);  /* SystTick configuration: an inter
98      NVIC_SetPriority (SysTick_IRQn, 1);    /* Update the SysTick IRQ priority
99
100     NVIC_EnableIRQ(ETH_IRQn);
101
102     spi_Init();
103 //  can_init(CAN1, CAN_BAUD_500K, 0x7FF, 0xFF);
104
105 #ifdef  CFG_USING_NET
106 //  Ethernet_Configuration(EN_TO_MDI);        //走调试网口
107     Ethernet_Configuration(EN_TO_SWITCH);     //走背板交换机
108 #endif
109 }
110 /*******************************************************************************
```

函数

```
1  Ethernet_Configuration(EN_TO_SWITCH);
```

当参数为 **EN_TO_MDI** 时 使用调试网口

当参数为 **EN_TO_SWITCH** 时 使用背板交换机