**MySTC\_B BSP\_Ver3.0**

**“STC\_B学习板”软件支持包**

**使用说明**

湖南大学信息科学与工程学院

徐成

2021年6月9日更新

目录

1. 资源管理与任务调度MySTC\_OS
2. 系统装载与初始化MySTC\_Init
3. 系统函数库Sys.Lib
4. 显示模块函数库Displayer.lib
5. 按键模块函数库Key.lib
6. 无源蜂鸣器函数库Beep.lib
7. 音乐模块函数库Music.lib
8. 霍尔传感器模块函数库Hall.lib
9. 振动传感器模块函数库Vib.lib
10. 模数转换模块函数库ADC.lib
11. 异步通信模块函数库Uart1.lib

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Ver3.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(1) 系统工作时钟频率可以在main.c中修改 SysClock赋值（单位Hz）。

如：code long SysClock=11059200; 定义系统工作时钟频率为11059200Hz（也即11.0592MHz）

系统工作频率必须与实际工作频率（下载时选择的）一致，以免与定时相关的所有功能出现误差或错误。

(2) 系统定时器类资源规划：

定时器0：系统1mS定时

定时器1：串口1（Uart1）通信用

定时器2： 保留或串口2通信

CCP0： 保留或红外通信38KHz产生用

CCP1： 蜂鸣器发声控制

CCP2： 保留（IAP15W4K61S4无CCP2中断）

(3) 使用方法：

1，在工程中加载main.c文件和STC\_BSP.lib库文件

2，在main.c中选择包含以下头文件（如果要使用可选模块提供的函数和方法，就必须包含其头文件） ：

#include "STC15F2K60S2.H" //必须。"STC-B学习板"使用MCU指定的头文件

#include "sys.H" //必须。sys（MySTC\_OS调度程序）头文件

#include "display.H" //可选，display（显示模块）头文件。

#include "key.H" //可选，key（按键模块）头文件。

#include "hall.H" //可选，hall（霍尔传感器模块）头文件。

#include "Vib.h" //可选，Vib（振动传感器模块）头文件。

#include "beep.H" //可选，beep（蜂鸣器模块）头文件。

#include "music.h" //可选，music（音乐播放）头文件。

#include "adc.h" //可选，adc（热敏、光敏、导航按键、

扩展接口ADC功能）头文件。

3，MySTC\_Init()是sys初始化函数，必须执行一次；

MySTC\_OS()是sys调度函数，应置于while（1）循环中；

4，各可选模块如果选用，必须在使用模块其它函数和方法前执行一次模块所提供的驱动函数（设置相关硬件、并在sys中加载其功能调度）：

DisplayerInit(); //可选，显示模块驱动

Key\_Init(); //可选，按键模块驱动

HallInit(); //可选，霍尔传感器模块驱动

VibInit(); //可选，振动传感器模块驱动

BeepInit(); //可选，蜂鸣器模块驱动

MusicPlayerInit(); //可选，音乐播放模块驱动

AdcInit(); //可选，ADC模块驱动（包含：热敏电阻、光敏电阻、导航按键、扩展接口ADC）

5，sys和各模块共提供以下异步事件：

enumEventSys10mS: 10mS事件 （"10毫秒时间间隔到"事件）

enumEventSys100mS: 100mS事件（"100毫秒时间间隔到"事件）

enumEventSys1S: 1S事件 （"1秒时间间隔到"事件）

enumEventKey: 按键事件 K1、K2、K3 三个按键有"按下"或"抬起"操作）

enumEventHall: 霍尔传感器事件（霍尔传感器有"磁场接近"或"磁场离开"事件）

enumEventVib: 振动传感器事件（振动感器检测到"振动"事件）

enumEventNav: 导航按键事件 （导航按键5个方向、或按键K3 有"按下"或"抬起"操作）

enumEventXADC: 扩展接口上完成一次ADC转换事件（P1.0、P1.1采取到一组新数据）

enumEventRxd: Uart1收到了一个符合指定要求（数据包头匹配、数据包大小一致）的数据包

对这些异步事件，应采用"回调函数"方法响应（即用sys提供的SetEventCallBack()设置用户回调函数），以提高系统性能。

6，各可选模块提供的其它函数和具体使用方法请参见：

各模块头文件中的说明；

main.c提供的推荐程序框架和部分示例；

其它可能技术文档或应用示例（陆续提供）

编写：徐成（电话18008400450） 2021年2月26日设计，2021年3月29日更新

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* sys Ver3.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(1) sys.c构成STC程序基本架构。提供：

a, 系统初始化MySTC\_Init(),系统调度函数MySTC\_OS(),加载用户回调函数SetEventCallBack()

b, 若干可设置和触发回调函数的事件；

enumEventSys10mS : 10mS定时到

enumEventSys100mS : 100mS定时到

enumEventSys1S : 1S定时到

enumEventKey : 按键事件

enumEventNav : 导航按键事件“：

enumEventHall : hall传感器事件

enumEventVib : 振动传感器事件

enumEventXADC : 扩展接口上新的AD值事件“

enumEventRxd : 串口上收到一个符合格式定义的事件

c, 系统运行性能评价参数：struct SysPerF SysPerformance

(2) MySTC\_Init()：sys初始化函数，必须执行一次。

(3) MySTC\_OS()：sys调度函数，应在while（1）循环中。

(4) SetEventCallBack(char event, void \*(user\_callback)):加载"事件"用户回调函数.事件有:

(5) 系统运行性能评估参数struct SysPerF SysPerformance：

a, SysPerformance.LoopsPS unsigned long 每秒主循环次数(正常应大于1000以上，越大越好)

b, SysPerformance.SysClickSkipedPS unsigned char 每秒sys基本调度时间片（1mS）被跳过次数(正常应为0)（超过255只记到255）

c, SysPerformance.SysAveClksPC unsigned char 每循环sys程序平均消耗系统时钟数

d，SysPerformance.UserAveClksPC unsigned int每循环用户程序平均消耗系统时钟数

e, SysPerformance.SysSourceRemained unsigned char系统（计算能力）资源剩余评估（%）

c, SysPerformance.Summary unsigned char 系统性能总体评分（100分标准）

80-100分：优秀，系统不会发生任务调度错误；

60-80分 ：中等，系统可基本正常； <60分 ：差， 系统存在很大出错或奔溃风险；

<30分 ：很差，系统会出现明显错误，或崩溃

(6) 补充说明：

a, sys.c 占用定时器0给系统产生1mS定时中断。

b, 系统时钟在12Mhz时，MySTC\_OS()在每循环大约消耗115个系统时钟数（约9.58uS）（2021年3月29测试）。在不同系统时钟频率响会有少量出别

b, 1mS为本架构基本要求(受STC单片机资源和功能限制的原因造成),因此要求用户的任何程序片段，其单次循环执行时间累加起来应小于1mS。

编写：徐成（电话18008400450） 2021年2月26日设计，2021年3月29日更新

\*/

#ifndef \_sys\_h\_

#define \_sys\_h\_

extern void MySTC\_Init();

extern void MySTC\_OS();

extern void SetEventCallBack(char event, void \*(user\_callback));

enum event{enumEventSys10mS,enumEventSys100mS,enumEventSys1S,enumEventKey,

enumEventHall,enumEventVib,enumEventNav,enumEventXADC,enumEventRxd};

typedef struct //用于各模块加载驱动

{ void (\*DrvDisplayer)() reentrant;

void (\*DrvKey)() reentrant;

char (\*GetKey)() reentrant;

void (\*DrvHall)() reentrant;

char (\*GetHall)() reentrant;

void (\*DrvVib)() reentrant;

char (\*GetVib)() reentrant;

void (\*DrvBeep)() reentrant;

void (\*DrvMusic)() reentrant;

void (\*DrvADC)() reentrant;

char (\*GetNav)() reentrant;

char (\*GetXADC)() reentrant;

char (\*GetUart1Rxd)() reentrant;

} SysPt;

extern xdata SysPt stcDrivers;

typedef struct //系统性能评估参数,每秒更新一次

{ unsigned long LoopsPS;

unsigned char SysClickSkipedPS;

unsigned char SysAveClksPC;

unsigned int UserAveClksPC;

unsigned char SysSourceRemained;

char Summary;

} SysPerF;

extern xdata SysPerF SysPerformance;

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* display V2.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

displayer用于控制“STC-B学习板”上8个7段数码管（Seg7）和8个指示灯（Led）工作.提供显示模块加载和三个应用函数：

(1) displayerInit()：显示模块加载函数；

(2) SetDisplayerArea(char Begin\_of\_scan,char Ending\_of\_Scan)：设置LED启用区域。8个数码管从左至右编号分别为0——7，

函数参数：

Begin\_of\_scan设定启用数码管起始编号，Ending\_of\_Scan为结束编号。设定范围内的数码管才工作和显示。

注：正常情况下，两个参数取值范围为0——7，且Ending\_of\_Scan>Begin\_of\_scan。但利用动态扫描和人眼视觉效果，可设置超出

该范围的参数，以实现特殊显示效果：如软件调整显示亮度，或非灯亮度显示，等；

(3) Seg7Print(char d0,char d1,char d2,char d3,char d4,char d5,char d6,char d7)：将8个参数值分别译码显示到对应的数码管上。

显示译码表(code char decode\_table[])在main.c中,用户可以修改和增减.

(4) LedPrint(char led\_val)：控制8个指示灯开关。参数light\_val的8个bit位对应8个指示灯的开关，“1”——指示灯“亮”

编写：徐成（电话18008400450） 2021年2月26日设计，2021年3月15日更新

\*/

#ifndef \_displayer\_H\_

#define \_displayer\_H\_

extern void DisplayerInit();

extern void SetDisplayerArea(char Begin\_of\_scan,char Ending\_of\_Scan);

extern void Seg7Print(char d0,char d1,char d2,char d3,char d4,char d5,char d6,char d7);

extern void LedPrint(char led\_val);

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beep V2.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Beep用于控制“STC-B学习板”上无源蜂鸣器的发声。Beep模块共提供1个驱动函数、2个应用函数：

(1) BeepInit()：蜂鸣器模块驱动函数

(2) Set\_Beep(unsigned int Beep\_freq, unsigned char Beep\_time)：控制蜂鸣器发声；

函数参数：

Beep\_freq：指定发声频率，单位Hz。小于<10 Hz, 不发音

Beep\_time：指定发声时长。发声时长＝10\*Beep\_time (mS) ，最长 655350mS

函数返回值：

enumSetBeepOK：调用成功，

enumSetBeepFail：调用失败（或因蜂鸣器正在发音）

(3) GetBeepStatus(void): 获取Beep当前状态

函数返回值：

enmuBeepFree:空闲

enumBeepBusy ,正在发音

(4) Beep模块使用了STC内部CCP模块1通道

编写：徐成（电话18008400450） 2021年3月3日设计，2021年3月26日更新

\*/

#ifndef \_beep\_H\_

#define \_beep\_H\_

extern void BeepInit();

extern char SetBeep(unsigned int Beep\_freq, unsigned int Beep\_time);

extern unsigned char GetBeepStatus(void);

enum BeepActName {enumBeepFree=0,enumBeepBusy,enumSetBeepOK,enumSetBeepFail};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* key V2.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Key模块用于获取“STC-B学习板”上三个按键的状态。提供按键模块加载和一个应用函数,一个“按键事件：enumEventKey：

(1) Key\_Init()：按键模块加载函数；

(2) char GetKeyAct(char Key)：获取按键状态。

函数参数：Key，指定要获取状态的按键。Key取值：

enumKey1

enumKey2

enumKey3

（当参数取值超出有效范围，函数将返回fail）

函数返回值：

enumKeyNull（无按键动作）

enumKeyPress（按下）

enumKeyRelease（抬起）

enumKeyFail（失败）

返回值是经过多次检测按键实时状态和统计检测结果后（软件消抖）的有效事件。

每个按键查询一次后,事件值变成enumKeyNull。事件值仅查询一次有效。

(3) 按键事件：enumEventKey

当三个按键（enumKey1,enumKey2,enumKey3）中任意一个按键有”按下“或”抬起“动作时，将产生一个”按键事件“，响应按键事件的用户处理函数由用户编写,并有sys中提供的SetEventCallBack函数设置.

编写：徐成（电话18008400450） 2021年3月5日设计，2021年3月26日更新

\*/

#ifndef \_key\_H\_

#define \_key\_H\_

extern void Key\_Init();

extern unsigned char GetKeyAct(char Key) ;

enum KeyName {enumKey1,enumKey2,enumKey3};

enum KeyActName {enumKeyNull,enumKeyPress,enumKeyRelease,enumKeyFail};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* hall V2.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hall模块用于获取“STC-B学习板”上hall传感器状态。hall模块共提供1个加载函数、1个应用函数，一个Hall事件：enumEventHall

（1）HallInit()：hall模块初始化函数

（2）unsigned char GetHallAct(void)：获取hall事件。

函数返回值：

enumHallNull（无变化）

enumHallGetClose（磁场接近）

enumHallGetAway（磁场离开）

查询一次后,事件值变成enumEventHall (仅查询一次有效)

（3）hall传感器事件：

当Hall检测到有"磁场接近"或"磁场离开"事件时，将产生一个Hall传感器事件(enumEventHall).响应事件的用户处理函数由用户编写, 并有sys中提供的SetEventCallBack()函数设置事件响应函数.

编写：徐成（电话18008400450） 2021年3月15日设计 2021年3月26日修改

\*/

#ifndef \_hall\_H\_

#define \_hall\_H\_

extern void HallInit(void);

extern unsigned char GetHallAct(void);

enum HallActName {enumHallNull,enumHallGetClose,enumHallGetAway};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 振动传感器SV V2.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SV模块用于获取"STC-B学习板"上Vib传感器状态.提供一个模块加载函数和一个应用函数,一个Vib事件enumEventVib：

(1) VibInit()：振动传感器Vib模块初始化函数；

(2) char GetVibAct()：获取Vib事件。

函数返回值：

enumVibNull——无，

enumVibQuake——发生过振动

查询一次后,事件值变成enumVibNull，(仅查询一次有效)

(3) Vib传感器事件enumEventVib：

当Vib检测到有”振动“事件时，将产生一个”振动事件“，响应事件的用户处理函数由用户编写,并有sys中提供的SetEventCallBack()函数设置振动事件用户处理函数.

编写：徐成（电话18008400450） 2021年3月5日设计，2021年3月26日更新

\*/

#ifndef \_Vib\_H\_

#define \_Vib\_H\_

extern void VibInit();

extern unsigned char GetVibAct(void) reentrant;

enum VibActName {enumVibNull,enumVibQuake};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* music V2.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Music模块在Beep和Displayer模块基础上再次封装，用于控制“STC-B学习板”上播放音乐。加载该模块，将同时加载Beep模块、displayer模块。music模块共提供1个music驱动函数、4个应用函数：

（1）MusicPlayerInit()：驱动music模块；

（2）char PlayTone(unsigned char tone, unsigned char beatsPM ,unsigned char scale, unsigned char beats)：播放音乐音阶，实现用指定音调、指定节拍率、发指定音阶、发音节拍数。

函数参数：

tone：指定音调，有效值：0xFA、0xFB、0xFC、0xFD、0xFE、0xFF、0xF9分别对应音乐A、B、C、D、E、F、G调

beatsPM：节拍率，即每分钟节拍数，值范围10~255拍/分钟

scale：音乐简谱音高，1字节。

0x00——休止符

高4位：1——低8度音 2——中8度音 3——高8度音

低3位：1-7对应简谱音。其它值无效。

如：0x13表示低音3（mi）

beats：音长(节拍数),单位1/16拍。

如：16（0x10）对应1拍，32（0x20）对应2拍，8（0x08）对应半拍...... 。

函数返回值：enumBeepOK：调用成功

enumBeepBusy：忙（上一音未按设定发完，或因蜂鸣器正在发音）

enumBeepFail：调用失败（音调参数tone不对，或音高编码scale不对）

（见Beep.h中定义BeepActName）

（3）SetMusic(unsigned char beatsPM, unsigned char tone, unsigned char \*pt, unsigned int datasize, unsigned char display)：设定或改变要播放音乐和播放参数。

函数参数：

beatsPM：节拍率，即每分钟节拍数，值范围10~255拍/分钟，如果参数值为“invalid”将不改变原beatsPM；

tone：指定音调， 有效值：0xFA、0xFB、0xFC、0xFD、0xFE、0xFF、0xF9分别对应音乐A、B、C、D、E、F、G调，参数值为“invalid”或其它值将不改变原tone

\*pt ：指向要播放的音乐编码的首地址

datasize：要播放的音乐编码的长度（字节数）

display：Seg7和Led是否用来显示播放音乐信息，取值：

enumMscNull —— 不用

enumMscDrvSeg7 —— 用7段数码管Seg7（显示信息）

enumMscDrvLed —— 用Led指示灯（打拍） enumMscDrvSeg7andLed —— 用Seg7 和 Led

参数值为“invalid”或其它值将不改变原display

音乐编码规则：

1，常规音乐简谱发音编码（成对出现，不可分开，中间不能插入其它编码和控制字）

基本格式：音高（1字节），节拍数（1字节），音高，节拍，......

其中“音高”部分：

0x11 — 0x17 ：对应低音 do、re、mi、fa、so、la、si、 0x21 — 0x27 ：对应中音 do、re、mi、fa、so、la、si

0x31 — 0x37 ：对应高音 do、re、mi、fa、so、la、si

其中“节拍数”部分：

0x01-0xFF：单位1/16拍。也即十六进制中，高4位表示整拍数，低4位表示分拍数（1/16）

如：发音2拍： 0x20

发音半拍： 0x08

发音1拍半：0x18

2，音乐编码中可以插入以下控制字，用于设定音乐播放参数等（前6个也可以通过函数，用程序设定和实现）:

enumMscNull ： 不用

enumMscDrvSeg7 ： 用7段数码管Seg7（显示信息）

enumMscDrvLed ： 用Led指示灯（打拍）

enumMscDrvSeg7andLed ： 用Seg7 和 Led

enumMscSetBeatsPM ： 设置节拍率， 后面再跟 节拍率（1字节）

enumMscSetTone ： 设置音调， 后面再跟 音调（1字节）

0xFA 或 0xFB 或 0xFC 或 0xFD 或 0xFE 或 0xFF 或 0xF9

分别对应音乐：A调 或B调………

enumMscRepeatBegin ： 设置音乐播放重复开始处。重复一次（暂不支持多次），暂不能嵌套（嵌套无效或可能导致不可预期结果）

enumMscRepeatEnd ： 设置音乐播放重复结束处

（4）SetPlayerMode(unsigned char play\_ctrl) ：音乐播放控制函数。

函数参数：

play\_ctrl：enumModePlay ： ”播放“

enumModePause ： ”暂停“

enumModeStop ：”停止/结束“

（其它参数无效）

所有操作在当前”音“播放完成后生效；

（5）char GetPlayerMode(void)： 获取当前播放状态

函数返回值：（play\_ctrl值）

enumModePlay ： 播放状态

enumModePause ： 暂停状态

enumModeStop ： 停止/结束

（其它功能应用型函数可根据需要设置和增加）

编写：徐成（电话18008400450） 2021年3月5日设计，2021年3月26日更新

\*/

#ifndef \_music\_H\_

#define \_music\_H\_

extern void MusicPlayerInit();

extern char PlayTone(unsigned char tone, unsigned char beatsPM ,

unsigned char scale, unsigned char beats);

extern void SetPlayerMode(unsigned char play\_ctrl);

extern char GetPlayerMode(void); //获取当前播放状态

enum PlayerMode {enumModeInvalid=0,enumModePlay,enumModePause,

enumModeStop};

enum MusicKeyword {enumMscNull=0xF0,enumMscDrvSeg7,enumMscDrvLed,

enumMscDrvSeg7andLed,enumMscSetBeatsPM,enumMscSetTone,

enumMscRepeatBegin,enumMscRepeatEnd};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* ADC V1.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ADC模块用于“STC-B学习板”上与ADC相关电路:温度Rt、光照Rop、导航按键Nav、扩展接口EXT上的ADC转换。提供ADC模块初始化函数、2个应用函数,2个事件：

(1) AdcInit(char ADCsel)：ADC模块初始化函数；

函数参数: ADCsel选择扩展接口EXT是否用作ADC功能，取值：

enumAdcincEXT : 含对扩展接口EXT设置ADC功能（EXT上P1.0、P1.1不可作数字IO功能使用）

enumAdcexpEXT : 不含对扩展接口EXT设置ADC功能（EXT上P1.0、P1.1可作数字IO功能使用）

(2) GetAdcResult(char ADC\_name)：获取指定通道ADC值。

函数参数：

ADC\_name：指定要获取ADC值的对象。取值：

enumAdcXP10 —— 扩展接口上的 P1.0脚

enumAdcXP11 —— 扩展接口上的 P1.1脚

enumAdcRt —— 热敏电阻

enumAdcRop —— 光敏电阻

enumAdcNav —— 导航按键

补充说明1：对于导航按键，下面GetAdcNavAct函数输出消抖后导航按键事件和状态，更方便使用；

补充说明2：当参数取值不在有效范围，函数将返回enumKeyFail

函数返回值：返回所查对象10bit AD值(电压）。

每个数字表示 VCC/1024（单位：V），其中VCC为电路板供电电压（USB接口一般为5V左右）

(3) char GetAdcNavAct(char Nav\_button)：获取导航按键（包含K3）状态

函数参数：Nav\_button：指定要获取状态的导航按键。取值：

enumAdcNavKey3（K3键）,

enumAdcNavKeyRight（右按）,

enumAdcNavKeyDown（下按）,

enumAdcNavKeyCenter（中心按）

enumAdcNavKeyLeft（左按）,

enumAdcNavKeyUp（上按）.

（当参数取值不在有效范围，函数将返回enumKeyFail）

函数返回值：返回当前按键事件，返回值：（同Key模块GetAdcKeyAct（）返回值）

enumKeyNull（无按键动作）,

enumKeyPress（按下）,

enumKeyRelease（抬起）,

enumKeyFail（失败）

返回值是经过多次检测按键实时状态和统计检测结果后（软件消抖）的有效事件。

每个按键查询一次后,事件值变成 enumKeyNull (仅查询一次有效)

(4) 导航按键事件“：enumEventNav

当导航按键5个方向或按键K3有任意”按下“或”抬起“动作时，将产生一个”导航按键事件“enumEventNav。响应导航按键事件的用户回调函数由用户编写,并由sys提供的SetEventCallBack()函数设置响应函数.

(5) 扩展接口EXT上P1.0、P1.1两个端口有新的AD值事件“：enumEventXADC

当ADC模块对P1.0、P1.1进行ADC转换，获得了它们新的ADC结果时，将产生enumEventXADC事件，通知用户进行处理。响应enumEventXADC事件的用户回调函数由用户编写,并有sys提供的SetEventCallBack()函数设置响应函数.

ADC模块对P1.0、P1.1进行ADC转换速度为3mS，也即每3mS或每秒钟333次转换.

(6) 补充说明：

a：对EXT上P1.0、P1.0的转换速度为3mS，也即每秒钟提供333次转换结果，提供了有新转换结果事件：enumEventXADC，方便用户处理

b：对于Rt、Rop，转换速度为9mS，也即每秒钟提供111次转换结果。没有提供相应“事件”，用户随时用函数GetAdcResult()查询和使用

c：对导航按键进行了软件消抖处理，最快可支持导航按键每秒12次操作速度，提供了导航按键发生了操作事件：enumEventNav

d：由于导航按键与K3键共用了单片机同一个端口（P1.7），启用ADC模块后，P1.7口IO功能失效，只能用GetAdcNavAct(char Nav\_button)函数获取K3按键的事件或状态。

e: STC-B板上Rt型号为：10K/3950 NTC热敏电阻，光敏电阻Rop型号为：GL5516. 它们的ADC值与温度、光照强度关系请参阅它们的数据手册与STC-B电路图进行换算。

编写：徐成（电话18008400450） 2021年3月25日设计

\*/

#ifndef \_adc\_H\_

#define \_adc\_H\_

extern void AdcInit(char ADCsel);

enum ADC\_model {enumAdcincEXT=0x9B,enumAdcexpEXT=0x98};

extern unsigned int GetAdcResult(char ADC\_name);

enum ADC\_SourceName {enumAdcXP10=0,enumAdcXP11,enumAdcRt,

enumAdcRop,enumAdcNav};

extern unsigned char GetAdcNavAct(char Nav\_button);

enum KN\_name {enumAdcNavKey3=0,enumAdcNavKeyRight,enumAdcNavKeyDown,

enumAdcNavKeyCenter,enumAdcNavKeyLeft,enumAdcNavKeyUp};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Uart1串行通信模块 V1.0 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Uart1模块提供Uart1模块初始化函数、3个应用函数,1个事件（enumEventRxd）：

1. Uart1Init(unsigned int band)：Uart1模块初始化函数。

函数参数：unsigned int band定义通信（8个数据位、1个停止位，无奇偶校验位）的波特率（单位：bps）

1. void SetUart1Rxd(char \*RxdPt, unsigned int Nmax, char \*matchhead, unsigned int matchheadsize);

设置串口接收参数：数据包存放位置、大小，包头匹配字符、匹配字符个数。收到符合条件的数据包时将产生enumEventRxd事件。

函数参数： char \*RxdPt：指定接收数据包存放区（首地址）

unsigned int Nmax：接收数据包大小（字节数），最大65535

char \*matchhead： 需要匹配的数据包头（首地址）

unsigned int matchheadsize：需要匹配的字节数

补充说明：

Nmax=1：为单字节接收，即收到一个字节就产生enumEventRxd事件（如果定义了匹配，还需满足匹配条件）；

0 < matchheadsize < Nmax：要求接收数据中连续matchheadsize个字节与\*matchhead处数据完全匹配，才在收到Nmax数据时产生enumEventRxd事件；

matchheadsize = Nmax：设定接收数据包完全匹配

matchheadsize=0 或 matchheadsize > Nmax：将不做匹配，接收到任意Nmax数据时产生enumEventRxd事件；

在enumEventRxd事件发出后，接收到的数据包应及时使用或取出，收到下一个数据时将破坏和覆盖前面收到的数据包

(3) char Uart1Print(void \*pt, unsigned int num)：发送数据包，非阻塞函数（即函数不等到所设定任务全部完成才返回），该函数从被调用到返回大约1uS左右时间。

函数参数： void \*pt ：指定发送数据包位置

unsigned int num：发送数据包大小；

函数返回值：enumTxOK：调用成功，即所设定的发送数据包请求已被系统sys正确接受，sys将尽硬件资源最大可能及时发送数据。

enumTxFailure：调用失败（主要原因是：串口正忙（上一数据包未发完）

补充说明：串口上发送1个字节数据大约需要时间0.1mS～10mS（视所设定的波特率），对计算机来说，如果发送多个字节是一个很要时间才能完成的事。类似于 前面用蜂鸣器演奏音乐，对这类事件与程序"异步"的问题，编程时不仅要注意程序逻辑性、还有注意程序时效性。（这个问题其实总是要注意、必须要注意的）

(4) char GetUart1TxStatus(void)： 获取Uart1发送状态

函数返回值：enumUart1TxFree:串口1发送空闲

enumUart1TxBusy,串口1发送正忙

(5) Uart1接收事件：enumEventRxd。表示收到了一个符合指定要求（数据包头匹配、数据包大小一致）的数据包。

编写：徐成（电话18008400450） 2021年3月28日设计

\*/

#ifndef \_uart\_H\_

#define \_uart\_H\_

extern void Uart1Init(unsigned int band);

extern void SetUart1Rxd(void \*RxdPt, unsigned int Nmax, void \*matchhead, unsigned int matchheadsize);

extern char Uart1Print(void \*pt, unsigned int num);

extern char GetUart1TxStatus(void);

enum Uart1ActName {enumUart1TxFree=0,enumUart1TxBusy,enumUart1TxOK,

enumUart1TxFailure};

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*StepMotor 说明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

StepMotor用于STC-B板控制步进电机。共提供1个驱动函数、3个应用函数：

(1) StepMotorInit()：步进电机模块驱动函数

(2) SetStepMotor(char StepMotor,unsigned char speed ,int steps ) 指定步进电机、按指定转动速度、转动指定步

函数参数：StepMotor 指定步进电机，取值（enum StepMotorName中定义）

enumStepMotor1：SM 接口上的步进电机

enumStepMotor2：此时，用L0～L3四个LED模拟一个4相步进电机

enumStepMotor3：此时，用L4～L7四个LED模拟一个4相步进电机

speed 步进电机转动速度（0～255），单位：步/S。 (实际每步时间=int(1000mS/speed) mS），与设置速度可能存在一定误差

steps 步进电机转动步数（-32768～32767），负值表示反转

函数返回：enumSetStepMotorOK：调用成功（enum StepMotorActName中定义）

enumSetStepMotorFail：调用失败（电机名不在指定范围，或speed=0,或调用时正在转动）

（3）EmStop(char StepMotor) 紧急停止指定步进电机转动

函数参数：StepMotor 指定步进电机。函数参数不对将返回0值。

函数返回：剩余未转完的步数

（4）GetStepMotorStatus(char StepMotor) 获取指定步进电机状态

函数参数：StepMotor 指定步进电机

函数返回：enumStepMotorFree:自由（enum StepMotorActName中定义）

enumStepMotorBusy,忙（正在转动）

enumSetStepMotorFail：调用失败（步进电机名不在指定范围）

编写：徐成（电话18008400450） 2021年4月16日设计，2021年4月18日更新