**RH850\_F1L集成手册**

**V0.1**

Tan Fumin

2017-05-10

变更历史：

|  |  |  |
| --- | --- | --- |
| **日期** | **版本号** | **变更内容** |
| 2017-5-10 | V0.1 | 建立初版 |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[1. F1L系列MCU概况 4](#_Toc483315705)

[2. 代码执行流程 14](#_Toc483315706)

[3. MCU型号配置和模块使能 14](#_Toc483315707)

[4. 复位相关的配置 22](#_Toc483315708)

[5. 中断模块配置 24](#_Toc483315709)

[6. 时钟模块配置 28](#_Toc483315710)

[7. 栈相关的接口 35](#_Toc483315711)

[8. 休眠唤醒配置 36](#_Toc483315712)

[9. Port配置 40](#_Toc483315713)

[10. OSTM配置 45](#_Toc483315714)

[11. PWGA配置 47](#_Toc483315715)

[12. TUAx配置 49](#_Toc483315716)

[13. EINT配置 57](#_Toc483315717)

[14. ADC配置 59](#_Toc483315718)

[15. RLIN3\_UART配置 63](#_Toc483315719)

[16. IIC配置 66](#_Toc483315720)

[17. CSIx配置 70](#_Toc483315721)

[18. DMA配置 76](#_Toc483315722)

[19. MCU内存规划文件 79](#_Toc483315723)

[20. 工程集成过程 81](#_Toc483315724)

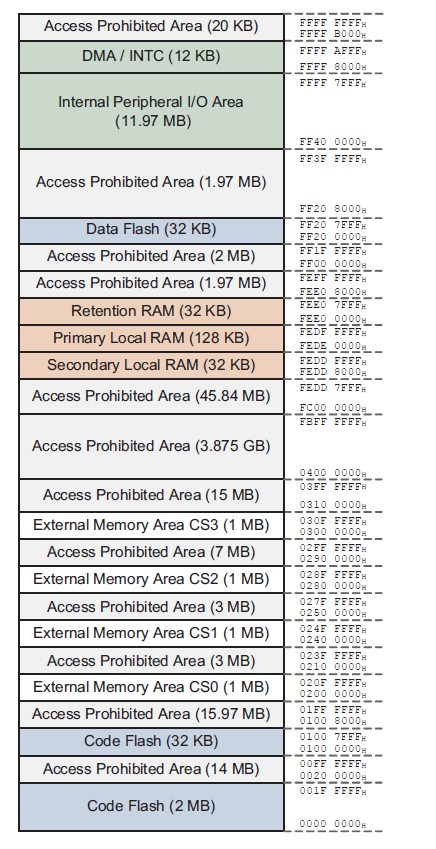
[21. 基本仿真操作 90](#_Toc483315725)

[22. 基本烧录操作 98](#_Toc483315726)

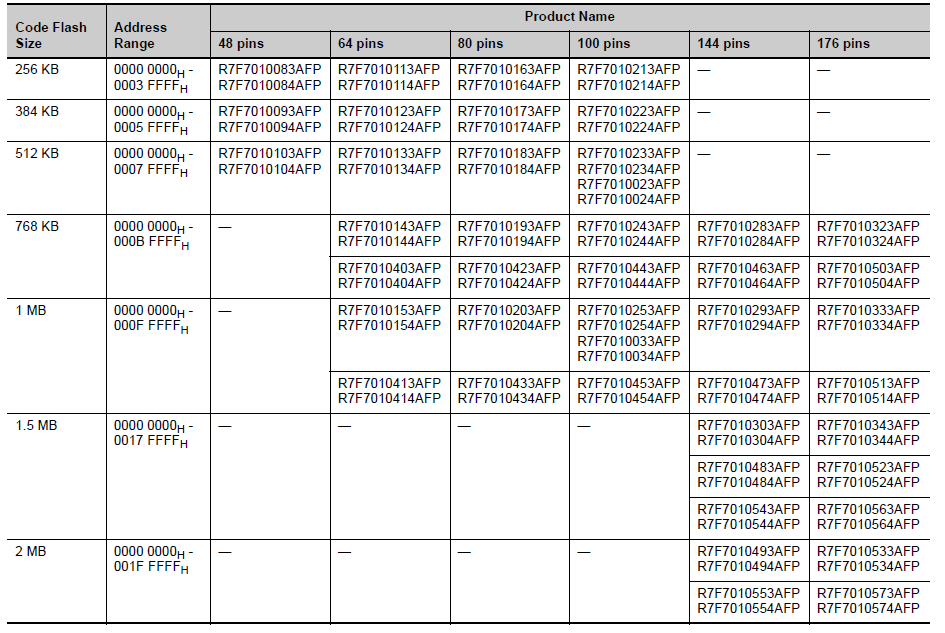
[23. 相关文档 102](#_Toc483315727)

# F1L系列MCU概况

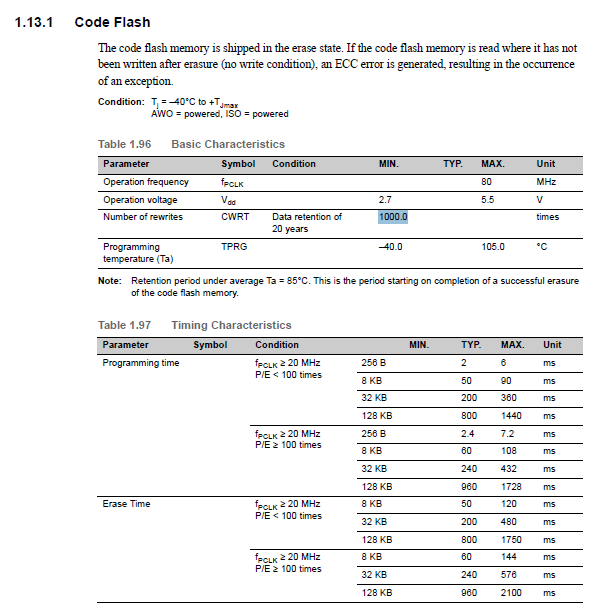
1. 内存分布



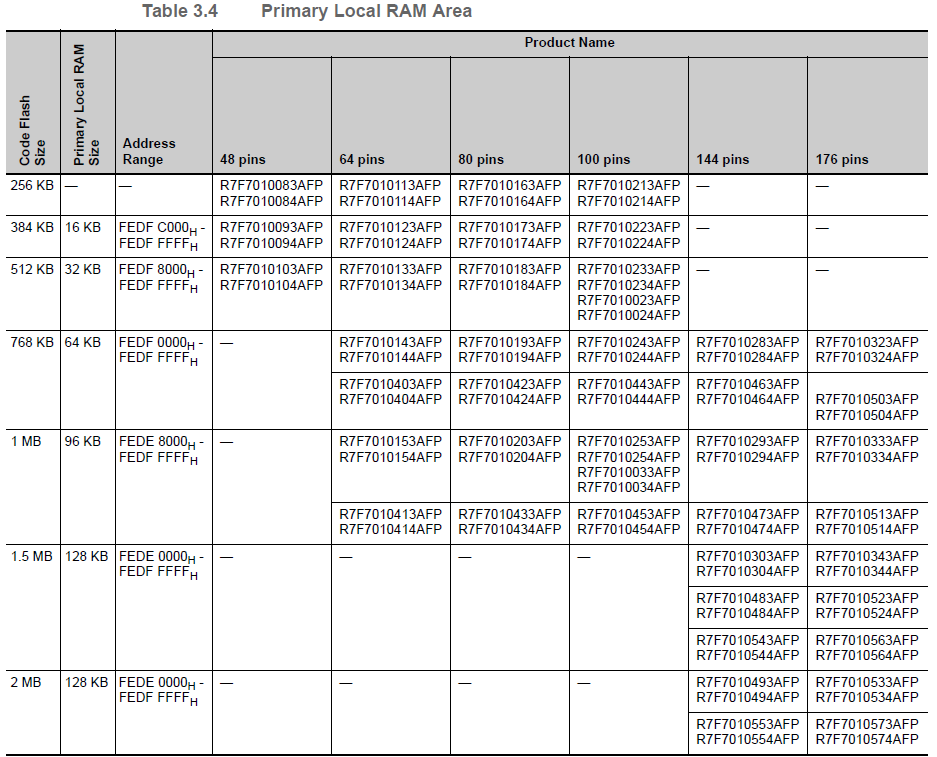
1. 器件CODE FLASH 大小分布

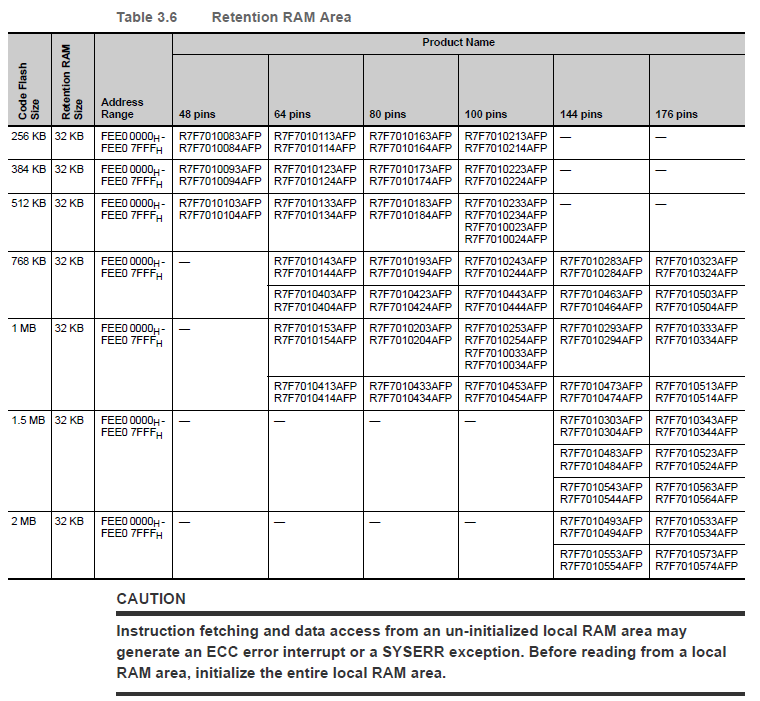


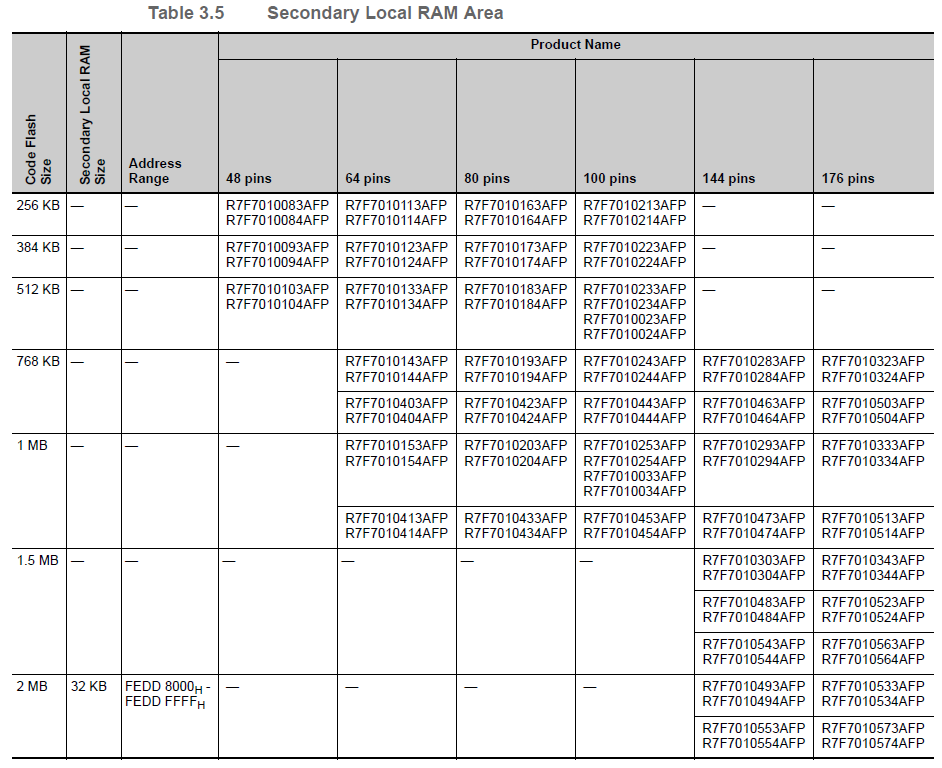
1. 器件CODE FLASH 读写性能



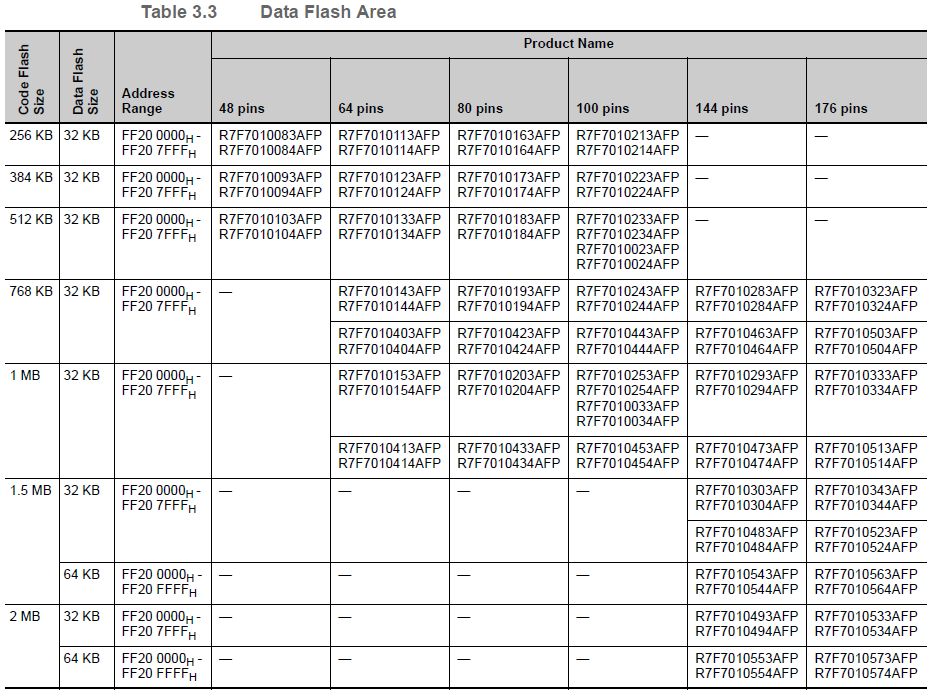
1. 器件RAM 大小分布



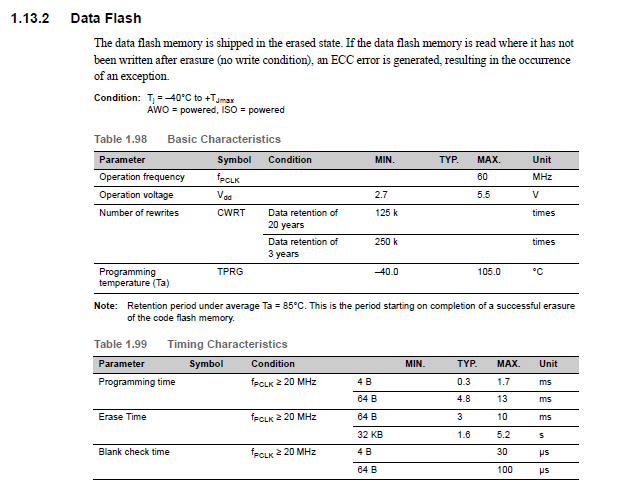




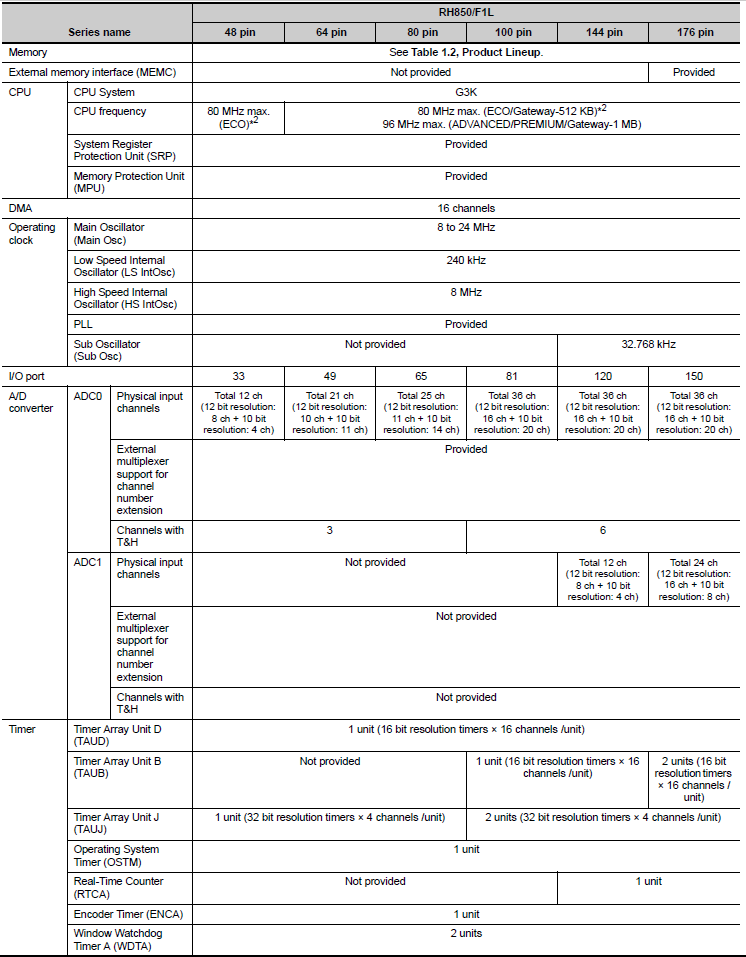
1. 器件DATA FLASH 大小分布

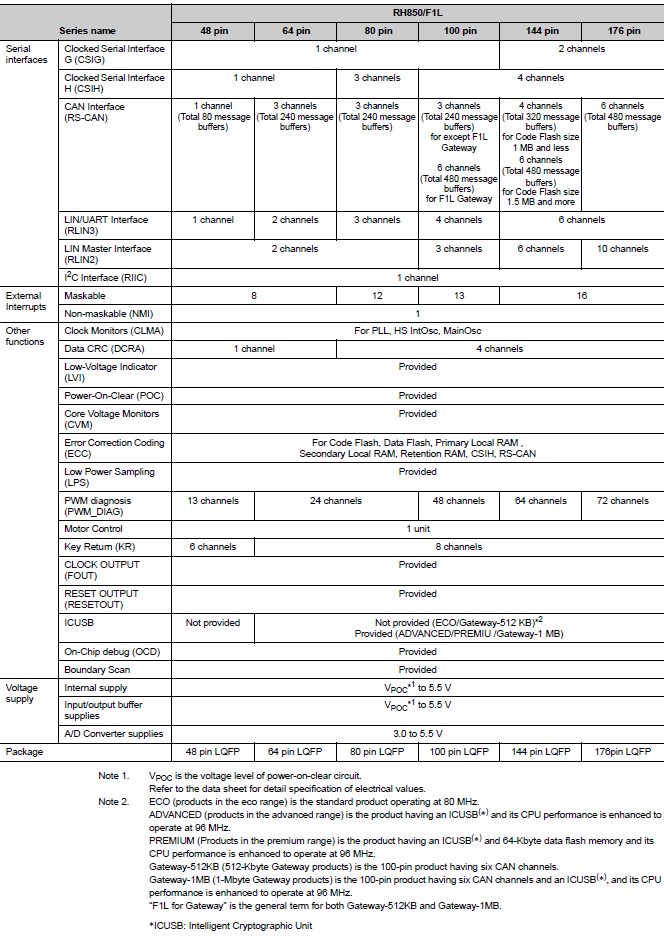


1. DATA FLASH 读写性能

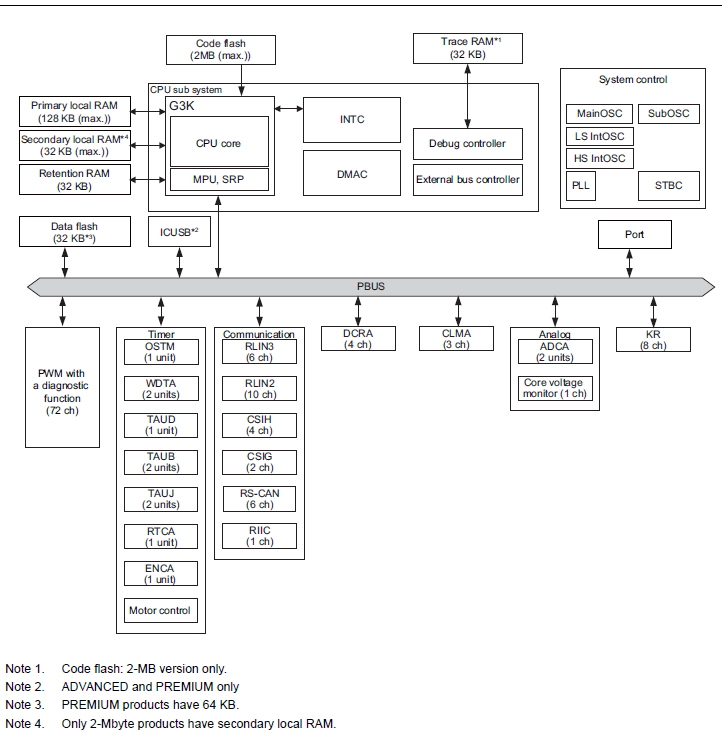


1. 内置资源性能概览





1. 176pin最大内部逻辑图



# 代码执行流程

函数的执行流程文件顺序大致如下：

\_F1L\_StartUp.c ===》 执行汇编代码，实现PC初始化，SP初始化，中断向量基址初始化

清除local RAM和Retention RAM，不清除Not\_Clear\_RAM的内容

执行\_\_\_low\_level\_init函数实现必要的硬件初始化

跳到库函数执行start，由start跳到C函数入口main函数

此文件同时实现中断向量表的按地址堆叠，C的中断函数入口由此进入

\_RH850.c ===》C语言main的函数实现，

根据用户的配置，实现各个MCU片内的硬件模块的初始化，包含clock，port，watchdog，PWGA，ADC，EINT等模块

调用用户端的外设初始化接口，并开启中断总开关

调用用户的主程序入口SYS\_vMainLoop,此时即跳转到了用户端的程序，剩下由用户控制CPU

用户端文件===》

\_F1L\_Vector.c===》C语言的中断函数的具体实现，此文件的中断函数被\_F1L\_StartUp.c引用并且按照向量表排

布，同时，他将调用不同具体模块的中断服务函数ISR，ISR再根据用户的具体配置调用用户的

回调函数

# MCU型号配置和模块使能

* 1. MCU型号配置及编译器选择

在MCU\_CFG.h文件内，选择适当的MCU型号可以剪切相关的驱动包文件进行编译

RH850\_F1L系列列举了以下芯片类型：

#define MCU\_ID\_7F1022 0x21u

#define MCU\_ID\_7F1023 0x22u

#define MCU\_ID\_7F1024 0x23u

#define MCU\_ID\_7F1025 0x24u

编译器类型列举了以下：

#define FUJITSU\_Softune 1

#define RENESAS\_IAR 2

#define RENESAS\_CSplus 3

#define RENESAS\_GreenHills 4

1. 选择芯片型号，如使用7F1022这个芯片，则应该定义如下

#define MCU\_UsingType MCU\_ID\_7F1022

1. 选择编译器类型，如选择greenhills编译器，则应该定义如下：

#define COMPILER\_TYPE (RENESAS\_GreenHills)

本包驱动代码支持greenhills和CSplus编译器

* 1. DET模块的配置

DET模块用于代码运行时的错误检测，DET.h文件的DET\_HaltWhenErrorHappen用于配置错误发生时的动作状态，

DET\_HaltWhenErrorHappen = 0时，将忽略错误，继续执行代码，但是可能发生其他不确定的错误状态

DET\_HaltWhenErrorHappen = 1时，将进行错误报告，且进入死循环，等待错误移除，

非常重要：在代码调试期间，此定义设为1可以快速知道代码错误的行与错误类型，右侧可以进行快速查错，但是在代码发布或者实际工程使用时，根据实际情况配置，如果设定为1，则MCU将进行看门狗复位，设定为0，则错误不处理，但是可能会发生其他错误。

* 1. 模块使能

当MCU\_UsingType的宏定义的芯片类型在F1L的列表范围内，F1L系列的文件见会被使能，此时可以在F1L\_UsrCfg.h文件中对各个资源进行是否使能的配置，

F1L\_UsrCfg.h文件配置列表

|  |  |
| --- | --- |
| 配置项 | 配置说明 |
| OPT byte\_0的配置 | MCU的默认状态设定配置：  OPT\_byte0 = 0x9AAFFFFF  注意：   1. 此OPT byte在程序烧录时需要烧录到MCU内部，不能通过软件的方法进行设定 2. 具体细节请参考硬件说明书章节35.11 Option Bytes |
| APP\_CODE\_MAP\_START | 此宏被F1L\_Startup.c文件引用，用于配置APP层代码空间的开始地址，以免在与BootLoader合并时被裁剪掉   1. 如果使用了BootLoader[链接文件为\_F1L\_Map\_WithBootLoader.ld]，则此宏必须定义为iROM\_Boot section的结束地址 2. 如果不使用BootLoader[链接文件为\_F1L\_Map\_NotBootLoader.ld]，则此宏必须定义0   默认值=0  注意：在使用BootLoader情况时，从0地址到此宏定义的值的ROM区间将被规划到BootLoader区，所以要保证APP的代码不能分配在此区间，以保证合并时不被剪切掉 |
| IRQ\_TABLE\_START | 此宏被F1L\_Startup.c文件引用，用于复位时对中断向量表的基址进行初始化，其中0x200地址前是MCU的系统级别中断，不能被占用，用户级别的中断向量表由此宏定义的地址开始，  默认值=0x200  注意：在BootLoader情况时，应该将此地址定义在非BootLoader区域，以保证合并时不被剪切掉 |
| MCU\_mINTNestingEnable | 中断嵌套使能，用于控制中断过程是否能被其他高优先级别的中断抢占  0：中断独占CPU，不能被中断，直到中断服务完成，  1：中断执行过程可以被其他高优先级的中断抢占  默认值=0，即中断不能被抢占  注意：本驱动包默认情况下，所有的中断都使用TABLE模式，即每个中断都有自己的独立入口，而没有采用多个优先级别的优先级模式 |
| \_RH850\_Debug | MCU底层调试开关，此开关只在调试底层时用到  0：关闭底层驱动调试  1：打开底层驱动调试  默认值=0，即关闭底层调试功能  注意：当打开时，   1. 将不会编译F1L\_USRCFG.c文件，而会编译F1L\_DEBUG.c文件 2. 将不会跑外设初始化函数及SYS\_vMainLoop函数，而会跑MCU\_vDebugMain函数 |
| MCU\_mINTPriorityChgEnable | 用于开启中断优先级调整接口的使能，  0：关闭此功能  1：开启此功能  默认值=0，即不使用优先级调整接口，因为中断不被抢占，所有中断请求在中断空闲时可以申请，优先顺序按照请求的MCU默认的优先级执行 |
| MCU\_mLVI\_Level | 配置MCU的低压重启功能的复位电压    0: LVI is ignored  1: 4.0+/-0.1 V (drop), 4.0+/-0.13 V (rise)  2: 3.7+/-0.1 V (drop), 3.7+/-0.13 V (rise)  3: 3.5+/-0.1 V (drop), 3.5+/-0.13 V (rise)  默认值=2，即VCC供电低于3.7V时进行MCU复位 |
| MCU\_mResetTime\_WD0  MCU\_mResetTime\_WD1 | 配置看门狗的复位时间，复位时间长度为：  2^(9 +设定值) \* LS(240 kHz)  默认值=6，即默认的设定复位时间为2^(9+6)/24000 = 136ms，实际复位时间受到内部低速晶振的精度影响  注意：默认情况下通过OPT byte设定看门狗的运行时刻为看门狗初始化函数第一次进行喂狗以后。 |
| MCU\_mClockMonitorEnable | 时钟偏差监控功能模块使能，用于开启MCU内部的时钟监控，如果时钟监控值超过偏差的设定百分比，则产生异常，此功能没有完善，请不要开启此功能  默认值=0，即不开启此功能 |
| MCU\_mCPUModeCtrlEnable | CPU运行模式控制，用于开启CPU休眠模式的控制接口，如果需要切换CPU模式接口MCU\_vStandbyCtrl\_ChgToMode，请开启此定义  默认值=0，即不开启此功能 |
| MCU\_mGetWakeUpFactorEnable | 休眠模式下的唤醒原因获取接口MCU\_u32GetWakeUpFactor的开关  默认值=0，即不开启此功能 |
| MCU\_mFEinterruptEnable | 控制FE等级的中断开关，如果需要使用到FE级别中断或者需要下列函数，请开启此功能  MCU\_u16InterruptInit\_FE  MCU\_u16InterruptCtrl\_FE  MCU\_u16InterruptFlagClean\_FE  默认值=0，即不开启此功能 |
| MCU\_mClearIF\_Enable | 控制软件清除EI级别中断请求位的接口MCU\_u16InterruptFlagClean的开关，一般情况下，中断请求位在调用中断函数时自动被清零，没有调用中断函数时则请求位维持，  默认值=0，即不开启此功能，请求位由硬件自动清除 |
| MCU\_mSackFunctionEnable | 栈相关的接口使能开关，用于使能下面函数  MCU\_u32GetStackBottom  MCU\_u32GetStackTop  MCU\_u32GetCurrentStack  默认值=0，即不开启此功能， |
| MCU\_mPortModuleEnable | 端口模块的使能控制，端口和引脚操作相关的函数接口的总开关  默认值=1，即开启此功能， |
| MCU\_mPinDataOperEnable | 引脚数据读写接口的开关，用于使能下面函数  MCU\_u16PinRead  MCU\_u16PinWrite  默认值=1，即开启此功能， |
| MCU\_mPinDirectOperEnable | 引脚方向的控制接口开关，用于使能下面的函数  MCU\_u16PinDirectCfg  默认值=0，即不开启此功能， |
| MCU\_mPinPullUpOperEnable | 引脚上拉功能的接口开关，用于使能下面的函数  MCU\_u16PinPullUpCfg  默认值=0，即不开启此功能， |
| MCU\_mPinPullDownOperEnable | 引脚下拉功能的接口开关，用于使能下面的函数  MCU\_u16PinPullDownCfg  默认值=0，即不开启此功能， |
| MCU\_mPinOpenDEnable | 引脚开漏功能的接口开关，用于使能下面的函数  MCU\_u16PinOpenDCfg  默认值=0，即不开启此功能， |
| MCU\_mPinAltFunChgEnable | 引脚功能选择的接口开关，用于使能下面的函数  MCU\_u16PinAltFunCfg  默认值=0，即不开启此功能， |
| MCU\_mOSTMModuleEnable | OSTM模块的使能开关，用于使能整个OSTM模块  默认值=0，即不开启此功能， |
| MCU\_mOstmCtrlEnable | OSTM定时器的开始停止接口开关，用于使能下面的函数  MCU\_u16OSTMctrl  默认值=0，即不开启此功能， |
| MCU\_mOstmChangeEnable | OSTM定时器的变更定时长度接口开关，用于使能下面的函数  MCU\_u16OSTMChgInterval  默认值=0，即不开启此功能， |
| MCU\_mPWGAModuleEnable | PWGA模块的使能开关，用于使能整个PWGA模块  默认值=0，即不开启此功能， |
| MCU\_mPWGAdutyChangeEnable | PWGA模块调节占空比的接口开关，用于使能下面的函数  MCU\_u16PWGASetDuty  默认值=0，即不开启此功能， |
| MCU\_mTAUBDModuleEnable | TAUB,TAUD定时器模块的使能开关，用于使能整个TAUB,TAUD模块  默认值=0，即不开启此功能， |
| MCU\_mTAUJModuleEnable | TAUJ定时器模块的使能开关，用于使能整个TAUJ模块  默认值=0，即不开启此功能， |
| MCU\_mTauxSetCounterEnable | TAUX定时器模块更新计数个数的使能开关，用于使能下面的函数  MCU\_u16TAUxSetCounter  默认值=0，即不开启此功能， |
| MCU\_mTauxSetHzEnable | TAUX定时器模块使用PWM功能时，变更输出频率和占空比的接口的使能开关，用于使能下面的函数  MCU\_u16TAUxPpgSetHz  默认值=0，即不开启此功能， |
| MCU\_mTauxSetDutyEnable | TAUX定时器模块使用PWM功能时，变更输出占空比的接口的使能开关，用于使能下面的函数  MCU\_u16TAUxPpgSetDuty  默认值=0，即不开启此功能， |
| MCU\_mTuaxRunCtrlEnale | TAUX定时器的开始停止接口开关，用于使能下面的函数  MCU\_vTAUxRunCtrl  默认值=0，即不开启此功能， |
| MCU\_mTauxGetCounterEnable | TAUX定时器的获取计数结果接口开关，用于使能下面的函数  MCU\_u32TAUxGetCounter  默认值=0，即不开启此功能， |
| MCU\_mTauxGetTimeEnable | TAUX定时器的获取时间结果接口开关，一般用于输入计时功能，脉宽测量功能，用于使能下面的函数  MCU\_u32TAUxGetTime  默认值=0，即不开启此功能， |
| MCU\_mEINTModuleEnable | 外部中断模块的使能控制，与外部中断操作相关的函数接口的总开关  默认值=0，即不开启此功能， |
| MCU\_mEINTtrigChgFuncEnable | 变更外部中断请求的接口开关，用于使能下面的函数  MCU\_u16EINTchgIntCondition  默认值=0，即不开启此功能， |
| MCU\_mEINTOnOffCtrlFuncEnable | 外部中断请求开启或者关闭的接口开关，用于使能下面的函数  MCU\_u16EINTctrl  默认值=0，即不开启此功能， |
| MCU\_mEINTDeInitFuncEnable | 外部中断请求反初始化的接口开关，用于使能下面的函数  MCU\_vEINTDeInit  默认值=0，即不开启此功能， |
| MCU\_mADCModuleEnable | ADC转换模块的使能控制，与ADC相关函数接口的总开关  默认值=0，即不开启此功能， |
| MCU\_mADCstopEnable | ADC转换模块停止的接口开关，用于使能下面的函数  MCU\_u16ADCstop  默认值=0，即不开启此功能， |
| MCU\_mADCDeInitFuncEnable | ADC转换模块的反初始化接口开关，用于使能下面的函数  MCU\_vADCDeInit  默认值=0，即不开启此功能， |
| MCU\_mDMAModuleEnable | DMA模块的使能控制，与DMA操作相关的函数的总开关  默认值=0，即不开启此功能， |
| MCU\_mDMASetUpIfEnable | DMA用户配置参数的接口开关，用于使能下面函数  MCU\_u16DMAsetup  默认值=0，即不开启此功能， |
| MCU\_DMAdefaultSetEnable | DMA使用默认配置参数的接口开关，用于使能下面函数  MCU\_u16DMADefaultSetup  默认值=0，即不开启此功能， |
| MCU\_mDMADeInitFuncEnable | DMA使用默认配置参数的接口开关，用于使能下面函数  MCU\_vDMADeInit  默认值=0，即不开启此功能， |
| MCU\_mRLIN3UartModuleEnable | RLIN3模块的使能控制，本代码只实现了UART功能（LIN节点功能一般使用购买代码），与LIN3(UART功能)操作相关的函数的总开关  默认值=0，即不开启此功能， |
| MCU\_mRlin3CtrlEnale | UART运行控制的接口开关，用于使能下面函数  MCU\_u16RLIN3UartCtrl  默认值=0，即不开启此功能， |
| MCU\_mRLIN3DeInitFuncEnable | UART反初始化的接口开关，用于使能下面函数  MCU\_vRLIN3Uart\_Deinit  默认值=0，即不开启此功能， |
| MCU\_mIICModuleEnable | IIC模块的使能控制，本代码只实现了IIC主节点的读写基本功能，从节点读写没有实现，与IIC操作相关的函数的总开关  默认值=0，即不开启此功能， |
| MCU\_mIICReadFunctionEnable | IIC模块的主节点读取功能的接口开关，用于使能下面函数  MCU\_u16IIC\_MasterRead  默认值=0，即不开启此功能， |
| MCU\_mIICRestartEnable | IIC模块的主节点产生re-start信号的接口开关，用于使能下面函数  MCU\_u16IIC\_MasterReStartCondition  默认值=0，即不开启此功能， |
| MCU\_mI2CMaxSlaveInMaster | IIC模块的从节点个数，不得小于1  默认值=1，即IIC挂载一个从节点， |
| MCU\_mIICdeinitEnable | IIC模块的主节点反初始化的接口开关，用于使能下面函数  MCU\_vIICdeinit  默认值=0，即不开启此功能， |
| MCU\_mCSIGModuleEnable | CSIG模块的使能控制，与CSIG模块操作函数的总开关，本代码只实现主节点模式下的功能  默认值=0，即不开启此功能， |
| MCU\_mCSIGMasterSynchRWEnable | CSIG模块主节点同步读写的接口开关，用于使能下面函数  MCU\_u16CSIG\_MasterSynchRW  默认值=0，即不开启此功能， |
| MCU\_mCSIGMasterWriteEnable | CSIG模块主节点发送的接口开关，用于使能下面函数  MCU\_u16CSIG\_MasterWrite  默认值=0，即不开启此功能， |
| MCU\_mCSIGMasterReadEnable | CSIG模块主节点发送的接口开关，用于使能下面函数  MCU\_u16CSIG\_MasterRead  默认值=0，即不开启此功能， |
| MCU\_mCSIGClkPhaseChgEnable | CSIG模块主节点时钟相位变更的接口开关，用于使能下面函数  MCU\_u16CSIG\_ChgClkType  默认值=0，即不开启此功能， |
| MCU\_mCSIGdeinitEnable | CSIG模块主节点反初始化的接口开关，用于使能下面函数  MCU\_vCSIGdeinit  默认值=0，即不开启此功能， |
| MCU\_mCSIHModuleEnable | CSIH模块的使能控制，与CSIH模块操作函数的总开关，本代码只实现主节点模式下的功能  默认值=0，即不开启此功能， |
| MCU\_mCSIHMasterSynchRWEnable | CSIH模块主节点同步读写的接口开关，用于使能下面函数  MCU\_u16CSIH\_MasterSynchRW  默认值=0，即不开启此功能， |
| MCU\_mCSIHMasterWriteEnable | CSIH模块主节点发送的接口开关，用于使能下面函数  MCU\_u16CSIH\_MasterWrite  默认值=0，即不开启此功能， |
| MCU\_mCSIHMasterReadEnable | CSIH模块主节点发送的接口开关，用于使能下面函数  MCU\_u16CSIH\_MasterRead  默认值=0，即不开启此功能， |
| MCU\_mCSIHClkPhaseChgEnable | CSIH模块主节点时钟相位变更的接口开关，用于使能下面函数  MCU\_u16CSIH\_ChgClkType  默认值=0，即不开启此功能， |
| MCU\_mCSIHJobEnable | CSIH模块主节点JOB功能的接口开关，用于使能下面函数  MCU\_vCSIHJobLastData  及JOB中断  默认值=0，即不开启此功能， |
| MCU\_mCSIHdeinitEnable | CSIH模块主节点反初始化的接口开关，用于使能下面函数  MCU\_vCSIHdeinit  默认值=0，即不开启此功能， |

# 复位相关的配置

可被配置的与复位相关的选项如下：

* 1. OPT\_Byte0的配置

OPT\_Byte0的详细描述请参考spec的35.11章节，本代码要求OPT\_Byte0的设定值为0x9AAFFFFF，

具体含义为

1. watchdog0和watchdog1都启用，且使用软件初始化以后才开始进行工作
2. 看门狗的时间设定通过用户配置表进行配置
3. 看门狗溢出以后，MCU将进行复位
4. JP口默认设定为通用的IO口
5. 高低压监测使能
   1. Watchdog的配置

看门狗的时间配置见MCU\_mResetTime\_WD0和MCU\_mResetTime\_WD1的参数配置和MCU\_stClockCfg\_F1L与看门狗相关的时钟项的配置watchdog\_0对应ClkDiv\_C\_AWO\_WDTAD，watchdog\_1不能改变，只能使用内部低速晶振

* 1. 低压重启功能

低压重启功能的电压边界配置参考MCU\_mLVI\_Level，

* 1. 接口说明

具体的接口说明如下：

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_vLowVoltDetectInit | 【内部函数】低压重启的初始化函数 | 输入：  无  返回值：  无 |
| MCU\_vWatchDogInit | 【内部函数】看门狗的初始化函数，看门狗在执行完这个函数以后才开始进行计数，如果溢出则进行复位 | 输入：  无  返回值：  无 |
| MCU\_vWatchDogClean | 【外部函数】看门狗计数清零函数接口，调用此函数以防止看门狗溢出复位 | 输入：  无  返回值：  无 |
| MCU\_vResetCtrl\_Reset | 【外部函数】MCU软件复位函数接口，调用此函数复位MCU | 输入：  无  返回值：  无 |
| MCU\_u32GetResetReason | 【外部函数】MCU复位原因的获取 | 输入：  无  返回值：  返回值为一个uint32的数，一个bit代表一种复位原因，具体如下：  MCU\_mResetReason\_DeepSTOP bit\_10  MCU\_mResetReason\_PowerUp bit\_9  MCU\_mResetReason\_ExternalReset bit\_8  MCU\_mResetReason\_CVM bit\_7  MCU\_mResetReason\_LVI bit\_6  MCU\_mResetReason\_CLMA2 bit\_5  MCU\_mResetReason\_CLMA1 bit\_4  MCU\_mResetReason\_CLMA0 bit\_3  MCU\_mResetReason\_WDTA1 bit\_2  MCU\_mResetReason\_WDTA0 bit\_1 MCU\_mResetReason\_SoftwareRest bit\_0 |

* 1. 关联模块

F1L\_USRCFG.h中关于看门狗时间长度的配置

F1L\_USRCFG.h中关于低压复位电压的配置

F1L\_USRCFG.c中关于看门狗时钟源的配置

OPT\_BYTE0的配置

* 1. 注意事项

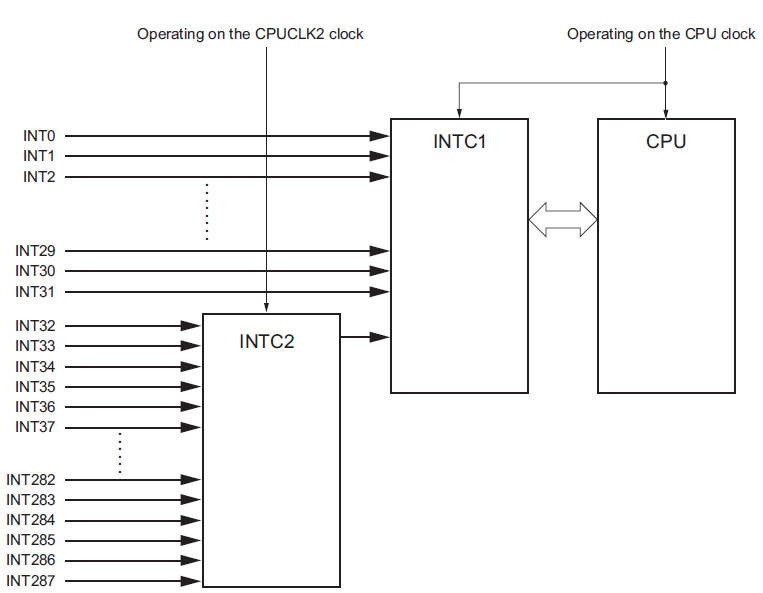
1. 用户只需配置相关的参数即可，内部函数由底层内部调用，用户无需操作，用户可根据实际需要调用外部函数即可。
2. 此章节的内容如果配置不当，可能会引起周期复位

# 中断模块配置

* 1. 模块内部逻辑图

FE级别的内部逻辑直接连接到各个对应中断的特定入口，详见说明书

EI中断内部的逻辑图如下：



* 1. 模块特性

本模块分为两个等级的中断，

一个等级为FE级别， FE级别又可以分为可恢复中断级别FEINT和不可恢复中断级别FENMI

可恢复的中断可以像普通中断一样使用，而不可恢复的中断则只能复位操作才能清除该中断请求

一个等级为EI级别，EI级别的中断分别INTC1和INTC2片区的中断源，同时一个中断向量可能对应两个不同的中断请求，具体的中断请求还需要SELB\_INTC1与SELB\_INTC2来确定，本代码不要求配置，只使用默认的请求向量号进行中断请求。且请求是基于table模式的，而不是优先级模式的。

同一个中断，如果使能了FE级别的中断，则EI级别的中断将不会再响应

* 1. 具体参数配置

模块开关配置请参见3.2 模块使能章节

模块配置项，F2L\_UsrCfg.c中关于中断模块的配置如下：

MCU\_stFEMICfg用于配置可恢复FE级别的中断配置

MCU\_stFENMICfg用于配置不可恢复FE级别的中断配置

EI级别的中断配置由各个应用模块单独配置，具体细节请参考具体模块的配置。

**重要**：

1. 当EI级别的中断请求处于边沿触发时，在进入中断服务函数后硬件自动清除请求，而处于电平触发时，中断请求需要软件进行清除！详细参考page299的table6.4，detection type栏。
2. FE级别的不可恢复类型（NMI）中断请求在进入中断服务时会自动清除请求位
3. FE级别的可恢复类型中断请求需要软件进行清除

默认情况下，以下中断都不使用，用MCU\_vNotExpINT函数入口代表中断不开启

MCU\_stFEMICfg的具体配置项说明如下：

|  |  |
| --- | --- |
| 配置项 | 说明 |
| LVIL | LVI检测的下降沿中断入口，LVI有复位模式和中断模式，当使用中断模式时检测到下降沿，LVI请求将进入此入口  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCDEEP0 | Data Flash ECC错误2bits  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCDCNRAM | RSCAN0 ECC错误2bits  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCDCSIH0 | CSIH0 ECC错误2bits  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCDCSIH1 | CSIH1 ECC错误2bits  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCDCSIH2 | CSIH2 ECC错误2bits  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCDCSIH3 | CSIH3 ECC错误2bits  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCSDFLI0 | Code Flash ECC错误1bits  默认值 = 不开启，即MCU\_vNotExpINT |
| ECCRAM | RAM ECC错误1bits  默认值 = 不开启，即MCU\_vNotExpINT |
| OSTM0\_FE | OSTM中断请求  默认值 = 不开启，即MCU\_vNotExpINT，OSTM中断默认情况下使用EI级别的中断请求 |
| LVIH | LVI检测的上升沿中断入口，LVI有复位模式和中断模式，当使用中断模式时检测到上升沿，LVI请求将进入此入口  默认值 = 不开启，即MCU\_vNotExpINT |

MCU\_stFENMICfg的具体配置项说明如下：

|  |  |
| --- | --- |
| 配置项 | 说明 |
| TNMInterrupt | 外部引脚输入的NMI信号请求，不可屏蔽的外部中断  默认值 = 不开启，即MCU\_vNotExpINT |
| WDTA0\_Isr | Watchdog\_0溢出中断  默认值 = 不开启，即MCU\_vNotExpINT |
| WDTA1\_Isr | Watchdog\_1溢出中断  默认值 = 不开启，即MCU\_vNotExpINT |

* 1. 接口说明

FE级别中断相关接口

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_u16InterruptInit\_FE | 【内部函数】FE级别的中断初始化，根据用户配置判定是否需要开启FE级别的中断请求  此函数只在复位时运行一次 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16InterruptCtrl\_FE | 【外部函数】应用层FE级别中断的使能控制接口，用于使能或者不使能FE级别的某个中断 | 输入：  u16VectNum：MCU\_enIsrID\_FE枚举类型的向量编号  u8IntEnable：期望值，0=关闭中断  返回值：  函数执行结果 |

EI级别中断相关接口

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_u16InterruptCtrl | 【外部函数】应用层EI级别中断的使能控制接口，用于使能或者不使能EI级别的某个中断 | 输入：  u16VectNum：MCU\_enIsrID枚举类型的向量编号  u8IntEnable：期望值，0=关闭中断 |
| MCU\_u16InterruptFlagClean | 【外部函数】EI级别的请求位清除，LEVEL触发的中断请求调用此函数时，没有作用，edge触发的中断由硬件自动清除，详细参考硬件spec的page299的table6.4，detection type栏 | 输入：  u16VectNum：MCU\_enIsrID枚举类型的向量编号  返回值：函数执行结果 |
| MCU\_u16ChangeVectorPriority | 【外部函数】EI级别的中断的优先级变更，本代码使用独占CPU的table模式，故一般不使用此函数 | 输入：  u16VectNum：MCU\_enIsrID枚举类型的向量编号  u16TargePri：期望的优先级，取值0到7，0为最高优先级  返回值：函数执行结果 |

无用中断接口的填充

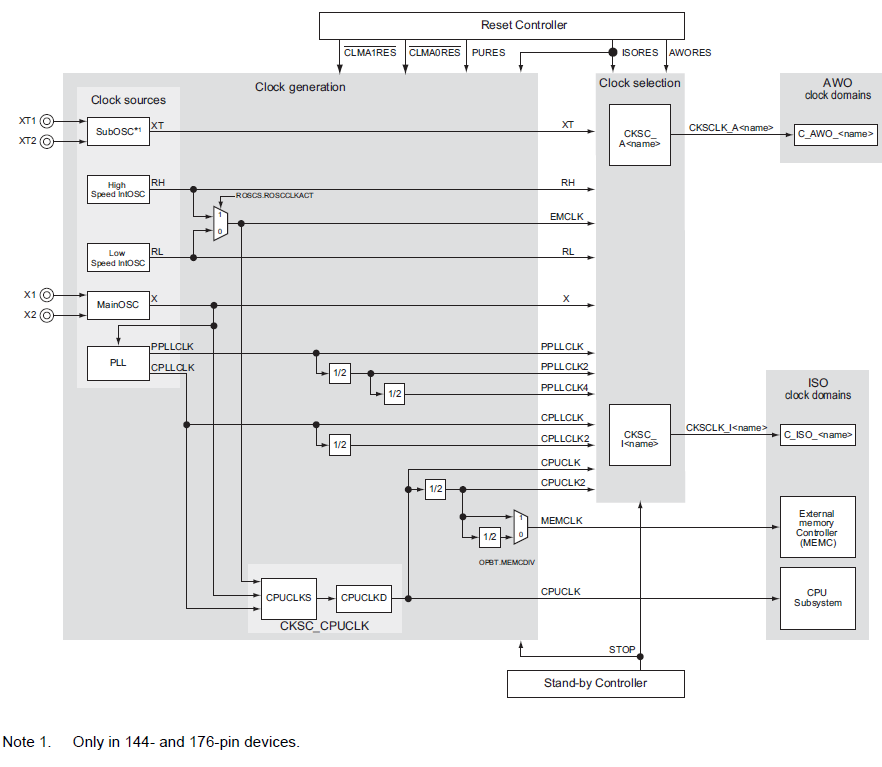
|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_vNotExpINT | 【外部函数】用于填充或者标记中断不使用 | 输入：  无  返回值：  无 |
| MCU\_vNotExpINT\_u8 | 【外部函数】用于填充带uint8参数的不使用中断 | 输入：  u8Dummy：仅用于填充  返回值：  无 |
| MCU\_vNotExpINT\_u16 | 【外部函数】用于填充带uint16参数的不使用中断 | 输入：  U16Dummy：仅用于填充  返回值：  无 |

* 1. 关联配置项

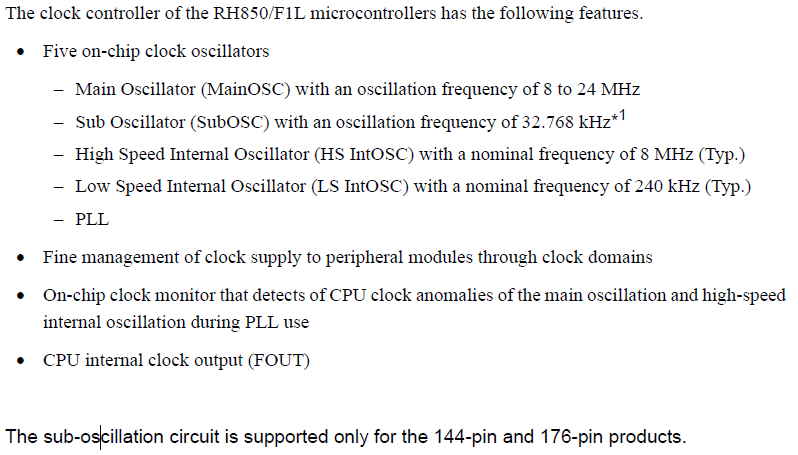
1. F1L\_USRCFG.c中FE级别的中断配置
2. 各个资源模块的回调函数的配置
   1. 注意事项
3. FE级别的中断请求位在ISR函数内完成，
4. EI级别的中断请求中，edge触发的中断硬件自动清零，level触发的中断由硬件控制，无法进行清除
5. EI级别的IIC中断请求位在ISR函数内完成，
6. 本代码的中断建议使用中断不可嵌套的方式进行，即MCU\_mINTNestingEnable定义为0
7. 本代码的中断使用的table列表的向量方式进行响应，不使用优先级的方式

# 时钟模块配置

* 1. 模块内部逻辑图



* 1. 模块特性



* 1. 具体配置选项说明

F1L\_USRCFG.c文件的时钟模块的配置属性结构体为MCU\_tstClockUsrCfg，具体参数说明将下表：

|  |  |
| --- | --- |
| 配置项 | 配置说明 |
| MainOsc\_StableTime | 主晶振起振的等待时间，单位为us  具体项目应该配置成实际起振时间的2倍，以保证晶振起振正常  默认1000us |
| MainOscStopInStandbyMode | MCU进入休眠模式时主晶振的工作状态控制  0：主晶振在休眠模式时继续工作  1：主晶振在休眠模式时停止工作  默认值=1，即休眠时主晶振停止工作 |
| SubOsc\_FTX\_Speed | MCU外接从晶振的频率，只适用于带外部从晶振的型号  0：无外接从晶振  32768：接了外部从晶振（外部从晶振只支持32.768KHz） |
| SubOsc\_StableTime | MCU外接从晶振的等待时间，只适用于带外部从晶振的型号  具体项目应该配置成实际起振时间的2倍，以保证晶振起振正常  默认1000us |
| IntOscLS\_FRL\_Speed | 内部低速时钟的频率，  默认值=240000Hz，不能修改 |
| IntOscHS\_FRH\_Speed | 内部高速时钟的频率，  默认值=8000000Hz，不能修改 |
| CPLLCLK\_Speed | CPU运行频率选择，根据不同外部晶振对应的CPU频率选择，可以选择一下的其中一个：  #define PLL\_MainOsc\_8M\_80M 0  #define PLL\_MainOsc\_8M\_64M 1  #define PLL\_MainOsc\_8M\_40M 2  #define PLL\_MainOsc\_8M\_30M 3  #define PLL\_MainOsc\_16M\_80M 4  #define PLL\_MainOsc\_16M\_64M 5  #define PLL\_MainOsc\_16M\_40M 6  #define PLL\_MainOsc\_16M\_30M 7  默认值= PLL\_MainOsc\_8M\_80M |
| MEMCLK\_Speed | 外部存储时钟速度控制，只有176pin产品有此功能，在此处不适用  具体分频由OPT byte的OPBT.MEMCDIV位控制，  MEMCDIV = 0: fMEMCLK = fCPUCLK / 2  MEMCDIV = 1: fMEMCLK = fCPUCLK / 4 |
| ClkDiv\_C\_AWO\_WDTAD | Watchdog0时钟分频的选择，鉴于watchdog\_0和watchdog\_1的硬件默认时钟不一样，当前软件的配置默认为2，即与watchdog\_1一样直接使用内部低速时钟  01B: LS IntOSC / 128 (HW reset value)  10B: LS IntOSC / 1  默认值= 10B: LS IntOSC，  watchdog\_1的时钟源不能变更，只能使用LS IntOSC |
| ClkSel\_C\_AWO\_TAUJS | 配置AWO\_TAUJS的时钟源，  000B: Disabled  001B: HS IntOSC (HW reset value)  010B: MainOSC  011B: LS IntOSC  100B: PPLLLCLK/2  默认值=2，选择外部晶振作为AWO\_TAUJS的时钟源  实际输出的时钟频率为（时钟源/分频） |
| ClkDiv\_C\_AWO\_TAUJD | 配置AWO\_TAUJS的时钟源分频  001B: X/1 (HW reset value)  010B: X/2  011B: X/4  100B: X/8  默认值=1，不分频 |
| TAUJStopInStandbyMode | 配置AWO\_TAUJS时钟在休眠模式下的工作状态  0：休眠时继续工作  1：休眠时停止工作  默认值=1，即休眠时AWO\_TAUJS时钟停止 |
| ClkDiv\_C\_AWO\_RTCAD | 配置AWO\_RTCAD的时钟源，  00B: Disable (HW reset value)  01B: SubOSC  10B: MainOSC  11B: LS IntOSC  默认值=0，即不使能RTCA模块  时钟实际输出频率为（时钟源/分频） |
| ClkDiv\_C\_AWO\_RTCAD | 配置AWO\_RTCAD的时钟源的分频数  001B: X/1 (HW reset value)  010B: X/2  011B: X/4  100B: X/8  默认值=0，即不使能时钟源 |
| RTCAStopInStandbyMode | 配置AWO\_RTCAD的时钟在休眠模式下的工作状态  0：休眠时继续工作  1：休眠时停止工作  默认值=1，即休眠模式时，AWO\_RTCAD时钟停止 |
| ClkSel\_C\_AWO\_ADCAS | 配置AWO\_ADCAS的时钟源，  00B: Disabled  01B: HS IntOSC (HW reset value)  10B: MainOSC  11B: PPLLCLK/2  默认值=1，即使用内部高速晶振做时钟源  时钟实际输出频率为（时钟源/分频） |
| ClkDiv\_C\_AWO\_ADCAD | 配置AWO\_ADCAS时钟的分频数  001B: X/1 (HW reset value)  010B: X/2  默认值=1，即不分频 |
| ADCAStopInStandbyMode | 配置AWO\_ADCAS在休眠模式下的工作状态  0：休眠时继续工作  1：休眠时停止工作  默认值=1，即休眠时AWO\_ADCAS停止 |
| ClkSel\_C\_AWO\_FOUT | 配置AWO\_FOUT时钟输出的时钟源  000B: Disabled (HW reset value)  001B: MainOSC  010B: HS IntOSC  011B: LS IntOSC  100B: SubOSC  101B: CPLLCLK/2  110B: PPLLCLK/4  默认值=0，即不使用时钟输出功能  FOUT实际输出频率为（时钟源/分频数） |
| ClkDiv\_C\_AWO\_FOUT | 配置AWO\_FOUT的分频数，有效范围0~63  00H: Clock output is stopped  01H: N = 1  02H: N = 2  ...  3EH: N = 62  3FH: N = 63  默认值=0.即不输出FOUT |
| ClkSel\_C\_ISO\_CPUCLKS | 配置ISO\_CPUCLKS的时钟源  00B: Setting prohibited  01B: EMCLK (HW reset value)  10B: MainOSC  11B: CPLLCLK  默认值=3，即使用PLL作为CPU的时钟源，PLL通过倍频以后得到CPLLCLK\_Speed频率的输入，实际频率=（时钟源/分频数） |
| ClkDiv\_C\_ISO\_CPUCLKD | 配置ISO\_CPUCLKD的分频数  001B: X/1 (HW reset value)  010B: X/2  011B: X/4  100B: X/8  默认值=1，即不分频 |
| ClkSel\_C\_ISO\_PER1 | 配置ISO\_PER1的时钟源  00B: Disabled  01B: CPUCLK/2 (HW reset value)  10B: PPLLCLK  默认值=1，即CPU的2分频 |
| ClkSel\_C\_ISO\_PER2 | 配置ISO\_PER2的时钟源  00B: Disabled  01B: CPUCLK/2 (HW reset value)  10B: PPLLCLK  默认值=1，即CPU的2分频 |
| ClkSel\_C\_ISO\_LINS | 配置ISO\_LINS的时钟源  00B: Disabled  01B: CPUCLK/2 (HW reset value)  10B: MainOSC  11B: PPLLCLK/2  默认值=2，即使用外部晶振做ISO\_LINS的时钟源  实际输出频率为（时钟源/分频数） |
| ClkDiv\_C\_ISO\_ILIND | 配置ISO\_LINS的分频数  001B: X/1 (HW reset value)  010B: X/4  011B: X/8  默认值=1，即不分频 |
| LINStopInStandbyMode | 配置ISO\_LINS时钟在休眠模式时的工作状态  0：休眠时ISO\_LINS继续工作  1：休眠时ISO\_LINS停止工作  默认值=1，即休眠时ISO)LIN停止工作 |
| ClkSel\_C\_ISO\_ADCAS | 配置ISO\_ADCA时钟源  00B: Disabled  01B: HS IntOSC RH (8 MHz) (HW reset value)  10B: MainOSC  11B: PPLLCLK/2  默认值=1，即使用内部高速时钟  实际输出频率为（时钟源/分频数） |
| ClkDiv\_C\_ISO\_ADCAD | 配置ISO\_ADCA的时钟分频数  001B: X/1 (HW reset value)  010B: X/2  默认值=1，即不分频 |
| ClkSel\_C\_ISO\_CANS | 配置ISO\_CANS的时钟源  00B: Disabled  01B: MainOSC  10B: CPLLCLK  11B: CPUCLK (HW reset value)  默认值=3，即使用硬件复位时的时钟为ISO\_CANS的时钟源 |
| CANStopInStandbyMode | 配置ISO\_CANS时钟在休眠模式下的工作状态  0：休眠时继续工作  1：休眠时停止工作  默认值=1，即休眠模式下ISO\_CANS停止工作 |
| ClkSel\_C\_ISO\_CSIS | 配置ISO\_CSIS的时钟源  00B: Disabled  01B: CPUCLK (default)  10B: PPLLCLK  默认值=1，即不使能ISO\_CSIS |

备注：

1. 在时钟选择时，在同一个时钟串上的时钟设定可能会相互影响，如修改了CPU时钟，PER1，PER2的时钟频率也可能被影响到
2. 时钟监控模块clock\_monitor没有实现，请将模块配置关闭
3. CAN，LIN模块的驱动代码一般由购买得到，可以选择此文件不配置，由各自的驱动端进行配置
   1. 接口说明

时钟模块相关的接口说明如下表：

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_vClockInit\_Basic | 【内部函数】初始化mainOSC，PLL时钟和CPU时钟，subOSC | 输入：  无  返回值：  无 |
| MCU\_vClockInit\_AWO | 【内部函数】初始化各个AWO时钟模块的设定，如WDTA,TUAx等模块的时钟源 | 输入：  无  返回值：  无 |
| MCU\_vClockInit\_ISO | 【内部函数】初始化各个ISO时钟模块的设定，如PER1，PER2等模块的时钟源 | 输入：  无  返回值：  无 |
| MCU\_vClockInit\_ClockMonitor | 【内部函数】初始化各个时钟监控模块的参数 | 输入：  无  返回值：  无 |
| MCU\_u16ClockInit | 【内部函数】时钟模块初始化的总接口 | 输入：  无  返回值：  函数执行结果 |

* 1. 关联模块

1. F1L\_USRCFG.c的关于时钟的配置
2. 各个资源模块的时钟源的选择都由章节决定
   1. 注意事项
3. 本代码不支持变更时钟设定的接口，时钟设定只能在配置宏或者配置结构体内实现
4. 全局变量MCU\_stClockStatus用于记录各个时钟的运行频率

# 栈相关的接口

* 1. 栈相关的信息

栈的规划由链接文件决定，具体细节请参考链接章节

此系列的MCU栈的生长方向是由RAM高地址向下生长的，即栈顶的物理地址小于栈底的地址，函数调用一层则SP往低地址端堆叠一层。

* 1. 栈相关的接口

栈的接口列表如下：

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_u32GetStackBottom | 【外部函数】获取栈的底部地址 | 输入：  无  返回值：  为链接器提供的栈底的地址 |
| MCU\_u32GetStackBottom | 【外部函数】获取栈的顶部地址 | 输入：  无  返回值：  为链接器提供的栈顶的地址 |
| MCU\_u32GetCurrentStack | 【外部函数】获取当前SP的地址 | 输入：  无  返回值：  当前SP的地址 |

* 1. 关联模块

F1L\_USRCFG.h文件中MCU\_mSackFunctionEnable的宏开关用于使能以上接口。

* 1. 注意事项

1. 栈顶和栈底的地址为链接器给出，是一个常数
2. 获取当前栈的使用深度时，需要加上当前函数使用到的深度，即：

总使用深度= MCU\_u32GetCurrentStack + 当前函数用到的栈的深度

1. 调用MCU\_u32GetCurrentStack函数时，建议在中断内使用，以免因为函数调用的层次一样导致返回一样的执行结果

# 休眠唤醒配置

* 1. 模块特性

本MCU支持的CPU运行模式如下：

1. RUN模式

普通的运行模式，所有资源模块处于正常状态

1. HALT模式：

CPU停止操作，其他的各个时钟和IO状态均保持当前状态，耗电流较大

1. STOP模式

CPU停止操作，各个时钟根据F1L\_USRCFG.c的配置进行停止或者继续工作，IO状态保持当前状态，好电流小

1. DeepStop模式

与STOP模式类似，但是ISO的时钟被强制停止，耗电流更小，但是被唤醒到普通运行模式时，CPU进行复位，且复位以后IO口处于IOHold状态，必须先将IO release以后才能进行IO的输出操作

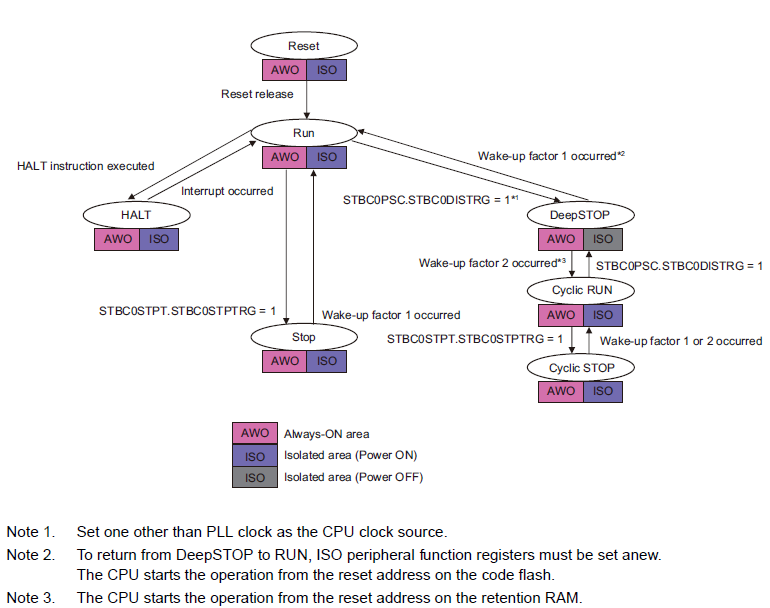
1. CyclicRUN模式

DeepStop模式下的一个特殊模式，用于CPU不复位，代码在retentionRAM内执行，被唤醒以后会执行retentionRAM的起始的地址入口

1. CyclicSTOP模式

与CyclicRUN模式相对应的STOP模式，

在DeepStop模式下，由唤醒因素的配置决定CPU的运行状态是RUN模式还是CyclicRUN模式，具体的模式转换键下图：



* 1. 模块参数配置

F1L\_USRCFG.c内的休眠唤醒模块的配置结构体为MCU\_stSTBCCfg，分别为唤醒因素WU1（wakeup factor 1，包含ISO类别的因素），WU2 （wakeup factor 2），在不同模式下的不同唤醒因素会让CPU处于不同的运行模式，

WU0因素配置：

|  |  |
| --- | --- |
| 配置项 | 配置说明 |
| TNMI | 不可屏蔽外部中断唤醒因素  默认值 = 0，即不使能此因素 |
| WDTA0NMI | 看门狗FE级别的中断唤醒因素  默认值 = 0，即不使能此因素 |
| INTLVIL | VCC低压检测的下降沿唤醒因素  默认值 = 0，即不使能此因素 |
| INTPx | 外部中断唤醒因素  默认值 = 0，即不使能此因素 |
| INTWDTA0 | 看门狗窗口中断唤醒因素  默认值 = 0，即不使能此因素 |
| WUTRGx | LPS中断唤醒因素  默认值 = 0，即不使能此因素 |
| INTTAUJ0Ix | TAUJ0中断唤醒因素  默认值 = 0，即不使能此因素 |
| INTRTCA01S  INTRTCA0AL  INTRTCA0R | RTCA模块中断唤醒因素  默认值 = 0，即不使能此因素 |
| INTDCUTDI | JTAG中断唤醒因素  默认值 = 0，即不使能此因素 |
| INTKR0 | 按键模块中断唤醒因素  默认值 = 0，即不使能此因素 |
| INTRCANGRECC  INTRCANxREC | CAN模块中断唤醒因素  默认值 = 0，即不使能此因素 |

WU2因素配置

|  |  |
| --- | --- |
| 配置项 | 配置说明 |
| INTADCA0Ix | ADC模块唤醒因素  默认值 = 0，即不使能此因素 |
| INTRLIN3x | RLIN3x模块唤醒因素  默认值 = 0，即不使能此因素 |
| INTTAUJ0Ix | TUAJx模块沿唤醒因素  默认值 = 0，即不使能此因素 |
| INTRTCA01S  INTRTCA0AL  INTRTCA0R | RTCA模块中断唤醒因素  默认值 = 0，即不使能此因素 |

* 1. 模块接口说明

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_vStandbyCtrl\_IoHold | 【内部函数】用于IO release | 输入：  PortIDmaskBit：端口掩码，为0  返回值：  无 |
| MCU\_vStandbyCtrl\_ChgToMode | 【外部函数】CPU运行模式控制的接口 | 输入：  u8ExpMode：MCU\_enRunModeID枚举类型，期望运行模式  返回值：  无 |
| MCU\_vStandbyCtrl\_SetWakeUpFactor | 【内部函数】根配置据设定复位因素 | 输入：  无  返回值：  无 |
| MCU\_vStandbyCtrl\_CleanAllWakeUpFactor | 【外部函数】清除所有唤醒请求因素 | 输入：  无  返回值：  无 |
| MCU\_u32GetWakeUpFactor | 【外部函数】获取唤醒因素 | 输入：  无  返回值：  MCU\_enWakeFactorID枚举类型的唤醒原因 |

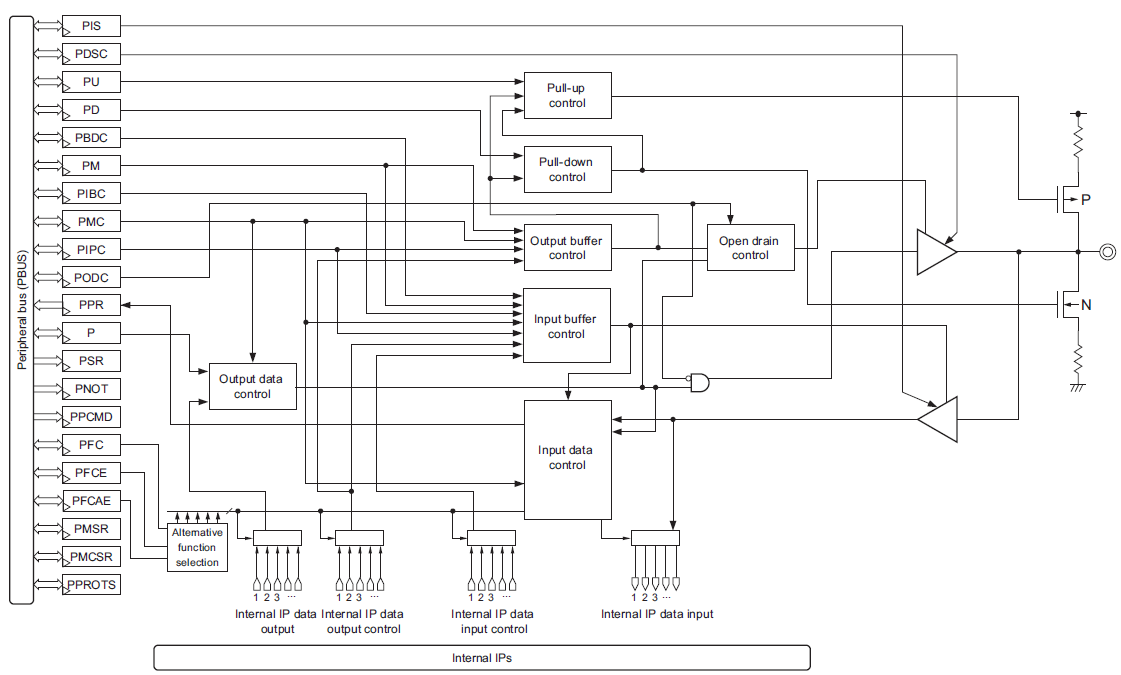
* 1. 注意事项

1. 唤醒原因的中断响应要正常开启，如使用了TAUJ的唤醒，则TAUJ的对应中断需要使能，这部分在各个资源模块的配置中实现
2. 底层在复位以后就使能了唤醒因素，在RUN模式下是不影响CPU运行的，
3. 在RUN模式进入其他模式时，在适当的时候调用清除唤醒请求的接口函数以清除RUN模式下产生的唤醒请求，但是在不恰当的时候清除唤醒请求可能导致唤醒因素被清除而无法被唤醒
4. DeepStop模式唤醒到RUN模式时，MCU会进行复位且IO处于Hold状态，底层代码在进入用户层代码时，会将IOrelease到可以操作状态
5. DeepStop模式唤醒到cyclicRUN模式时，MCU将调用retentionRAM的起始地址进行执行代码
6. 如果使用到ISO部分唤醒，则该ISO模块的时钟在休眠模式下不能停止

# Port配置

* 1. 模块内部逻辑

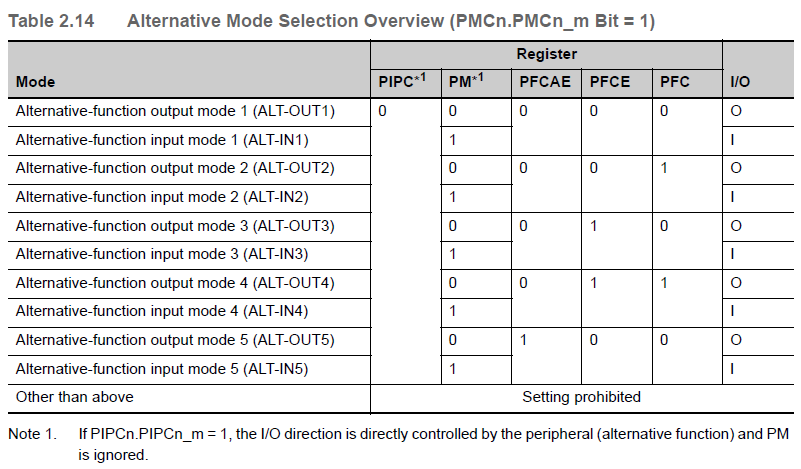
F1L的port内部逻辑如下：



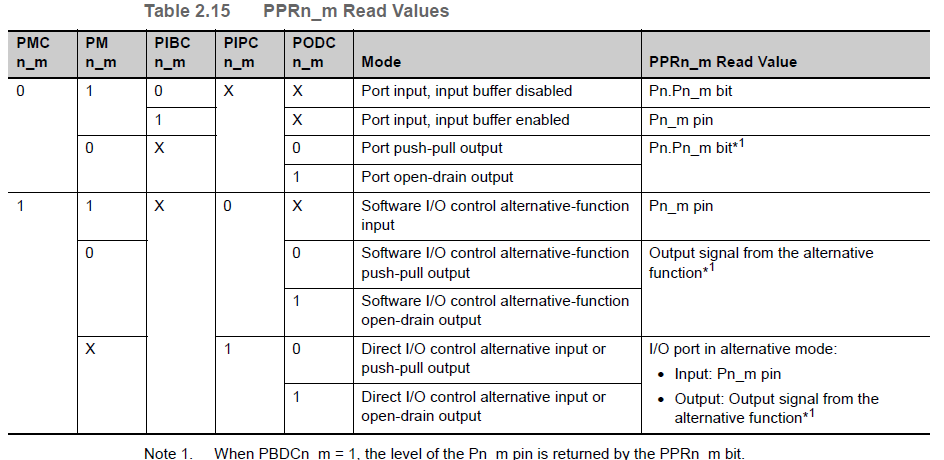
* 1. 模块特性

Port模块的寄存器组关系如下：

1. 如果引脚使用普通IO的输入输出功能，则复用功能的选择不需要开启，即PMC=0
2. 引脚的上拉下拉功能独立于复用功能，漏级开路功能（OpenDrain）也是独立于复用功能
3. 用于引脚的功能多选一的配置表：各个引脚的有效复用功能请参见spec的page142到page182详细列表



1. PBDC开启时，引脚电平监控寄存器PPR与引脚的输入输出方向无关，他反应的实际状态如下表



* 1. 具体配置参数

在F1L\_USRCFG.c文件的结构体MCU\_stPinUserCfg\_Port，MCU\_stPinUserCfg\_JPort，MCU\_stPinUserCfg\_APort，MCU\_stPinUserCfg\_IPort分别用于配置Port类型，JPort类型，APort类型，IPort类型的端口，各类端口的各个引脚均可进行配置，配置项说明如下：

|  |  |
| --- | --- |
| 配置项 | 配置说明 |
| AltFun | 引脚的功能选择：  #define GPIO\_Out 0u  #define GPIO\_In 1u  #define AltOut\_1 2u  #define AltIn\_1 3u  #define AltOut\_2 4u  #define AltIn\_2 5u  #define AltOut\_3 6u  #define AltIn\_3 7u  #define AltOut\_4 8u  #define AltIn\_4 9u  #define AltOut\_5 10u  #define AltIn\_5 11u  #define AltOut\_6 12u  #define AltIn\_6 13u  #define AltOut\_7 14u  #define AltIn\_7 15u  #define ADinput GPIO\_In  可以根据具体使用场景进行配置  默认值 = 255，即保持HW reset value |
| ResetLevel | 引脚用作GPIO的输出模式时，port初始化以后的默认电平  默认值 = 0， 即输出低电平 |
| PullUpEnable | 引脚的上拉电阻功能使能控制  默认值 = 0，即不开启此功能 |
| PullDownEnable | 引脚的下拉电阻功能使能控制  默认值 = 0，即不开启此功能 |
| OpenDEnable | 引脚的漏级开路功能使能控制  默认值 = 0，即不开启此功能 |
| PinPutBufType | 引脚的输入缓存控制，（SHMT模式，TTL模式等）  默认值 = 0，即采用默认值 |
| PinHighSpeedEnable | 引脚的输出缓存速度控制，  默认值 = 0，即采用低于10MHz，等于1时为小于40MHz |
| PbdcEnable | 引脚的输入输出方向多向控制控制，  默认值 = 0，即不使能此功能 |

备注：

1. port类型的配置可以通过宏PinConf来快速配置一个引脚
2. port类型如果使用到IIC功能的引脚，则请用宏PinConf\_IICpin来进行配置
3. Jport类型的配置可以通过宏JPinConf来快速配置一个引脚
4. Aport类型的配置可以通过宏APinConf来快速配置一个引脚.
5. Iport类型的配置可以通过宏IPinConf来快速配置一个引脚.
6. Iport端口只能作为输入端口使用
7. 所有的配置项只能将原有的填充宏\_\_ThisPin\_NotMount\_\_替换掉，不能增减，
8. 如果使用到一个引脚的Special Function，则可以选择不用配置该引脚，引脚自动具备该功能
   1. 接口说明

端口相关的函数列表如下：

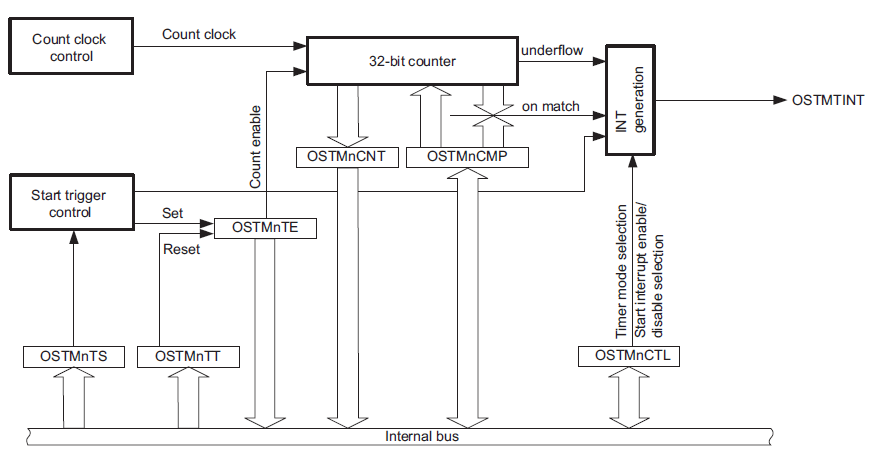
|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 参数说明 |
| MCU\_u16PortInit\_Port | 【内部函数】port类型的端口初始化函数，此函数根据用户配置MCU\_stPinUserCfg\_Port初始化端口 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16PortInit\_JPort | 【内部函数】Jport类型的端口初始化函数，此函数根据用户配置MCU\_stPinUserCfg\_JPort初始化端口 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16PortInit\_APort | 【内部函数】Aport类型的端口初始化函数，此函数根据用户配置MCU\_stPinUserCfg\_APort初始化端口 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16PortInit\_IPort | 【内部函数】Iport类型的端口初始化函数，此函数根据用户配置MCU\_stPinUserCfg\_IPort初始化端口 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16PortInit | 【内部函数】端口初始化函数的总接口，此函数会调用以上4个函数 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16PinRead | 【外部函数】获取引脚的电平值 | 输入：  u16PinID：MUC\_enPinID枚举类型的引脚编号  \*u8PinValue：获取引脚结果的存储指针，引脚结果由此参数传出  返回值：  函数执行结果 |
| MCU\_u16PinWrite | 【外部函数】写入引脚的电平值 | 输入：  u16PinID：MUC\_enPinID枚举类型的引脚编号  u8PinValue：需要写入的期望电平值  返回值：  函数执行结果 |
| MCU\_u16PinDirectCfg | 【外部函数】引脚的数据方向控制 | 输入：  u16PinID：MUC\_enPinID枚举类型的引脚编号  u8InOut：引脚的期望数据方向，0=output，1=input  返回值：  函数执行结果 |
| MCU\_u16PinPullUpCfg | 【外部函数】引脚的上拉使能控制 | 输入：  u16PinID：MUC\_enPinID枚举类型的引脚编号  u8OnOff：引脚的期望上拉状态，0=off，1=on  返回值：  函数执行结果 |
| MCU\_u16PinPullDownCfg | 【外部函数】引脚的下拉使能控制 | 输入：  u16PinID：MUC\_enPinID枚举类型的引脚编号  u8OnOff：引脚的期望下拉状态，0=off，1=on  返回值：  函数执行结果 |
| MCU\_u16PinOpenDCfg | 【外部函数】引脚的漏级开路（OD）使能控制 | 输入：  u16PinID：MUC\_enPinID枚举类型的引脚编号  u8OnOff：引脚的期望OD状态，0=off，1=on  返回值：  函数执行结果 |
| MCU\_u16PinAltFunCfg | 【外部函数】引脚的功能选择控制 | 输入：  u16PinID：MUC\_enPinID枚举类型的引脚编号  AltFun：引脚的期望功能，MCU\_enPinAltFun 枚举类型  返回值：  函数执行结果 |

* 1. 注意事项

1. 端口的属性值在复位时底层代码会根据用户的MCU\_stPinUserCfg\_xPort进行配置，在引脚功能不复用的情况，用户不需要调用接口配置，直接使用配置值即可。
2. 普通情况下，用户配置完成MCU\_stPinUserCfg\_xPort以后，MCU\_u16PinRead，MCU\_u16PinWrite函数即可满足普通IO的使用，对于使用ALT功能的情况，引脚状态由相应的资源模块直接控制

# OSTM配置

* 1. 模块逻辑图



* 1. 模块特性

1. OSTM的源时钟有时钟模块MCU\_stClockCfg\_F1L进行配置，频率为CPU频率的一半
2. OSTM可以产生FE级别和EI级别的中断，本代码只使用EI级别的中断
3. 本代码只使用OSTM的周期定时功能，不使用free-run功能
   1. 配置参数说明

F1L\_USRCFG.c中OSTM配置项相关的信息如下：

配置结构体为MCU\_stOSTMCfg：

|  |  |
| --- | --- |
| 配置项 | 配置说明 |
| u32Interval\_Us | OSTM中断的时间设定，单位为us，当此设定值为0时，表示此定时器不启用  默认值 = 0，即不启用定时器  其他设定值即为启用以设定值（us）为定时间隔 |
| u32RunInRst | OSTM在资源初始化完成以后是否立即开始工作，  0 = 不立即工作，由用户调用相应接口控制  1 = 立即工作，用户不需要操作OSTM即开始计时  默认值 = 0，即不立即工作 |
| vpOSTM\_IRQ | OSTM定时中断的vpFun类型回调函数，用户可以配置相应的回调函数来完成相应的工作 |

* 1. 接口说明

OSTM相关的函数列表如下：

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_u16OSTMinit | 【内部函数】OSTM的模块初始化函数，如果用户设定了u32RunInRst不为0，则OSTM在执行完此函数以后就已经开始计时了 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16OSTMctrl | 【外部函数】OSTM运行状态控制，用于控制定时器的工作或者停止功能 | 输入：  OSTM\_ID：MCU\_enOstmID枚举类型的定时器编号  OnOff：期望的定时器运行状态，0 = 停止工作  1 = 开始工作  返回值：  函数执行结果 |
| MCU\_u16OSTMChgInterval | 【外部函数】OSTM运行中断时长控制，用于控制定时器的中断时间长度 | 输入：  OSTM\_ID：MCU\_enOstmID枚举类型的定时器编号  u32Interval\_Us：期望的定时器中断时间长度，单位为us，  1 < = 停止工作  other = 设定该期望值  返回值：  函数执行结果 |
| MCU\_vOSTM\_Isr | 【内部函数】OSTM运行中断接口，只由硬件中断调用，此函数会调用用户的回调函数 | 输入：  OSTM\_ID：MCU\_enOstmID枚举类型的定时器编号  返回值：  无 |

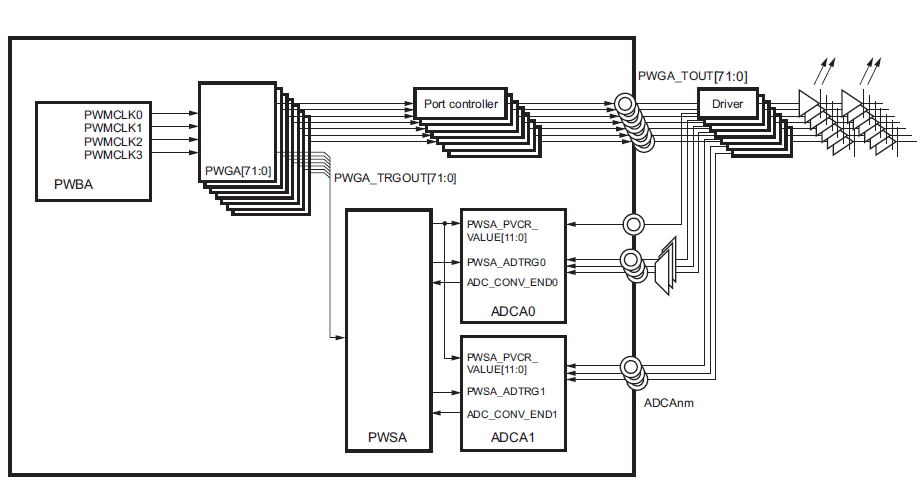
* 1. 注意事项

1. 用户配置的中断回调函数应该尽量短小，
2. 回调函数内是处于中断关闭状态的，所以不能在中断内等待某一个中断的处理请求
3. 相关接口的开启需要在F1L\_USRCFG.h文件内打开相应的宏定义

# PWGA配置

* 1. 模块逻辑图

模块的逻辑如下：



* 1. 模块特性

1. 本模块原本支持用ADC诊断PWGA的输出，但本代码不支持此功能
2. PWGA的输出频率是离散的点，具体的输出频率点见<F1L_PWGA频率计算.xlsx>
3. PWGA不使用中断功能
4. PWGA的输出频率只有4个，所有的PWGA输出的频率点只能4选一
5. 由于PWGA输出频率只能4选一，变更某个通道的频率可能会影响到其他共用频率的通道所以本代码不支持变更PWGA的频率改变
   1. 参数配置

F1L\_USRCFG.c中关于PWGA的配置项为MCU\_stPWGACfg，具体参数见下列表格：

|  |  |
| --- | --- |
| 参数项 | 参数说明 |
| Frequency | PWGA的频率点设定，4元素的数值，设定值见[F1L\_PWGA频率计算.xlsx](file:///D:\HVACproject\Project\ProjectCode\09_McuDrv\RH850\Document\F1L_PWGA频率计算.xlsx)，设定值为0xffffffff时表示不使用此频率设定  默认值 = 0xffffffff，即不使能频率点 |
| FrequencySelect | FrequencySelect用于选择设定频率点Frequency的标号，  此通道频率 = Frequency[FrequencySelect]的设定值  FrequencySelect取值范围[0,3]  默认值 = 255，即不使能此通道的PWGA |
| DefaultDuty | 设定此通道PWGA复位时的输出占空比  设定范围[0,1000]，  设定0 = duty 0%  设定1000 = duty 100%  默认值 = 0，即复位输出低电平 |

* 1. 接口说明

PWGA相关接口如下列表：

|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_u16PWGAinit | 【内部函数】PWGA模块的初始化函数，用于根据用户的配置进行模块的初始化 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16PWGASetDuty | 【外部函数】改变PWGA通道的输出占空比 | 输入：  u16PPG\_ID：MCU\_enPwgaID枚举类型的PWGA通道标号  u16Duty： 期望输出的占空比设定范围[0,1000]，  设定0 = duty 0%  设定1000 = duty 100%  返回值：  函数执行结果 |

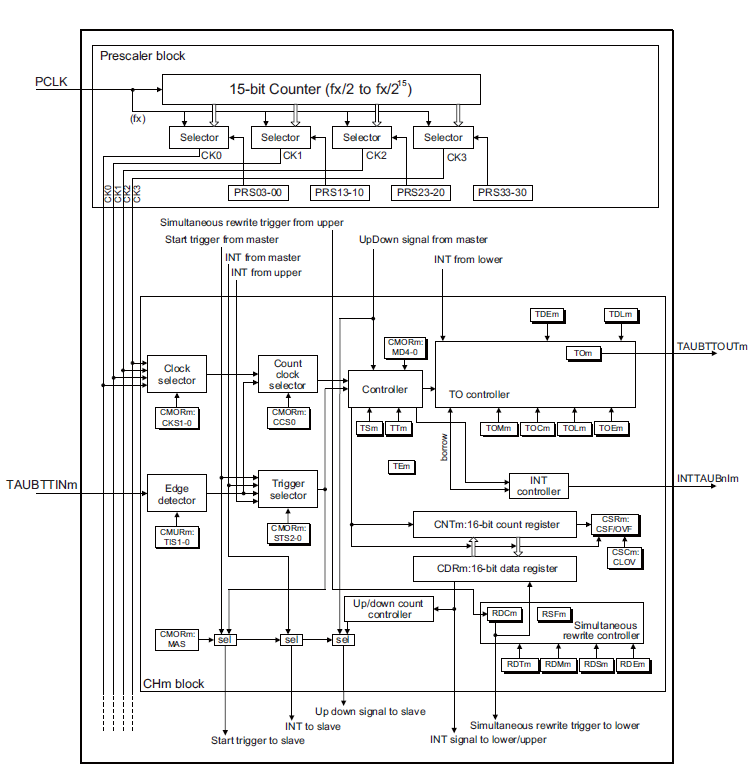
* 1. 注意事项

1. 配置项的填充值只能替换，不能增减
2. PWGA模块内部使用的时钟源时PER2时钟，具体设定请参考时钟模块

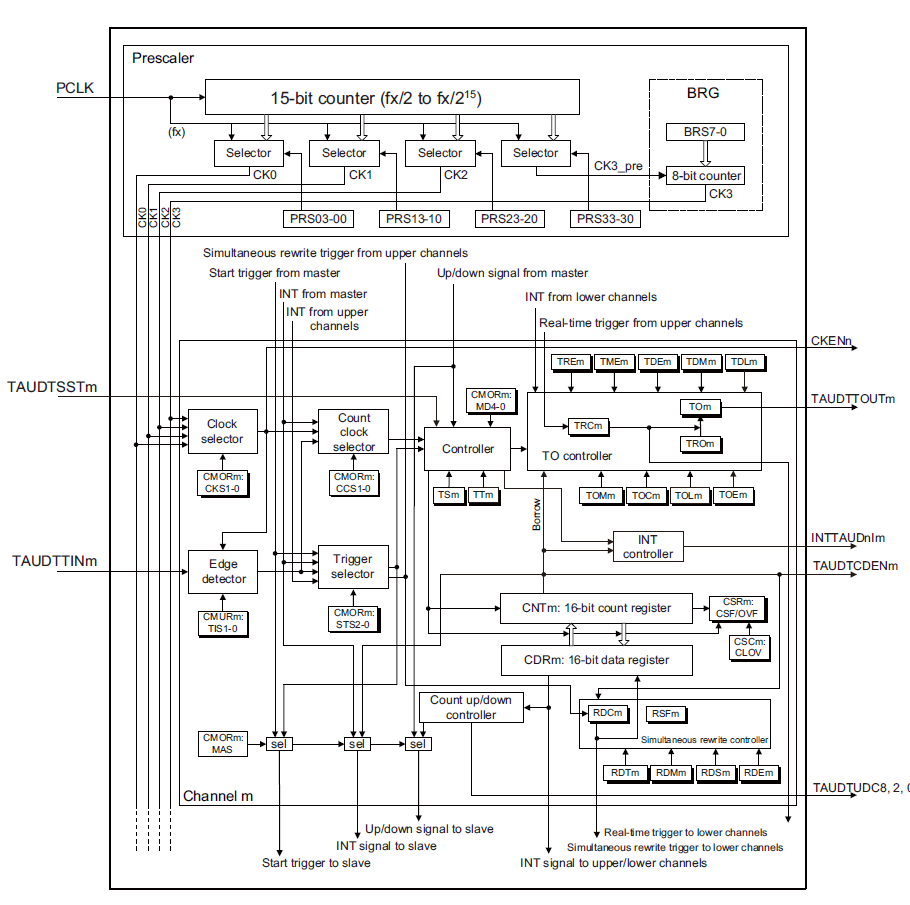
# TUAx配置

* 1. 模块逻辑图

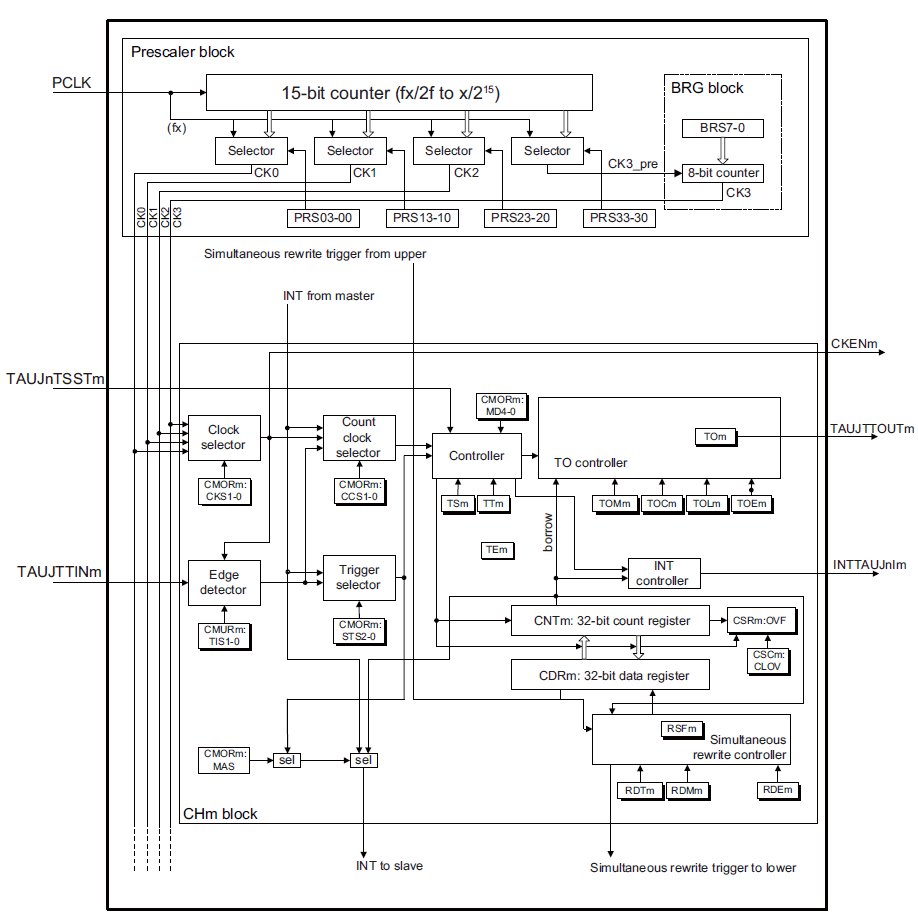
TUAB模块逻辑：



TAUD模块逻辑：



TAUJ模块逻辑图：



* 1. 模块特性

1. TAUB模块使用的时钟源是PER2
2. TAUD模块使用的时钟源是PER1
3. TAUJ0模块使用的时钟源是ATAUJ，在时钟配置模块进行配置
4. TAUJ1模块使用的时钟源是PER1
5. TAUB,TAUD模块可以组成多相的PWM信号，但是本代码没有支持此功能
6. 使用PPG模式时，只有偶数通道的TUAx可以做主节点，且主节点的作用范围只在本组内（小于ID 15的组内）
7. 当使用外部的电平触发时，中断请求位需要用户调用请求位清除的函数接口进行清除
   1. 参数配置

F1L\_USRCFG.c文件内关于TUAx的配置有两项：

1. 配置各组的时钟分频选择的配置：MCU\_stTAUxClkDivCfg
2. 各组内各个定时器通道的配置：MCU\_stTAUXCfg

下面对配置项进行说明：

MCU\_stTAUxClkDivCfg配置项：

|  |  |
| --- | --- |
| 配置项 | 参数说明 |
| MCU\_stTAUxClkDivCfg | 用于配置每一组TAUx的时钟源分频，uint32类型，请直接使用宏TAUxClk\_Div来定义分频0到分频3的数值，  分频取值范围[0,15]，  分频倍数=(2^设定值)，  即定时器实际使用的时钟频率 = 时钟源/(2^设定值) |
|  |  |

MCU\_stTAUXCfg配置项：

|  |  |
| --- | --- |
| 配置项 | 参数说明 |
| TimerMode | 定时器工作模式的选择，选择范围[0,17]，具体请参考配置宏 |
| ClkSelect | 定时器的时钟分频数选择，  实际值 = MCU\_stTAUxClkDivCfg[x][ClkSelect] |
| Counter | 定时器的计数属性值，具体请参考配置宏 |
| Condition | 定时器的触发属性值，具体请参考配置宏 |
| AutoRun | 配置定时器复位时候是否开始运行，具体请参考配置宏 |
| TimerCB | 定时器的中断回调函数，具体请参考配置宏 |

定时器的配置宏列表：

|  |  |
| --- | --- |
| 配置宏 | 配置宏说明 |
| #define TimerMode | 将timer设定为定时器工作模式：  配置参数：  ClkSel：选择时钟分频索引，范围[0,3]  interval\_us：定时器的中断时间，单位为us  RunInRest：配置定时器复位时是否开始工作，范围[0,1]，  0 = 由用户控制开启停止  1 = 模块初始化以后自动开始计时  CB：定时器的中断回调函数，vpFun类型 |
| #define PinTriggerTimer | 将timer设定为引脚输入即重新开始计时的定时器工作模式：  配置参数：  ClkSel：选择时钟分频索引，范围[0,3]  interval\_us：定时器的中断时间，单位为us  Trigger：重新计数的检测边沿，可以设定以下值：  #define TAUxTrige\_Failing 0  #define TAUxTrige\_Rising 1  #define TAUxTrige\_BothEdge 2  RunInRest：配置定时器复位时是否开始工作，范围[0,1]，  0 = 由用户控制开启停止  1 = 模块初始化以后自动开始计时  CB：定时器的中断回调函数，vpFun类型  【与定时器关联的引脚功能需要在port模块进行配置】 |
| #define FrequencyDivider | 将timer设定为分频工作模式：用于对输入信号的分频中断  配置参数：  divider：分频倍数，实际中断时间为设定值个数的输入秒冲  RunInRest：配置定时器复位时是否开始工作，范围[0,1]，  0 = 由用户控制开启停止  1 = 模块初始化以后自动开始计时  CB：定时器的中断回调函数，vpFun类型  【与定时器关联的引脚功能需要在port模块进行配置】 |
| #define CounterMode | 将timer设定为外部输入计数器工作模式：用于对输入信号的个数计数  配置参数：  ClkSel：选择时钟分频索引，范围[0,3]  counter：计数个数，检测到设定值个数的输入脉冲后进入中断  Trigger：输入脉冲的有效边沿，可以设定以下值：  #define TAUxTrige\_Failing 0  #define TAUxTrige\_Rising 1  #define TAUxTrige\_BothEdge 2  RunInRest：配置定时器复位时是否开始工作，范围[0,1]，  0 = 由用户控制开启停止  1 = 模块初始化以后自动开始计时  CB：定时器的中断回调函数，vpFun类型  【与定时器关联的引脚功能需要在port模块进行配置】 |
| #define PulseInterval | 将timer设定为外部输入信号的周期捕捉工作模式：用于捕捉输入信号的周期  配置参数：  ClkSel：选择时钟分频索引，范围[0,3]  counter：计数个数，检测到设定值个数的输入脉冲后进入中断  Trigger：输入脉冲的有效边沿，可以设定以下值：  #define TAUxTrige\_Failing 0  #define TAUxTrige\_Rising 1  RunInRest：配置定时器复位时是否开始工作，范围[0,1]，  0 = 由用户控制开启停止  1 = 模块初始化以后自动开始计时  CB：定时器的中断回调函数，vpFun类型  【与定时器关联的引脚功能需要在port模块进行配置】 |
| #define PulseWidth | 将timer设定为外部输入信号的宽度捕捉工作模式：用于捕捉输入信号的高电平或者低电平长度  配置参数：  ClkSel：选择时钟分频索引，范围[0,3]  Level：输入脉冲的有效电平，可以设定以下值：  #define TAUxTrige\_LowLevel 2  #define TAUxTrige\_HighLevel 3  RunInRest：配置定时器复位时是否开始工作，范围[0,1]，  0 = 由用户控制开启停止  1 = 模块初始化以后自动开始计时  CB：定时器的中断回调函数，vpFun类型  【与定时器关联的引脚功能需要在port模块进行配置】 |
| #define PpgMaster | 将timer设定为PPG模式的主通道模式，与从通道一起构成一路PPG输出  配置参数：  ClkSel：选择时钟分频索引，范围[0,3]  DefaultHZ：PPG的频率  【只能偶数的通道可以配置成主节点】 |
| #define PpgSlave | 将timer设定为PPG模式的从通道模式，与主通道一起构成一路PPG输出  配置参数：  DefaultDuty：PPG复位时输出的占空比  设定范围[0,1000]，  设定0 = duty 0%  设定1000 = duty 100%  【与PPG输出关联的引脚功能需要在port模块进行配置】  【PPG的输出信号在与从通道相关联的TAUXOx引脚上】 |

备注：

1. 由于定时器的一些功能比较偏门，适用的场合较少，所以没有对其功能进行测试和说明，具体细节可以参考spec的TAUx相关章节
2. 对于各个模式的具体工作细节，请参考spec章节
3. TAUJ模块的功能比TAUB，TAUD模块少，请在代码的配置后面关注该功能在TAUJ模块是否适用
   1. 接口说明

TAUx相关的接口说明如下表：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 参数说明 |
| MCU\_vTAUx\_Isr | 【内部函数】TAUx模块的中断服务函数，此函数只能由硬件中断调用 | 输入：  u32TAUxCh：MCU\_enTauxID枚举类型的timer标号  返回值：  无 |
| MCU\_vTAUx\_INTctrl | 【外部函数】TAUx模块的中断控制，用于控制timer的中断开启或者关闭 | 输入：  TauxID：MCU\_enTauxID枚举类型的timer标号  IntEnable：期望中断的状态  0 = 中断请求关闭  1 = 中断请求开启  返回值：  无 |
| MCU\_u32GetTAUxClkFrequency | 【外部函数】获取一个timer的运行时钟频率 | 输入：  TauxID：MCU\_enTauxID枚举类型的timer标号  返回值：  Timer的在使用时钟的频率 |
| MCU\_u16TAUxClkDivSetting | 【内部函数】TAUx模块设定所有timer组的时钟源分频，根据用户配置的MCU\_stTAUxClkDivCfg参数进行设置 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16TAUBD\_Init | 【内部函数】TAUB，TAUD模块的初始化函数接口 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16TAUJ\_Init | 【内部函数】TAUJ模块的初始化函数接口 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16TAUxInit | 【内部函数】TAUx模块的初始化函数接口，会调用以上三个接口函数 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16TAUxSetCounter | 【外部函数】更新timer内部的计数器的值，请关注计数器变更以后，不同分频，不同工作模式下的具体影响可能不一样 | 输入：  TauxID：MCU\_enTauxID枚举类型的timer标号  u32NewCounte：期望的timer计数值  返回值：  Timer的在使用时钟的频率 |
| MCU\_u16TAUxPpgGetMasterID | 【内部函数】获取PPG模式下与当前从通道相关联的主通道标号 | 输入：  u32SlaveChID：MCU\_enTauxID枚举类型的timer标号，必须为从通道标号  返回值：  与当前从通道关联的主通道标号 |
| MCU\_u16TAUxPpgSetHz | 【外部函数】更新PPG模式下输出的频率和占空比 | 输入：  u32SlaveChID：MCU\_enTauxID枚举类型的timer标号，必须为从通道标号  u32Hz：期望输出的频率，单位为Hz  u16Duty：期望输出的占空比，设定范围[0,1000]，  设定0 = duty 0%  设定1000 = duty 100% |
| MCU\_u16TAUxPpgSetDuty | 【外部函数】更新PPG模式下输出的占空比，频率属性维持不变 | 输入：  u32SlaveChID：MCU\_enTauxID枚举类型的timer标号，必须为从通道标号  u16Duty：期望输出的占空比，设定范围[0,1000]，  设定0 = duty 0%  设定1000 = duty 100% |
| MCU\_vTAUxRunCtrl | 【外部函数】TAUx运行控制的函数接口，用于控制timer运行状态 | 输入：  TauxID：MCU\_enTauxID枚举类型的timer标号，  OnOff：期望的运行状态，设定范围[0,1]，  设定0 = 停止  设定1 = 开始 |
| MCU\_u32TAUxGetCounter | 【外部函数】获取TAUx计数结果的函数接口，用于在周期捕捉，电平长度捕捉，计数器等模式下获取timer的结果，结果的属性是计数的个数 | 输入：  TauxID：MCU\_enTauxID枚举类型的timer标号，  返回值：  计数结果 |
| MCU\_u32TAUxGetTime | 【外部函数】获取TAUx计数结果的函数接口，用于在周期捕捉，电平长度捕捉等模式下获取timer的结果，结果的属性是计数的时间长度，单位为us | 输入：  TauxID：MCU\_enTauxID枚举类型的timer标号，  返回值：  计数的时间结果，单位为us |

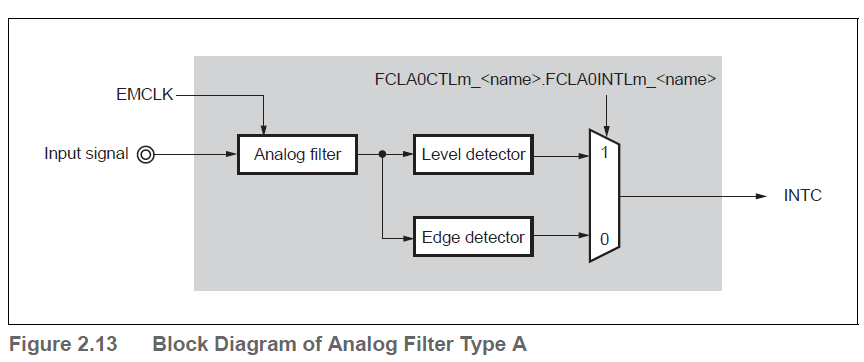
* 1. 注意事项

1. TAUx的一些功能没有进行测试，详情见测试文档
2. TAUJ的功能比TAUB，TAUD的少，详情见代码配置处的注释说明
3. 所有的配置填充项只能替代，不能删减

# EINT配置

* 1. 模块逻辑图

外部中断模块的内部逻辑图如下：



* 1. 模块特性

EINT模块没有特定的控制模块，只要将电平检测的设定和中断请求开启即可完成EINT请求

* 1. 参数配置

F1L\_USRCFG.c文件内关于EINT模块的设定参数结构体为MCU\_stEINTCfg，具体的配置细节如下：

|  |  |
| --- | --- |
| 配置参数 | 参数说明 |
| u16DetectCondition | EINT触发的条件，配置的条件如下：  #define RisingEdge 1u  #define FaillingEdge 2u  #define BothEdge 3u  #define LowLevel 4u  #define HighLevel 5u  #define EINT\_NotUse 0xFFFFu  默认值 = 0xFFFF，即不使能外部中断  【使用电平触发时，中断请求位需要软件清除】 |
| u16RunWhenReset | EINT在初始化完成以后是否开始工作：  0 = 用户控制器开始停止工作  1 = 初始化完成以后立即开始工作  默认值 = 0，即由用户控制 |
| vpINTservice | EINT中断服务回调函数，用于配置相应的用户服务函数，vpFun类型，MCU\_vNotExpINT代表中断服务函数不使能，用户可以根据需要填充函数名称  默认值 = MCU\_vNotExpINT，即中断服务部使能 |

* 1. 接口说明

EINT模块相关的接口列表如下：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 参数说明 |
| MCU\_u16EINTinit | 【内部函数】EINT模块初始化接口函数，根据用户配置的通道属性进行初始化EINT | 输入：  无  返回值：  函数执行结果 |
| MCU\_vEINTIsr | 【内部函数】EINT模块的ISR服务函数，此函数被硬件中断函数调用，并由此函数调用用户配置的中断函数 | 输入：  u32EINTid：MCU\_enEintID枚举类型的中断通道标号  返回值：  无 |
| MCU\_u16EINTctrl | 【外部函数】EINT模块的开始或者停止的控制函数 | 输入：  u32EINTid：MCU\_enEintID枚举类型的中断通道标号  OnOff：通道的期望运行状态  返回值：  函数执行结果 |
| MCU\_u16EINTchgIntCondition | 【外部函数】EINT模块的触发条件的控制函数 | 输入：  u32EINTid：MCU\_enEintID枚举类型的中断通道标号  u32NewTrigerCondition：MCU\_enEintTriggerID枚举类型的参数，为通道的期望触发条件  返回值：  函数执行结果 |
| MCU\_vEINTDeInit | 【外部函数】EINT模块的反初始化函数 | 输入：  无  返回值：  函数执行结果 |

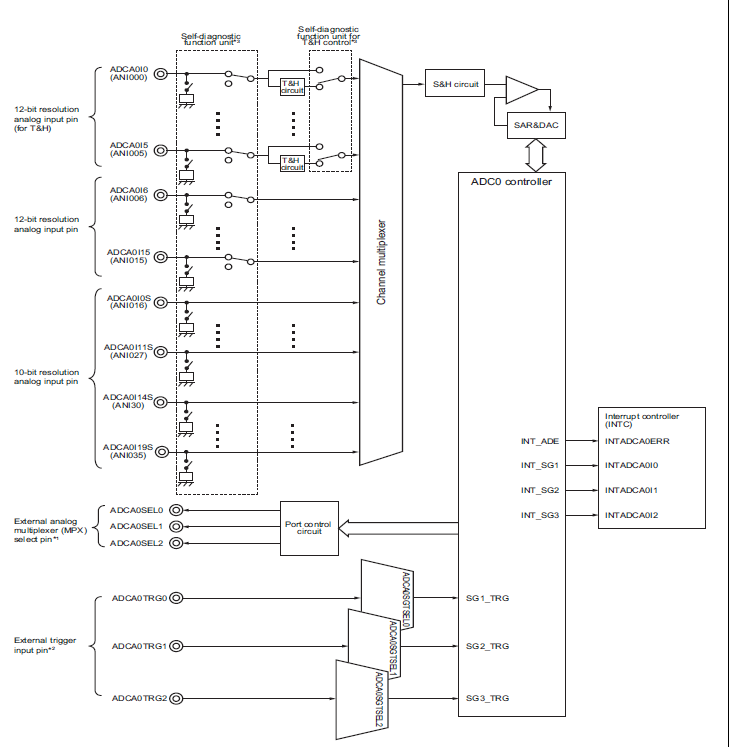
* 1. 注意事项

1. 中断相关的引脚功能由引脚配置模块进行配置到相应的功能项
2. 如果引脚的对应特殊功能是中断请求输入，则保持默认的配置即可

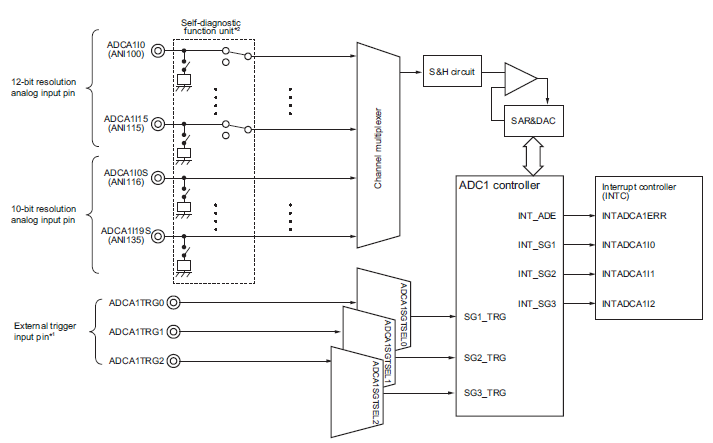
# ADC配置

* 1. 模块逻辑图

ADC0模块的内部理你图如下：



ADCA1的内部逻辑图如下：



* 1. 模块特性

1. ADC对应的引脚一部分处于特殊功能，此部分引脚功能不用配置，一部分的引脚处于复用功能上，对于此部分是需要进行配置的
2. 本模块支持10bitADC和12bitADC，但是本代码只支持10bitADC格式
3. 本模块支持ADC中断，但是本代码不使用期中断，只是将ADC的结果存储在相应的寄存器内，ADC由应用控制周期性触发ADC采用即可
4. 本模块支持多个通道分组，也支持虚拟通道对物理通道的映射，但是本代码只支持物理通道和虚拟通道的一一映射，且只使用一个分组，ADC采样时，不管ADC通道是否有使用都进行采样，应用层只需获取对应使用到的ADC通道的结果即可
5. 本模块支持ADC越界【超出设定高值或者低于设定低值时产生中断】，但是本模块不支持此功能
6. 本模块支持外部引脚触发，但是不代码不支持此功能
   1. 配置参数

F1L\_USRCFG.c内关于ADC模块的配置结构体为MCU\_stADCACfg，具体参数配置如下：

|  |  |
| --- | --- |
| 配置项 | 配置项说明 |
| u32ADCusingMask\_ADCA0\_P1 | ADCA0的0~31通道配置，对应的bit位设为1时代表启用该通道的ADC  默认值 = 0，即全部不启用 |
| u32ADCusingMask\_ADCA0\_P2 | ADCA0的32~63通道配置，对应的bit位设为1时代表启用该通道的ADC  默认值 = 0，即全部不启用 |
| u32ADCusingMask\_ADCA1\_P1 | ADCA1的0~31通道配置，对应的bit位设为1时代表启用该通道的ADC  默认值 = 0，即全部不启用 |
| u32ADCusingMask\_ADCA1\_P2 | ADCA1的32~63通道配置，对应的bit位设为1时代表启用该通道的ADC  默认值 = 0，即全部不启用 |

* 1. 接口说明

ADC模块的接口列表如下：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 参数说明 |
| MCU\_u16ADCinit | 【内部函数】ADC模块的初始化函数，根据用户配置对各个通道的ADC进行配置 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16ADCstart | 【外部函数】ADC模块采样开始的控制接口，煤调用一次则ADC对全部通道进行采样一次，完成以后自动停止，等待一下调用此函数 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16ADCstop | 【外部函数】ADC模块采样停止的控制接口，由于ADC采用开始一次全部通道采样后自动停止的模式，所以一般不调用此函数 | 输入：  无  返回值：  函数执行结果 |
| MCU\_vADCIsr | 【内部函数】ADC模块采样的中断函数，由于ADC不使用中断，此函数闲置 | 输入：  无  返回值：  无 |
| MCU\_u16ADCgetResault | 【外部函数】获取ADC采样通道的结果 | 输入值：  u32ChannelID：MCU\_enAdcaID枚举类型的通道标号  \*u16AD\_Resault：最终结果输出的指针，用于给调用者本次转换的结果  返回值：  函数执行结果 |
| MCU\_vADCDeInit | 【外部函数】ADC反初始化函数 | 输入值：  无  返回值：  无 |

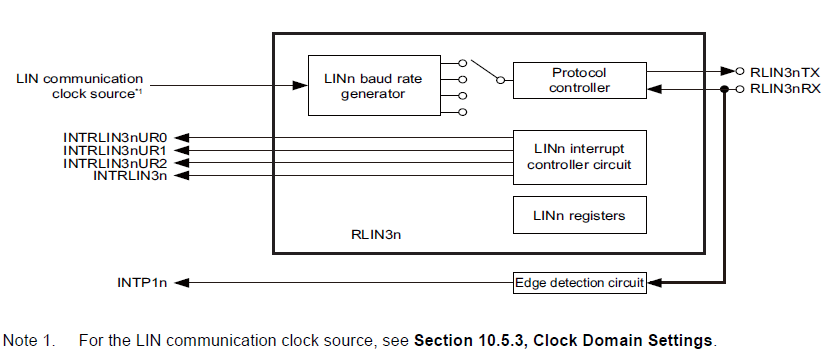
* 1. 注意事项

1. MCU\_u16ADCstart需要周期调用以确保ADC循环进行采样
2. MCU\_u16ADCgetResault获取的结果是上一次采样结果的缓存值

# RLIN3\_UART配置

* 1. 模块逻辑图

LIN3模块的内部逻辑图如下：



* 1. 模块特性

1. RLIN3模块可以用作LIN主节点和从点，也可以用作UART节点，由于LIN部分均购买，所以此部分代码只支持UART模式，
2. RX引脚与EINT引脚功能内部相连，所以在休眠时，该引脚选择不同复用功能时可以实现EINT或者LIN/UART不同功能的切换
   1. 模块配置参数

F1L\_USRCFG.c内关于UART的配置结构体为MCU\_stRLIN3UsrCfg，具体的配置项说明如下：

|  |  |
| --- | --- |
| 配置名称 | 配置项说明 |
| u16BaudRate | 波特率设定，0代表该通道不启用  默认值 = 0，即不开启通道 |
| u16Msb | 设定串口的数据方向，设定范围[0,1]  0 = LSB  1 = MSB  默认值 = 0，即LSB（低位先传输） |
| u16StopBit | 设定停止位的个数，设定范围[0,1]  0 = 1bit停止位  1 = 2bit停止位  默认值 = 0，即使用一个停止位 |
| u16Parity | 设定串口的校验方式，可以设定为以下的值：  #define RLin\_ParityNone 0u  #define RLin\_ParityEven 1u  #define RLin\_ParityZero 2u  #define RLin\_ParityOdd 3u  默认值 = 0，不使用校验位 |
| u16Rx\_InversionOutput | 设定接收时需不需要反向接收，设定范围[0,1]  0 = 接收时按正常电平发送，0=低电平，1=高电平  1 = 接收时按反向电平发送，0=高电平，1=低电平  默认值 = 0，即按照正常电平接收 |
| u16Tx\_InversionInput | 设定发送时需不需要反向发送，设定范围[0,1]  0 = 发送时按正常电平发送，0=低电平，1=高电平  1 = 发送时按反向电平发送，0=高电平，1=低电平  默认值 = 0，即按照正常电平发送 |
| vpSendFinish | 设定发送完成时的回调函数，vpFun类型，在发送完客户设定的数据长度以后，会回调此函数，MCU\_vNotExpINT填充代表不使用此中断  默认值 = MCU\_vNotExpINT，即不使用此中断 |
| vu8RecCallBack | 设定接收一个单元数据时的回调函数，vpFun\_u8类型，形参即为接收到的数据，在接收到数据以后，会回调此函数，MCU\_vNotExpINT\_u8填充代表不使用此中断  默认值 = MCU\_vNotExpINT\_u8，即不使用此中断 |
| vu8ErrCallBack | 设定接收错误的回调函数，vpFun\_u8类型，形参即为错误状态，在接收数据发生错误以后，会回调此函数，MCU\_vNotExpINT\_u8填充代表不使用此中断  默认值 = MCU\_vNotExpINT\_u8，即不使用此中断 |

* 1. 模块接口

UART相关的接口说明如下：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 参数说明 |
| MCU\_u16RLIN3Uart\_Init | 【内部函数】RINL3模块的初始化接口，根据用户的配置对相关通道进行初始化 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16RLIN3UartSend | 【外部函数】使用UART模式对外发生数据 | 输入：  u16channel：MCU\_enRlin3UartID枚举类型的通道标号  pu8DataHead：uint8 类型的指针，指向简要发送的数据缓存的地址  u16SendLen：期望发送的数据长度  返回值：  函数执行结果 |
| MCU\_u16RLIN3UartCtrl | 【外部函数】UART模式收发状态的控制 | 输入：  u16channel：MCU\_enRlin3UartID枚举类型的通道标号  u8SendEnable：控制该通道是否允许发送，设定范围[0,1]，  0 = 禁止发送  1 = 允许发送  u8RecEnable：控制该通道是否允许接收，设定范围[0,1]  0 = 禁止接收  1 = 允许接收  返回值：  函数执行结果 |
| MCU\_vRLIN3UartIsr\_Tx | 【内部函数】UART模式发送中断函数，此函数被硬件中断函数调用，发送完成既定的数据长度以后，会调用用户配置的发送完成回调函数 | 输入：  u16channel：MCU\_enRlin3UartID枚举类型的通道标号  返回值：  无 |
| MCU\_vRLIN3UartIsr\_Rx | 【内部函数】UART模式接收中断函数，此函数被硬件中断函数调用，且接收的数据以形参的形式传入用户配置的接收回调函数， 如果接收过程出现错误，则会先调用错误回调函数 | 输入：  u16channel：MCU\_enRlin3UartID枚举类型的通道标号  返回值：  无 |
| MCU\_vRLIN3Uart\_Deinit | 【内部函数】UART模式反初始化函数 | 输入：  无  返回值：  无 |

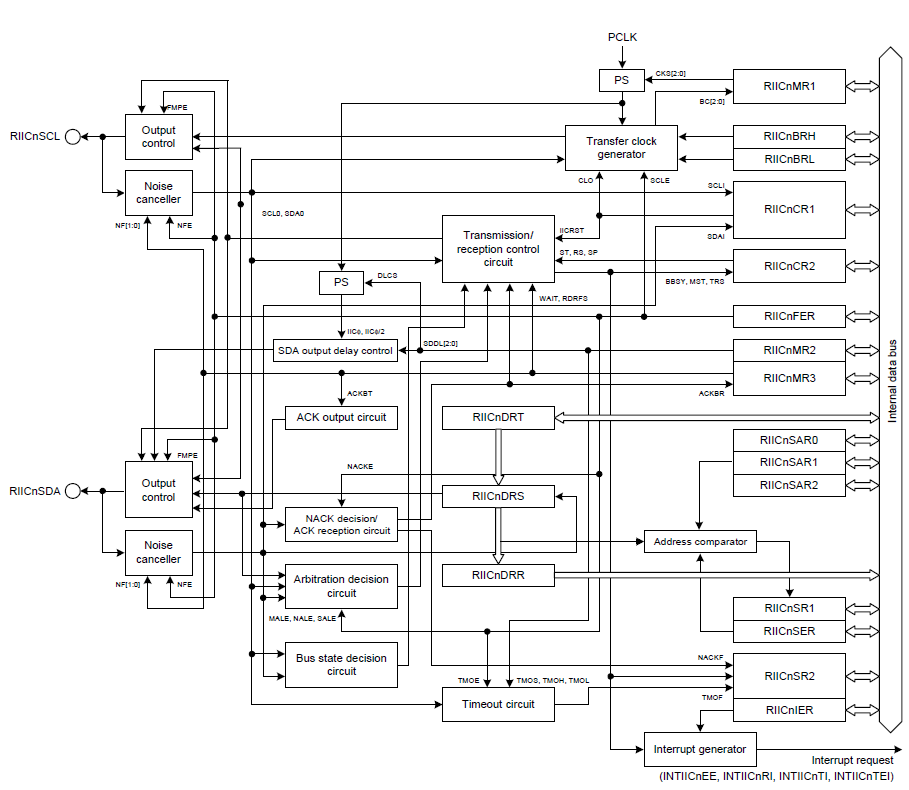
* 1. 注意事项

1. RLIN3模块的时钟源配置由时钟模块进行，细节请参考时钟模块
2. 此模块的时钟源的频率范围4MHz到40MHz，设定值不能超出此范围
3. 本代码使用的buff方式，把buff内全部的数据发送完成才进入一次中断，以减少发送中断的进入次数，

# IIC配置

* 1. 模块逻辑图

IIC模块的内部逻辑图如下：



* 1. 模块特性

1. 起始条件和结束条件由寄存器控制，软件不能控制引脚
2. 起始条件只能在总线空闲的时候产生
3. Re-start条件只能在总线忙的时候产生
4. 总线自己发生开始或者结束条件时，自己会检测，当检测到起始条件时，总线会进入忙状态，当检测到结束条件时，会进入空闲状态
5. 本模块支持主节点方式和从节点方式，但是本代码只支持组节点方式
   1. 参数配置

F1L\_USRCFG.c中关于IIC的配置结构体为MCU\_tstRIICCfg，具体的参数配置细节如下表：

|  |  |
| --- | --- |
| 配置项名称 | 参数说明 |
| u32TansferSpeed | 配置IIC通道的传输速率bps，0代表不启用该来IIC  默认值 = 0，即不启用IIC |
| stSlaveCB | 用于配置一个从节点的属性，因为一个主节点IIC上可能配备MCU\_mI2CMaxSlaveInMaster个从节点，他下面的配置项则属于相应的从节点 |
| u32SlaveAddr | 从节点的IIC地址，7bit长度，具体配置请参考相应的节点说明书 |
| vpSendfinishCB\_BuffEmpty | 发送缓存空闲的中断回调函数，vpFun类型，当发送完成了既定的数据长度任务且缓存空闲时，会调用此函数 |
| vpSendfinishCB | 发送数据完成的中断回调函数，vpFun类型，当发送完成了既定的数据长度任务时，会调用此函数 |
| vpRecFinishCB | 接收数据完成的中断回调函数，vpFun类型，当接收完成了既定的数据长度任务时，会调用此函数 |
| vu8ErrCB | 收发数据发生错误时的回调函数，vpFun\_u8类型，形参即为错误代码，当发生错误时，会调用此函数 |

* 1. 接口说明

IIC模块的相关接口列表如下：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 参数说明 |
| MCU\_u16IICinit | 【内部函数】IIC模块的初始化函数接口，根据用户的配置进行初始化设定 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16IIC\_MasterStartCondition | 【内部函数】用于触发起始条件 | 输入：  u8I2C\_masterID：MCU\_enRiicID枚举类型的IIC主节点标号  返回值：  函数执行结果 |
| MCU\_u16IIC\_MasterStopCondition | 【内部函数】用于触发结束条件 | 输入：  u8I2C\_masterID：MCU\_enRiicID枚举类型的IIC主节点标号  返回值：  函数执行结果 |
| MCU\_u16IIC\_MasterReStartCondition | 【外部函数】用于触发re-start条件，只能在IIC忙时调用 | 输入：  u8I2C\_masterID：MCU\_enRiicID枚举类型的IIC主节点标号  返回值：  函数执行结果 |
| MCU\_u8IIC\_GetSlaveIndex | 【内部函数】用于获取当前从节点所在主节点上的索引 | 输入：  u8I2C\_masterID：MCU\_enRiicID枚举类型的IIC主节点标号  u8SalveAddr：需要查找的从节点地址  返回值：  从节点在主节点网路上的索引 |
| MCU\_u16IIC\_MasterWrite | 【外部函数】主节点对外发送数据 | 输入：  u8I2C\_masterID：MCU\_enRiicID枚举类型的IIC主节点标号  u8SalveAddr：需要写入的对象的从节点地址  pDataHead：需要发送数据的头地址  u16Lenth：需要发送的数据长度，单位为byte  返回值：  函数执行结果  【本函数已经包含了start条件】 |
| MCU\_u16IIC\_MasterRead | 【外部函数】主节点从从节点读取数据 | 输入：  u8I2C\_masterID：MCU\_enRiicID枚举类型的IIC主节点标号  u8SalveAddr：需要读取的对象的的从节点地址  pDataHead：存储数据的头地址  u16Lenth：需要读取的数据长度，单位为byte  返回值：  函数执行结果  【本函数已经包含了start条件】 |
| MCU\_vRIIC\_Isr\_WriteByteFinish | 【内部函数】主节点发送完成中断，此函数由硬件中断ISR调用，且此函数会调用用户的回调函数 | 输入：  u16Ch：MCU\_enRiicID枚举类型的IIC主节点标号u32BufferEmpty\_Req：用于区分是缓存空闲的中断还是发送完成的中断，  0 = 发送完成中断  1 = 缓存空闲中断  返回值：  无  【在完成所有的既定数据长度发送时，本函数已经包含了stop条件】 |
| MCU\_vRIIC\_Isr\_ReadByteFinish | 【内部函数】主节点接收中断，此函数由硬件中断ISR调用，且此函数会调用用户的回调函数 | 输入：  u16Ch：MCU\_enRiicID枚举类型的IIC主节点标号  返回值：  无  【在完成所有的既定数据长度发送时，本函数已经包含了stop条件】 |
| MCU\_vRIIC\_Isr\_Err | 【内部函数】IIC错误中断，此函数由硬件中断ISR调用，且此函数会调用用户的回调函数 | 输入：  u16Ch：MCU\_enRiicID枚举类型的IIC主节点标号  返回值：  无  【在完成所有的既定数据长度发送时，本函数已经包含了stop条件】 |
| MCU\_vIICdeinit | 【外部函数】IIC模块的反初始化函数 | 输入：  无  返回值：  无 |

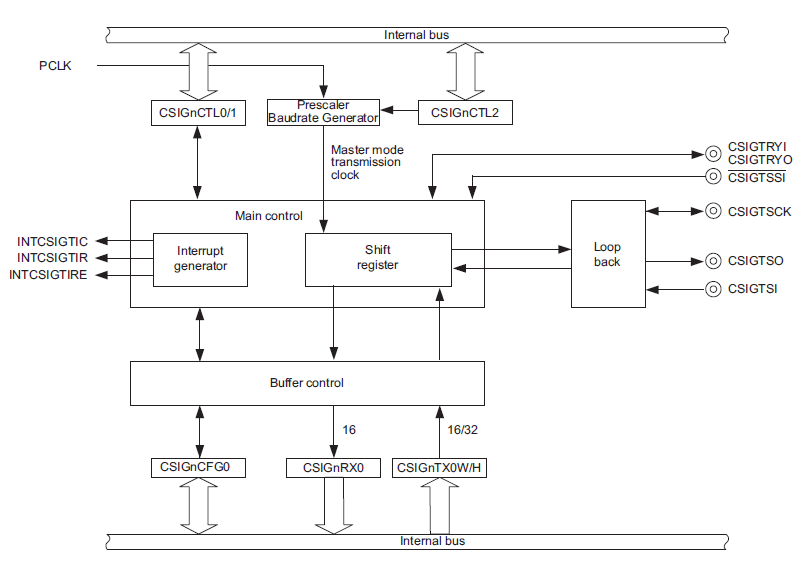
* 1. 注意事项

1. 本模块的SLK，SDA高低电平比例支持调节，但是本代码不支持此功能

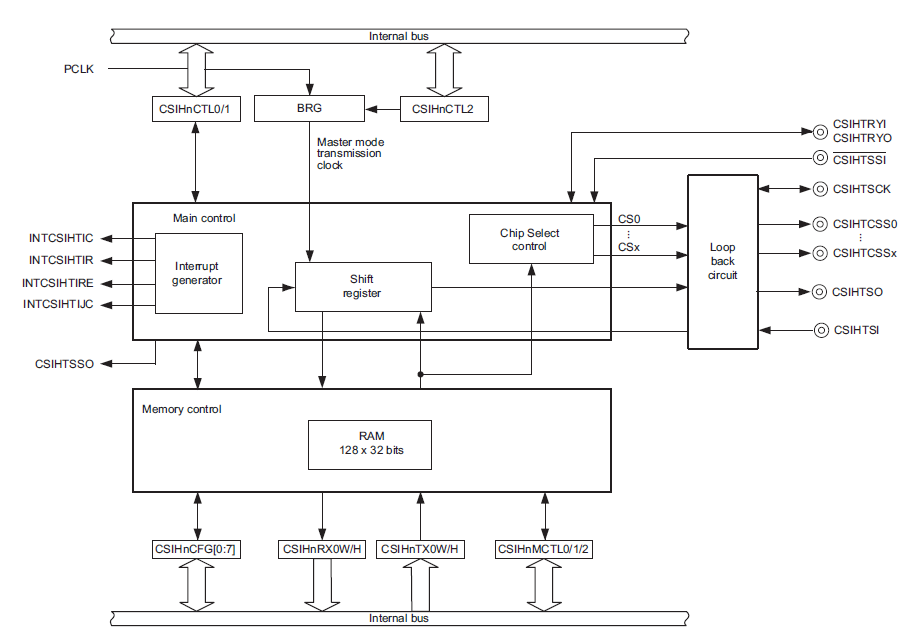
# CSIx配置

* 1. 模块逻辑图

本模块包含CSIH和CSIG模块，CSIG模块的逻辑图如下：



CSIH模块的内部逻辑图如下：



* 1. 模块特性

1. CSIH模块硬件支持对从节点的选择，但是本代码不支持硬件选取功能，只支持有函数接口控制的从节点
2. CSIH,CSIG模块支持握手协议，但是本代码不支持
3. CSIH,CSIG模块支持主从节点模式，但是本代码只支持主节点模式
4. CSIH支持AOTUSAR的JOB功能，但是本代码不支持
5. 本代码只支持普通模式下的SPI操作，其他复杂的功能均不支持
6. CSIH，CSIG支持数据长度8到16bit，但是本代码只支持16bit数据长度
   1. 模块参数配置

F1L\_USRCFG.c中关于CSIH，CSIG的配置项分别为MCU\_tstCSIHCfg，MCU\_tstCSIGCfg，具体的说明如下表：

|  |  |
| --- | --- |
| 配置名称 | 配置项说明 |
| Bps | 配置数据传输的速度，bps，0代表不启用该总线  默认值 = 0，即不开启总线 |
| LSB\_shift | 配置数据传输方式是LSB还是MSB  0 = MSB  1 = LSB  默认值为0，即MSB |
| ClkType | 配置时钟相位，配置范围[0,3]，代表的相位如下：  #define SpiClkType0 0  #define SpiClkType1 1  #define SpiClkType2 2  #define SpiClkType3 3    默认值 = 0，区第0个相位 |
| HankShakeEnable | 握手协议，本代码不支持，请设定其为0 |
| SlaveMode | 从节点模式，本代码不支持，请设定其为0 |
| Parity | 校验和的类型，可以配置如下：  #define SpiParityNone 0u---不使用校验  #define SpiParityZero 1u---零校验  #define SpiParityOdd 2u---奇校验  #define SpiParityEven 3u---偶校验  默认值 = 0，即不使用校验 |
| SendFinshCB | 数据发送完成中断，当总线完成了既定数据长度的发送任务时，会调用此回调函数，vpFun类型，MCU\_vNotExpINT代表不使用该中断  默认值 = MCU\_vNotExpINT，即不开启中断请求 |
| vpFun | 数据接收完成中断，当总线完成了既定数据长度的接收任务时，会调用此回调函数，vpFun\_u16类型，MCU\_vNotExpINT代表不使用该中断  默认值 = MCU\_vNotExpINT，即不开启中断请求 |
| ErrCB | 数据接收时的错误中断，当总线接收数据过程出现错误时，会调用此回调函数，形参即为错误代码，vpFun类型，MCU\_vNotExpINT\_u16代表不使用该中断  默认值 = MCU\_vNotExpINT\_u16，即不开启中断请求 |

* 1. 模块接口说明

下面以CSIH模块的接口做说明，CSIG模块的功能与此一样，具体见下表：

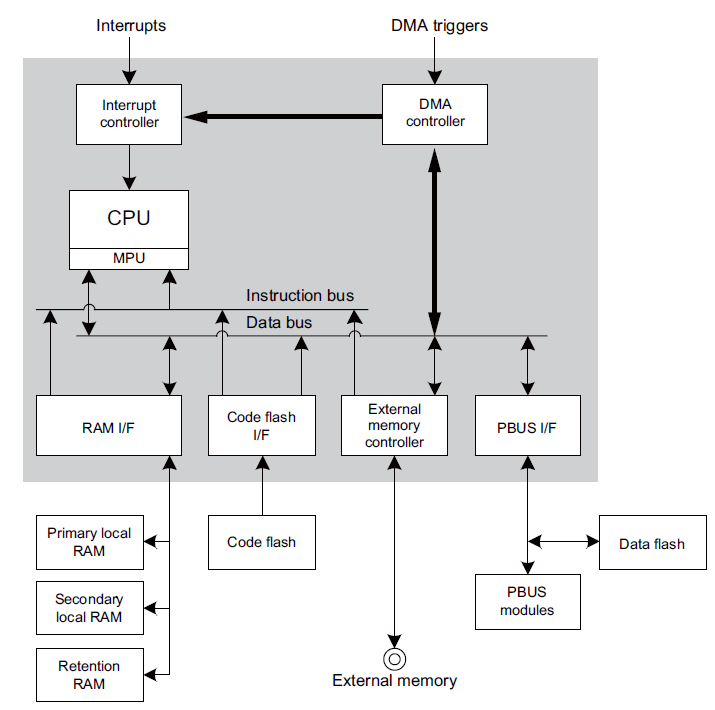
|  |  |  |
| --- | --- | --- |
| 接口名字 | 接口说明 | 参数说明 |
| MCU\_u16CSIHinit | 【内部函数】CSIH模块的初始化函数接口，根据用户的配置进行总线设定 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16CSIH\_MasterSynchRW | 【外部函数】CSIH模块对外设进行同步读写的函数接口，向外设发送一个单元的数据同时读取一个单元数据 | 输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  pSendDataHead：需要发送的数据的头地址，uint16类型的指针  pRecDataHead：存储接收到数据的头地址，uint6类型的指针  u16Lenth\_Byte： 需要发送/接收的数据长度，单位为byte  返回值：  函数执行结果 |
| MCU\_u16CSIH\_MasterWrite | 【外部函数】CSIH模块对外设进行写入数据的函数接口， | 输入：  输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  pSendDataHead：需要写入的数据的头地址，uint16类型的指针  u16Lenth\_Byte： 需要写入的数据长度，单位为byte  返回值：  函数执行结果 |
| MCU\_u16CSIH\_MasterRead | 【外部函数】CSIH模块对外设进行同步读取数据的函数接口， | 输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  pRecDataHead：存储接收到数据的头地址，uint6类型的指针  u16Lenth\_Byte： 需要发送/接收的数据长度，单位为byte  返回值：  函数执行结果 |
| MCU\_vCSIH\_Isr\_WriteUnitFinish | 【内部函数】发送数据单元的中断服务函数，此函数被硬件ISR调用，再完成所有的发送任务以后会调用用户的发送完成回调函数 | 输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  返回值：  无 |
| MCU\_vCSIH\_Isr\_ReadUnitFinsh | 【内部函数】接收数据单元的中断服务函数，此函数被硬件ISR调用，再完成所有的接收任务以后会调用用户的发送完成回调函数 | 输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  返回值：  无 |
| MCU\_vCSIH\_Isr\_Err | 【内部函数】接收数据单元的错误中断服务函数，当接收发生错误时会被硬件ISR调用， 且此函数会调用用户配置的回调函数 | 输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  返回值：  无 |
| MCU\_vCSIH\_Isr\_JobFinish | 【内部函数】JOB中断函数，被硬件中断调用，本代码不支持此功能 | 输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  返回值：  无 |
| MCU\_vCSIHJobLastData | 【内部函数】JOB中最后一个单元的数据发送的钩子程序，在JOB完成前的最后一个单元时，在此函数中处理下一个JOB的准备事宜，本代码不支持此功能 | 输入：  u32CsihID：MCU\_enCsihID枚举类型的通道标号  返回值：  无 |
| MCU\_vCSIHdeinit | 【外部函数】CSIH模块的反初始化函数 | 输入：  无  返回值：  无 |

* 1. 注意事项

1. 由于除法的关系，bps的实际值可能会比设定值偏大，需要在集成过程中关注他们的误差，适当调整bps值以达到期望的速率
2. 在主模式下，输出的最高频率为10MHz且不超过CPU时钟的1/4，从模式下最高频率为5MHz且不超过CPU时钟的1/6.
3. 本代码不支持IO缓存内存模式

# DMA配置

* 1. 模块内部逻辑图



* 1. 模块特性

1. 本模块共有16个通道，优先级顺序通道标号越低的优先级越高
2. 16个通道被分成两个部分，0到7位第一部分，8到15位第二部分，各个部分的触发源是有限制的，第一部分的触发源只能使用第一部分的通道，第二部分的触发源只能使用第二部分的通道，他们不能越过相关的通道编号进行使用
3. 数据格式支持8bit，16bit和32bit
4. 数据搬运过程只支持LSB格式
5. DMA每次使用前都要进行初始化，因为寄存器内的值每次都会被更新掉
6. DMA只负责数据的搬运工作，数据的处理，操作细节需要额外进行，如IIC，CSIx模块的具体数据操作，都由相关模块进行处理
7. DMA使能期间，如果触发条件的中断请求开启，则同样会响应该请求，所以DMA工作期间应该关闭触发条件的中断请求，在DMA完成了数据传送以后在软件开启该请求，以节省CPU的时间
   1. 配置参数

F1L\_USRCFG.c文件中关于DMA的配置结构体为MCU\_stDMACfg，各个通道具体的配置参数如下：

|  |  |
| --- | --- |
| 配置项名称 | 配置项说明 |
| u32SourceAddr | 数据源的头地址  默认值 = 0，即不启用 |
| u32DestAddr | 数据搬运的目标地址  默认值 = 0，即不启用 |
| u16AddrChgType | DMA完成一个单元数据以后地址变更的类型，可以配置如下：  #define Source\_Increment\_Dest\_Increment 0  源地址和目标地址都是往上增长，如用于数据一对一copy  #define Source\_Increment\_Dest\_Fix 1  源地址增长而目标地址固定，如用于RAM到寄存器  #define Source\_Fix\_Dest\_Increment 4  源地址固定而目标地址增长，如用于寄存器到RAM  #define Source\_Fix\_Dest\_Fix 5  源地址和目标地址都固定，如用于寄存器到寄存器  默认值 = 0， |
| u16DataUnitSize | DMA搬运的数据的单元位长度，可以配置为：  #define DMA\_Uint\_8bit 0 即按8bit长度进行搬运  #define DMA\_Uint\_16bit 1即按16bit长度进行搬运  #define DMA\_Uint\_32bit 2即按32bit长度进行搬运  默认值 = 0， |
| u16DataLenth | DMA搬运的总数据长度，单位为byte：  默认值 = 0， |
| u16InterruptTrigID | DMA使能的触发条件，分为第一组和第二组的触发源配置，不能串组进行。具体的触发源可选列表参见spec的page342描述  默认值 = 0， |
| vpINTservice | DMA搬运完成以后的回调函数，vpFun类型，MCU\_vNotExpINT代表不启用此中断，当DMA完成既定的传输任务以后会调用此函数，  默认值 = MCU\_vNotExpINT，即不开启此中断 |

* 1. 接口说明

DMA模块的相关接口如下列表：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 接口说明 | 参数说明 |
| MCU\_u16DMAsetup | 【外部函数】用于用户自己设定使用某一DMA通道的具体设定 | 输入：  stDMAsetup：MCU\_tstDMA\_SetUp结构体类型的参数设定，具体元素如下：  u32DMAChannel：需要使用到的DMA通道标号，设定值为MCU\_enDmaID枚举内的值  其他相同的元素配置参数见上一章节  返回值：  函数执行结果 |
| MCU\_u16DMAinit | 【内部函数】DMA模块的初始化函数 | 输入：  无  返回值：  函数执行结果 |
| MCU\_u16DMADefaultSetup | 【外部函数】对选定的通道使用用户配置的默认值进行设定 | 输入：  u32DMAchn：  返回值：需要使用默认值的DMA通道标号，设定值为MCU\_enDmaID枚举内的值  返回值：  函数执行结果 |
| MCU\_vDMAIsr | 【内部函数】DMA的完成中断函数，当完成既定长度的数据传输以后，此函数被硬件中断调用，且会调用用户配置的回调函数 | 输入：  u32DMAchn：  返回值：中断申请的DMA通道标号，设定值为MCU\_enDmaID枚举内的值  返回值：  函数执行结果 |
| MCU\_vDMADeInit | 【外部函数】DMA模块的反初始化函数 | 输入：  无  返回值：  无 |

* 1. 注意事项

1. DMA的设定每一次都要调用，因为他的寄存器值每个传输过程都被更新掉
2. 牵涉到应用模块的具体设定均由应用模块设定，如使用CSI中断时，CSI的发送接收接口的长度参数设定为1即可，而DMA的设定需要根据实际的数据长度进行设定，因为CSI的中断只有在所有的DMA数据传输完成以后进入一次而已。

# MCU内存规划文件

F1L的内存分布见前面部分，分别为：

* 1. Primary Local RAM和retention RAM

此部分在CPU复位时，Primary Local RAM数据会丢失，同时在F1L\_STARTUP.c中的下列代码会对此部分区域进行软件清零的动作：

--------------------------- Clear local RAM data to 0x00000000 at reset\_up

mov \_\_\_ghs\_ramstart, r6 -- start of local main RAM

mov \_\_\_ghs\_ramend, r7 -- end of local main RAM

mov r0, r1

1:

st.dw r0, 0[r6]

addi 8, r6, r6

cmp r7, r6

bl 1b

retention RAM的数据在软件复位时，硬件上RAM的数据是可以保留了，为安全起见，F1L\_STARTUP.c中会用软件对其进行清零的动作，代码如下：

--------------------------- Clear retention RAM data to 0x00000000 at reset\_up

mov \_\_\_ghs\_rramstart, r6 -- start of local Retention RAM

mov \_\_\_ghs\_rramend, r7 -- end of local Retention RAM

mov r0, r1

2:

st.dw r0, 0[r6]

addi 8, r6, r6

cmp r7, r6

bl 2b

在retention RAM的最后面的位置上，设定了2K的RAM不清零，以备代码中使用复位时的数据存留，具体的细节请参考\_F1L\_Map.ld文件的MEMORY段的规划

* 1. Stack的配置

Stack的配置见\_F1L\_Map.ld文件的.stack段，align(4) pad(0x0800)代表以4byte的数据对齐，size分配的大小时0x0800 byte，可以根据实际需求配置stack的大小

* 1. RAM大致分配情况

RAM区间的大致分配情况为：

RAM\_0：装载RAMCODE，带初始值的data，stack

RAM\_1：装载无初值的data，FDL段的data

RAM\_NotClear：装载内容由用户指定，如

#pragma ghs section bss = “. Not\_Clear\_RAM”

/\*用户数据 ===不带初值的全局变量\*/

#pragma ghs section bss = default

* 1. ROM区间分配

1. 不带BootLoader区间的ROM分配，默认的分配顺序是系统向量表，EI级别的table表，const数据，RAM中初值数据的镜像，最后堆叠执行代码，同时宏定义APP\_CODE\_MAP\_START设定必须为0，链接文件使用\_F1L\_Map\_NotBootLoader.ld
2. 带BootLoader区间的ROM分配，默认如上，只是从0地址到0xC200的区间分配给BootLoader层，链接文件使用\_F1L\_Map\_WithBootLoader.ld，宏定义APP\_CODE\_MAP\_START设定值必须为链接文件中iROM\_Boot section的结束地址，
   1. 注意事项
3. ROM的区间与地址范围必须按照相应的MCU类型做相应的修改，以保证代码分配在规定范围内
4. RAM0，RAM1的区间与地址范围必须按照相应的MCU类型做相应的修改，以保证代码分配在规定范围内，以上两项的范围值请参考MCU第一章节或者spec
5. Stack的范围可以根据相应的需求修改，pack命令内的参数即代表stack的分配大小

# 工程集成过程

工程的集成基本步骤如下：

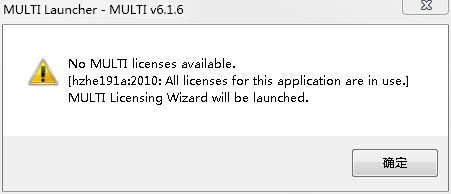
1. 在MCU\_CFG.h文件中配置MCU\_UsingType，确认需要使用的MCU型号，COMPILER\_TYPE选取相应的编译器
2. 在F1L\_USRCFG.h文件中，开启相应的资源模块和相应的接口，详细介绍参见第三章内容
3. 在F1L\_USRCFG.c文件中，对相应的资源模块进行具体的参数配置，根据具体项目的应用场景进行具体的配置，各个资源模块的配置细节请参考此文档前面内容
4. 根据MCU\_UsingType的配置MCU的内存信息，包括ROM，RAM0，RAM1，stack大小等
5. 建立编译工程：

**非常重要：对于MCU底层的集成，只要加入MCU.c文件和包含MCU\_CFG.h目录路径即可！**

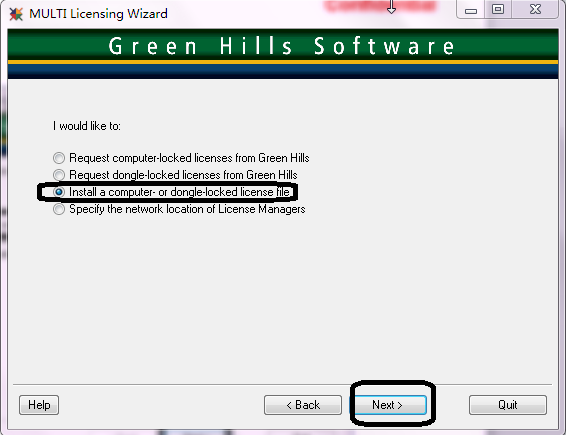
5.1 开启IDE软件：

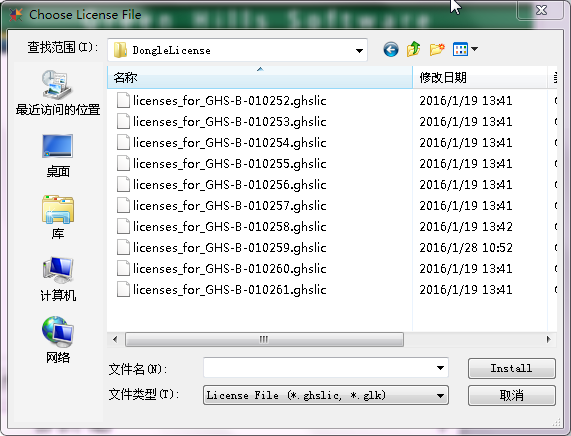


出现以下画面为缺少license：否则直接开启进入5.2



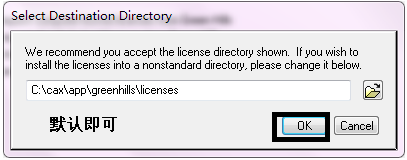
选择安装licence





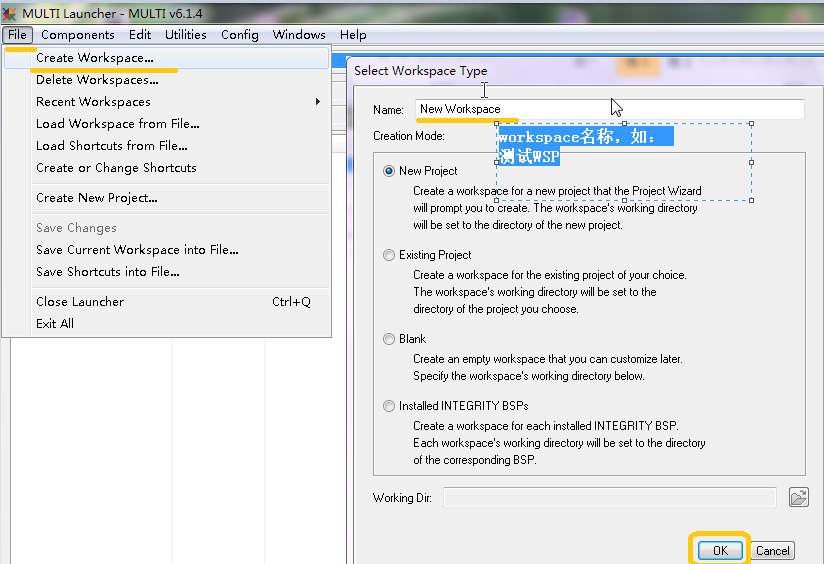
X:\05\_SW1\4-SW Tools\1-9-GreenHills\3-License\DongleLicense

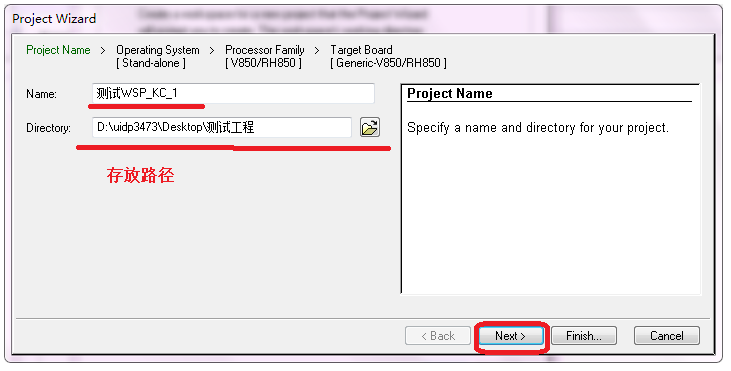
选择相应狗的文件

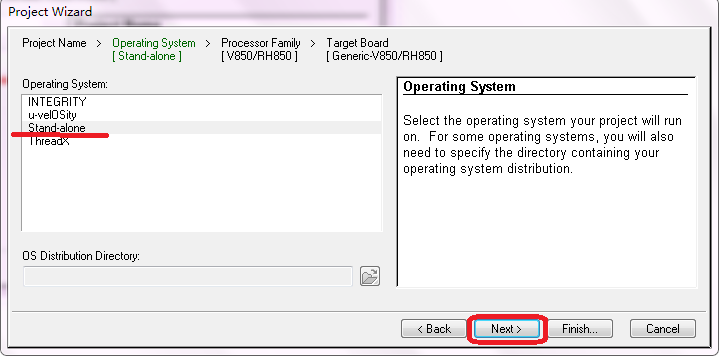


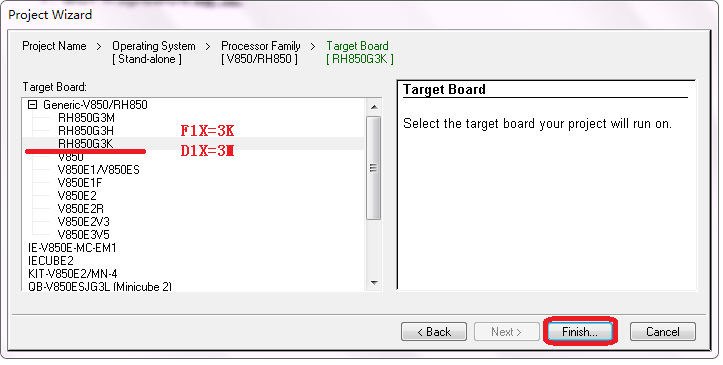


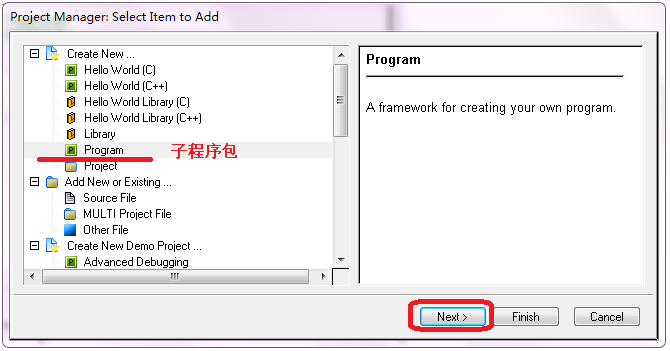
5.2 建立工程

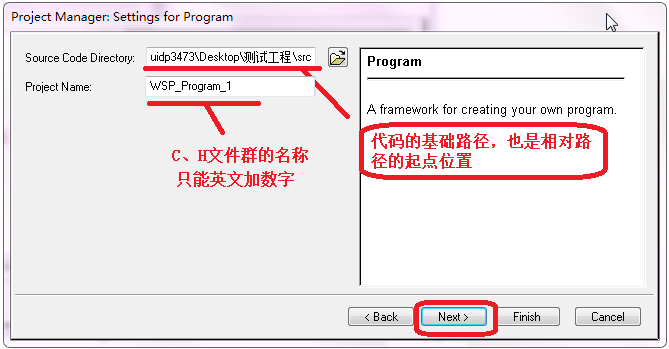


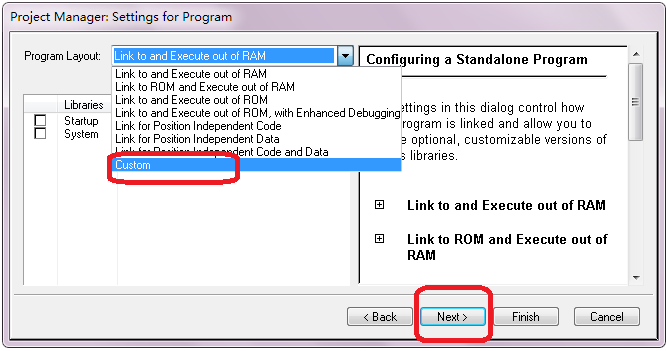


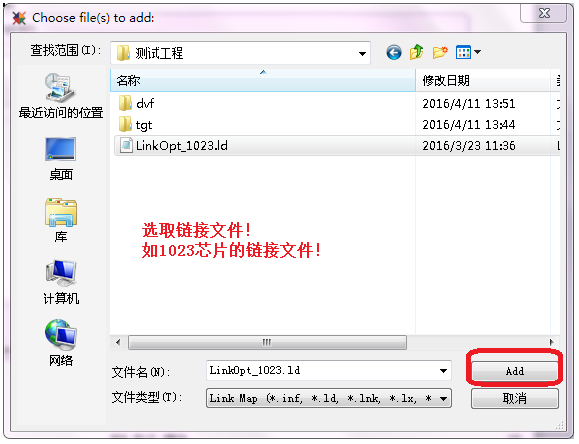




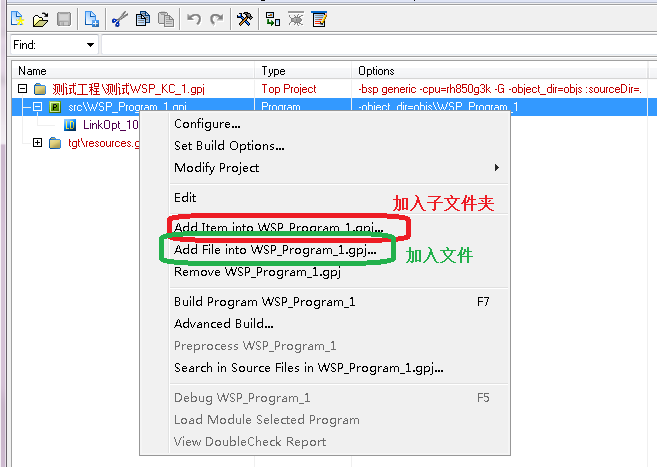


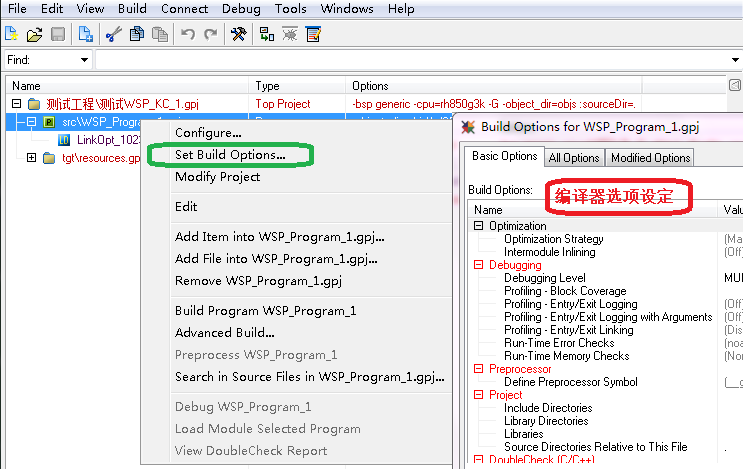


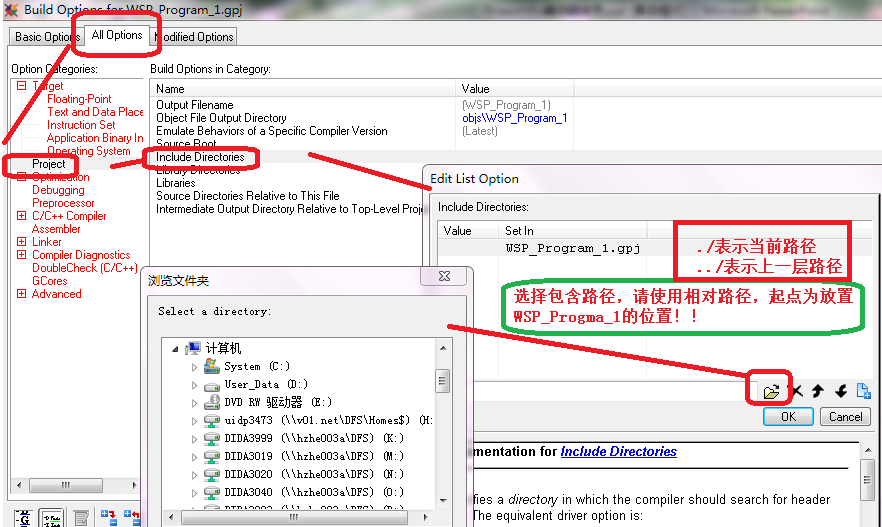


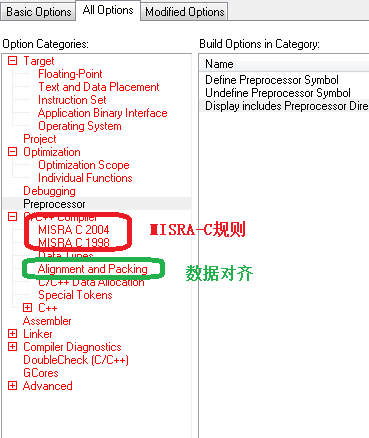


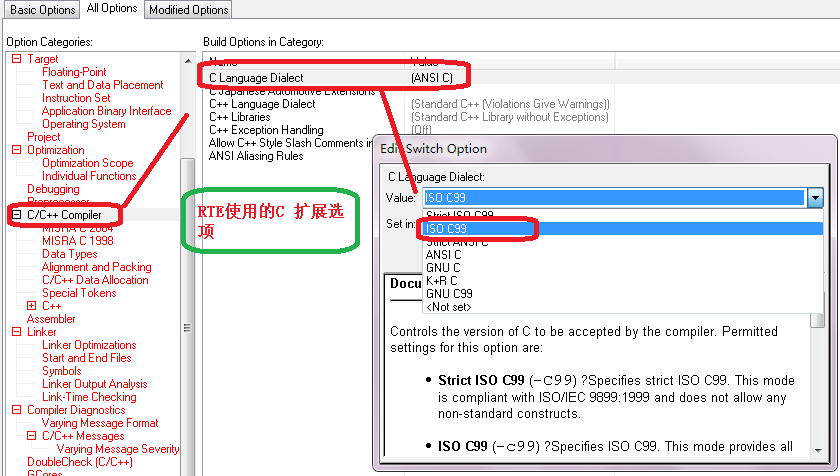
**这里请选择F1L\_MAP\_\_XXXX.ld文件**

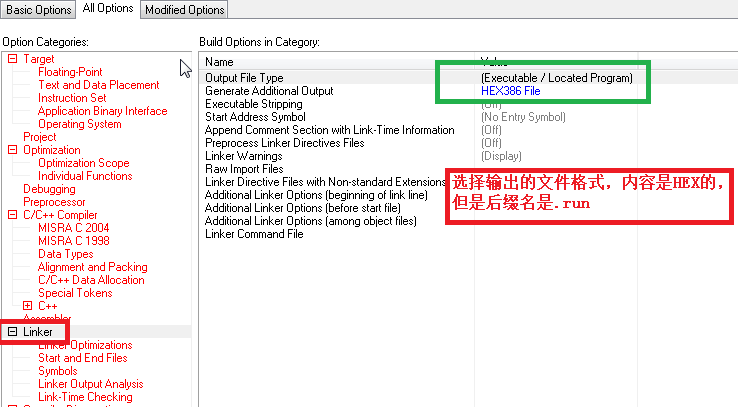




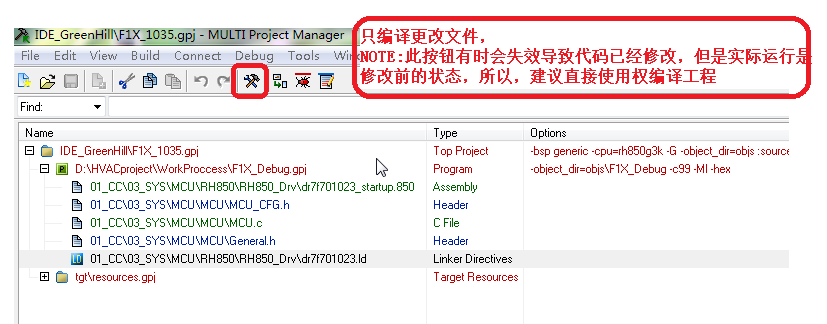


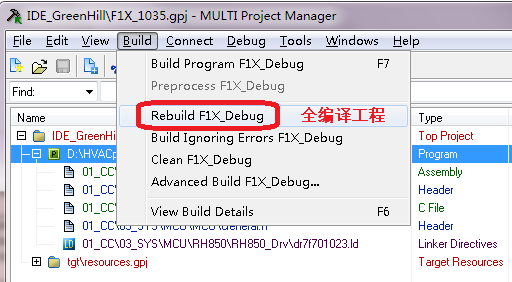


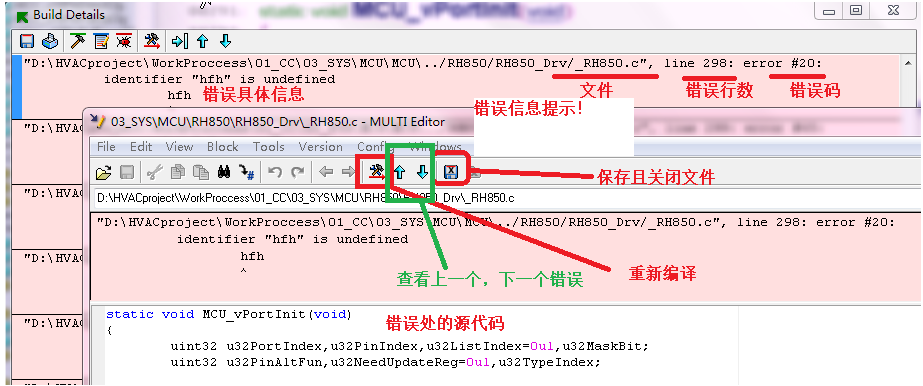




选择此选项以后，在IDE的目录下可以产生.run文件，此文件为可以烧录到MCU上运行的hex文件。



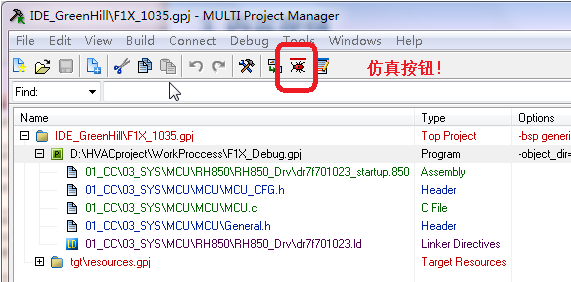




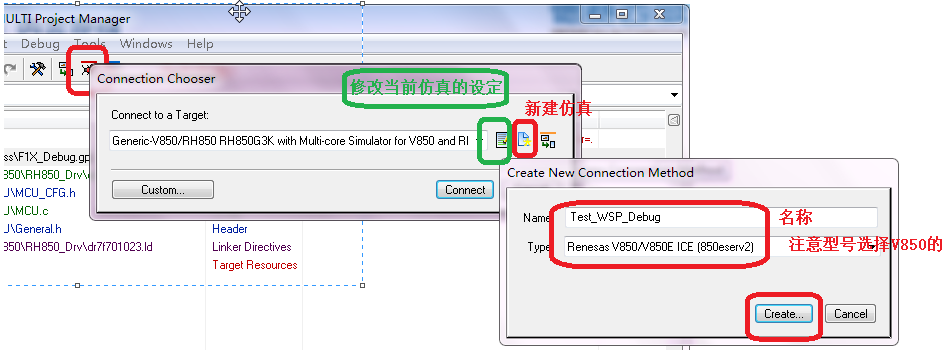
1. 非常重要：请确保LD文件内的内存限定是正确的！

# 基本仿真操作

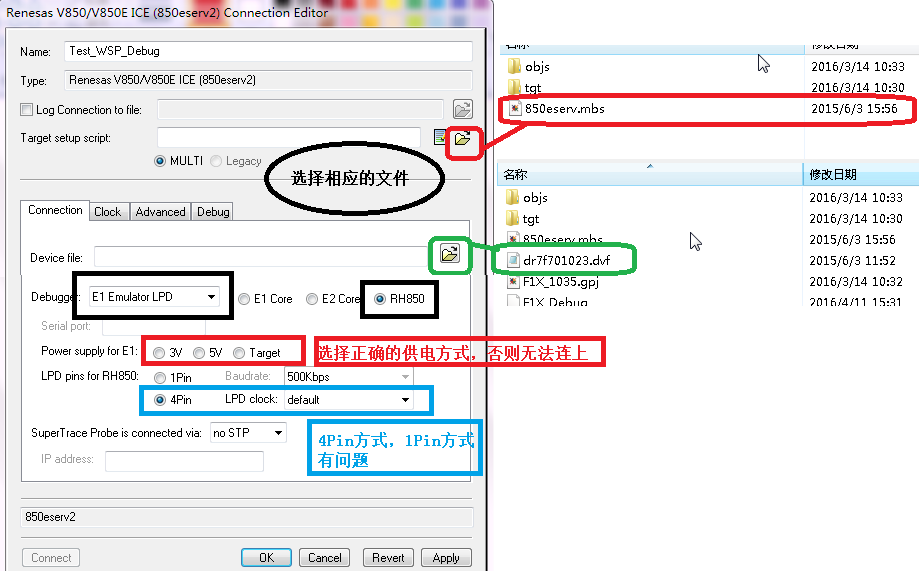
在工程编译通过以后，可以对工程进行仿真以查找相应的问题，具体的操作步骤如下：



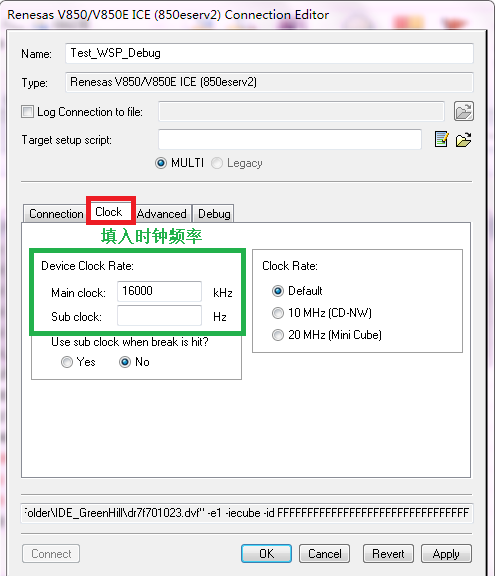
创建一个新的仿真工程，请选择V850的仿真型号：



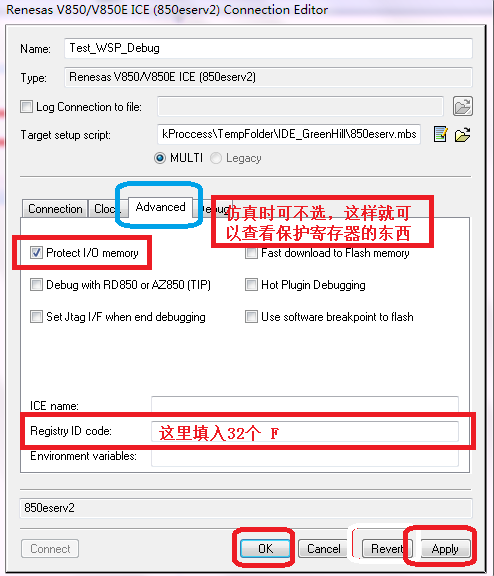
选择相应的仿真命令文件，根据实际情况选择供电方式，如果使用仿真器的供电，则请关闭外部电源，



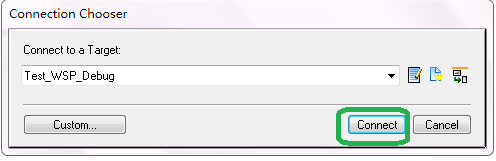
根据实际PCB板上的mainOSC的频率填入



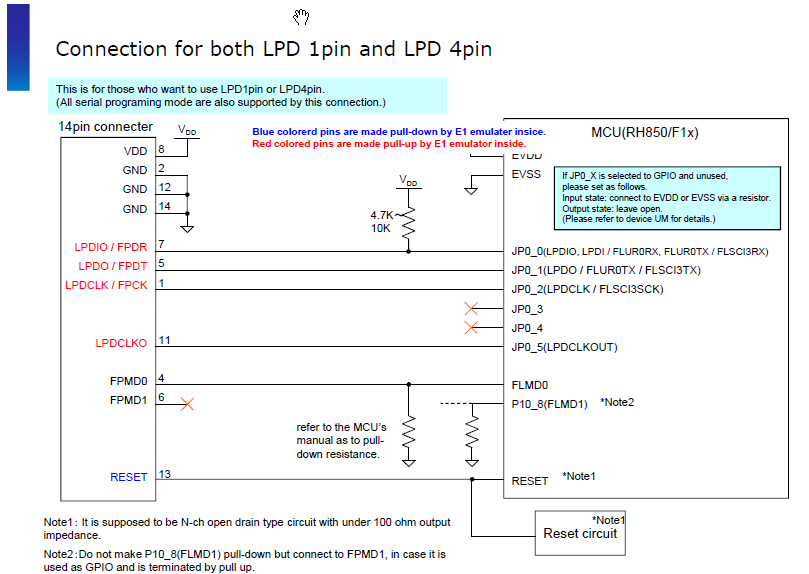
进阶选项的配置，protect IO的选项可以取消勾选，这样就可以在仿真的时候查看受保护的寄存器的值



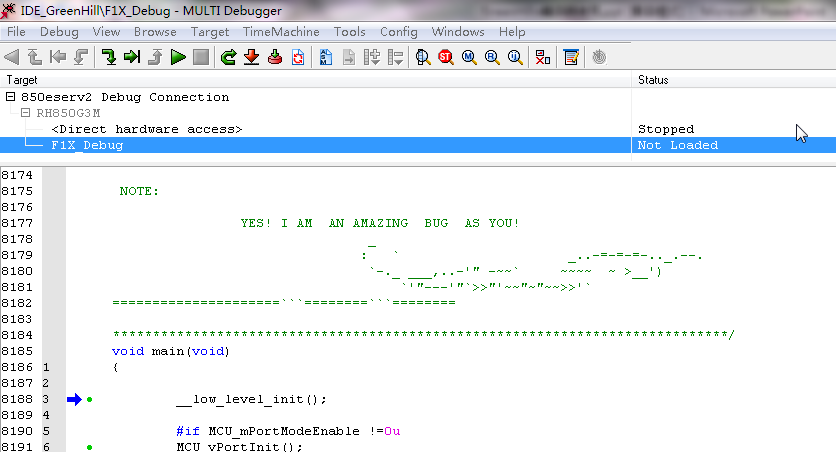
连接到仿真版：



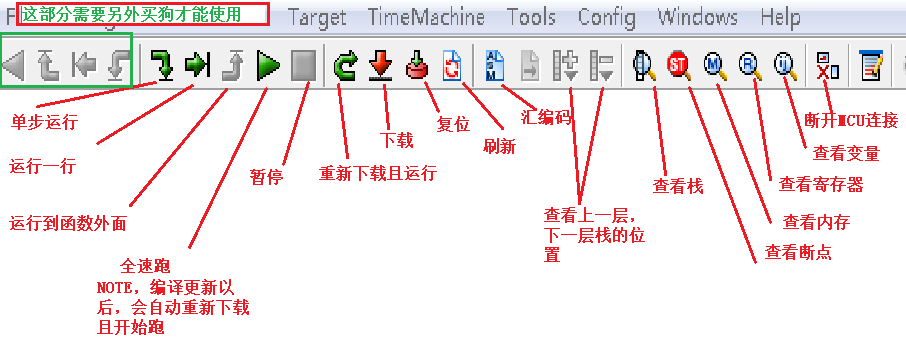
基本连线图：左边的E1的14pin接口，右边是MCU端的具体引脚，电路的连接如下所示。



仿真连接正常时候的

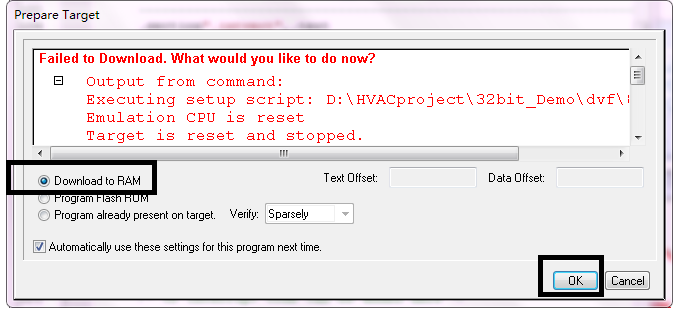


各个图标的含义：

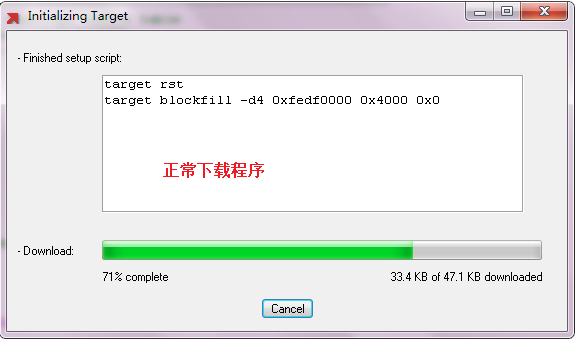


程序下载和开始仿真：

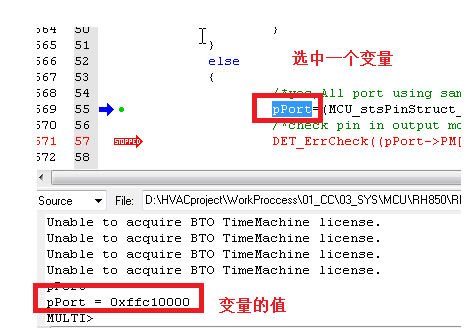




如果出现其他情况，请检查线路连接和电源供电是否出现异常！

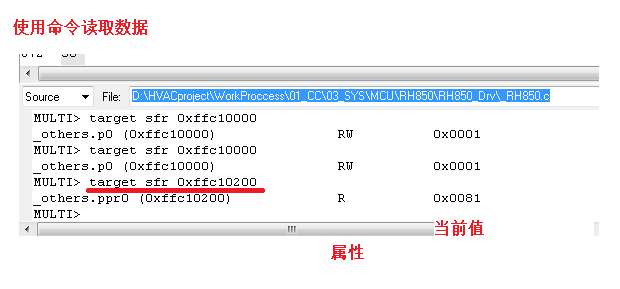


查看一个变量：

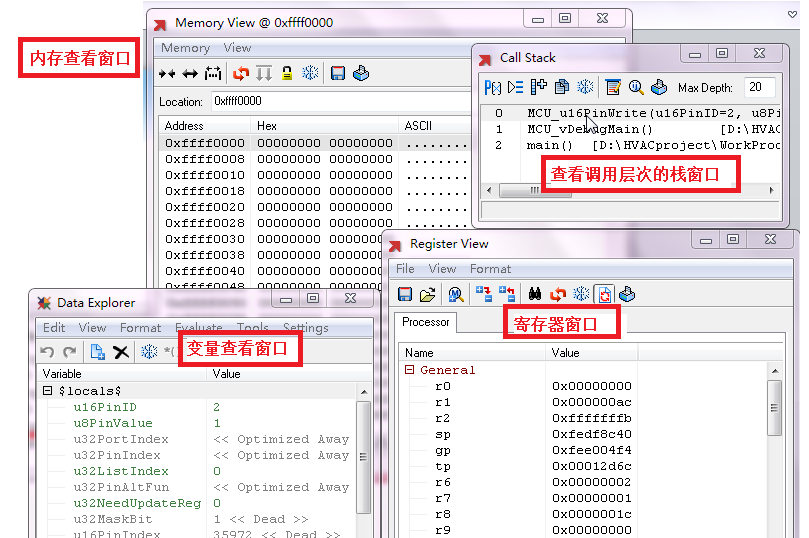


快速查看文件中的某些信息：

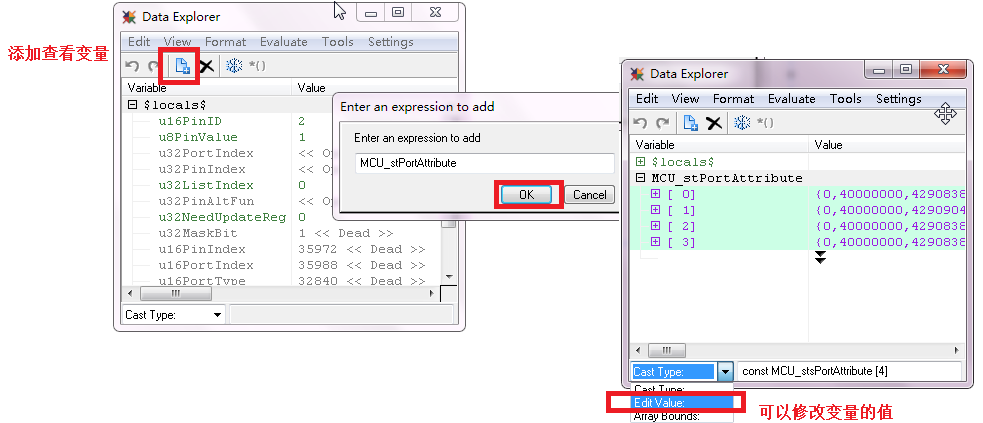




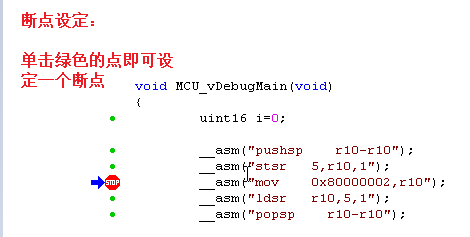
查看各种信息的窗口：



添加一个变量进行查看：



设定一个断点：



通过上面的基本操作，可以进行程序的仿真调试工作。

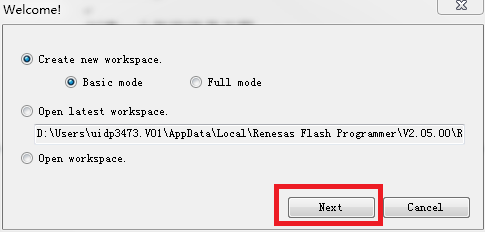
# 基本烧录操作

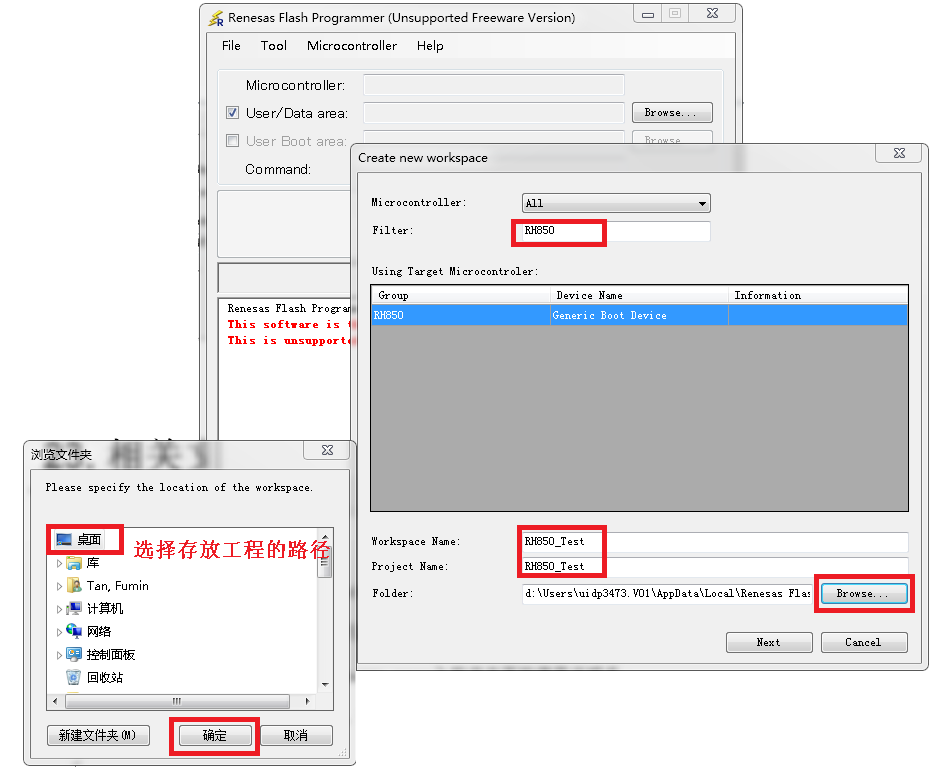
经过编译和仿真以后，需要将得到的可执行文件烧录到MCU内进行运行，具体步骤如下：

开启烧录软件：

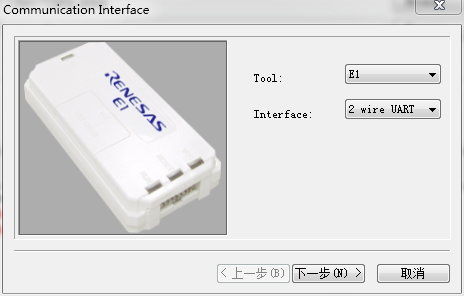


创建一个新的烧录工程：

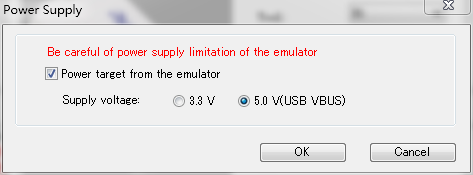




使用E1和2线模式，



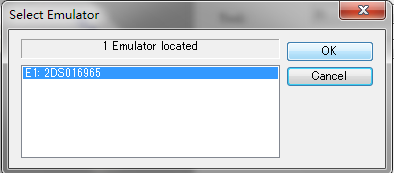
下一步选择供电模式，如果PCB板由外部供电，则不需要勾选，否则请勾选且选择5V供电，中央PCB烧录时的供电由E1负责：



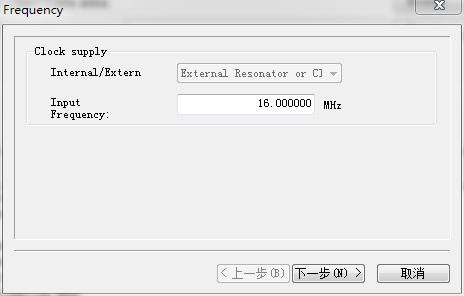
此时请使用烧录器与MCU连接好，然后点击确定，E1会对MCU的信息进行读取：



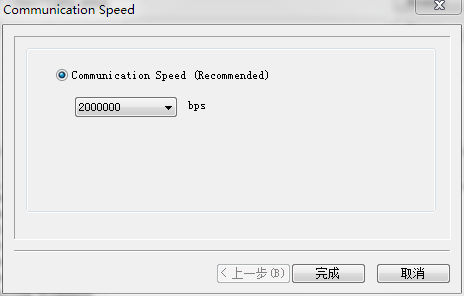
E1信息，点击确定：



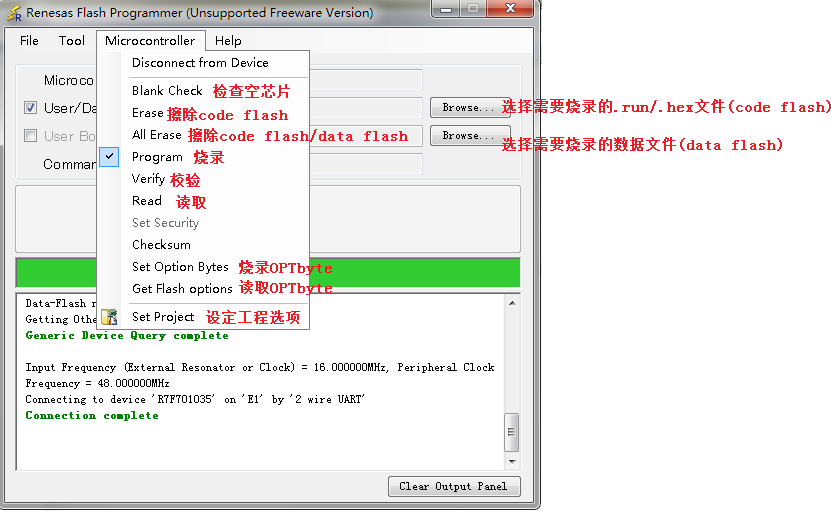
如果连接正常，则出现下列对话框：根据实际情况填入PCB板的mainOSC频率，



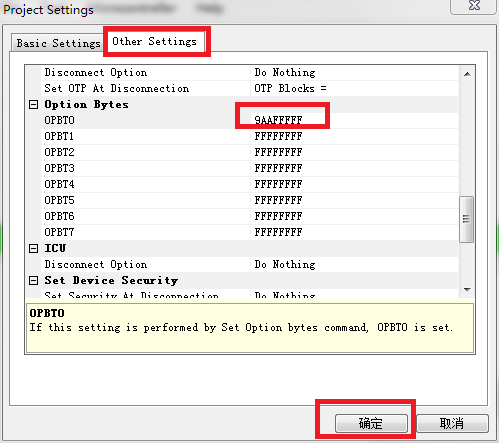
设定成功：

具体操作：



烧录HEX文件时，请使用烧录命令，外加烧录OPTbyte命令，一共两个动作，否则OPTbyte将采用默认值，导致程序运行不正常，具体的OPTbyte的配置在设定工程选项中，对于本代码，请将OPBT0设定为 **9AAFFFFF**，如下：



**通过配置OPTbyte，烧录执行文件，烧录OPTbyte的操作以后，一个MCU就可以根据用户的代码进行运行了。**

# 相关文档

* 1. 芯片手册\_UM\_RH850\_F1L\_V1.2.pdf ====》即我们所说的spec，

此文档描述了所有MCU硬件相关的信息，如寄存器地址，寄存器作用，RAM，Flash等相关信息

* 1. 软件手册\_UM\_RH850G3K User Manual = software.pdf ====》软件方面的使用说明书

此文档描述了MCU软件相关的信息，如CPU工作模式，数据类型，指令操作，中断服务，寄存器特性与作用等

* 1. 电器性能\_UM\_RH850G3K DataSheet\_l(100 pin).pdf ====》电子电器方面的规格说明

此文档描述了MCU的各种时钟属性，总线属性，LVI特性，复位时序，电子布板规范等电子信息

* 1. IDE基本操作\_GHS\_MULTI\_for\_V850\_Workshop.pdf ====》IDE的基本操作，用于快速入门
  2. 最小系统\_RH850\_F1L - 176pin Minimum components.pdf ====》用于搭建芯片运行的最小环境系统
  3. 烧录仿真接线\_RH850F1L\_DebugConnectionCircuit.pdf ====》用于仿真和烧录的接线
  4. 瑞萨免费编译CC的编译手册.pdf =====》瑞萨家免费编译器的使用手册，一般情况下我们用的是greenhills编译器
  5. 编译手册\_RH850\_Greenhills\_build\_v800.pdf ====》greenhills编译器的使用
  6. 编译特性\_R20AN0330ED0200\_RH850.pdf ====》greenhill编译器在RH850中的编译特性，专门介绍编译器在此芯片上的一下处理方式。