|  |  |  |
| --- | --- | --- |
|  | **使用MSP430中的一些总结**   |  | | --- | | 以下是在使用MSP430中的一些总结：  1.系统时钟问题： 系统默认使用DCO，使用外部高速晶振XT2时必须自己开启XT2，并延时50us等待XT2起振，然后手工清除IFG1中的OFIFG位 !!!!一定要注意操作顺序：打开XT2->等待XT2稳定->切换系统时钟为XT2 若后面两步操作反了，在通常情况下不会出现问题，但是在电压不稳MCU频繁复位的情况下，非常容易造成MCU死掉，只能掉电后重新上电才能可靠复位。   2.早期版本的IAR开发环境 必须在Project->Option->C-Spy中选择对应器件的DDF文件，否则调试时无SFR窗口。  3.在写Flash期间，一定要关闭中断，此时CPU无法执行程序  4.调试的时候，不要选中"Release JTAG on Go"  5.在调试时，需要关闭看门狗，否则在打开看门狗的情况下，每当执行到断点，程序将会跳转到入口点从头执行。或者，暂停运行或停止到断点处时，当需要继续运行时，程序不再执行，而是需要使用“同步JTAG”才能继续运行(从头运行?)  6. IAR EWB标识符是区分大小写的  7. 结构变量内部存在对齐要求，通常按照2个字节的位置对齐，在"C/EC++ Compiler Reference Guide"的P93又如下例子： struct { short s; /\* stored in byte 0 and 1 \*/ char c; /\* stored in byte 2 \*/ long l; /\* stored in byte 4, 5, 6, and 7 \*/ char c2; /\* stored in byte 8 \*/ } s; sizeof(s) 为10，而不是8  可以使用#pragma pack来改变这种对齐方式，但是会导致，这个结构只能按字节方式存取  8. MSP430 IAR C/EC++ Compiler支持两个运行库 1. IAR CLIB : 主要用于8或16为处理器，不完全兼容ISO/ANSI C,也不完全支持IEEE 754浮点数，不支持Embedded C++. 2. IAR CLIB : 支持ISO/ANSI C和Embedded C++.  9. cstartup的定制 (1). 在\_\_low\_level\_init()中添加代码。该函数可以用来初始化I/O寄存器，并决定数据段是否在cstartup被初始化。文件430\src\lib\lowinit.c给出了框架，copy到项目目录中使用，文件中有一定的使用说明。 (2) 若\_\_low\_level\_init()中添加代码不能满足要求，则把430\src\lib\cstartup.s43拷贝到自己的工作目录，修改所需代码；然后在将该文件添加到项目，在项目选型的XLINK的include页下选择Ignore CSTARTUP in library即可。  10. 使用#include "io430x14x.h"和#include "in430.h"来替代#include <msp430x14x.h> 可以使用定义的位变量  可以使用以下方式定义位变量，但是编译器最终还是转换为字节操作： struct { unsigned char WDTIE : 1; unsigned char OFIE : 1; unsigned char : 2; unsigned char NMIIE : 1; unsigned char ACCVIE : 1; unsigned char URXIE0 : 1; unsigned char UTXIE0 : 1; } IE1\_bit;  不推荐使用bitfields，效率很低  11. 数组的索引值用int型效率最高,char型也好略低 数组类型：char型数组效率最高， 其他类型的数组在索引时，都采用了乘法 | |
| zwbmln0928 当前离线 |  |