嵌入式软件设计相关规范

3.1 CodeWarrior 环境下工程组织规范

现在，实验室使用的 16 位、32 位芯片的软件开发平台主要是 CodeWarrior IDE，所以下面就以 CW 环境下的工程文件组织方法为例来介绍嵌入式软件工程组织规范。当然，如果使用其他 MCU 软件开发环境，也应当尽量遵循这个规范。

1. 工程文件结构

如下图给出了小灯闪烁工程相关源文件的树型结构。

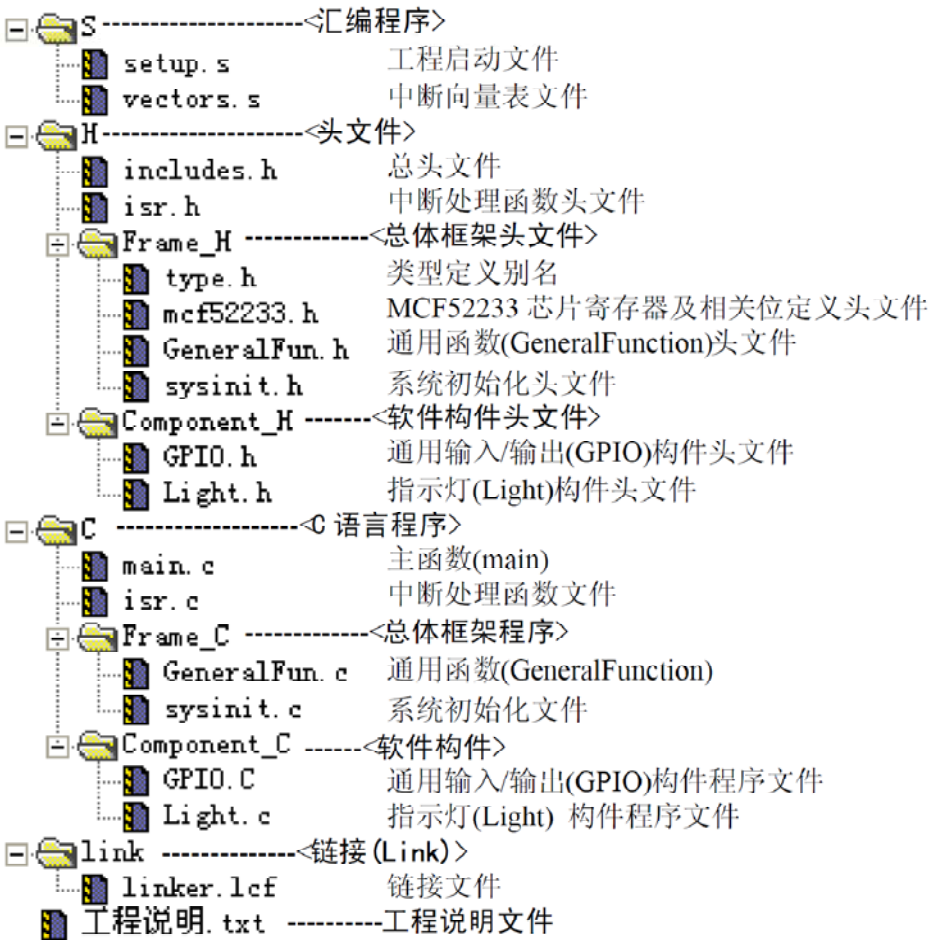

图1 小灯闪烁工程相关源文件的树型结构

1）汇编程序文件

在“汇编程序”中包含的文件与工程初始化相关，包括工程启动文件“steup.s”与中断向量表文件“vectors.s”，由于源代码完全使用汇编语言编写，故将其独立作为一类。

2） C语言程序文件和头文件

C 语言程序文件和头文件基本上是一一对应的，例如，includes.h 与 main.c、isr.h 与 isr.c、GeneralFun.h 与 GeneralFun.c、GPIO.h 与 GPIO.c 等。C 语言程序文件又分为如下 3 类：

（1）主函数文件和中断函数处理文件

一般嵌入式软件的执行流程有两条线，一条是主循环，另一条就是中断请求处理，分别对应了 main.c 和 isr.c 文件，因此将这两个文件与其他文件分开管理。

（2）总体框架程序文件

总体框架程序文件中包含的是通用程序文件，这些文件在每一个工程都会使用到。例如，图 1 小灯工程中的 GeneralFun.c 文件，它提供了常用的功能性子函数，如延时子函数等，像这些框架性的文件，每个工程都要包含。

（3）软件构件

每个功能实体，或叫作“构件”，都对应一个软件构件程序文件。例如，图 1 小灯工程中的 Light.c 就是用于指示灯控制的“Light”构件。

由于头文件基本上是与 C 语言程序文件一一对应的，所以对头文件也是如此分类，这里不再赘述。

3） 链接文件

链接文件的后缀名为.lcf，它是一个地址链接文件，用于告诉编译器代码是如何安放在具体的地址空间的。

4） 工程说明

工程说明文件用来放对构件的描述、测试用例和系统的硬件接线等信息，还可以将调试心得以及工程的变更信息和注意事项放在其中。特别注意，要重视工程说明文件，必要的工程说明对于软件开发者的交互是非常必要的。

2. 不带操作系统的嵌入式程序组织结构

1）保证各模块的松散耦合性质

各个模块之间不能有横向联系，从程序中删除某个模块的代码，不能影响到整个系统的运行。模块之间禁止使用全局变量传递参数，底层模块的驱动不能使用全局变量，尽可能少的使用文件级变量。编写底层驱动模块不要考虑具体项目应用，只依照模块本身应有的功能来编写，要保证其通用性。

2）嵌入式程序组织

无操作系统的嵌入式程序分主程序和中断处理程序两条主线，设计时主程序一定要有流程图的概念，中断处理程序与主程序通过全局变量进行交互。主程序的流程一定要清晰化，中断处理程序一定简单化，不要过多占用系统时间。

3）主程序的结构

主程序的一般结构为：芯片初始化—>模块初始化—>内存变量初始化—>开放中断—>进入系统总循环。通常各个模块中断的开启与关闭语句，可在相关的头文件中用宏定义来固定成类似函数调用的形式供主函数和中断处理函数调用。在主程序中开放总中断的语句，应该放置在进入系统总循环之前的几句代码处。在开放总中断之前，一定要保证芯片、模块以及内存变量的初始化圆满完成。若检测到看门狗复位，可以视情况决定是否重新执行内存变量初始化过程。若检测到短时间内在同一地方产生多次 COP 复位，则可判定系统在该处有问题。

要考虑主循环周期的两个极限，即最快和最慢时执行一次的时间。尽可能降低最大极限时间的数值，使得最小和最大两个极限时间的差异不是很大。看门狗喂食不要到处加，不能

加在中断处理程序中，一般加在主循环中的开头的语句中，要考虑最大极限时间的影响。

要留两个小灯用于系统状态的显示：运行指示灯和错误指示灯。

在主循环中，对任务进行合理的切分，用全局的状态变量作为信号量来实现各个任务之间的通信。这样可以有效降低主循环的最大极限时间，提高系统的实时性能。

4）中断处理程序

除非实时要求非常高，否则不要在中断中直接干预硬件，且中断程序尽可能简单化，使用的全局变量也应尽量少。

3. 注意事项

在实际编程过程中，除了要按上述方法组织工程文件之外，还要注意如下几点：

（1）工程中可以定义全局变量，但只能在 includes.h 中定义。

（2)只有 main.c 和 isr.c 可以直接使用全局变量,其他程序文件尽量不要使用全局变量，如果要用，也要用传址的方式实现，不得直接使用。

（3）main.c 文件要尽量做到与芯片无关，即主程序不应直接控制底层构件，而要通过调用构件函数来控制构件。

（4）主循环中的代码如果太长，可以按功能划分成多个函数，放在主函数之后，这些函数在 main.c 中声明即可，不必放在 includes.h 中。

（5）在每个软件构件的头文件中，要有对该构件的描述信息，一般用“/*……*/”括起来，以方便软件构件的移植和使用。

（6）总中断和模块中断的开和关，应当在 isr.h 文件中以宏的形式定义。

3.2 MCU-C 程序基本编程规范

1．概述

为了提高源程序的质量和可维护性，从而最终提高软件产品生产力，特编写此规范。本标准规定了程序设计人员进行程序设计时必须遵循的规范。本规范主要针对单片机编程语言和 08 编译器而言，包括排版、注释、命名、变量使用、代码可测性、程序效率、质量保证等内容。

2．命名规则

1）命名基本原则

（1）命名清晰明了，有明确含义，使用完整单词或约定俗成的缩写。通常，较短的单词可通过去掉元音字母形成缩写；较长的单词可取单词的头几个字母形成缩写。即"见名知意"。

（2）命名风格要自始至终保持一致。

（3）命名中若使用特殊约定或缩写，要有注释说明。

（4）为了代码复用，命名中应避免适用与具体项目相关的前缀。

（5）应使用英语命名。

2）预定义(#define)

只使用大写字母，下划线和数字。

例如： #define MAX_LENGTH 1

3）宏和常量命名

只使用大写字母，下划线和数字。宏和常量用全部大写字母来命名，词与词之间用下划线分隔。对程序中用到的数字均应用有意义的枚举或宏来代替。

4）变量命名

变量命名规则： <<范围>_类型>名称。

范围	前最
结构体的成员变量	m_
静态变量	s_
全局变量	g_
局部变量	无前缀

类型	前缀
指针（Pointer）	p
枚举（Enumeration）	e
布尔（Boolean）	b
浮点（Float）	f
双精度（Double）	d
字符（Char）	c
结构（Structure）	st
其他数字类型, e.g. byte, (unsigned) int, (unsigned)long int, (unsigned)short, (unsigned) short int, (unsigned) long long	n

例如：

```
//全局变量
int g_nMaxCount;
    //函数体内局部变量
MyEnumType    eParsingMode;
```

局部循环体控制变量优先使用 i、j、k 等；局部长度变量优先使用 len、num 等；临时中间变量优先使用 temp、tmp 等。

5）结构和类型定义（typedef）

按 Camel-Style 方式命名，如 ThisIsAnExampleStructOrEnumOrTyedef 。避免使用下划线。

6）枚举

按 Camel-Style 方式命名，如 ThisIsAnExampleStructOrEnumOrTyedef 。避免使用下划线。

```
Enum DataSetState
{
    ValidModified = 1,
    InvalidModified = 2,
    AllDataLoaded = 3
}
```

7）函数命名

按 Camel-Style 方式命名， ThisIsAnExampleMethod。

8）文件命名

一个文件包含一类功能或一个模块的所有函数，文件名称应清楚表明其功能或性质。每个.c 文件应该有一个同名的.h 文件作为头文件。

3．注释

1）注释基本原则

有助于对程序的阅读理解，说明程序在"做什么"，解释代码的目的、功能和采用的方法。一般情况源程序有效注释量在 30％左右。 注释语言必须准确、易懂、简洁。边写代码边注释，修改代码同时修改相应的注释，不再有用的注释要删除。 汇编和 C 中都用"//"，取消";"不使用段注释" /* */ "（调试时可用）。

2）文件注释

文件注释必须说明文件名、项目名称、函数功能、创建人、创建日期、版本信息等相关信息。修改文件代码时，应在文件注释中记录修改日期、修改人员，并简要说明此次修改的目的。所有修改记录必须保持完整。 文件注释放在文件顶端，用"/*……*/"格式包含。 注释文本每行缩进 4 个空格；每个注释文本分项名称应对齐。

```
/*************************************************************
文件名称：
项目名称：
作 者：
版 本：
```

说 明：
修改记录：
**********/

3）函数注释

函数头部注释应包括函数名称、函数功能、入口参数、出口参数等内容。如有必要还可增加作者、创建日期、修改记录（备注）等相关项目。 函数头部注释放在每个函数的顶端，用"/*……*/"的格式包含。其中函数名称应简写为 FunctionName()，不加入、出口参数等信息。

```
/**********
函数名称：
函数功能：
入口参数：
出口参数：
备 注：
**********/
```

4）代码注释

代码注释应与被注释的代码紧邻，放在其上方或右方，不可放在下面。如放于上方则需与其上面的代码用空行隔开。一般少量注释应该添加在被注释语句的行尾，一个函数内的多个注释左对齐；较多注释则应加在上方且注释行与被注释的语句左对齐。通常，分支语句（条件分支、循环语句等）必须编写注释。其程序块结束行"} "的右方应加表明该程序块结束的标记"end of ……", 尤其在多重嵌套时。例如：

```
int     nNum = 7;            //   注释 。。。

// 注释。。。
for(int i = 0 ; I < nNum; i++)
{
    If( i = = 0)
    {
        // code……
    }
    else
    {
            //code
    } //end of if(i == 0)

} // end of for (i)
```

5）变量、常量、宏的注释

同一类型的标识符应集中定义，并在定义之前一行对其共性加以统一注释。对单个标识符的注释加在定义语句的行尾。 全局变量一定要有详细的注释，包括其功能、取值范围、哪些函数或过程存取它以及存取时的注意事项等。 注释用"//…//"的格式。

4．统一类型别名定义

```
typedef unsigned char        uint8;      //  8 位无符号数
typedef unsigned short int   uint16;     // 16 位无符号数
typedef unsigned long int    uint32;     // 32 位无符号数
typedef char                 int8;       //  8 位有符号数
typedef short int            int16;      // 16 位有符号数
typedef int                  int32;      // 32 位有符号数
//不优化变量类型
typedef volatile uint8       vuint8;     //  8 位无符号数
typedef volatile uint16      vuint16;    // 16 位无符号数
typedef volatile uint32      vuint32;    // 32 位无符号数
typedef volatile int8        vint8;      //  8 位有符号数
typedef volatile int16       vint16;     // 16 位有符号数
typedef volatile int32       vint32;     // 32 位有符号数
```

5．编码

- 代码的每一级均往右缩进 4 个空格的位置
- 不使用 Tab 键
- 相对独立的程序块之间要加空行
- 括号内侧（即左括号后面和右括号前面）不加空格，多重括号间不加空格。如：SetName(GetFunc())
- 函数形参之间应该有且只有一个空格（形参逗号后面加空格），如：CallFunction(para1, para2, para3)，而 CallFunction(para1,para2,para3) 不符合要求。
- 操作符前后均加一个空格，如： nSum = nNunm1 + nNum2 。而 nSum=nNunm1+nNum2 则不符合要求。
- 单目操作符，如"!"、"~"、"++"、"-"、"&"（地址运算符）等，后面不加空格，如：i++ ，pName = &name，bRes = !(x < 10)。

- if、else if、else、for、while 语句无论其执行体是一条语句还是多条语句都必须加花括号，且左右花括号各独占一行。
- Switch 语句必须包含 default 分支。如：

```
        Switch(nNum)
{
    Case 1:
        break;
    Case 2:
        break;
    Default:
        break
}
```

- 一个函数不要超过 80 行代码。

3.3 项目基本工作流程

1．创建工作目录

只有在规范的项目流程指导下，才能做出规范的项目。下面是苏州大学 Freescale 实验室近几年来做若干项目基础上总结出来的一些规范，仅供大家参考。

首先，我们在项目开始的时候，要给项目建立一个文件夹，专门来记录项目有关的材料，文件夹以项目命名。在这个文件夹下，我们列出五个子文件夹，分别命名为 01 文档、02 资料、03 软件、04 硬件、05 其它。这样就可以将各种各样的资料分门别类的放在相应的文件夹中。当然在子文件夹中也可以根据需要再建立子文件。

01 文档中存放在实验过程中写的一些材料，比如说需求分析、项目技术要点等文档。Word 文档有对应的规范，详细请见实验室内部资料 Word 文档规范；

02 资料里面可以存放实验室内部的一些参考资料，也可以是从网上下载下来的一些资料，用子文件夹将其整理好，最好标上日期，方便以后查找使用方便；

03 硬件存放用 Protel 软件制作的制造硬件板的原理图和 PCB 板图。使用 Protel 软件也有一定的规范，详细请见实验室内部资料 Protel 制作 PCB 板的规范。

04 软件用于存放软件设计材料。实验室也有内部规范，详细请见实验室内部资料。

其他材料放在 05 其他里面。

其次，我们在做项目的时候，遇到问题要跟客户进行交流，写出需求分析，放在 01 文档子文件夹中。记得要写备忘录，记上日期。可以放在 05 其他中也可以放在 01 文档里。

再次，在设计文档的最后留一些地方，用来存放后序工作的补充，文档记录，过程记录，USB 测试等等。

最后，在项目认为可以交的时候，交给实验室一份，包括剩下来的硬件材料等等。

对于一个新的项目，我们主要从以下几个方面来开始一个新项目的设计。

2．需求分析

本节主要是在原始需求文档的基础上理解需求方的具体要求及细节，完全吃透需要方的要求，并以文档的形式记录下来。

需求分析主要可分为系统概述和输入输出分析两个部分，具体内容可以按实际需求进行细分。

1）系统概述

系统概述部分主要是在原始需求文档基础上进一步理解需求方的意图，转化为自己的语言。

为更加清晰的表达需求，可以采用立体图的形式表示整个系统，图中应尽量体现项目中所有的要素，并配合简要的解释说明。同时可以用表格的形式对关键要素进行分析比较。

对于系统的整个工作流程也应该在这部分进行叙述，对于复杂的工作流程可以采用流程图的形式。同时系统的工作流程可能有几种不同的情况，也要分别进行叙述。

2）输入输出分析

在一个具体的项目中，输入输出信号一般比较多，所以在本节开始的地方就应该使用一个系统框图来详细描述所有的输入输出量。在该系统框图中用箭头的形式表示输入输出，同

时图中一定不要涉及到 MCU 的具体引脚。具体可参照标准文档。

在系统框图之后应该采用表格形式具体描述每一个输入输出信号。列表时首先对所有的输入输出信号按模块进行归类并给其相应的编号；同时将开关量输入、开关量输出、模拟量输入和模拟量输出也要分开。表头可具体有：类型、编号、名称、命名、来源和备注这几列。其中备注是用来描述信号的具体功能及其要注意的地方等信息。

这一部分可以直接拷贝标准文档，直接修改里面的内容即可。

3．硬件设计

硬件设计是在充分分析之后，根据系统的所需进行相应的硬件电路的设计。

在硬件原理设计部分主要是按各个硬件功能模块用虚线框括起来，并配备简短的说明，可以让读者一目了然。布 PCB 板的时候需要注意细节问题，如电源线宽，最小系统等部分。

硬件设计方法和规则请参见：

错误！未找到引用源。

错误！未找到引用源。

错误！未找到引用源。

错误！未找到引用源。

错误！未找到引用源。

具体的硬件设计细节可参见详细的电路设计要求和标准的电路图。

4．软件设计

软件设计部分主要可分为功能模块的程序设计和总体的功能实现，实际上功能模块的程序设计是可以与硬件设计部分同时进行的。

功能模块的程序主要就是实现各个功能模块的驱动程序，例如串行通信、键盘、液晶显示等。这部分程序实验室很大一部分是有基础的，可以在参考的基础上进行相应的改动。

总体功能的实现就需求方最终的产品功能，就是按照需求分析的结果来进行程序设计。

在程序设计上需要安装一定的规范来编写，如头文件的合理安排、一行的长度、注释声明等的规范，具体请参见详细的软件设计规范。

需要注意的是，在项目交付前，必须检查以下工作是否已经完成：

（1）考虑热复位和冷复位，防止死机现象，即使是有休眠状态的设备也要考虑该问题。在重要的阶段将变量值写到 flash。

冷复位（Restart）：单片机从没加电到加上电源,而自动产生的复位称为冷复位。

热复位（Reset）：单片机在已经通电的情况下,给它一个复位信号,称为热复位。

冷复位会使单片机的特殊功能寄存器（SFR）和数据存储器（RAM）的内容都改变;而热复位只是特殊功能寄存器（SFR）的内容改变而单片机的内部数据存储器（RAM）的内容不变。

（2）在程序合适的地方加看门狗。

（3）重视对一些特殊的模块定期做初始化，如 LCD、LED、IC 卡和时钟模块等。

（4）在适当的时间将整个系统人工复位一下。

（5）充分利用指示灯，如运行、故障、报警指示灯等；充分利用显示器件，如 LCD 等。在主循环之前就要对指示灯进行干预。

（6）加上保密字节。

5. 测试

1）系统测试

系统测试是用来测试我们所开的发产品的功能和性能是否符合客户的要求。所以，一旦需求规格说明书制定以后，就需要制定系统测试计划，根据功能点编写测试用例，测试结束后填写测试报告。

测试计划主要制定：

(1) 需要测试的对象
(2) 测试对象所需要用到的测试用例
(3) 测试环境
(4) 测试工具

测试用例主要用来制定某个功能点的测试步骤，定义测试前提，测试通过与否的标准。如下表格给出了测试用例的样例。

测试用例名称	标签正确循环发送
测试内容	本测试用例用来测试标签是否将标签数据按照每 3 秒发送一次的频率向外发送
测试工具和对象	标签 1 个，测试用 PC 一台(连接标签发送监控器)，标签发送监控器，TagSendingMonitor.exe。
前提条件	1.测试用的串口工作正常 2.标签发送监控器正常工作 3.标签写入测试软件 TagSendingMonitor.exe 正常工作 4.标签中已写入有效标签数据
特别说明	
测试步骤	
1	标签发送监控器与电脑串口相连。
2	运行 TagSendingMonitor.exe，设置波特率为 38400
3	标签监控器上电运行一到两分钟
4	观察 TagSendingMonitor.exe 界面输出
5	若界面输出的标签内容与写入的数据一致，并且大概每 3 秒收到一个标签数据，则认为测试通过；否则认为失败。

测试报告主要汇总各个测试用例的测试结果，记录测试过程中的一些数据。

2）构件测试

构件是以面向硬件对象的设计理念，封装出来的软件模块，是嵌入式应用软件的组成基础。所以每个构建必须经过严格的测试。具体测试方法和流程，请参见：

错误！未找到引用源。

3.4 需求分析

需求分析是指理解用户需求，就软件功能与客户达成一致，估计软件风险和评估项目代价，最终形成开发计划的一个复杂过程。在这个过程中，用户的确是处在主导地位，需求分析工程师和项目经理要负责整理用户需求，为之后的软件设计打下基础。需求分析阶段结束后，要求得到：①SRS 文档(System Requirement Specification)；②DRM 文档；③Acceptance Plan。

从广义上理解：需求分析包括需求的获取、分析、规格说明、变更、验证、管理的一系列需求工程。狭义上理解：需求分析指需求的分析、定义过程。

1. 为什么要需求分析

需求分析就是分析软件用户的需求是什么。如果投入大量的人力，物力，财力，时间，开发出的软件却没人要，那所有的投入都是徒劳。如果费了很大的精力，开发一个软件，最后却不满足用户的要求，从而要重新开发过，这种返工是让人痛心疾首的。(相信大家都有体会)比如，用户需要一个 for linux 的软件，而你在软件开发前期忽略了软件的运行环境，忘了向用户询问这个问题，而想当然的认为是开发 for windows 的软件，当你千辛万苦地开发完成向用户提交时才发现出了问题，那时候你是欲哭无泪了，恨不得找块豆腐一头撞死。

需求分析之所以重要，就因为他具有决策性，方向性，策略性的作用，他在软件开发的过程中具有举足轻重的地位。大家一定要对需求分析具有足够的重视。在一个大型软件系统的开发中，他的作用要远远大于程序设计。

2. 需求分析的任务

简言之，需求分析的任务就是解决“做什么”的问题，就是要全面地理解用户的各项要求，并准确地表达所接受的用户需求。

3. 需求分析的过程

需求分析阶段的工作，可以分为四个方面：问题识别、分析与综合、制订规格说明、评审。

1）问题识别

就是从系统角度来理解软件，确定对所开发系统的综合要求，并提出这些需求的实现条件，以及需求应该达到的标准。这些需求包括：功能需求(做什么)，性能需求(要达到什么指标)，环境需求(如机型，操作系统等)，可靠性需求(不发生故障的概率)，安全保密需求，用户界面需求，资源使用需求(软件运行是所需的内存，CPU 等)，软件成本消耗与开发进度需求，预先估计以后系统可能达到的目标。

2）分析与综合

逐步细化所有的软件功能，找出系统各元素间的联系，接口特性和设计上的限制，分析他们是否满足需求，剔除不合理部分，增加需要部分。最后，综合成系统的解决方案，给出要开发的系统的详细逻辑模型(做什么的模型)。

3）制订规格说明书

即编制文档，描述需求的文档称为软件需求规格说明书。请注意，需求分析阶段的成果是需求规格说明书，向下一阶段提交。

4）评审

对功能的正确性，完整性和清晰性，以及其它需求给予评价。评审通过才可进行下一阶段的工作，否则重新进行需求分析。

4. 需求分析的方法

需求分析的方法有很多。这里只强调原型化方法，其它的方法如：结构化方法，动态分析法等(个人认为，对初学者不必深究这些方法，实际上我也从来没用过这些方法)在此不讨论。

原型化方法是十分重要的。原型就是软件的一个早期可运行的版本，它实现了目标系统的某些或全部功能。

原型化方法就是尽可能快地建造一个粗糙的系统，这系统实现了目标系统的某些或全部功能，但是这个系统可能在可靠性，界面的友好性或其他方面上存在缺陷。建造这样一个系统的目的是为了考察某一方面的可行性，如算法的可行性，技术的可行性，或考察是否满足用户的需求等。如，为了考察是否满足用户的要求，可以用某些软件工具快速的建造一个原型系统，这个系统只是一个界面，然后听取用户的意见，改进这个原型。以后的目标系统就在原型系统的基础上开发。

原型主要有三种类型：探索型，实验型，进化型。探索型：目的是要弄清楚对目标系统的要求，确定所希望的特性，并探讨多种方案的可行性。实验型：用于大规模开发和实现前，考核方案是否合适，规格说明是否可靠。进化型：目的不在于改进规格说明，而是将系统建造得易于变化，在改进原型的过程中，逐步将原型进化成最终系统。

在使用原型化方法是有两种不同的策略：废弃策略，追加策略。废弃策略：先建造一个功能简单而且质量要求不高的模型系统，针对这个系统反复进行修改，形成比较好的思想，据此设计出较完整，准确，一致，可靠的最终系统。系统构造完成后，原来的模型系统就被废弃不用。探索型和实验型属于这种策略。

追加策略：先构造一个功能简单而且质量要求不高的模型系统，作为最终系统的核心，然后通过不断地扩充修改，逐步追加新要求，发展成为最终系统。进化型属于这种策略。

5. 需求分析的 20 条法则

客户与开发人员交流需要好的方法。下面建议 20 条法则，客户和开发人员可以通过评审以下内容并达成共识。如果遇到分歧，将通过协商达成对各自义务的相互理解，以便减少以后的磨擦(如一方要求而另一方不愿意或不能够满足要求)。

1）分析人员要使用符合客户语言习惯的表达

需求讨论集中于业务需求和任务，因此要使用术语。客户应将有关术语(例如：采价、印花商品等采购术语)教给分析人员，而客户不一定要懂得计算机行业的术语。

2）分析人员要了解客户的业务及目标

只有分析人员更好地了解客户的业务，才能使产品更好地满足需要。这将有助于开发人员设计出真正满足客户需要并达到期望的优秀软件。为帮助开发和分析人员，客户可以考虑邀请他们观察自己的工作流程。如果是切换新系统，那么开发和分析人员应使用一下目前的旧系统，有利于他们明白目前系统是怎样工作的，其流程情况以及可供改进之处。

3）分析人员必须编写软件需求报告

分析人员应将从客户那里获得的所有信息进行整理，以区分业务需求及规范、功能需求、质量目标、解决方法和其他信息。通过这些分析，客户就能得到一份“需求分析报告”，此份报告使开发人员和客户之间针对要开发的产品内容达成协议。报告应以一种客户认为易于翻阅和理解的方式组织编写。客户要评审此报告，以确保报告内容准确完整地表达其需求。一份高质量的“需求分析报告”有助于开发人员开发出真正需要的产品。

4）要求得到需求工作结果的解释说明

分析人员可能采用了多种图表作为文字性“需求分析报告”的补充说明，因为工作图表能很清晰地描述出系统行为的某些方面，所以报告中各种图表有着极高的价值；虽然它们不太难于理解，但是客户可能对此并不熟悉，因此客户可以要求分析人员解释说明每个图表的作用、符号的意义和需求开发工作的结果，以及怎样检查图表有无错误及不一致等。

5）开发人员要尊重客户的意见

如果用户与开发人员之间不能相互理解，那关于需求的讨论将会有障碍。共同合作能使大家“兼听则明”。参与需求开发过程的客户有权要求开发人员尊重他们并珍惜他们为项目成功所付出的时间，同样，客户也应对开发人员为项目成功这一共同目标所做出的努力表示尊重。

6）开发人员要对需求及产品实施提出建议和解决方案

通常客户所说的“需求”已经是一种实际可行的实施方案，分析人员应尽力从这些解决方法中了解真正的业务需求，同时还应找出已有系统与当前业务不符之处，以确保产品不会无效或低效；在彻底弄清业务领域内的事情后，分析人员就能提出相当好的改进方法，有经验且有创造力的分析人员还能提出增加一些用户没有发现的很有价值的系统特性。

7）描述产品使用特性

客户可以要求分析人员在实现功能需求的同时还注意软件的易用性，因为这些易用特性或质量属性能使客户更准确、高效地完成任务。例如：客户有时要求产品要“界面友好”或“健壮”或“高效率”，但对于开发人员来讲，太主观了并无实用价值。正确的做法是，分析人员通过询问和调查了解客户所要的“友好、健壮、高效所包含的具体特性，具体分析哪些特性对哪些特性有负面影响，在性能代价和所提出解决方案的预期利益之间做出权衡，以确保做出合理的取舍。

8）允许重用已有的软件组件

需求通常有一定灵活性，分析人员可能发现已有的某个软件组件与客户描述的需求很相符，在这种情况下，分析人员应提供一些修改需求的选择以便开发人员能够降低新系统的开发成本和节省时间，而不必严格按原有的需求说明开发。所以说，如果想在产品中使用一些已有的商业常用组件，而它们并不完全适合您所需的特性，这时一定程度上的需求灵活性就显得极为重要了。

9）要求对变更的代价提供真实可靠的评估

有时，人们面临更好、也更昂贵的方案时，会做出不同的选择。而这时，对需求变更的影响进行评估从而对业务决策提供帮助，是十分必要的。所以，客户有权利要求开发人员通过分析给出一个真实可信的评估，包括影响、成本和得失等。开发人员不能由于不想实施变更而随意夸大评估成本。

10）获得满足客户功能和质量要求的系统

每个人都希望项目成功，但这不仅要求客户要清晰地告知开发人员关于系统“做什么”所需的所有信息，而且还要求开发人员能通过交流了解清楚取舍与限制，一定要明确说明您的假设和潜在的期望，否则，开发人员开发出的产品很可能无法让您满意。

11）给分析人员讲解您的业务

分析人员要依靠客户讲解业务概念及术语，但客户不能指望分析人员会成为该领域的专家，而只能让他们明白您的问题和目标；不要期望分析人员能把握客户业务的细微潜在之处，他们可能不知道那些对于客户来说理所当然的“常识”。

12）抽出时间清楚地说明并完善需求

客户很忙，但无论如何客户有必要抽出时间参与“头脑高峰会议”的讨论，接受采访或其他获取需求的活动。有些分析人员可能先明白了您的观点，而过后发现还需要您的讲解，这时请耐心对待一些需求和需求的精化工作过程中的反复，因为它是人们交流中很自然的现象，何况这对软件产品的成功极为重要。

13）准确而详细地说明需求

编写一份清晰、准确的需求文档是很困难的。由于处理细节问题不但烦人而且耗时，因此很容易留下模糊不清的需求。但是在开发过程中，必须解决这种模糊性和不准确性，而客户恰恰是为解决这些问题作出决定的最佳人选，否则，就只好靠开发人员去正确猜测了。

在需求分析中暂时加上“待定”标志是个方法。用该标志可指明哪些是需要进一步讨论、

分析或增加信息的地方，有时也可能因为某个特殊需求难以解决或没有人愿意处理它而标注上“待定”。客户要尽量将每项需求的内容都阐述清楚，以便分析人员能准确地将它们写进“软件需求报告”中去。如果客户一时不能准确表达，通常就要求用原型技术，通过原型开发，客户可以同开发人员一起反复修改，不断完善需求定义。

14）及时作出决定

分析人员会要求客户作出一些选择和决定，这些决定包括来自多个用户提出的处理方法或在质量特性冲突和信息准确度中选择折衷方案等。有权作出决定的客户必须积极地对待这一切，尽快做处理，做决定，因为开发人员通常只有等客户做出决定才能行动，而这种等待会延误项目的进展。

15）尊重开发人员的需求可行性及成本评估

所有的软件功能都有其成本。客户所希望的某些产品特性可能在技术上行不通，或者实现它要付出极高的代价，而某些需求试图达到在操作环境中不可能达到的性能，或试图得到一些根本得不到的数据。开发人员会对此作出负面的评价，客户应该尊重他们的意见。

16）划分需求的优先级

绝大多数项目没有足够的时间或资源实现功能性的每个细节。决定哪些特性是必要的，哪些是重要的，是需求开发的主要部分，这只能由客户负责设定需求优先级，因为开发者不可能按照客户的观点决定需求优先级；开发人员将为您确定优先级提供有关每个需求的花费和风险的信息。

在时间和资源限制下，关于所需特性能否完成或完成多少应尊重开发人员的意见。尽管没有人愿意看到自己所希望的需求在项目中未被实现，但毕竟是要面对现实，业务决策有时不得不依据优先级来缩小项目范围或延长工期，或增加资源，或在质量上寻找折衷。

17）评审需求文档和原型

客户评审需求文档，是给分析人员带来反馈信息的一个机会。如果客户认为编写的“需求分析报告”不够准确，就有必要尽早告知分析人员并为改进提供建议。更好的办法是先为产品开发一个原型。这样客户就能提供更有价值的反馈信息给开发人员，使他们更好地理解您的需求；原型并非是一个实际应用产品，但开发人员能将其转化、扩充成功能齐全的系统。

18）需求变更要立即联系

不断的需求变更，会给在预定计划内完成的质量产品带来严重的不利影响。变更是不可避免的，但在开发周期中，变更越在晚期出现，其影响越大；变更不仅会导致代价极高的返工，而且工期将被延误，特别是在大体结构已完成后又需要增加新特性时。所以，一旦客户发现需要变更需求时，请立即通知分析人员。

19）遵照开发小组处理需求变更的过程

为将变更带来的负面影响减少到最低限度，所有参与者必须遵照项目变更控制过程。这要求不放弃所有提出的变更，对每项要求的变更进行分析、综合考虑，最后做出合适的决策，以确定应将哪些变更引入项目中。

20）尊重开发人员采用的需求分析过程

软件开发中最具挑战性的莫过于收集需求并确定其正确性，分析人员采用的方法有其合理性。也许客户认为收集需求的过程不太划算，但请相信花在需求开发上的时间是非常有价值的；如果您理解并支持分析人员为收集、编写需求文档和确保其质量所采用的技术，那么整个过程将会更为顺利。

6. 点评需求分析误区

要想说什么是好的需求分析，不如说什么是不好的需求分析，知道什么是不好的，自然也就知道了什么是好的。以下就是一些不好的情况：

1）创意和求实

毋庸质疑的，每个人都会为自己的一个新的 idea 而激动万分，特别是当这个 idea 受到一些根本不知道你原本要干嘛的人的惊赞时。但是请注意，当你激动得意的时候，你可能已经忘了你原本是在描述一个需求，而不是在策划一个创意、创造一个概念。很多刚开始做需求分析的人员都或多或少的会犯这样的错误，陶醉在自己的新想法和新思路中，却违背了需求的原始客观性和真实性原则。

永远别忘了：需求不是空中楼阁，是实实在在的一砖一瓦。

2）解剖的快感

几乎所有搞软件的人，做需求分析的时候，一上来就会把用户告诉你的要求，完完整整的作个解剖，切开分成几个块，再细分成几个子块，然后再条分缕析。可是当用户迷惑的看着你辛辛苦苦做出来的分析结果问你：我想作一个数据备份的任务，怎么做？这时，你会发现，需要先后打开三个窗口才能完成这个任务。

永远别忘了：分解是必需的，但最终的目的是为了更好的组合，而不是为了分解。

3）角度和思维

经常听到这样的抱怨：“用户怎么可以提出这样苛刻的要求呢？”。细细一了解，你会发现，用户只不过是要求把一个需要两次点击的功能，改成只有一次点击。这样会导致需要改变需求、改变编码、甚至重新测试，增加工作量。可是，如果换个角度来想想，这个功能，开发的时候只用了几次、几十次，可是用户每天都要用几百次甚 至几千次几万次，改动一下就减少了一半的工作量，对他来说，这样的需求难道会苛刻吗？

永远别忘了：没有任何需求是不对的，不对的只是你的需求分析。试着站在用户的思维角度想想，你的需求分析就会更加的贴近用户，更加的合理。软件应该是以人为本的。

4）程序员逻辑

从程序员成长为系统分析员是一个普遍的轨迹，但并不是一个好的程序员就必然能成为一个好的系统分析员。一些程序员的固化逻辑，使得他们在做需求分析的时候往往钻进了一些牛角里面。比如说 1 / 0 逻辑(或者是说黑白逻辑)，认为不是这样就是那样，没有第三种情况。可实际情况往往是，在一定的时候是这样，其它时候是那样。又比如穷举逻辑，喜欢上来就把所有一二三可能的情况列举出来，然后一个一个分别处理，每个占用三分之一的时间；可是实际的情况往往是，三分之一的情况占了 9 9 %的比例，其它两种情况一年都不会遇到一次。实际中还有很多这样的例子，不一一列举了。

永远别忘了：需求分析和程序设计不尽相同，合理、可行是才是重要的。跳出程序设计的圈子，站在系统的角度上来看问题，你的结论会截然不同。

7. 嵌入式系统的需求分析中应该注意的问题

1）弄清问题框架

从总的角度来把握问题，包括：机械问题、电器问题、计算机问题（包括 MCU 方和 PC 方控制）等。

善于提出问题。以 MCU 为中心来定义输入、输出、通信。

注意输入的数量级，要有数量级的概念。哪些是我们控制的范围。被测量的物理量的测试地点。

2）硬件体系的数量级的分析

（1）输出

①开关量

MCU 的电压的范围：工作电压为 3V 的 MCU，一般 0.3V 左右代表“0”，1.8 左右代表“1”；工作电压为 5V 的 MCU，一般 0.5V 左右代表 “0”，3.6 左右代表 “1”。

MCU 的电流的范围：贴片封装的一般约为 5mA 左右，双列直插的约 10mA 左右。

对常用器件的认识，例：

a. 发光二极管的电流一般多大？约为 3mA，5mA，7～8mA，10mA。

b. 如果通过 MCU 来控制 220V、100W 的电灯(电流约 0.5A)，其控制电路该如何设计？控制电路依次为：二极管，光电隔离措施，继电器线圈，继电器触电电路，电灯电路。又如如果要控制 100A 的电路，就要用到交流接触器。

②模拟量

模拟量的电压输出一般为 0～3V(工作电压为 3V 的 MCU)、0～5V(工作电压为 5V 的 MCU)，电流输出约为 4～20mA。

一般情况下，MCU 的直接模拟量输出不足以驱动外部电路，需要通过各种放大电路加以放大再加以使用。

③通信量

主要有 RS-232、RS-485、CAN、IIC、SPI、USB 等。通信量一般要考虑到：通信速度、通信距离以及抗干扰程度。如 RS232C 的通信速度在 20Kbps 时的最大通信距离约为 30m；当通信速度低至 300bps 时，最远通信距离可达到 300m，但由于距离太长，此时的数据传输的准确性和稳定性难以保证。

对于抗干扰措施，可以采用差分信号进行数据传输，如 RS-422、RS-485、USB、CAN、10/100BASE-T 等。

对于 RS485 接口，其电气特性如下：

发送端：逻辑“1”以两线间的电压差为+(2~6)V 表示；逻辑“0”以两线间的电压差为-(2~6)V 表示。

接收端：RXD+比 RXD-高 200mV 以上即认为是逻辑“1”，RXD+比 RXD-低 200mV 以上即认为是逻辑“0”。

USB 系统采用四芯电缆。电缆的总长度一般不超过 5 m。其中 D+和 D-为传输数据信号，传输的信号采用差分信号。在全速和低速时，(D+)-(D-)>200mv 表示 1 信号，(D+)-(D-)<-200 mV 表示 0 信号。VBUS 和 GND 为电源线和地线，VBUS 由主机提供，一般为+5 V。

对于 10/100BASE-T 以太网，其传输也采用差分信号，使用四个信号线：两根用来发送，两根用来接收。每对信号线中的一根承载 0～+2.5V 的信号电压，而另一根承载的电压是 0～-2.5V，因此就可以产生一个 5Vpp 的信号差。

（2）输入

①开关量

包括机械式的开关量输入、电子式的开关量输入等等。对于开关量的采集要考虑到对被测试对象是否有干扰，以及被测对象会不会对其他对象产生干扰。对于模拟量也可以通过 A/D 转换把模拟量处理成开关量来使用。

②模拟量

一般通过各种传感器来获取模拟量。传感器取得的值一般在 mV、mA 或μV、μA 的级别，所以通常也需要通过放大器进行放大之后在处理。如果要在中断中对模拟量进行处理，要注意实时性的问题。

3）基本硬件测试实验

与控制相关的；收集的资料；原有的资料。

规范 I-5　工程交付前的工作（单片机方面）

（1）考虑热复位和冷复位，防止死机现象，即使是有休眠状态的设备也要考虑该问题。在重要的阶段将变量值写到 flash。

（2）冷复位（Restart）：单片机从没加电到加上电源，而自动产生的复位称为冷复位。热复位（Reset）：单片机在已经通电的情况下，给它一个复位信号，称为热复位。冷复位会使单片机的特殊功能寄存器（SFR）和数据存储器（RAM）的内容都改变；而热复位只是特殊功能寄存器（SFR）的内容改变而单片机的内部数据存储器（RAM）的内容不变。

（3）在程序合适的地方加看门狗。

（4）重视对一些特殊的模块定期做初始化，如 LCD、LED、IC 卡和时钟模块等。

（5）在适当的时间将整个系统人工复位一下。

（6）充分利用指示灯，如运行、故障、报警指示灯等；充分利用显示器件，如 LCD 等。在主循环之前就要对指示灯进行干预。

（7）加上保密字节。