Keil C51 μVision4的使用

和春技术学院 资工系 李鸿鹏老师编写

教育人员或学生可任意下载使用或連结但不得有商业行为

Keil C51 μVision4的使用

Keil C51 μVision4是美国Keil Software公司出品的51系列兼容单芯片C语言整合发展环境(Integrated Development Environment, IDE)，与汇编语言相比，C语言在功能上、架构性、可讀性、可维护性上有明显的优势，因而易学易用。用过汇编语言后再使用C语言來开发，会更加体会深刻。目前最新的版本是C51 Version 9.00 Release，也就是所谓的μVision4。

Keil C51软件提供丰富的函數库和功能强大的除错工具，及全窗口界面。另外重要的一点，只要看一下编译后生成的汇编语言代码，就能体会到Keil C51生成的目标代码效率非常之高，多數语句生成的汇编语言代码很紧凑，容易理解。在开发大型软件时更能体现高级语言的便利与优势。

Keil C51 μVision4较之前的版本，新增了下面几个功能：

多重的监控和弹性的窗口管理系统。

系统查看器(System Viewer) - 显示设备周边缓存器的讯息。

除错恢復检视(Debug Restore Views) - 建立和储存多重除错窗口设计。

多项目工作区(Multi-Project Workspace) - 与许多项目简化工作。

源码和解组译連结(Source and Disassembly Linking) - 解组译窗口和源码窗口完全同步使程序除错和光标导航较容易。

内存窗口固定(Memory Window Freeze) - 储存目前内存窗口检视允许容易在不同的点及时比较。

设备模拟 - 更新支持很多新设备(例如Infineon XC88x，SiLABS C8051Fxx， Atmel SAM7/9和从Luminary, NXP, and Toshiba 來的Cortex M3 MCUs)。

支持硬件除错转接器(Support for Hardware debug adapters) - 包括ADI。miDAS-Link，Atmel SAM-ICE，Infineon DAS, 和ST-Link。

新资料和指令追踪(New Data and instruction trace) - 对ARM和Cortex MCUs。

基于XML的项目文件(XML based Project files) - 建立，检视和修改项目如同容易可讀的XML本文档案一样。

串列的窗口 - 扩充到提供一个基本的100-VT终端机，ASCII 模式，混合模式，和十六进制模式检视。

拖放档案开启(Drag & Drop File Opening) - 档案拖进μVision4项目空间自动会被开启。

监控点和邏辑分析仪(Watchpoints and Logic Analyzer) - 现在更容易设定变數。

下面详细介绍Keil C51μVision4 IDE基本的功能和使用。

第一章 建立第一个Keil C51程序-使用C语言

随着单芯片技术的不断发展，以单芯片C语言为主的高级语言IDE，也不断的被开发出來，而且受到许多的单芯片爱好者和工程师所喜爱，更在学校中被广泛的使用在单芯片课程或微处理机课程教学上。Keil C51 μVision4 是众多单芯片IDE软件中优秀的软件之一，它支持许多不一样公司的MCS-51架构的芯片，它集编辑(Edit)，编译(Compiler)，模拟(Simulation)等于一体，同时还支持，PLM，汇编语言和C语言的程序设计，它的界面和微软的VC++的界面相似，易学易用，在程序除错，软件仿真方面也有很强大的功能。使用C51写好C程序，然后用C51的编译程序把写好的C程序编译为机器码，这样单芯片才能执行编写好的C程序。

下面结合MCS-51介绍单芯片C语言的优越性︰

不须完全懂得单芯片的硬件架构，也能够编写出符合硬件实际的专业水平的程序。

不懂完全得单芯片的指令集，也能够编写单芯片程序。

不同函數的數据实行覆盖，有效利用单芯片上有限的RAM空间。

提供auto, static, 和const等存储類型和专门针对8051单芯片的data, idata, pdata, data, 和code等存储類型，自动为变數合理地配置地址。

C语言提供复杂的數据類型（阵列(Array)、结构(Structure)、聯合(Union)、枚举(Enumeration)、指标(Pointer)等），极大地增强了程序处理能力和靈活性。

提供small, compact, 和large等编译模式，以适应单芯片上内存的大小。

中断服务程序的现场保护和恢復，中断向量表的填写，是直接与单芯片相关的，都是由C编译程序代办。

程序具有坚固性︰數据被破坏是导致程序执行異常的重要因素。C语言对數据进行了许多专业性的处理，避免了执行中间不正常的破坏。

提供常用的标准函數库，以供用户直接使用。

有严格的句法检查，错误很少。

可方便地接受多种实用程序的服务︰如单芯片上资源的初始化有专门的实用程序自动生成，简化用户程序设计，提升执行的安全性等等。

表头档案(header)中定义、說明复杂數据類型和函數原型，有利于程序的移植和支持单芯片的系列化产品的开发。

以上简单介绍了Keil C51软件，要使用Keil C51软件，必需先要安装它，这也是学习单芯片C语言所要求的第一步的建立学习环境。

使用者可到Keil C51的官方网站下载(https://www.keil.com/demo/eval/c51.htm)，在网页中填妥个人资料，就可下载免费版，如图1-1所示。此免费版有2K ROM大小的限制，不过一般通常在学校使用，使用2K ROM的大小就绰绰有余了，若使用超过2K ROM的大小的话，就必须购买正式版了。安装方法很简单，安装时只要点选C51V900.exe，就可自动执行安装了，其他后续版本也都一样，这裡就不做介绍了。

图1-1 Keil C51 μVision4官方下载网站

安装好C51后，C51的初始主画面如图1-2所示。

图1-2 Keil C51初始主画面

接着下面就让我们一起來建立自己的第一个单芯片C语言程序吧。请根据下面步骤

粗体的欄位要输入

一步步的來，你绝对可以在短时间内熟悉C51的使用。

(1) 点击Project(专案)选单，选择弹出的下拉式选单中的“New μVision Project…＂，如图1-3。接着弹出一个标准Windows档案对话窗口，如图1-4。在“储存于＂中选择您要存放的资料夹，一个项目最好存在一个资料夹内，若此资料夹不存在，请先建立它，或按“建立新资料夹”按钮以建立新资料夹。在“档名＂中输入您的第一个项目名称，这裡我们用“test1＂。“存档類型＂为uvproj，这是 Keil μVision4 项目文件预设的扩展名，以后只要直接点击此项目文件，即可打开此项目。

图1-3 New μVision Project选单

图1-4 New μVision Project选单

若第一次使用，则会出现图1-5的画面，若您要将您之前建立的旧版的项目档扩展名更名为新版的项目档扩展名，则按确定，否则按取消，用户可根据自己的需求选择不变或更名。因为第一次使用，按取消即可。

图1-5 选择旧版或新版的专案档扩展名

(2) 选择所要的单芯片型号，这裡选择常用的Ateml公司的AT89S51。目前Atmel已经停产AT89C51/AT89C52，改由AT89S51/AT89S52來取代之。AT89S系列较AT89C系列更为便宜，工作频率可高达33 MHz，且重复烧錄次數至少可达一千次以上，非常适合学校学习或工程师开发产品之用。所以本讲义裡的大部分程序都是基于此AT89S51芯片的，此时屏幕如图1-6所示。在右边图中的“Description”方块内，会简单的介绍AT89S51有什么功能及特点。点选OK按钮后，会出现图1-7，询问你是否需要拷贝标准的8051启动码程序(STARTUP.a51)到你的项目资料夹，并且将此档案加入项目“Copy Standard 8051 Startup Code to Project Folder and Add File to Project”，点选“是”后，就可以进行程序的编写了。

图1-6 选取芯片型号

图1-7 询问是否需要加入8051启动码

STARTUP.a51的主要工作是把包含idata、xdata、及pdata在内的内存区块清除为0，并且初始化递归指标。STARTUP.a51的内容在附錄1中，用户可自行參考。注意，若是编写汇编程序，则不需加入此启动程序。在完成上述的初始化程序之后，8051的控制权才会交给main()主程序开始执行用户的程序。

(3) 首先在项目中建立新的程序档案或加入旧程序档案。如果您没有现成的程序或是第一次使用，那么就要新建一个C程序档案。在C51中有一些程序的范例，但是在这裡我们还是以一个C程序为例介绍如何新建一个C程序，和如何加到您的第一个项目中吧。点击图1-8中1的新建文件的图标按钮，在2中出现一个新的文字编辑窗口，或是也能透过选单File/New或是按下快捷键Ctrl+n來实现。接着现在就能编写程序了。

图1-8 新建文件

下面是一段一颗LED在闪烁的程序，在上图2的文件编辑工作区内键入下面的程序，先不管程序的语法和意思，先看看如何把它存档，加入到项目中存档，和如何编译及执行。AT89X51.h的内容在附錄2中，若你用的单芯片是AT89S51，则用<AT89X52.h>。

#include <AT89X51.h>

main()

{

int i;

while(1)

{

P0\_0=1;

for(i=1;i<20000;i++);

P0\_0=0;

for(i=1;i<20000;i++);

}

}

(4) 点击图1-9中的储存档案图标按钮，也能用选单File/Save或按快捷键Ctrl+S，则出现图1-10的窗口。把此程序命名为test1.c，储存在项目所在的资料夹中，再按储存钮。这个时候您会发现程序单字有了不同的颜色，这表示Keil的C语言语法检查开始作用了。

1

2

项目窗口

编辑窗口

输出窗口

图1-9 储存档案

图1-10储存c档案

(5) 滑鼠在屏幕左边的Source Group1资料夹图示上右键单击，弹出一选单，如图1-11所示，在这裡能做在项目中增加减少档案等操作。选“Add Files to Group ｀Source Group 1…＂弹出档案窗口，选择刚刚储存的档案，按下Add按钮，将此c档案加入到

1.选取专案资料夹

2.输入全部档名

3.按储存钮

此专案中。按下close按钮，关闭档案窗口，如图1-12所示，则此test1.c程序档案已加到此项目中了，如图1-13所示。

图1-11 选取“Add Files to Group ‘Source Group 1’…＂

图1-12 选取要加入到项目中的c档案

选取c档案

图1-13 把c档案加入到项目中了

若用户写的是汇编语言档案，那必须存成\*.a51或\*.asm档，然后将汇编语言档案加入到项目中。

(6) 接下來要來做一些基本的设定选项的工作。滑鼠在屏幕左边的Target 1资料夹图示上右键单击，弹出一选单，如图1-14所示，然后选取“Options for target ‘Target 1’…”。

加入到项目了

图1-14 选取“Options for target ‘Target 1’…”

在图1-15的Target标签页中，更改所选用单芯片的工作频率为12。MCS-51系列单芯片系统一般常选用11.059 MHz或12 MHz。前者适用于产生各种标准的鲍率(baud rate)，后者的一个机器周期为1 μs，便于产生精确延迟时间。本程序中假设使用频率为12 MHz的晶体振荡频率。另外，勾选Use On-Chip ROM(0x0-0xFFF)，以使用单芯片上的Flash ROM。

图1-15 在Target标签页中，更改所选用单芯片的工作频率

再來切换到Output标签页，只要勾选“Create HEX File”就好了，以产生烧錄档，如图2-13所示。如果用户只是单纯的做練习，那就省略此步骤了。若要更改存放目的档的资料夹，则点击“Select Folder for Objects…”，若要更改编译后的主档名，则在“Name of Executable:”右边的空格内输入主档名即可，一般而言，这2个选项都采用默认值，用户不需更改他们。

图1-16 勾选“Create HEX File”，以产生烧錄档

(7) 完成基本的选项设定后，下面就剩下编译执行了。先來看这三个按钮，这三个都是编译按钮。按钮是用于编译目前工作区的档案但不做連结(Link)，按钮是用于编译整个项目文件并連结，如果之前编译过一次之后档案没有做任何编辑的话，这个时候再点击是不会再次重新编译的。按钮是重新编译整个项目文件并連结，每点击一次均会再次编译連结一次，不管程序是否有改变。是停止编译按钮，只有点击了前三个中的任一个，停止按钮才会生效。或是从选单Project中，也可执行编译，在此按下“Build target”或F7快捷键，如图1-17所示。

图1-17 编译项目

编译完成后，在下方的Build Output区域中，可看到编译的讯息，如图1-18所示。若有出现错误讯息，则再根据错误讯息，回到程序中修改，编译完全正确后，才能产生正确的烧錄档test1.hex。

编译正确

图1-18 项目编译正确

(8) 选取选单Debug/Start/Stop Debug Session，或按快捷键Ctrl+F5，或按在档案工具列右边有一个小放大镜的按钮，则进入除错(Debug)模式，并显示不同的工作窗口，如图1-19所示。进入除错模式之前，会先出现一个小窗口，告诉你目前用的版本是免费的评估版，有2K ROM大小的限制，点击确定即可进入除错模式。在除错工具列中，(Reset)按钮表示重置单芯片，并使程序回到最开头处执行。(Run)按钮表示执行，(Stop)按钮表示停止，当程序处于执行狀态时，停止按钮才有效。(Step Into)按钮表示单步执行会进入函數内，(Step Over)按钮表示单步执行不会进入函數内，(Step Out)按钮表示離开函數，(Run to Cursor)按钮表示执行到光标所在处。

图1-19 程序除错画面

为了要检视输出结果是否正确，则必须叫出P0输出入埠观察输出结果。选取选单Peripherals/I/O-Ports/Port 0，如图1-20所示。

反组译窗口

缓存器窗口

呼叫堆栈窗口

命令窗口

图1-20 选取Port 0

出现Parallel Port 0小窗口，并显示每一个位的值，也可移动到其他位置观察，如图1-21所示。

图1-21 显示Port 0的窗口

打勾表示1

最后要准备执行此程序了，先单击重置按钮，让单芯片及程序回到最初狀态，再按下执行按钮后，则程序开始执行。我们可以看到Parallel Port 0窗口中的P0\_0位不断的被设定与清除。

图1-22显示Port 0的内容及停止程序执行

若要观察汇编程序编译后所产生的运算码，在ROM的存放情形，则选取选单View/Memory Windows/Memory 1，或按右下方的Memory 1按钮，然后在Address欄位内输入0x800或0800h，则Memory 1窗口从0x0800开始显示运算码，如图1-23所示。

P0\_0输出产生变化

图1-23 Memory 1窗口显示运算码

程序执行正确后，最后要停止程序执行回到档案编辑模式中，就要先按停止按钮再按开启\关闭除错模式按钮。若要关闭此专案，则选取选单Project/Close Project，关闭此专案。Keil C51 μVision4的C语言程序初次使用，到此告一段落，下一章我们看Keil C51 μVision4的汇编程序的初次使用。

第二章 建立第二个Keil C51程序-使用汇编语言

虽然以C语言或以汇编语言撰写MCS-51程序來相比，C语言在功能上、架构性、可讀性、可维护性上有明显的优势，因而易学易用。然而汇编语言所写出來的程序经编译(compiler)后，所产生出來的运算码或机械语言码所占用的内存，会比C语言所写出來的程序经编译后，所产生出來的运算码所占用的内存少，因此执行效率较高。C语言要经过编译后转成汇编语言，而转出來的汇编语言的内容的写法有时后让人觉得有划蛇添足的感觉。然而对程序如果要求要写得简单、直接、有效率的话，通常是会选择汇编语言。一般而言，对讀电子电机或资工系的学生，最好是C语言和汇编语言都要学。

接着下面就让我们一起來建立自己的第二个单芯片汇编程序吧。若您对上一章的操作已经有了初步认識的话，那你对下面的步骤，就会觉得更容易。

(1) 点击Project(专案)选单，选择弹出的下拉式选单中的“New μVision Project…＂，如图2-1。接着弹出一个标准Windows档案对话窗口，如图2-2。在“储存于＂中选择您要存放的资料夹，一个项目最好存在一个资料夹内，若此资料夹不存在，请先建立它，或按“建立新资料夹”按钮以建立新资料夹。在“档名＂中输入您的第二个项目名称，这裡我们用“test2＂。“存档類型＂为uvproj，这是 Keil μVision4 项目文件预设的扩展名，以后只要直接点击此项目文件，即可打开此项目。

图2-1 New μVision Project选单

图2-2 New μVision Project选单

因第二次使用，所以不会出现图1-5的画面。

(2) 选择所要的单芯片型号，这裡仍然选择常用的Ateml公司的AT89S51，此时屏幕如图2-3所示。在右边图中的“Description”方块内，会简单的介绍AT89S51有什么功能及特点。点选OK按钮后，会出现图2-4，询问你是否需要拷贝标准的8051启动码程序到你的项目资料夹，并且将此档案加入项目“Copy Standard 8051 Startup Code to Project Folder and Add File to Project”，因为汇编语言不需要加入此STARTUP.a51，点选“否”后，就可以进行程序的编写了。

图2-3 选取芯片型号

图2-4 询问是否需要加入8051启动码

(3) 首先在项目中建立新的程序档案或加入旧程序档案。如果您没有现成的程序或是第一次使用汇编语言，那么就要新建一个空白程序档案。在C51中有一些程序的范例，但是在这裡我们还是以一个组合程序为例介绍如何新建一个组合程序，和如何加到您的项目中吧。点击图2-5中1的新建文件的图标按钮，在2中出现一个新的文字编辑窗口，或是也能透过选单File/New或是按下快捷键Ctrl+n來实现。接着现在就能编写程序了。

图2-5 新建文件

下面是一个跑马灯的程序，在上图2的文件编辑工作区内键入下面的程序，先不管程序的语法和意思，先看看如何把它存档，加入到项目中存档，和如何编译及执行。

MOV A,#0FEH

START: MOV P1,A

RL A

ACALL DELAY

AJMP START

DELAY: MOV R0,#248

D1: MOV R1,#200

D2: DJNZ R1,D2

DJNZ R0,D1

RET

END

(4) 点击图2-6中的储存档案图标按钮，也能用选单File/Save或按快捷键Ctrl+S，则出现图2-7的窗口。把此程序命名为test2.a51，储存在项目所在的资料夹中，再按储存钮。这个时候您会发现程序单字有了不同的颜色，这表示Keil的汇编语言语法检查开始作用了。

1

2

项目窗口

编辑窗口

输出窗口

图2-6 储存档案

图2-7 储存a51档案

(5) 滑鼠在屏幕左边的Source Group1资料夹图示上右键单击，弹出一选单，如图2-8所示，在这裡能做在项目中增加减少档案等操作。选“Add Files to Group ｀Source Group 1…＂弹出档案窗口，选择刚刚储存的档案，按下Add按钮，将此.a51档案加入

1.选取专案资料夹

2.输入全部档名

3.按储存钮

到此专案中。按下close按钮，关闭档案窗口，如图2-9所示，则此test2.a51程序档案已加到此项目中了，如图2-10所示。

图2-8 选取“Add Files to Group ‘Source Group 1’…＂

图2-9 选取要加入到项目中的汇编语言档案

选取.a51档案

图2-10 把汇编语言档案加入到项目中了

(6) 接下來要來做一些基本的设定选项的工作，在每一个项目裡面都要设定一次。滑鼠在屏幕左边的Target 1资料夹图示上右键单击，弹出一选单，如图2-11所示，然后选取“Options for target ‘Target 1’…”，则出现图2-12的窗口。

加入到项目了

图2-11 选取“Options for target ‘Target 1’…”

在图2-12的Target标签页中，如同图1-15，更改所选用单芯片的工作频率为12，并勾选Use On-Chip ROM(0x0-0xFFF)，以使用单芯片上的Flash ROM。

图2-12 在Target标签页中，更改所选用单芯片的工作频率

再來切换到Output标签页，只要勾选“Create HEX File”就好了，以产生烧錄档，如图2-13所示。如果用户只是单纯的做練习，那就省略此步骤了。若要更改存放目的档的资料夹，则点击“Select Folder for Objects…”，若要更改编译后的主档名，则在“Name of Executable:”右边的空格内输入主档名即可，一般而言，这2个选项都采用默认值，用户不需更改他们。

图2-13 勾选“Create HEX File”，以产生烧錄档

(7) 完成基本的选项设定后，下面就剩下编译执行了。在此按下按钮，“Build target”，或F7快捷键，如图2-14所示。

图2-14 编译项目

编译完成后，在下方的Build Output区域中，可看到编译的讯息，如图2-15所示。若有出现错误讯息，则再根据错误讯息，回到程序中修改，编译完全正确后，才能产生正确的烧錄档test2.hex。

图2-15 项目编译正确

(8) 选取选单Debug/Start/Stop Debug Session，或按快捷键Ctrl+F5，或按在档案工具列右边有一个小放大镜的按钮，则进入除错(Debug)模式，并显示不同的工作窗口，如图2-16所示。进入除错模式之前，同样地会先出现一个小窗口，告诉你目前用的版本是免费的评估版，有2K ROM大小的限制，点击确定即可进入除错模式。在除错工具列中，(Reset)按钮表示重置单芯片，并使程序回到最开头处执行。(Run)按钮表示执行，(Stop)按钮表示停止，当程序处于执行狀态时，停止按钮才有效。(Step Into)按钮表示单步执行会进入函數内，(Step Over)按钮表示单步执行不会进入函數内，(Step Out)按钮表示離开函數，(Run to Cursor)按钮表示执行到光标所在处。

编译正确

图2-16 程序除错画面

为了要检视输出结果是否正确，则必须叫出P1输出入埠观察输出结果。选取选单Peripherals/I/O-Ports/Port 1，如图2-17所示。

图2-17 选取Port 1

反组译窗口

缓存器窗口

呼叫堆栈窗口

命令窗口

出现Parallel Port 1小窗口，并显示每一个位的值，也可移动到其他位置观察，如图2-18所示。

图2-18 显示Port 1的窗口

最后要准备执行此程序了，先单击重置按钮，让单芯片及程序回到最初狀态，再按下执行按钮后，则程序开始执行。我们可以看到Parallel Port 1窗口中的P1\_0位到P1\_7位不断的被设定与清除，如图2-19所示。

打勾表示1

图2-19 显示Port 1的内容

若要观察汇编程序编译后所产生的运算码，在ROM的存放情形，则选取选单View/Memory Windows/Memory 1，或按右下方的Memory 1按钮，然后在Address欄位内输入0，则Memory 1窗口从0x0000开始显示运算码，如图2-20所示。

P1埠输出产生变化

图2-20 Memory 1窗口显示运算码

程序执行正确后，最后要停止程序执行回到档案编辑模式中，就要先按停止按钮再按开启\关闭除错模式按钮。若要关闭此专案，则选取选单Project/Close Project，关闭此专案。Keil C51 μVision4的汇编程序使用，到此告一段落。其他进一步的使用，请參考相关书籍或Keil C51的Help說明。

附錄 1

STARTUP.a51

$NOMOD51

;------------------------------------------------------------------------------

; This file is part of the C51 Compiler package

; Copyright (c) 1988-2005 Keil Elektronik GmbH and Keil Software, Inc.

; Version 8.01

;

; \*\*\* <<< Use Configuration Wizard in Context Menu >>> \*\*\*

;------------------------------------------------------------------------------

; STARTUP.A51: This code is executed after processor reset.

;

; To translate this file use A51 with the following invocation:

;

; A51 STARTUP.A51

;

; To link the modified STARTUP.OBJ file to your application use the following

; Lx51 invocation:

;

; Lx51 your object file list, STARTUP.OBJ controls

;

;------------------------------------------------------------------------------

;

; User-defined <h> Power-On Initialization of Memory

;

; With the following EQU statements the initialization of memory

; at processor reset can be defined:

;

; <o> IDATALEN: IDATA memory size <0x0-0x100>

; <i> Note: The absolute start-address of IDATA memory is always 0

; <i> The IDATA space overlaps physically the DATA and BIT areas.

IDATALEN EQU 80H

;

; <o> XDATASTART: XDATA memory start address <0x0-0xFFFF>

; <i> The absolute start address of XDATA memory

XDATASTART EQU 0

;

; <o> XDATALEN: XDATA memory size <0x0-0xFFFF>

; <i> The length of XDATA memory in bytes.

XDATALEN EQU 0

;

; <o> PDATASTART: PDATA memory start address <0x0-0xFFFF>

; <i> The absolute start address of PDATA memory

PDATASTART EQU 0H

;

; <o> PDATALEN: PDATA memory size <0x0-0xFF>

; <i> The length of PDATA memory in bytes.

PDATALEN EQU 0H

;

;</h>

;------------------------------------------------------------------------------

;

;<h> Reentrant Stack Initialization

;

; The following EQU statements define the stack pointer for reentrant

; functions and initialized it:

;

; <h> Stack Space for reentrant functions in the SMALL model.

; <q> IBPSTACK: Enable SMALL model reentrant stack

; <i> Stack space for reentrant functions in the SMALL model.

IBPSTACK EQU 0 ; set to 1 if small reentrant is used.

; <o> IBPSTACKTOP: End address of SMALL model stack <0x0-0xFF>

; <i> Set the top of the stack to the highest location.

IBPSTACKTOP EQU 0xFF +1 ; default 0FFH+1

; </h>

;

; <h> Stack Space for reentrant functions in the LARGE model.

; <q> XBPSTACK: Enable LARGE model reentrant stack

; <i> Stack space for reentrant functions in the LARGE model.

XBPSTACK EQU 0 ; set to 1 if large reentrant is used.

; <o> XBPSTACKTOP: End address of LARGE model stack <0x0-0xFFFF>

; <i> Set the top of the stack to the highest location.

XBPSTACKTOP EQU 0xFFFF +1 ; default 0FFFFH+1

; </h>

;

; <h> Stack Space for reentrant functions in the COMPACT model.

; <q> PBPSTACK: Enable COMPACT model reentrant stack

; <i> Stack space for reentrant functions in the COMPACT model.

PBPSTACK EQU 0 ; set to 1 if compact reentrant is used.

;

; <o> PBPSTACKTOP: End address of COMPACT model stack <0x0-0xFFFF>

; <i> Set the top of the stack to the highest location.

PBPSTACKTOP EQU 0xFF +1 ; default 0FFH+1

; </h>

;</h>

;------------------------------------------------------------------------------

;

; Memory Page for Using the Compact Model with 64 KByte xdata RAM

; <e>Compact Model Page Definition

;

; <i>Define the XDATA page used for PDATA variables.

; <i>PPAGE must conform with the PPAGE set in the linker invocation.

;

; Enable pdata memory page initalization

PPAGEENABLE EQU 0 ; set to 1 if pdata object are used.

;

; <o> PPAGE number <0x0-0xFF>

; <i> uppermost 256-byte address of the page used for PDATA variables.

PPAGE EQU 0

;

; <o> SFR address which supplies uppermost address byte <0x0-0xFF>

; <i> most 8051 variants use P2 as uppermost address byte

PPAGE\_SFR DATA 0A0H

;

; </e>

;------------------------------------------------------------------------------

; Standard SFR Symbols

ACC DATA 0E0H

B DATA 0F0H

SP DATA 81H

DPL DATA 82H

DPH DATA 83H

NAME ?C\_STARTUP

?C\_C51STARTUP SEGMENT CODE

?STACK SEGMENT IDATA

RSEG ?STACK

DS 1

EXTRN CODE (?C\_START)

PUBLIC ?C\_STARTUP

CSEG AT 0

?C\_STARTUP: LJMP STARTUP1

RSEG ?C\_C51STARTUP

STARTUP1:

IF IDATALEN <> 0

MOV R0,#IDATALEN - 1

CLR A

IDATALOOP: MOV @R0,A

DJNZ R0,IDATALOOP

ENDIF

IF XDATALEN <> 0

MOV DPTR,#XDATASTART

MOV R7,#LOW (XDATALEN)

IF (LOW (XDATALEN)) <> 0

MOV R6,#(HIGH (XDATALEN)) +1

ELSE

MOV R6,#HIGH (XDATALEN)

ENDIF

CLR A

XDATALOOP: MOVX @DPTR,A

INC DPTR

DJNZ R7,XDATALOOP

DJNZ R6,XDATALOOP

ENDIF

IF PPAGEENABLE <> 0

MOV PPAGE\_SFR,#PPAGE

ENDIF

IF PDATALEN <> 0

MOV R0,#LOW (PDATASTART)

MOV R7,#LOW (PDATALEN)

CLR A

PDATALOOP: MOVX @R0,A

INC R0

DJNZ R7,PDATALOOP

ENDIF

IF IBPSTACK <> 0

EXTRN DATA (?C\_IBP)

MOV ?C\_IBP,#LOW IBPSTACKTOP

ENDIF

IF XBPSTACK <> 0

EXTRN DATA (?C\_XBP)

MOV ?C\_XBP,#HIGH XBPSTACKTOP

MOV ?C\_XBP+1,#LOW XBPSTACKTOP

ENDIF

IF PBPSTACK <> 0

EXTRN DATA (?C\_PBP)

MOV ?C\_PBP,#LOW PBPSTACKTOP

ENDIF

MOV SP,#?STACK-1

; This code is required if you use L51\_BANK.A51 with Banking Mode 4

;<h> Code Banking

; <q> Select Bank 0 for L51\_BANK.A51 Mode 4

#if 0

; <i> Initialize bank mechanism to code bank 0 when using L51\_BANK.A51 with Banking Mode 4.

EXTRN CODE (?B\_SWITCH0)

CALL ?B\_SWITCH0 ; init bank mechanism to code bank 0

#endif

;</h>

LJMP ?C\_START

END

附錄 2

AT89X51.H

/\*--------------------------------------------------------------------------

Header file for the low voltage Flash Atmel AT89C51 and AT89LV51.

Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.

All rights reserved.

--------------------------------------------------------------------------\*/

#ifndef \_\_AT89X51\_H\_\_

#define \_\_AT89X51\_H\_\_

/\*------------------------------------------------

Byte Registers

------------------------------------------------\*/

sfr P0 = 0x80;

sfr SP = 0x81;

sfr DPL = 0x82;

sfr DPH = 0x83;

sfr PCON = 0x87;

sfr TCON = 0x88;

sfr TMOD = 0x89;

sfr TL0 = 0x8A;

sfr TL1 = 0x8B;

sfr TH0 = 0x8C;

sfr TH1 = 0x8D;

sfr P1 = 0x90;

sfr SCON = 0x98;

sfr SBUF = 0x99;

sfr P2 = 0xA0;

sfr IE = 0xA8;

sfr P3 = 0xB0;

sfr IP = 0xB8;

sfr PSW = 0xD0;

sfr ACC = 0xE0;

sfr B = 0xF0;

/\*------------------------------------------------

P0 Bit Registers

------------------------------------------------\*/

sbit P0\_0 = 0x80;

sbit P0\_1 = 0x81;

sbit P0\_2 = 0x82;

sbit P0\_3 = 0x83;

sbit P0\_4 = 0x84;

sbit P0\_5 = 0x85;

sbit P0\_6 = 0x86;

sbit P0\_7 = 0x87;

/\*------------------------------------------------

PCON Bit Values

------------------------------------------------\*/

#define IDL\_ 0x01

#define STOP\_ 0x02

#define PD\_ 0x02 /\* Alternate definition \*/

#define GF0\_ 0x04

#define GF1\_ 0x08

#define SMOD\_ 0x80

/\*------------------------------------------------

TCON Bit Registers

------------------------------------------------\*/

sbit IT0 = 0x88;

sbit IE0 = 0x89;

sbit IT1 = 0x8A;

sbit IE1 = 0x8B;

sbit TR0 = 0x8C;

sbit TF0 = 0x8D;

sbit TR1 = 0x8E;

sbit TF1 = 0x8F;

/\*------------------------------------------------

TMOD Bit Values

------------------------------------------------\*/

#define T0\_M0\_ 0x01

#define T0\_M1\_ 0x02

#define T0\_CT\_ 0x04

#define T0\_GATE\_ 0x08

#define T1\_M0\_ 0x10

#define T1\_M1\_ 0x20

#define T1\_CT\_ 0x40

#define T1\_GATE\_ 0x80

#define T1\_MASK\_ 0xF0

#define T0\_MASK\_ 0x0F

/\*------------------------------------------------

P1 Bit Registers

------------------------------------------------\*/

sbit P1\_0 = 0x90;

sbit P1\_1 = 0x91;

sbit P1\_2 = 0x92;

sbit P1\_3 = 0x93;

sbit P1\_4 = 0x94;

sbit P1\_5 = 0x95;

sbit P1\_6 = 0x96;

sbit P1\_7 = 0x97;

/\*------------------------------------------------

SCON Bit Registers

------------------------------------------------\*/

sbit RI = 0x98;

sbit TI = 0x99;

sbit RB8 = 0x9A;

sbit TB8 = 0x9B;

sbit REN = 0x9C;

sbit SM2 = 0x9D;

sbit SM1 = 0x9E;

sbit SM0 = 0x9F;

/\*------------------------------------------------

P2 Bit Registers

------------------------------------------------\*/

sbit P2\_0 = 0xA0;

sbit P2\_1 = 0xA1;

sbit P2\_2 = 0xA2;

sbit P2\_3 = 0xA3;

sbit P2\_4 = 0xA4;

sbit P2\_5 = 0xA5;

sbit P2\_6 = 0xA6;

sbit P2\_7 = 0xA7;

/\*------------------------------------------------

IE Bit Registers

------------------------------------------------\*/

sbit EX0 = 0xA8; /\* 1=Enable External interrupt 0 \*/

sbit ET0 = 0xA9; /\* 1=Enable Timer 0 interrupt \*/

sbit EX1 = 0xAA; /\* 1=Enable External interrupt 1 \*/

sbit ET1 = 0xAB; /\* 1=Enable Timer 1 interrupt \*/

sbit ES = 0xAC; /\* 1=Enable Serial port interrupt \*/

sbit ET2 = 0xAD; /\* 1=Enable Timer 2 interrupt \*/

sbit EA = 0xAF; /\* 0=Disable all interrupts \*/

/\*------------------------------------------------

P3 Bit Registers (Mnemonics & Ports)

------------------------------------------------\*/

sbit P3\_0 = 0xB0;

sbit P3\_1 = 0xB1;

sbit P3\_2 = 0xB2;

sbit P3\_3 = 0xB3;

sbit P3\_4 = 0xB4;

sbit P3\_5 = 0xB5;

sbit P3\_6 = 0xB6;

sbit P3\_7 = 0xB7;

sbit RXD = 0xB0; /\* Serial data input \*/

sbit TXD = 0xB1; /\* Serial data output \*/

sbit INT0 = 0xB2; /\* External interrupt 0 \*/

sbit INT1 = 0xB3; /\* External interrupt 1 \*/

sbit T0 = 0xB4; /\* Timer 0 external input \*/

sbit T1 = 0xB5; /\* Timer 1 external input \*/

sbit WR = 0xB6; /\* External data memory write strobe \*/

sbit RD = 0xB7; /\* External data memory read strobe \*/

/\*------------------------------------------------

IP Bit Registers

------------------------------------------------\*/

sbit PX0 = 0xB8;

sbit PT0 = 0xB9;

sbit PX1 = 0xBA;

sbit PT1 = 0xBB;

sbit PS = 0xBC;

sbit PT2 = 0xBD;

/\*------------------------------------------------

PSW Bit Registers

------------------------------------------------\*/

sbit P = 0xD0;

sbit F1 = 0xD1;

sbit OV = 0xD2;

sbit RS0 = 0xD3;

sbit RS1 = 0xD4;

sbit F0 = 0xD5;

sbit AC = 0xD6;

sbit CY = 0xD7;

/\*------------------------------------------------

Interrupt Vectors:

Interrupt Address = (Number \* 8) + 3

------------------------------------------------\*/

#define IE0\_VECTOR 0 /\* 0x03 External Interrupt 0 \*/

#define TF0\_VECTOR 1 /\* 0x0B Timer 0 \*/

#define IE1\_VECTOR 2 /\* 0x13 External Interrupt 1 \*/

#define TF1\_VECTOR 3 /\* 0x1B Timer 1 \*/

#define SIO\_VECTOR 4 /\* 0x23 Serial port \*/

#endif