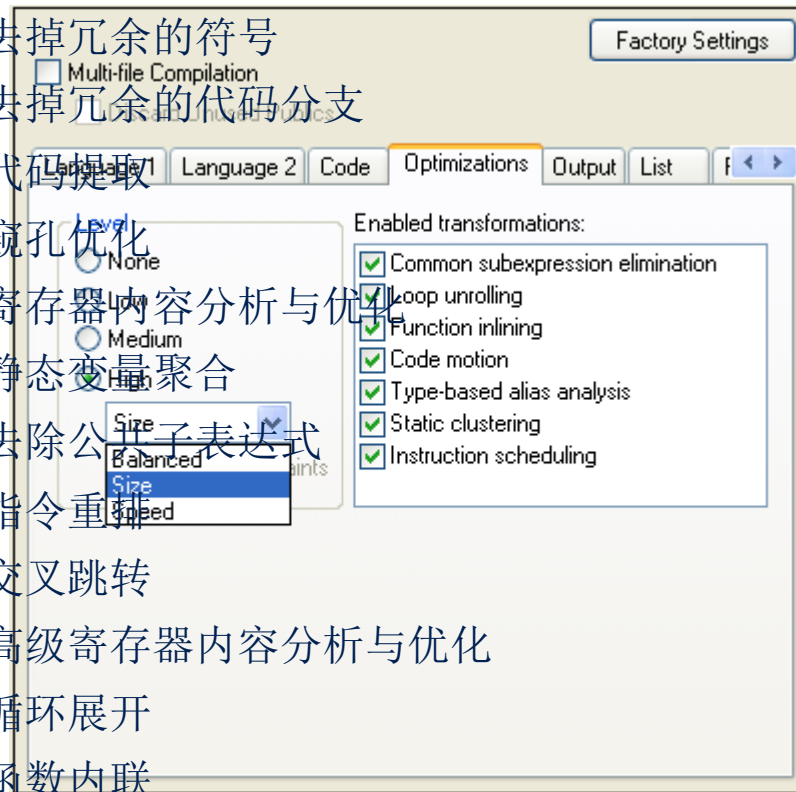


STM32 IAR 优化选项介绍

IAR优化选项

- 无优化
- 低等级优化
- 中等优化
- 高等级优化
 - 平衡
 - 代码量优先（最少代码）
 - 代码执行速度优先（最快运行速度）
- 可选项：
 - 去掉公共子表达式
 - 循环展开
 - 函数内联
 - 代码移动
 - 基于类型的别名分析
 - 静态变量重组
 - 指令重排

- 分析并优化无用的计算结果
- 去掉无用的代码
- 去掉冗余的符号
- 去掉冗余的代码分支
- 代码提取
- 窥孔优化
- 寄存器内容分析与优化
- 静态变量聚合
- 去除公共子表达式
- 指令重排
- 交叉跳转
- 高级寄存器内容分析与优化
- 循环展开
- 函数内联
- 移动代码
- 基于类型的别名分析



可选择的优化选项:

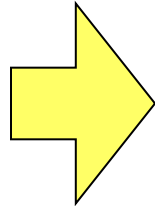
- 公共子表达式压缩
- 循环展开
- 函数内联
- 循环不变量外提
- 基于类型的别名分析
- 静态变量重组
- 指令规划

公共子表达式压缩

(Common sub-expression elimination)

4

a=b*c+g
d=b*c*d



tmp=b*c
a=tmp+g
d=tmp*d

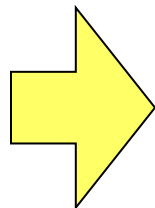
- 优化既可以减少生成的代码，也可以缩短执行时间
- 优化针对于数组索引计算、大量重复的宏计算等。
- 优化后的代码可能会比较难进行调试

可选择的优化选项:

- 公共子表达式压缩
- **循环展开**
- 函数内联
- 循环不变量外提
- 基于类型的别名分析
- 静态变量重组
- 指令规划

循环展开 (Loop Unrolling)

```
/* copy 20 elements */  
for(i=0; i<20; ++i)  
{  
    a[i]=b[i];  
}
```



```
/* unrolled four times */  
for (i=0; i<20; i+=4)  
{  
    a[i]=b[i];  
    a[i+1]=b[i+1];  
    a[i+2]=b[i+2];  
    a[i+3]=b[i+3];  
}
```

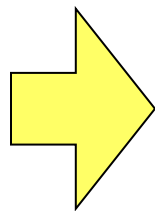
- 适合于小循环，因为小循环的循环体外运行开销比重比较大
- 优化可以缩短执行时间，但会增加代码的大小
- 优化后的代码可能会比较难进行调试

可选择的优化选项:

- 公共子表达式压缩
- 循环展开
- **函数内联**
- 循环不变量外提
- 基于类型的别名分析
- 静态变量重组
- 指令规划

函数内联 (Function Inlining)

```
int bar(int a)
{
    return a*7;
}
void foo()
{
    b = y+bar(x);
    ...
}
```



```
void foo()
{
    b = y+x*7;
    ...
}
```

- 可以减少函数调用时所产生的运行开销，缩短执行时间，但可能会增加代码的大小，一般情况下，选择代码量优先时使用该优化不会增加代码的大小
- 是否进行函数内联优化取决于编译器进行的试探性编译
- 优化后的代码可能会比较难进行调试

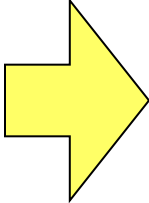
可选择的优化选项:

- 公共子表达式压缩
- 循环展开
- 函数内联
- **循环不变量外提**
- 基于类型的别名分析
- 静态变量重组
- 指令规划

循环不变量外提 (Loop-invariant Code Motion)

10

```
for(int i=0; i<n; i++)  
{  
    x=y+z;  
    a[i]=6*i+x*x;  
}  
  
x=y+z;  
t1=x*x;  
for(int i=0; i<n;i++)  
{  
    a[i]=6*i+t1;  
}
```



- 将循环处理中运算结果不变的部分提取到循环的外部
- 既可以节省代码空间也可以提高运行的效率
- 优化后的代码可能会比较难进行调试

可选择的优化选项:

- 公共子表达式压缩
- 循环展开
- 函数内联
- 循环不变量外提
- **基于类型的别名分析**
- 静态变量重组
- 指令规划

Type-based Alias Analysis

基于类型的别名分析

12

- 别名变量：两个或者更多的指针访问同一个地址。
- 对于标准C或C++程序，这种优化可以减少代码的大小，降低执行时间。
- 对于非标准的C或C++程序，可能会导致生成错误的代码。

```
short F(short *p1, long *p2)  
{  
    *p2 = 0;  
    *p1 = 1;  
    return *p2;  
}
```

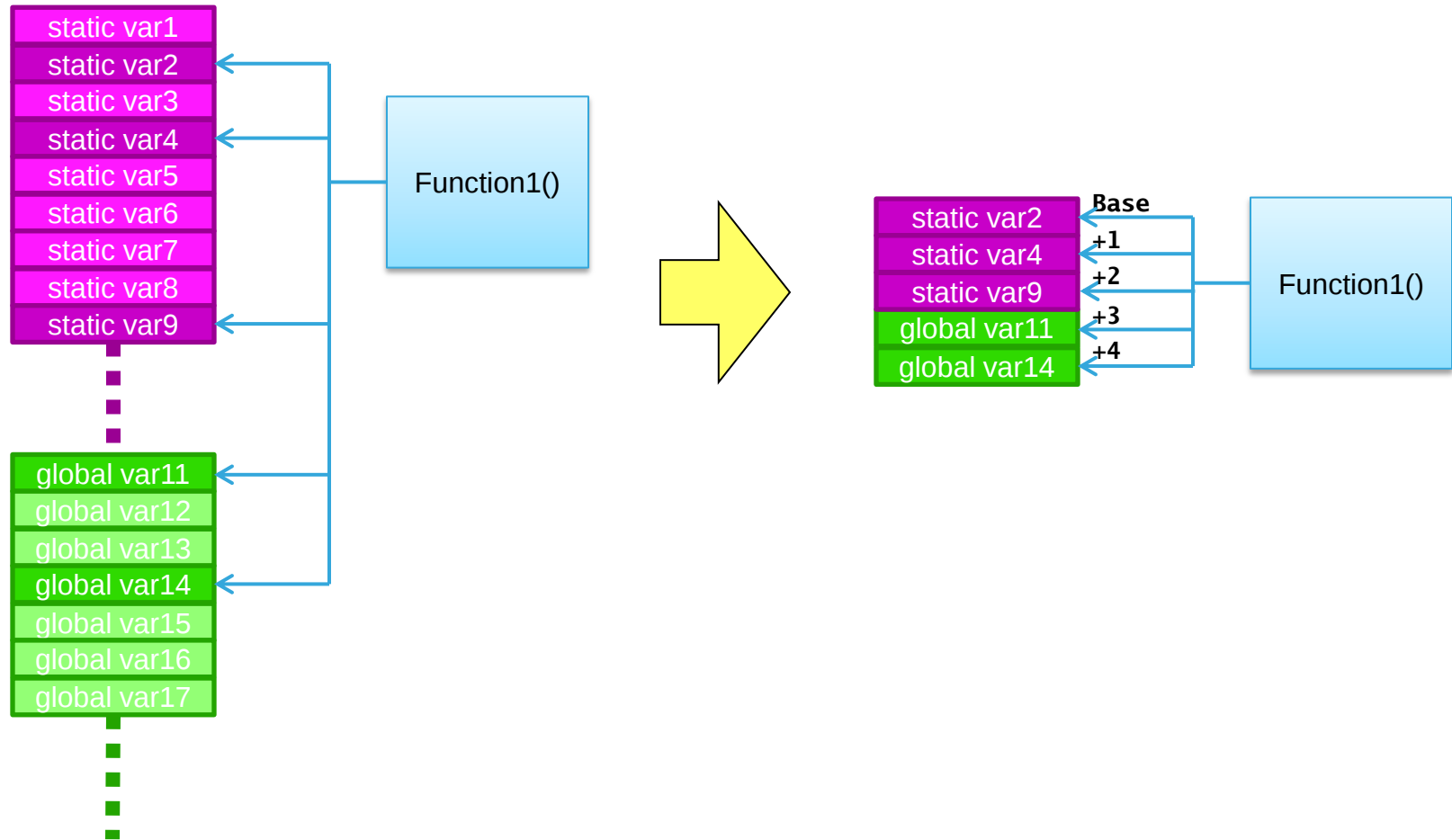
可选择的优化选项:

- 公共子表达式压缩
- 循环展开
- 函数内联
- 循环不变量外提
- 基于类型的别名分析
- **静态变量重组**
- 指令规划

Static Clustering

静态变量重组

14



可选择的优化选项:

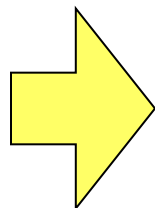
- 公共子表达式压缩
- 循环展开
- 函数内联
- 循环不变量外提
- 基于类型的别名分析
- 静态变量重组
- **指令规划**

Instruction Scheduling

指令规划

16

ADR R0, LocalData
STR R1, [R0]
LDR R2, [R0]
MUL R3, R4
LSR R5, R6, #6



ADR R0, LocalData
STR R1, [R0]
MUL R3, R4
LSR R5, R6, #6
LDR R2, [R0]

该指令可能需要几个时钟周期才能够完成，这会导致下一条指令多等待几个周期

指令重新规划后，该条指令不需要多等待若干周期

- 优化后的代码可能会比较难进行调试

其他代码优化

- 分析并优化无用的计算结果
(Live-dead analysis and optimization)
- 去掉无用的代码(Dead code elimination)
- 去掉冗余的符号(Redundant label elimination)
- 去掉冗余的代码分支(Redundant branch elimination)
- 代码提取(Code hoisting)
- 窥孔优化(Peephole optimization)
- 交叉跳转(Cross Jumping)
- 寄存器内容分析与优化(Register content analysis and optimization)
- 高级寄存器内容分析与优化
(Advanced register content analysis and optimization)

- 优化选项调整的次序：从低级到高级，从部份到全局
- 在C程序中使用“**#pragma optimize**”预处理器命令，可以精确调整单个C函数的编译器优化方向与级别。

避免编译器进行不恰当的优化

- 某些情况下不希望编译器进行的优化操作

- 调整运算或赋值的次序
- 删除编译器认为没有作用的代码
- 将变量分配在通用寄存器里

- 例如：

- 读写共享变量
- 读写外设寄存器端口
- 存在副作用的其他操作

使用 **volatile** !

• 使用**volatile**的情况

- 对象的值会在编译器不知道的情况下发生改变
 - 例如外设寄存器的值发生改变
- 操作具有副作用
 - 例如连续读或写某外设寄存器两次，硬件上具有特定的意义
- 对象被多个程序所共享
 - 操作系统中的多个任务
 - 主程序和中断服务程序

• 使用**volatile**对编译器的影响

- **volatile**变量不会被分配在通用寄存器中
- 对**volatile**变量的访问次序不会发生变化
- 对**volatile**变量的访问不会被删除

谢谢！

CONFIDENTIALITY OBLIGATIONS:

THIS DOCUMENT CONTAINS SENSITIVE INFORMATION.

IT IS CLASSIFIED "MICROCONTROLLERS, MEMORIES & SECURE MCUs (MMS) RESTRICTED AND ITS DISTRIBUTION IS SUBMITTED TO ST/MMS AUTHORIZATION

AT ALL TIMES YOU SHOULD COMPLY WITH THE FOLLOWING SECURITY RULES:

- DO NOT COPY OR REPRODUCE ALL OR PART OF THIS DOCUMENT
- KEEP THIS DOCUMENT LOCKED AWAY
- FURTHER COPIES CAN BE PROVIDED ON A "NEED TO KNOW BASIS", PLEASE CONTACT YOUR LOCAL ST SALES OFFICE OR DOCUMENT WRITTER.

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is registered trademark of STMicroelectronics
All other names are the property of their respective owners

© 2015 STMicroelectronics - All Rights Reserved

STMicroelectronics group of companies Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States.

www.st.com