doi: 10.7690/bgzdh.2021.07.008

IAR 开发环境下添加 SM9B100MAL 处理器支持的原理与方法

吴昌昊[1]，范　云[2]，黄　菊[1]，王文俊[1]，张自圃[1]，邵雨新[1]
（1. 中国兵器装备集团自动化研究所有限公司特种计算机事业部，四川 绵阳 621000；
2. 陆装审价中心，北京 100072）

摘要：为解决 SM9B100MAL 处理器官方未提供 IAR 开发环境支持的问题，提出为 IAR 添加处理器支持的方法。通过对 IAR 开发环境、C-SPY 调试器、Flash Loader 框架和设备描述配置等多方面的机理分析，给出参考配置与代码及其相关解释，展现添加处理器支持的过程。结果表明：添加支持后，即可在 IAR 开发环境中实现快速建立代码工程、一键下载程序、调试会话中结构化展示寄存器内容等功能。

关键词：IAR；C-SPY；Flash Loader；SM9B100MAL；开发环境；国产处理器；调试
中图分类号：TP311.1　　**文献标志码：**A

Principle and Method of Adding SM9B100MAL Device Support for IAR Development Environment

Wu Changhao[1], Fan Yun[2], Huang Ju[1], Wang Wenjun[1], Zhang Zipu[1], Shao Yuxin[1]
(1. *Department of Specialty Computer*, *Automation Research Institute Co*., *Ltd. of China South Industries Group Corporation*, *Mianyang* 621000, *China*;
2. *Audit Cenrter*, *Army Equipment Department*, *Beijing* 100072, *China*)

Abstract: To solve the problem that the vendor provides no support for developing embedded software based on SM9B100MAL platform in IAR integrated development environment (IDE), the method of adding new device support for IAR is provided. By analyzing the IAR IDE, C-SPY debugger, Flash Loader framework, and device description configuration, etc. Provide the reference configuration, codes and comments, and display the process for adding processor support. The results show that after adding support, under IAR IDE, realize the functions more efficiently, such as the fast creating code process, one-click downloading program, debugging dialog structure display register.

Keywords: IAR; C-SPY; Flash Loader; SM9B100MAL; IDE; homegrown processor; debug

0　引言

在目前半导体产业与 IT 产业的国产化热潮下，各个国产厂商都在积极推动国产处理器的发展，但产业整体发展程度仍与国外存在差距。由于缺乏完整的配套软硬件生态系统，国产处理器配合国外开发工具的组合是一种常见且将长期存在的开发组合。充分利用国外成熟的开发工具，可帮助基于国产处理器项目进行快速开发和验证，有利于加速国产替代产品的研发和迭代，因此有必要对国外开发工具支持的处理器范围进行扩展。

本文中目标来源于某个基于 SM9B100MAL 处理器的硬件平台设计项目。由于该处理器官方暂时还没有自研开发环境，也未提供对 IAR 开发环境的支持，为减少同时开发多个相关项目时涉及的开发环境种类，并提高在 IAR 开发环境中的开发效率，在 IAR 开发环境中添加该处理器的支持成为迫切需求。

笔者通过 IAR 内部功能原理、硬件架构、配置与程序编写等多个方面，描述添加 SM9B100MAL 处理器支持的原理与方法，展示添加支持后的运行效果，解决了在 IAR 开发环境中快速建立处理器平台工程、调试会话中寄存器观测、Flash 烧写等工程性问题，为该处理器增添了一种使用效果良好的开发环境。

1　IAR 开发环境

IAR 是瑞典 IAR Systems 公司为微处理器开发的一个集成开发环境，提供 ARM、AVR、MSP430、RISC-V 等多种架构的支持，为该公司最著名的产品之一[1]。

IAR 开发环境集成了编译器、汇编器、链接器、C-SPY 调试器、CPU 模拟器、C-STAT 静态分析器和 C-RUN 动态检查等工具，以一体式形式提供文件编辑、项目管理、在线调试、状态监视等功能[2]。

收稿日期：2021-03-24；**修回日期：**2021-04-28
作者简介：吴昌昊(1990—)，男，四川人，工程师，从事通用嵌入式平台软件开发、国产处理器平台软件开发、通信总线设计、实时操作系统设计研究。E-mail: wchcommander@qq.com。

1.1　C-SPY

C-SPY 是被 IAR 集成的嵌入式应用开发的高级语言调试器，配合 IAR 编译器与汇编器使用，使用户可在单一软件中完成开发和调试工作[3]。

C-SPY 在 IAR 开发环境中的定位如图 1 所示。

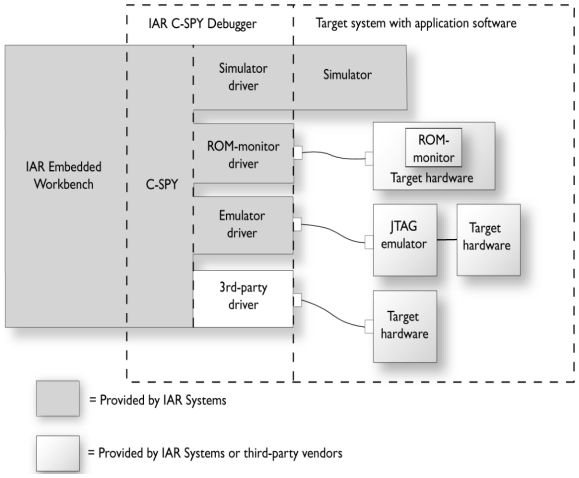

图 1　C-SPY 与 IAR 和目标软硬件关系

为实现 Flash 下载、寄存器定义与编组查看等功能，需与 C-SPY 及其子系统对接。

1.1.1　Macro

Macro(宏)系统属于 IAR C-SPY 内部子系统，用于定义一些调试需要的复杂动作，通常适用于自动化测试、硬件配置、硬件信号模拟和调试辅助等场景[3]。

该子系统在普通嵌入式应用调试和 Flash Loader 程序执行过程中都会被自动使用，C-SPY 会在执行调试或下载任务的不同阶段对特定的回调宏函数进行调用。

1.1.2　DDF

设备描述文件(device description file，DDF)，扩展名为.ddf，为 C-SPY 调试器提供目标平台硬件系统信息以及其他功能的描述[4]。

在调试普通嵌入式程序时，借助该文件的描述，通过 C-SPY 调试器可访问处理器寄存器、内存空间等资源。

1.2　Flash Loader

Flash Loader 为一种下载程序或文件到目标平台的代理程序，运行在目标平台的 RAM 上，使用文件 IO 机制，通过调试器读取主机端的文件内容并烧写到目标平台的指定 Flash 芯片中[5]。

Flash Loader 需 C-SPY 才能实现和使用，所以 C-SPY 的 Macro 等子系统同样适用于 Flash Loader。

1.2.1　基本原理

Flash 存在擦除、编程不同阶段，同时具有 page、block 等 RAM 没有的概念，数据导入 Flash 的原理与写入 RAM 不同，不能通过调试器直接下载到 RAM。

如图 2 所示，为了成功烧写 Flash，需在处理器的 RAM 上运行 Flash Loader 程序；再将烧写数据下载到如图 3 所示的 RAM 缓冲区；进而 Flash Loader 可读取缓冲区内容并将数据以符合 Flash 操作规则的方式写入 Flash 设备。

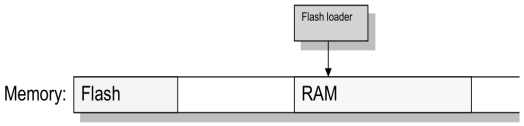

图 2　Flash Loader 被下载到目标平台 RAM

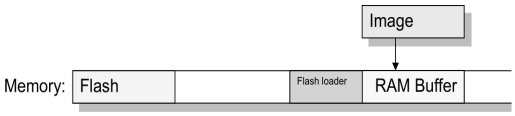

图 3　程序文件下载到目标平台 RAM 缓冲区

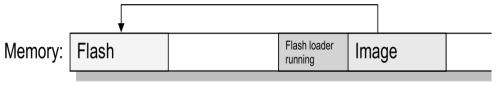

图 4　Flash Loader 将 RAM 缓冲区中数据烧写到 Flash

1.2.2　配置文件

Flash Loader 有 2 种必需配置文件：

1) Flash 设备配置文件：描述单个 Flash 设备特性，扩展名为.flash。

2) Flash 系统配置文件：以板卡或系统环境角度描述 Flash 设备烧写规则，扩展名为.board，为.flash 文件的父级配置文件，根据板卡实际包含的 Flash 设备数量和系统实际需要使用的目标 Flash 设备情况配置与.flash 文件的关联和烧写选项。

1.2.3　开发框架

Flash Loader 开发框架代码文件由 IAR 提供，通过统一的顶层 API，开发环境可忽略硬件平台底层差异，实现一致的烧写步骤。该框架在整个开发环境中的定位参考图 5，代码文件描述参考表 1。

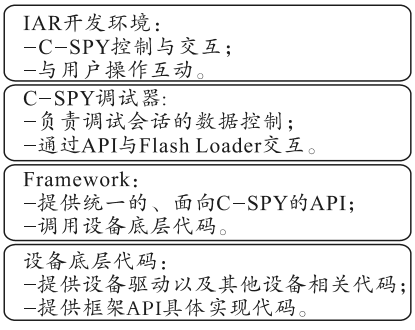

图 5　Flash Loader 开发框架定位

表 1 Flash Loader 开发框架代码文件描述

文件名	描述
flash_loader.c	框架基本源代码。提供必要基础符号、变量定义，提供 FlashInit()、FlashWrite()、FlashErase()等重要用户可实现接口的外部对接函数
flash_loader.h	框架基本源代码头文件。提供必要基础符号定义、函数声明等
flash_loader_extra.h	框架额外的符号定义。当 Flash Loader 需要使用一些特殊功能时，需要该头文件
flash_loader_asm.s	底层处理器依赖的框架汇编代码，提供启动代码、中断向量等
template\flash_config.h	Flash Loader 全局配置头文件模板

合理实现开发框架规定的函数即可实现 Flash 烧写与擦除功能，相关函数 API 参考表 2 的描述。

表 2 FlashWrite()函数参数含义

API	必需	描述
FlashWrite()	√	进行 Flash 指定地址范围的烧写操作
FlashErase()	√	进行 Flash 指定地址范围的块擦除操作
FlashInit()	√	烧写阶段开始前的初始化操作
FlashChecksum()		进行 Flash 数据校验计算
FlashSignoff()		进行 Flash 烧写和校验后的收尾工作

1.2.4 通用步骤

IAR 借助 C-SPY 和 Flash Loader 开发框架为不同硬件架构平台提供了一种通用的烧写过程，步骤如下：

1) C-SPY 读取并解析目标平台的 Flash 系统配置文件(扩展名.board)。

① 查看烧写阶段(pass)数量。若存在多个阶段，每个阶段都对应一个程序文件的地址范围，则不同范围会对应到不同的烧写阶段；若不存在多阶段，整个程序文件作为整体烧写。

② 每个阶段对应一个 Flash 设备配置文件(扩展名.flash)，以及其他可选配置。

2) 读取烧写阶段对应的 Flash 配置内容，以及 Flash Loader 程序。

3) C-SPY 下载 Flash Loader 到目标平台 RAM 中。如果烧写阶段设置了偏移值(offset)，那么所有程序文件中所有记录会进行重定位。

4) C-SPY 设置处理器 PC 寄存器到 Flash Loader 程序入口(最终会调用 FlashInit()函数)。

5) 参数与数据写入到 RAM 缓冲区。

6) Flash Loader 程序开始执行，在 FlashInit()函数执行后，当命中特殊断点时 C-SPY 重新获取平台控制权。

7) C-SPY 根据 Flash 的 page 与 block 大小和缓冲区大小将程序文件数据分为合适大小(总是 page 的整数倍大小)。

8) 在 block 被写入前，必须先被擦除。若需要擦除，进入下一步；若已经擦除，从步骤 11)继续。

9) RAM 参数设置为擦除块大小与地址。

10) C-SPY 设置处理器 PC 寄存器为 FlashErase()并开始执行。当操作完成时，命中断点。

11) C-SPY 写入数据到 RAM 缓冲区。

12) C-SPY 设置处理器 PC 寄存器为 FlashWrite()并开始执行。当操作完成时，命中断点。

13) 如果还有更多数据需要写入，返回步骤 8)。

14) 如果还有更过烧写步骤，返回步骤 2)。

15) C-SPY 读取应用程序文件内的调试信息。

16) C-SPY 设置处理器 PC 寄存器为应用程序的入口地址。

1.3 设备选择

在 IAR 开发环境中创建工程时需选择正确的构建目标设备，而设备若要作为可选项出现在 IAR 设备选择界面中，需借助设备选择文件(device selection file)的配置，用作 IAR 设置支持的入口。

设备选择文件同时与其他相关配置关联，即选中一个设备后，该设备的必要配置会自动出现在 IAR 工程中，其关系如图 6 所示。

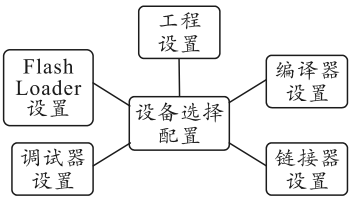

图 6 设备选择配置与其他配置的关联

2 SM9B100MAL

SM9B100MAL 是国微电子有限公司开发的 32 位 RISC 高性能嵌入式 SOPC 芯片。芯片处理器遵循 ARM 体系架构(5TEJ)，片上集成 16 KB 指令 Cache 和 16 KB 数据 Cache；拥有与 CPU 同频的 512 KB 高速程序存储器和 128 KB 高速数据存储器。产品内嵌 100 万门的 FPGA。片内总线采用 2 条独立的 32 位 AHB 总线分别作为控制总线和数据总线；内嵌入 512 KB SRAM 和 2 MB SPI_FLASH。片上集成 1553B、429、UART、LVDS、SPI、I2C、CAN、GPIO、FC、PWM 等丰富的外设控制接口[6]。芯片内部模块架构图 7 所示，可看出片内 SPI FLASH 与控制器 SPI0(即 SSI0)连接。

芯片内部处理器可直接访问的地址空间分为 TCM 空间和总线空间 2 级。前者地址空间分布如图 8 所示，后者地址空间参考分布如图 9 所示。

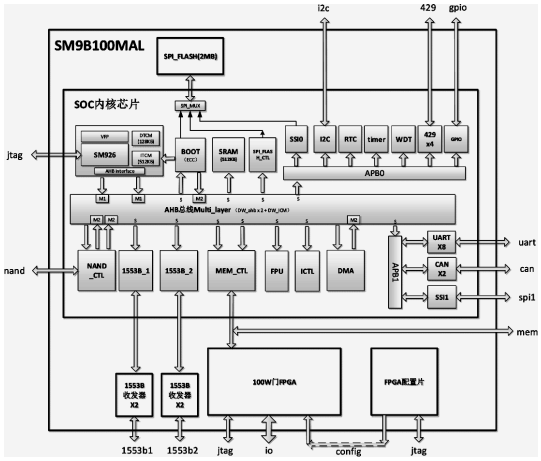

图 7　SM9B100MAL 内部架构

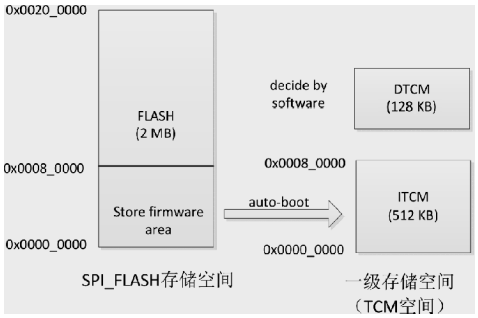

图 8　SM9B100MAL TCM 空间地址分布

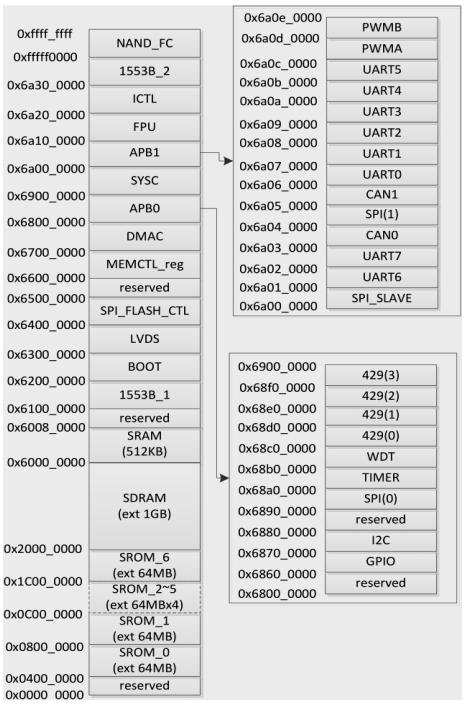

图 9　SM9B100MAL 总线空间地址分布

由图 8 可看出，芯片还存在代表片内集成的 SPI FLASH 的存储空间，但该空间的数据不能被 CPU 直接通过指令寻址得到。在设备上电或复位时，硬件会自动读取最多 512 KB(ITCM 区域最大容量)的数据到 ITCM 中，以便 CPU 执行代码和使用数据。FLASH 高于 512 KB 空间的内容不会被自动搬运，只能应用软件自行进行访问，此时只有通过 SPI 控制器，以正确的协议来操作 FLASH。

片内 FLASH 兼容 GD25Q16，使用兼容该型号的软件驱动配合 SPI 控制器驱动即可实现 CPU 对 FLASH 操作的功能。

3　添加设备支持

3.1　目标

1) IAR 新建代码工程后，在工程选项页面中能直接选择 SM9B100MAL 设备作为目标设备，并自动填充设备必需的构建设置。

2) IAR 调试会话中可方便地查看各个控制器的所有寄存器内容，无需手动敲入寄存器地址。

3) IAR 界面一键进行处理器内部 Flash 烧写、擦除操作，并能一次性实现下载和启动调试操作(Download & Debug)。

3.2　步骤简述

1) 选择厂商缩写名 SSMEC 作为 IAR 配置文件目录中代表厂商的文件夹名。

2) 选择 SM9B100 作为处理器设备名，因 SM9B100MAL 的 MAL 后缀不代表功能差异，这样命名普适性更强。

3) 创建<IAR 目录>\arm\config\devices\SSMEC 文件夹，在其中创建名为 sm9b100.menu、sm9b100.i79 2 个文件，用于设备选择信息描述。文件具体细节参考 3.3.1 与 3.3.2 节。

4) 创建 <IAR 目录>\arm\config\debugger\SSMEC 文件夹，在其中创建文件：sm9b100.ddf、sm9b100.dmac，用于 C-SPY 调试过程中的设备描述和功能定义，文件具体细节参考 3.3.3 与 3.3.4 节。

5) 创建<IAR 目录>\arm\config\linker\SSMEC 文件夹，在其中创建 sm9b100.icf 文件，作为 IAR 工程默认使用的链接配置。文件内容参考 3.3.5 节。

6) 基于目前已添加的 SM9B100 支持内容，建立用于测试的普通嵌入式应用工程(虽然此时的支持内容还缺少 Flash Loader 部分，但不影响测试工程的构建和使用，因为不使用 Flash Loader 功能)。编写和添加处理器启动、时钟配置、引脚配置、调试辅助、SPI 驱动、FLASH 驱动等方面的代码，完成基本 FLASH 操作测试，确认驱动正确性和之前步骤添加的各种处理器支持配置的正确性。

7) 底层代码测试完成后，创建<IAR 目

录>\arm\config\flashloader\SSMEC 文件夹，在其中创建文件：FlashSM9B100XXX.board、FlashSM9B100XXX_2M.flash、FlashSM9B100XXX.mac，作为 Flash Loader 程序的配置。文件内容参考 3.3.6 节。

8) 在任意位置新建 Flash Loader 工程(也可按照 IAR 参考代码在<IAR 目录>\arm\src\flashloader 目录下创建)。

9) 编写、修改 Flash Loader 代码、链接配置，并完成程序构建。

10) 构建生成程序文件(扩展名为.out)放入<IAR 目录>\arm\config\flashloader\SSMEC 目录。

11) 使用已有或新的基于 SM9B100 建立的 IAR 工程，启用 Flash Loader 功能，测试 Flash 烧写、擦除和一键下载调试功能是否正常。若正常，进入下一步；若不正常，返回步骤 9)。

12) 完成在 IAR 中添加 SM9B100 处理器支持。

3.3 实施细节

3.3.1 sm9b100.menu

该文件为 SM9B100 系列处理器的设备选择文件之一，用于在 IAR 界面中提供选择入口。存放路径为：<IAR 安装目录>\arm\config\devices\SSMEC(SSMEC 代表设备厂商)。

该设备选择文件中记录了芯片名称、.i79 文件(另一种设备选择文件)的路径，文件格式符合 XML 标准。所有配置内容都属于 XML 元素<optionMenuItem>，且每个配置项都由一个 XML 元素代表。该文件内配置项元素含义如表 3 所示，内容示意如下：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<optionMenuItem>
  <tag>SM9B100</tag>
  <display>SSMEC SM9B100XXX</display>
  <data>$CUR_DIR$\sm9b100.i79</data>
</optionMenuItem>
```

表 3 .menu 文件元素含义

元素名	含义
tag	芯片的完整型号名
display	IAR 代码项目设置中选择芯片时的显示名称
data	设备选择配置文件(.i79 文件)的路径 CUR_DIR为开发环境变量，表示本.menu 文件所在目录位置

3.3.2 sm9b100.i79

该文件同样为 SM9B100 系列处理器的设备选择文件，与.menu 文件同路径。它定义了 SM9B100 系列处理器的 IAR 工程内置配置，包含编译器设置、调试配置、设备描述文件路径、默认链接脚本路径和默认 Flash Loader 配置文件路径等。

该文件与其他配置文件关系如图 10 所示。文件语法类似为典型的 INI 配置语法格式，配置项说明如表 4 所示。其中图 10 中出现的其他类型的配置文件将会在后文说明。

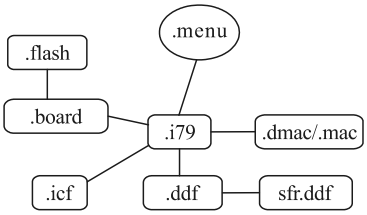

图 10 设备选择文件与其他配置文件的关联

表 4 .i79 文件元素含义

区域	键名	含义
FILEFORMAT	rev	文件版本
CHIP	name	芯片型号名
	endiansupport	处理器大小端设置，le 代表小端
	thumbsupport	ARM 处理器 Thumb 运行模式是否可用，true 代表可用
	armsupport	ARM 处理器 ARM 运行模式是否可用，true 代表可用
	fpu	浮点协处理器版本或软浮点设置，VFPv2 代表了实际硬件使用的浮点协处理器型号和版本
	SIMD	SIMD 指令集、NEON 协处理器是否可用； false，表示不包含 SIMD 指令集合 NEON 协处理器
	DeviceMacros	芯片自动关联的 C-SPY Macro 文件路径
CORE	name	处理器核架构
DDF FILE	name	默认关联的设备描述文件路径，进入 C-SPY Debug Session 时自动使用
LINKER FILE	name	默认链接文件路径
FLASH LOADER	name	默认 Flash 系统配置文件(.board 文件)路径

该文件主要内容示意如下：

```
[FILEFORMAT]
rev=1.6
[CHIP]
name=SM9B100
endiansupport=le
thumbsupport=true
armsupport=true
fpu=VFPv2
SIMD=false
DeviceMacros=$TOOLKIT_DIR$\config\debugger\SSMEC\sm9b100.dmac
[CORE]
name=ARM926EJ-S
```

```
[DDF FILE]
name=SSMEC\sm9b100.ddf
[LINKER FILE]
name=$TOOLKIT_DIR$\config\linker\SSMEC\sm9b
100.icf
[FLASH LOADER]
name=$TOOLKIT_DIR$\config\flashloader\SSMEC\
FlashSM9B100XXX.board
```

3.3.3　sm9b100.ddf

该文件为 SM9B100 系列处理器的设备描述文件，主要包含 C-SPY 调试器提供的内存空间布局定义和其他设备描述文件引用 2 方面内容。文件实际存放路径应与 sm9b100.i79 文件中定义的路径一致。

本文件典型内容格式示意如下：

```
[Memory]
Memory= ITCM Memory 0x00000000 0x0007FFFF
RW
Memory= EXTMEM_NOR Memory 0x04000000
0x07FFFFFF R
Memory= EXTMEM1_6 Memory 0x08000000
0x1FFFFFFF RW
Memory= SDRAM Memory 0x20000000
0x5FFFFFFF RW
Memory= SRAM Memory 0x60000000 0x6007FFFF
RW
Memory= PER0 Memory 0x61000000 0x64FFFFFF
RW
Memory= PER1 Memory 0x66000000 0xFFFFFFFF
RW
[Sfr]
sfr= "GPIO_A_DOUT", "Memory", 0x68600000, 4,
base=16
sfr= "GPIO_A_DIR", "Memory", 0x68600004, 4,
base=16
[SfrGroupInfo]
group= "GPIO", "GPIO_A_DOUT", "GPIO_A_DIR"
```

由上文可见，文件包含 2 个区域：

1) [Memory]：内存空间布局区域。区域内每条定义格式：Memory = “name”, “zone”, start address, end address, access type。格式内容含义如表 5 所示。

表 5　内存空间布局定义格式含义

格式名	格式含义
name	空间名称，例如：RAM0、PERH、USB 等任意名称
zone	内存地址空间名，目前仅有“Memory”
start address	空间起始地址
end address	空间结束地址，注意该地址为不超过范围的最后一个字节地址
access type	访问类型：R，只读；W，只写；RW，读写

2) [Sfr]：寄存器及比特位定义区域。区域内每条定义格式：sfr = “name”, “zone”, address, size, base=radix, [, bitRange=range]。格式内容含义如表 6 所示。

表 6　寄存器及比特位定义格式含义

格式名	格式含义
name	寄存器名。若包含小数点“.”，则小数点前为寄存器名，后为位域名
zone	寄存器所在内存空间，目前仅支持“Memory”
address	寄存器地址
size	寄存器字节大小：1，2，4，8
radix	显示寄存器值使用的进制：10，十进制；16，16 进制；float/double，单/双精度浮点
range	比特范围，格式：i-j。比如：0-2 表示位域区域为 bit0～bit2

3) [SfrGroupInfo]：寄存器组定义。区域内每条定义格式：group = "groupname", "name0", "name1", "name2" …。格式内容含义如表 7 所示。

表 7　寄存器组定义格式

格式名	格式含义
groupname	寄存器组名称。例如包含多个寄存器的外设控制器名
name	在[Sfr]区域定义过的寄存器名称

3.3.4　sm9b100.dmac

该文件为用于 SM9B100 系列处理器的调试会话内置宏代码。其实际存放路径应与 sm9b100.i79 文件中的定义一致。内容可根据不同实际项目应用需求进行修改和功能添加，以完成调试辅助和自动化测试等功能。本文中目标为添加处理器支持，对于该文件仅需提供框架和思路，无需添加任何特别内容，即可达到目标。

3.3.5　sm9b100.icf

该文件为 SM9B100 系列处理器 IAR 工程默认使用的链接配置文件(linker configuration file)。负责告诉链接器 ILINK 目标平台内存空间分布、如何处理各个代码段与数据段[7]，使链接器最终生成的程序文件能包含正确的符号地址并正确运行。

配置文件实际存放路径应与 sm9b100.i79 文件中的定义一致。

该文件在不同应用场景中内容会不同，但此处为默认配置，用于 IAR 快速建立代码工程时使用。具体设计为适用于无操作系统的嵌入式应用的配置，利于快速开发硬件验证程序、bootloader、简单应用程序等类型的程序，参考内容如下所示。

```
define memory mem with size = 4G;
define region RAM_region = mem:[from 0x00000000
to 0x0007FFFF];
```

```
define block CSTACK with alignment = 8, size = 0x1000 { };
define block SVC_STACK with alignment = 8, size = 0x40 { };
define block IRQ_STACK with alignment = 8, size = 0x100 { };
define block FIQ_STACK with alignment = 8, size = 0x00 { };
define block UND_STACK with alignment = 8, size = 0x00 { };
define block ABT_STACK with alignment = 8, size = 0x00 { };
define block HEAP with alignment = 8, size = 0x2000 { };
initialize by copy { readwrite };
do not initialize { section .noinit };
place at address mem:0x00000000 { readonly section .intvec };
place in RAM_region { readonly };
place in RAM_region { readwrite,
block CSTACK, block SVC_STACK, block IRQ_STACK, block FIQ_STACK, UND_STACK, block ABT_STACK, block HEAP};
```

由上文可见如下特性：

1) RAM_region，默认使用 ITCM 作为程序和数据空间。

2) XXXSTACK 系列定义代表了处理器不同模式下的栈空间，HEAP 定义了堆空间。

3) 属于 readwrite 类型的段在程序启动时自动初始化，数据打包算法由链接器自动决定。

4) 属于.noinit 段的数据在程序启动时不会自动初始化。

5) 期望.intvec 段用于存放处理器异常向量(还需相关代码文件配合)，并存放于指定的基地址(RAM 空间中)。

6) 其他所有数据段存放于 RAM 空间其他位置，不限定顺序。

3.3.6 Flash Loader

3.3.6.1 FlashSM9B100XXX.board

该文件为 SM9B100 系列处理器默认的 FLASH 系统配置文件，内容格式符合 XML 标准，每个配置项都由一个 XML 元素代表。配置项分为 2 个层级，顶层为<flash_board>，表示系统级的烧写配置；次顶层为<pass>，代表烧写阶段。如此的 2 层配置设计，表示进行 Flash 烧写时可根据实际需求和硬件特性包含多个阶段。

SM9B100 系列处理器片内 FLASH 无复杂特性，单阶段烧写即可满足需求，且仅需将文件整体烧入 FLASH，不做分割和填充，仅需关联一个.flash 文件。该文件的配置项元素含义如表 8—9 所示，文件内容示意如下：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<flash_board>
<pass>
<loader>$TOOLKIT_DIR$\config\flashloader\SSMEC\FlashSM9B100XXX_2M.flash</loader>
</pass>
</flash_board>
```

表 8 Flash 系统配置文件第一层级元素含义

元素名	含义
pass	代表单次 Flash 烧写阶段内容。至少包含 1 个 pass 元素定义，可以有多个不同定义，代表针对不同 Flash 设备的烧写

表 9 Flash 系统配置文件 pass 元素内子元素含义

元素名	是否强制	含义
loader	√	指定 Flash 设备配置文件（.flash 文件）路径 可使用类似$TOOLKIT_DIR$的开发环境变量
args		以 argc/argv 的形式传输参数到 FlashInit()函数 参数之间使用换行符分割 此处定义的参数会附加在.flash 文件中同名元素定义的参数之后。此处的参数是否会覆盖.flash 文件中的定义的特性由 Flash Loader 程序实际代码实现决定

3.3.6.2 FlashSM9B100XXX_2M.flash

该文件为 SM9B100 系列处理器片内 FLASH (2 MB 空间)的 Flash 设备配置文件，内容格式符合 XML 文件标准，所有配置项目都属于 XML 元素<flash_device>中，每个配置项由一个 XML 元素代表。

片内 FLASH 兼容 GD25Q16，页(page)大小为 256 Byte，最小擦除块(block)为 4 KB，且整片 FLASH 的擦除块平均分布，则可得到如下文所示的文件内容，对应配置项元素含义如表 10 所示。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<flash_device>
<exe>$TOOLKIT_DIR$\config\flashloader\SSMEC\FlashSM9B100XXX.out</exe>
<page>256</page>
<block>512 0x1000</block>
<flash_base>0x00000000</flash_base>
<macro>$TOOLKIT_DIR$\config\flashloader\SSMEC\FlashSM9B100XXX.mac</macro>
<online>0</online>
<aggregate>0</aggregate>
</flash_device>
```

表 10　Flash 设备配置文件元素含义

元素名	是否强制	含义
exe	√	指定 Flash Loader 程序路径。$TOOLKIT_DIR$为开发环境路径，代表 IAR 安装路径
flash_base	√	Flash 设备基地址
page	√	Flash 设备页（page）字节大小。256，与片内 Flash 硬件特性相同
block	√	Flash 设备块（block）信息。依次包含数量（十进制）与块字节大小（16 进制） 512 0x1000，代表 512 个 4KB 块，与片内 Flash 硬件特性相同
marco		C-SPY 宏文件路径。该宏文件与 Flash Loader 一同加载，3 个函数（execUserFlashInit()、execUserFlashReset()、execUserFlashExit()）若被定义会在特定时机调用（函数具体描述在 C-SPY Macro 章节描述）。$TOOLKIT_DIR$为开发环境路径，代表 IAR 安装路径
online		指定 Flash Loader 是否支持 Flash 断点。0，表示不支持。本芯片的片内 Flash 不能用于代码执行
aggregate		若该元素设置为 1，C-SPY 会尝试通过合并多个 block 的写操作来更高效率地用 RAM 缓冲区。仅在 block 远小于 RAM 缓冲大小时，能提高性能。该设置要求 Flash Loader 支持单次操作进行多 block 的操作
args		以 argc/argv 的形式传输参数到 FlashInit()函数。参数之间使用换行符分割。本 Flash Loader 无需任何参数，固忽略
args_doc		args 元素中的参数文字描述。这些描述会显示在开发环境的 Flash Loader Configuration 配置界面中

3.3.6.3　FlashSM9B100XXX.mac

代表 Flash Loader 运行过程中专用的 C-SPY Macro 文件(调试普通应用程序时不会被使用)，内容如下所示。

```
execUserFlashReset()
{
  __writeMemory32(0xFFFFFFFF,  0x69000000, "Memory"); // 使能外设时钟
  __delay(10); // 延时 10 毫秒
  __writeMemory32(0, 0x69000004, "Memory"); // 外设复位状态关闭
  __delay(100); // 延时 100 毫秒
}
execUserFlashExit()
{
  __var ret;
  ret = __hwRunToBreakpoint(0x0, 1000);
  if (ret < 0) {
    __message "__hwRunToBreakpoint unsuccessful";
  }
}
```

函数 excUserFlashReset()会在 Flash Loader 加载并处理复位触发后被自动调用，用于内部外设控制器时钟信号并取消其复位状态，为后续操作准备。

函数 execUserFlashExit()会在 Flash Loader 程序操作完成后或卸载前被调用，在 0 地址设置临时断点并执行代码，用于配合代码实现 IAR 的一键程序下载与调试功能(Download & Debug)。

3.3.6.4　工程与程序

基于 SM9B100MAL 的硬件特性和 Flash Loader 程序的特殊性，在工程配置和程序方面有额外特点：

1) Flash Loader 框架文件中的 flash_loader.c、flash_loader.h、flash_loader_extra.h、flash_loader_asm.s 4 个文件不能修改，必须原样添加到工程中。只有 flash_config.h 文件可根据实际需求调整。

2) 程序不能包含 main()函数。

3) 程序必须实现 FlashInit()、FlashWrite()、FlashErase() 3 种函数。

4) 程序必须实现 FlashSignoff()才能让 IAR 一键下载，调试功能正确执行，虽然该函数并非 Flash Loader 所必需。

5) 使用 flash_loader_asm.s 文件提供的 FlashInitEntry 符号作为程序入口。

3.3.6.5　链接配置

因 Flash Loader 需求和特性不同，需要独立的链接配置，不能直接使用默认链接配置，具体内容如下所示。

```
define memory mem with size = 4G;
define region RAM_region = mem:[from 0x00000000 to 0x0007FFFF];
define block CSTACK with alignment = 8, size = 0x8000  { };
initialize by copy { readwrite };
do not initialize  { section .noinit };
place at start of RAM_region {
    block RAMLOW with fixed order {
        readonly section .intvec,
        readonly,
        section LOWEND
    }
};
place at end of RAM_region {
    block RAMHIGH with fixed order {
        section HIGHSTART,
        readwrite,
        block CSTACK
```

```
    }
};
```

由上文可看出，大部分内容与默认链接配置相同，但又存在下列不同：

1) RAM 空间使用片内 SRAM，而非 ITCM。

2) RAM 空间总体划分为低端(RAMLOW)、高端(RAMHIGH) 2 个区域，且区域内的段按照配置顺序排布。

3) 低端空间以 LOWEND 段作为末尾，高端空间以 HIGHSTART 段作为起始。这 2 个段名会被 C-SPY 使用，不能修改名称，被 IAR 用于判断 Flash Loader 为烧写预留的数据缓冲区大小。

3.3.6.6　FlashInit()

该函数实现在烧写操作前的初始化动作。此处未设计.board/.flash 配置文件的参数解析，对于单一目标的烧写操作不需要，函数流程如图 11 所示。

3.3.6.7　FlashWrite()

该函数实现在指定位置开始一段范围的数据烧写，因函数本调用时的烧写数据长度会长于页大小，不一定为页大小整数倍，则函数内部处理需考虑页内偏移、跨页等情况。函数流程如图 12 所示。

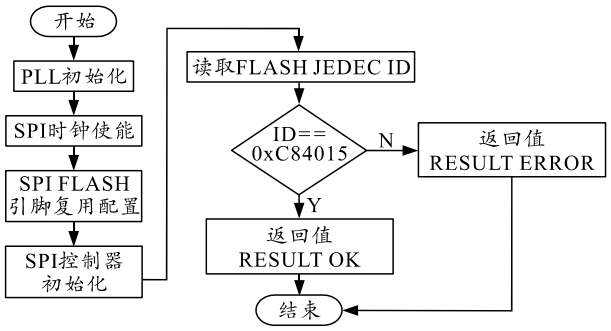

图 11　FlashInit()流程

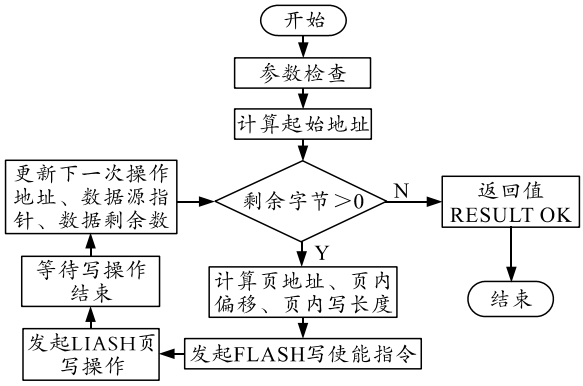

图 12　FlashWrite()流程

3.3.6.8　FlashErase()

该函数实现从指定位置开始的块擦除操作，且总是以.flash 配置中配置的 block 值为单位进行擦除，无需考虑跨块擦除的情况。函数流程如图 13 所示。

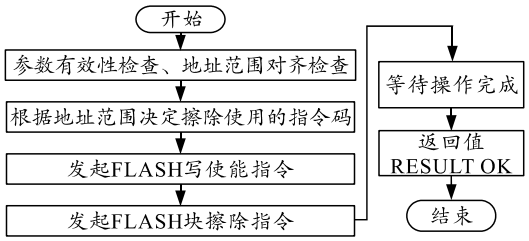

图 13　FlashErase()流程

3.3.6.9　FlashSignoff()

该函数用于最后的收尾工作，只进行读取 FLASH 低 512 KB 数据到 ITCM，以便调试 ITCM 空间的程序时可以使用一键下载调试功能，也同时要求 Flash Loader 程序本身需运行在片内 SRAM 空间，而非 ITCM 空间，避免程序自身失效。函数流程如图 14 所示。

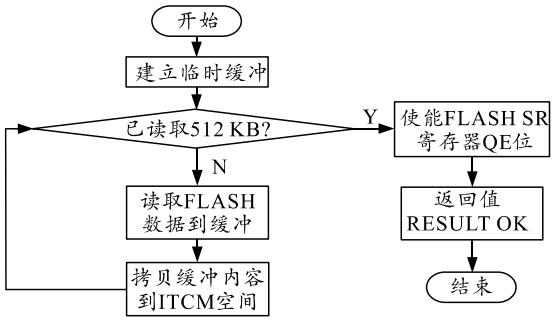

图 14 FlashSignoff()流程

4　效果

当所有工作结束后，打开 IAR 开发环境，可建立全新的嵌入式应用工程对支持效果进行实验。

如图 15 所示，工程建立后，进入工程配置界面，可在“General Options”→“Target”→“Device”一栏的选择菜单中找到新添加的处理器。该选项由 sm9b100.menu 文件直接提供。

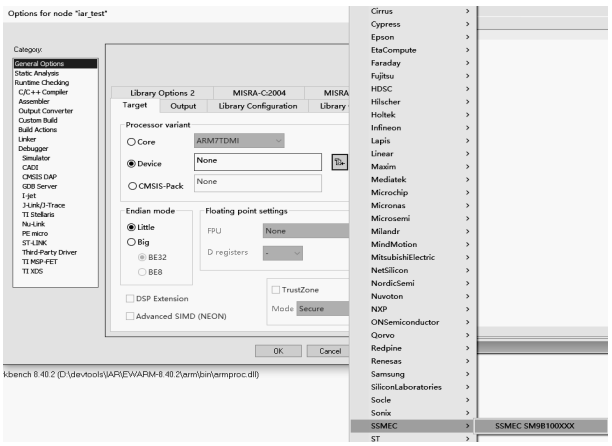

图 15　新增处理器后的设备选择效果

如图 16 所示，选择后立刻可以看到当前工程设置页面有变化，处理器模式、FPU 设置、扩展指令集自动被配置。

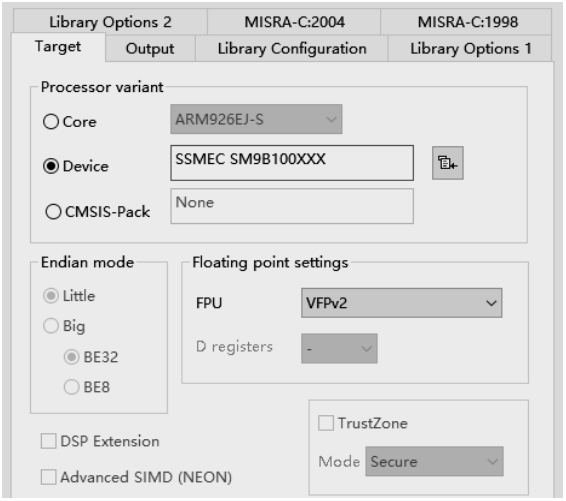

图 16　新增处理器工程自动设置效果（基本）

如图 17 所示，“C/C++ Compiler”→“Code”页面的“Processor Mode”也可选择 2 种模式。

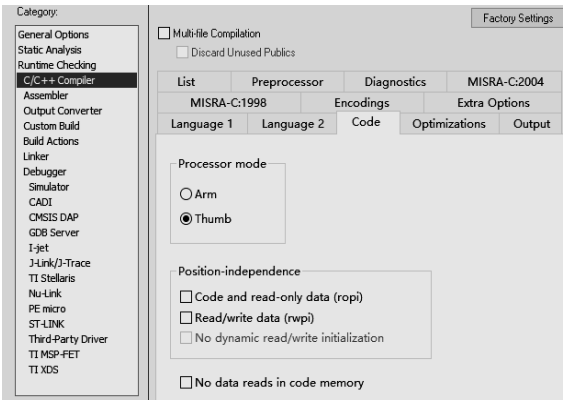

图 17　新增处理器工程自动设置效果（处理器模式）

如图 18 所示，“Linker”→“Config”页面的“Linker Configuration File”栏目中，默认关联了链接脚本文件。

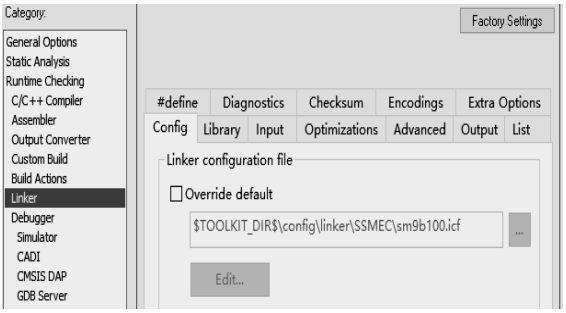

图 18　新增处理器工程自动设置效果（默认链接脚本）

如图 19 所示，“Debugger”→“Setup”页面的“Device Description File”栏目中，默认关联了设备描述文件。

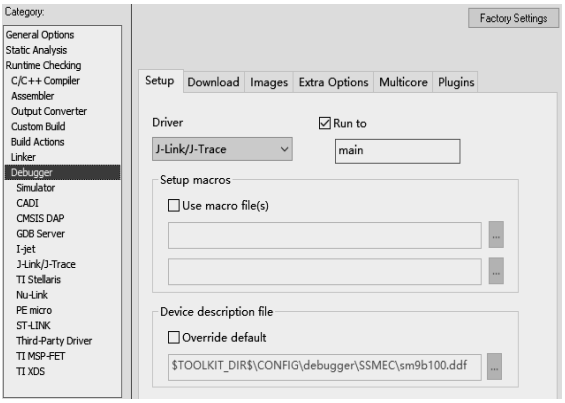

图 19　新增处理器工程自动设置效果（设备描述文件）

如图 20 所示，“Debugger”→“Download”页面中，默认关联了使用 Flash Loader 时所需要的 Flash 系统配置文件。

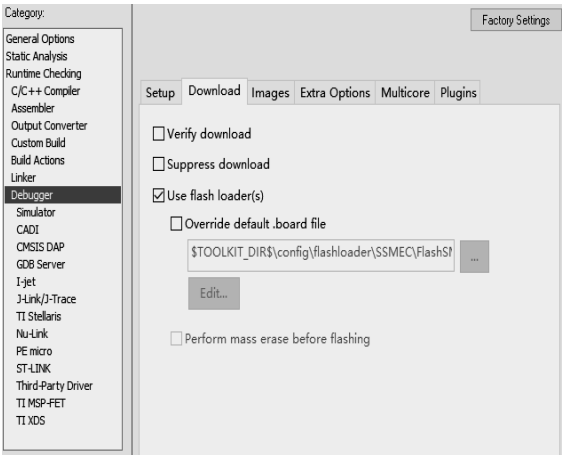

图 20　新增处理器工程自动设置效果

以上给出的工程自动配置效果皆来源于 sm9b100.i79 文件中的配置信息。

在进行工程调试过程中，打开寄存器面板（点击“View”→“Registers”→“Registers 1”），可看到自定义的 GPIO 寄存器组和相应寄存器值，详见图 21 所示。寄存器组和寄存器的定义由 sm9b100.ddf 文件提供支持，参考 3.3.3 节。

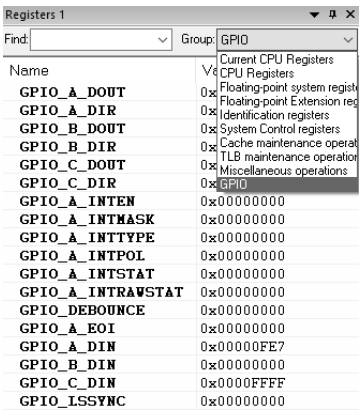

图 21　工程调试界面寄存器组效果

在调试界面中，打开 C-SPY 宏文件注册面板（点击“View”→“Macros”→“Macro Registration”），可看到默认关联的宏文件 sm9b100.dmac，参见图 22 所示。当该文件内具备有效宏函数时，在调试过程中可使用，但对于内容的描述超出了本文中探讨的范围。

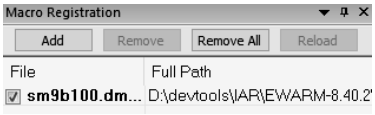

图 22 工程调试界面默认宏文件注册效果

将程序烧入芯片内置 Flash 时，需 Flash Loader 的协助。如图 23 所示，首先需在工程配置中使能 Flash Loader，即在工程配置的“Debugger”→“Download”页面中选中“Use Flash Loader(s)”。

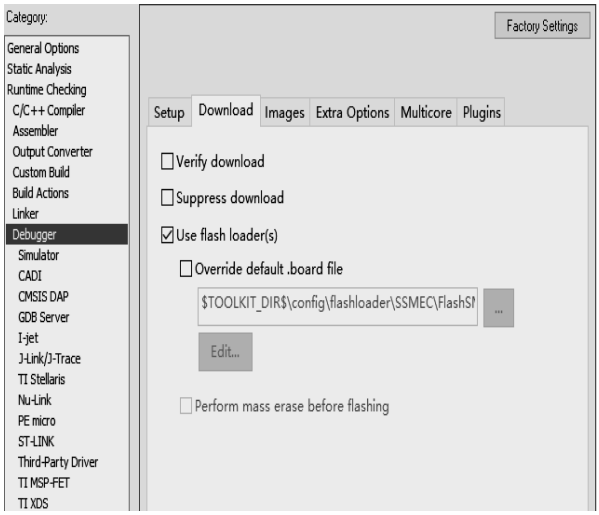

图 23 工程启用 Flash Loader

为工程启用 Flash Loader 后，在开发环境主界面点击“Project”→“Download”→“Download active application”可实现当前工程程序。点击“Project”→“Download”→“Erase memory”可实现 Flash 擦除操作，适用需完全清除数据的场景。

无论单独执行 Flash 烧写或擦除，过程中的界面显示都相同，结束后直接返回主界面，表示操作已完成。烧写和擦除基本功能的成功使用正是由 Flash Loader 开发框架和自定义实现的 FlashInit()、FlashWrite()、FlashErase()函数提供的支持，具体可参考 3.3.6 节。

当需要使用开发环境的一键程序下载与调试(Download & Debug)等进阶功能时，点击“Project”→“Download and Debug”后立即开始。烧写过程的执行界面效果与单独烧写或擦除的效果相同，但不同的是结束后直接进入调试界面，且调试器会使程序执行自动暂停在 main()函数，等待调试者后续操作。实现这样的效果不仅需要 Flash Loader 开发框架、FlashInit()、FlashWrite()、FlashErase()函数支持，还需 FlashSignoff()函数进行收尾工作，具体可参考 3.3.6.9 节。

至此，已展示了为 IAR 添加 SM9B100MAL 处理器支持后，在工程配置、调试界面寄存器、调试宏文件、工程 Flash 烧写与擦除等方面的改变和效果。成功为 IAR 开发环境添加了 SM9B100MAL 处理器的支持，不仅可实现新工程配置、调试期观测寄存器组、Flash 烧写与擦除等基本功能，而且实现了一键程序下载与调试功能，与国外处理器相比，调试过程体验差异不大，提高了开发效率。

基于本文中描述的 IAR 开发环境基础架构、C-SPY 特性、Flash Loader 原理等方面的信息，只要能处理好底层代码、内存空间配置、寄存器定义等处理器高度依赖部分的差异，可将该方法扩展到更多的国产 ARM 处理器中，让其更充分、更高效地应用到 IAR 开发环境。

5 结束语

笔者基于项目应用需求，详细分析了 IAR 的内部功能原理、硬件架构等，明确了在 IAR 中成功添加 SM9B100MAL 处理器支持时所需的具体目标与实施细节，展示了成功添加支持后的运行效果，解决了基于该处理器平台的 IAR 工程、调试期间查看寄存器、Flash 烧写等情形下效率低的问题，在当前没有 IAR 平台官方开发支持的情况下，增添了一种开发选择。笔者提出的思路和配置及代码细节，有助于其他使用该开发环境的开发人员做出改进，并将支持范围扩展到更多国产 ARM 处理器上。

参考文献：

[1] IAR Embedded Workbench[EB/OL]. [2020-06-18]. http://baike.baidu.com/item/IAR%20Embedded%20Workbench.
[2] IAR Systems. IDE Project Management and Building Guide for Arm Limited’s Arm® Cores[S].
[3] IAR Systems. C-SPY® Debugging Guide for Arm Limited’s Arm® cores[S].
[4] IAR Systems. IAR device description file format IAR Embedded Workbench® for ARM[S].
[5] IAR Systems. Flash Loader Development Guide for IAR Embedded Workbench®[S].
[6] 深圳市国微电子有限公司. SM9B100MAL 产品用户手册 v1.1[S].
[7] IAR Systems. IAR C/C++ Development Guide[S].