九齐科技股份有限公司

Nyquest Technology Co., Ltd.

用户手册

NYC_NY8L

C Compiler for NY8L series

Version 1.2

Feb. 19, 2025

目　录

1 简介

NYC_NY8L 为针对九齐科技的 NY8L 系列 8 位 MCU IC 而提供的 C 语言编译程序（Compiler）。它被上层开发工具软件 *NYIDE* 所调用以编译 C 程序，并结合 *NYASM* 组译器（Assembler）进一步组译及连结目的档来产生.bin 档，然后.bin 档可下载到板子或烧录到 OTP IC。

1.1 如何使用本手册

1. 简介

为何需要 *NYC_NY8L* 与安装 *NYC_NY8L* 的基本需求。

2. *使用 NYC_NY8L*

如何透过 *NYIDE* 使用 *NYC_NY8L*。

3. 语法与使用

介绍 *NYC_NY8L* 的语法与使用方式

1.2 系统需求

以下列出使用 *NYC_NY8L* 的系统需求。

- Pentium 1.3GHz 或更高级处理器，Win7、Win8、Win10、Win11 操作系统。
- 至少 2G SDRAM。
- 至少 2G 硬盘空间。
- Visual C++ 2015-2022 Redistributable（32bit）。

1.3 安装 *NYC_NY8L*

请联系九齐科技来取得 *NYC_NY8L* 的安装程序档，双击执行档后进入安装程序向导，然后依照画面提示将可轻松完成安装流程。如果您的计算机没有安装 Visual C++ 2015-2022 Redistributable（32bit），在安装过程中会自动下载，此步骤需要因特网联机。若您的环境没有因特网联机，请预先至微软网站下载 V Visual C++ 2015-2022 Redistributable（32bit）并安装。

2 使用 NYC_NY8L

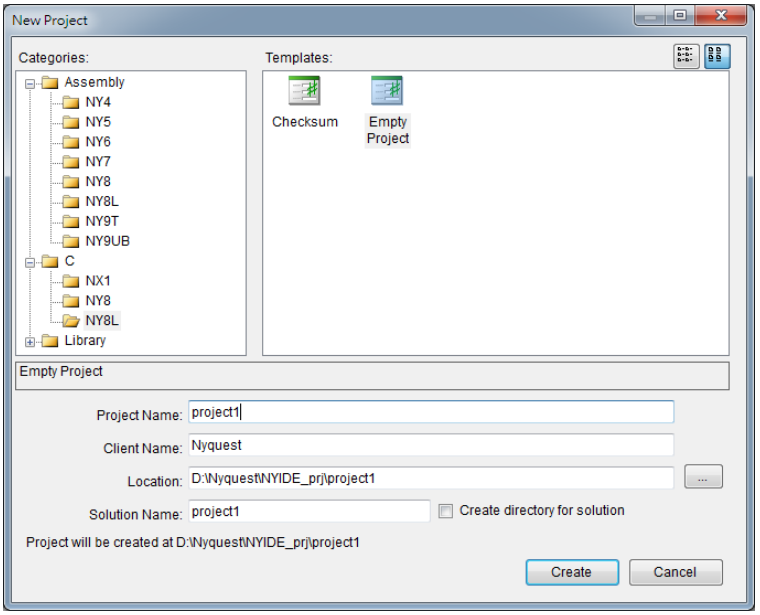
当用户使用 NY8L 软件开发工具 *NYIDE* 编写 C 程序后，在 *NYIDE* 界面上按下 Build 时，软件开发工具会自动寻找并使用已安装于计算机上的 *NYC_NY8L* 作编译。以下将说明透过 *NYIDE* 来使用 *NYC_NY8L* 的流程。

2.1 通过 *NYIDE* 使用 *NYC_NY8L*

NYIDE 为九齐科技提供以开发 NY4 / 5 / 6 / 7 / 8 / 9T / 9UB / NX1 系列微控制器程序的整合性开发工具，主要目的为提供用户以汇编语言（Assembly）和 C 语言来编写程序，并拥有建置和强大的除错功能。当使用 *NYIDE* 开发 NY8 项目，在建置和除错时，*NYIDE* 会自动寻找并使用安装于计算机上的 *NYC_NY8L* 工具链。以下简单的介绍使用 *NYIDE* 开发 NY8L 项目。更详细的操作方式，请参考 *NYIDE* 使用手册。

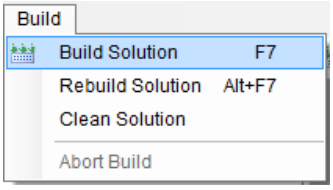
2.1.1 建立新项目

打开 *NYIDE*，选择建立新项目。在 New Project 窗口内，左边的 Categories 选择 C，然后选择 NY8L。指定项目名称及类型后，按下 Create。*NYIDE* 将会自动产生必要文件，项目已于可建置状态。



2.1.2 建置

在 *NYIDE* 主画面上的菜单选择 Build / Build Solution 进行建置（或按下快捷键 F7），即会调用 *NYC_NY8L* 执行建置动作。若建置成功会在项目目录产生.bin 文件，以进一步提供下载或烧录。



3 语法与使用

NYC_NY8L 支持标准的 ANSI C89 语法，并且针对 NY8L 系列 IC 新增了一些特殊语法。

3.1 标准 C 语法

NYC_NY8L 支持标准的 ANSI C89 语法，有关详细语言定义请参考：Standard ISO/IEC 9899 (http://www.open-std.org/jtc1/sc22/wg14/www/standards.html#9899)

3.1.1 批注

批注支持两种格式，以双斜线起头的单行批注，以及/*起头，至*/结尾的多行批注。

范例：

```
// single line comment

/*
Multi line comment
*/
```

3.1.2 数据型态

以下表格列出 *NYC_NY8L* 所使用的基本数据型态及允许的数据范围。

型态	长度	范围
char	1 byte	0 ~ 255
signed char	1 byte	-128 ~ 127
short	2 bytes	-32768 ~ 32767
unsigned short	2 bytes	0 ~ 65535 (0xFFFF)
int	2 bytes	-32768 ~ 32767
unsigned int	2 bytes	0 ~ 65535 (0xFFFF)
long	4 bytes	-2147483648 ~ 2147483647
unsigned long	4 bytes	0 ~ 4294967295 (0xFFFFFFFF)

3.2 扩充语法

3.2.1 保留字

以下列出所有保留字，用户定义的符号不可和保留字相同。

__Pragma	__near__	else	near	unsigned
__AX__	asm	enum	register	void
__A__	auto	extern	restrict	volatile
__EAX__	break	far	return	while
__X__	case	fastcall	short	
__Y__	cdecl	float	signed	
__asm__	char	for	sizeof	
__attribute__	const	goto	static	
__cdecl__	continue	if	struct	
__far__	default	inline	switch	
__fastcall__	do	int	typedef	
__inline__	double	long	union	

3.2.2 中断服务程式（ISR）

NY8L 系列有多个中断，每个中断都有一个 16 位的中断向量，用以存放中断服务程序（ISR）的地址。当中断发生时，需要暂停目前的程序，并保存当前的系统状态，例如 X、Y、A 三个寄存器，然后根据中断的种类查找中断向量，取得相应的中断服务程序的地址，并执行中断服务程序，执行结束恢复状态并以 `RTI` 指令返回中断前地址。使用 NY8L C，只要在中断服务程序标示属性 `INTERRUPT_XXX`，*NYC_NY8L* 就会自动产生保存状态及在中断向量定义中断服务程序地址等所必须的程序代码。下面的范例程序示范如何建立 TM1 中断服务程序。

```
#include <ny8l.h>

void tm1_isr(void) INTERRUPT_TM1 {
    INTF = ~C_INTF_TM1_Flag;
}
```

可用的函数属性列表如下，可以在安装目录的 `include/ny8l_common.h` 文件找到属性的原始定义。

INTERRUPT_TM2	INTERRUPT_TM1	INTERRUPT_TM0	INTERRUPT_FT
INTERRUPT_ST	INTERRUPT_EXT	INTERRUPT_ADC	INTERRUPT_KSB
INTERRUPT_NMI	INTERRUPT_BRK		

NYC_NY8L 1.10 以上才提供中断服务程序的函数属性，在 1.10 以前的版本必须由使用者自行编写内建的汇编语言 vector.s 文件来定义中断向量。当旧版 *NYC_NY8L* 所开发的项目移至 1.10 版以上，如果有使用此函数属性，必须自行删除旧版项目中的 vector.s 文件相应的定义，以免发生冲突，建议若有使用

函数属性就将 vector.s 文件删除全部改用 C 的函数属性。在 1.10 版本以上中断向量可选择使用汇编或 C 语言的函数属性，但建议择其一使用，不要两个混合使用。

3.2.3 中断服务程序（ISR）使用汇编语言

在 C 语言项目中可以混合使用扩展名为.s 的汇编语言文件，也可以使用汇编语言来撰写中断服务程序。底下提供 TM2 中断服务程序的汇编语言范例。

```
    .export Vector_tm2
    .import ___exit_isr_pop_axy
.segment "CODE"
Vector_tm2:
    phy
    phx
    pha
    lda     #$FE
    sta     _INTF
    jmp     ___exit_isr_pop_axy
```

在上面的范例中，导出符号 Vector_tm2 为内建函数库所规定的中断服务程序名称，若名称不符合，编译程序将无法把中断服务程序地址定义到中断向量。中断服务程序名称列表如下：

Vector_tm2	Vector_tm1	Vector_tm0	Vector_ft
Vector_st	Vector_ext	Vector_adc	Vector_ksb
Vector_nmi	Vector_brk		

而导入符号 `___exit_isr_pop_axy` 为内建函数库所提供的函数，将系统状态复原并返回中断前的地址，内容如下：

```
___exit_isr_pop_axy:
        pla
        plx
        ply
        rti
```

3.2.4 寄存器位址定义

所有 NY8L IC 的特殊寄存器都已经被定义在”NY8L.h”的文件中，并置放于程序安装文件夹的 include 目录下。建议使用者直接使用该头文件，不须自行定义特殊寄存器。

3.2.5 读写指定地址

读取绝对地址内存的方式是将所要读取的地址作为立即值转型为指针再取值。例如以下范例读取于绝对地址 0x7F0 及 0x7F1 的 2 bytes int。

```
uint16_t value = (*(uint16_t*)0x7f0);
```

而写入绝对地址寄存器方法相似，以下范例将内存地址 0x80 写入值 0x12。

```
*(uint8_t*)0x80 = 0x12;
```

指针的型态将决定所要存取的单位大小。

3.2.6 内嵌汇编语言（Inline assembly）

在 C 语言之中可以内嵌汇编语言，使用__asm__关键词即可插入汇编语言程序。

范例：

```
__asm__(“nop”);
```

如果需要在组合语言之中使用 C 语言定义的变量，可以使用%v 格式化文本。

范例：

```
__asm__(“lda %v”, my_var);
```

Inline assembly 跟一般 C 语言产生的汇编语言一样会受到优化的影响。目前没有办法单独将 inline assembly 程序片段排除在优化之外，只能选择整个函数是否要开启优化。如果最佳化会造成您的困扰，可以使用 volatile 关键词让包含__asm__的函数不被优化。

范例：

```
__asm__ volatile("lda #0");
```

3.3 系统标头档

在 NYC_NY8L 安装目录中的 include 文件夹有所有 C 语言使用的标头档（header file），本章节介绍这些标头档的内容及使用方法。

3.3.1 特殊指令宏

ny8lcommon.h 文件定义了常用的特殊汇编语言指令宏，以较为低阶的方式有效率地控制 IC 行为，用户可在适当的时机调用这些宏。

宏	说明
CLI()	打开中断功能
SEI()	关闭中断功能
CLRWDT()	清除 watch dog 定时器
SLEEP()	进入 sleep
ROM_BANK(addr)	取得(addr >> 16) & 0xFF 的值(bank address)

3.3.2 特殊功能寄存器（SFR）

ny8l_common.h 文件定义了特殊功能寄存器（SFR）的名称，使用者可以藉由这些定义存取 SFR。但是不建议使用者直接引用 ny8l_common.h，建议只引用头文件 NY8L.h。使用者不应该更改 *NYC_NY8L* 文件，因为此文件必须与内建函数库中的命名对应，更改此文件将造成连结失败。

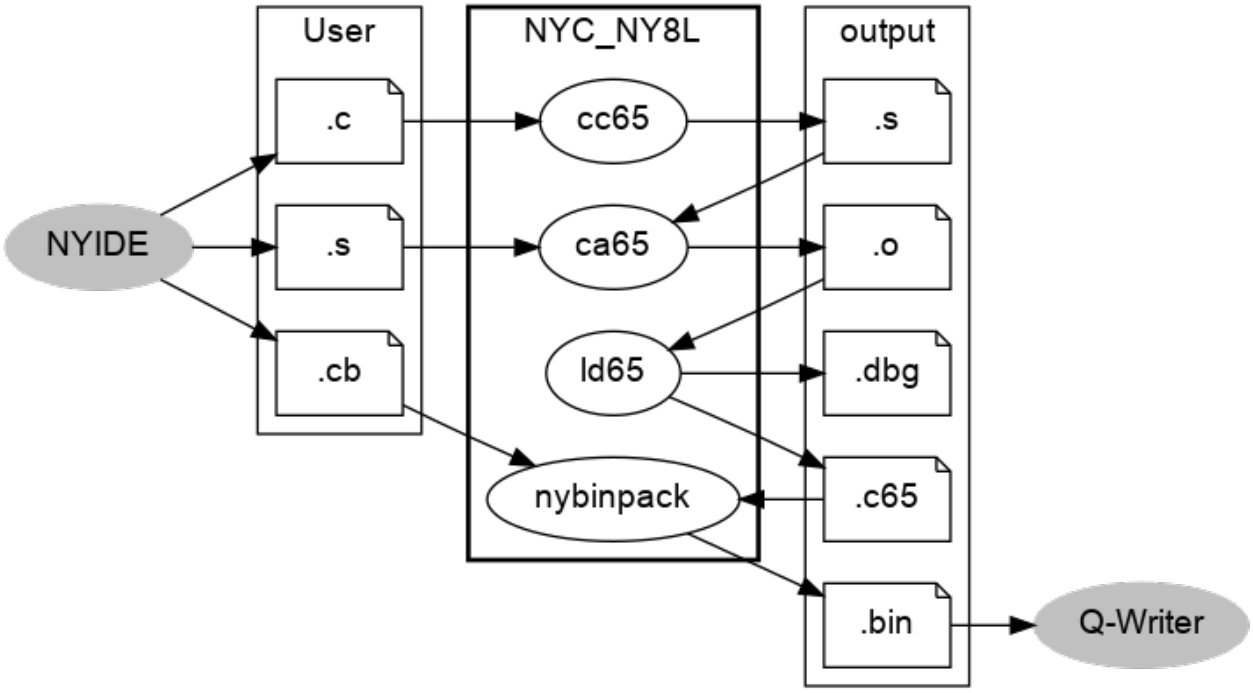
3.4 选项

使用 *NYIDE* 开发 C 语言项目，可以设定若干项目建置选项。这些选项可以控制编译程序、组译器、链接器的行为，点选菜单的项目（Project）/项目设定（Project Settings）可打开设定界面。

- 产生汇编语言列表档（Generate ASM listing file）：组译完成产生列表档，档名为 *.lst。不勾选此选项可加快编译速度。
- 产生链接栏表档（Generate listing file）：链接后产生列表档，档名为 *.link.lst。此档为最终.bin 文件的反组译结果，不勾选此选项可以加快编译速度。
- 产生地址对应档（Generate map file）：链接后产生地址对应档，档名为 *.map。此文件包含地址分配信息，不勾选此选项可加快编译速度。
- 优化（Optimization）：优化所产生的程序代码。C 语言编译为汇编语言的优化，可以产生较为精简的程序代码。
- 开机清空内存（Clear RAM to zero on startup）：在 IC 开机尚未进入使用者 main 函数前将所有内存设为 0。
- 元文件目录（Intermediate Directory）：编译过程中的元文件存放目录。
- 汇编语言引用路径（ASM include path）：指定汇编语言.s 文件的引用搜寻路径。默认为项目根目录及 NYC_NY8L 安装目录的 asminc 文件夹。用户可以增加自定义的路径。
- 引用文件路径（Include path）：设定 C 语言 include 关键词引用标头档的搜寻路径。默认的路径为项目根目录及 *NYC_NY8L* 安装目录的 include 文件夹。用户可以增加自定义的路径。

3.5 开发流程

使用 *NYIDE* 编写 C 语言程序，并设定项目所需的组态档.cb，*NYIDE* 建置时自动调用 *NYC_NY8L* 之中的 *cc65.exe* 产生汇编语言文件.s，*ca65.exe* 组译汇编语言文件，ld65.exe 连结所有目标文件，nybinpack.exe 合并组态文件与程序文件，产生最终.bin 档。最后可以藉由 *Q-Writer* 将.bin 档烧录至 IC。



3.6 使用建议

以下提出一些开发 C 语言项目的建议。

- 尽量使用无符号（unsigned）变量，在部分的运算不用判断正负号执行速度会比较快。
- 表达式之中不要交互使用常数与变量，将常数集中才能有效的优化。
  例如 1 + a + 2 就是不好的写法，1 跟 2 无法在编译时期计算。建议写成 a+1+2，如此一来 1+2 可在编译时期计算，执行期间只需要计算 a+3 即可。

4 改 版 记 录

版本	日期	内 容 描 述	修正页
1.0	2018/05/31	新发布。	-
1.1	2018/07/03	修改 ISR 使用方式。	7, 8
1.2	2025/02/19	系统需求与函数库更新。	3