九齊科技股份有限公司
Nyquest Technology Co., Ltd.

User Manual

NYC_NY8L

C Compiler for NY8L series

Version 1.1

Jul. 3, 2018

Revision History

Version	Date	Description	Modified Page
1.0	2018/05/31	Formal release.	-
1.1	2018/07/03	Modify ISR using method.	7, 8

Table of Contents

1 Introduction

NYC_NY8L is the C Compiler for Nyquest 8-bit MCU “NY8L series”. *NYC_NY8L* is called by the upper level development tools *NYIDE* to compile C program into assembly, *NYASM* Assembler will then assemble and link the object files to generate .bin file, which is used to download to the board or program to OTP IC.

1.1 Outline of the manual

1. Introduction

   This chapter explains the role *NYC_NY8L* plays and the basic requirements for the installation of *NYC_NY8L*.

2. *Use NYC_NY8L*

   How to use *NYC_NY8L* through *NYIDE*.

3. Syntax and usage

   Introduce the syntax and usage of *NYC_NY8L*.

1.2 System Requirements

- A PC equipped with Pentium 1.3GHz or higher CPU, Windows 7/ 8/ 10.
- At least 2G SDRAM.
- At least 2G free space on the hard disk.
- Visual C++ 2015 Redistributable (32bit).

1.3 The Installation of *NYC_NY8*

Please contact Nyquest Technology to obtain the latest installation program. Double click the execution icon to activate installation wizard, and follow the instructions to complete the installation process. Visual C++ 2015 Redistributable (32bit) is required to run *NYC_NY8L*. If Visual C++ 2015 Redistributable (32bit) is not installed on your computer, it will be downloaded automatically. It requires internet connection when downloading, and if no internet is connected and fails to download, you may visit Microsoft website to download Visual C++ 2015 Redistributable (32bit) afterwards and install.

2 Use *NYC_NY8L*

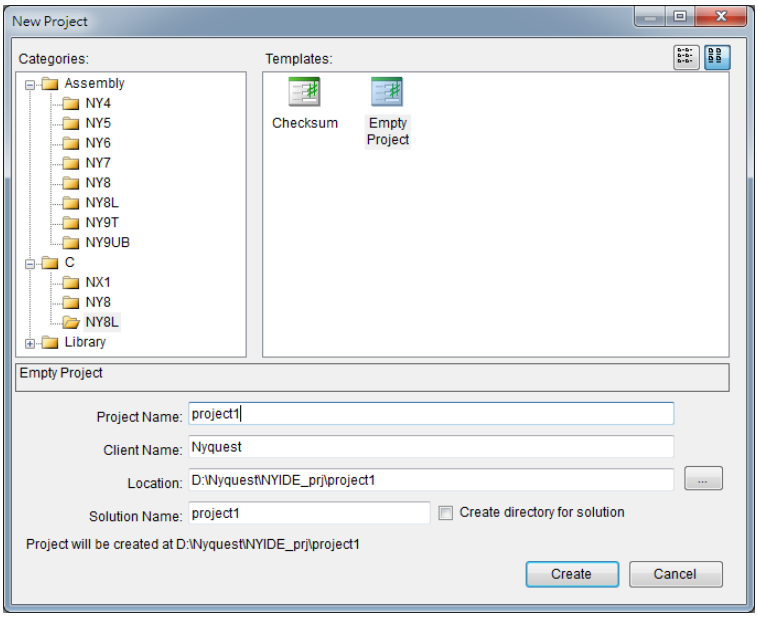
After finishing a program in NY8L software development tool - *NYIDE*, pressing Build in the *NYIDE* menu will automatically search for installed *NYC_NY8L* to compile and link. The procedures for using *NYC_NY8L* in *NYIDE* are described below.

2.1 Use *NYC_NY8L* through *NYIDE*

NYIDE is an integrated tool provided by Nyquest for developing application of NY4 / 5 / 6 / 7 / 8 / 9T / 9UB / NX1 series microcontroller. The main purpose is to provide a platform for programming with Assembly language and C language, as well as build and strong debug functions. When using *NYIDE* to develop NY8L projects, *NYIDE* will automatically search for installed *NYC_NY8L* tool chain on computer for building and debugging. The following is an introduction of using *NYIDE* to develop NY8 projects. More detailed operations please refer to the *NYIDE* user manual.

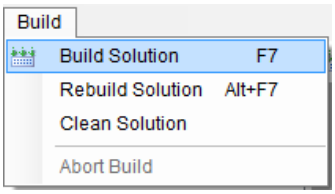
2.1.1 Create New Project

Open *NYIDE*, and select New Project. In the New Project window, choose C on the Categories and select NY8L. Specify project name and type, then press “Create”, and *NYIDE* will automatically generate the necessary files.



2.1.2 Build

When user selects the Build / Build Solution menu (or press the shortcut key F7) on the *NYIDE* main screen, *NYC_NY8L* will be called to perform the build action. If it is successfully built, the .bin file will be generated in the project directory for downloading or programming.



3 Syntax and Usage

NYC_NY8L supports standard ANSI C89 syntax, and adds some specific syntax for NY8L series IC.

3.1 Standard C Syntax

NYC_NY8L supports standard ANSI C89 syntax. For more detailed regarding language definitions, please refer to: Standard ISO/IEC 9899 (http://www.open-std.org/jtc1/sc22/wg14/www/standards.html#9899).

3.1.1 Comment

There are 2 forms of Comment. The single line comment begins with double slash, and the multi-line comment begins with /* and ends with */.

Example:

```
// single line comment

/*
Multi line comment
*/
```

3.1.2 Data Type

The following table is the basic data types and the data range of *NYC_NY8L*.

Type	Length	Range
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 Extended Syntax

3.2.1 Reserved Word

All reserved words are listed below, and the user-defined symbols can not be the same as the reserved words.

__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 Interrupt Service Routine

There are multiple interrupts in NY8L series, and every interrupt has a 16-bit vector, that is a pointer, to store the interrupt service routine (ISR). When interrupt occurs, the current program needs to be suspended and the current system state is saved, for example, three registers X, Y, and A. Then find the corresponding interrupt vector according to the type of interrupt, obtain the address of the interrupt service routine, and execute the interrupt service routine. After the ISR is serviced, the status is recovered and is returned to the previously suspended program with a RTI instruction. In C program, simply use INTERRUPT_XXX attribute to define an interrupt service routine, and *NYC_NY8L* will automatically generate the necessary code to save the status and store the interrupt service program address in the interrupt vector. The following sample program demonstrates how to create TM1 interrupt service routine.

```
#include <ny8l.h>

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

The list of available function attributes is as follows. The original definition of the attribute can be found in the include/ny8l_common.h file of installation directory.

INTERRUPT_TM2	INTERRUPT_TM1	INTERRUPT_TM0	INTERRUPT_FT

INTERRUPT_ST	INTERRUPT_EXT	INTERRUPT_ADC	INTERRUPT_KSB
INTERRUPT_NMI	INTERRUPT_BRK		

NYC_NY8L supports INTERRUPT_XXX attribute for interrupt service routines above 1.10. Prior to 1.10, users have to write assembly in a built-in vector.s file to define interrupt vectors. When the project developed by the old version *NYC_NY8L* is moved to version 1.10 or later, must be aware that not to redefine INTERRUPT_XXX attribute. For example, if INTERRUPT_TM1 is defined, you must delete the corresponding definition in vector.s file to avoid conflicts. It is recommended to delete the vector.s file and use C's attributes only, if you use one of the attributes. Interrupt vector can choose to use assembly or C language function attributes above version 1.10, but it is recommended to use one of them instead of two.

3.2.3 Interrupt Service Routine Uses Assembly Language

In the C language project, assembly language file with a .s file extension can be used, and an interrupt service routine can also be written using assembly language. Below is an example of assembly language that provides TM2 interrupt service routines.

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

In the above example, the export symbol Vector_tm2 is the name of the interrupt service routine specified by the built-in function library. If the name does not match, the compiler cannot define the interrupt service routine address to the interrupt vector. The list of interrupt service routine names is as follows:

Vector_tm2	Vector_tm1	Vector_tm0	Vector_ft
Vector_st	Vector_ext	Vector_adc	Vector_ksb
Vector_nmi	Vector_brk		

The import symbol __exit_isr_pop_axy is a function provided by the built-in function library, which restores the system state and returns the address before the interrupt. The contents are as follows:

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

3.2.4 Register Address Definition

All registers of NY8L IC have been defined in the header file ”NY8L.h” in “include” directory of the installation folder. It is recommended to use the header file directly, which will save the efforts to define special registers.

3.2.5 Read and Write specified address

The way to read the absolute address memory is to write the address as the immediate value and convert to an pointer, and then dereference it. For example, the following example program reads 2-byte int at the absolute address 0x7F0 and 0x7F1.

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

The method of writing to the absolute address register is similar. The following example program writes the value 0x12 to the memory address 0x80.

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

The type of pointer decides the size of data to be accessed.

3.2.6 Inline Assembly

The assembly can be embedded in the C language, use the keyword "__asm__" to insert any assembly programs.

Example:

```
__asm__(“nop”);
```

If you need to use C-defined variables in the assembly language, please use %v to format the text.

Example:

```
__asm__(“lda %v”, my_var);
```

As the assembly codes generated by C codes are affected by optimization, so are Inline assembly codes. There is no way to exclude the inline assembly code from optimization, but only to choose whether the entire function will be optimized. If optimization causes some unexpected result, please use the volatile keyword after __asm__ to keep it from being optimized.

Example:

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

3.3 System Header File

The “include” folder in the *NYC_NY8L* installation directory has C header files for all NY8L IC. This section describes the contents of these header files and how to use them.

3.3.1 Special Command Macro

The ny8lcommon.h file defines commonly used assembly macros that control IC behavior in a lower-level, and user can call these macros at the proper time.

Macro	Description
CLI()	Enable interrupt.
SEI()	Disable interrupt.
CLRWDT()	Clear the watch dog timer.
SLEEP()	Sleep.
ROM_BANK(addr)	Get the value of (addr >> 16) & 0xFF(bank address)

3.3.2 Special Function Register

The ny8l_common.h file defines the name of Special Function Register (SFR) that users can use them to access SFR. However, it is not recommended that users directly refer to ny8l_common.h. It is recommended to use the header file NY8L.h instead. The user should not change the NYC_NY8L file because the file must correspond to the name in the built-in library. Changing this file will cause the link process fails.

3.4 Option

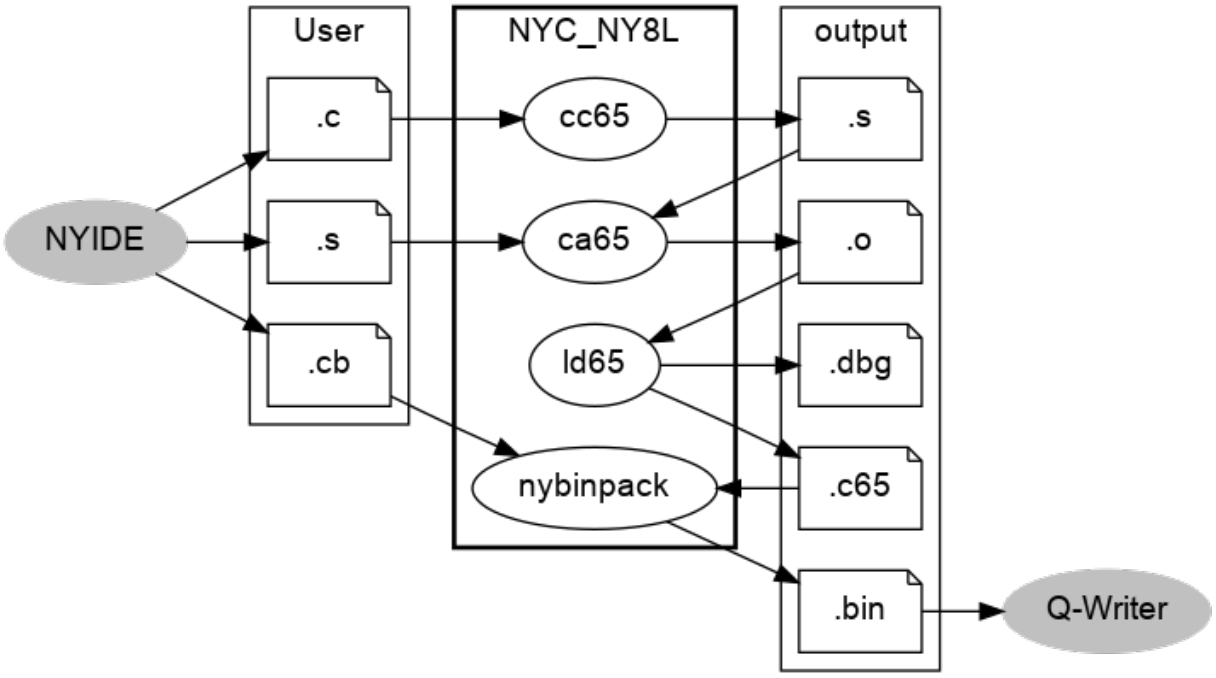
Using *NYIDE* to develop a C language project, there are several project build options can be set. These options can control the compiler, assembler and linker behavior. User can select the Project / Project Settings on Menu to open the setting interface.

- Generate ASM listing file: The listing file named *.lst will be produced after assembling, deselecting this option can speed up the compiling speed.
- Generate listing file: The listing file named *.link.lst will be produced after linking. This file is the disassembled result of the final .bin file. Deselecting this option can speed up the compiling speed.
- Generate map file: The listing file named *.map will be produced after linking. This file contains address assignment information. Deselecting this option can speed up the compiling speed.
- Optimization: Optimize the generated code. Compiling C language into optimization of assembly language can produce more streamlined code.
- Clear RAM to zero on startup: Set all memory to 0 after power on and before entering into the user's main function.
- Intermediate Directory: The directory for saving intermediate files.

- ASM include path: Set the search path for .s file inclusion. The default path is the “asminc” folder of the project root directory and the *NYC_NY8L* installation directory. User can add a custom path.
- Include path: Set the search path for the C include header file. The default path is the “include” folder of the project root directory and the *NYC_NY8L* installation directory. User can add a custom path.

3.5 Development Process

Use *NYIDE* to write the C language program and set the configuration file ".cb" required for the project. *NYIDE* will automatically call *NYC_NY8L* to generate the assembly file ".s" when building and then call the *ca65.exe* to assemble the assembly code ld65.exe and the configuration file nybinpack.exe to produce the final .bin file. Finally, user can use the *Q-Writer* to burn the .bin file to IC.



3.6 Suggestion

Some suggestions for developing C language projects are listed below.

- Try to use unsigned variables. In some operations which do not judge plus or minus, it will be faster.
- Do not use constants and variables interactively in the expression, intensively using the constants will have an optimized code.

  Eg. "1 + a + 2" is a bad coding style, as 1 and 2 can not be calculated while compiling. It is recommended to write "a+1+2", for 1+2 can be calculated while compiling, and it only needs to calculate "a+3" in the execution.