

Nuclei N100 SDK

Nuclei N100 embedded Software Development Kit

Nuclei N100 SDK

Release 0.2.1

Nuclei

Aug 06, 2025

CONTENTS:

1	Overview	1
1.1	Introduction	1
1.2	Design and Architecture	1
1.3	Get Started	2
1.4	Contributing	2
1.5	Copyright	2
1.6	License	3
2	Quick Startup	5
2.1	Use Nuclei N100 SDK in Nuclei Studio	5
2.2	Setup Tools and Environment	5
2.2.1	Use Prebuilt Tools in Nuclei Studio	5
2.3	Get and Setup Nuclei N100 SDK	6
2.4	Build, Run and Debug Sample Application	7
2.4.1	Hardware Preparation	10
2.4.2	Build Application	10
2.4.3	Run Application	11
2.4.4	Debug Application	12
2.5	Create helloworld Application	14
2.6	Advanced Usage	16
3	Developer Guide	17
3.1	Code Style	17
3.2	Build System based on Makefile	17
3.2.1	Makefile Structure	17
3.2.2	Makefile targets of make command	23
3.2.3	Makefile variables passed by make command	24
3.2.4	Makefile variables used only in Application Makefile	30
3.2.5	Build Related Makefile variables used only in Application Makefile	37
3.3	Application Development	42
3.3.1	Overview	42
3.3.2	Add Extra Source Code	43
3.3.3	Add Extra Include Directory	43
3.3.4	Add Extra Build Options	43
3.3.5	Optimize For Code Size	44
3.3.6	Change Link Script	44
3.3.7	Set Default Make Options	44
3.4	Build Nuclei N100 SDK Documentation	44
3.4.1	Install Tools	44
3.4.2	Build The Documentation	45

4	Contributing	47
4.1	Port your Nuclei SoC into Nuclei N100 SDK	47
4.2	Submit your issue	50
4.3	Submit your pull request	50
4.4	Git commit guide	51
5	Design and Architecture	53
5.1	Overview	53
5.1.1	Directory Structure	53
5.1.2	Project Components	56
5.2	Nuclei Processor	57
5.2.1	Introduction	57
5.2.2	NMSIS in Nuclei N100 SDK	57
5.2.3	SoC Resource	58
5.3	SoC	59
5.3.1	Nuclei Eval SoC	59
5.4	Board	61
5.4.1	Nuclei FPGA Evaluation Kit	61
5.5	Peripheral	64
5.5.1	Overview	64
5.5.2	Usage	64
5.6	RTOS	65
5.6.1	Overview	65
5.6.2	FreeRTOS	65
5.6.3	UCOSII	66
5.6.4	RT-Thread	66
5.7	Application	67
5.7.1	Overview	67
5.7.2	Bare-metal applications	68
5.7.3	FreeRTOS applications	76
5.7.4	UCOSII applications	77
5.7.5	RT-Thread applications	78
6	Changelog	81
6.1	V0.2.1	81
6.2	V0.2.0	81
6.3	V0.1.0	82
7	FAQ	85
7.1	Why I can't download application?	85
7.2	How to select correct FTDI debugger?	86
7.3	Why I can't download application in Linux?	86
7.4	Why the provided application is not running correctly in my Nuclei FPGA Evaluation Board?	87
7.5	Access to github.com is slow, any workaround?	87
7.6	`.text` will not fit in region `ilm` or `.bss` will not fit in region `ram`	87
7.7	undefined reference to `__errno` when using libnrt library	88
7.8	undefined reference to `fclose/sprintf` similar API provided in system libraries	89
7.9	fatal error: `rvintrin.h`: No such file or directory	89
7.10	risvcv-nuclei-elf-gcc: not found when using Nuclei Studio 2023.10	89
8	License	91
9	Glossary	97
10	Appendix	99

11 Indices and tables

101

Index

103

OVERVIEW

1.1 Introduction

Note: This is Nuclei N100 SDK which is modified based on **Nuclei SDK 0.5.0** to support Nuclei 100 series CPU.

If you are looking for Nuclei SDK for Nuclei **200/300/600/900/1000** CPU, please refer to https://doc.nucleisys.com/nuclei_sdk

Please use **Nuclei Studio >= 2025.02** with this Nuclei N100 SDK.

The **Nuclei N100 Software Development Kit (SDK)** is an open-source software platform to speed up the software development of SoCs based on Nuclei Processor 100 series Cores.

This Nuclei N100 SDK is built based on the **modified NMSIS**¹ for N100 and also the APIs that provided by Nuclei N100 SDK which mainly for on-board peripherals access such as UART etc.

Nuclei N100 SDK provides a good start base for embedded developers which will help them simplify software development and improve time-to-market through well-designed software framework.

Note: To get a pdf version of this documentation, please click [Nuclei N100 SDK Document](#)²

1.2 Design and Architecture

The Nuclei N100 SDK general design and architecture are shown in the block diagram as below.

As *Nuclei N100 SDK Design and Architecture Diagram* (page 2) shown, The Nuclei N100 SDK provides the following features:

- Nuclei Core API service is built on top of **modified NMSIS**³, so silicon vendors of Nuclei processors can easily port their SoCs to Nuclei N100 SDK, and quickly evaluate software on their SoC.
- Mainly support *Nuclei Eval SoC* (page 59)
- Provided realtime operation system service via *FreeRTOS* (page 65), *UCOSII* (page 66) and *RT-Thread* (page 66)
- Provided bare-metal service for embedded system software beginners and resource-limited use-cases.
- Applications are logically separated into three parts:

¹ <https://github.com/Nuclei-Software/NMSIS>

² https://doc.nucleisys.com/nuclei_n100_sdk/nuclei_n100sdk.pdf

³ <https://github.com/Nuclei-Software/NMSIS>

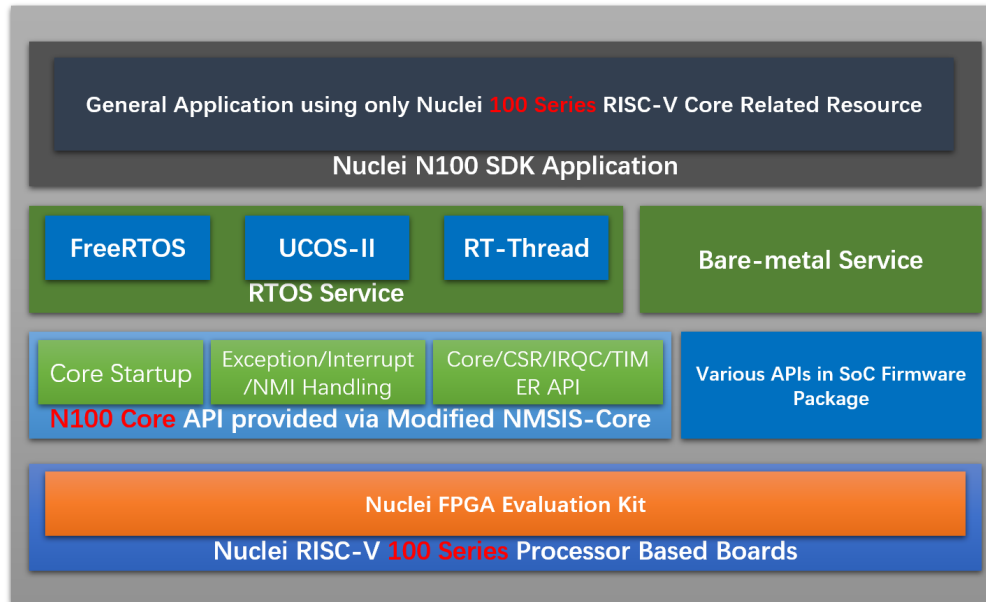


Fig. 1: Nuclei N100 SDK Design and Architecture Diagram

- **General applications for Nuclei 100 series Processors:** In the Nuclei N100 SDK software code, the applications provided are all general applications which can run on all Nuclei Processors, with basic UART service to provide `printf` function.
- **Nuclei Eval SoC applications:** These applications are not included in the Nuclei N100 SDK software code, and it is *maintained separately*, which will use resource from Nuclei Eval SoC and its evaluation boards to develop applications, which will not be compatible with different boards.

1.3 Get Started

Please refer to [Quick Startup](#) (page 5) to get started to take a try with Nuclei N100 SDK.

1.4 Contributing

Contributing to Nuclei N100 SDK is welcomed, if you have any issue or pull request want to open, you can take a look at [Contributing](#) (page 47) section.

1.5 Copyright

Copyright (c) 2019 - Present, Nuclei System Technology. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Nuclei System Technology., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. NY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.6 License

Nuclei N100 SDK is an opensource project licensed by *Apache License 2.0* (page 91).

QUICK STARTUP

2.1 Use Nuclei N100 SDK in Nuclei Studio

You can download **Nuclei Studio IDE >= 2025.02** from [Nuclei Download Center](#)⁴, and follow [Nuclei Studio and Nuclei Tools User Guide](#)⁵ to learn how to use it.

But if you want to use latest source code of Nuclei N100 SDK, please follow the rest part of this guide to build and run using Nuclei N100 SDK Build System in Makefile.

2.2 Setup Tools and Environment

To start to use Nuclei N100 SDK, you need to install the following tools:

2.2.1 Use Prebuilt Tools in Nuclei Studio

Since **2020.10** release version of Nuclei Studio, you just need to download the **Nuclei Studio IDE** from [Nuclei Download Center](#)⁶ for your development OS, and no need to do the following steps below, the prebuilt tools are already included.

For example:

- In Windows, if you have extracted the Nuclei Studio IDE to `D:\Software\NucleiStudio_IDE_202502`, then you can find the prebuilt tools in `D:\Software\NucleiStudio_IDE_202502\NucleiStudio\toolchain`.
- In Linux, if you have extracted the Nuclei Studio IDE to `/home/labdev/NucleiStudio_IDE_202502`, then you can find the prebuilt tools in `/home/labdev/NucleiStudio_IDE_202502/NucleiStudio/toolchain`.

You can also update tools located in the Nuclei Studio prebuilt tools `toolchain` by downloading newer version from [Nuclei Tools](#)⁷ and replace it.

If you have downloaded and extracted the Nuclei Studio, then you can jump to [Get and Setup Nuclei N100 SDK](#) (page 6) and ignore below steps.

⁴ <https://nucleisys.com/download.php>

⁵ https://doc.nucleisys.com/nuclei_tools/

⁶ <https://nucleisys.com/download.php>

⁷ <https://nucleisys.com/download.php>

2.3 Get and Setup Nuclei N100 SDK

The source code of Nuclei N100 SDK is maintained in [Github](#)⁸ and [Gitee](#)⁹.

- We mainly maintained github version, and gitee version is mirrored, just for fast access in China.
- Check source code in [Nuclei N100 SDK in Github](#)¹⁰ or [Nuclei N100 SDK in Gitee](#)¹¹ according to your network status.
- Stable version of Nuclei N100 SDK is maintained in **master** version, if you want release version of **Nuclei N100 SDK**, please check in [Nuclei N100 SDK Release in Github](#)¹².

Here are the steps to clone the latest source code from Github:

- Make sure you have installed Git tool, see <https://git-scm.com/download/>
- Then open your terminal, and make sure git command can be accessed
- Run `git clone -b master_n100 https://github.com/Nuclei-Software/nuclei-sdk nuclei-n100-sdk` to clone source code into `nuclei-n100-sdk` folder

Note:

- **develop_n100** branch is develop branch for Nuclei 100 series RISC-V CPU support.
- **master_n100** branch is stable branch for Nuclei 100 series RISC-V CPU support.
- If you have no access to github.com, you can also use command `git clone -b master_n100 https://gitee.com/Nuclei-Software/nuclei-sdk nuclei-n100-sdk` to clone from gitee.
- If you have no internet access, you can also use pre-downloaded `nuclei-n100-sdk` code, and use it.
- If the backup repo is not up to date, you can import github repo in gitee by yourself, see <https://gitee.com/projects/import/url>

-
- Create tool environment config file for Nuclei N100 SDK

Note: If you want to use **Terapines ZCC** toolchain, you can download it from <https://www.terapines.com/>, or use **Nuclei Studio** >= **2025.02**, a **Terapines ZCC Lite** version is integrated in `<NucleiStudio>/toolchain/zcc` folder, and you also need to add extra **PATH** into your environment, like this:

- **Windows:** execute `set PATH=%path%\to\zcc\bin;%PATH%` in windows cmd terminal before run Nuclei SDK
- **Linux:** execute `set PATH=/path/to/zcc/bin:$PATH` in linux shell terminal before build Nuclei SDK

– **Windows**

If you want to use Nuclei N100 SDK in **Windows Command Prompt** terminal, you need to create `setup_config.bat` in `nuclei-n100-sdk` folder, and open this file your editor, and paste the following content, assuming you followed *Setup Tools and Environment* (page 5), and prebuilt tools located in `D:\Software\NucleiStudio_IDE_202502\NucleiStudio\toolchain`, otherwise please use your correct tool root path.

⁸ <https://github.com>

⁹ <https://gitee.com>

¹⁰ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100

¹¹ https://gitee.com/Nuclei-Software/nuclei-sdk/tree/develop_n100

¹² <https://github.com/Nuclei-Software/nuclei-sdk/releases>

```
set NUCLEI_TOOL_ROOT=D:\Software\NucleiStudio_IDE_202502\NucleiStudio\
↪toolchain
```

If you want to use Nuclei N100 SDK in **Windows PowerShell** terminal, you need to create a `setup_config.ps1` in `nuclei-n100-sdk` folder, and edit this file with content below if your prebuilt tools are located in `D:\Software\NucleiStudio_IDE_202502\NucleiStudio\toolchain`:

```
$NUCLEI_TOOL_ROOT="D:\Software\NucleiStudio_IDE_202502\NucleiStudio\
↪toolchain"
```

– Linux

Create `setup_config.sh` in `nuclei-n100-sdk` folder, and open this file your editor, and paste the following content, assuming you followed *Setup Tools and Environment* (page 5) and prebuilt tools located in `/home/labdev/NucleiStudio_IDE_202502/NucleiStudio/toolchain`, otherwise please use your correct tool root path.

```
NUCLEI_TOOL_ROOT=/home/labdev/NucleiStudio_IDE_202502/NucleiStudio/toolchain
```

2.4 Build, Run and Debug Sample Application

Assume you have followed steps in *Get and Setup Nuclei N100 SDK* (page 6) to clone source code and create files below:

- `setup_config.bat` for run in **Windows Command Prompt** terminal
- `setup_config.ps1` for run in **Windows PowerShell** terminal
- `setup_config.sh` for run in **Linux Bash** terminal

To build, run and debug application, you need to open command terminal in `nuclei-n100-sdk` folder.

- For **Windows** users, you can open **Windows Command Prompt** terminal and `cd` to `nuclei-n100-sdk` folder, then run the following commands to setup build environment for Nuclei N100 SDK, the output will be similar as this screenshot *Setup Build Environment for Nuclei N100 SDK in Windows Command Prompt* (page 8):

```
1 setup.bat
2 echo %PATH%
3 where riscv64-unknown-elf-gcc openocd make rm
4 make help
```

- For **Linux** users, you can open **Linux Bash** terminal and `cd` to `nuclei-n100-sdk` folder, then run the following commands to setup build environment for Nuclei N100 SDK, the output will be similar as this screenshot *Setup Build Environment for Nuclei N100 SDK in Linux Bash* (page 9):

```
1 source setup.sh
2 echo $PATH
3 which riscv64-unknown-elf-gcc openocd make rm
4 make help
```

Note:

- Only first line `setup.bat` or `source setup.sh` are required before build, run or debug application. The `setup.bat` and `setup.sh` are just used to append Nuclei RISC-V GCC Toolchain, OpenOCD and Build-Tools binary paths into environment variable **PATH**

```

C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.18363.657]
(c) 2019 Microsoft Corporation. 保留所有权利。

D:\workspace\Sourcecode\nuclei-sdk setup.bat 1
Setup Nuclei SDK Tool Environment
NUCLEI_TOOL_ROOT=D:\Software\Nuclei

D:\workspace\Sourcecode\nuclei-sdk echo %PATH% 2
D:\Software\Nuclei\gcc\bin;D:\Software\Nuclei\openocd\bin;D:\Software\Nuclei\build-tools\bin;C:\Program
Files (x86)\Common Files\Oracle\Java\javapath;C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:
\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSS
H\C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files\NVIDIA Corporation\NVIDIA NvD
LISR\C:\Program Files\Git\cmd;C:\Program Files\Pandoc\C:\Program Files (x86)\Google\Cloud SDK\google-cl
oud-sdk\bin;C:\Users\57856\AppData\Local\Programs\Python\Python38\Scripts\;C:\Users\57856\AppData\Local
\Programs\Python\Python38\;C:\Users\57856\AppData\Local\Microsoft\WindowsApps\;C:\Users\57856\AppData\Loc
al\Programs\Microsoft VS Code\bin

D:\workspace\Sourcecode\nuclei-sdk where riscv-nuclei-elf-gcc openocd make rm 3
D:\Software\Nuclei\gcc\bin\riscv-nuclei-elf-gcc.exe
D:\Software\Nuclei\openocd\bin\openocd.exe
D:\Software\Nuclei\gcc\bin\make.exe
D:\Software\Nuclei\build-tools\bin\make.exe
D:\Software\Nuclei\build-tools\bin\rm.exe

D:\workspace\Sourcecode\nuclei-sdk make help 4
make -C application/baremetal/helloworld help
make[1]: Entering directory 'D:/workspace/Sourcecode/nuclei-sdk/application/baremetal/helloworld'
"Nuclei N/NX-series RISC-V Embedded Processor Software Development Kit"
"== Make variables used in Nuclei SDK =="
"SOC:      Select SoC built in Nuclei SDK, will select hbird by default"
"BOARD:    Select SoC's Board built in Nuclei SDK, will select hbird_eval by default"
"CORE:     Not required for all SoCs, currently only hbird require it, n307fd by default"
"DOWNLOAD: Not required for all SoCs, use ilm by default, optional flashxip/ilm/flash"
"V:        V=1 verbose make, will print more information, by default V=0"
"== How to Use with Make =="
"1. Build Application:"
"all [PROGRAM=flash/flashxip/ilm]"
"   Build a software program to load with the debugger."
"2. Upload Application to Board using OpenOCD and GDB:"
"upload [PROGRAM=flash/flashxip/ilm]"
"   Launch OpenOCD to flash your program to the on-board Flash."
"3: (Option 1) Debug Application using OpenOCD and GDB"
"  3.1: run_openocd"
"  3.2: run_gdb [PROGRAM=flash/flashxip/ilm]"
"   Step 1: Launch OpenOCD for Debugger connection: make run_openocd"
"   Step 2: Launch GDB to connect openocd server, you can set breakpoints using gdb and debug it."
"   If you want to load your application, you need to run load in gdb command terminal"
"   to load your program, then use gdb to debug it."
"3: (Option 2) Debug Application using OpenOCD and GDB"
"debug [PROGRAM=flash/flashxip/ilm]"
"   Launch GDB and OpenOCD to debug your application on-board, you can set breakpoints using gdb and deb
ug it."
"   If you want to load your application, you need to run load in gdb command terminal"
"   to load your program, then use gdb to debug it."
"
"

```

Fig. 1: Setup Build Environment for Nuclei N100 SDK in Windows Command Prompt

```

hcfang@hcfang-ubuntu [13:01:04]: ~/workspace/software/nuclei-sdk
$ source setup.sh 1
Setup Nuclei SDK Tool Environment
NUCLEI_TOOL_ROOT=/home/hcfang/Software/Nuclei
hcfang@hcfang-ubuntu [13:01:07]: ~/workspace/software/nuclei-sdk
$ echo $PATH 2
/home/hcfang/Software/Nuclei/gcc/bin:/home/hcfang/Software/Nuclei/openocd/bin:/home/hcfang/mysofts/Nuclei/spike/rt
enocd/bin:/home/hcfang/mysofts/Nuclei/gcc/bin:/home/hcfang/pycharm-community/bin:/home/hcfang/mysofts/typora-linux
usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
hcfang@hcfang-ubuntu [13:01:26]: ~/workspace/software/nuclei-sdk
$ which riscv-nuclei-elf-gcc openocd make rm 3
/home/hcfang/Software/Nuclei/gcc/bin/riscv-nuclei-elf-gcc
/home/hcfang/Software/Nuclei/openocd/bin/openocd
/usr/bin/make
/bin/rm
hcfang@hcfang-ubuntu [13:01:38]: ~/workspace/software/nuclei-sdk
$ make help 4
make -C application/baremetal/helloworld help
make[1]: 进入目录“/home/hcfang/workspace/software/nuclei-sdk/application/baremetal/helloworld”
Nuclei N/NX-series RISC-V Embedded Processor Software Development Kit
== Make variables used in Nuclei SDK ==
SOC:      Select SoC built in Nuclei SDK, will select hbird by default
BOARD:    Select Soc's Board built in Nuclei SDK, will select hbird_eval by default
CORE:     Not required for all socs, currently only hbird require it, n307fd by default
DOWNLOAD: Not required for all socs, use ilm by default, optional flashxip/ilm/flash
V:        V=1 verbose make, will print more information, by default V=0
== How to Use with Make ==
1. Build Application:
all [PROGRAM=flash/flashxip/ilm]
   Build a software program to load with the debugger.
2. Upload Application to Board using OpenOCD and GDB:
upload [PROGRAM=flash/flashxip/ilm]
   Launch OpenOCD to flash your program to the on-board Flash.
3:(Option 1) Debug Application using OpenOCD and GDB
  3.1: run_openocd
  3.2: run_gdb [PROGRAM=flash/flashxip/ilm]
   Step 1: Launch OpenOCD for Debugger connection: make run_openocd
   Step 2: Launch GDB to connect openocd server, you can set breakpoints using gdb and debug it.
   If you want to load your application, you need to run load in gdb command terminal
   to load your program, then use gdb to debug it.
3:(Option 2) Debug Application using OpenOCD and GDB
debug [PROGRAM=flash/flashxip/ilm]
   Launch GDB and OpenOCD to debug your application on-board, you can set breakpoints using gdb and debug it.
   If you want to load your application, you need to run load in gdb command terminal

```

Fig. 2: Setup Build Environment for Nuclei N100 SDK in Linux Bash

- line 2-4 are just used to check whether build environment is setup correctly, especially the **PATH** of Nuclei Tools are setup correctly, so we can use the `riscv64-unknown-elf-xxx`, `openocd`, `make` and `rm` tools
- If you know how to append Nuclei RISC-V GCC Toolchain, OpenOCD and Build-Tools binary paths to **PATH** variable in your OS environment, you can also put the downloaded Nuclei Tools as you like, and no need to run `setup.bat` or `source setup.sh`
- If you want to run in **Windows PowerShell**, please run `.\setup.ps1` instead of `setup.bat`, and `setup_config.ps1` must be created as described in *Get and Setup Nuclei N100 SDK* (page 6).

Here for a quick startup, this guide will take board *Nuclei FPGA Evaluation Kit* (page 61) for example to demonstrate how to setup hardware, build run and debug application in Windows.

The demo application, we will take `application/baremetal/helloworld` for example.

First of all, please reuse previously build environment command terminal.

Run `cd application/baremetal/helloworld` to cd the `helloworld` example folder.

2.4.1 Hardware Preparation

Please check *Board* (page 61) and find your board's page, and follow **Setup** section to setup your hardware, mainly **JTAG debugger driver setup and on-board connection setup**.

- Power on the *Nuclei FPGA Evaluation Kit* (page 61) board, and use USB Type-C data cable to connect the board and your PC, make sure you have setup the JTAG driver correctly, and you can see JTAG port and serial port.
- Open a UART terminal tool such as *TeraTerm in Windows*¹³ or *Minicom in Linux*¹⁴, and monitor the serial port of the Board, the UART baudrate is *115200 bps*
- If you are building example for your own SoC and Board, please pass correct *SOC* (page 24) and *BOARD* (page 25) make variable. eg. If you SoC is *evalsoc* and Board is *nuclei_fpga_eval*, just pass *SOC=evalsoc BOARD=nuclei_fpga_eval* to make instead of the one mentioned below. If your default board for this *evalsoc* is *nuclei_fpga_eval*, then you don't need to pass *BOARD=nuclei_fpga_eval*.
- If you don't pass any SOC or BOARD via make, *evalsoc* and *nuclei_fpga_eval* are default SoC and Board.

2.4.2 Build Application

We need to build application for this board *Nuclei FPGA Evaluation Kit* (page 61) using this command line:

Note:

- Since below steps are taking *evalsoc* SoC based board *nuclei_fpga_eval* to do demonstration, and when you pass *SOC=evalsoc*, the default *BOARD* will be *nuclei_fpga_eval*, so do you don't need to pass *BOARD=nuclei_fpga_eval*
- You can check default *SOC/BOARD/CORE* information passed by using *make target info*, eg. *make SOC=evalsoc info*, for more information, please check *Makefile targets of make command* (page 23).

```
# clean application if build in other application before or build for other board
make SOC=evalsoc clean
# first build choice: using full command line
make SOC=evalsoc all
# second build choice: using simple command line, since when SOC=evalsoc, default BOARD_
↪is nuclei_fpga_eval
make SOC=evalsoc all
```

Here is the sample output of this command:

```
# NOTICE: You can check this configuration whether it matched your desired configuration
Current Configuration: TOOLCHAIN=nuclei_gnu RISCV_ARCH=rv32ic RISCV_ABI=ilp32 RISCV_
↪TUNE=nuclei-100-series RISCV_CMODEL=medlow SOC=evalsoc BOARD=nuclei_fpga_eval_
↪CORE=n100 ARCH_EXT= DOWNLOAD=ilm STDCLIB=newlib_nano SMP= BOOT_HARTID=
"Assembling : " ../../../../SoC/evalsoc/Common/Source/GCC/intexc_evalsoc.S
"Assembling : " ../../../../SoC/evalsoc/Common/Source/GCC/startup_evalsoc.S
...
"Compiling : " ../../../../SoC/evalsoc/Common/Source/Stubs/write.c
"Compiling : " ../../../../SoC/evalsoc/Common/Source/evalsoc_soc.c
"Compiling : " ../../../../SoC/evalsoc/Common/Source/system_evalsoc.c
"Compiling : " main.c
```

(continues on next page)

¹³ <http://tssh2.osdn.jp/>

¹⁴ <https://help.ubuntu.com/community/Minicom>

(continued from previous page)

```
"Linking      : " helloworld.elf
text  data    bss    dec    hex filename
13022  112     2290  15424  3c40 helloworld.elf
```

As you can see, that when the application is built successfully, the elf will be generated and will also print the size information of the helloworld.elf.

Note:

- In order to make sure that there is no application build before, you can run `make SOC=evalsoc clean` to clean previously built objects and build dependency files.
- About the make variable or option(**SOC**, **BOARD**) passed to make command, please refer to *Build System based on Makefile* (page 17).

2.4.3 Run Application

If the application is built successfully for this board *Nuclei FPGA Evaluation Kit* (page 61), then you can run it using this command line:

```
make SOC=evalsoc upload
```

Here is the sample output of this command:

```
"Download and run helloworld.elf"
riscv64-unknown-elf-gdb helloworld.elf -ex "set remotetimeout 240" \
    -ex "target remote | openocd -c \"gdb_port pipe; log_output openocd.log\" -f ../.
↳ ../SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg" \
    --batch -ex "monitor halt" -ex "monitor halt" -ex "monitor flash protect 0 0
↳ last off" -ex "load" -ex "monitor resume" -ex "monitor shutdown" -ex "quit"
D:\Software\Nuclei\gcc\bin\riscv64-unknown-elf-gdb.exe: warning: Couldn't determine a
↳ path for the index cache directory.
Nuclei OpenOCD, 64-bit Open On-Chip Debugger 0.10.0+dev-00014-g0eae03214 (2019-12-12-
↳ 07:43)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
_start0800 () at ../../../../SoC/evalsoc/Common/Source/GCC/startup_evalsoc.S:359
359          j 1b
cleared protection for sectors 0 through 127 on flash bank 0

Loading section .init, size 0x266 lma 0x8000000
Loading section .text, size 0x2e9c lma 0x8000280
Loading section .rodata, size 0x1f0 lma 0x8003120
Loading section .data, size 0x70 lma 0x8003310
Start address 0x800015c, load size 13154
Transfer rate: 7 KB/sec, 3288 bytes/write.
shutdown command invoked
A debugging session is active.

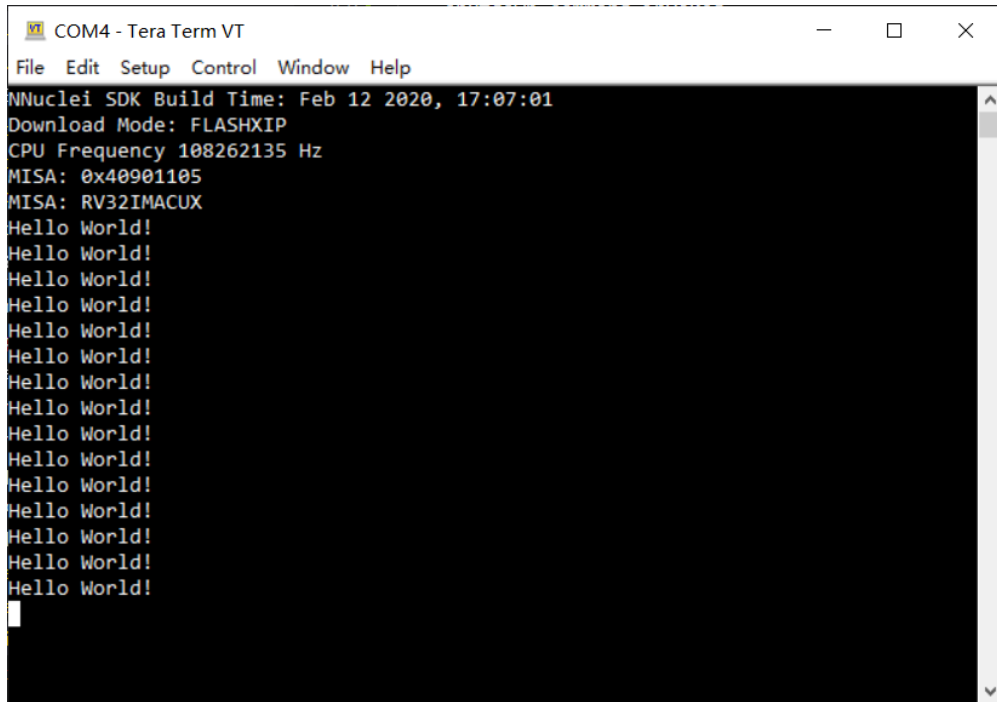
    Inferior 1 [Remote target] will be detached.
```

(continues on next page)

(continued from previous page)

```
Quit anyway? (y or n) [answered Y; input not from terminal]
[Inferior 1 (Remote target) detached]
```

As you can see the application is uploaded successfully using `openocd` and `gdb`, then you can check the output in your UART terminal, see *Nuclei N100 SDK Hello World Application UART Output* (page 12).



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
NNuclei SDK Build Time: Feb 12 2020, 17:07:01
Download Mode: FLASHXIP
CPU Frequency 108262135 Hz
MISA: 0x40901105
MISA: RV32IMACUX
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Fig. 3: Nuclei N100 SDK Hello World Application UART Output

2.4.4 Debug Application

If the application is built successfully for this board *Nuclei FPGA Evaluation Kit* (page 61), then you can debug it using this command line:

```
make SOC=evalsoc debug
```

1. The program is not loaded automatically when you enter to debug state, just in case you want to debug the program running on the board.

```
"Download and debug helloworld.elf"
riscv64-unknown-elf-gdb helloworld.elf -ex "set remotetimeout 240" \
    -ex "target remote | openocd -c \"gdb_port pipe; log_output openocd.log\" -
↪f ../../../../SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg"
D:\Software\Nuclei\gcc\bin\riscv64-unknown-elf-gdb.exe: warning: Couldn't determine
↪a path for the index cache directory.
GNU gdb (GDB) 8.3.0.20190516-git
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
```

(continues on next page)

(continued from previous page)

```

This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.
--Type <RET> for more, q to quit, c to continue without paging--

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from helloworld.elf...
Remote debugging using | openocd -c \"gdb_port pipe; log_output openocd.log\" -f ../
->../SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg
Nuclei OpenOCD, 64-bit Open On-Chip Debugger 0.10.0+dev-00014-g0eae03214 (2019-12-
->12-07:43)
Licensed under GNU GPL v2
For bug reports, read
  http://openocd.org/doc/doxygen/bugs.html
_start0800 () at ../../SoC/evalsoc/Common/Source/GCC/startup_evalsoc.S:359
359      j 1b

```

- If you want to load the built application, you can type load to load the application.

```

(gdb) load
Loading section .init, size 0x266 lma 0x8000000
Loading section .text, size 0x2e9c lma 0x8000280
Loading section .rodata, size 0x1f0 lma 0x8003120
Loading section .data, size 0x70 lma 0x8003310
Start address 0x800015c, load size 13154
Transfer rate: 7 KB/sec, 3288 bytes/write.

```

- If you want to set a breakpoint at *main*, then you can type b main to set a breakpoint.

```

(gdb) b main
Breakpoint 1 at 0x8001b04: file helloworld.c, line 85.

```

- If you want to set more breakpoints, you can do as you like.
- Then you can type c, then the program will stop at **main**

```

(gdb) c
Continuing.
Note: automatically using hardware breakpoints for read-only addresses.

Breakpoint 1, main () at helloworld.c:85
85      srand(__get_rv_cycle() | __get_rv_instret() | __RV_CSR_READ(CSR_
->MCYCLE));

```

- Then you can step it using n (short of next) or s (short of step)

```
(gdb) n
86      uint32_t rval = rand();
(gdb) n
87      rv_csr_t misa = __RV_CSR_READ(CSR_MISA);
(gdb) s
89      printf("MISA: 0x%lx\r\n", misa);
(gdb) n
90      print_misa();
(gdb) n
92      printf("Hello World!\r\n");
(gdb) n
93      printf("Hello World!\r\n");
```

7. If you want to quit debugging, then you can press CTRL - c, and type q to quit debugging.

```
(gdb) Quit
(gdb) q
A debugging session is active.

        Inferior 1 [Remote target] will be detached.

Quit anyway? (y or n) y
Detaching from program: D:\workspace\Sourcecode\nuclei-n100-sdk\application\
↪baremetal\helloworld\helloworld.elf, Remote target
Ending remote debugging.
[Inferior 1 (Remote target) detached]
```

Note:

- More about how to debug using gdb, you can refer to the [GDB User Manual](#)¹⁵.
- If you want to debug using Nuclei Studio, you can open Nuclei Studio, and create a debug configuration, and choose the application elf, and download and debug in IDE.

2.5 Create helloworld Application

If you want to create your own helloworld application, it is also very easy.

There are several ways to achieve it, see as below:

- **Method 1:** You can find a most similar sample application folder and copy it, such as `application/baremetal/helloworld`, you can copy and rename it as `application/baremetal/hello`
 - Open the Makefile in `application/baremetal/hello`
 1. Change `TARGET = helloworld` to `TARGET = hello`
 - Open the `helloworld.c` in `application/baremetal/hello`, and replace the content using code below:

```
1 // See LICENSE for license details.
2 #include <stdio.h>
```

(continues on next page)

¹⁵ <https://www.gnu.org/software/gdb/documentation/>

(continued from previous page)

```

3 #include <time.h>
4 #include <stdlib.h>
5 #include "nuclei_sdk_soc.h"
6
7 int main(void)
8 {
9     printf("Hello World from Nuclei RISC-V Processor!\r\n");
10    return 0;
11 }

```

– Save all the changes, and then you can follow the steps described in *Build, Run and Debug Sample Application* (page 7) to run or debug this new application.

- **Method 2:** You can also do it from scratch, with just create simple Makefile and main.c

- Create new folder named hello in application/baremetal
- Create two files named Makefile and main.c
- Open Makefile and edit the content as below:

```

1 TARGET = hello
2
3 NUCLEI_SDK_ROOT = ../../..
4
5 SRCDIRS = .
6
7 INCDIRS = .
8
9 include $(NUCLEI_SDK_ROOT)/Build/Makefile.base

```

- Open main.c and edit the content as below:

```

1 // See LICENSE for license details.
2 #include <stdio.h>
3 #include <time.h>
4 #include <stdlib.h>
5 #include "nuclei_sdk_soc.h"
6
7 int main(void)
8 {
9     printf("Hello World from Nuclei RISC-V Processor!\r\n");
10    return 0;
11 }

```

- Save all the changes, and then you can follow the steps described in *Build, Run and Debug Sample Application* (page 7) to run or debug this new application.

Note:

- If you are looking for how to run for other boards, please ref to *Board* (page 61).
- Please refer to *Application Development* (page 42) and *Build System based on Makefile* (page 17) for more information.
- If you want to access SoC related APIs, please use `nuclei_sdk_soc.h` header file.

- If you want to access SoC and board related APIs, please use `nuclei_sdk_hal.h` header file.
 - For simplified application development, you can use `nuclei_sdk_hal.h` directly.
-

2.6 Advanced Usage

For more advanced usage, please follow the items as below:

- Click *Design and Architecture* (page 53) to learn about Nuclei N100 SDK Design and Architecture, Board and SoC support documentation.
 - Click *Developer Guide* (page 17) to learn about Nuclei N100 SDK Build System and Application Development.
 - Click *Application* (page 67) to learn about each application usage and expected output.
-

Note:

- If you met some issues in using this guide, please check *FAQ* (page 85), if still not solved, please *Submit your issue* (page 50).
 - If you are trying to **develop Nuclei N100 SDK application in IDE**, now you have three choices:
 1. **Recommended:** Since Nuclei Studio 2025.02, Nuclei N100 SDK will be deeply integrated with Nuclei Studio IDE, you can easily create a Nuclei N100 SDK Project in Nuclei Studio through IDE Project Wizard, and easily configure selected Nuclei N100 SDK project using SDK Configuration Tool, for more details, please click [Nuclei Tools](#)¹⁶ to download Nuclei Studio IDE, and refer to the [Nuclei Studio and Nuclei Tools User Guide](#)¹⁷ for how to use it.
 2. You can take a try using IAR workbench, we provided prebuilt projects directly in Nuclei N100 SDK, just check the `ideprojects/iar` folder to learn about it.
-

¹⁶ <https://nucleisys.com/download.php>

¹⁷ https://doc.nucleisys.com/nuclei_tools/

DEVELOPER GUIDE

3.1 Code Style

In Nuclei N100 SDK, we use [EditorConfig](#)¹⁸ to maintain our development coding styles and [astyle](#)¹⁹ tool to format our source code.

- Our `editorconfig` file²⁰ is maintained in the root directory of Nuclei N100 SDK, called `.editorconfig`.
- Our `astyle` option file is maintained in the root directory of Nuclei N100 SDK, called `.astylerc`.

For example, if you want to format your application code(`.c/.h`) located in `application/baremetal/demo_timer`, you can run the following command:

```
# make sure astyle is present in PATH
which astyle
# format code
astyle --options=.astylerc --recursive application/baremetal/demo_timer/*.c,*.h
```

You can install `editorconfig` plugins for your editor, see <https://editorconfig.org/#download>.

We use `doxygen`²¹ to comment C/C++ source code.

3.2 Build System based on Makefile

Nuclei N100 SDK's build system is based on Makefile, user can build, run or debug application in Windows and Linux.

3.2.1 Makefile Structure

Nuclei N100 SDK's Makefiles mainly placed in `<NUCLEI_SDK_ROOT>/Build` directory and an extra *Makefile* located in `<NUCLEI_SDK_ROOT>/Makefile`.

This extra `<NUCLEI_SDK_ROOT>/Makefile` introduce a new Make variable called **PROGRAM** to provide the ability to build or run application in `<NUCLEI_SDK_ROOT>`.

For example, if you want to *rebuild and upload* application `application/baremetal/timer_test`, you can run `make PROGRAM=application/baremetal/timer_test clean upload` to achieve it.

The `<NUCLEI_SDK_ROOT>/Build` directory content list as below:

¹⁸ <https://editorconfig.org/>

¹⁹ <http://astyle.sourceforge.net/>

²⁰ <https://github.com/Nuclei-Software/nuclei-sdk/tree/master/.editorconfig>

²¹ <http://www.doxygen.nl/manual/docblocks.html>

```
gmsl/  
toolchain/  
Makefile.base  
Makefile.conf  
Makefile.core  
Makefile.components  
Makefile.files  
Makefile.global -> Created by user  
Makefile.misc  
Makefile.rtos  
Makefile.rules  
Makefile.soc
```

The file or directory is used explained as below:

Makefile.base

This **Makefile.base** file is used as Nuclei N100 SDK build system entry file, application's Makefile need to include this file to use all the features of Nuclei N100 SDK build system.

It will expose Make variables or options such as **BOARD** or **SOC** passed by make command, click [Makefile variables passed by make command](#) (page 24) to learn more.

This file will include optional [Makefile.global](#) (page 22) and [Makefile.local](#) (page 23) which allow user to set custom global Makefile configurations and local application Makefile configurations.

This file will include the following makefiles:

- [gmsl](#) (page 18): additional library functions provided via gmsl
- [toolchain](#) (page 19): additional library functions provided via gmsl
- [Makefile.misc](#) (page 19): misc functions and OS check helpers
- [Makefile.conf](#) (page 19): main Makefile configuration entry
- [Makefile.rules](#) (page 19): make rules of this build system

gmsl

The **gmsl** directory consist of the [GNU Make Standard Library \(GMSL\)](#)²², which is an a library of functions to be used with GNU Make's \$(call) that provides functionality not available in standard GNU Make.

We use this **gmsl** tool to make sure we help us achieve some linux command which is only supported in Linux.

²² <http://sourceforge.net/projects/gmsl/>

toolchain

The **toolchain** directory contains different toolchain support makefiles, such as Nuclei GNU toolchain, Nuclei LLVM toolchain and Terapines toolchain, if you want to add a different toolchain support, you also need to add a new toolchain makefile in it, you can refer to existing ones.

Since different toolchain support is added, in application Makefile, if your toolchain options are not compatible with others, to provide a compatible application for different toolchain, we recommend you to add `toolchain_$(TOOLCHAIN).mk` file in your application folder, and in application Makefile include this file, you can refer to `application/baremetal/benchmark/coremark` to see example usage.

Makefile.misc

This **Makefile.misc** file mainly provide these functions:

- Define **get_csrcs**, **get_asmsrcs**, **get_cxxsrcs** and **check_item_exist** make functions
 - **get_csrcs**: Function to get *.c or *.C source files from a list of directories, no ability to do recursive match. e.g. `$(call get_csrcs, csrc csrc/abc)` will return c source files in csrc and csrc/abc directories.
 - **get_asmsrcs**: Function to get *.s or *.S source files from a list of directories, no ability to do recursive match. e.g. `$(call get_asmsrcs, asmsrc asmsrc/abc)` will return asm source files in asmsrc and asmsrc/abc directories.
 - **get_cxxsrcs**: Function to get *.cpp or *.CPP source files from a list of directories, no ability to do recursive match. e.g. `$(call get_cxxsrcs, cppsrc cppsrc/abc)` will return cpp source files in cppsrc and cppsrc/abc directories.
 - **check_item_exist**: Function to check if item existed in a set of items. e.g. `$(call check_item_exist, sram, sram ilm)` will check sram whether existed in sram ilm, if existed, return sram, otherwise return empty.
- Check and define OS related functions, and also a set of trace print functions.

Makefile.conf

This **Makefile.conf** file will define the following items:

- Toolchain related variables used during compiling
- Debug related variables
- Include *Makefile.files* (page 20) and *Makefile.rtos* (page 21)
- Collect all the C/C++/ASM compiling and link options

Makefile.rules

This **Makefile.rules** file will do the following things:

- Collect all the sources during compiling
- Define all the rules used for building, uploading and debugging
- Print help message for build system

Makefile.files

This **Makefile.files** file will do the following things:

- Define common C/C++/ASM source and include directories
- Define common C/C++/ASM macros

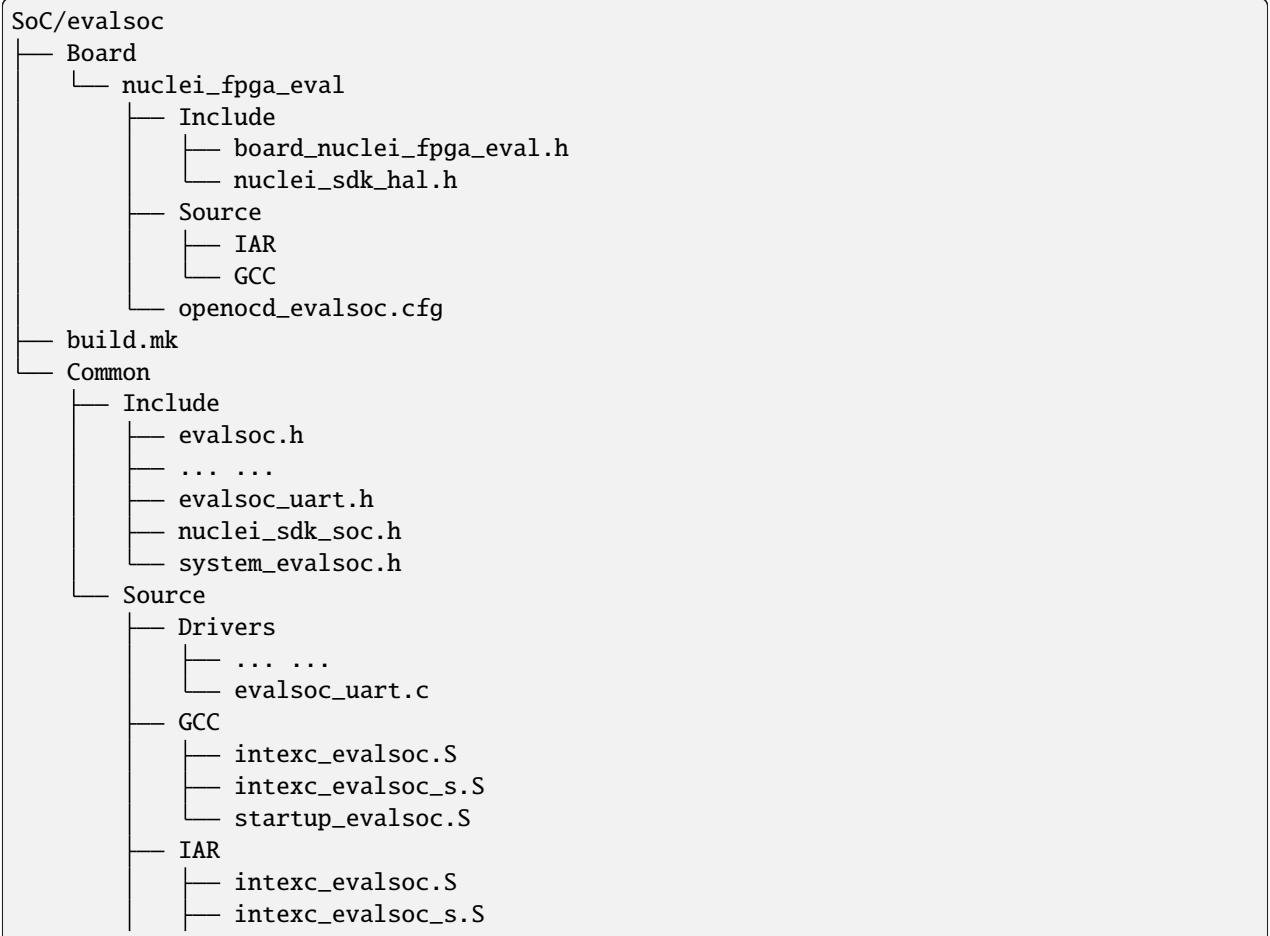
Makefile.soc

This **Makefile.soc** will include valid makefiles located in `<NUCLEI_SDK_ROOT>/SoC/<SOC>/build.mk` according to the *SOC* (page 24) makefile variable setting.

It will define the following items:

- **DOWNLOAD** and **CORE** variables
 - For *Nuclei Eval SoC* (page 59), we can support all the modes defined in *DOWNLOAD* (page 26), and **CORE** list defined in *Makefile.core* (page 22)
- Linker script used according to the **DOWNLOAD** mode settings
- OpenOCD debug configuration file used for the SoC and Board
- Some extra compiling or debugging options

A valid SoC should be organized like this, take evalsoc as example:



(continues on next page)

(continued from previous page)

```

├── startup_evalsoc.c
├── Stubs
│   ├── newlib
│   ├── libncrt
│   └── iardlib
├── evalsoc_common.c
└── system_evalsoc.c

```

Makefile.rtos

This **Makefile.rtos** will include `<NUCLEI_SDK_ROOT>/OS/<RTOS>/build.mk` according to our *RTOS* (page 31) variable.

A valid rtos should be organized like this, take UCOSII as example:

```

OS/UCOSII/
├── arch
├── build.mk
├── license.txt
├── readme.md
└── source

```

If no *RTOS* (page 31) is chosen, then RTOS code will not be included during compiling, user will develop baremetal application.

If **FreeRTOS**, **UCOSII** or **RTThread** RTOS is chosen, then FreeRTOS UCOSII, or RTThread source code will be included during compiling, and extra compiler option `-DRTOS_$(RTOS_UPPER)` will be passed, then user can develop RTOS application.

For example, if FreeRTOS is selected, then `-DRTOS_FREERTOS` compiler option will be passed.

Makefile.components

This **Makefile.components** will include `build.mk` Makefiles of selected components defined via makefile variable *MIDDLEWARE* (page 32), the Makefiles are placed in the sub-folders of `<NUCLEI_SDK_ROOT>/Components/`.

A valid middleware component should be organized like this, take `fatfs` as example :

```

Components/fatfs/
├── build.mk
├── documents
├── LICENSE.txt
└── source

```

For example, if there are two valid middleware components in `<NUCLEI_SDK_ROOT>/Components/`, called `fatfs` and `tjpgd`, and you want to use them in your application, then you can set *MIDDLEWARE* like this `MIDDLEWARE := fatfs tjpgd`, then the application will include these two middlewares into build process.

Makefile.core

This **Makefile.core** is used to define the RISC-V ARCH and ABI used during compiling of the CORE list supported.

If you want to add a new **CORE**, you need to add a new line before **SUPPORTED_CORES**, and append the new **CORE** to **SUPPORTED_CORES**.

For example, if you want to add a new **CORE** called **n101**, and the **n101**'s **ARCH** and **ABI** are **rv32imac** and **ilp32**, then you can add a new line like this **N101_CORE_ARCH_ABI = rv32imac ilp32**, and append **n101** to **SUPPORTED_CORES** like this **SUPPORTED_CORES = n100e n100em n100ezmmul n100 n100m n100zmmul n101**

Note:

- The appended new **CORE** need to lower-case, e.g. *n101*
 - The new defined variable **N101_CORE_ARCH_ABI** need to be all upper-case.
-

Makefile.global

This **Makefile.global** file is an optional file, and will not be tracked by git, user can create own **Makefile.global** in **<NUCLEI_SDK_ROOT>/Build** directory.

In this file, user can define custom **SOC**, **BOARD**, **DOWNLOAD** options to overwrite the default configuration.

For example, if you will use only the *Nuclei FPGA Evaluation Kit* (page 61), you can create the **<NUCLEI_SDK_ROOT>/Build/Makefile.global** as below:

```
SOC ?= evalsoc
BOARD ?= nuclei_fpga_eval
DOWNLOAD ?= sram
```

Note:

- If you add above file, then you can build, run, debug application without passing **SOC**, **BOARD** and **DOWNLOAD** variables using make command for *Nuclei FPGA Evaluation Kit* (page 61) board, e.g.
 - Build and run application for *Nuclei FPGA Evaluation Kit* (page 61): **make run**
 - Debug application for *Nuclei FPGA Evaluation Kit* (page 61): **make debug**
 - If you create the **Makefile.global** like above sample code, you will also be able to use Nuclei N100 SDK build system as usually, it will only change the default **SOC**, **BOARD** and **DOWNLOAD**, but you can still override the default variable using make command, such as **make SOC=evalsoc BOARD=nuclei_fpga_eval DOWNLOAD=sram**
-

Makefile.local

As the *Makefile.global* (page 22) is used to override the default Makefile configurations, and the **Makefile.local** is used to override application level Makefile configurations, and also this file will not be tracked by git.

User can create `Makefile.local` file in any of the application folder, placed together with the application Makefile, for example, you can create `Makefile.local` in `application/baremetal/helloworld` to override default make configuration for this **helloworld** application.

If you want to change the default board for **helloworld** to use *Nuclei FPGA Evaluation Kit* (page 61), you can create `application/baremetal/helloworld/Makefile.local` as below:

```
SOC ?= evalsoc
BOARD ?= nuclei_fpga_eval
DOWNLOAD ?= sram
```

Note:

- This local make configuration will override global and default make configuration.
- If you just want to change only some applications' makefile configuration, you can add and update `Makefile.local` for those applications.

3.2.2 Makefile targets of make command

Here is a list of the *Make targets supported by Nuclei N100 SDK Build System* (page 23).

Table 1: Make targets supported by Nuclei N100 SDK Build System

target	description
help	display help message of Nuclei N100 SDK build system
info	display selected configuration information
showflags	display asm/c/cxx/ld flags and other info
showtoolver	display toolchain/qemu/openocd version
all	build application with selected configuration
clean	clean application with selected configuration
dasm	build and disassemble application with selected configuration
bin	build and generate application binary with selected configuration
upload	build and upload application with selected configuration
run_openocd	run openocd server with selected configuration, and wait for gdb at port specified by \$(GDB_PORT)
run_gdb	build and start gdb process with selected configuration, and connect to local-host:\$(GDB_PORT)
debug	build and debug application with selected configuration
run_qemu	run application on qemu machine with selected configuration
run_xlspike	run application on xlspike with selected configuration
size	show program size

Note:

- The selected configuration is controlled by *Makefile variables passed by make command* (page 24)

- For `run_openocd` and `run_gdb` target, if you want to change a new gdb port, you can pass the variable `GDB_PORT` (page 29)
 - For `run_qemu`, only `SOC=evalsoc` supported, when do this target, you can pass `SIMU=qemu` to support auto-exit, project recompiling is required.
 - For `run_xlspike`, only `SOC=evalsoc` supported, when do this target, you can pass `SIMU=xlspike` to support auto-exit, project recompiling is required.
-

3.2.3 Makefile variables passed by make command

In Nuclei N100 SDK build system, we exposed the following Makefile variables which can be passed via make command.

- `SOC` (page 24)
- `BOARD` (page 25)
- `VARIANT` (page 25)
- `TOOLCHAIN` (page 26)
- `DOWNLOAD` (page 26)
- `CORE` (page 27)
- `ARCH_EXT` (page 27)
- `CPU_SERIES` (page 28)
- `SIMULATION` (page 29)
- `SEMIHOST` (page 28)
- `GDB_PORT` (page 29)
- `V` (page 30)
- `SILENT` (page 30)

Note:

- These variables can also be used and defined in application Makefile
 - If you just want to fix your running board of your application, you can just define these variables in application Makefile, if defined, then you can simply use `make clean`, `make upload` or `make debug`, etc.
-

SOC

`SOC` variable is used to declare which SoC is used in application during compiling.

`evalsoc` is the default SoC, if no `SOC` passed or environment variable set, you can check default settings by run `make info`, it will show default settings without any overriding make variable.

You can easily find the supported SoCs in the `<NUCLEI_SDK_ROOT>/SoC` directory.

Currently we support the following SoCs, see [Supported SoCs](#) (page 25).

Table 2: Supported SoCs

SOC	Reference
evalsoc	<i>Nuclei Eval SoC</i> (page 59)

Note: If you are our SoC subsystem customer, in the SDK delivered to you, you can find your soc name in this <NUCLEI_SDK_ROOT>/SoC directory, take ns SoC as example, when SOC=ns, the SoC source code in <NUCLEI_SDK_ROOT>/SoC/ns/Common will be used.

This documentation just document the open source version of Nuclei N100 SDK's supported SOC and Board.

BOARD

BOARD variable is used to declare which Board is used in application during compiling.

The **BOARD** variable should match the supported boards of chosen **SOC**. You can easily find the supported Boards in the <NUCLEI_SDK_ROOT>/<SOC>/Board/ directory.

- *Supported Boards when SOC=evalsoc* (page 25)

Currently we support the following SoCs.

Table 3: Supported Boards when SOC=evalsoc

BOARD	Reference
nu- clei_fpga_eval	<i>Nuclei FPGA Evaluation Kit</i> (page 61)

Note:

- If you only specify **SOC** variable in make command, it will use default **BOARD** and **CORE** option defined in <NUCLEI_SDK_ROOT>/SoC/<SOC>/build.mk
- If you are our SoC subsystem customer, in the SDK delivered to you, you can check the board supported list in <NUCLEI_SDK_ROOT>/<SOC>/Board/, take SOC=ns BOARD=fpga_eval as example, the board source code located <NUCLEI_SDK_ROOT>/ns/Board/fpga_eval will be used.

VARIANT

VARIANT variable is used to declare which variant of board is used in application during compiling.

It might only affect on only small piece of board, and this is SoC and Board dependent.

This variable only affect the selected board or soc, and it is target dependent.

TOOLCHAIN

This variable is used to select different toolchain to compile application. Currently we support 3 toolchain in Nuclei N100 SDK.

- **nuclei_gnu**: default, it will choose nuclei gnu toolchain, distributed with Nuclei Toolchain.
- **nuclei_llvm**: supported for N100, still in experiment, nuclei customized extensions not yet supported, distributed with Nuclei Toolchain.
- **terapines**: still in experiment, it depends on the toolchain vendor about the supported extensions, if you want to take a try with it, just visit <https://www.terapines.com/> and request an terapines toolchain evaluation.

For **nuclei_gnu/nuclei_llvm** toolchain both newlib and libncrt library are supported, but nuclei_llvm toolchain multilib selection mechanism is not as good as gnu toolchain, you need to take care of the arch isa string order, please see `riscv64-unknown-unknown-elf-clang -v` output for supported multilib and its isa string order.

And IAR compiler support is also done in Nuclei N100 SDK(not yet ready for N100), you can take a try with it via `ideprojects/iar` folder provided prebuilt ide projects.

DOWNLOAD

DOWNLOAD variable is used to declare the download mode of the application, currently it has these modes supported as described in table *Supported download modes* (page 26)

Table 4: Supported download modes

DOWN-LOAD	Description
sram	Program will be downloaded into sram and run directly in sram, program will lost when poweroff

Note:

- This variable now target dependent, and its meaning depending on how this variable is implemented in SoC's build.mk
- macro `DOWNLOAD_MODE` and `DOWNLOAD_MODE_STRING` will be defined in Makefile, eg. when `DOWNLOAD=sram`, macro will be defined as `-DDOWNLOAD_MODE=DOWNLOAD_MODE_SRAM`, and `-DDOWNLOAD_MODE_STRING=\ "sram\"`, the sram will be in upper case, currently `DOWNLOAD_MODE_STRING` macro is used in `system_<Device>.c` when banner is print.

CORE

CORE variable is used to declare the Nuclei processor core of the application.

Currently it has these cores supported as described in table *Supported Nuclei Processor cores* (page 27).

Table 5: Supported Nuclei Processor cores

CORE	ARCH	ABI	TUNE
n100e	rv32ec	ilp32e	nuclei-100-series
n100em	rv32emc	ilp32e	nuclei-100-series
n100ezmmul	rv32ec_zmmul	ilp32e	nuclei-100-series
n100	rv32ic	ilp32	nuclei-100-series
n100m	rv32imc	ilp32	nuclei-100-series
n100zmmul	rv32ic_zmmul	ilp32	nuclei-100-series

When **CORE** is selected, the **ARCH**, **ABI** and **TUNE** (optional) are set, and it might affect the compiler options in combination with **ARCH_EXT** (page 27) depended on the implementation of SoC build.mk.

Take SOC=evalsoc as example.

- If **CORE=n100zmmul ARCH_EXT=_zca_zcb_zcmp_zcmt**, then **ARCH=rv32i_zmmul_zca_zcb_zcmp_zcmt**, **ABI=ilp32** **TUNE=nuclei-100-series**. riscv arch related compile and link options will be passed, for this case, it will be **-march=rv32i_zmmul_zca_zcb_zcmp_zcmt -mabi=ilp32 -mtune=nuclei-100-series**.
- If **CORE=n100e ARCH_EXT=_zicond**, it will be **-march=rv32ec_zicond -mabi=ilp32e -mtune=nuclei-100-series**.

For riscv code model settings, the **RISCV_CMODEL** variable will be set to medlow for RV32 targets, otherwise it will be medany.

ARCH_EXT

ARCH_EXT variable is used to select extra RISC-V arch extensions supported by Nuclei RISC-V Processor, except the **iemafdc**.

Note: *Nuclei Toolchain 2023.10*²³ now bump gcc version from gcc 10 to gcc 13, which introduced incompatible **-march** option, so **ARCH_EXT** usage is also incompatible now.

About the incompatible **march** option change, please see <https://github.com/riscv-non-isa/riscv-toolchain-conventions/pull/26>, which is already present in latest gcc and clang release.

Here are several examples when using **ARCH_EXT** only for **Nuclei 100 series** RISC-V Processors:

- If you want to use just **Zicond extension**²⁴, you can pass **ARCH_EXT=_zicond**
- If you want to use **Zc 1.0 extension**²⁵
 - You can use it together with **C** extension, which means it should be concat with isa string like **rv32im_zca_zcb_zcmp_zcmt**
 - In Nuclei N100 SDK, the isa string processing is done in build system
 - If you want to use with 100 series, you can pass **ARCH_EXT=_zca_zcb_zcmp_zcmt**

²³ <https://github.com/riscv-mcu/riscv-gnu-toolchain/releases/tag/nuclei-2023.10>

²⁴ <https://github.com/riscvarchive/riscv-zicond>

²⁵ <https://github.com/riscv/riscv-code-size-reduction/releases/tag/v1.0.4-3>

- If you want to use both **Zicond** and **Zc** extension, you can pass **ARCH_EXT=_zca_zcb_zcmp_zcmt_zicond**
- You can check prebuilt multilib for gcc and clang using `riscv64-unknown-elf-gcc --print-multi-lib` and `riscv64-unknown-elf-clang --print-multi-lib`

It is suggested to use this **ARCH_EXT** with other arch options like this, can be found in `SoC/evalsoc/build.mk`:

```
# Set RISCV_ARCH and RISCV_ABI
CORE_UPPER := $(call uc, $(CORE))
CORE_ARCH_ABI := $($$(CORE_UPPER)_CORE_ARCH_ABI)
RISCV_ARCH ?= $(word 1, $($$(CORE_ARCH_ABI))$(ARCH_EXT))
RISCV_ABI ?= $(word 2, $($$(CORE_ARCH_ABI)))
```

CPU_SERIES

This variable will be auto set if your CORE variable match the following rules:

- **100**: CORE start with `10`, the CPU_SERIES will be 100.

It can also be defined in Makefile itself directly or passed via make command.

It will also define an macro called **CPU_SERIES**, eg. for CPU_SERIES=200, it will define macro CPU_SERIES=200.

This variable is currently used in benchmark cases, and require application Makefile changes.

SEMIHOST

If **SEMIHOST=1**, it means it will enable semihost support using openocd.

From 0.5.0, both newlib and libnrt support semihosting feature, and when using semihost, no need to implement the clib stub functions, which is done by newlib or libnrt semihosting library.

And for Nuclei QEMU >= 2023.10 version, you can also use semihosting feature, simple usage is like below for qemu:

```
cd application/baremetal/helloworld
# clean project first
make SOC=evalsoc SEMIHOST=1 clean
make SOC=evalsoc SEMIHOST=1 all
# run on qemu, SEMIHOST=1 is required to pass when run qemu
make SOC=evalsoc SEMIHOST=1 run_qemu
```

When using semihosting feature with openocd, debug message will print via openocd console.

You need to use it like this(assume you are run on evalsoc, CORE=n100):

In terminal 1, open openocd and monitor the output:

```
cd application/baremetal/helloworld
make SOC=evalsoc CORE=n100 run_openocd
# when terminal 2 has download program and start to run, you will be able to see output.
↪here
```

In terminal 2, gdb connect to the openocd exposed gdb port and load program, and run

```
# in normal shell terminal
cd application/baremetal/helloworld
make SOC=evalsoc CORE=n100 SEMIHOST=1 clean
```

(continues on next page)

(continued from previous page)

```

make SOC=evalsoc CORE=n100 SEMIHOST=1 run_gdb

# now in gdb command terminal, run the following command
monitor reset halt
load
## when run continue, you will be able to see output in previous terminal 1 running.
↪ openocd
continue

```

SIMULATION

If **SIMULATION=1**, it means the program is optimized for hardware simulation environment.

Currently if **SIMULATION=1**, it will pass compile option **-DCFG_SIMULATION**, application can use this **CFG_SIMULATION** to optimize program for hardware simulation environment.

Note:

- Currently the benchmark applications in **application/baremetal/benchmark** used this optimization
-

GDB_PORT

This variable is not used usually, by default the **GDB_PORT** variable is 3333.

If you want to change a debug gdb port for openocd and gdb when run **run_openocd** and **run_gdb** target, you can pass a new port such as 3344 to this variable.

For example, if you want to debug application using **run_openocd** and **run_gdb** and specify a different port other than 3333.

You can do it like this, take **nuclei_fpga_eval** board for example, such as port 3344:

- Open openocd server: `make SOC=evalsoc BOARD=nuclei_fpga_eval CORE=n100 GDB_PORT=3344 run_openocd`
- connect gdb with openocd server: `make SOC=evalsoc BOARD=nuclei_fpga_eval CORE=n100 GDB_PORT=3344 run_gdb`

JTAGSN

This variable is used specify jtag adapter serial number in openocd configuration, need to be supported in openocd configuration file and makefile, currently **evalsoc** is supported. It is used by openocd adapter **serial**.

Assume you have a jtag adapter, serial number is FT6S9RD6, and you want to download program through this jtag to a fpga with ux900 bitstream on it, you can do it like this.

For windows, you need to pass extra A, eg. **JTAGSN=FT6S9RD6A**

```

# cd to helloworld
cd application/baremetal/helloworld
# clean program
make SOC=evalsoc JTAGSN=FT6S9RD6 clean

```

(continues on next page)

(continued from previous page)

```
# upload program
make SOC=evalsoc JTAGSN=FT6S9RD6 upload
```

BANNER

If **BANNER=0**, when program is rebuilt, then the banner message print in console will not be print, banner print is default enabled via `NUCLEI_BANNER=1` in `nuclei_sdk_hal.h`.

when **BANNER=0**, an macro `-DNUCLEI_BANNER=0` will be passed in Makefile.

The banner message looks like this:

```
Nuclei N100 SDK Build Time: Jul 23 2021, 10:22:50
Download Mode: SRAM
CPU Frequency 15999959 Hz
```

V

If **V=1**, it will display compiling message in verbose including compiling options.

By default, no compiling options will be displayed in make console message just to print less message and make the console message cleaner. If you want to see what compiling option is used, please pass **V=1** in your make command.

SILENT

If **SILENT=1**, it will not display any compiling message.

If you don't want to see any compiling message, you can pass **SILENT=1** in your make command.

3.2.4 Makefile variables used only in Application Makefile

The following variables should be used in application Makefile at your demand, e.g. `application/baremetal/demo_timer/Makefile`.

- *TARGET* (page 31)
- *NUCLEI_SDK_ROOT* (page 31)
- *MIDDLEWARE* (page 32)
- *RTOS* (page 31)
- *STDCLIB* (page 32)
- *RISCV_ARCH* (page 36)
- *RISCV_ABI* (page 36)
- *RISCV_CMODEL* (page 36)
- *RISCV_TUNE* (page 36)
- *NOGC* (page 37)
- *RTTHREAD_MSH* (page 37)

TARGET

This is a necessary variable which must be defined in application Makefile.

It is used to set the name of the application, it will affect the generated target filenames.

Warning:

- Please don't put any spaces in TARGET variable
- The variable shouldn't contain any space

```
# invalid case 1
TARGET ?= hello world
# invalid case 2
TARGET ?= helloworld # before this # there is a extra space
```

NUCLEI_SDK_ROOT

This is a necessary variable which must be defined in application Makefile.

It is used to set the path of Nuclei N100 SDK Root, usually it should be set as relative path, but you can also set absolute path to point to Nuclei N100 SDK.

RTOS

RTOS variable is used to choose which RTOS will be used in this application.

You can easily find the supported RTOSes in the <NUCLEI_SDK_ROOT>/OS directory.

- If **RTOS** is not defined, then baremetal service will be enabled with this application. See examples in application/baremetal.
- If **RTOS** is set the the following values, RTOS service will be enabled with this application.
 - FreeRTOS: FreeRTOS service will be enabled, extra macro RTOS_FREERTOS will be defined, you can include FreeRTOS header files now, and use FreeRTOS API, for FreeRTOS application, you need to have an FreeRTOSConfig.h header file prepared in you application. See examples in application/freertos.
 - UCOSII: UCOSII service will be enabled, extra macro RTOS_UCOSII will be defined, you can include UCOSII header files now, and use UCOSII API, for UCOSII application, you need to have app_cfg.h, os_cfg.h and app_hooks.c files prepared in you application. See examples in application/ucosii.
 - RTThread: RT-Thread service will be enabled, extra macro RTOS_RTTHREAD will be defined, you can include RT-Thread header files now, and use RT-Thread API, for UCOSII application, you need to have an rtconfig.h header file prepared in you application. See examples in application/rthread.

MIDDLEWARE

MIDDLEWARE variable is used to select which middlewares should be used in this application.

You can easily find the available middleware components in the `<NUCLEI_SDK_ROOT>/Components` directory.

- If **MIDDLEWARE** is not defined, not leave empty, no middleware package will be selected.
- If **MIDDLEWARE** is defined with more than 1 string, such as `fatfs tjpgd`, then these two middlewares will be selected.

STDCLIB

STDCLIB variable is used to select which standard c runtime library will be used. If not defined, the default value will be `newlib_nano`.

In Nuclei GNU Toolchain, we distributed `newlib/newlib-nano/Nuclei` c runtime library, so user can select different c runtime library according to their requirement.

Newlib is a simple ANSI C library, math library, available for both RV32 and RV64.

Nuclei C runtime library is a highly optimized c library designed for deeply embedded user cases, can provided smaller code size and highly optimized floating point support compared to Newlib.

To support both gcc and clang compiler, we decided not to use `--specs=` option to select system library, instead of that, we start to use `--no-defaultlibs` options, and link the required system libraries by the **STDCLIB** variable choice, so need to link desired libraries such as:

- `-lgcc`: a standard library (linked by default, excluded by `-no-defaultlibs`) that provides internal subroutines to overcome shortcomings of particular machines, see <https://gcc.gnu.org/onlinedocs/gccint/Libgcc.html>.
- `-lgcov`: a library used to test coverage program, known as `gcov/gprof`, see <https://gcc.gnu.org/onlinedocs/gcc/Gcov.html>
- `-lc/-lc_nano`: newlib c library or newlib nano c library, see <https://sourceware.org/newlib/docs.html>
- `-lm`: newlib math library, see <https://sourceware.org/newlib/libm.html>
- `-lstdc++`: gnu standard c++ library, see <https://gcc.gnu.org/onlinedocs/libstdc++>
- `-lsemihost`: riscv semihosting library which implement a set of standard I/O and file I/O operations, see <https://github.com/riscv-mcu/riscv-newlib/tree/nuclei/newlib-4.3.0/libgloss/riscv>
- `-lnosys`: a set of stub functions which implement a set of standard I/O operations but does nothing, and when link with it, it will throw link warning, see <https://github.com/riscv-mcu/riscv-newlib/blob/nuclei/newlib-4.3.0/libgloss/libnosys>
- `-ln crt_pico/-ln crt_nano/-ln crt_small/-ln crt_balanced/-ln crt_fast`: Nuclei `libn crt` library, it provides `pico/nano/small/balanced/fast` variant to provide standard c library, math library, and `libgcc` library features, and need to use together with `-lheapops_minimal/-lheapops_basic/-lheapops_realtime` heap operation API, and `-lfileops_uart/-lfileops_semi/-lfileops_rtt` file io operation API, when using this `libn crt` library, please don't link `-lgcc -lc_nano/-lc -lm -lsemihost -lnosys`, and it also can't link with `-lstdc++`
- Upgrading `libn crt` from Nuclei GNU Toolchain 2022.12 to Nuclei Toolchain 2023.10, please change it like this, take `libn crt_small` as example:
 - **asm/c/c++ options**: `--specs=libn crt_small.specs -> --specs=libn crt_small.specs` works for gcc, or `-isystem=/include/libn crt` works for both gcc and clang

- **ld options:** `--specs=libncrt_small.specs -> --specs=libncrt_small.specs`
`-lheapops_basic -lfileops_uart` works for gcc, `-nodefaultlibs -lncrt_small`
`-lheapops_basic -lfileops_uart` works for both gcc and clang
- We recommend you to use later version works for both gcc and clang, `-nodefaultlibs` is used to exclude startup crt, libgcc and c library in default gcc or clang, use the version specified by us to use libncrt.

Table 6: Available STDCLIB choices

STDCLIB	Description
<code>newlib_full</code>	Normal version of newlib, optimized for speed at cost of size. It provided full feature of newlib, with file io supported.
<code>newlib_fast</code>	Newlib nano version, with printf float and scanf float support.
<code>newlib_small</code>	Newlib nano version, with printf float support.
<code>newlib_nano</code>	Newlib nano version, without printf/scanf float support.
<code>libncrt_fast</code>	Nuclei C runtime library optimized for speed, full feature
<code>libncrt_balanced</code>	Nuclei C runtime library balanced at speed and code size, full feature
<code>libncrt_small</code>	Nuclei C runtime library optimized for code size, full feature
<code>libncrt_nano</code>	Nuclei C runtime library optimized for code size, without float/double support
<code>libncrt_pico</code>	Nuclei C runtime library optimized for code size, without long/long long/float/double support
<code>nostd</code>	no std c library will be used, and don't search the standard system directories for header files
<code>nospec</code>	no std c library will be used, not pass any <code>-specs</code> options

Note:

- For clang based compiler, if `-u _print_float` is not passed in linker options, it may fail during link process, so here we pass `-u _print_float` for `newlib_nano`, then it means for `nuclei_llvm` and `terapines` toolchain, `STDCLIB=newlib_nano` equals to `STDCLIB=newlib_small`
- Nuclei libncrt library couldn't be used with `terapines` toolchain, so you can't use any libncrt library when you are using `terapines` toolchain.
- About Newlib and Newlib nano difference, please check <https://github.com/riscv-collab/riscv-newlib/blob/riscv-newlib-3.2.0/newlib/README>
- About Nuclei C runtime library, it provided basic libgcc, c library and math library feature, but it didn't provided all the features that newlib can do, it is highly optimized for deeply embedded scenery, user no need to link with `-lm` when using libncrt library when math library is needed.
- Nuclei C runtime library is only available in Nuclei GNU Toolchain released after Nov 2021, about how to use this library, please follow doc located in `gcc\share\pdf`, changes need to be done in startup code, linker script, stub code, and compiler options, you can check commit history of `nuclei sdk` for support of libncrt.
- Nuclei C runtime library(libncrt) only support RV32 CPU target, so you cannot use it with RV64 CPU.
- Since there are different c runtime library can be chosen now, so developer need to provide different stub functions for different library, please check `SoC/evalsoc/Common/Source/Stubs/` and `SoC/evalsoc/build.mk` for example.

NCRTHEAP

This variable is only valid when using libnrcrt c library \geq v3.0.0, and you can choose different heapops when using libnrcrt c library to do heap related operations such as malloc or free.

- **basic**: default, this is previous release of libnrcrt c library used one. A low-overhead best-fit heap where allocation and deallocation have very little internal fragmentation
- **realtime**: A real-time heap where allocation and deallocation have O(1) performance
- **minimal**: An allocate-only heap where deallocation and reallocation are not implemented

For previous libnrcrt library, this heapops is default binded with libnrcrt library, so you can't choose different heap type, but now you can choose according to your requirements.

NCRTIO

This variable is only valid when using libnrcrt c library \geq v3.0.0, and you can choose different fileops when using libnrcrt c library to do basic input/output operations.

- **uart**: default, lower level input/output via uart, developer need to implement metal_tty_putc/getc
- **semi**: input/output via semihosting, if you pass **SEMIHOST=1** in make, it will default choose this one when using libnrcrt library.
- **rtt**: input/output via jlink rtt, require to use JLink tool.

SMP

SMP variable is used to control smp cpu core count, valid number must > 1 .

When **SMP** variable is defined, extra gcc options for ld is passed `-Wl,--defsym=__SMP_CPU_CNT=$(SMP)`, and extra c macro `-DSMP_CPU_CNT=$(SMP)` is defined this is passed in each SoC's build.mk, such as SoC/evalsoc/build.mk.

When **SMP** variable is defined, extra openocd command `set SMP $(SMP)` will also be passed when run openocd upload or create a openocd server.

For **SMP** application, please check `application/baremetal/smphello`, if you want to implement a smp application, you need to reimplement `smp_main`, which all harts will run to this function instead of `main`, if you don't implement it, a weak `smp_main` in `startup_<Device>.S` will be used, and only boot hartid specified by **BOOT_HARTID** will enter to main, other harts will do wfi.

BOOT_HARTID

This variable is used to control the boot hartid in a multiple core system. If **SMP** variable is specified, it means this application is expected to be a smp application, otherwise it means this application is expected to be a amp application.

For amp application, only the boot hart specified by **BOOT_HARTID** will run, other harts will directly do wfi when startup, but for smp application, other hartid will do normal boot code instead of code/data/bss init, and do sync harts to make sure all harts boots.

For both amp and smp application, the program should execute on a share memory which all harts can access, not hart private memory such as ilm/dlm.

Currently **SMP** and **BOOT_HARTID** support all require SOC support code to implement it, currently evalsoc support it, currently qemu simulation didn't work for SMP/AMP use case.

Here is some basic usage for **SMP** and **BOOT_HARTID** on UX900 x4, run on external ddr.

```
# cd to helloworld
cd <Nuclei N100 SDK>/application/baremetal/helloworld
# clean program
make SOC=evalsoc clean
# AMP: choose hart 1 as boot hartid, other harts spin
make SOC=evalsoc BOOT_HARTID=1 DOWNLOAD=ddr clean upload
cd <Nuclei N100 SDK>/application/baremetal/smphello
# SMP: choose hart 2 as boot hartid
make SOC=evalsoc BOOT_HARTID=2 SMP=4 DOWNLOAD=ddr clean upload
```

HARTID_OFS

This variable is used to set hartid offset relative to real hart index in a complex AMP SoC system.

eg.

In a SoC system, it has 2 CPU, CPU 0 has 2 smp core, CPU 1 has 1 core, and CPU 0 hartid is 0, 1, and CPU 1 hartid is 2, so for CPU 0, HARTID_OFS is 0, for CPU 1, HARTID_OFS is 2.

STACKSZ

STACKSZ variable is used to control the per core stack size reserved in linker script, this need to cooperate with link script file and linker options.

In link script file, `__STACK_SIZE` symbol need to use `PROVIDE` feature of `ld` to define a weak version, such as `PROVIDE(__STACK_SIZE = 2K);`, and `gcc` will pass `ld` options `-Wl,--defsym=__STACK_SIZE=$(STACKSZ)` to overwrite the default value if **STACKSZ** is defined.

STACKSZ variable must be a valid value accepted by `ld`, such as `0x2000`, `2K`, `4K`, `8192`.

For SMP version, stack size space need to reserve **STACKSZ** x SMP Core Count size.

You can refer to `SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/gcc_evalsoc_sram.ld` for smp version.

HEAPSZ

HEAPSZ variable is used to control the heap size reserved in linker script, this need to cooperate with link script file and linker options.

In link script file, `__HEAP_SIZE` symbol need to use `PROVIDE` feature of `ld` to define a weak version, such as `PROVIDE(__HEAP_SIZE = 2K);`, and `gcc` will pass `ld` options `-Wl,--defsym=__HEAP_SIZE=$(HEAPSZ)` to overwrite the default value if **HEAPSZ** is defined.

HEAPSZ variable must be a valid value accepted by `ld`, such as `0x2000`, `2K`, `4K`, `8192`.

RISCV_ARCH

RISCV_ARCH variable is used to control compiler option `-mmodel=$(RISCV_ARCH)`.

It might override **RISCV_ARCH** defined in SoC build.mk, according to your build.mk implementation.

RISCV_ARCH might directly affect the gcc compiler option depended on the implementation of SoC build.mk.

Take `SOC=evalsoc` for example.

- **CORE=n100 RISCV_ARCH=rv32imc_zicond RISCV_ABI=ilp32 ARCH_EXT=_zba_zbb_zbc_zbs**, then final compiler options will be `-march=rv32imc_zicond -mabi=ilp32 -mtune=nuclei-100-series`. The **ARCH_EXT** is ignored.

RISCV_ABI

RISCV_ABI variable is used to control compiler option `-mmodel=$(RISCV_ABI)`.

It might override **RISCV_ABI** defined in SoC build.mk, according to your build.mk implementation.

RISCV_CMODEL

RISCV_CMODEL is used to control compiler option `-mmodel=$(RISCV_CMODEL)`.

For RV32, default value is `medlow`, otherwise `medany` for RV64.

You can set **RISCV_CMODEL** to override predefined value.

RISCV_TUNE

RISCV_TUNE is used to control compiler option `-mtune=$(RISCV_TUNE)`.

It is defined in SoC build.mk, you can override it if your implementation allow it.

APP_COMMON_FLAGS

Note:

- Added in 0.4.0 release.
-

This variable is used to define app common compiler flags to all `c/asm/cpp` compiler. You can pass it via `make` command to define extra flags to compile application.

APP_ASMFLAGS

This variable is similiar to **APP_COMMON_FLAGS** but used to pass extra app asm flags.

APP_CFLAGS

This variable is similar to **APP_COMMON_FLAGS** but used to pass extra app c flags.

APP_CXXFLAGS

This variable is similar to **APP_COMMON_FLAGS** but used to pass extra app cxx flags.

APP_LDFLAGS

This variable is similar to **APP_COMMON_FLAGS** but used to pass extra app linker flags.

NOGC

NOGC variable is used to control whether to enable gc sections to reduce program code size or not, by default GC is enabled to reduce code size.

When GC is enabled, these options will be added:

- Adding to compiler options: `-ffunction-sections -fdata-sections`
- Adding to linker options: `-Wl,--gc-sections -Wl,--check-sections`

If you want to enable this GC feature, you can set **NOGC=0** (default), GC feature will remove sections for you, but sometimes it might remove sections that are useful, e.g. For Nuclei N100 SDK test cases, we use ctest framework, and we need to set **NOGC=1** to disable GC feature.

When **NOGC=0** (default), extra compile options `-ffunction-sections -fdata-sections`, and extra link options `-Wl,--gc-sections -Wl,--check-sections` will be passed.

RTTHREAD_MSH

RTTHREAD_MSH variable is valid only when **RTOS** is set to **RTThread**.

When **RTTHREAD_MSH** is set to **1**:

- The RTThread MSH component source code will be included
- The MSH thread will be enabled in the background
- Currently the msh getchar implementation is using a weak function implemented in `rt_hw_console_getchar` in `OS/RTThread/libcpu/risc-v/nuclei/cpuport.c`

3.2.5 Build Related Makefile variables used only in Application Makefile

If you want to specify additional compiler flags, please follow this guidance to modify your application Makefile.

Nuclei N100 SDK build system defined the following variables to control the build options or flags.

- *INCDIRS* (page 38)
- *C_INCDIRS* (page 38)
- *CXX_INCDIRS* (page 38)
- *ASM_INCDIRS* (page 39)
- *SRC_DIRS* (page 39)

- *C_SRCDIRS* (page 39)
- *CXX_SRCDIRS* (page 39)
- *ASM_SRCDIRS* (page 39)
- *C_SRCS* (page 40)
- *CXX_SRCS* (page 40)
- *ASM_SRCS* (page 40)
- *EXCLUDE_SRCS* (page 40)
- *COMMON_FLAGS* (page 40)
- *CFLAGS* (page 41)
- *CXXFLAGS* (page 41)
- *ASMFLAGS* (page 41)
- *LDFLAGS* (page 41)
- *LDLIBS* (page 41)
- *LIBDIRS* (page 41)
- *LINKER_SCRIPT* (page 42)

INCDIRS

This **INCDIRS** is used to pass C/CPP/ASM include directories.

e.g. To include current directory `.` and `inc` for C/CPP/ASM

```
INCDIRS = . inc
```

C_INCDIRS

This **C_INCDIRS** is used to pass C only include directories.

e.g. To include current directory `.` and `cinc` for C only

```
C_INCDIRS = . cinc
```

CXX_INCDIRS

This **CXX_INCDIRS** is used to pass CPP only include directories.

e.g. To include current directory `.` and `cppinc` for CPP only

```
CXX_INCDIRS = . cppinc
```

ASM_INCDIRS

This **ASM_INCDIRS** is used to pass ASM only include directories.

e.g. To include current directory `.` and `asminc` for ASM only

```
ASM_INCDIRS = . asminc
```

SRCDIRS

This **SRCDIRS** is used to set the source directories used to search the C/CPP/ASM source code files, it will not do recursively.

e.g. To search C/CPP/ASM source files in directory `.` and `src`

```
SRCDIRS = . src
```

C_SRCDIRS

This **C_SRCDIRS** is used to set the source directories used to search the C only source code files(`*.c`, `*.C`), it will not do recursively.

e.g. To search C only source files in directory `.` and `csrc`

```
C_SRCDIRS = . csrc
```

CXX_SRCDIRS

This **CXX_SRCDIRS** is used to set the source directories used to search the CPP only source code files(`*.cpp`, `*.CPP`), it will not do recursively.

e.g. To search CPP only source files in directory `.` and `cppsrc`

```
CXX_SRCDIRS = . cppsrc
```

ASM_SRCDIRS

This **ASM_SRCDIRS** is used to set the source directories used to search the ASM only source code files(`*.s`, `*.S`), it will not do recursively.

e.g. To search ASM only source files in directory `.` and `asmsrc`

```
ASM_SRCDIRS = . asmsrc
```

C_SRCS

If you just want to include a few of C source files in directories, you can use this **C_SRCS** variable, makefile wildcard pattern supported.

e.g. To include `main.c` and `src/hello.c`

```
C_SRCS = main.c src/hello.c
```

CXX_SRCS

If you just want to include a few of CPP source files in directories, you can use this **CXX_SRCS** variable, makefile wildcard pattern supported.

e.g. To include `main.cpp` and `src/hello.cpp`

```
CXX_SRCS = main.cpp src/hello.cpp
```

ASM_SRCS

If you just want to include a few of ASM source files in directories, you can use this **ASM_SRCS** variable, makefile wildcard pattern supported.

e.g. To include `asm.s` and `src/test.s`

```
ASM_SRCS = asm.s src/test.s
```

EXCLUDE_SRCS

If you just want to exclude a few of c/asm/cpp source files in directories, you can use this **EXCLUDE_SRCS** variable, makefile wildcard pattern supported.

e.g. To exclude `test2.c` and `src/test3.c`

```
EXCLUDE_SRCS = test2.c src/test3.c
```

COMMON_FLAGS

This **COMMON_FLAGS** variable is used to define common compiler flags to all c/asm/cpp compiler.

For example, you can add a newline `COMMON_FLAGS += -O3 -funroll-loops -fpeel-loops` in your application Makefile and these options will be passed to C/ASM/CPP compiler.

This variable should be defined in Makefile.

CFLAGS

Different to **COMMON_FLAGS**, this **CFLAGS** variable is used to define common compiler flags to C compiler only. For example, you can add a newline `CFLAGS += -O3 -funroll-loops -fpeel-loops` in your application Makefile and these options will be passed to C compiler.

CXXFLAGS

Different to **COMMON_FLAGS**, this **CXXFLAGS** variable is used to define common compiler flags to cpp compiler only.

For example, you can add a newline `CXXFLAGS += -O3 -funroll-loops -fpeel-loops` in your application Makefile and these options will be passed to cpp compiler.

ASMFLAGS

Different to **COMMON_FLAGS**, this **ASMFLAGS** variable is used to define common compiler flags to asm compiler only.

For example, you can add a newline `ASMFLAGS += -O3 -funroll-loops -fpeel-loops` in your application Makefile and these options will be passed to asm compiler.

LDLFLAGS

This **LDLFLAGS** is used to pass extra linker flags, for example, if you want to use standard system libraries when linking, you can add a newline `LDLFLAGS += -nodefaultlibs` in you application Makefile.

If you want to link with other libraries, please use **LDLIBS** and **LIBDIRS** variables.

LDLIBS

This **LDLIBS** variable is library flags or names given to compilers when they are supposed to invoke the linker.

Non-library linker flags, such as `-L`, should go in the **LIBDIRS** or **LDLFLAGS** variable.

LIBDIRS

This **LIBDIRS** variable is used to store the library directories, which could be used together with **LDLIBS**.

For example, if you have a library located in `$(NUCLEI_SDK_ROOT)/Library/DSP/libnmsis_dsp_rv32imac.a`, and you want to link it, then you can define these lines:

```
LDLIBS = -lnmsis_dsp_rv32imac
LIBDIRS = $(NUCLEI_SDK_ROOT)/Library/DSP
```

LINKER_SCRIPT

This `LINKER_SCRIPT` variable could be used to set the link script of the application.

By default, there is no need to set this variable, since the build system will define a default linker script for application according to the build configuration. If you want to define your own linker script, you can set this variable.

For example, `LINKER_SCRIPT := gcc.ld`.

3.3 Application Development

3.3.1 Overview

Here will describe how to develop an Nuclei N100 SDK application.

To develop a Nuclei N100 SDK application from scratch, you can do the following steps:

1. Create a directory to place your application code.
2. Create **Makefile** in the new created directory, the minimal **Makefile** should look like this

```
1 TARGET = your_target_name
2
3 NUCLEI_SDK_ROOT = path/to/your_nuclei_n100_sdk_root
4
5 SRCDIRS = .
6
7 INC_DIRS = .
8
9 include $(NUCLEI_SDK_ROOT)/Build/Makefile.base
```

3. Copy or create your application code in new created directory.

Note:

- If you just want to SoC related resource, you can include header file `nuclei_sdk_soc.h` in your application code.
- If you just want to SoC and Board related resource, you can include header file `nuclei_sdk_hal.h` in your application code.
- For simplicity, we recommend you to use `nuclei_sdk_hal.h` header file

4. Follow *Build System based on Makefile* (page 17) to change your application Makefile.

3.3.2 Add Extra Source Code

If you want to add extra source code, you can use these makefile variables:

To add all the source code in directories, recursive search is not supported.

- *SRCDIRS* (page 39): Add C/CPP/ASM source code located in the directories defined by this variable.
- *C_SRCDIRS* (page 39): Add C only source code located in the directories defined by this variable.
- *CXX_SRCDIRS* (page 39): Add CPP only source code located in the directories defined by this variable.
- *ASM_SRCDIRS* (page 39): Add ASM only source code located in the directories defined by this variable.

To add only selected c/cxx/asm source files

- *C_SRCS* (page 40): Add C only source code files defined by this variable.
- *CXX_SRCS* (page 40): Add CPP only source code files defined by this variable.
- *ASM_SRCS* (page 40): Add ASM only source code files defined by this variable.

To exclude some source files

- *EXCLUDE_SRCS* (page 40): Exclude source files defined by this variable.

3.3.3 Add Extra Include Directory

If you want to add extra include directories, you can use these makefile variables:

- *INCDIRS* (page 38): Include the directories defined by this variable for C/ASM/CPP code during compiling.
- *C_INCDIRS* (page 38): Include the directories defined by this variable for C only code during compiling.
- *CXX_INCDIRS* (page 38): Include the directories defined by this variable for CPP only code during compiling.
- *ASM_INCDIRS* (page 39): Include the directories defined by this variable for ASM only code during compiling.

3.3.4 Add Extra Build Options

If you want to add extra build options, you can use these makefile variables:

- *COMMON_FLAGS* (page 40): This will add compiling flags for C/CPP/ASM source code.
- *CFLAGS* (page 41): This will add compiling flags for C source code.
- *CXXFLAGS* (page 41): This will add compiling flags for CPP source code.
- *ASMFLAGS* (page 41): This will add compiling flags for ASM source code.
- *LD_FLAGS* (page 41): This will add linker flags when linking.
- *LDLIBS* (page 41): This will add extra libraries need to be linked.
- *LIBDIRS* (page 41): This will add extra library directories to be searched by linker.

3.3.5 Optimize For Code Size

If you want to optimize your application for code size, you set `COMMON_FLAGS` in your application Makefile like this:

```
COMMON_FLAGS := -Os
```

If you want to optimize code size even more, you use this link time optimization(LTO) as below:

```
COMMON_FLAGS := -Os -flto
```

For more details about gcc optimization, please refer to [Options That Control Optimization in GCC²⁶](#).

3.3.6 Change Link Script

If you want to change the default link script defined by your make configuration(SOC, BOARD, DOWNLOAD). You can use `LINKER_SCRIPT` (page 42) variable to set your linker script.

The default linker script used for different boards can be found in [Board](#) (page 61).

3.3.7 Set Default Make Options

Set Default Global Make Options For Nuclei N100 SDK

If you want to change the global Make options for the Nuclei N100 SDK, you can add the [Makefile.global](#) (page 22).

Set Local Make Options For Your Application

If you want to change the application level Make options, you can add the [Makefile.local](#) (page 23).

3.4 Build Nuclei N100 SDK Documentation

In Nuclei N100 SDK, we use Sphinx and restructured text as documentation tool.

Here we only provide steps to build sphinx documentation in Linux environment.

3.4.1 Install Tools

To build this the documentation, you need to have these tools installed.

- Python3
- Python Pip tool

Then you can use the pip tool to install extra python packages required to build the documentation.

```
pip install -r doc/requirements.txt
```

²⁶ <https://gcc.gnu.org/onlinedocs/gcc-9.2.0/gcc/Optimize-Options.html#Optimize-Options>

3.4.2 Build The Documentation

Then you can build the documentation using the following command:

```
# cd to document folder
cd doc
# Build Sphinx documentation
make html
```

The documentation will be generated in *doc/build/html* folder.

You can open the *doc/build/html/index.html* in your browser to view the details.

CONTRIBUTING

Contributing to Nuclei N100 SDK project is always welcome.

You can always do a lot of things to help Nuclei N100 SDK project improve and grow stronger.

- *Port your Nuclei SoC into Nuclei N100 SDK* (page 47)
- *Submit your issue* (page 50)
- *Submit your pull request* (page 50)

4.1 Port your Nuclei SoC into Nuclei N100 SDK

Note: If you just want to do quick porting based on evalsoc implementation of Nuclei SDK to get quick ramp up, you can refer to [Quick Porting to you SoC based on Evalsoc in Nuclei N100 SDK](#)²⁷

If you want to port you Nuclei Processor Core based Board to Nuclei N100 SDK, you need to follow these steps:

And the best example is our evalsoc support, although it may contains many unused features you may not want to use, but it is our evaluation SoC and will supply to provide best support for Nuclei RISC-V CPU features, if you are already using it, please keep in update of the evalsoc support changes in each release, you can track it by diff each release changes, and please also remember Nuclei N100 SDK release may also bump with NMSIS release.

Assume your SoC name is `ncstar`, based on Nuclei core **n100**, and **RISCV_ARCH** is `rv32imc`, **RISCV_ABI** is `ilp32`, and you made a new board called `ncstar_eval`, and this SoC only support **FlashXIP** download mode.

Make sure the SoC name and Board name used in this Nuclei N100 SDK is all in lowercase.

1. Create a folder named `ncstar` under **SoC** directory.
 - Create folder named `Board` and `Common` under `ncstar`
 - Create directory structure under `ncstar/Common` like below:

```
<ncstar/Common>
├── Include
│   ├── peripheral_or_device_headers.h
│   ├── .....
│   ├── ncstar.h
│   ├── nuclei_sdk_soc.h
│   └── system_ncstar.h
└── Source
```

(continues on next page)

²⁷ https://doc.nucleisys.com/nuclei_studio_supply/28-quick_porting_from_evalsoc_to_customsoc_based_on_Nuclei_SDK/

(continued from previous page)

```

├── Drivers
│   ├── peripheral_or_device_sources.c
│   └── .....
├── GCC
│   ├── intexc_ncstar.S
│   └── startup_ncstar.S
├── Stubs
│   ├── newlib
│   └── libncrt
├── ncstar_soc.c
└── system_ncstar.c

```

Note:

- The directory structure is based on the NMSIS device template, please refer to https://doc.nucleisys.com/nmsis/core/core_templates.html
- The folder names must be exactly the same as the directory structure showed
- **peripheral_or_device_sources.c** means the SoC peripheral driver source code files, such as uart, gpio, i2c, spi driver sources, usually get from the SoC firmware library, it should be placed in **Drivers** folder.
- **peripheral_or_device_headers.h** means the SoC peripheral driver header files, such as uart, gpio, i2c, spi driver headers, usually get from the SoC firmware library, it should be placed in **Include** folder.
- The **Stubs** folder contains the stub code files for newlib c library and nuclei c runtime library porting code, take SoC/evalsoc/Common/Stubs as reference.
- The **GCC** folder contains *startup* and *exception/interrupt* assemble code, if your board share the same linker script files, you can also put link script files here, the linker script files name rules can refer to previously supported *evalsoc* SoC.
- If you want to do IAR compiler support, you also need to implement IAR related stuff, which is located in folder called IAR.
- The **nuclei_sdk_soc.h** file is very important, it is a Nuclei SoC Header file used by common application which can run access different SoC, it should include the SoC device header file `ncstar.h`

- Create directory structure under `ncstar/Board` like below:

```

<ncstar/Board>
├── ncstar_eval
│   ├── Include
│   │   ├── ncstar_eval.h
│   │   └── nuclei_sdk_hal.h
│   ├── openocd_ncstar.cfg
│   ├── Source
│   │   ├── GCC
│   │   │   └── gcc_ncstar_flashxip.ld
│   └── ncstar_eval.c

```

Note:

- The **ncstar_eval** is the board folder name, if you have a new board, you can create a new folder in the same level

- **Include** folder contains the board related header files
- **Source** folder contains the board related source files
- **GCC** folder is optional, if your linker script for the board is different to the SoC, you need to put your linker script here
- **openocd_ncstar.cfg** file is the board related openocd debug configuration file
- **ncstar_eval.h** file contains board related definition or APIs and also include the **SoC** header file, you can refer to previously supported board such as `nuclei_fpga_eval`
- **nuclei_sdk_hal.h** is very important, it includes the **ncstar_eval.h** header file. This file is used in application as entry header file to access board and SoC resources.

2. Create Makefile related to ncstar in *Nuclei N100 SDK build system* (page 17)

- Create **SoC/ncstar/build.mk**, the file content should be like this:

```
##### Put your SoC build configurations below #####

BOARD ?= ncstar_eval

# override DOWNLOAD and CORE variable for NCSTAR SoC
# even though it was set with a command argument
override CORE := n100
override DOWNLOAD := flashxip

NUCLEI_SDK_SOC_BOARD := $(NUCLEI_SDK_SOC)/Board/$(BOARD)
NUCLEI_SDK_SOC_COMMON := $(NUCLEI_SDK_SOC)/Common

#no ilm on NCSTAR SoC
LINKER_SCRIPT ?= $(NUCLEI_SDK_SOC_BOARD)/Source/GCC/gcc_ncstar_flashxip.ld
OPENOCD_CFG ?= $(NUCLEI_SDK_SOC_BOARD)/openocd_ncstar.cfg

RISCV_ARCH ?= rv32imc
RISCV_ABI ?= ilp32

##### Put your Source code Management configurations below #####

INCDIRS += $(NUCLEI_SDK_SOC_COMMON)/Include

C_SRCDIRS += $(NUCLEI_SDK_SOC_COMMON)/Source \
             $(NUCLEI_SDK_SOC_COMMON)/Source/Drivers

ifneq ($(findstring libncrt,$(STDCLIB)),)
C_SRCDIRS += $(NUCLEI_SDK_SOC_COMMON)/Source/Stubs/libncrt
else ifneq ($(findstring newlib,$(STDCLIB)),)
C_SRCDIRS += $(NUCLEI_SDK_SOC_COMMON)/Source/Stubs/newlib
else
# no stubs will be used
endif

ASM_SRCS += $(NUCLEI_SDK_SOC_COMMON)/Source/GCC/startup_ncstar.S \
            $(NUCLEI_SDK_SOC_COMMON)/Source/GCC/intexc_ncstar.S
```

(continues on next page)

(continued from previous page)

```
# Add extra board related source files and header files
VALID_NUCLEI_SDK_SOC_BOARD := $(wildcard $(NUCLEI_SDK_SOC_BOARD))
ifneq ($(VALID_NUCLEI_SDK_SOC_BOARD),)
INCDIRS += $(VALID_NUCLEI_SDK_SOC_BOARD)/Include
C_SRCDIRS += $(VALID_NUCLEI_SDK_SOC_BOARD)/Source
endif
```

3. If you have setup the source code and build system correctly, then you can test your SoC using the common applications, e.g.

```
# Test helloworld application for ncstar_eval board
## cd to helloworld application directory
cd application/baremetal/helloworld
## clean and build helloworld application for ncstar_eval board
make SOC=ncstar BOARD=ncstar_eval clean all
## connect your board to PC and install jtag driver, open UART terminal
## set baudrate to 115200bps and then upload the built application
## to the ncstar_eval board using openocd, and you can check the
## run message in UART terminal
make SOC=ncstar BOARD=ncstar_eval upload
```

Note:

- You can always refer to previously supported SoCs for reference, such as the evalsoc SoC, we suggest you follow the evalsoc implementation, since it is well maintained to support latest nuclei riscv cpu feature.
- The evalsoc SoC is a FPGA based evaluation platform, it have ilm and dlm, so it support many *download modes* (page 26)
- The **nuclei_sdk_soc.h** must be created in SoC include directory, it must include the device header file <device>.h and SoC firmware library header files.
- The **nuclei_sdk_hal.h** must be created in Board include directory, it must include **nuclei_sdk_soc.h** and board related header files.

4.2 Submit your issue

If you find any issue related to Nuclei N100 SDK project, you can open an issue in <https://github.com/Nuclei-Software/nuclei-sdk/issues>

4.3 Submit your pull request

If you want to contribute your code to Nuclei N100 SDK project, you can open an pull request in <https://github.com/Nuclei-Software/nuclei-sdk/pulls>

Regarding to code style, please refer to *Code Style* (page 17).

4.4 Git commit guide

If you want to contribute your code, make sure you follow the guidance of git commit, see here <https://chris.beams.io/posts/git-commit/> for details

- Use the present tense (“Add feature” not “Added feature”)
- Use the imperative mood (“Move cursor to...” not “Moves cursor to...”)
- Limit the first line to 80 characters or less
- Refer github issues and pull requests liberally using #
- Write the commit message with an category name and colon:
 - soc: changes related to soc
 - board: changes related to board support packages
 - nmsis: changes related to NMSIS
 - build: changes related to build system
 - library: changes related to libraries
 - rtos: changes related to rtoses
 - test: changes related to test cases
 - doc: changes related to documentation
 - ci: changes related to ci environment
 - application: changes related to applications
 - misc: changes not categorized
 - env: changes related to environment

DESIGN AND ARCHITECTURE

5.1 Overview

Nuclei N100 SDK is developed based on **NMSIS**, all the SoCs supported in it are following the [NMSIS-Core Device Templates Guidance](#)²⁸.

So this Nuclei N100 SDK can be treated as a software guide for how to use NMSIS.

The build system we use in Nuclei N100 SDK is `Makefile`, it support both Windows and Linux, and when we develop Nuclei N100 SDK build system, we keep it simple, so it make developer can easily port this Nuclei N100 SDK software code to other IDEs.

Click [Overview](#) (page 1) to learn more about the Nuclei N100 SDK project overview.

For example, we have ported Nuclei N100 SDK to use Segger embedded Studio, IAR Workbench and PlatformIO.

5.1.1 Directory Structure

To learn deeper about Nuclei N100 SDK project, the directory structure is a good start point.

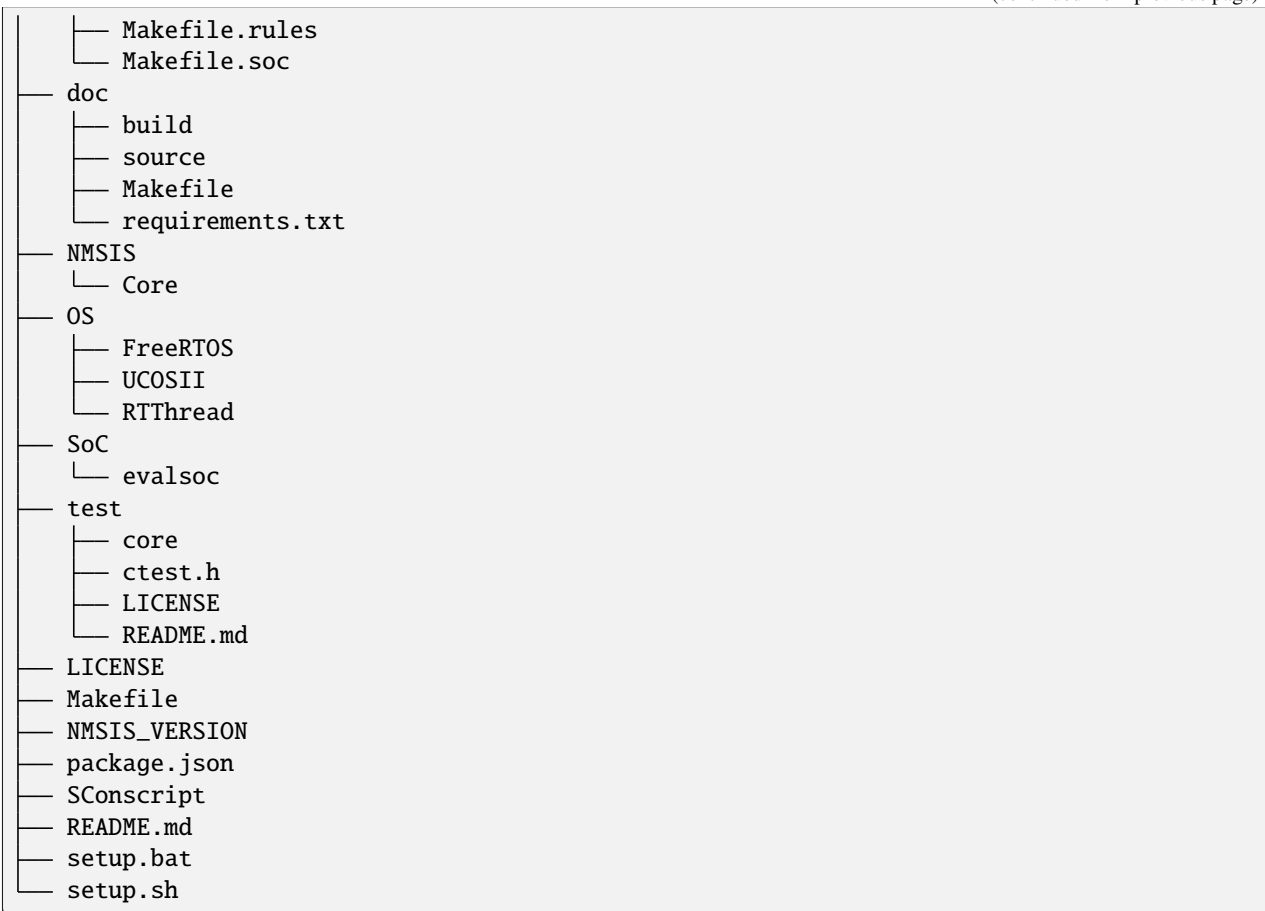
Below, we will describe our design about the Nuclei N100 SDK directory structure:

Here is the directory structure for this Nuclei N100 SDK.

```
$NUCLEI_SDK_ROOT
├── application
│   ├── baremetal
│   ├── freertos
│   ├── ucosii
│   └── rtthread
├── Build
│   ├── gmsl
│   ├── toolchain
│   ├── Makefile.base
│   ├── Makefile.conf
│   ├── Makefile.core
│   ├── Makefile.components
│   ├── Makefile.files
│   ├── Makefile.global
│   ├── Makefile.misc
│   └── Makefile.rtos
```

(continues on next page)

²⁸ https://doc.nucleisys.com/nmsis/core/core_templates.html



- **application**

This directory contains all the application softwares for this Nuclei N100 SDK.

The application code can be divided into mainly 4 parts, which are:

- **Baremetal** applications, which will provide baremetal applications without any OS usage, these applications will be placed in *application/baremetal/* folder.
- **FreeRTOS** applications, which will provide FreeRTOS applications using FreeRTOS RTOS, placed in *application/freertos/* folder.
- **UCOSII** applications, which will provide UCOSII applications using UCOSII RTOS, placed in *application/ucosii/* folder.
- **RTThread** applications, which will provide RT-Thread applications using RT-Thread RTOS, placed in *application/rthread/* folder.

- **SoC**

This directory contains all the supported SoCs for this Nuclei N100 SDK, the directory name for SoC and its boards should always in lower case.

Here we mainly support Nuclei processor cores running in Nuclei FPGA evaluation board, the support package placed in *SoC/evalsoc/*.

In each SoC's include directory, *nuclei_sdk_soc.h* must be provided, and include the soc header file, for example, *SoC/evalsoc/Common/Include/nuclei_sdk_soc.h*.

In each SoC Board's include directory, *nuclei_sdk_hal.h* must be provided, and include the board header file, for example, *SoC/evalsoc/Board/nuclei_fpga_eval/Include/nuclei_sdk_hal.h*.

- **Build**

This directory contains the key part of the build system based on Makefile for Nuclei N100 SDK.

- **NMSIS**

This directory contains the NMSIS header files, which is widely used in this Nuclei N100 SDK, you can check the *NMSIS_VERSION* file to know the current *NMSIS* version used in **Nuclei-SDK**.

We will also sync the changes in [NMSIS project](#)²⁹ when it provided a new release.

- **OS**

This directory provided three RTOS package we supported which are **FreeRTOS**, **UCOSII** and **RT-Thread**.

- **LICENSE**

Nuclei N100 SDK license file.

- **NMSIS_VERSION**

NMSIS Version file. It will show current NMSIS version used in Nuclei N100 SDK.

- **package.json**

PlatformIO package json file for Nuclei N100 SDK, used in [Nuclei Platform for PlatformIO](#)³⁰.

- **SConscript**

RT-Thread package scon build script, used in [RT-Thread package development](#)³¹.

- **Makefile**

An external Makefile just for build, run, debug application without cd to any corresponding application directory, such as *application/baremetal/helloworld/*.

- **setup.sh**

Nuclei N100 SDK environment setup script for **Linux**. You need to create your own *setup_config.sh*.

```
# you can export this variable to Nuclei Studio's toolchain folder
NUCLEI_TOOL_ROOT=/path/to/your_tool_root
```

In the **\$NUCLEI_TOOL_ROOT** for **Linux**, you need to have Nuclei RISC-V GNU GCC toolchain and OpenOCD installed as below.

```
$NUCLEI_TOOL_ROOT
├── gcc
│   ├── bin
│   ├── include
│   ├── lib
│   ├── libexec
│   ├── riscv64-unknown-elf
│   └── share
├── openocd
│   ├── bin
│   └── contrib
```

(continues on next page)

²⁹ <https://github.com/Nuclei-Software/NMSIS>

³⁰ <https://platformio.org/platforms/nuclei/>

³¹ <https://www.rt-thread.org/document/site/development-guide/package/package/>

(continued from previous page)

```

├── distro-info
├── OpenULINK
├── scripts
└── share

```

- **setup.bat**

Nuclei N100 SDK environment setup bat script for **Windows**. You need to create your own *setup_config.bat*.

```
set NUCLEI_TOOL_ROOT=\path\to\your_tool_root
```

In the `%NUCLEI_TOOL_ROOT%` for **Windows**, you need to have Nuclei RISC-V GNU GCC toolchain, necessary Windows build tools and OpenOCD installed as below.

```

%NUCLEI_TOOL_ROOT%
├── build-tools
│   ├── bin
│   ├── gnu-mcu-eclipse
│   └── licenses
├── gcc
│   ├── bin
│   ├── include
│   ├── lib
│   ├── libexec
│   ├── riscv64-unknown-elf
│   └── share
└── openocd
    ├── bin
    ├── contrib
    ├── distro-info
    ├── OpenULINK
    ├── scripts
    └── share

```

5.1.2 Project Components

This Nuclei N100 SDK project components is list as below:

- *Nuclei Processor* (page 57): How Nuclei Processor Core is used in Nuclei N100 SDK
- *SoC* (page 59): How Nuclei processor code based SoC device is supported in Nuclei N100 SDK
- *Board* (page 61): How Nuclei based SoC's Board is supported in Nuclei N100 SDK
- *Peripheral* (page 64): How to use the peripheral driver in Nuclei N100 SDK
- *RTOS* (page 65): What RTOSes are supported in Nuclei N100 SDK
- *Application* (page 67): How to use pre-built applications in Nuclei N100 SDK

5.2 Nuclei Processor

Nuclei processor core are following and compatible to RISC-V standard architecture, but there might be some additions and enhancements to the original standard spec.

Click [Nuclei Spec](#)³² to learn more about Nuclei RISC-V Instruction Set Architecture.

5.2.1 Introduction

Nuclei provides the following [RISC-V IP Products](#)³³ for AIoT:

- **N100 series:** Designed for mixed digital and analog, IoT or other extremely low-power and small area scenarios, which is the perfect replacement of traditional 8051 cores, it has a separated databook, please contact with our AE to get it.
- **N200 series:** Designed for ultra-low power consumption and embedded scenarios, perfectly replaces the arm Cortex-M series cores.
- **N300 series:** Designed for extreme energy efficiency ratio, requiring DSP and FPU features, as IoT and industrial control scenarios.
- **600/900/1000 series:** Fully support Linux for high-performance edge computing and smart AIoT.

Note:

- **N100 series** is supported by **modified NMSIS** and **Nuclei N100 SDK**
 - **200/300/600/900/1000 series** are supported by **NMSIS** and **Nuclei SDK**
-

5.2.2 NMSIS in Nuclei N100 SDK

This Nuclei N100 SDK is built based on the modified [NMSIS](#)³⁴ framework, user can access modified NMSIS Core API

These modified NMSIS APIs are mainly responsible for accessing Nuclei RISC-V 100 series Processor Core.

Note:

- To support RT-Thread in Nuclei-SDK, we have to modify the **startup_<device>.S**, to use macro `RTOS_RTTHREAD` defined when using RT-Thread as below:

```
#ifndef RTOS_RTTHREAD
    // Call entry function when using RT-Thread
    call entry
#else
    call main
#endif
```

- In order to support RT-Thread initialization macros `INIT_XXX_EXPORT`, we also need to modify the link script files, add lines after ``(.rodata .rodata.)`` as below:

³² https://doc.nucleisys.com/nuclei_spec/

³³ <https://nucleisys.com/product.php>

³⁴ <https://github.com/Nuclei-Software/NMSIS>

```
. = ALIGN(4);
*(.rdata)
*(.rodata .rodata.*)
/* RT-Thread added lines begin */
/* section information for initial. */
. = ALIGN(4);
__rt_init_start = .;
KEEP(*(SORT(.rti_fn*)))
__rt_init_end = .;
/* section information for finsh shell */
. = ALIGN(4);
__fsymtab_start = .;
KEEP(*(FSymTab))
__fsymtab_end = .;
. = ALIGN(4);
__vsymtab_start = .;
KEEP(*(VSymTab))
__vsymtab_end = .;
/* RT-Thread added lines end */
*(.gnu.linkonce.r.*)
```

5.2.3 SoC Resource

Regarding the SoC Resource exclude the Nuclei RISC-V Processor Core, it mainly consists of different peripherals such UART, GPIO, I2C, SPI, CAN, PWM, DMA, USB and etc.

The APIs to access to the SoC resources are usually defined by the SoC Firmware Library Package provided by SoC Vendor.

In Nuclei N100 SDK, currently we just required developer to provide the following common resources:

- A UART used to implement several stub functions for `printf` function
 - When using `newlib` library, please check stub functions list in `SoC/evalsoc/Common/Stubs/newlib`
 - When using `libnrcrt` library, please check stub functions list in `SoC/evalsoc/Common/Stubs/libnrcrt`
 - When using `iar dlib` library, please check stub functions list in `SoC/evalsoc/Common/Stubs/iardlib`
- Common initialization code defined in **System_<Device>.c/h** in each SoC support package in Nuclei N100 SDK.
- Before enter to main function, these resources must be initialized:
 - The UART used to print must be initialized as 115200 bps, 8bit data, none parity check, 1 stop bit
 - Global interrupt is disabled

Note:

- If you want to learn more about SoC, please click [SoC](#) (page 59)
 - If you want to learn more about Board, please click [Board](#) (page 61)
 - If you want to learn more about Peripheral, please click [Peripheral](#) (page 64)
-

5.3 SoC

5.3.1 Nuclei Eval SoC

Note: Nuclei CPU IP now with iregion feature will use totally new evaluation SoC, this SoC is different from previous Demo SoC, please take care.

Nuclei DemoSoC is now removed in 0.5.0 release, and please use evalsoc now.

Nuclei Eval SoC is an evaluation FPGA SoC from Nuclei for customer to evaluate Nuclei RISC-V Process Core, and it is a successor for Demo SoC.

Overview

To easy user to evaluate Nuclei Processor Core, the prototype SoC (called Nuclei Eval SoC) is provided for evaluation purpose.

This prototype SoC includes:

- Processor Core, it can be Nuclei N class, NX class or UX class Processor Core.
- On-Chip SRAMs for instruction and data.
- The SoC buses.
- The basic peripherals, such as UART, SPI etc.

With this prototype SoC, user can run simulations, map it into the FPGA board, and run with real embedded application examples.

If you want to learn more about this evaluation SoC, please get the <Nuclei_Eval_SoC_Intro.pdf> from [Nuclei³⁵](#).

Supported Boards

In Nuclei N100 SDK, we support the following boards based on **Nuclei evalsoc** SoC, see:

- *Nuclei FPGA Evaluation Kit* (page 61), default Board when this SoC selected.

Usage

Note: In latest N100 CPU RTL generation flow, it will also generate an Nuclei N100 SDK to match CPU and EvalSoC RTL configuration, please use the generated Nuclei N100 SDK to evaluate your CPU and EvalSoC feature.

The generated Nuclei SDK by **nuclei_gen** will do the following tasks:

- Generate SoC/evalsoc/cpufeature.mk: which will define **CORE**, **ARCH_EXT**, **QEMU_SOCCFG** or **SIMULATION** default value.
- Generate SoC/evalsoc/Common/Include/cpufeature.h: which will define current cpu feature macros.
- Generate SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/evalsoc.memory: which will define the sram/flash base address and size and mvtv and mtvec address.

³⁵ <https://nucleisys.com/>

- Modify SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg: Mainly change workmem_base/workmem_size/flashxip_base/xipnuspi_base to adapt the evalsoc configuration.

If you want to use the generated Nuclei SDK by **nuclei_gen** In Nuclei Studio IDE, you need to zip it first, and then **import** it using RV-Tools -> NPK Package Management in Nuclei Studio IDE's menu, and when creating a IDE project using New Nuclei RISC-V C/C++ Project, please select the correct sdk and version which you can check it in the <SDK>/npk.yml file, and in the project example configuration wizard window, you should click the **SDK gen by nuclei_gen**, and configure the **Nuclei RISC-V Core** and **ARCH Extensions**, **Nuclei Cache Extensions** according to your configured CPU ISA, and CPU feature defined in generated cpufeature.h. Currently you still need to modify IAR linker script by yourself, it is not automatically modified.

In latest evalsoc for n100, the interrupt vector table + reset_vector + exception_entry will be placed tightly in the top of an RO/RW memory.

It will look like this as below:

```

Disassembly of section .init:          -> top of RO/RW memory

a0000000 <vector_base>:                -> vector table for interrupt, which is the MTVT
    ...                               eg. for this case, there are 30 external
↳ interrupts(CFG_IRQ_NUM),            totally 32 interrupts

a0000080 <_reset_vector>:              -> reset vector right following the vector_table
↳ array, size may variable according to the external interrupt number count

a0000080:  0080006f          j      a0000088 <_start>          -> the reset vector
↳ code, just jump to real startup code

a0000084:  0bc0006f          j      a0000140 <exc_entry>      -> exception
↳ entry(MTVEC), just jump to real exception handling code

a0000088 <_start>:
a0000088:  30047073          csrc   mstatus,8

**NOTE**: Since evalsoc implementation is just a reference, you can customized your
↳ vector table, reset vector, exception entry as you want,
but you **MUST** modify the startup code and linker script code to match your real CPU
↳ design.
    
```

If you want to use this **Nuclei evalsoc SoC** in Nuclei N100 SDK, you need to set the *SOC* (page 24) Makefile variable to evalsoc.

```

# Choose SoC to be evalsoc
# the following command will build application
# using default evalsoc SoC based board
# defined in Build System and application Makefile
make SOC=evalsoc all
    
```

5.4 Board

5.4.1 Nuclei FPGA Evaluation Kit

Overview

Nuclei have customized different FPGA evaluation boards (called Nuclei FPGA Evaluation Kit), which can be programmed with Nuclei Demo/Eval SoC FPGA bitstream.

- **Nuclei FPGA Evaluation Kit, 100T version**

This **100T** version is a very early version which widely used since 2019, it has a Xilinx XC7A100T FPGA chip on the board.

- a: FPGA_RESET
- b: FPGA_PROG
- c: MCU_WKUP
- d: MCU_RESET
- e1: User button 1
- e2: User button 2
- e3: User button header
- Y1: GCLK
- Y2: RTC_CLK
- I: MCU_FLASH
- 2: FPGA_FLASH
- 3: FPGA_JTAG
- 4: MCU_JTAG
- 5: Power switch

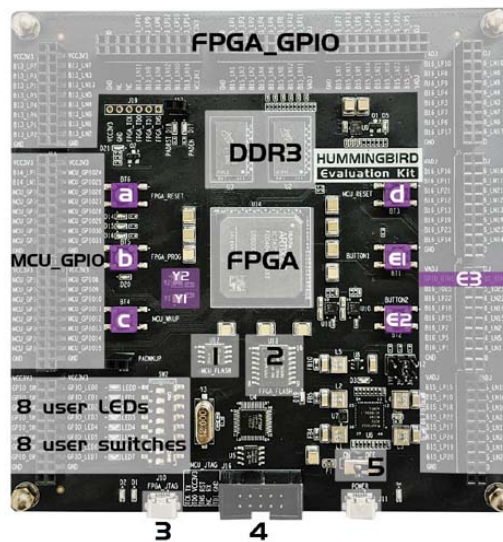


Fig. 1: Nuclei FPGA Evaluation Kit, 100T Version

- **Nuclei FPGA Evaluation Kit, DDR 200T version**

This **DDR 200T** version is a latest version which provided since 2020.09, it has a Xilinx XC7A200T FPGA chip on the board, and the onboard DDR could be connected to Nuclei RISC-V Core.

This board is a choice to replace the *100T version*, and it could be use to evaluate any Nuclei RISC-V core.

We also use this version of board to evaluate Nuclei UX class core which can run Linux on it, if you want to run Linux on this board, please refer to [Nuclei Linux SDK](https://github.com/Nuclei-Software/nuclei-linux-sdk)³⁶.

- **Nuclei FPGA Evaluation Kit, MCU 200T version**

This **MCU 200T** version is a latest version which provided since 2020.09, it has a Xilinx XC7A200T FPGA chip on the board, but there is no DDR chip on the board.

³⁶ <https://github.com/Nuclei-Software/nuclei-linux-sdk>

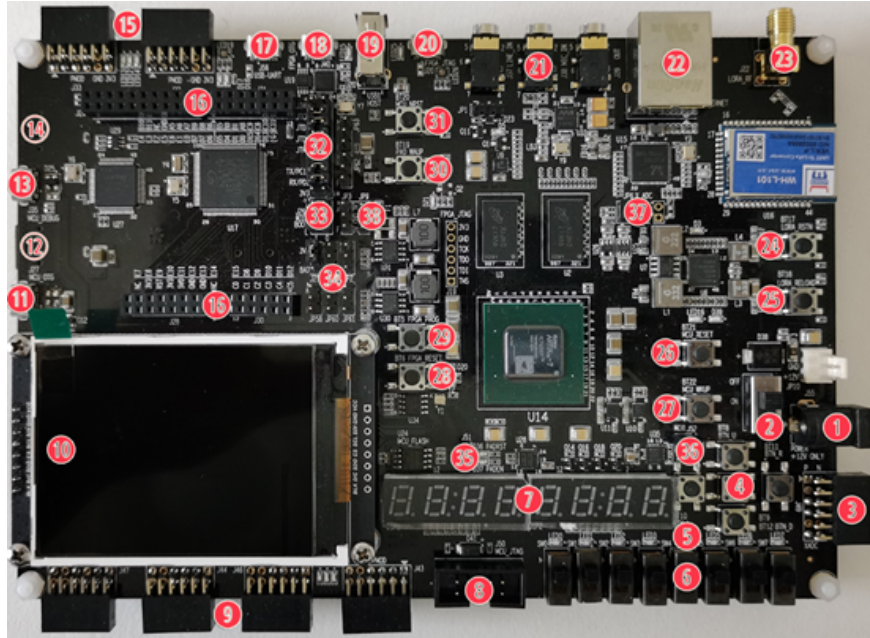


Fig. 2: Nuclei FPGA Evaluation Kit, DDR 200T Version

This board is a choice to replace the *100T version*, and it could be use to evaluate any Nuclei RISC-V core with don't use DDR.

There are also other fpga board we supported, such as KU060 and VCU118 board, please contact with our sales for details.

Click [Nuclei FPGA Evaluation Kit Board Documents](#)³⁷ to access the documents of these boards.

Setup

Follow the guide in [Nuclei FPGA Evaluation Kit Board Documents](#)³⁸ to setup the board, make sure the following items are set correctly:

- Use **Nuclei FPGA debugger** to connect the **MCU-JTAG** on board to your PC in order to download and debug programs and monitor the UART message.
- Power on the board using USB doggle(for 100T) or DC 12V Power(for MCU 200T or DDR 200T).
- The Nuclei FPGA SoC FPGA bitstream with Nuclei RISC-V evaluation core inside is programmed to FPGA on this board.
- Following steps in [debugger kit manual](#)³⁹ to setup JTAG drivers for your development environment

³⁷ <https://nucleisys.com/developboard.php>

³⁸ <https://nucleisys.com/developboard.php>

³⁹ https://www.nucleisys.com/theme/package/Nuclei_FPGA_DebugKit_Intro.pdf

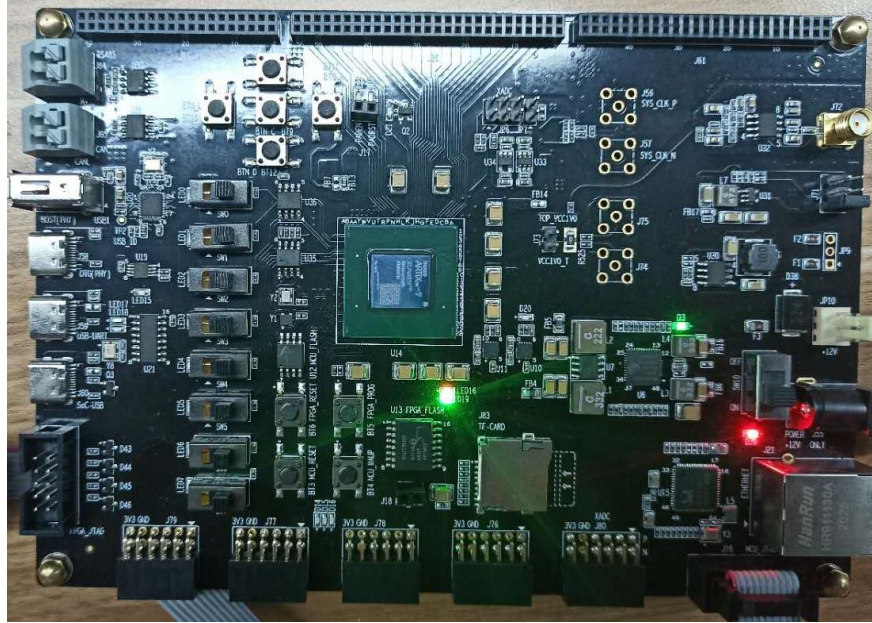


Fig. 3: Nuclei FPGA Evaluation Kit, MCU 200T Version

How to use

For Nuclei FPGA Evaluation board:

- evalsoc can run on this fpga board, please choose the correct SoC.
- **DOWNLOAD** support all the modes list in [DOWNLOAD](#) (page 26)
 - You can find default used linker scripts for different download modes in SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/.
 - * evalsoc.memory: Memory information for evalsoc
 - * gcc_evalsoc_sram.ld: Linker script file for DOWNLOAD=sram
 - If you want to specify your own modified linker script, you can follow steps described in [Change Link Script](#) (page 44)
 - If you want to change the base address or size of SRAM of linker script file, you can adapt the evalsoc.memory file.
- **CORE** support all the cores list in [CORE](#) (page 27)
- Its openocd configuration file can be found in SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg

To run this application in Nuclei FPGA Evaluation board in Nuclei N100 SDK, you just need to use this **SOC** and **BOARD** variables.

```
### For evalsoc
# Clean the application with DOWNLOAD=sram CORE=n100
make SOC=evalsoc BOARD=nuclei_fpga_eval DOWNLOAD=sram CORE=n100 clean
# Build the application with DOWNLOAD=sram CORE=n100
make SOC=evalsoc BOARD=nuclei_fpga_eval DOWNLOAD=sram CORE=n100 all
# Upload the application using openocd and gdb with DOWNLOAD=sram CORE=n100
```

(continues on next page)

(continued from previous page)

```
make SOC=evalsoc BOARD=nuclei_fpga_eval DOWNLOAD=sram CORE=n100 upload
# Debug the application using openocd and gdb with DOWNLOAD=sram CORE=n100
make SOC=evalsoc BOARD=nuclei_fpga_eval DOWNLOAD=sram CORE=n100 debug
### For evalsoc
# Clean the application with DOWNLOAD=sram CORE=n100
make SOC=evalsoc BOARD=nuclei_fpga_eval DOWNLOAD=sram CORE=n100 clean
# Upload the application using openocd and gdb with DOWNLOAD=sram CORE=n100
make SOC=evalsoc BOARD=nuclei_fpga_eval DOWNLOAD=sram CORE=n100 upload
```

If you want to try other toolchain, such as nuclei llvm or terapines zcc, you can pass an extra *TOOLCHAIN* (page 26) make variable.

Note:

- You can change the value passed to **CORE** according to the Nuclei Demo SoC Evaluation Core the Nuclei FPGA SoC you have.
- You can also change the value passed to **DOWNLOAD** to run program in different modes.
- The FreeRTOS and UCOSII demos maybe not working in when run in xipflash in Nuclei FPGA board due to program running in Flash is really too slow. If you want to try these demos, please use sram download mode.

5.5 Peripheral

5.5.1 Overview

Regarding the peripheral support(such as UART, GPIO, SPI, I2C and etc.) in Nuclei N100 SDK, we didn't define a device or peripheral layer for different SoCs, so the peripheral drivers are directly tighted with each SoC, and if developer want to use the drivers, they can directly use the driver API defined in each SoC.

Considering this peripheral driver difference in each SoC, if you want to write portable code in Nuclei N100 SDK, you can use include the `nuclei_sdk_soc.h`, then you can write application which only use the resources of Nuclei Core.

If you want to use all the board resources, you can include the `nuclei_sdk_hal.h`, then you can write application for your own board, but the application can only run in the board you provided.

5.5.2 Usage

If you want to learn about what peripheral driver you can use, you can check the `nuclei_sdk_soc.h` of each SoC, and `nuclei_sdk_hal.h` of each board.

For SoC firmware library APIs:

- You can find the **Nuclei Eval SoC firmware library APIs** in `SoC/evalsoc/Common/Include`

If you just want to use SoC firmware library API, you just need to include `nuclei_sdk_soc.h`, then you can use the all the APIs in that SoC include directory.

For Board related APIs:

- You can find the **Nuclei FPGA Evaluation Board related APIs** in `SoC/evalsoc/Board/nuclei_fpga_eval/Include`

If you just want to use all the APIs of Board and SoC, you just need to include `nuclei_sdk_hal.h`, then you can use the all the APIs in that Board and SoC include directory.

5.6 RTOS

5.6.1 Overview

In Nuclei N100 SDK, we have support three most-used RTOSes in the world, **FreeRTOS**, **UCOSII** and **RT-Thread** from China.

If you want to use RTOS in your application, you can choose one of the supported RTOSes.

Note: When you want to develop RTOS application in Nuclei N100 SDK, please don't reconfigure `SysTimer` and `SysTimer Software Interrupt`, since it is already used by RTOS portable code.

5.6.2 FreeRTOS

FreeRTOS⁴⁰ is a market-leading real-time operating system (RTOS) for microcontrollers and small microprocessors.

In our FreeRTOS portable code, we are using `SysTimer Interrupt` as RTOS `SysTick Interrupt`, and using `SysTimer Software Interrupt` to do task switch.

These two interrupts are kept as lowest level, and `SysTimer Interrupt` is initialized as non-vector interrupt, and `SysTimer Software Interrupt` is initialized as vector interrupt and interrupt handler implemented using asm code.

If you want to learn about how to use FreeRTOS APIs, you need to go to its website to learn the FreeRTOS documentation in its website.

In Nuclei N100 SDK, if you want to use **FreeRTOS** in your application, you need to add `RTOS = FreeRTOS` in your application Makefile.

And in your application code, you need to do the following things:

- Add FreeRTOS configuration file -> `FreeRTOSConfig.h`
- Include FreeRTOS header files

Note:

- You can check the `application\freertos\demo` for reference
 - Current version of FreeRTOS used in Nuclei N100 SDK is `V10.3.1`
 - If you want to change the OS ticks per seconds, you can change the `configTICK_RATE_HZ` defined in `FreeRTOSConfig.h`
-

More information about FreeRTOS get started, please click <https://www.freertos.org/FreeRTOS-quick-start-guide.html>

⁴⁰ <https://www.freertos.org/>

5.6.3 UCOSII

UCOSII⁴¹ a priority-based preemptive real-time kernel for microprocessors, written mostly in the programming language C. It is intended for use in embedded systems.

In our UCOSII portable code, we are using `SysTimer Interrupt` as RTOS `SysTick Interrupt`, and using `SysTimer Software Interrupt` to do task switch.

If you want to learn about UCOSII, please click <https://www.micrium.com/books/ucosii/>

We are using the opensource version of UC-OS2 source code from <https://github.com/SiliconLabs/uC-OS2>, with optimized code for Nuclei RISC-V processors.

In Nuclei N100 SDK, if you want to use **UCOSII** in your application, you need to add `RTOS = UCOSII` in your application Makefile.

And in your application code, you need to do the following things:

- Add UCOSII application configuration header file -> `app_cfg.h` and `os_cfg.h`
- Add application hook source file -> `app_hooks.c`
- Include UCOSII header files

Note:

- You can check the `application\ucosii\demo` for reference
 - The UCOS-II application configuration template files can also be found in <https://github.com/SiliconLabs/uC-OS2/tree/master/Cfg/Template>
 - Current version of UCOSII used in Nuclei N100 SDK is `V2.93.00`
 - If you want to change the OS ticks per seconds, you can change the `OS_TICKS_PER_SEC` defined in `os_cfg.h`
-

Warning:

- For Nuclei N100 SDK release > v0.2.2, the UCOSII source code is replaced using the version from <https://github.com/SiliconLabs/uC-OS2/>, and application development for UCOSII is also changed, the `app_cfg.h`, `os_cfg.h` and `app_hooks.c` files are required in application source code.

5.6.4 RT-Thread

RT-Thread (page 66) RT-Thread was born in 2006, it is an open source, neutral, and community-based real-time operating system (RTOS).

RT-Thread is mainly written in C language, easy to understand and easy to port (can be quickly port to a wide range of mainstream MCUs and module chips).

It applies object-oriented programming methods to real-time system design, making the code elegant, structured, modular, and very tailorable.

In our support for RT-Thread, we get the source code of RT-Thread from a project called *RT-Thread Nano*⁴², which only provide kernel code of RT-Thread, which is easy to be integrated with Nuclei N100 SDK.

⁴¹ <https://www.micrium.com/>

⁴² <https://github.com/RT-Thread/rthread-nano>

In our RT-Thread portable code, we are using `SysTimer Interrupt` as RTOS `SysTick Interrupt`, and using `SysTimer Software Interrupt` to do task switch.

And also the `rt_hw_board_init` function is implemented in our portable code.

If you want to learn about RT-Thread, please click:

- For Chinese version, click <https://www.rt-thread.org/document/site/>
- For English version, click <https://github.com/RT-Thread/rt-thread#documentation>

In Nuclei N100 SDK, if you want to use **RT-Thread** in your application, you need to add `RTOS = RTThread` in your application Makefile.

And in your application code, you need to do the following things:

- Add RT-Thread application configuration header file -> `rtconfig.h`
- Include RT-Thread header files
- If you want to enable RT-Thread MSH feature, just add `RTTHREAD_MSH := 1` in your application Makefile.

Note:

- In RT-Thread, the `main` function is created as a RT-Thread thread, so you don't need to do any OS initialization work, it is done before `main`
 - We also provide good support directly through RT-Thread official repo, you can check Nuclei processor support for RT-Thread in [RT-Thread BSP For Nuclei⁴³](#).
-

5.7 Application

5.7.1 Overview

In Nuclei N100 SDK, we just provided applications which can run in different boards without any changes in code to demonstrate the baremetal service, freertos service and ucousii service features.

The provided applications can be divided into three categories:

- Bare-metal applications: Located in `application/baremetal`
- FreeRTOS applications: Located in `application/freertos`
- UCOSII applications: Located in `application/ucousii`
- RTThread applications: Located in `application/rthread`

If you want to develop your own application in Nuclei N100 SDK, please click [Application Development](#) (page 42) to learn more about it.

The following applications are running Nuclei Eval SoC.

Note:

- Only benchmark/helloworld can run on Nuclei Qemu >= 2025.02 now.
- Most of the application demonstrated below using `SOC=evalsoc`, you can easily change it to other SoC such as `evalsoc` by change it to `SOC=evalsoc`

⁴³ <https://github.com/RT-Thread/rt-thread/tree/master/bsp/nuclei/>

- Some applications may not be able to be run on your SoC using Nuclei CPU due to lack of cpu feature required to run on it.
 - Almost all the applications required Nuclei CPU configured with irqc and System Timer hardware feature.
 - Almost all the application required UART to print message, so you need to implement an UART drivers and clib stub functions, if you use *SEMIHOST* (page 28) to print message, it is not required.
-

5.7.2 Bare-metal applications

helloworld

This `helloworld` application⁴⁴ is used to print hello world, and also will check this RISC-V CSR `MISA` register value.

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the helloworld directory
cd application/baremetal/helloworld
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:19:03
Download Mode: ILM
CPU Frequency 16001597 Hz
CPU HartID: 0
Hart 0, MISA: 0x40001104
MISA: RV32IMC
Got rand integer 455647 using seed 1933213352.
0: Hello World From Nuclei RISC-V Processor!
1: Hello World From Nuclei RISC-V Processor!
2: Hello World From Nuclei RISC-V Processor!
3: Hello World From Nuclei RISC-V Processor!
4: Hello World From Nuclei RISC-V Processor!
5: Hello World From Nuclei RISC-V Processor!
6: Hello World From Nuclei RISC-V Processor!
7: Hello World From Nuclei RISC-V Processor!
8: Hello World From Nuclei RISC-V Processor!
9: Hello World From Nuclei RISC-V Processor!
10: Hello World From Nuclei RISC-V Processor!
11: Hello World From Nuclei RISC-V Processor!
12: Hello World From Nuclei RISC-V Processor!
13: Hello World From Nuclei RISC-V Processor!
14: Hello World From Nuclei RISC-V Processor!
15: Hello World From Nuclei RISC-V Processor!
16: Hello World From Nuclei RISC-V Processor!
17: Hello World From Nuclei RISC-V Processor!
```

(continues on next page)

⁴⁴ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/helloworld

(continued from previous page)

```
18: Hello World From Nuclei RISC-V Processor!  
19: Hello World From Nuclei RISC-V Processor!
```

demo_timer

This `demo_timer` application⁴⁵ is used to demonstrate how to use the CORE TIMER API including the Timer Interrupt and Timer Software Interrupt.

- First the timer interrupt will run for 5 times
- Then the software timer interrupt will start to run for 5 times

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment  
# cd to the demo_timer directory  
cd application/baremetal/demo_timer  
# Clean the application first  
make SOC=evalsoc clean  
# Build and upload the application  
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:20:01  
Download Mode: ILM  
CPU Frequency 16000942 Hz  
CPU HartID: 0  
init timer and start  
MTimer IRQ handler 1  
MTimer IRQ handler 2  
MTimer IRQ handler 3  
MTimer IRQ handler 4  
MTimer IRQ handler 5  
MTimer SW IRQ handler 1  
MTimer SW IRQ handler 2  
MTimer SW IRQ handler 3  
MTimer SW IRQ handler 4  
MTimer SW IRQ handler 5  
MTimer msip and mtip interrupt test finish and pass
```

demo_irqc

This `demo_irqc` application⁴⁶ is used to demonstrate how to use the irqc API and Interrupt.

- The timer interrupt and timer software interrupt are used

How to run this application:

⁴⁵ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/demo_timer

⁴⁶ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/demo_irqc

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the demo_irqc directory
cd application/baremetal/demo_irqc
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:21:42
Download Mode: ILM
CPU Frequency 16000942 Hz
CPU HartID: 0
Initialize timer and start timer interrupt periodically
-----
[IN TIMER INTERRUPT]timer interrupt hit 0 times
[IN TIMER INTERRUPT]trigger software interrupt
[IN TIMER INTERRUPT]software interrupt will run when timer interrupt finished
[IN TIMER INTERRUPT]timer interrupt end
[IN SOFTWARE INTERRUPT]software interrupt hit 0 times
[IN SOFTWARE INTERRUPT]software interrupt end
-----
[IN TIMER INTERRUPT]timer interrupt hit 1 times
[IN TIMER INTERRUPT]trigger software interrupt
[IN TIMER INTERRUPT]software interrupt will run when timer interrupt finished
[IN TIMER INTERRUPT]timer interrupt end
[IN SOFTWARE INTERRUPT]software interrupt hit 1 times
[IN SOFTWARE INTERRUPT]software interrupt end
-----
[IN TIMER INTERRUPT]timer interrupt hit 2 times
[IN TIMER INTERRUPT]trigger software interrupt
[IN TIMER INTERRUPT]software interrupt will run when timer interrupt finished
[IN TIMER INTERRUPT]timer interrupt end
[IN SOFTWARE INTERRUPT]software interrupt hit 2 times
[IN SOFTWARE INTERRUPT]software interrupt end
-----
```

demo_extirq

This `demo_extirq` application⁴⁷ is used to demonstrate how to use external interrupt with irqc controller.

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
cd application/baremetal/demo_extirq
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

⁴⁷ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/demo_extirq

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:22:37
Download Mode: ILM
CPU Frequency 16000942 Hz
CPU HartID: 0
You can press any key now to trigger uart receive interrupt
Enter uart0 interrupt, you just typed: 1
Enter uart0 interrupt, you just typed: 2
Enter uart0 interrupt, you just typed: 3
```

lowpower

This [lowpower application](#)⁴⁸ is used to demonstrate how to use low-power feature of RISC-V processor.

Timer interrupt is setup before enter to wfi mode, and global interrupt will be disabled, so interrupt handler will not be entered, and will directly resume to next pc of wfi.

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# Assume your processor has enabled low-power feature
# cd to the low-power directory
cd application/baremetal/lowpower
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:24:42
Download Mode: ILM
CPU Frequency 16001597 Hz
CPU HartID: 0
CSV, WFI Start/End, 205728/205743
CSV, WFI Cost, 15
```

coremark

This [coremark benchmark application](#)⁴⁹ is used to run EEMBC CoreMark Software.

EEMBC CoreMark Software is a product of EEMBC and is provided under the terms of the CoreMark License that is distributed with the official EEMBC COREMARK Software release. If you received this EEMBC CoreMark Software without the accompanying CoreMark License, you must discontinue use and download the official release from www.coremark.org.

In Nuclei N100 SDK, we provided code and Makefile for this coremark application. You can also optimize the COMMON_FLAGS defined in coremark application Makefile to get different score number.

- By default, this application runs for 15 iterations, you can also change this in Makefile. e.g. Change this `-DITERATIONS=15` to value such as `-DITERATIONS=20`
- macro `PERFORMANCE_RUN=1` is defined

⁴⁸ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/lowpower

⁴⁹ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/benchmark/coremark

- `STDCLIB ?= newlib_small` is added in its Makefile to enable float value print

Note: N100's time and cycle counter is only 24bit, easy to overflow, so make sure the iteration is small.

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the coremark directory
cd application/baremetal/benchmark/coremark
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:37:47
Download Mode: ILM
CPU Frequency 16000286 Hz
CPU HartID: 0
Start to run coremark for 15 iterations
2K performance run parameters for coremark.
CoreMark Size      : 666
Total ticks        : 5052270
Total time (secs): 0.315755
Iterations/Sec     : 47.505194
ERROR! Must execute for at least 10 secs for a valid result!
Iterations         : 15
Compiler version   : GCC13.1.1 20230713
Compiler flags     : -Ofast -fno-code-hoisting -fno-tree-vectorize -fno-common -finline-
↳functions -falign-functions=4 -falign-jumps=4 -falign-loops=4 -finline1
Memory location    : STACK
seedcrc           : 0xe9f5
[0]crclist        : 0xe714
[0]crcmatrix      : 0x1fd7
[0]crcstate       : 0x8e3a
[0]crcfinal       : 0x2d47
Errors detected

Print Personal Added Additional Info to Easy Visual Analysis

(Iterations is: 15
(total_ticks is: 5052270
(*) Assume the core running at 1 MHz
So the CoreMark/MHz can be calculated by:
(Iterations*1000000/total_ticks) = 2.968962 CoreMark/MHz

CSV, Benchmark, Iterations, Cycles, CoreMark/MHz
CSV, CoreMark, 15, 5052270, 2.968
```

dhystone

This `dhystone` benchmark application⁵⁰ is used to run DHRYSTONE Benchmark Software.

The Dhystone benchmark program has become a popular benchmark for CPU/compiler performance measurement, in particular in the area of minicomputers, workstations, PC's and microprocessors.

- It apparently satisfies a need for an easy-to-use integer benchmark;
- it gives a first performance indication which is more meaningful than MIPS numbers which, in their literal meaning (million instructions per second), cannot be used across different instruction sets (e.g. RISC vs. CISC).
- With the increasing use of the benchmark, it seems necessary to reconsider the benchmark and to check whether it can still fulfill this function.

In Nuclei N100 SDK, we provided code and Makefile for this `dhystone` application. You can also optimize the `COMMON_FLAGS` defined in `dhystone` application Makefile to get different score number.

- `STDCLIB ?= newlib_small` is added in its Makefile to enable float value print

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the dhystone directory
cd application/baremetal/benchmark/dhystone
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:38:59
Download Mode: ILM
CPU Frequency 16000942 Hz
CPU HartID: 0

Dhystone Benchmark, Version 2.1 (Language: C)

Program compiled without 'register' attribute

Please give the number of runs through the benchmark:
Execution starts, 2000 runs through Dhystone
Execution ends

Final values of the variables used in the benchmark:

Int_Glob:          5
    should be:    5
Bool_Glob:         1
    should be:    1
Ch_1_Glob:         A
    should be:    A
Ch_2_Glob:         B
    should be:    B
```

(continues on next page)

⁵⁰ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/benchmark/dhystone

(continued from previous page)

```

Arr_1_Glob[8]:      7
    should be:     7
Arr_2_Glob[8][7]:  2010
    should be:     Number_Of_Runs + 10
Ptr_Glob->
Ptr_Comp:          -1879032512
    should be:     (implementation-dependent)
Discr:            0
    should be:     0
Enum_Comp:        2
    should be:     2
Int_Comp:         17
    should be:     17
Str_Comp:         DHRYSTONE PROGRAM, SOME STRING
    should be:     DHRYSTONE PROGRAM, SOME STRING
Next_Ptr_Glob->
Ptr_Comp:          -1879032512
    should be:     (implementation-dependent), same as above
Discr:            0
    should be:     0
Enum_Comp:        1
    should be:     1
Int_Comp:         18
    should be:     18
Str_Comp:         DHRYSTONE PROGRAM, SOME STRING
    should be:     DHRYSTONE PROGRAM, SOME STRING
Int_1_Loc:        5
    should be:     5
Int_2_Loc:        13
    should be:     13
Int_3_Loc:        7
    should be:     7
Enum_Loc:         1
    should be:     1
Str_1_Loc:        DHRYSTONE PROGRAM, 1'ST STRING
    should be:     DHRYSTONE PROGRAM, 1'ST STRING
Str_2_Loc:        DHRYSTONE PROGRAM, 2'ND STRING
    should be:     DHRYSTONE PROGRAM, 2'ND STRING

```

Measured time too small to obtain meaningful results
Please increase number of runs

(*) User_Cycle for total run through Dhrystone with loops 2000:
1042022

So the DMIPS/MHz can be calculated by:

$$1000000 / (\text{User_Cycle} / \text{Number_Of_Runs}) / 1757 = 1.092399 \text{ DMIPS/MHz}$$

CSV, Benchmark, Iterations, Cycles, DMIPS/MHz
CSV, Dhrystone, 2000, 1042022, 1.092

whetstone

This whetstone benchmark application⁵¹ is used to run C/C++ Whetstone Benchmark Software (Single or Double Precision).

The Fortran Whetstone programs were the first general purpose benchmarks that set industry standards of computer system performance. Whetstone programs also addressed the question of the efficiency of different programming languages, an important issue not covered by more contemporary standard benchmarks.

In Nuclei N100 SDK, we provided code and Makefile for this whetstone application. You can also optimize the COMMON_FLAGS defined in whetstone application Makefile to get different score number.

- **STDCLIB ?= newlib_small** is added in its Makefile to enable float value print
- Extra **LDLIBS := -lm** is added in its Makefile to include the math library

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the whetstone directory
cd application/baremetal/benchmark/whetstone
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:41:32
Download Mode: ILM
CPU Frequency 16000942 Hz
CPU HartID: 0

#####
Single Precision C Whetstone Benchmark Opt 3 32 Bit
Calibrate
    14.54 Seconds          1  Passes (x 100)

Use 1  passes (x 100)

          Single Precision C/C++ Whetstone Benchmark
Loop content          Result          MFLOPS          MOPS          Seconds
N1 floating point -1.12475013732910156          0.148          0.130
N2 floating point -1.12274742126464844          0.149          0.901
N3 if then else    1.00000000000000000          226.099          0.000
N4 fixed point    12.00000000000000000          0.764          0.412
N5 sin,cos etc.   0.49909299612045288          0.015          5.601
N6 floating point 0.99999982118606567          0.142          3.804
N7 assignments    3.00000000000000000          71.241          0.003
N8 exp,sqrt etc.  0.75110614299774170          0.010          3.693

MWIPS          0.688          14.544
```

(continues on next page)

⁵¹ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/baremetal/benchmark/whetstone

(continued from previous page)

MWIPS/MHz	0.043	14.544
-----------	-------	--------

CSV, Benchmark, MWIPS/MHz

CSV, Whetstone, 0.042

5.7.3 FreeRTOS applications

demo

This `freertos demo` application⁵² is to show basic freertos task functions.

- Two freertos tasks are created
- A software timer is created

In Nuclei N100 SDK, we provided code and Makefile for this `freertos demo` application.

- **RTOS = FreeRTOS** is added in its Makefile to include FreeRTOS service
- The `configTICK_RATE_HZ` in `FreeRTOSConfig.h` is set to 100, you can change it to other number according to your requirement.

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the freertos demo directory
cd application/freertos/demo
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:44:39
Download Mode: ILM
CPU Frequency 16000942 Hz
CPU HartID: 0
Before StartScheduler
Enter to task_1
task1 is running 0.....
Enter to task_2
task2 is running 0.....
task1 is running 1.....
task2 is running 1.....
task1 is running 2.....
task2 is running 2.....
task1 is running 3.....
task2 is running 3.....
task1 is running 4.....
```

(continues on next page)

⁵² https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/freertos/demo

(continued from previous page)

```
task2 is running 4.....
timers Callback 0
task1 is running 5.....
task2 is running 5.....
task1 is running 6.....
task2 is running 6.....
task1 is running 7.....
```

5.7.4 UCOSII applications

demo

This `ucosii demo` application⁵³ is show basic ucosii task functions.

- 4 tasks are created
- 1 task is created first, and then create 3 other tasks and then suspend itself

In Nuclei N100 SDK, we provided code and Makefile for this `ucosii demo` application.

- **RTOS = UCOSII** is added in its Makefile to include UCOSII service
- The **OS_TICKS_PER_SEC** in `os_cfg.h` is by default set to 50, you can change it to other number according to your requirement.

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the ucosii demo directory
cd application/ucosii/demo
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:45:42
Download Mode: ILM
CPU Frequency 16000286 Hz
CPU HartID: 0
Start ucosii...
create start task success
start all task...
task3 is running... 1
task2 is running... 1
task1 is running... 1
task3 is running... 2
task2 is running... 2
task1 is running... 2
task3 is running... 3
task2 is running... 3
```

(continues on next page)

⁵³ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/ucosii/demo

(continued from previous page)

```
task3 is running... 4
task2 is running... 4
task1 is running... 3
task3 is running... 5
task2 is running... 5
task3 is running... 6
task2 is running... 6
```

5.7.5 RT-Thread applications

demo

This `rt-thread demo application`⁵⁴ is show basic rt-thread thread functions.

- main function is a pre-created thread by RT-Thread
- main thread will create 5 test threads using the same function `thread_entry`

In Nuclei N100 SDK, we provided code and Makefile for this `rtthread demo` application.

- **RTOS = RTThread** is added in its Makefile to include RT-Thread service
- The **RT_TICK_PER_SECOND** in `rtconfig.h` is by default set to `100`, you can change it to other number according to your requirement.

How to run this application:

```
# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the rtthread demo directory
cd application/rtthread/demo
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload
```

Expected output as below:

```
N100 Nuclei SDK Build Time: Jun  4 2024, 14:47:24
Download Mode: ILM
CPU Frequency 15999303 Hz
CPU HartID: 0

\ | /
- RT -   Thread Operating System
/ | \    3.1.3 build Jun  4 2024
2006 - 2019 Copyright by rt-thread team
Main thread count: 0
thread 0 count: 0
thread 1 count: 0
thread 2 count: 0
thread 3 count: 0
thread 4 count: 0
```

(continues on next page)

⁵⁴ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/rtthread/demo

(continued from previous page)

```

thread 0 count: 1
thread 1 count: 1
thread 2 count: 1
thread 3 count: 1
thread 4 count: 1
Main thread count: 1
thread 0 count: 2
thread 1 count: 2
thread 2 count: 2
thread 3 count: 2

```

msh

This `rt-thread msh application`⁵⁵ demonstrates msh shell in serial console which is a component of rt-thread.

- `MSH_CMD_EXPORT(nsdk, msh nuclei sdk demo)` exports a command `nsdk` to msh shell

In Nuclei N100 SDK, we provided code and Makefile for this `rtthread msh` application.

- `RTOS = RTThread` is added in its Makefile to include RT-Thread service
- `RTTHREAD_MSH := 1` is added in its Makefile to include RT-Thread msh component
- The `RT_TICK_PER_SECOND` in `rtconfig.h` is by default set to `100`, you can change it to other number according to your requirement.

How to run this application:

```

# Assume that you can set up the Tools and Nuclei N100 SDK environment
# cd to the rtthread msh directory
cd application/rtthread/msh
# Clean the application first
make SOC=evalsoc clean
# Build and upload the application
make SOC=evalsoc upload

```

Expected output as below:

```

N100 Nuclei SDK Build Time: Jun  4 2024, 14:48:20
Download Mode: ILM
CPU Frequency 16000286 Hz
CPU HartID: 0

\ | /
- RT -   Thread Operating System
/ | \    3.1.3 build Jun  4 2024
2006 - 2019 Copyright by rt-thread team
Hello RT-Thread!
msh >help
RT-Thread shell commands:
list_timer      - list timer in system
list_mailbox    - list mail box in system
list_sem        - list semaphore in system

```

(continues on next page)

⁵⁵ https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100/application/rtthread/msh

(continued from previous page)

```
list_thread    - list thread
version        - show RT-Thread version information
ps             - List threads in the system.
help          - RT-Thread shell help.
nsdk          - msh nuclei sdk demo

msh >ps
thread  pri  status      sp      stack size max used left tick  error
-----  ---  -
tshell   6  ready  0x000000d8 0x00001000    10%  0x00000005 000
tidle   7  ready  0x00000078 0x00000200    23%  0x00000020 000
main    2  suspend 0x000000b8 0x00000400    17%  0x00000013 000
msh >nsdk
Hello Nuclei SDK!
msh >
```

CHANGELOG

6.1 V0.2.1

Note:

- Please use Nuclei N100 SDK with **Nuclei Studio 2025.02**, get it from <https://nucleisys.com/download.php#tools>
 - For Terapines ZCC toolchain, please download it from <https://www.terapines.com/products/zcc> and replace Nuclei Studio IDE `toolchain/zcc` folder existing content
-

This is release version `0.2.1` of N100 SDK.

- SoC
 - Fix missing return for `IRQC_Register_IRQ` for `evalsoc`
 - Add NMI exception support for `evalsoc`, exception code is `0xFFF`
 - Support `zcm` extension in `evalsoc` linker script file
 - Fix wrong `npk` dependency described in `npk.yml` file which cause wrongly depends on `nuclei sdk`
- Tools
 - Add filter configuration feature in `nsdk_cli` tools, you can filter certain arches which you dont want to run via `SDK_IGNORED_EXTS` environment variable

6.2 V0.2.0

Note:

- Please use Nuclei N100 SDK with **Nuclei Studio 2025.02**, get it from <https://nucleisys.com/download.php#tools>
 - This version of N100 SDK only support latest `n100/ns100` cpu core.
-

This is release version `0.2.0` of N100 SDK.

- Application
 - Add an empty project for `n100 sdk`
 - Add an macro `TIMER_RELOAD` to control `demo_timer` timer interrupt reload via `TIME` or `TIMECMP`
 - reset cycle and instret when start to do benchmark

- NMSIS
 - Add `__set_rv_instret` and `__set_rv_cycle` API for N100 NMSIS Core
 - Fix `SysTick_Config` API by using `SysTick_Reload` instead of directly set `MTIMECMP` register to avoid easy 24b overflow
 - Remove `IRQC_SetPendingIRQ` and `IRQC_ClearPendingIRQ` API due to N100 only support level interrupt
 - Update `CSR_MCAUSE_Type/CSR_MSTATUS_Type/CSR_MSTATUSH_Type` union member
- SoC
 - Merge newlib stub code files into one stub code file
 - Add `CODESIZE` make variable to reduce application code size
 - Modify startup code for both gcc and iar startup code and linker script to support new startup and exception rules, see *Usage* (page 59)
 - Now only `sram` linker script is provided for reference, you can always modify startup and linker script code to match your real cpu design
 - Fix `delay_1ms` API may work wrongly due to `TIME` register overflow
 - N100 evalsoc external interrupt number changed from 30 to 16
 - N100 exception entry alignment changed to 4bytes to match n100 design
- Build System
 - Add `nuclei_llvm` and `terapines` toolchain supported, required Nuclei Studio 2025.02
 - Fix IAR prebuild projects build issues

6.3 V0.1.0

This is release version **0.1.0** of N100 SDK.

Note:

- Please use Nuclei N100 SDK with **Nuclei Studio 2024.06**, get it from <https://nucleisys.com/download.php#tools>
- Nuclei N100 SDK is modified based on Nuclei SDK **0.5.0** release, and will not merge back to Nuclei SDK in the future.
- This SDK is not compatible with Nuclei SDK for **200/300/600/900/1000** series, if you want SDK for these series, please switch to **master** or **develop** branch, see https://doc.nucleisys.com/nuclei_sdk/

-
- Application
 - Added baremetal/freertos/ucosii/rthread examples.
 - Baremetal examples contains helloworld, demo_irqc, demo_timer, demo_extirq, lowpower cases to show cpu interrupt and timer usage.
 - Baremetal examples also contains benchmark examples such as coremark, dhrystone, whetstone which can demonstrate the performance of Nuclei CPU.
 - RTOS examples contains different samples to show how to use FreeRTOS, UCOSII and RT-Thread on Nuclei N100.

- Build System
 - Support Nuclei 100 series RISC-V CPU Cores.
 - Support Nuclei RISC-V GCC toolchain, IAR Compiler and Terapines ZCC toolchain.
- NMSIS
 - NMSIS in N100 SDK is **not compatible with** standard NMSIS, this is modified to match Nuclei 100 series CPU.
 - Most of the APIs in this **modified NMSIS** are similar to standard NMSIS, but some APIs have been changed or added.
 - Please refer to the header files in `NMSIS/Core/Include` directory for more details.
- SoC
 - **Only Nuclei Evaluation SoC for 100 series is supported by this SDK.**
 - If you want to port to your own SoC, you modify based on this `evalsoc` implementation.
 - This SDK will also be generated by 100 series `nuclei_gen` tool, please take a look at the `evalsoc.memory`, `openocd_evalsoc.cfg`, `cpufeature.h` and `cpufeature.mk`, for IAR projects, you need to also check the linker script inside it.
 - Only `sram` and `flashxip` download modes are supported in this SDK, and the linker script is quite different to normal Nuclei SDK, please take care, especially the vector table and exception entry address are RTL configurable which means when your RTL configuration is different to our `evalsoc`, you need to modify the linker script to match your rtl configuration.
 - IAR support is also added in this SDK, and linker and startup/exception code are different from GCC, please take care when you port to your own SoC.
- RTOS
 - FreeRTOS/UCOSII/RT-Thread port for 100 series CPU are added in this SDK.
 - IAR compiler port and gcc/clang port are also supported by these RTOSes.
- IDE support
 - **Nuclei Studio 2024.06** will support this Nuclei N100 SDK via NPK solution just like Nuclei SDK.
 - IAR Workbench support is also done in this SDK, please take a try with it in `ideprojects/iar` folder.
- Documentation
 - The documentation is modified based on Nuclei SDK.
 - We have go through the whole documentation and modified it to match Nuclei N100 SDK, maybe some of them are not perfect, please feel free to correct me if you find any mistakes.

7.1 Why I can't download application?

- **Case 1: Remote communication error. Target disconnected.: Success.**

```
Nuclei OpenOCD, 64-bit Open On-Chip Debugger 0.10.0+dev-00014-g0eae03214 (2019-12-12-
↪07:43)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
Remote communication error. Target disconnected.: Success.
"monitor" command not supported by this target.
"monitor" command not supported by this target.
"monitor" command not supported by this target.
You can't do that when your target is ``exec'
"monitor" command not supported by this target.
"monitor" command not supported by this target.
```

Please check whether your driver is installed successfully via replace target upload to `run_openocd` as the board user manual described, especially, for **RV-STAR** and **Nuclei Eval SoC Evaluation** boards, For windows, you need to download the **HummingBird Debugger Windows Driver** from <https://nucleisys.com/developboard.php>, and install it.

If still not working, please check whether your JTAG connection is good or your CPU core is OK.

Note: The USB driver might lost when you re-plug the USB port, you might need to reinstall the driver.

- **Case 2: bfd requires flen 4, but target has flen 0**

```
bfd requires flen 4, but target has flen 0
"monitor" command not supported by this target.
"monitor" command not supported by this target.
"monitor" command not supported by this target.
You can't do that when your target is `exec'
"monitor" command not supported by this target.
"monitor" command not supported by this target.
```

bfd is abbreviation for **Binary File Descriptor**.

This is caused by the target core flen is 0, which means it didn't have float point unit in it, but your program is compiled using flen = 4, single point float unit used, which is incompatible, similar cases such as `bfd requires flen 8, but target has flen 4`

Just change your CORE to proper core settings and will solve this issue.

For example, if you compile your core with CORE=n307, just change it to CORE=n305.

- **Case 3: bfd requires xlen 8, but target has xlen 4**

```
bfd requires xlen 8, but target has xlen 4
"monitor" command not supported by this target.
"monitor" command not supported by this target.
"monitor" command not supported by this target.
You can't do that when your target is ``exec'
"monitor" command not supported by this target.
"monitor" command not supported by this target.
```

This issue is caused by the program is a riscv 64 program, but the core is a riscv 32 core, so just change your program to be compiled using a riscv 32 compile option.

For example, if you compile your core with CORE=ux600, just change it to CORE=n305.

7.2 How to select correct FTDI debugger?

From Nuclei N100 SDK release 0.2.9, the openocd configuration file doesn't contain `ftdi_device_desc`⁵⁶ line by default, so if there are more than one FTDI debuggers which has the same VID/PID(0x0403/0x6010) as Nuclei Debugger Kit use, then you might need to add extra `ftdi_device_desc` line in the openocd configuration file to describe the FTDI device description.

Or you can add extra `adapter serial your_serial_no` for your debugger, you can check its serial number via windows FT_PROG tool.

NOTE: for windows, you need to add an extra A to the serial number, eg. your serial number is FT6S9RD6, then this extra openocd config line should be `adapter serial "FT6S9RD6A"` for windows.

- For **Nuclei FPGA Evaluation Board**, you can check the openocd configuration file in `SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg`.
- For **Nuclei RVSTAR Board**, you can check the openocd configuration file in `SoC/gd32vf103/Board/gd32vf103v_rvstar/openocd_gd32vf103.cfg`.

For more details, please check [Debug with multiple FTDI devices](#)⁵⁷

7.3 Why I can't download application in Linux?

Please check that whether you have followed the [debugger kit manual](#)⁵⁸ to setup the USB JTAG drivers correctly. The windows steps and linux steps are different, please take care.

⁵⁶ <http://openocd.org/doc/html/Debug-Adapter-Configuration.html>

⁵⁷ https://doc.nucleisys.com/nuclei_studio_supply/27-debug_with_multiple_ftdi_devices/

⁵⁸ <https://nucleisys.com/developboard.php#ddr200t>

7.4 Why the provided application is not running correctly in my Nuclei FPGA Evaluation Board?

Please check the following items:

1. Did you program the correct Nuclei Evaluation FPGA bitstream?
2. Did you re-power the board, when you just programmed the board with FPGA bitstream?
3. Did you choose the right **CORE** as the Nuclei Evaluation FPGA bitstream present?
4. If your application is RTOS demos, did you run in `flashxip` mode, if yes, it is expected due to flash speed is really slow, you'd better try `ilm` or `flash` mode.
5. If still not working, you might need to check whether the FPGA bitstream is correct or not?

7.5 Access to github.com is slow, any workaround?

Access speed to github.com sometimes is slow and not stable, but if you want to clone source code, you can also switch to use our mirror site maintained in gitee.com.

This mirror will sync changes from github to gitee every 6 hours, that is 4 times a day.

You just need to replace the github to gitee when you clone any repo in **Nuclei-Software** or **riscv-mcu**.

For example, if you want to clone **nuclei-n100-sdk** using command `git clone -b develop_n100 https://github.com/Nuclei-Software/nuclei-sdk`, then you can achieve it by command `git clone -b develop_n100 https://gitee.com/Nuclei-Software/nuclei-sdk`

7.6 '.text' will not fit in region `ilm` or `.bss` will not fit in region `ram`

If you met similar message as below when build an application:

```
xxx/bin/ld: cifar10.elf section `.text' will not fit in region `ilm'
xxx/bin/ld: cifar10.elf section `.bss' will not fit in region `ram'
xxx/bin/ld: section .stack VMA [0000000009000f800,0000000009000ffff] overlaps section .bss_
↳ VMA [000000000900097c0,000000000900144eb]
xxx/bin/ld: region `ilm' overflowed by 43832 bytes
xxx/bin/ld: region `ram' overflowed by 0 bytes
```

It is caused by the program is too big, our default link script is 64K ILM, 64K DLM, 4M SPIFlash for Nuclei Demo/Eval SoC.

If your core has bigger ILM or DLM, you can change related linker script file according to your choice.

For example, if you want to change linker script for evalsoc on `nuclei_fpga_eval` ilm download mode: ILM to 512K, DLM to 256K, then you can change link script file `SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/gcc_evalsoc_ilm.ld` as below:

```
diff --git a/SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/gcc_evalsoc_ilm.ld b/SoC/
↳ evalsoc/Board/nuclei_fpga_eval/Source/GCC/gcc_evalsoc_ilm.ld
index 1ac5b90..08451b3 100644
--- a/SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/gcc_evalsoc_ilm.ld
+++ b/SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/gcc_evalsoc_ilm.ld
```

(continues on next page)

(continued from previous page)

```

@@ -28,8 +28,8 @@ ENTRY( _start )
MEMORY
{
- ilm (rxa!w) : ORIGIN = 0x80000000, LENGTH = 64K
- ram (wxa!r) : ORIGIN = 0x90000000, LENGTH = 64K
+ ilm (rxa!w) : ORIGIN = 0x80000000, LENGTH = 512K
+ ram (wxa!r) : ORIGIN = 0x90000000, LENGTH = 256K
}

```

7.7 undefined reference to `__errno` when using `libncrt` library

When you are using `libncrt` library, and linked with `-lm`, you may face below issues

```

/home/share/devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/
↳ riscv64-unknown-elf/13.1.1/../../../../riscv64-unknown-elf/bin/ld: /home/share/
↳ devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/riscv64-
↳ unknown-elf/13.1.1/../../../../riscv64-unknown-elf/lib/rv32imafdc/ilp32d/libm.a(libm_a-
↳ w_exp.o): in function `L1':
w_exp.c:(.text.exp+0x4a): undefined reference to `__errno'
/home/share/devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/
↳ riscv64-unknown-elf/13.1.1/../../../../riscv64-unknown-elf/bin/ld: /home/share/
↳ devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/riscv64-
↳ unknown-elf/13.1.1/../../../../riscv64-unknown-elf/lib/rv32imafdc/ilp32d/libm.a(libm_a-
↳ w_exp.o): in function `L0 ':
w_exp.c:(.text.exp+0x6e): undefined reference to `__errno'
/home/share/devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/
↳ riscv64-unknown-elf/13.1.1/../../../../riscv64-unknown-elf/bin/ld: /home/share/
↳ devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/riscv64-
↳ unknown-elf/13.1.1/../../../../riscv64-unknown-elf/lib/rv32imafdc/ilp32d/libm.a(libm_a-
↳ w_log.o): in function `log':
w_log.c:(.text.log+0x28): undefined reference to `__errno'
/home/share/devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/
↳ riscv64-unknown-elf/13.1.1/../../../../riscv64-unknown-elf/bin/ld: w_log.c:(.text.
↳ log+0x46): undefined reference to `__errno'
/home/share/devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/
↳ riscv64-unknown-elf/13.1.1/../../../../riscv64-unknown-elf/bin/ld: /home/share/
↳ devtools/toolchain/nuclei_gnu/linux64/newlibc/2023.10.14/gcc/bin/./lib/gcc/riscv64-
↳ unknown-elf/13.1.1/../../../../riscv64-unknown-elf/lib/rv32imafdc/ilp32d/libm.a(libm_a-
↳ math_err.o): in function `with_errno':
math_err.c:(.text.with_errno+0x12): undefined reference to `__errno'
collect2: error: ld returned 1 exit status

```

You can fix it by not link `-lm` library, since `libncrt` library already provided math library feature, so no need to link this math library.

7.8 undefined reference to fclose/sprintf similar API provided in system libraries

We no longer use `--specs=` option to select library we want to use, and we also passed `-nodefaultlibs` options to not use standard system libraries, this changes are made to support both gcc and clang toolchain, so in Nuclei N100 SDK build system, we control the needed system libraries to be linked as required by `STDCLIB` make variable, for details, please check `Build/toolchain/*.mk` makefiles, and also we use linker's group libraries feature `--start-group archives --end-group` to repeatedly search undefined reference in the group libraries, but this feature is not enabled in Eclipse CDT based IDE like Nuclei Studio, which undefined reference is searched in the order of library specified on the command line, so you may meet issue like undefined `fclose` reference even you linked `newlib nano c` library `-lc_nano` if the library order is not good, so to fix this issue, you may need to place the library in a good order and need to repeatedly link it, such as `-lgcc -lc_nano -lm -lsemihost -lgcov -lgcc -lc_nano`, and also we have opened an issue to track it, see <https://github.com/eclipse-embed-cdt/eclipse-plugins/issues/592>

7.9 fatal error: rvintrin.h: No such file or directory

If you are using Nuclei Toolchain 2023.10, `rvintrin.h` no longer exist for B extension, please don't include this header file. If you want to use an intrinsic API for B extension, you need to write using c asm intrinsic.

7.10 riscv-nuclei-elf-gcc: not found when using Nuclei Studio 2023.10

```riscv-nuclei-elf-gcc``(gcc10)` has changed to ```riscv64-unknown-elf-gcc``(gcc13)` since Nuclei Studio 2023.10 or Nuclei RISC-V Toolchain 2023.10, so if you are using older toolchain created npk package or ide project, you may face this build fail issue, you can follow the user guide of Nuclei Studio 2023.10 to fix this issue, see chapter 8.



LICENSE

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object

(continues on next page)

(continued from previous page)

form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(continues on next page)

(continued from previous page)

- (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
- (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
- (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each

(continues on next page)

(continued from previous page)

Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

(continues on next page)

(continued from previous page)

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.



## GLOSSARY

**API**

(Application Program Interface) A defined set of routines and protocols for building application software.

**DSP**

(Digital Signal Processing) is the use of digital processing, such as by computers or more specialized digital signal processors, to perform a wide variety of signal processing operations.

**ISR**

(Interrupt Service Routine) Also known as an interrupt handler, an ISR is a callback function whose execution is triggered by a hardware interrupt (or software interrupt instructions) and is used to handle high-priority conditions that require interrupting the current code executing on the processor.

**NN**

(Neural Network) is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes.

**XIP**

(eXecute In Place) a method of executing programs directly from long term storage rather than copying it into RAM, saving writable memory for dynamic data and not the static program code.



## APPENDIX

- **Nuclei Tools and Documents:** <https://nucleisys.com/download.php>
- **Nuclei Software Opensource Organization:** <https://github.com/Nuclei-Software>
- **RISC-V MCU Opensource Organization:** <https://github.com/riscv-mcu>
- **Nuclei Toolchain Repo:** <https://github.com/riscv-mcu/riscv-gnu-toolchain>
- **Nuclei OpenOCD Repo:** <https://github.com/riscv-mcu/riscv-openocd>
- **Nuclei QEMU Repo:** <https://github.com/riscv-mcu/qemu>
- **Nuclei N100 SDK:** [https://github.com/Nuclei-Software/nuclei-sdk/tree/develop\\_n100](https://github.com/Nuclei-Software/nuclei-sdk/tree/develop_n100)
- **NMSIS:** <https://github.com/Nuclei-Software/NMSIS>
- **Nuclei RISC-V IP Products:** <https://www.nucleisys.com/product.php>
- **RISC-V MCU Community Website:** <https://www.riscv-mcu.com/>
- **Nuclei RISC-V CPU Spec:** [https://doc.nucleisys.com/nuclei\\_spec](https://doc.nucleisys.com/nuclei_spec)
- **RISC-V ISA Specifications(Ratified):** <https://riscv.org/technical/specifications>
- **RISC-V Architecture Profiles:** <https://github.com/riscv/riscv-profiles>
- **RISC-V Bitmanip(B) Extension Spec:** <https://github.com/riscv/riscv-bitmanip>
- **RISC-V Packed SIMD(P) Extension Spec:** <https://github.com/riscv/riscv-p-spec>
- **RISC-V Cryptography(K) Extension Spec:** <https://github.com/riscv/riscv-crypto>
- **RISC-V Vector(V) Extension Spec:** <https://github.com/riscv/riscv-v-spec>
- **RISC-V Vector Intrinsic API Spec:** <https://github.com/riscv-non-isa/rvv-intrinsic-doc>
- **RISC-V ISA Extension Spec Status:** <https://wiki.riscv.org/display/HOME/Specification+Status>
- **Nuclei Bumblebee Core Document:** [https://github.com/nucleisys/Bumblebee\\_Core\\_Doc](https://github.com/nucleisys/Bumblebee_Core_Doc)



## INDICES AND TABLES

- genindex
- search



## INDEX

### A

API, [97](#)

### D

DSP, [97](#)

### I

ISR, [97](#)

### N

NN, [97](#)

### X

XIP, [97](#)