Hi3861 V100 / Hi3861L V100

FAQs

Issue	**01**
Date	**2020-04-30**

HiSilicon (Shanghai) Technologies Co., Ltd.

Address:	New R&D Center, 49 Wuhe Road, Bantian, Longgang District, Shenzhen 518129 P. R. China
Website:	https://www.hisilicon.com/en/
Email:	support@hisilicon.com

About This Document

Purpose

This document describes the frequently asked questions (FAQs) in the Hi3861 V100 and Hi3861L V100 solutions.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3861	V100
Hi3861L	V100

Intended Audience

The document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
⚠ DANGER	Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury.
⚠ WARNING	Indicates a hazard with a medium level of risk which, if not avoided, could result in death or serious injury.

Symbol	Description
⚠ CAUTION	Indicates a hazard with a low level of risk which, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
📖 NOTE	Supplements the important information in the main text. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

Change History

Issue	Date	Change Description
01	2020-04-30	This issue is the first official release. ● **3 System Errors** is added.
00B02	2020-02-12	● In **2.1.1 Serial Port Connection Failure on HiBurn**, **Figure 2-1** is updated. ● In **2.1.2 Message "errno" Is Displayed After HiBurn Is Connected to the Serial Port**, **Solution** is updated.
00B01	2020-01-15	This issue is the first draft release.

Contents

1 Development Environment and Usage of the SDK

This chapter describes the problems that occur during the environment setup and usage of the software development kit (SDK).

NOTE

- For details about how to set up the SDK development environment and use the SDK for Hi3861 V100 and Hi3861 V100, see the *Hi3861 V100/Hi3861L V100 SDK Development Environment Setup User Guide* and *Hi3861 V100/Hi3861L V100 Third-Party Software Porting Guide*.
- The figures in this section are used to describe the test code for problems. The actual version may be different. The version required by the build script prevails.

1.1 SDK Development Environment

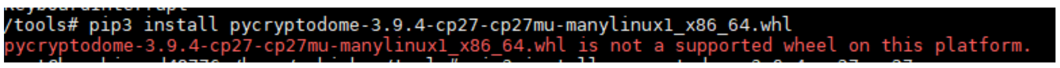
1.1.1 Python Library Installation Failure

1.1.1.1 Conflicts Between Python Library Version and Python Version

Symptom

The PyCryptodome tool library does not match the Python version, as shown in **Figure 1-1**.

Figure 1-1 Mismatched Python library version



```
/tools# pip3 install pycryptodome-3.9.4-cp27-cp27mu-manylinux1_x86_64.whl
pycryptodome-3.9.4-cp27-cp27mu-manylinux1_x86_64.whl is not a supported wheel on this platform.
```

Cause Analysis

The PyCryptodome library version does not match the Python version. In **Figure 1-1**, the PyCryptodome version is adapted to Python 2.7 (cp27). Therefore, the PyCryptodome library cannot be installed using **pip3**.

Solution

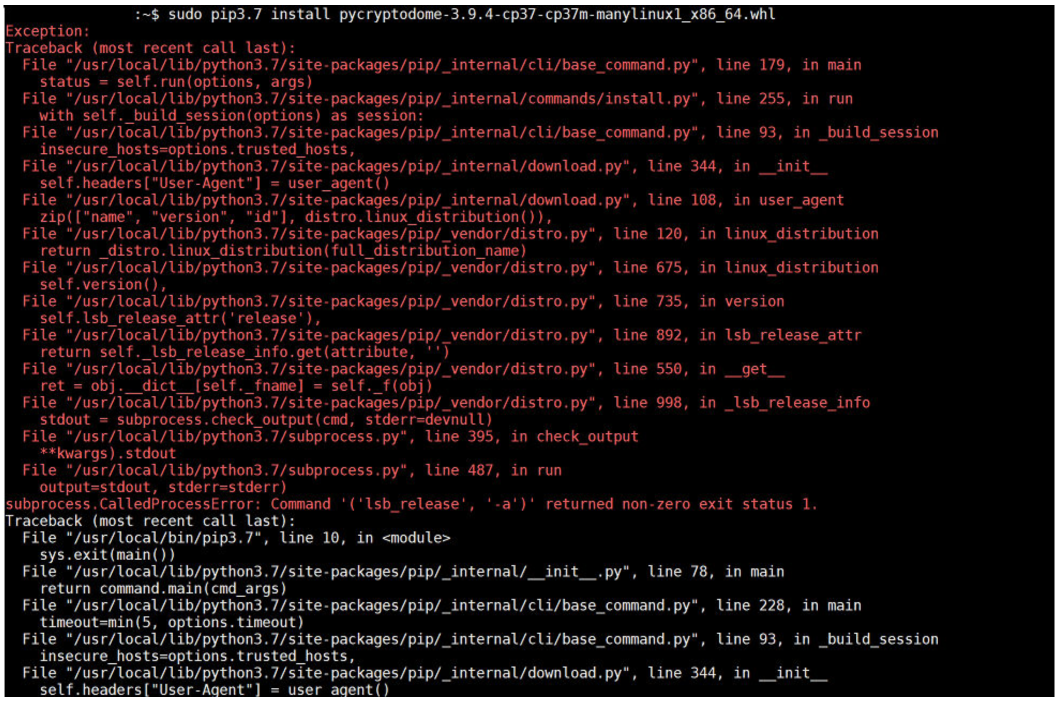
Query the tool library matching the current Python version from the official website of the tool, download the tool library, and install it again.

1.1.1.2 Lacking Linux System Tool During Tool Library Installation

Symptom

The installation of some Python libraries requires the support of Linux system tools. If a tool on which the Python libraries depend is lacking, the installation will fail, as shown in **Figure 1-2**.

Figure 1-2 Python library installation failure due to lack of a tool



```
               :~$ sudo pip3.7 install pycryptodome-3.9.4-cp37-cp37m-manylinux1_x86_64.whl
Exception:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 179, in main
    status = self.run(options, args)
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/commands/install.py", line 255, in run
    with self._build_session(options) as session:
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 93, in _build_session
    insecure_hosts=options.trusted_hosts,
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/download.py", line 344, in __init__
    self.headers["User-Agent"] = user_agent()
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/download.py", line 108, in user_agent
    zip(["name", "version", "id"], distro.linux_distribution()),
  File "/usr/local/lib/python3.7/site-packages/pip/_vendor/distro.py", line 120, in linux_distribution
    return _distro.linux_distribution(full_distribution_name)
  File "/usr/local/lib/python3.7/site-packages/pip/_vendor/distro.py", line 675, in linux_distribution
    self.version(),
  File "/usr/local/lib/python3.7/site-packages/pip/_vendor/distro.py", line 735, in version
    self.lsb_release_attr('release'),
  File "/usr/local/lib/python3.7/site-packages/pip/_vendor/distro.py", line 892, in lsb_release_attr
    return self._lsb_release_info.get(attribute, '')
  File "/usr/local/lib/python3.7/site-packages/pip/_vendor/distro.py", line 550, in __get__
    ret = obj.__dict__[self._fname] = self._f(obj)
  File "/usr/local/lib/python3.7/site-packages/pip/_vendor/distro.py", line 998, in _lsb_release_info
    stdout = subprocess.check_output(cmd, stderr=devnull)
  File "/usr/local/lib/python3.7/subprocess.py", line 395, in check_output
    **kwargs).stdout
  File "/usr/local/lib/python3.7/subprocess.py", line 487, in run
    output=stdout, stderr=stderr)
subprocess.CalledProcessError: Command '('lsb_release', '-a')' returned non-zero exit status 1.
Traceback (most recent call last):
  File "/usr/local/bin/pip3.7", line 10, in <module>
    sys.exit(main())
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/__init__.py", line 78, in main
    return command.main(cmd_args)
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 228, in main
    timeout=min(5, options.timeout)
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 93, in _build_session
    insecure_hosts=options.trusted_hosts,
  File "/usr/local/lib/python3.7/site-packages/pip/_internal/download.py", line 344, in __init__
    self.headers["User-Agent"] = user_agent()
```

Cause Analysis

During the installation of the PyCryptodome tool library, the lsb_release tool in Linux is required to query the Linux version. If the lsb_release tool is lacking, the Python tool library will fail to be installed. In **Figure 1-2**, the lsb_release tool is not installed in the Linux operating system (OS), which causes the failure.

Solution

Install the required tool in the Linux OS. Run the **sudo apt-get install lsb-core -y** command to install the lsb_release tool on Ubuntu. After the installation, run the

lsb_release -a command on the Linux terminal to check whether the command exists and whether the return is correct.

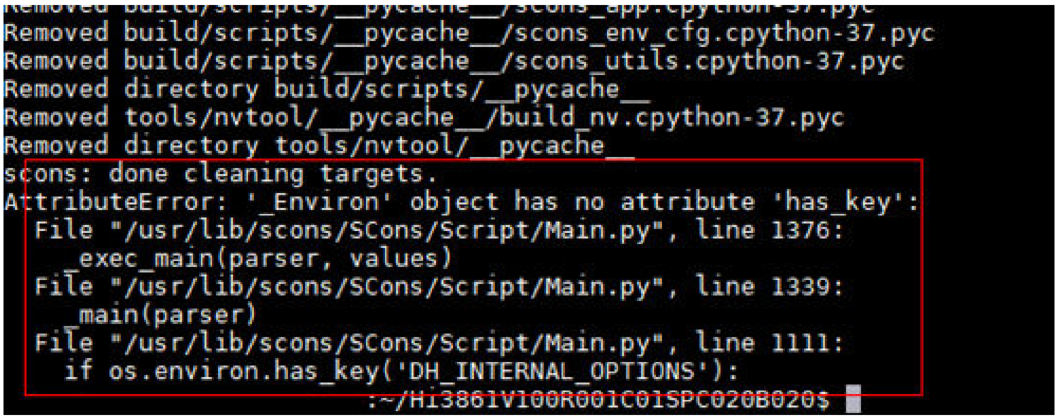
1.1.2 SCons Version Error

1.1.2.1 Conflicts Between SCons Version and Python Version

Symptom

The SCons version does not match the Python version, as shown in **Figure 1-3**.

Figure 1-3 Example 1 of SCons version mismatch



```
Removed build/scripts/__pycache__/scons_app.cpython-37.pyc
Removed build/scripts/__pycache__/scons_env_cfg.cpython-37.pyc
Removed build/scripts/__pycache__/scons_utils.cpython-37.pyc
Removed directory build/scripts/__pycache__
Removed tools/nvtool/__pycache__/build_nv.cpython-37.pyc
Removed directory tools/nvtool/__pycache__
scons: done cleaning targets.
AttributeError: '_Environ' object has no attribute 'has_key':
  File "/usr/lib/scons/SCons/Script/Main.py", line 1376:
    _exec_main(parser, values)
  File "/usr/lib/scons/SCons/Script/Main.py", line 1339:
    _main(parser)
  File "/usr/lib/scons/SCons/Script/Main.py", line 1111:
    if os.environ.has_key('DH_INTERNAL_OPTIONS'):
                        :~/Hi3861V100R001C01SPC020B020$
```

Cause Analysis

The SCons version in the SDK must be 3.0.1 or later. This version adapts to Python 3. The **has_key** function does not exist in the Python 3 syntax. The SCons version in **Figure 1-3** is adapted to Python 2, which does not meet the compilation requirements.

Solution

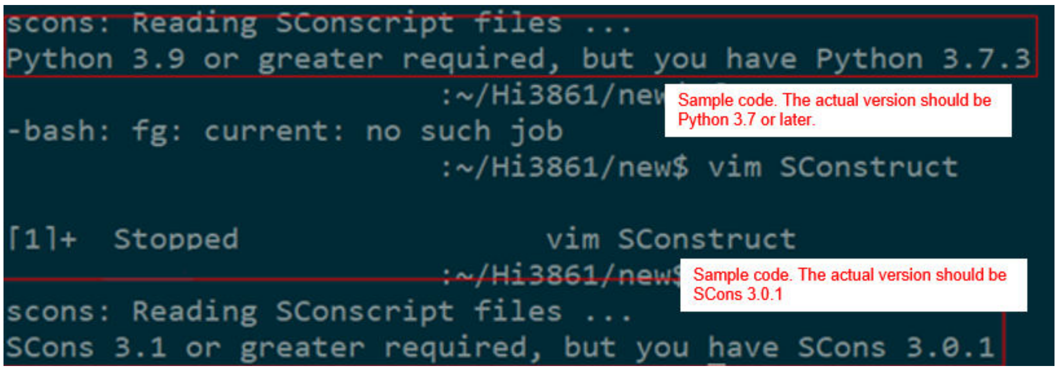
Run the **scons –v** command to check whether the version is correct. Uninstall the Scons of the earlier version and install the Scons of the correct version.

1.1.2.2 Conflicts Between the Build Script and the Actual Version

Symptom

The SCons version used in the build script is inconsistent with the actual version, as shown in **Figure 1-4**.

Figure 1-4 Example 2 of SCons version mismatch



```
scons: Reading SConscript files ...
Python 3.9 or greater required, but you have Python 3.7.3
                        :~/Hi3861/new
-bash: fg: current: no such job
                        :~/Hi3861/new$ vim SConstruct

[1]+  Stopped                 vim SConstruct
                        :~/Hi3861/new
scons: Reading SConscript files ...
SCons 3.1 or greater required, but you have SCons 3.0.1
```

Sample code. The actual version should be Python 3.7 or later.

Sample code. The actual version should be SCons 3.0.1

Cause Analysis

The SDK is built using SCons. The build script contains the option for selecting the SCons version. If the SCons version does not match, the script cannot run properly. You need to install the environment as required.

Solution

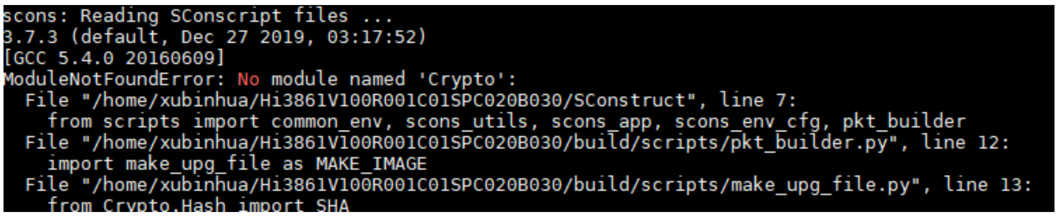
Uninstall the SCons of the earlier version and install the SCons of the specified version using the SCons build script.

1.1.3 Dependency Package Problem

Symptom

The version of the dependency package does not match the actual version, as shown in **Figure 1-5**.

Figure 1-5 Mismatch between the PyCryptodome version and Python version



```
scons: Reading SConscript files ...
3.7.3 (default, Dec 27 2019, 03:17:52)
[GCC 5.4.0 20160609]
ModuleNotFoundError: No module named 'Crypto':
  File "/home/xubinhua/Hi3861V100R001C01SPC020B030/SConstruct", line 7:
    from scripts import common_env, scons_utils, scons_app, scons_env_cfg, pkt_builder
  File "/home/xubinhua/Hi3861V100R001C01SPC020B030/build/scripts/pkt_builder.py", line 12:
    import make_upg_file as MAKE_IMAGE
  File "/home/xubinhua/Hi3861V100R001C01SPC020B030/build/scripts/make_upg_file.py", line 13:
    from Crypto.Hash import SHA
```

Cause Analysis

This problem often occurs because PyCryptodome is not installed or the installed version does not match the Python version. The Linux system allows multiple Python versions to be installed. After being installed based on the Python version, the dependency package cannot be shared. During installation of a dependency package, an incorrect Python version may be used.

Solution

The dependency package in the SDK must support Python 3.7. Ensure that the installed dependency package matches the Python version required for compilation. If the dependency package is not installed, use Python 3.7 to install it.

1.2 SDK Usage

1.2.1 Python Version Error

Symptom

The Python version error may have multiple causes. For example, the installation library version described in **1.1 SDK Development Environment** does not match. The following provides some common troubleshooting methods and solutions.

Solution

Step 1 Run the **python -v** command to check the default Python version (Python 3.7).

- If Python 3.7 is not installed, install it.
- If the default Python version cannot be changed, create a soft link (for example, **~/bin**) for Python 3.7 in the personal directory, add the directory to the **PATH** system parameter, and place the directory in the front of **PATH**. Log in to the terminal again and check the Python version.

Step 2 Run SCons again to check the Python version.

----End

1.2.2 No Specified Compiler

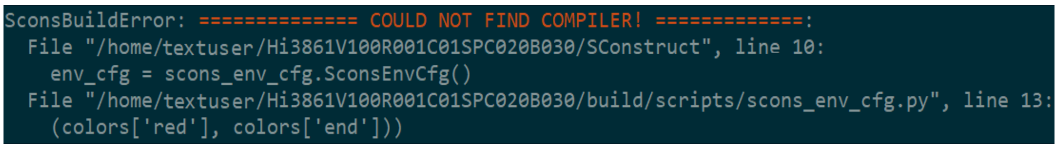
Symptom

Before the SDK compilation, the system checks whether the compiler is correctly specified. The possible causes are as follows:

- The compiler path is not added to **$PATH**, or the specified path is incorrect.
- **$PATH** specifies multiple compiler paths. As a result, the check fails.
- The compiler is not completely installed. The **bin** directory of the compiler does not contain **riscv32-unknown-elf-gcc**.

If the specified compiler is incorrect, information similar to **Figure 1-6** is displayed.

Figure 1-6 Compiler exception example



```
SconsBuildError: ============== COULD NOT FIND COMPILER! =============:
  File "/home/textuser/Hi3861V100R001C01SPC020B030/SConstruct", line 10:
    env_cfg = scons_env_cfg.SconsEnvCfg()
  File "/home/textuser/Hi3861V100R001C01SPC020B030/build/scripts/scons_env_cfg.py", line 13:
    (colors['red'], colors['end']))
```

Solution

Check the environment variable **$PATH** and correctly set the compiler path.

The compiler check must meet the following conditions:

- The path contains **hcc_riscv32**.
- The path ends with **bin** or **bin/**.
- The **bin** directory must contain **riscv32-unknown-elf-gcc**.
- Do not place compilers of multiple versions in the same path that meets the preceding three conditions. Otherwise, the system uses the first found compiler by default, which may not match the required version.

Common shell commands:

```
echo $PATH                     Query the path defined by the environment variable $PATH.
#export PATH=$PATH:/absolute_path   Set the environment variable $PATH.
```

2 Tool

This chapter describes the frequently asked questions (FAQs) and solutions for using HiBurn.

NOTE

For details about how to use the Hi3861 V100 or Hi3861 V100 HiBurn tool, see the *Hi3861 V100/Hi3861L V100 HiBurn User Guide*. Install and use the tool as prompted.

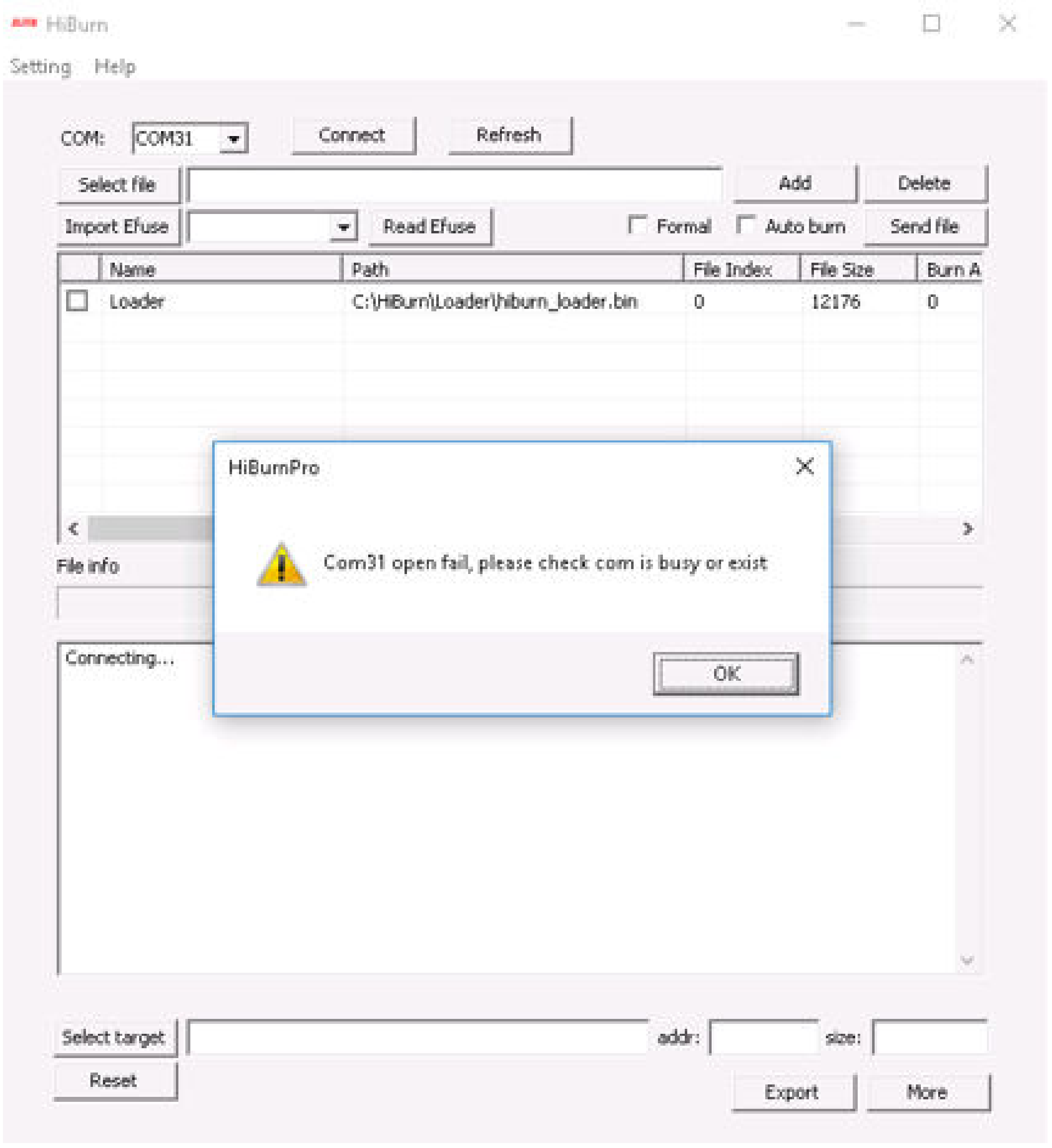
2.1 HiBurn FAQs

2.1.1 Serial Port Connection Failure on HiBurn

Symptom

HiBurn fails to connect to the serial port (COM port) because the hardware is not connected, the serial port driver is not installed, the serial port is incorrectly selected, or HiBurn is not properly used. **Figure 2-1** shows an example of the HiBurn GUI when a serial port is occupied.

Figure 2-1 HiBurn GUI when a serial port is occupied



	Name	Path	File Index	File Size	Burn A
☐	Loader	C:\HiBurn\Loader\hiburn_loader.bin	0	12176	0

```
Connecting...
```

Com31 open fail, please check com is busy or exist

Solution

The following provides some common solutions:

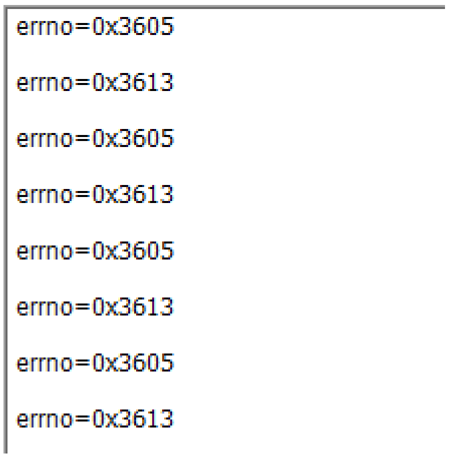
- Remove and insert the board, and check the connection between the hardware and the PC in **Device Manager**.
- Choose **Device Manager** > **Port** to check the installation of the serial port driver and select the correct port.
- Use HiBurn to check whether the serial port is occupied by other tools. If the serial port is occupied, close the tool that occupies the serial port, click **Refresh**, and then select and connect the port.
- After you press the board reset button, HiBurn displays a message, indicating that the serial port is connected properly.

2.1.2 Message "errno" Is Displayed After HiBurn Is Connected to the Serial Port

Symptom

When the download fails due to interruption or power-off, after you reconnect the board and press the reset button for multiple times, the board enters the error program of the download failure. **Figure 2-2** describes the error information.

Figure 2-2 Example of error information



```
errno=0x3605

errno=0x3613

errno=0x3605

errno=0x3613

errno=0x3605

errno=0x3613

errno=0x3605

errno=0x3613
```

Solution

Remove and insert the board, power it on, and connect it to HiBurn.

3 System Errors

This chapter describes watchdog-related problems and crash information printing problems caused by exceptions.

3.1 Watchdog FAQs

3.1.1 Watchdog

A watchdog is used to prevent abnormal scheduling of system tasks. For example, if a task or interrupt runs for a long time and the watchdog cannot be fed within the specified time, the system is considered abnormal and automatically resets for protection.

3.1.2 Watchdog Timeout

The default watchdog timeout period is **PRODUCT_CFG_AUTO_WDG_RESET_SYSTEM_TIMEOUT** (defined in the **hi_config.h** file). You can modify the value of the macro **PRODUCT_CFG_AUTO_WDG_RESET_SYSTEM_TIMEOUT** to define the watchdog timeout period based on the actual scenario. The user-defined timeout period must be greater than or equal to 6500 ms.

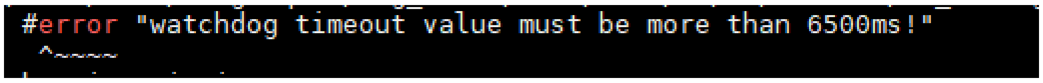
Because the system automatically triggers the watchdog feeding operation in the idle time, the watchdog timeout is generally caused by blocking. As a result, the system cannot enter the idle state. System blocking is classified into the following types:

- When the system is blocked in a task, the watchdog displays the crash information, which contains "watchdog isr". After a period of time, the system is reset.
- When the system is blocked in an interrupt, the watchdog displays the interrupt blocking position and directly resets the system.

3.1.3 User-Defined Watchdog Timeout

If a user-defined watchdog timeout occurs during compilation, check whether the user-defined timeout is less than 6500 ms. If the user-defined timeout is less than 6500 ms, a compilation error occurs. In this case, set the user-defined timeout to a value greater than 6500 ms.

Figure 3-1 User-defined watchdog timeout



```
#error "watchdog timeout value must be more than 6500ms!"
 ^~~~~
```

3.2 System Crash FAQs

3.2.1 Crash Information

The crash information area stores information about program running exceptions, such as stack overflow, invalid address access, and memory corruption.

3.2.2 Locating Crash Problems

3.2.2.1 Viewing and Exporting Crash Information

Call **hi_syserr_record_crash_info** in **app_main** to determine whether to store the crash information. After the board is restarted, you can run **AT+DUMP** to view the last crash information.

3.2.2.2 Locating a Crash Problem

To locate a system crash, you need to analyze the crash information, map file, and .asm file.

The fault needs to be analyzed based on the modifications. If necessary, determine the fault recurrence regularity based on the application scenario (such as voltage and clock). The fault locating procedure is as follows:

Step 1 Check whether the crash information is available. When the system crashes, the current crash information is displayed. To view the latest crash information, run **AT+DUMP**.

Step 2 Determine the crash type based on the following information: whether "watchdog_isr" is displayed in the crash information, whether the stack overflow flag in the task information is **1**, and CPU register values such as mcause and ccause.

Step 3 Check the .asm file based on the mepc to locate the function where the system crashes.

Step 4 Based on the track scheduling information and function call information, determine the position, task status, and function context before and after the exception, and modify the information.

----End

3.2.2.3 Parsing Crash Information

3.2.2.3.1 Crash Information Content

Table 3-1 describes the main content of the crash information.

Table 3-1 Crash information

Member	Description
Version number	SDK software version number of the current version, for example, **kernel_ver:Hi3861V100 V100R001C00SPC010**
Exception summary	Task name, task ID, task stack size, and exception type when an exception occurs
CPU register information	CPU register when an exception occurs, such as the mepc, mcause, and ccause
Memory pool information	Memory pool size, peak value, used memory size, and number of memory allocation failures when an exception occurs
Task information	Information about the task when an exception occurs, including the task stack name, status, ID, stack pointer, and top and bottom of the stack
Watchdog information	Watchdog information when an exception occurs. If the watchdog is reset, the PC triggers the instruction running address when the watchdog is abnormal.
Track scheduling information	Recent track scheduling information, which displays the system scheduling track, for example, interrupt and task
Function call stack information	Code execution records in the task stack, which indicate the code execution sequence from top to bottom. You can obtain the program context when an exception occurs based on the information.

3.2.2.3.2 Basic Information

The basic information includes the cyclic redundancy check (CRC) value, crash information length, and log version. You can also determine the cause of the system reset based on Table 3-3. The basic information is shown in Table 3-2.

Table 3-2 Basic information

Member	Description
crc_usable	Whether the crash information CRC is correct (**1** indicates that the CRC is correct)

Member	Description
info_len	Crash information length
crc_val	CRC16 value
log_ver	Log version
eid	Reset type. For details, see **Table 3-3**.
rid	Exception type. For details, see **Table 3-3**.
crash_tsec	Boot time relative to the exception occurrence
boot_ver	Boot version number in secure boot mode (The value is **0** in non-secure boot mode.)
type_name	Type name, corresponding to **eid**

The **eid** and **rid** parameters are used together to read the values of **eid** and **rid**. Then, you can quickly determine the cause of the system crash or exception by referring to **Table 3-3**.

For example, if the value of **eid** is **0x7** and the value of **rid** is **0x200**, the system reset is caused by the Wi-Fi module.

Table 3-3 Description of eid and rid

Event Description	eid	rid
Power-off restart, initial power-on, or hard reset	0x0	0x0
System crash in a task	0x1	mcause register value. For details, see **Table 3-5**.
System crash in an interrupt	0x2	mcause register value. For details, see **Table 3-5**.
System crash in a watchdog task	0x3	0x0
System crash in a watchdog interrupt	0x4	0x0
System soft reset (unknown reason)	0x5	0x0
System soft reset (automatic reset through AT commands)	0x5	0x1
System soft reset (upgrade reset. In the case of dual-partition upgrade, the system is running in partition A.)	0x5	0x2

Event Description	eid	rid
System soft reset (upgrade reset. In the case of dual-partition upgrade, the system is running in partition B.)	0x5	0x3
System soft reset (triggered by the Wi-Fi driver)	0x5	0x4
System soft reset (normal reset)	0x5	0x5
System soft reset (user-defined reset 0)	0x5	0x6
System soft reset (user-defined reset 1)	0x5	0x87
System soft reset (automatic reset when an AT command is busy for multiple consecutive times)	0x5	0x8

3.2.2.3.3 CPU Register Information

The CPU register information shown in **Table 3-4** is provided to help R&D engineers locate the cause and position of the system crash.

Table 3-4 Description of CPU registers related to system crash

Member	Description
mepc	Machine exception program counter. When an exception occurs, mepc points to the instruction that causes the exception. For an interrupt, mepc points to the position to be recovered after the interrupt is processed.
mstatus	Machine status register. The least significant bit (LSB) determines whether the interrupt is enabled. **0**: disabled; **1**: enabled
mtval	Machine trap register. It stores the error address in the address exception or the instruction where the instruction exception occurs. For other errors, the value is **0**.
mcause	Machine exception register. It stores the cause of the current exception or interrupt. You can obtain the type of the current exception or interrupt by querying **Table 3-5**.
ccause	ccause is a supplementary to mcause. For some exceptions, you can read the content of the ccause register to further determine the exception type.

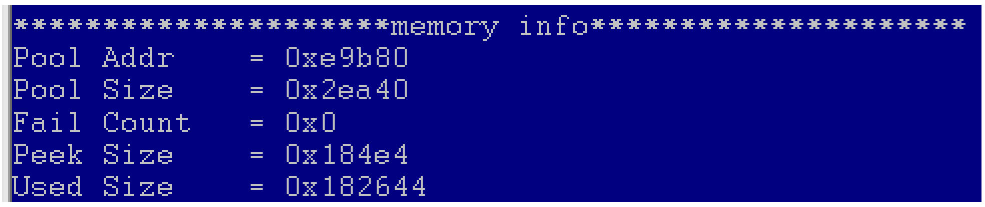
Table 3-5 Description of mcause and ccause exceptions

Error Code	mcause	ccause
0x80000021	Machine non-standard local interrupt	Not available
0x0	Instruction address misaligned	Not available
0x1	Instruction access fault	Memory map region access fault
0x2	Illegal instruction	AXIM error response
0x3	Breakpoint	AHBM error response
0x4	Load address misaligned	Crossing PMP entries
0x5	Load access fault	System register access fault
0x6	Store/AMO address misaligned	No PMP entry matched
0x7	Store/AMO access fault	PMP access fault
0x8	Environment call from U-mode	CMO access fault
0x9	Environment call from S-mode	CSR access fault
0xa	Reserved	LDM/STMIA instruction
0xb	Environment call from M-mode	ITCM write access fault
0xc	Instruction page fault	Not available
0xd	Load page fault	Not available
0xe	Reserved	Not available
0xf	Store/AMO page fault	Not available
> 0xf	Reserved	Not available

3.2.2.3.4 Memory Pool Information

Figure 3-2 shows the printed memory pool information. You can check whether the resource usage is normal based on the memory pool size, peak value, number of memory allocation failures, and used memory.

Figure 3-2 Example of memory pool information



```
*********************memory info*********************
Pool Addr    = 0xe9b80
Pool Size    = 0x2ea40
Fail Count   = 0x0
Peek Size    = 0x184e4
Used Size    = 0x182644
```

3.2.2.3.5 Track Scheduling Information

The current track scheduling item (**current_item**) and the number of scheduling items to be printed (**item_cnt**) are printed first. Then, a table is printed, which contains the following content:

- **Index**: ID of a track scheduling project
- **TrackType**: scheduling item type. For details, see **Table 3-6**.
- **TrackID**: scheduling item ID. The ID content is determined by **TrackType**.
  - Interrupt vector
  - Working queue or priority
  - Task ID
  - Event (The value is **0**.)
- **CurTime**: number of ticks in the system when an event occurs
- **Data1** and **Data2**: pointer to an event that happens on the PC. When the scheduling item type is task and the specific operation is task scheduling, **Data1** records the print information on the PC for an old task and **Data2** records the print information on the PC for a new task.

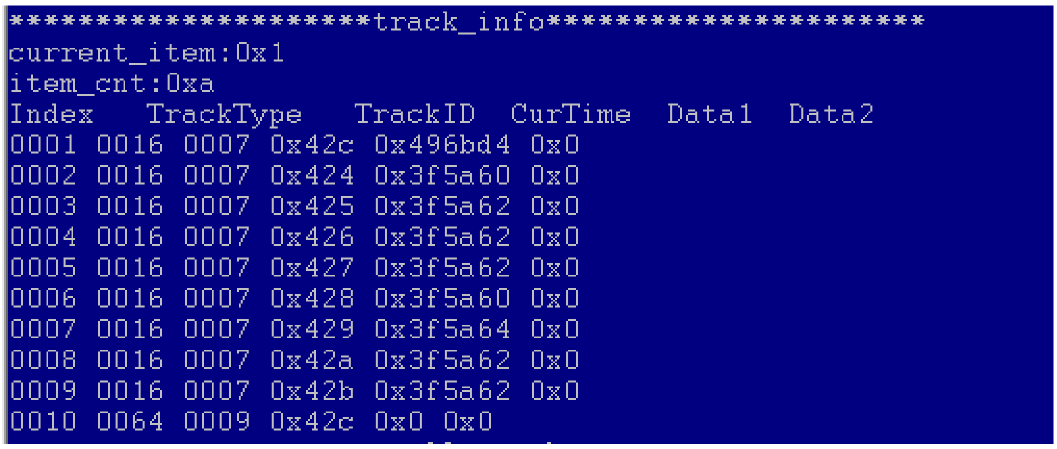
Table 3-6 TrackType

Scheduling Item type	Scheduling Item Operation	TrackType
TRACK_ISR	TRAP_INT	0016
TRACK_IRQ	ENABLE_IRQ	0032
	DISABLE_IRQ	0033
TRACK_TASK	CREATE_TSK	0064
	SCHEDULE_TSK	0065
	DELETE_TSK	0066
TRACK_WORKQ	ENQUEUE_WORKQ	0128
	SCHEDULE_WORKQ	0129
	DEQUEUE_WORKQ	0130
TRACK_SWTMR	CREATE_SWTMR	0256
	START_SWTMR	0257
	MODIFY_SWTMR	0258
	SCHDING_SWTMR	0259
	SCHDED_SWTMR	0260
	STOP_SWTMR	0261
	DEL_SWTMR	0262

Scheduling Item type	Scheduling Item Operation	TrackType
	DROP_SWTMR	0263
TRACK_MUX	MUX_CREATE	0512
	MUX_PEND	0513
	MUX_POST	0514
	MUX_DEL	0515
TRACK_SEM	SEM_CREATE	1024
	SEM_PEND	1025
	SEM_POST	1026
	SEM_DEL	1027
TRACK_QUE	QUE_CREATE	2048
	QUE_PEND	2049
	QUE_POST	2050
	QUE_DEL	2051
TRACK_EVENT	EVENT_CREATE	4096
	EVENT_WRITE	4097
	EVENT_READ	4098
	EVENT_CLEAR	4099
	EVENT_DEL	4100

Figure 3-3 shows an example of track scheduling information. You can check the running relationship before and after an exception occurs based on the track scheduling information.

Figure 3-3 Example of track scheduling information



```
*********************track_info**********************
current_item:0x1
item_cnt:0xa
Index    TrackType   TrackID  CurTime  Data1  Data2
0001 0016 0007 0x42c 0x496bd4 0x0
0002 0016 0007 0x424 0x3f5a60 0x0
0003 0016 0007 0x425 0x3f5a62 0x0
0004 0016 0007 0x426 0x3f5a62 0x0
0005 0016 0007 0x427 0x3f5a62 0x0
0006 0016 0007 0x428 0x3f5a60 0x0
0007 0016 0007 0x429 0x3f5a64 0x0
0008 0016 0007 0x42a 0x3f5a62 0x0
0009 0016 0007 0x42b 0x3f5a62 0x0
0010 0064 0009 0x42c 0x0 0x0
```

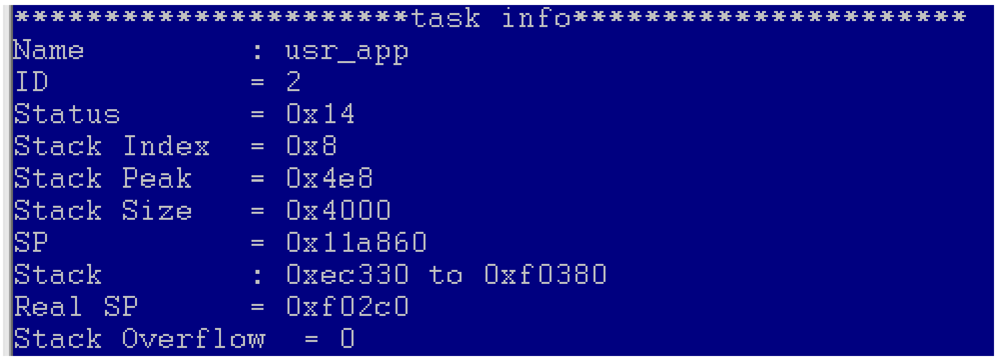
3.2.2.3.6 Task Information

The task information is about task execution when the exception occurs. It includes the following items:

- **Name**: task name
- **ID**: task ID
- **Status**: current task status
- **Stack Peak**: stack peak value
- **Stack Size**: task stack size
- **SP**: stack pointer. It is used to restore the actual stack pointer when the current task needs to be switched to another task. When an exception occurs, the value is switched to the exception stack pointer.
- **Stack**: pointer to the top and bottom of the task stack
- **Real SP**: real stack pointer. It indicates the actual stack pointer when an exception occurs, that is, the stack pointer before the SP switches to the exception stack.
- **Stack Overflow**: stack overflow flag

Figure 3-4 shows an example of task information.

Figure 3-4 Example of task information

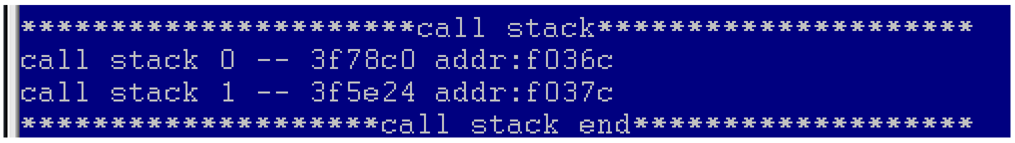


```
**********************task info**********************
Name           : usr_app
ID             = 2
Status         = 0x14
Stack Index    = 0x8
Stack Peak     = 0x4e8
Stack Size     = 0x4000
SP             = 0x11a860
Stack          : 0xec330 to 0xf0380
Real SP        = 0xf02c0
Stack Overflow    = 0
```

3.2.2.3.7 Function Call Stack Information

The function call stack displays all function call instructions related to exceptions. You can check the context of the function call when an exception occurs based on the function call stack to facilitate fault locating. Figure 3-5 shows the function call stack information.

Figure 3-5 Example of function call stack information



```
**********************call stack**********************
call stack 0 -- 3f78c0 addr:f036c
call stack 1 -- 3f5e24 addr:f037c
********************call stack end********************
```