BG96 ThreadX and QAPI User Guide

LTE Module Series

Rev. BG96_ThreadX_and_QAPI_User_Guide_V1.1

Date: 2017-05-03

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Office 501, Building 13, No.99, Tianzhou Road, Shanghai, China, 200233
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/salesupport.aspx

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/techsupport.aspx
Or email to: Support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2017-04-13	Hyman DING/ Sherlock ZHAO	Initial
1.1	2017-05-03	Hyman DING	Deleted the description of firmware upgrade thorough “QPST” tool

Contents

Table Index

Figure Index

1 Introduction

ThreadX is an advanced hard real-time operating system (RTOS) designed specially for deep embedded applications with small-scale, strong real-time performance and high reliability. It supports a large number of processors and SoCs. Combined with its superior ease-of-use, ThreadX is widely used in consumer electronics, automotive electronics, industrial automation, network solutions, military and aerospace, as well as many other fields.

QAPI (Qualcomm API) is the socket API for BG96's ThreadX OS. It is designed to facilitate the development of mobile station-based network applications.

This document mainly introduces how to set up ThreadX compiler environment for BG96 in Windows System, and how to use QAPI to realize customer requirements with ThreadX OS.

2 Compile ThreadX Apps Software

While compiling ThreadX apps software, the host's operating system and compilation tools shown as below are needed.

Table 1: Required Host Operating System and Compilation Tools

Component	Source or Binary-only	Toolchain Required for Building Source	Python Version	Perl Version	Cygwin	Supported Host System
apps_proc	Source	ARM complier tools 5.05 (build 106)	Python 2.7.6	Perl v5.14.2	Cygwin 2.2.1	Linux, Windows 7

The following chapters illustrate the operations on Windows system. For the operations on Linux, please pay attention to the subsequent revisions of this document.

2.1. Setup Windows Build Environment

Before issuing the build commands, certain command environment settings should be set to ensure the correct executable path and toolchain configuration. The specific environment settings vary based on the host's software installation, but it is similar to *myenviron_setting.cmd* script (for Windows), which sets the path to point to the ARM toolchain lib, include, bin, and license file configuration.

```
set COMPILE_TOOLS_PATH=c:\compile_tools
set ARMTOOLS=ARMCT5.05
set LM_LICENSE_FILE=%COMPILE_TOOLS_PATH%\license.dat
set ARMBIN=%COMPILE_TOOLS_PATH%\ARM_Compiler_5\bin
set ARMLIB=%COMPILE_TOOLS_PATH%\ARM_Compiler_5\lib
set ARMPATH=%COMPILE_TOOLS_PATH%\ARM_Compiler_5\bin
set ARMHOME=%COMPILE_TOOLS_PATH%\ARM_Compiler_5
set ARMINC=%COMPILE_TOOLS_PATH%\ARM_Compiler_5\include
set ARMINCLUDE=%COMPILE_TOOLS_PATH%\ARM_Compiler_5\include
set ARMCC5_ASMOPT=--licretry --diag_suppress=9931,9933
set ARMCC5_CCOPT=--licretry --diag_suppress=9931,9933
set ARMCC5_FROMELFOPT=--licretry --diag_suppress=9931,9933
set ARMCC5_LINKOPT=--licretry --diag_suppress=9931,9933
```

```
set PYTHON_PATH=%COMPILE_TOOLS_PATH%\Python27
set PYTHONPATH=%COMPILE_TOOLS_PATH%\Python27
set GNUPATH=%COMPILE_TOOLS_PATH%\cygwin\bin
set CRMPERL=%COMPILE_TOOLS_PATH%\Perl\bin
set PERLPATH=%COMPILE_TOOLS_PATH%\Perl\bin

set PATH=.;%GNUPATH%;%PYTHON_PATH%;%ARMBIN%;%PERLPATH%;%PATH%
```

2.2. Download and Install Associated Tools

This chapter introduces how to download and install associated tools in Windows build environment.

2.2.1. Download ARM Compiler Tool

Step 1: Create an account in the following page: https://silver.arm.com.

Step 2: Open the ARM complier tool download page: https://silver.arm.com/browse.

(1) Click "**Downloads**" → "**Development Tools**" → "**DS-5 Development Studio**", as illustrated below.

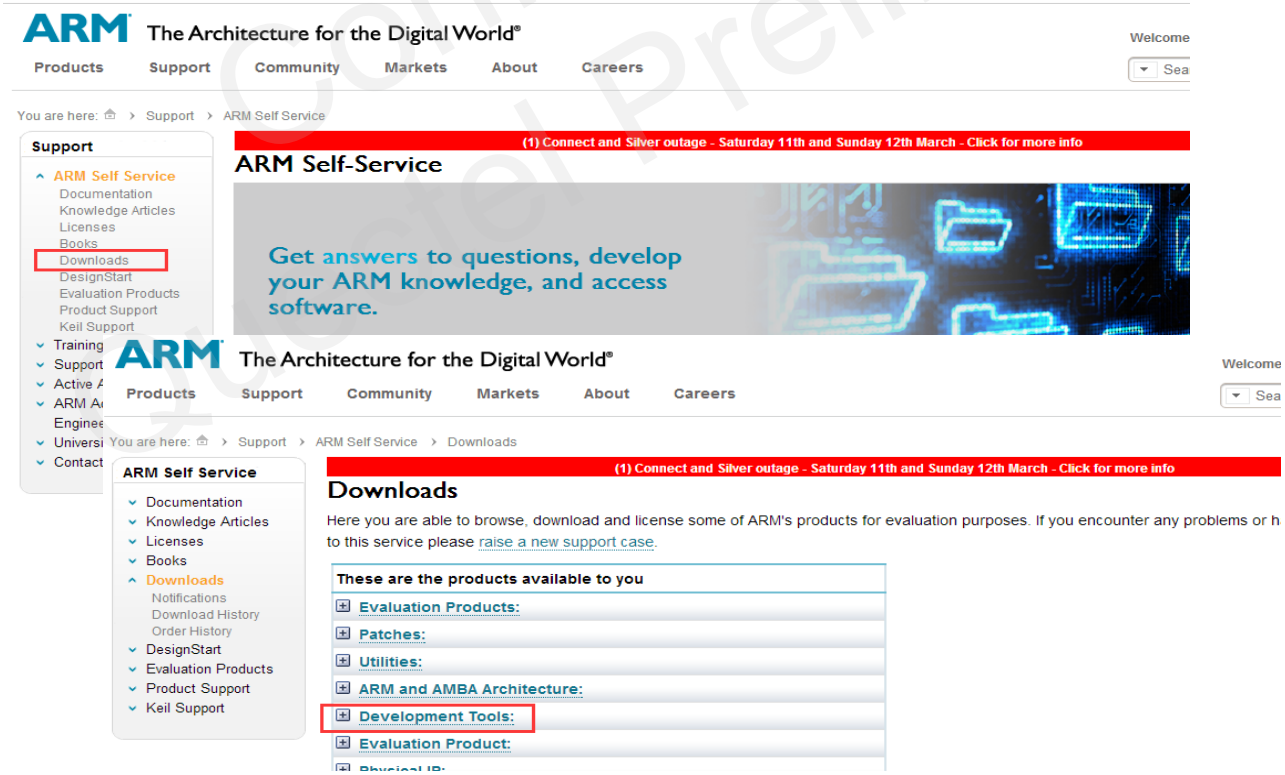

Figure 1: "Downloads" and "Development Tools" Pages

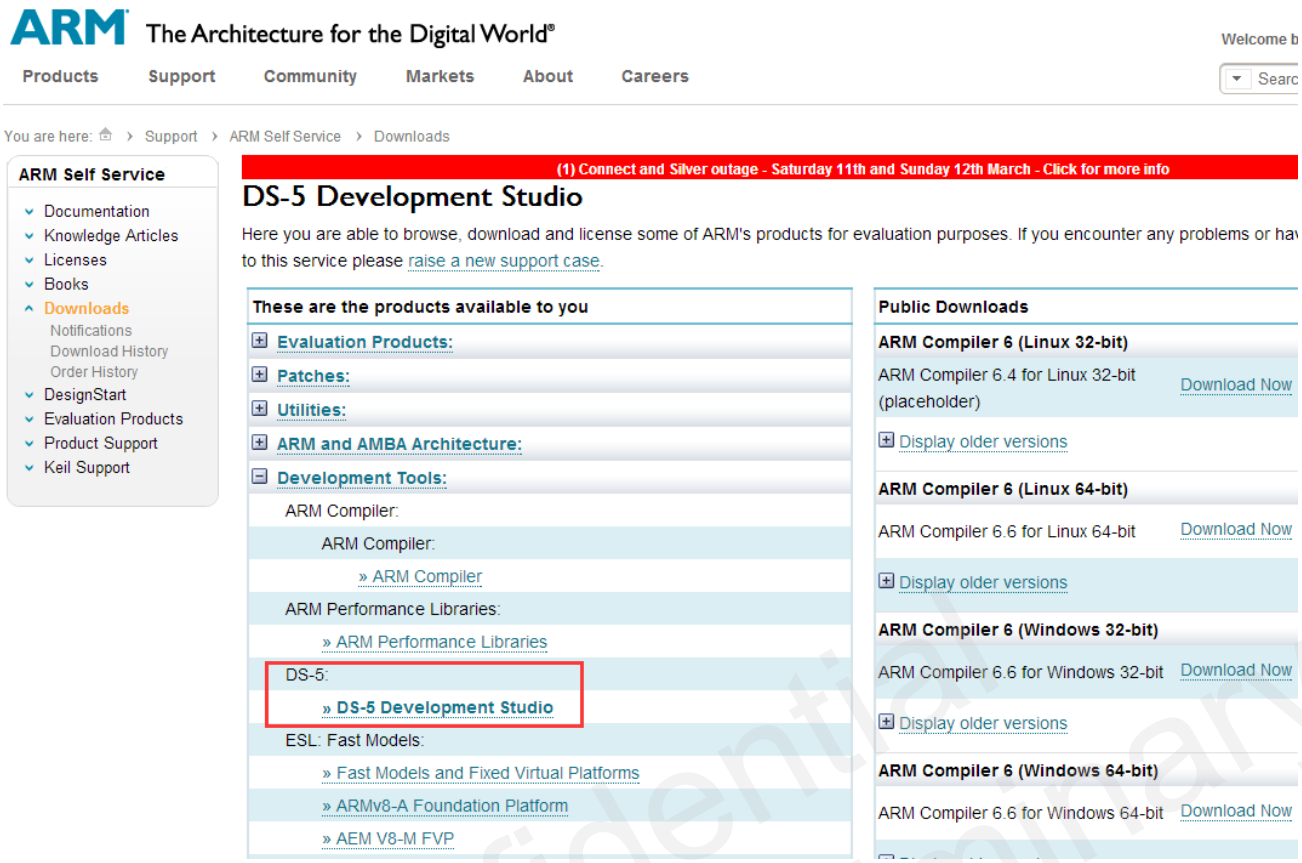

Figure 2: Click "DS-5 Development Studio"

(2) Under "**ARM Compiler 5 (Windows)**", click the "**Download Now**" button after "**ARM Compiler 5.05 update 1 (build 106) for Windows**" to download the corresponding ARM compiler tool for Windows.

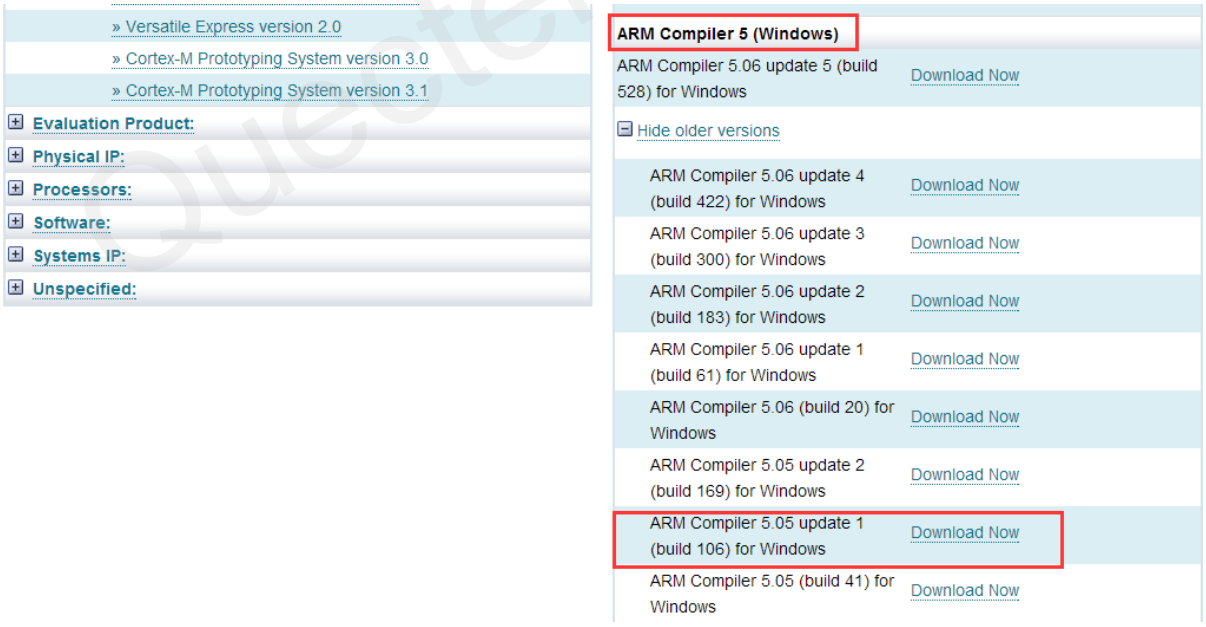

Figure 3: Download the Corresponding Tool

(3) After clicking "**Download Now**", there is a need to confirm the details shown as below:

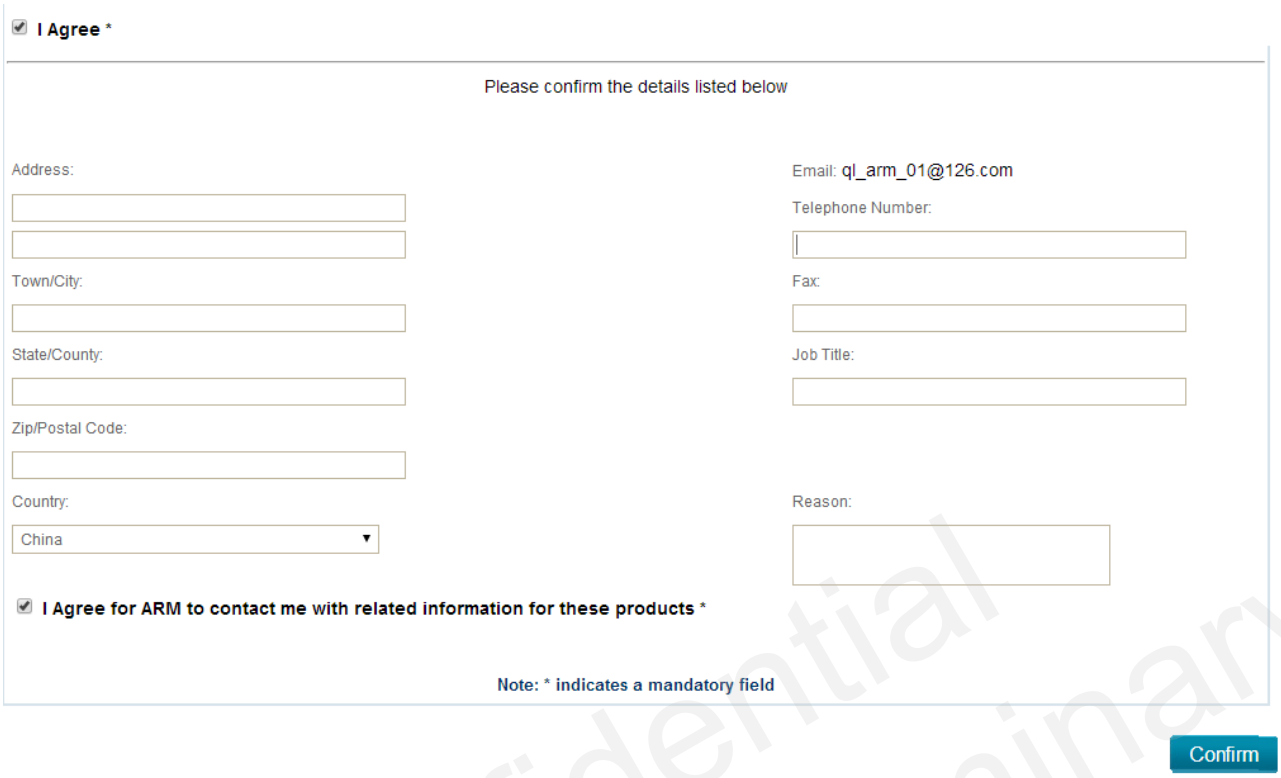

Figure 4: Confirmation of Details

Finally click "Confirm" button and then the tool packet will be downloaded.

2.2.2. Install ARM Compiler Tool

Please follow the steps illustrated in the figures below to finish installation of ARM compiler tool.

Figure 5: Click "Next" for ARM Complier 5 Setup

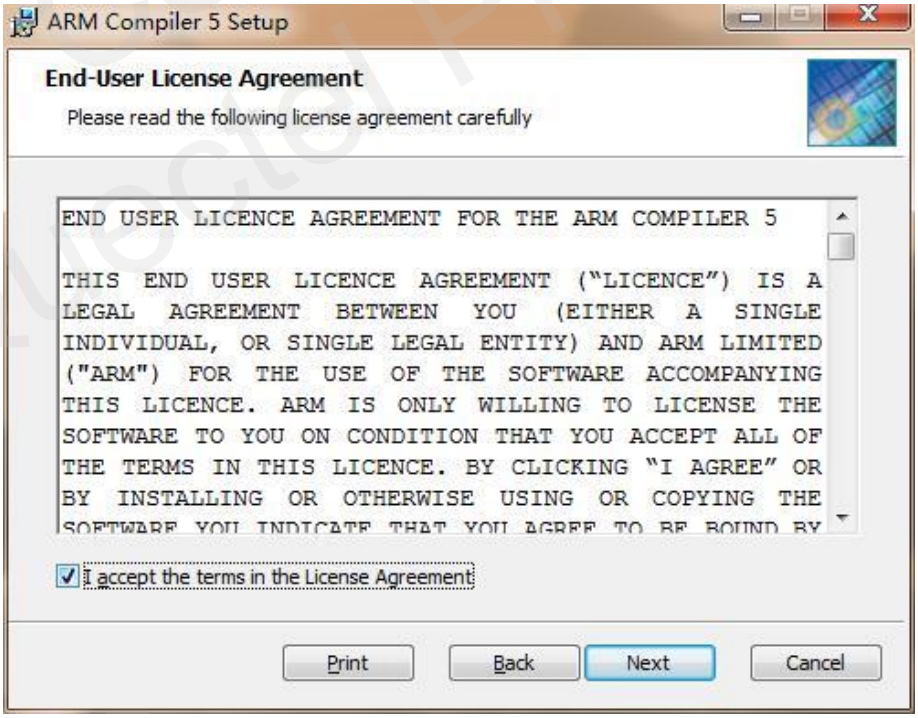

Figure 6: Accept End-User License Agreement of ARM Compiler 5 and Then Click "Next"

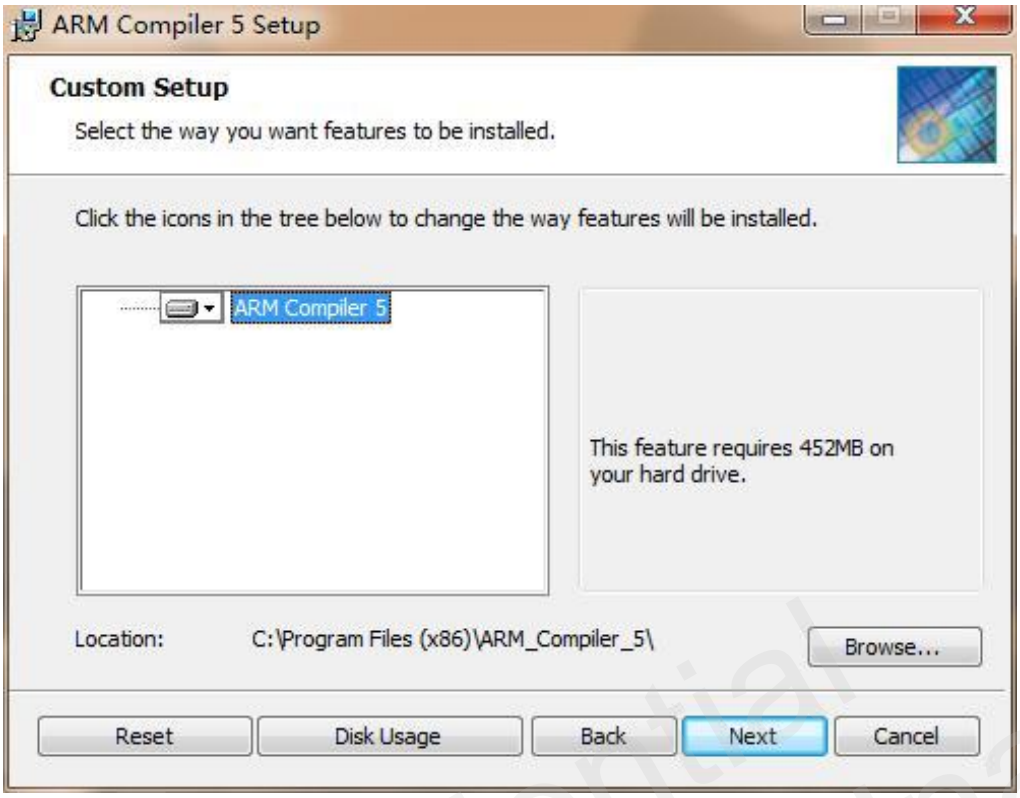

Figure 7: Custom Setup for ARM Complier 5

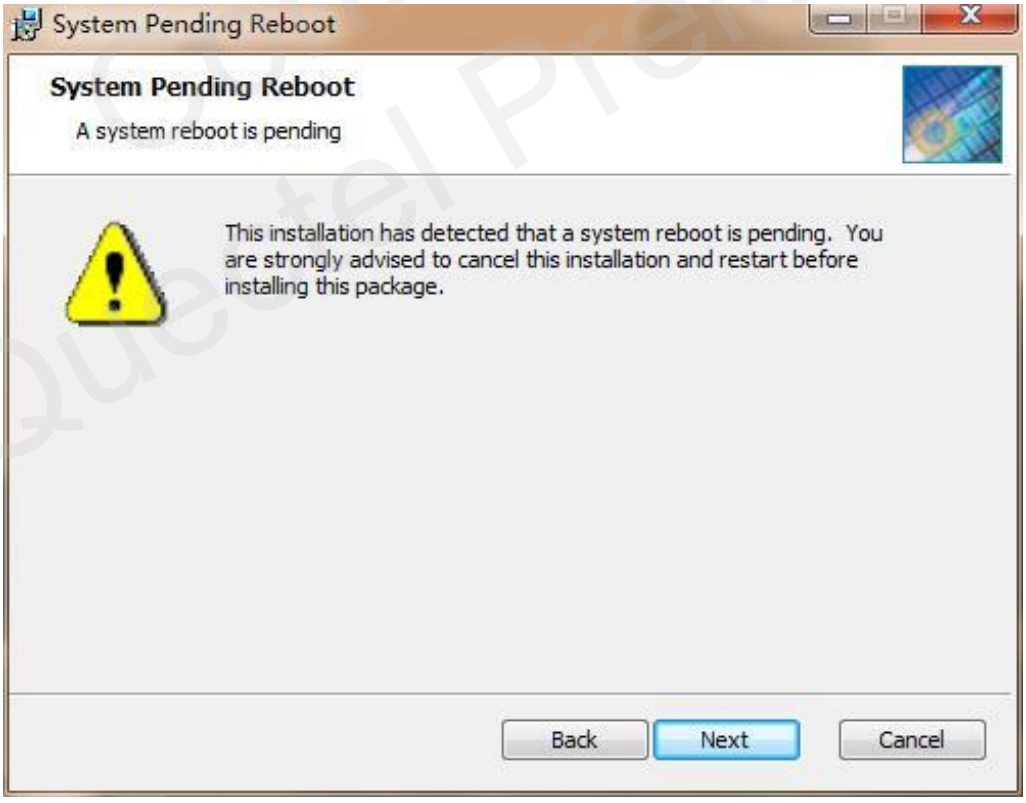

Figure 8: Click "Next" When "System Pending Reboot" Occurs

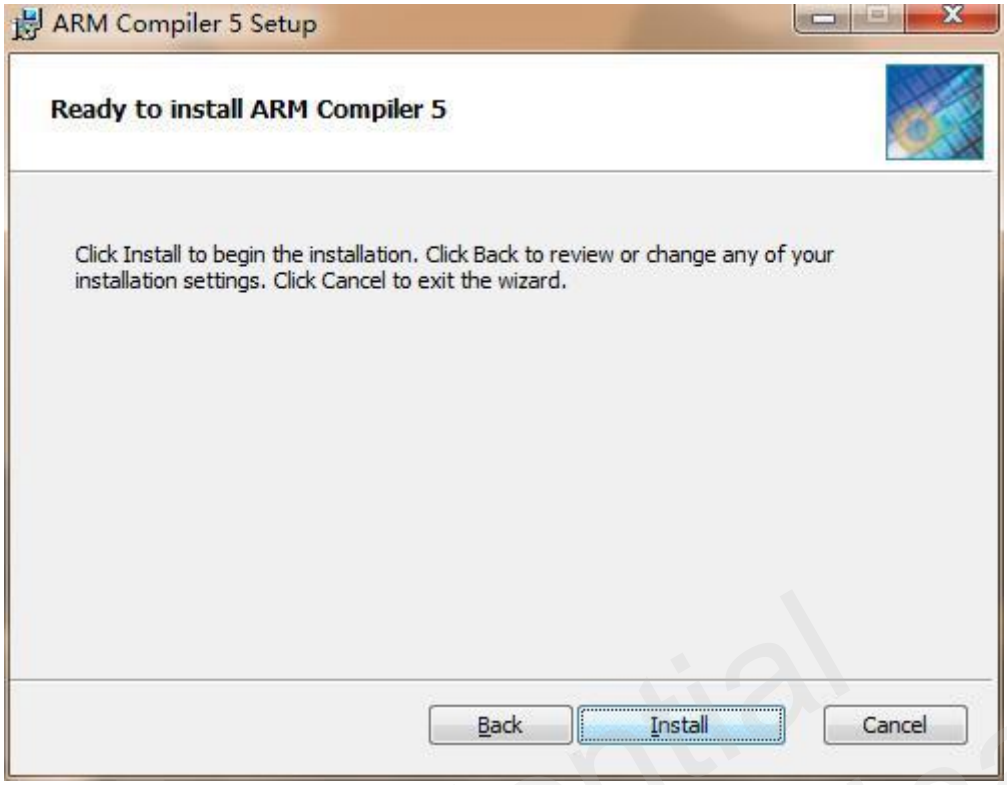

Figure 9: Click "Install" to Begin Installation of ARM Compiler 5

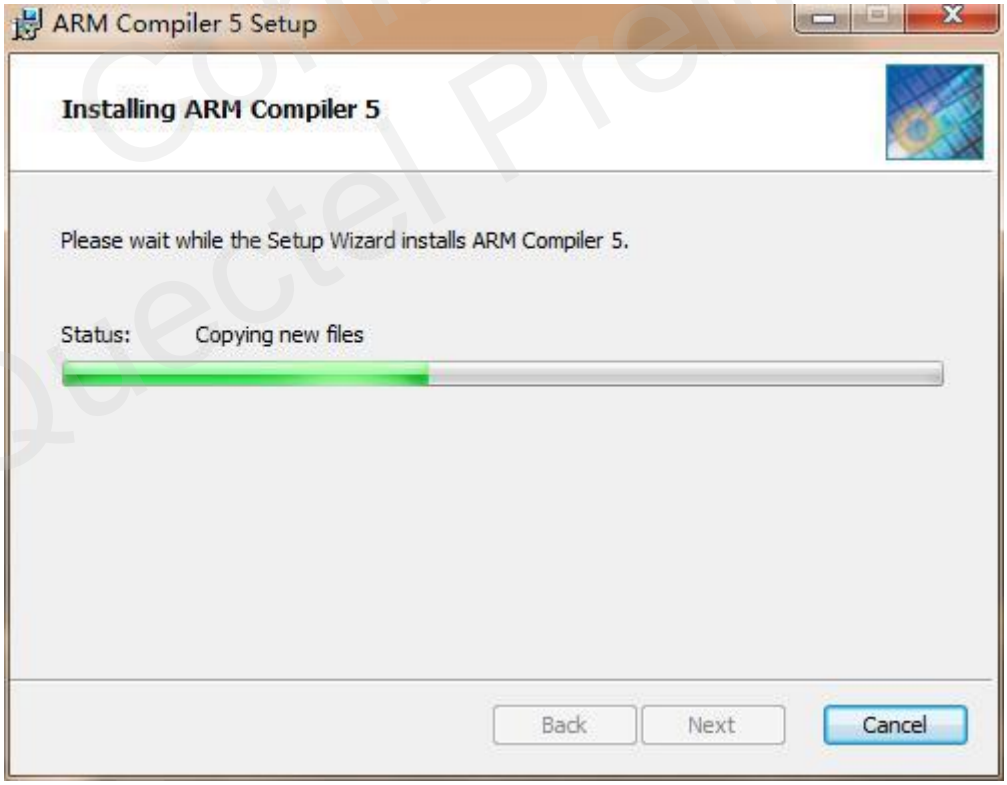

Figure 10: Wait While "Installing ARM Compiler 5"

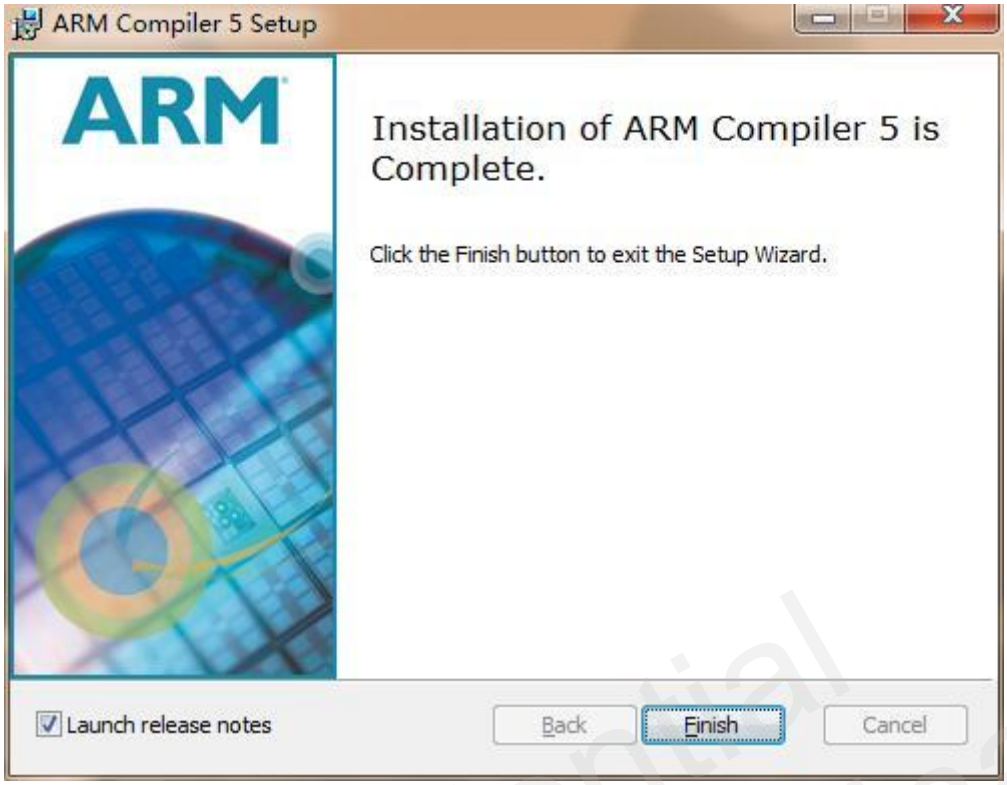

Figure 11: Finish Installation of ARM Complier 5

2.2.3. Download Python

Open the Python download page: https://www.python.org/downloads/windows, and then download the corresponding version.

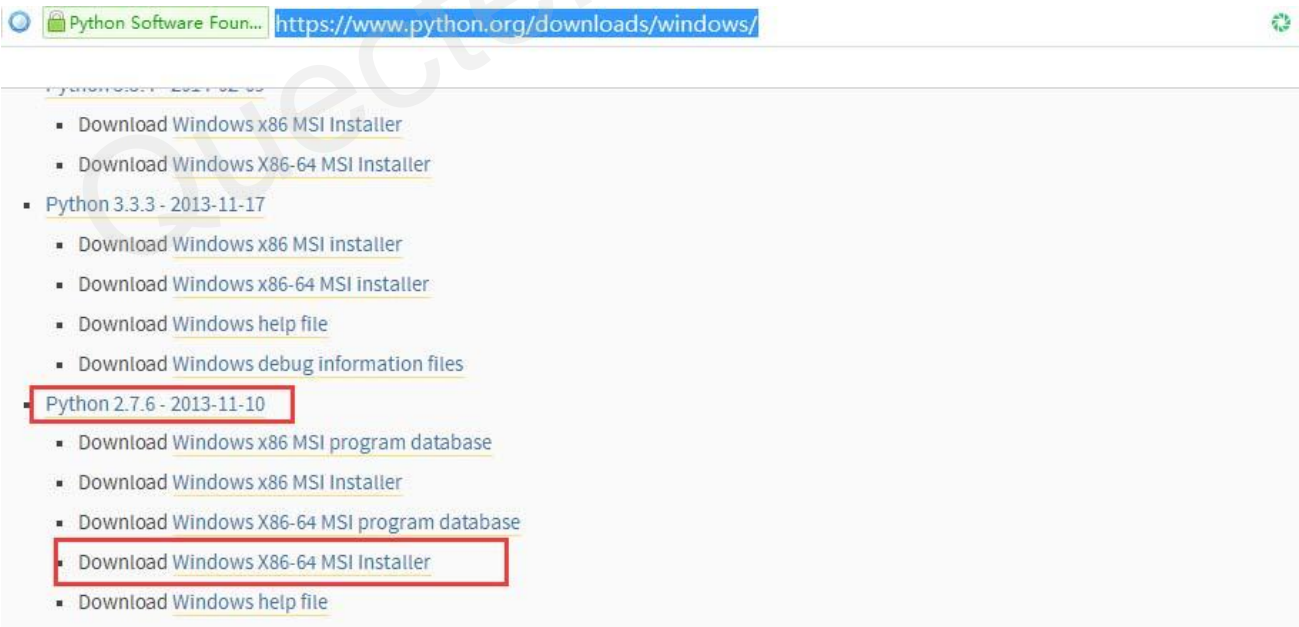

Figure 12: Download Python

2.2.4. Install Python

Please follow the default steps illustrated in the figures below to finish installation of Python.

Figure 13: Select "Install for all users" during Setup

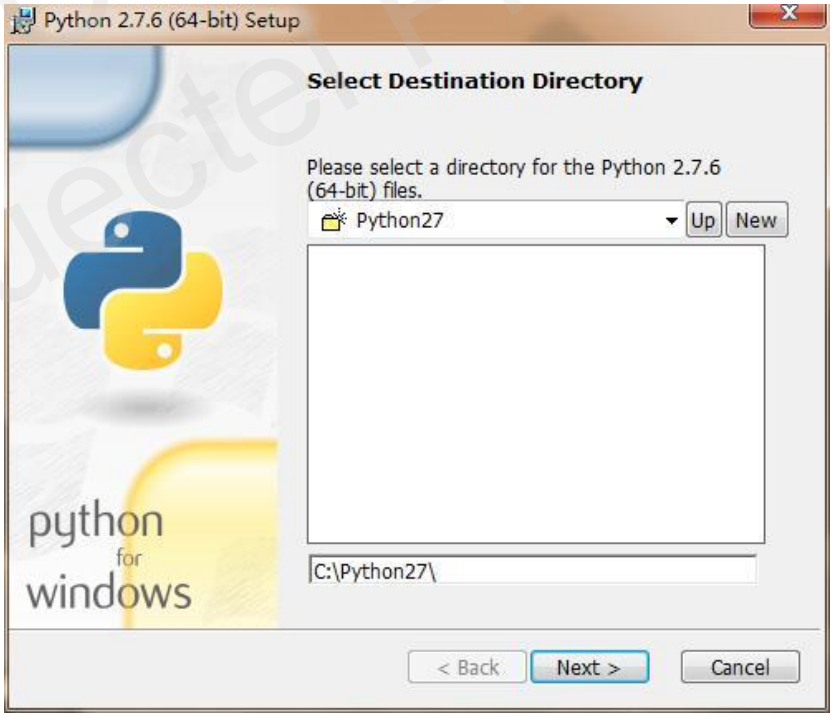

Figure 14: Select Destination Directory for Python

Figure 15: Customize Python

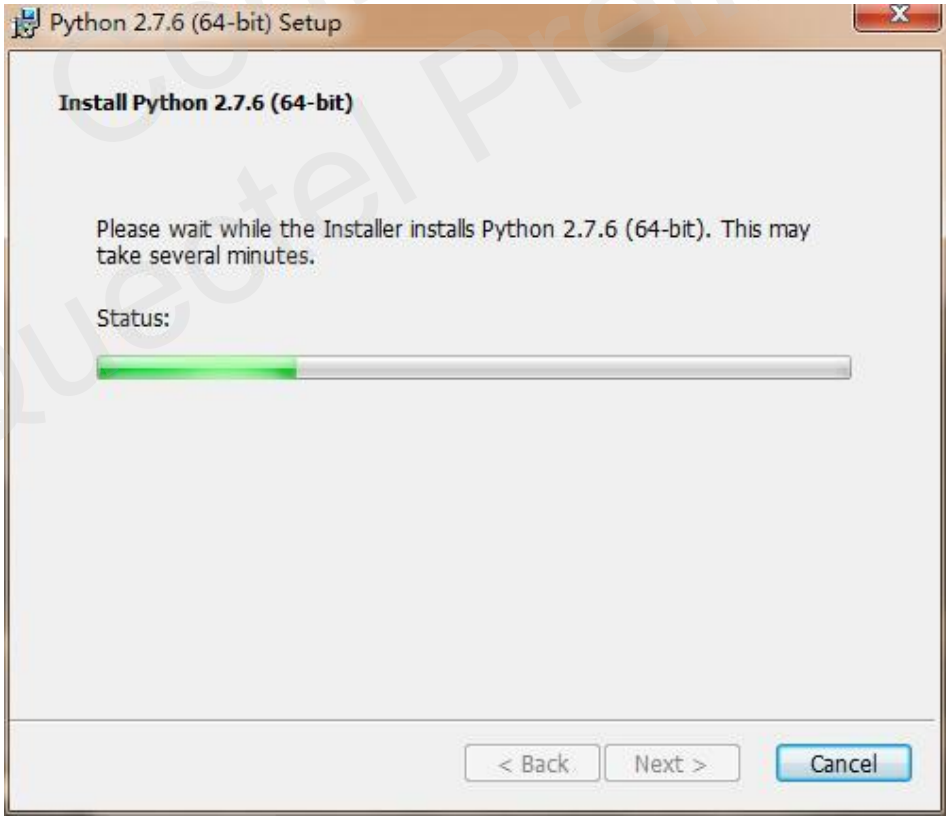

Figure 16: Wait While Installing Python

Figure 17: Finish Installation of Python

2.2.5. Download Pywin32

PyWin32 was designed to be a python extension for Windows. It provides access to the Win32 API and has the ability to create and use COM objects, and the Pythonwin environment.

Step 1: Open the Pywin32 download page shown as below to download the corresponding version for Windows: http://www.softpedia.com/get/Programming/Other-Programming-Files/PyWin32.shtml.

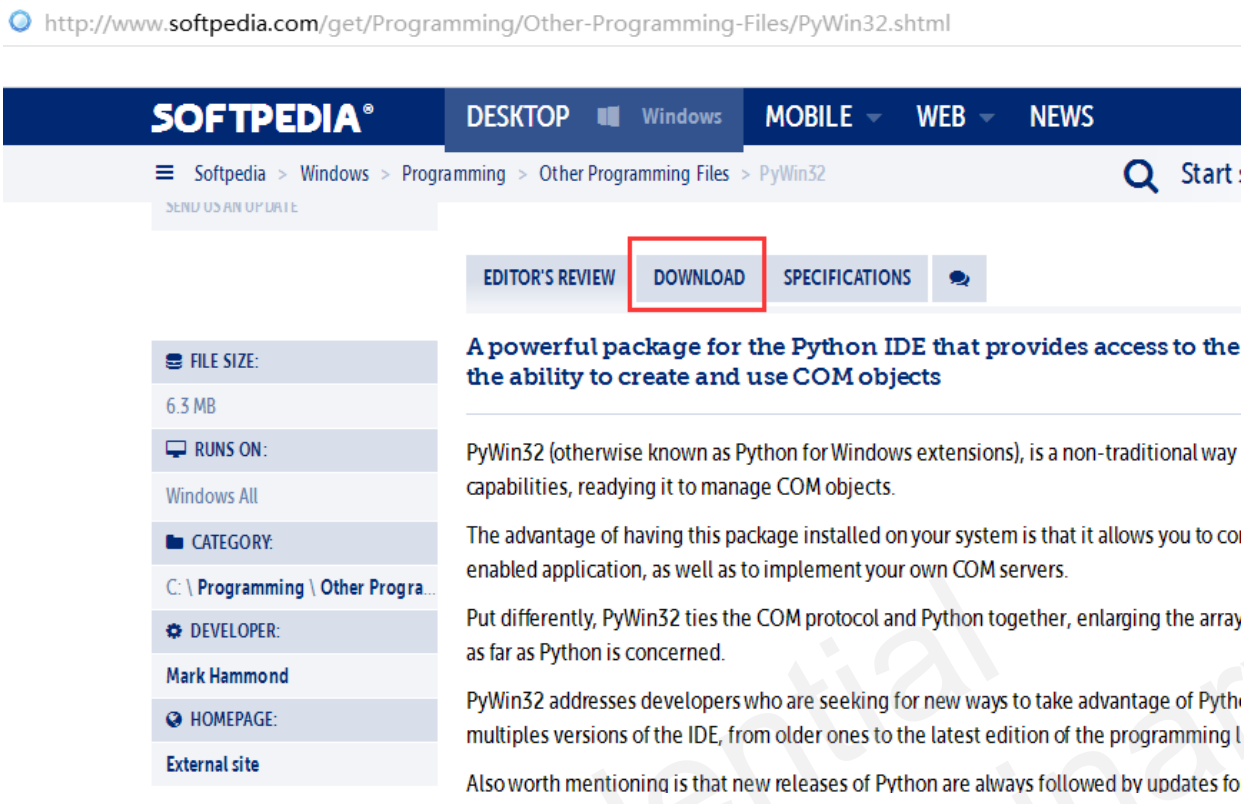

Figure 18: PyWin32 Downloading Page

Step 2: Click "**DOWNLOAD**" to enter into the following interface, and then select the right revision of Pywin32 that is required.

Figure 19: Corresponding Revision of PyWin32 to be Downloaded

Step 3: Click "**CLICK TO START IT MANUALLY**" to start downloading.

Figure 20: Start Downloading Pywin32

2.2.6. Install Pywin32

Please follow the default steps illustrated in the figures below to finish installation of Pywin32.

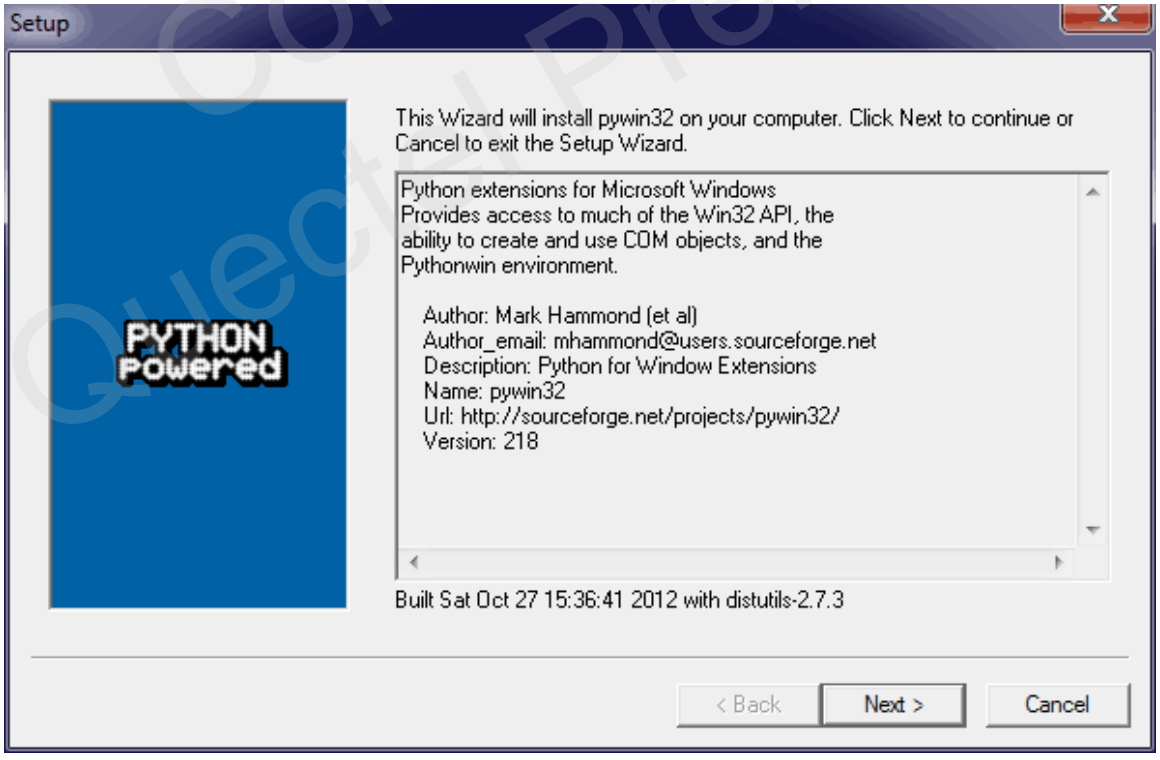

Figure 21: Click "Next" to Continue the Pywin32 Setup Wizard

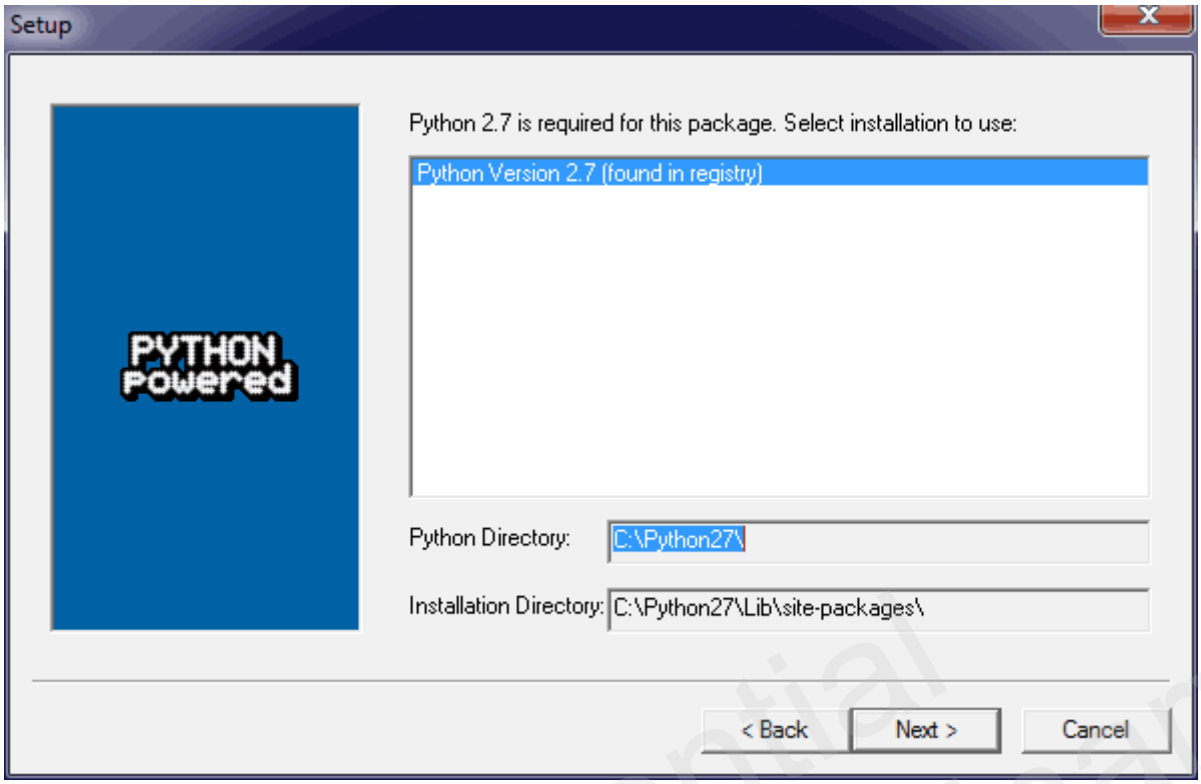

Figure 22: Select Required Python Revision

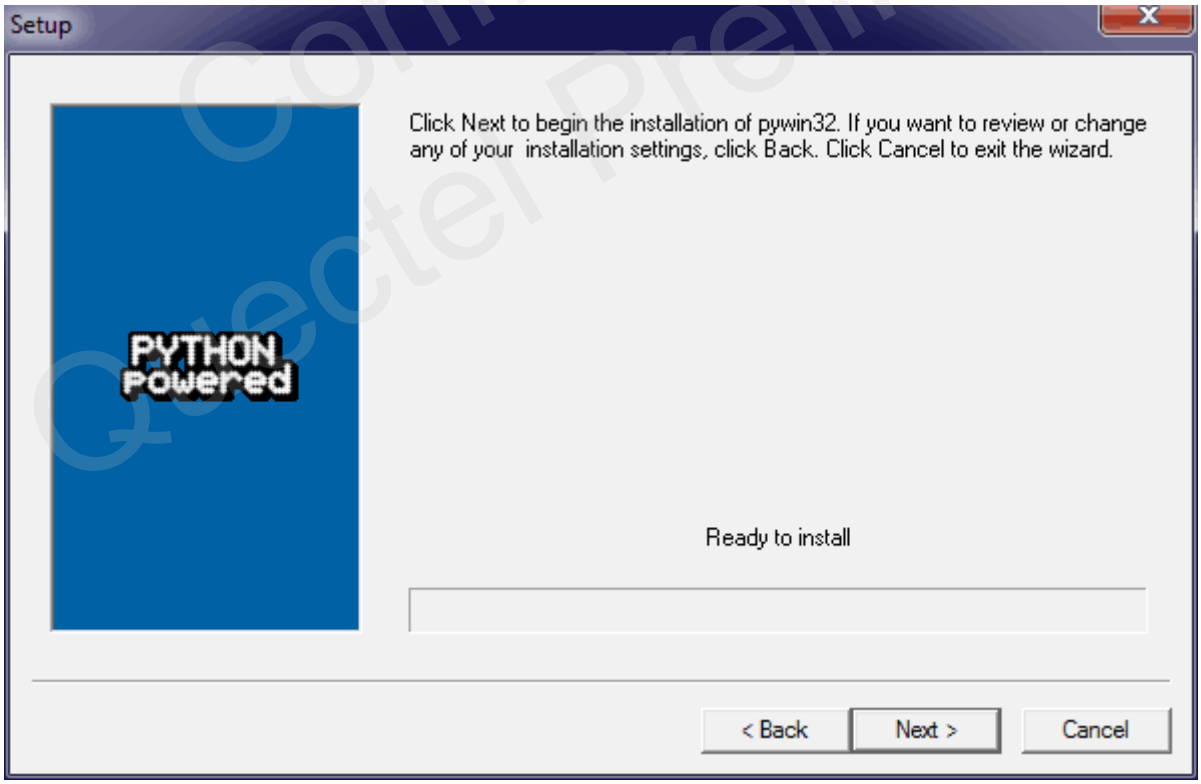

Figure 23: Click "Next" to Begin Installation of Pywin32

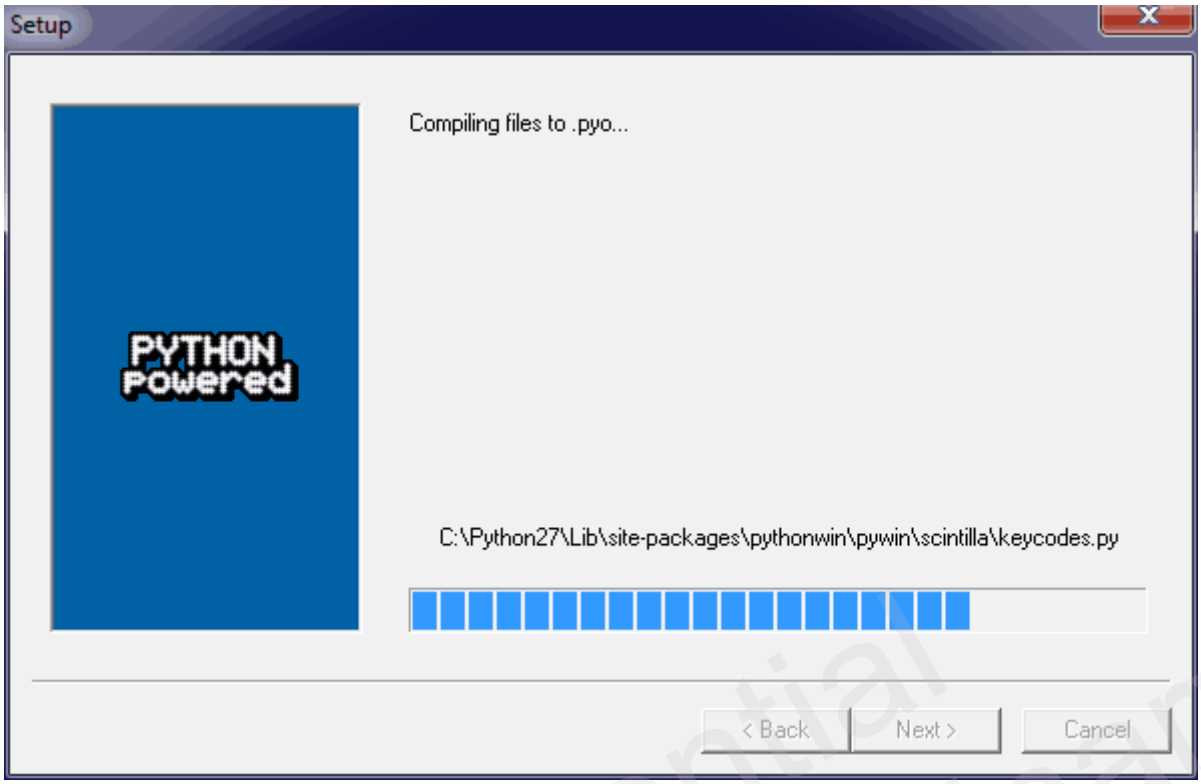

Figure 24: Wait While Installing Pywin32

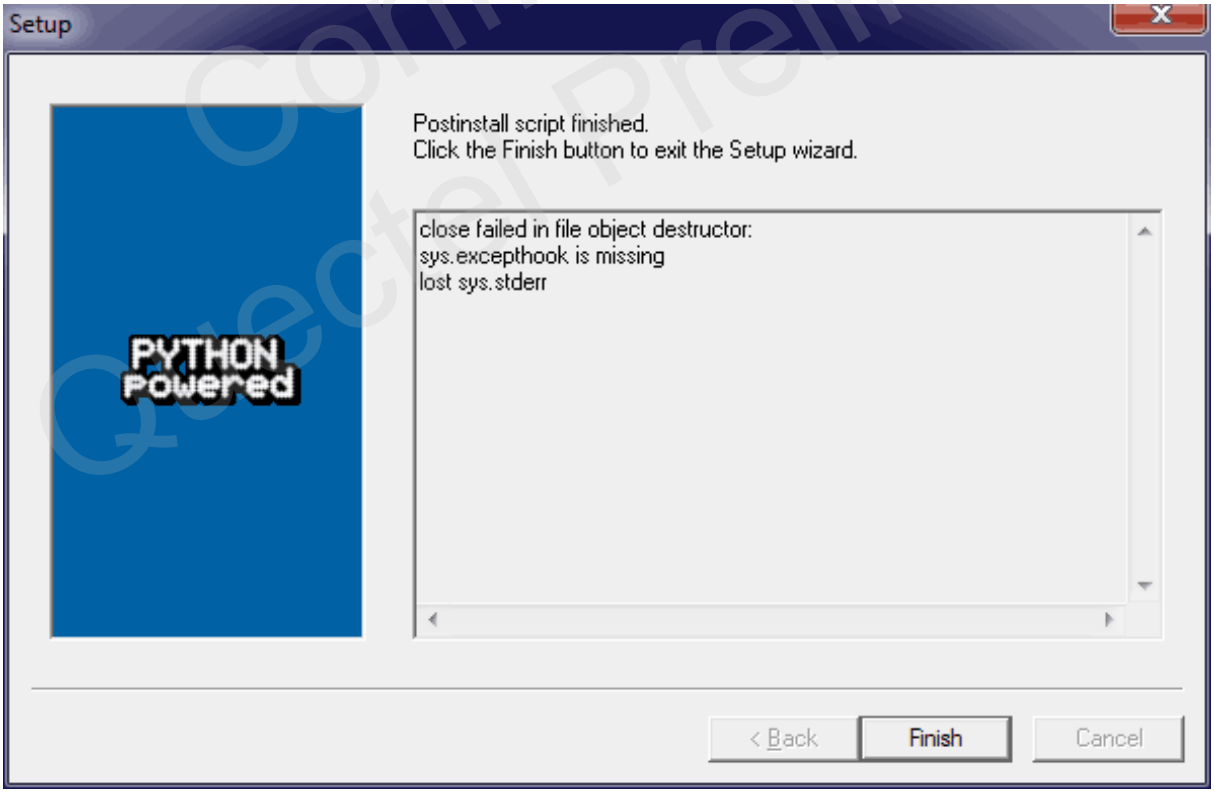

Figure 25: Finish Installation of Pywin32

2.2.7. Download Perl

Open the Perl download page shown as below to download the corresponding revision ("**ActiveState Perl**") for Windows: https://www.perl.org/get.html

Figure 26: Perl Download Page

2.2.8. Install ActivePerl

Please follow the default steps illustrated below to finish installation of ActivePerl.

Figure 27: Click "Next" to install ActivePerl

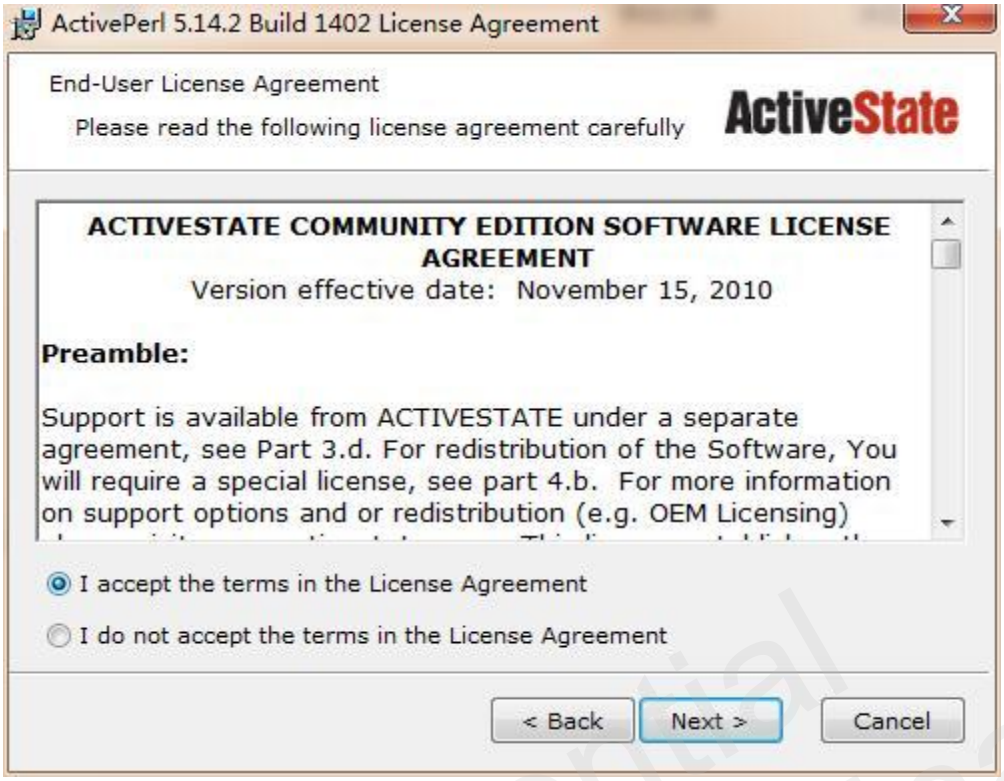

Figure 28: Accept End-User License Agreement of ActivePerl and The Click "Next"

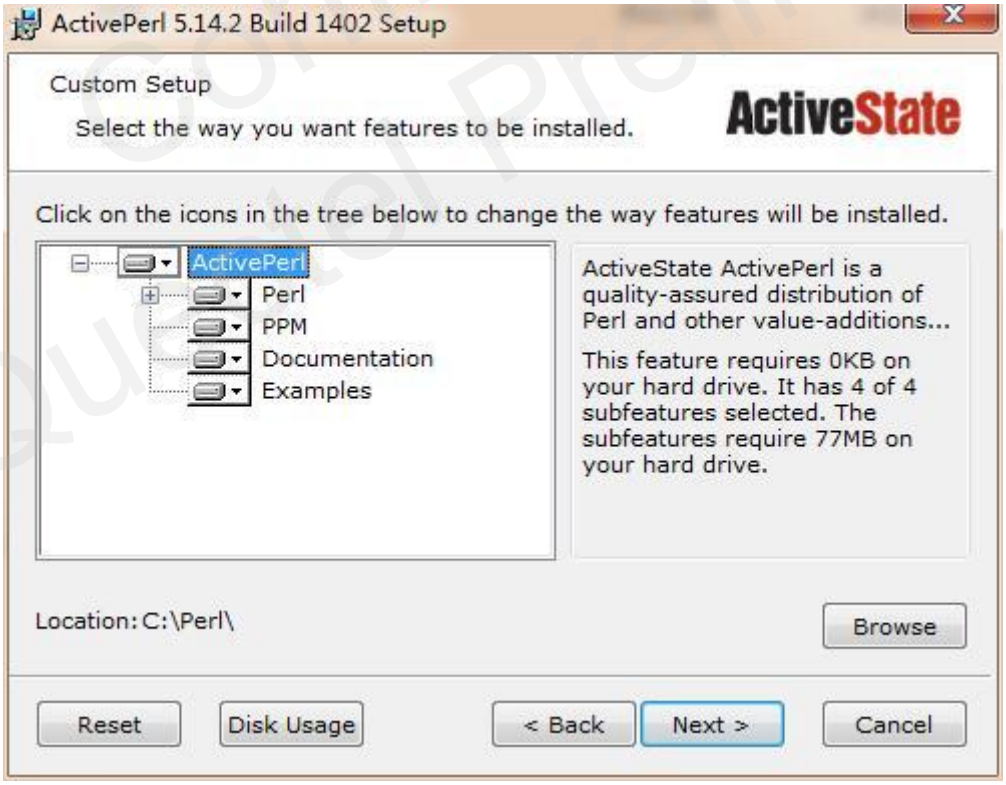

Figure 29: Custom Setup of ActivePerl

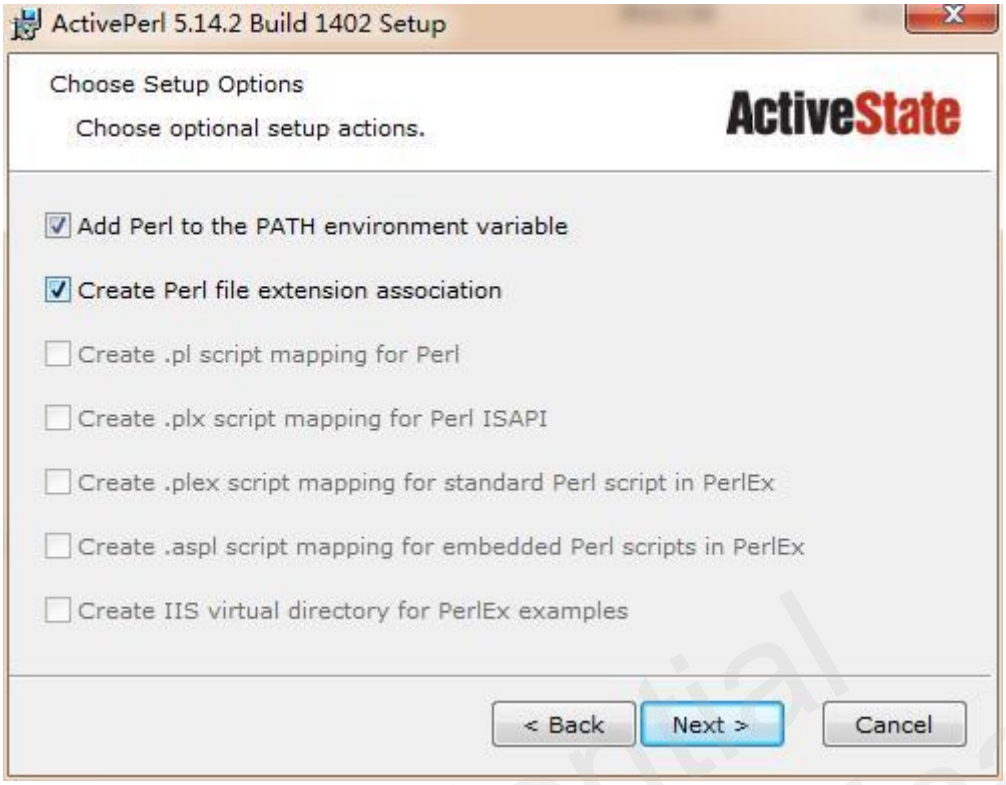

Figure 30: Choose Setup Options for ActivePerl

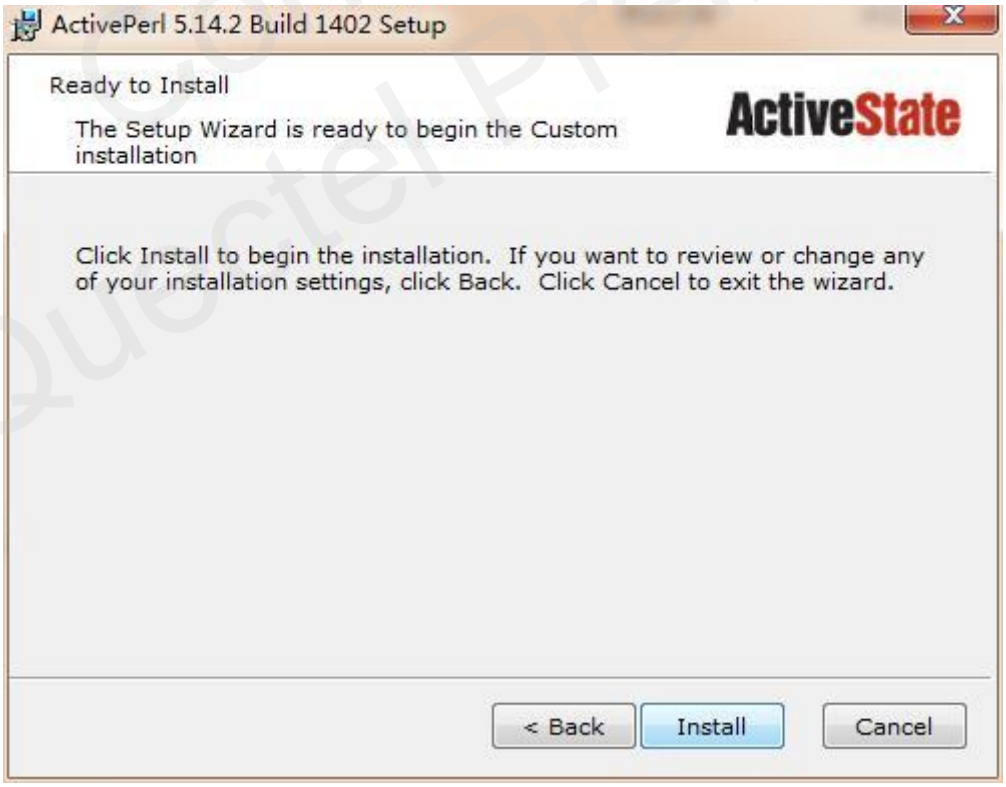

Figure 31: Click "Install" for Installation of ActivePerl

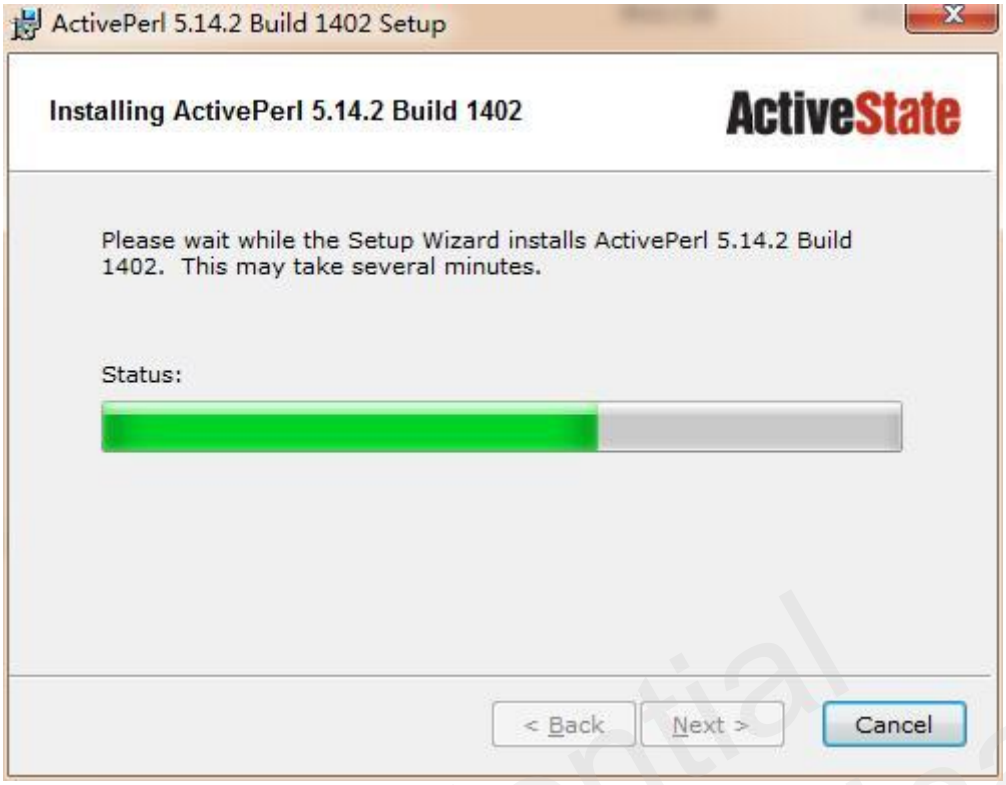

Figure 32: Wait While Installing ActivePerl

Figure 33: Finish Installation of ActivePerl

2.2.9. Download Cygwin

Open the Cygwin download page shown as below to download the corresponding revision of Cygwin for Windows: https://cygwin.com/install.html.

2.2.10. Install Cygwin

Install Cygwin by default steps.

2.3. Check Versions of Associated Tools

Please use following commands to check versions of associated tools before building ThreadX apps images.

```
Check Python version:
Python --version

Check Perl version:
Perl --version

Check Cygwin version:
Cygchek --c cygwin
```

2.4. Build ThreadX Apps Images

Use correct tool versions (refer to ***Table 1***) and set export paths appropriately.

1. Open a command prompt and change to *apps_proc\build\ms* directory by the following command :

```
cd apps_proc\build\ms
```

Use the appropriate commands listed in the table below based on the build environment.

Table 2: Build Environment and Corresponding Build Commands

Build Environment	Build Command
Linux	Build images: ./build.sh apps_images BUILD_ID=ACINAAAZ Clean the Build:

	./build.sh apps_images BUILD_ID=ACINAAAZ -c
Windows	Build images: build.cmd apps_images BUILD_ID=ACINAAAZ Clean the Build: build.cmd apps_images BUILD_ID=ACINAAAZ -c

Quectel provides a software packet which includes environment setting script and build script for Windows build environment, customers could use commands below to build or clean operations on Windows:

```
Build images:
build_app.bat new

Clean the build:
build_app.bat clean
```

3 QAPI Application

QAPI is designed to facilitate the development of mobile station-based network applications. For details about QAPI application, please refer to *dss_and_socket_api_for_mdm9x06_threadx_os.pdf* document.

4 Start an Application

The basic executable element in ThreadX is a thread. RCInit framework is supported to automate thread creation and startup, and customers can also choose to dynamically spawn threads. Please pay attention to the following matters before creating a new task:

1. ThreadX supports threads that are created with different priorities. To better control and manage priorities from a single place, RCInit allows the OEMs to ensure that the task inter-dependencies are met. In principle, lower the value of priority, higher is the priority of the task (0 is the highest priority and 255 is the lowest priority).
2. All the tasks, irrespective of it being created by RCInit or dynamically with absolute priority, should register the task with RCInit to get the priority from RCInit.
3. If any task is created dynamically with priority relative to any other tasks, the priority of the original task can be retrieved from RCInit using the above method and then perform the arithmetic to get the priority of the new task.

4.1. Create a Task Using RCinit

A separate application mainly includes source directory, header directory and build directory.

The configuration of tasks will be done in the *SCONS* configuration file. The path of the *SCONS* file is: *<Build>..\<your path>\build* (Build folder corresponding to the *src* file).

Example: *<your path>\dog\build* for dog_task (Build is in the same level as *src* directory).

Example of how to create a new task and sample codes for task definition are shown as below:

1. Add the following codes to the *.scons* file in the build directory *apps_proc\example\task_demo\build\task_demo.scons*, which is at the same level of customers' source directory.

```
RCINIT_IMG = ['APPS_PROC']
env.AddRCInitTask(
    RCINIT_IMG,
        {
            'sequence_group'        : 'RCINIT_GROUP_7',             # required
            'thread_name'           : 'task_demo_daemon',           # required
```

```
            'stack_size_bytes'         : '8192',                                  # required
            'thread_entry'             : 'quectel_task_demo_entry',        # opt
            'priority_amss_order'      : 'SHARED_BACKGROUND_PRI_ORDER', # priority
            'cpu_affinity'             : 'REX_ANY_SMT_MASK',
            'thread_type'              : 'RCINIT_TASK_QURTTASK',     # required
        })
```

2. The corresponding task definition should be in the *c* source file of the source directory: app_proc\example\task_demo\src\task_demo.c

```
int quectel_task_demo_entry(dword ignore)
{
    //Initialization is recommended to be non-blocking.
    …

    /* Start task must call rcinit_handshake */
    rcinit_handshake_startup();       /* Required */

    //Task codes.
    …
}
```

NOTES

1. All tasks and initialization functions are divided into 8 groups (Group 0 to Group7).
2. Every task (other than rcinit_task and rcinit_worker tasks) should fall into one of these groups.
3. The tasks are created using the QuRT services.
4. Application tasks with dependency on TMC are recommended to be in Group 7. Application tasks with no dependency on TMC can be in Group 6.
5. TMC cannot be used to create task in this solution.

4.2. Add a Timer in the Task

The rapid response to asynchronous external events is the most important feature of real-time embedded systems. However, many programs must perform the corresponding actions in a predetermined time interval. BG96 provides two types of timer, one is called “one-shot timer”, and the other is “periodic timer”. As the name implies, “one-shot timer” will expire once after setting and startup, but “periodic timer” will run periodically as long as the timer is initialized once, and no need for additional timer programming.

A timer application mainly includes two parts: timer setting and callback function. Generally, customers just call a timer API to define and initialize a timer directly, and then add custom callback for timer. The commonly used timer APIs are as follows:

1. Define and initialize a timer

```
timer_error_type timer_def_osal
(
  /* Timer to define */
  timer_ptr_type                timer,

  /* Group timer will belong to (NULL = the "NULL" timer group) */
  timer_group_ptr               group,

  /*ats timer callback type*/
  timer_notify_type             cb_type,

  /* Task to signal and/or task to queue APC call for */
  time_osal_notify_obj_ptr      sigs_func_addr,

  /* Task signals to set to the tcb task when timer expires */
  time_osal_notify_data         sigs_mask_data
)
```

Customers can register their own callback function for specified timer ID which indicates different timers. When this timer expires, customers can also send data to callback with “sigs_mask_data” parameter via this interface.

2. Sets an inactive or active timer to expire

Two function interfaces can be used to set an inactive or active timer to expire, shown as below:

```
timer_error_type timer_set_64
(
  /* Timer to set */
  timer_ptr_type                timer,

  /* Time (in units below) until first timer expiry */
  time_timetick_type            time,

  /* Period (in units) between repeated expiries (0 = not periodic) */
  time_timetick_type            reload,

  /* Unit to measure "ticks" in. */
  timer_unit_type               unit
)

void timer_set
```

```
(
  timer_ptr_type                timer,
  timetick_type                 time,
  timetick_type                 reload,
  timer_unit_type               unit
)
```

In fact, there are six types of “timer unit”, which are hour, minute, second, millisecond, microsecond and tick. Customers need to configure different timer unit types according to the actual application scenario. Default “timer_uint_type” is ticks.

3. User timer callback function

If the timer expires, the callback function will be executed. Customers must design their own callback function in *C* source codes.

At the end of this section, customers may know which interface is needed to add a timer into the task. Then please use timer_def_osal() to initialize a timer, and call timer_set() or timer_set_64() in the right place of the task to start the timer.

NOTE

Only if “timer_uint_type” is ticks, the clock frequency of timer_set() and timer_set_64() is different. timer_set() ticks are supplied in 32kHz domain and timer_set_64() ticks are supplied in 19MHz domain.

4.3. Use a Signal in the Task

ThreadX a multitasking operating system, it is common for multiple tasks to exist at the same time. User tasks often need to share data and exchange information with each other. A signal is one of the most simple method for communication provided by ThreadX and it is used to indicate the occurrence of an event.

A signal can suspend the current thread until either the specified signals are set or the specified timeout interval is exceeded. If a thread is waiting on a signal object for any of the specified set of signals to be set, and one or more of those signals is set in the signal object, then the thread is awakened. If a thread is waiting on a signal object for all of the specified set of signals to be set, and all of those signals are set in the signal object, then the thread is awakened.

The following shows the usage scenario of how to use a signal in the task:

There are two tasks, task A and task B. Task A must be scheduled after another one has completed some works. In this case, customers can wait for a specified signal object in task A, if task B has completed his

work, task B needs to set the specified signal. Task A will detect whether this signal is set and then continue to do his own works.

Sample codes are shown as follows:

Task A:

```
int quectel_taskA_demo_entry(dword ignore)
{
    …
    /* Initialize the qurt signal object */
    qurt_signal_init(&task_sig_cb_obj);

    …
    while (1)
    {
        …
        /* Wait on the signal object */
        qurt_signal_wait(&task_sig_cb_obj,
                         TASK_DEMO_SIG_MASK,
                         QURT_SIGNAL_ATTR_WAIT_ANY);
        qurt_signal_clear(&task_sig_cb_obj, TASK_DEMO_SIG_MASK);
        //Do some works.
    }
}
```

Task B:

```
…
{
    …
    /* set signal object bit with mask value */
    qurt_signal_set(&task_sig_cb_obj, TASK_DEMO_SIG_MASK);
}
…
```

NOTE

For detailed example codes, please refer to *example* directory in SDK package.

5 Appendix A Reference

Table 3: Terms and Abbreviations

Abbreviation	Description
App	Application
API	Application Programming Interface
MDM	Mobile Device Management,
OEM	Original Equipment Manufacturer
OS	Operating System
SDK	Software Development Kit
SoC	System on Chip
TMC	Task Management Component
USB	Universal Serial Bus