Amazon Kinesis Video Streams Producer on AmebaPro – Getting Started Guide

Realtek Semiconductor Corp.
No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211. Fax: +886-3-577-6047
www.realtek.com

COPYRIGHT

DISCLAIMER

Please Read Carefully:

Realtek Semiconductor Corp., (Realtek) reserves the right to make corrections, enhancements, improvements and other changes to its products and services. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

Reproduction of significant portions in Realtek data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Realtek is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions.

Buyers and others who are developing systems that incorporate Realtek products (collectively, “Customers”) understand and agree that Customers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Customers have full and exclusive responsibility to assure the safety of Customers' applications and compliance of their applications (and of all Realtek products used in or for Customers’ applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Customer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Customer agrees that prior to using or distributing any applications that include Realtek products, Customer will thoroughly test such applications and the functionality of such Realtek products as used in such applications.

Realtek’s provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation kits, (collectively, “Resources”) are intended to assist designers who are developing applications that incorporate Realtek products; by downloading, accessing or using Realtek’s Resources in any way, Customer (individually or, if Customer is acting on behalf of a company, Customer’s company) agrees to use any particular Realtek Resources solely for this purpose and subject to the terms of this Notice.

Realtek’s provision of Realtek Resources does not expand or otherwise alter Realtek’s applicable published warranties or warranty disclaimers for Realtek’s products, and no additional obligations or liabilities arise from Realtek providing such Realtek Resources. Realtek reserves the right to make corrections, enhancements, improvements and other changes to its Realtek Resources. Realtek has not conducted any testing other than that specifically described in the published documentation for a particular Realtek Resource.

Customer is authorized to use, copy and modify any individual Realtek Resource only in connection with the development of applications that include the Realtek product(s) identified in such Realtek Resource. No other license, express or implied, by estoppel or otherwise to any other Realtek intellectual property right, and no license to any technology or intellectual property right of Realtek or any third party is granted herein, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Realtek products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of Realtek Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from Realtek under the patents or other Realtek’s intellectual property.

Realtek’s Resources are provided “as is” and with all faults. Realtek disclaims all other warranties or representations, express or implied, regarding resources or use thereof, including but not limited to accuracy or completeness, title, any epidemic failure warranty and any implied warranties of merchantability, fitness for a particular purpose, and non-infringement of any third party intellectual property rights.

Realtek shall not be liable for and shall not defend or indemnify Customer against any claim, including but not limited to any infringement claim that related to or is based on any combination of products even if described in Realtek Resources or otherwise. In no event shall Realtek be liable for any actual, direct, special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of Realtek’s Resources or use thereof, and regardless of whether Realtek has been advised of the possibility of such damages. Realtek is not responsible for any failure to meet such industry standard requirements.

Where Realtek specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Customers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may not use any Realtek products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death. Such equipment includes, without limitation, all medical devices identified by the U.S.FDA as Class III devices and equivalent classifications outside the U.S.

Customers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Customers' own risk. Customers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

Customer will fully indemnify Realtek and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's non-compliance with the terms and provisions of this Notice.

TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

USING THIS DOCUMENT

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

1 AmebaPro RTL8715AD Board

1.1 AmebaPro Demo EVB

Ameba Demo board home page: https://www.amebaiot.com/zh/amebapro/

CPU

32-bit Arm v8M, up to 300MHz

32-bit Arm®Cortex®-M0, up to 4MHz

MEMORY

512KB RAM + 32MB LPDDR

KEY FEATURES

Integrated 802.11ac/n Wi-Fi SoC

Trustzone-M Security

Hardware SSL Engine

Root Trust Secure Boot

USB Host/Device

SD Host

LCDC

Codec

ISP

H.264

OTHER FEATURES

4 SPI interface

5 UART interface

2 I2S interface

4 I2C interface

11 ADC interface

16 PWM

2 PCM

Max 90 GPIO

1.2 PCB Layout Overview

The PCB layout of AmebaPro is shown in Fig 1-1.

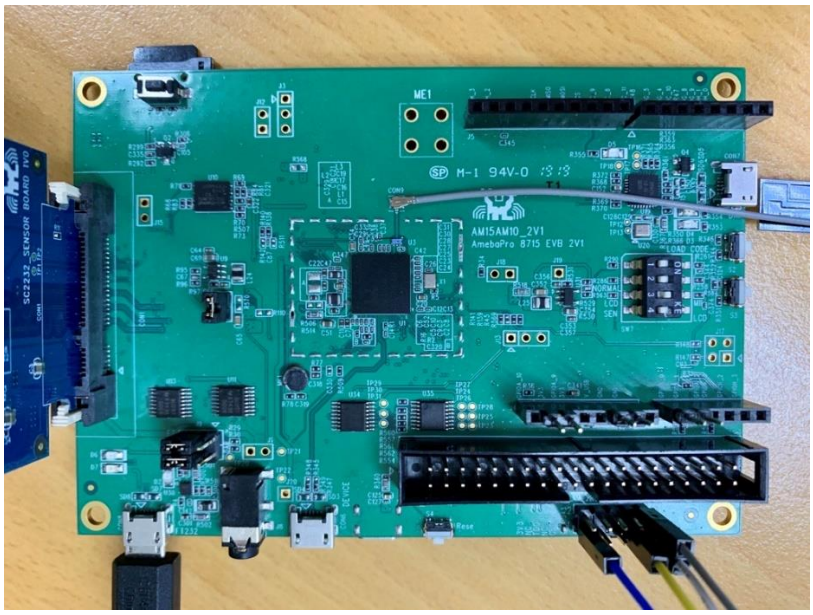


Fig 1-1 Demo board – PCB layout (2D)

1.3 LOGUART

The LOGUART is shown in Fig 1-2.

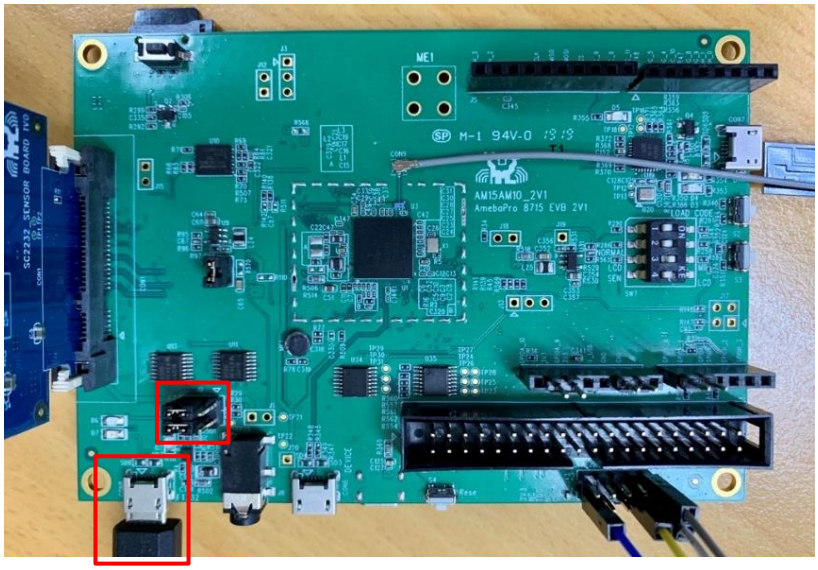


Fig 1-2 Demo board – LOGUART

1.4 JTAG/SWD

The SWD interface is shown in Fig 1-3.

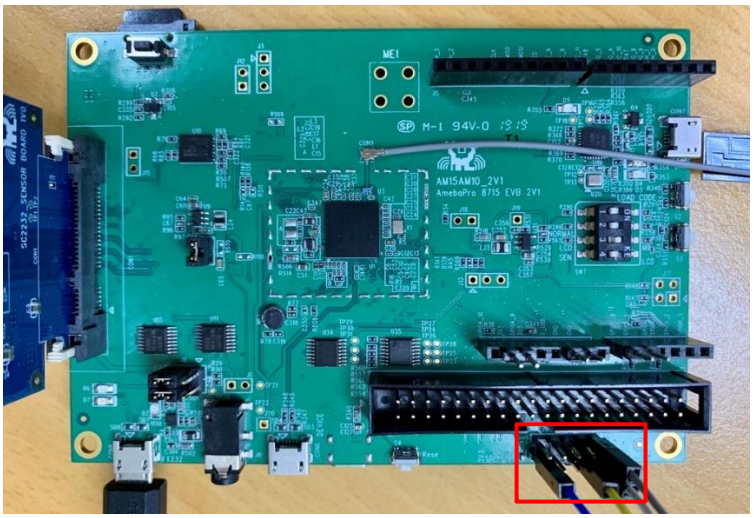


Fig 1-3 Demo board – JTAG/SWD

Note: If using 2V0、2V1 version AmebaPro. Please check SW7 pin 3 switch to ON before connection.

1.5 Image Sensor

There is an image sensor socket as shown in Fig 1-4.

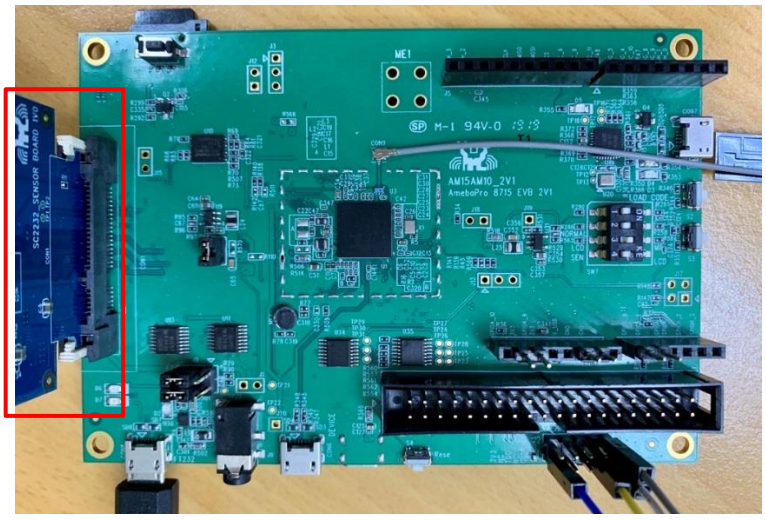


Fig 1-4 Demo board – image sensor

1.6 Requirement for Project Building

Supported IDE/toolchain: IAR, GCC

IAR Embedded Workbench - IAR Systems:

Please use IAR version 8.3 (There may be some compiler problems with v8.4)

GCC toolchain:

Linux: asdk-6.4.1-linux-newlib-build-3026-x86_64
Cygwin: asdk-6.4.1-cygwin-newlib-build-2778-i686

2 Set Up an AWS Account and Create an Administrator

Before you use Kinesis Video Streams for the first time, refer AWS official guide to complete the following tasks: (https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/gs-account.html)

- Sign Up for AWS (unless you already have an account)
- Create an Administrator IAM User
- Create an AWS Account Key

2.1 Sign Up for AWS

If you already have an AWS account, you can skip this step.

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Kinesis Video Streams. When you use Kinesis Video Streams, you are charged based on the amount of data ingested into, stored by, and consumed from the service. If you are a new AWS customer, you can get started with Kinesis Video Streams for free.

To create an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.
   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Write down your AWS account ID because you need it for the next task.

2.2 Create an Administrator IAM User

When you sign up for AWS, you provide an email address and password that is associated with your AWS account. This is your AWS account root user. Its credentials provide complete access to all of your AWS resources.

Note:
For security reasons, we recommend that you use the root user only to create an administrator, which is an IAM user with full permissions to your AWS account. You can then use this administrator to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the IAM User Guide.

To create an administrator and sign into the console

1. Create an administrator in your AWS account. For instructions, see Creating Your First IAM User and Administrators Group in the IAM User Guide.
2. As an administrator, you can sign in to the console using a special URL. For more information, see How Users Sign in to Your Account in the IAM User Guide.

The administrator can create more users in the account. IAM users by default don't have any permissions. The administrator can create users and manage their permissions. For more information, see Creating Your First IAM User and Administrators Group.

2.3 Create an AWS Account Key

You will need an AWS Account Key to access Kinesis Video Streams programmatically.

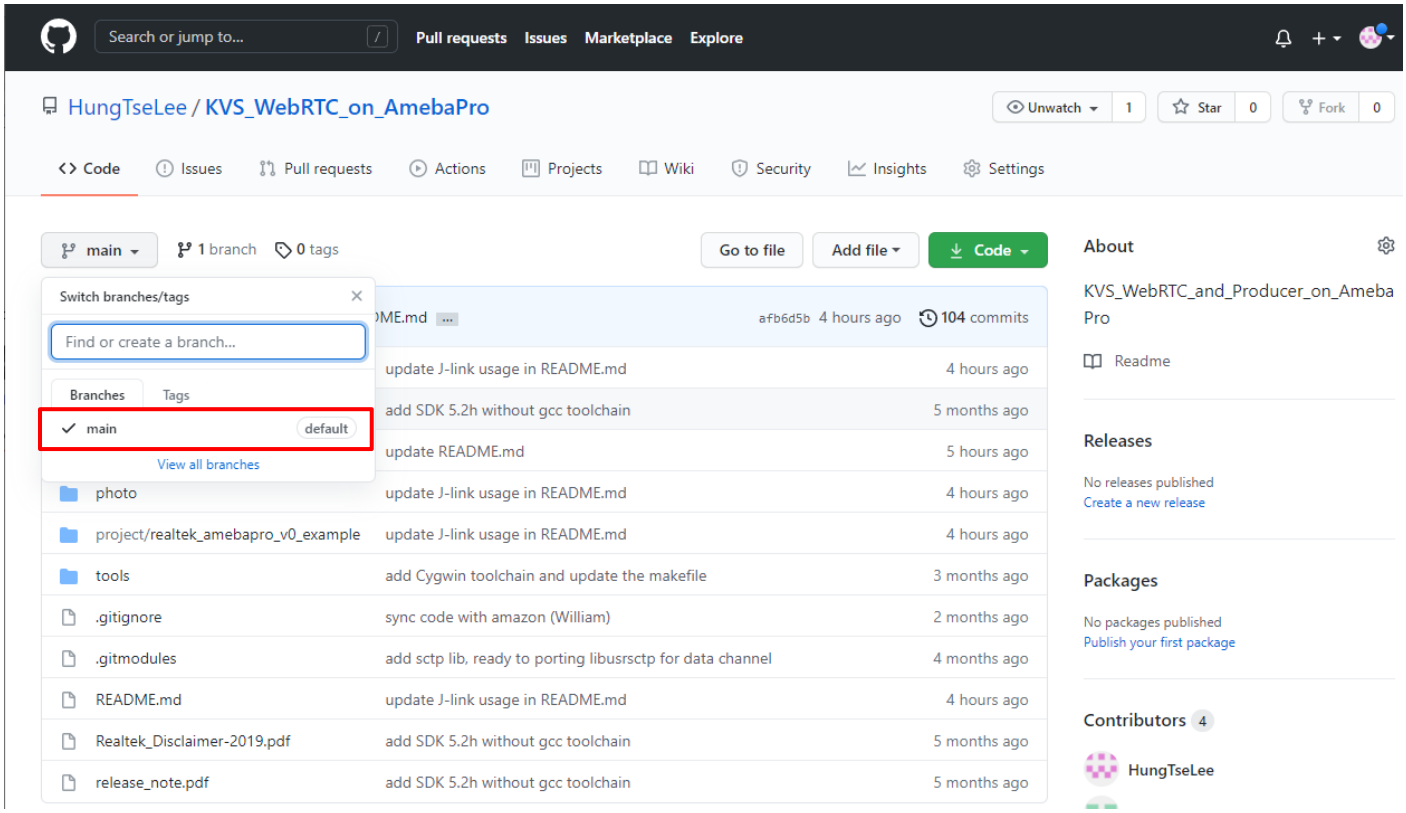
To create an AWS Account Key, do the following:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Users** in the navigation bar, and choose the **Administrator** user.
3. Choose the **Security credentials** tab, and choose **Create access key**.
4. Record the **Access key ID**. Choose **Show** under **Secret access key**, and then record the **Secret access key**.

3 Configure AmebaPro for Amazon KVS

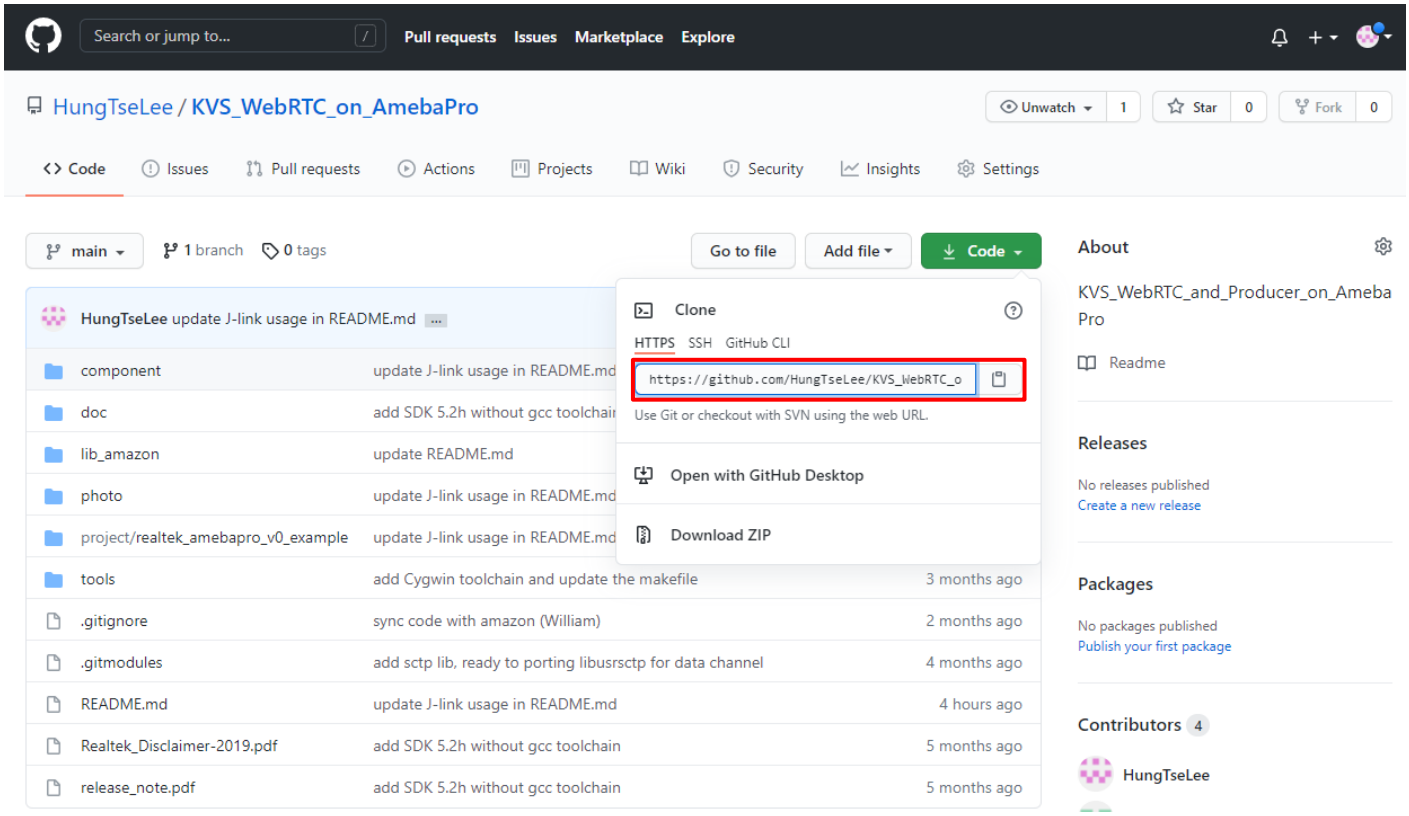
3.1 Download Source Code from Github

Open source link: https://github.com/HungTseLee/KVS_WebRTC_on_AmebaPro and select main branch for get newest source code. The stable version could be found by choosing specific tag.



3.1.1 Cloning the Repository by Git Command

On GitHub, navigate to the main page of the repository, and check its web URL.



Open a terminal on PC and run the command to download the whole project, including the libraries in submodule.

$ git clone --recurse-submodules https://github.com/HungTseLee/KVS_WebRTC_on_AmebaPro.git

If you already have a checkout, run the following command to sync submodules:

$ git submodule update --init

3.2 Choose Image sensor

Please check **image sensor module name** is correct in "**sensor.h"** located in **\project\realtek_amebapro_v0_example\inc**
For example, if I use the sensor model IMX307, the SENSOR_USE should be defined as SENSOR_IMX307.

```
#define ISP_FW_FLASH    0x00
#define ISP_FW_DRAW     0x01

#define ISP_FW_INTERNAL  0x00
#define ISP_FW_USERSPACE 0x01

#define ISP_AUTO_SEL_DISABLE 0X00
#define ISP_AUTO_SEL_ENABLE  0X01

#define SENSOR_USE          SENSOR_IMX307
#define SENSOR_AUTO_SEL         ISP_AUTO_SEL_ENABLE //Enalbe Auto select
```

3.3 Set Access Key ID and Secret Access Key on AmebaPro

After getting **Access key ID** and **Secret access key** in chapter 2.3, enter the key pair and stream name in file: **example_kvs_producer.h**
Additionally, you can change the kvs region by modifying **AWS_KVS_REGION**. The default region is **us-east-1**.

```
#define AWS_ACCESS_KEY          "xxxxxxxxxxxxxxxxxxx"
#define AWS_SECRET_KEY          "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

#define DEVICE_NAME             "DEV_12345678"

#define KVS_DATA_ENDPOINT_MAX_SIZE   ( 128 )

#define AWS_KVS_REGION       "us-east-1"
#define AWS_KVS_SERVICE      "kinesisvideo"
#define AWS_KVS_HOST         AWS_KVS_SERVICE "." AWS_KVS_REGION ".amazonaws.com"
#define HTTP_AGENT           "myagent"
#define KVS_STREAM_NAME      "my-kvs-stream"
//#define KVS_STREAM_NAME          "kvs_example_camera_stream"
#define DATA_RETENTION_IN_HOURS ( 2 )
```

3.4 Enable KVS Producer Demo

All examples provided by RTK exist in folder: SDK_path/common/example. Open platform_opts.h to specify the example to run.
For example, if users are going to use KVS Producer, compile flag CONFIG_EXAMPLE_KVS_PRODUCER should be set to 1, which means

#define CONFIG_EXAMPLE_KVS_PRODUCER 1

```
/* For KVS WebRTC example*/
#define CONFIG_EXAMPLE_KVS_WEBRTC               0
#if CONFIG_EXAMPLE_KVS_WEBRTC
#define CONFIG_FATFS_EN 1
#if CONFIG_FATFS_EN
// fatfs disk interface
#define FATFS_DISK_SD   1
#endif
#endif

/* For KVS Producer example*/
#define CONFIG_EXAMPLE_KVS_PRODUCER             1
```

Now you can start to compile AmebaPro Amazon KVS

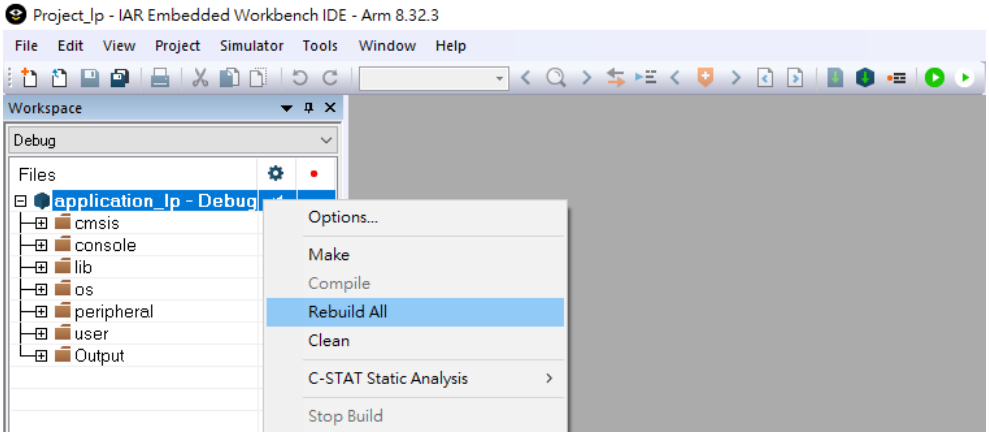
4 Compile AmebaPro Amazon KVS Project

4.1 IAR Embedded Workbench Build Environment Setup

AmebaPro use the newest Big-Little architecture. Since the big CPU will depend on the setting of small CPU, **it is necessary to compile the small CPU before the big CPU.**

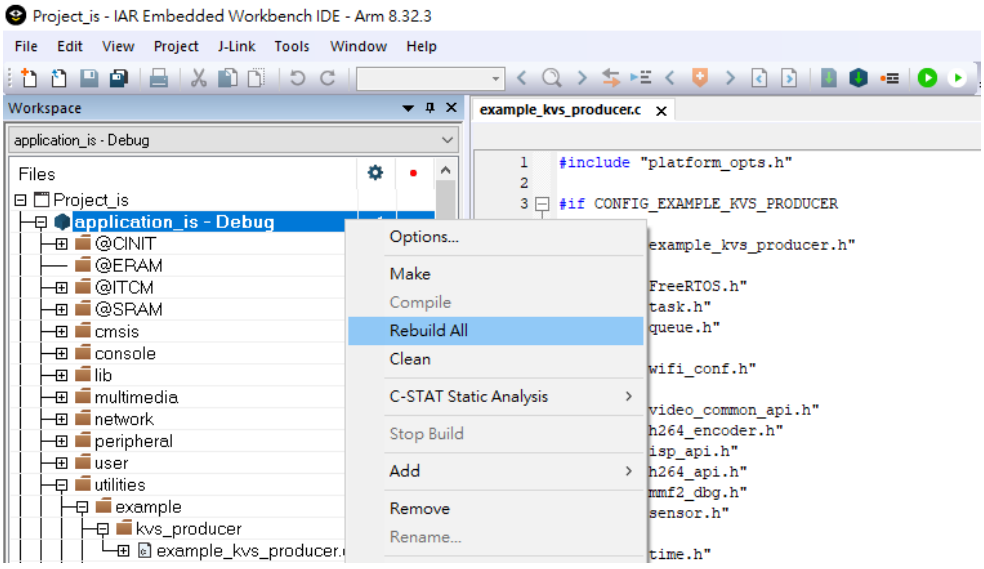
4.1.1 Compile Little CPU

step 1. Open SDK/project/realtek_amebapro_v0_example/EWARMRELEASE/Project_lp.eww.
step 2. Confirm application_lp in WorkSpace, right click application_lp and choose “Rebuild All” to compile.
step 3. Make sure there is no error after compile.



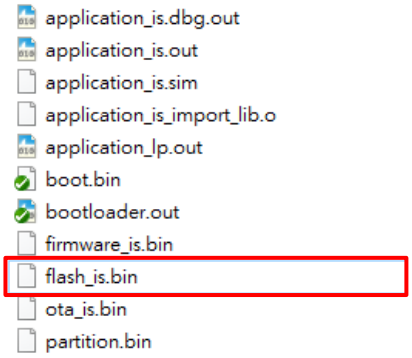
4.1.2 Compile Big CPU

step 1. Open SDK/project/realtek_amebapro_v0_example/EWARMRELEASE/Project_is.eww.
step 2. Confirm application_is in WorkSpace, right click application_is and choose “Rebuild All” to compile.
step 3. Make sure there is no error after compile.



4.1.3 Generating Image (Bin)

After compile, the images partition.bin, boot.bin, firmware_is.bin and flash_is.bin can be seen in the **EWARM-RELEASE\Debug\Exe**.
flash_is.bin links partition.bin, boot.bin and firmware_is.bin. Users need to choose **flash_is.bin** when downloading the image to board by Image Tool.



4.2 Compile Program with GCC Toolchain

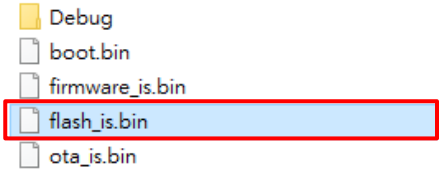
If using Linux environment or Cygwin on windows, follow the instructions below to build the project

$ cd project/realtek_amebapro_v0_example/GCC-RELEASE

Build the library and the example by running make in the directory

$ make -f Makefile_amazon_kvs all

If somehow it built failed, you can try to type **$ make -f Makefile_amazon_kvs clean** and then redo the make procedure.
After successfully build, there should be a directory named “application_is” created under GCC-RELEASE/ directory.
The image file **flash_is.bin** is located in ”application_is” directory.



Note:
if there is compile error with shell script, you may need to run following command to deal with the problem

$ dos2unix component/soc/realtek/8195b/misc/gcc_utility/*

5 Using Image Tool to Download Image

The tool **ImageTool.exe** can be find in **project\tools\AmebaPro\Image_Tool\ImageTool.exe**

5.1 Introduction

As show in the following figure, Image Tool has two tab pages:

- Download: used as image download server to transmit images to AmebaPro through UART
- Generate: concat separate images and generate a final image

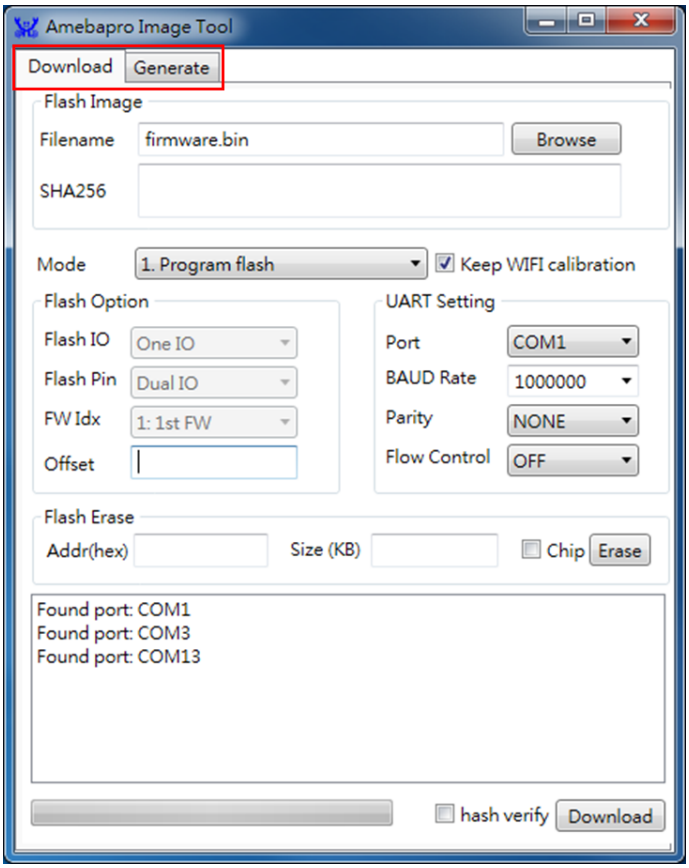


Fig 5-1 ImageTool UI

5.2 Environment Setup

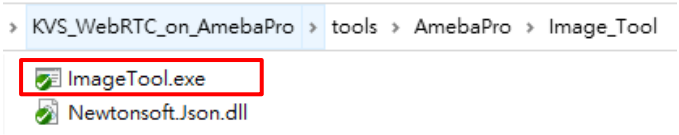
5.2.1 Hardware Setup

The hardware setup is shown in Fig 5-2.

Fig 5-2 Hardware setup

5.2.2 Software Setup

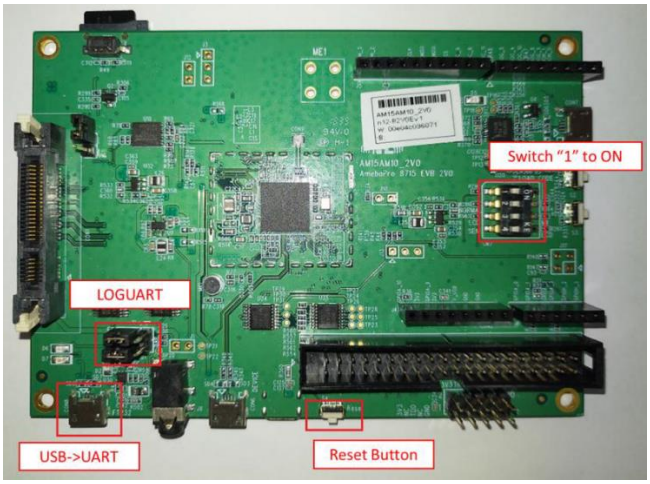
Execute **ImageTool.exe** from location **project\tools\AmebaPro\Image_Tool\ImageTool.exe**



5.3 Download

5.3.1 Enter the Download Mode to Ready

Image tool use UART to transmit image to AmebaPro board. Before performing image download function, AmebaPro need to enter UART_DOWNLOAD mode first. Please follow below steps to get AmebaPro into UART_DOWNLOAD mode:

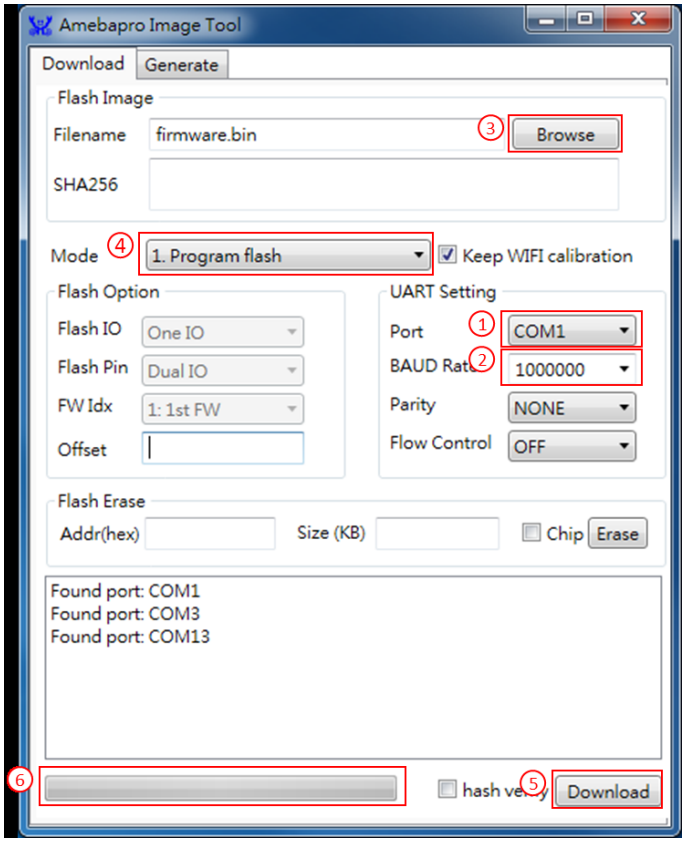


step 1. Connect LOGUART with FT pin by jumper cap.
step 2. Connect USB->UART to PC by using micro-USB wire.
step 3. Switch "1" to ON from SW7(2V0、2V1) or Switch "2" to ON from SW7(1V0)
step 4. Push reset button.

5.3.2 Download the Image to Flash

To download image through Image Tool, device need to enter UART_DOWNLOAD mode first.
Steps to download flash are as following:

step 1. Application will scan available UART ports. Please choose correct UART port. Please close other UART connection for the target UART port.
step 2. Choose desired baud rate between computer and AmebaPro.
step 3. Choose target flash binary image file “flash_xx.bin”
step 4. Check Mode is “1. Program flash”
step 5. Click “Download”
step 6. Progress will be shown on progress bar and result will be shown after download finish.
step 7. Switch “1” to OFF from SW7(2V0、2V1) or Switch “2” to OFF from SW7(1V0)
step 8. Push reset button to start the program.



6 Using J-Link to Download Image and Debug (GCC)

If under linux environment, using J-link to download image to EVB will be recommended.
AmebaPro supports J-Link for code download and enter debugger mode with GCC. The settings for J-Link debuggers are described below. Here, we will use **segger j-link** to demonstrate how to download image via SWD interface.

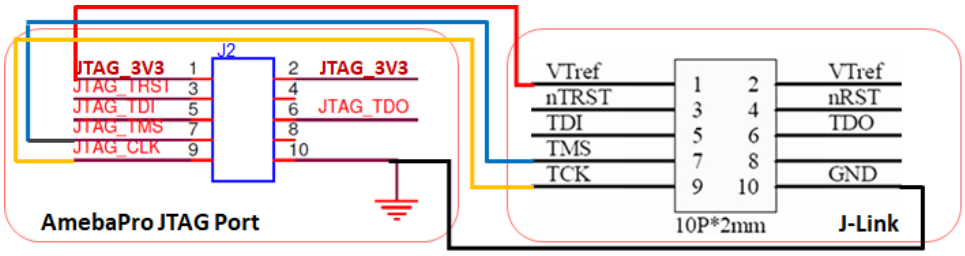
6.1 J-Link with SWD Interface

Note that if you are using Virtual Machine as your platform, please make sure the USB connection setting between VM host and client is correct so that the VM client can detect the device.
The external SWD interface requires two pins: bidirectional SWDIO signal and a clock, SWCLK, which can be input or output from the device.

Note:
If using 2V0、2V1 version AmebaPro. Please check SW7 pin 3 switch to ON before connection.

if using SWD, please check four pin (VTref[VDD]、TMS[SWDIO]、TCLK[SWCLK] and TDO[SWO]) connected to EVB correctly.

Reminder:
The JTAG pin names are incorrect on AmebaPro 2V0、2V1. Please follow the diagram in the following figure to connect AmebaPro to JTAG/SWD debugger.



6.2 Linux J-Link GDB Server

For J-Link GDB server, please check http://www.segger.com and download "**J-Link Software and Documentation Pack**" (https://www.segger.com/downloads/jlink). We suggest using Debian package manager to install the Debian version:

$ dpkg -i JLink_Linux_V698e_x86_64.deb

After the installation of the software pack, there should be a tool named "JLinkGDBServer" under JLink directory. Take Ubuntu 16.04 as example, the JLinkGDBServer can be found at **/opt/SEGGER/JLink/** directory. Please open a new terminal and type following command to start GDB server. Note that this terminal should NOT be closed if you want to download software or enter GDB debugger mode.

$ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m33 -if SWD

```
swinglass@ubuntu:~$ sudo /home/swinglass/Downloads/JLink_Linux_V632c_x86_64/JLinkGDBServer -device cortex-m33 -if SWD
SEGGER J-Link GDB Server V6.32c Command Line Version

JLinkARM.dll V6.32c (DLL compiled May 11 2018 16:32:39)

Command line: -device cortex-m33 -if SWD
-----GDB Server start settings-----
GDBInit file:                  none
GDB Server Listening port:     2331
SWO raw output listening port: 2332
Terminal I/O port:             2333
Accept remote connection:      yes
Generate logfile:              off
Verify download:               off
Init regs on start:            off
Silent mode:                   off
Single run mode:               off
Target connection timeout:     0 ms
------J-Link related settings------
J-Link Host interface:         USB
J-Link script:                 none
J-Link settings file:          none
------Target related settings-----
Target device:                 cortex-m33
Target interface:              SWD
Target interface speed:        4000kHz
Target endian:                 little

Connecting to J-Link...
J-Link is connected.
Firmware: J-Link EDU Mini V1 compiled Mar 29 2018 17:48:40
Hardware: V1.00
S/N: 801004052
Feature(s): GDB, FlashBP
Checking target voltage...
Target voltage: 3.30 V
Listening on TCP/IP port 2331
Connecting to target...Connected to target
Waiting for GDB connection...
```

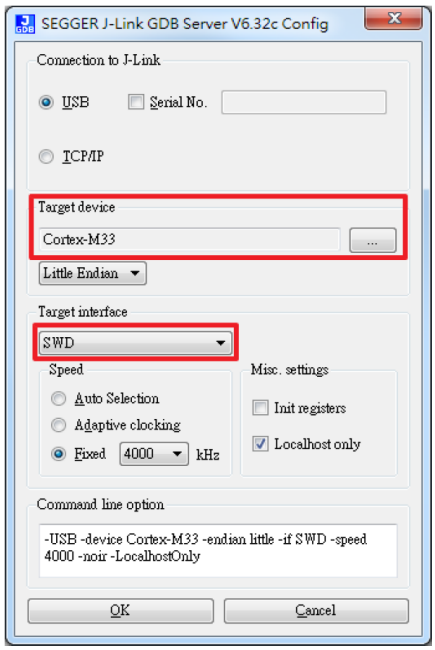
The started J-Link GDB server should looks like above figure. Please make sure the TCP/IP port is 2331 which should be the same as default setting in **component\soc\realtek\8195a\misc\gcc_utility\rtl_gdb_flash_write.txt**

On the project terminal you should type below command before you using J-Link to download software or enter GDB debugger:

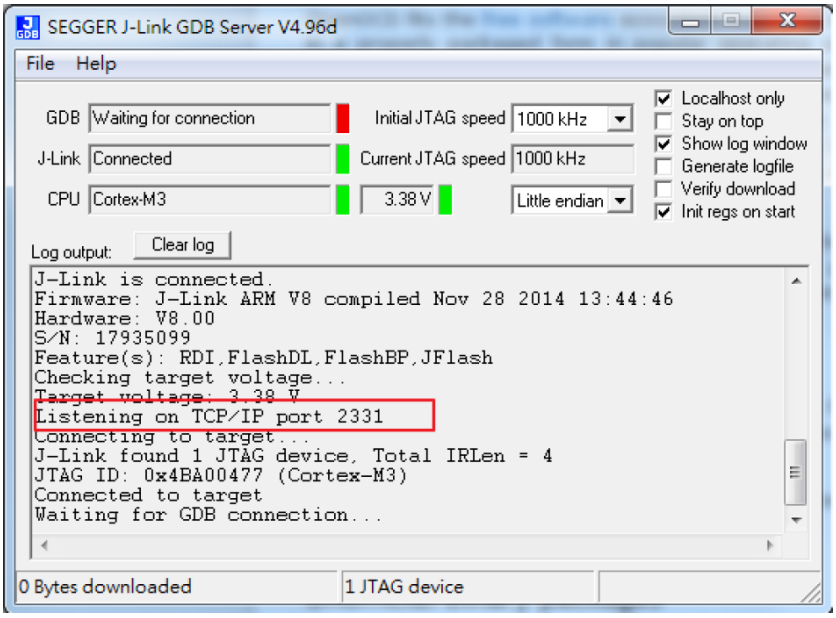
$ make -f Makefile_amazon_kvs setup GDB_SERVER=jlink

6.3 Windows J-Link GDB Server

Besides the hardware configuration, it also requires installing J-Link GDB server. For Windows, please check http://www.segger.com and download “J-Link Software and Documentation Pack” (https://www.segger.com/downloads/jlink). After the installation of the software pack, you should see a tool named “J-Link GDB Server”. Execute the J-Link GDB Server tool and choose the target device to Cortex-M33 and target interface to SWD to start GDB server:



The started J-Link GDB server should looks like below figure. And this window should **NOT** be closed if you want to download software or enter GDB debugger mode.



On the Cygwin terminal you should type below command before you using J-Link to download software or enter GDB debugger:

$ make -f Makefile_amazon_kvs setup GDB_SERVER=jlink

6.4 Download Image to Flash

After building the project in GCC, check that image exists in ”application_is” directory.
Then, go back to **project/realtek_amebapro_v0_example/GCC-RELEASE** and run the command:

$ make -f Makefile_amazon_kvs flash

Now, the image is being downloaded to EVB.
Press the reset botton on the EVB to run the example after downloading.

Note:
If there is no reponse after run the command above, quit the GDB mode and press the reset botton on EVB. Try the command again.

6.5 Enter GDB Debugger

type below command to enter GDB debug mode:

$ make -f Makefile_amazon_kvs debug

For further information about GDB debugger and its commands, please check https://www.gnu.org/software/gdb/ and https://sourceware.org/gdb/current/onlinedocs/gdb/.

7 KVS Producer Demo

7.1 Get Device Log

Install **Tera Term** or other terminal emulator to get device log

Fig 7-1 Hardware setup

The serial port is same with ImageTool that get from 5.3.2 or use device manager to get the right serial port of device.

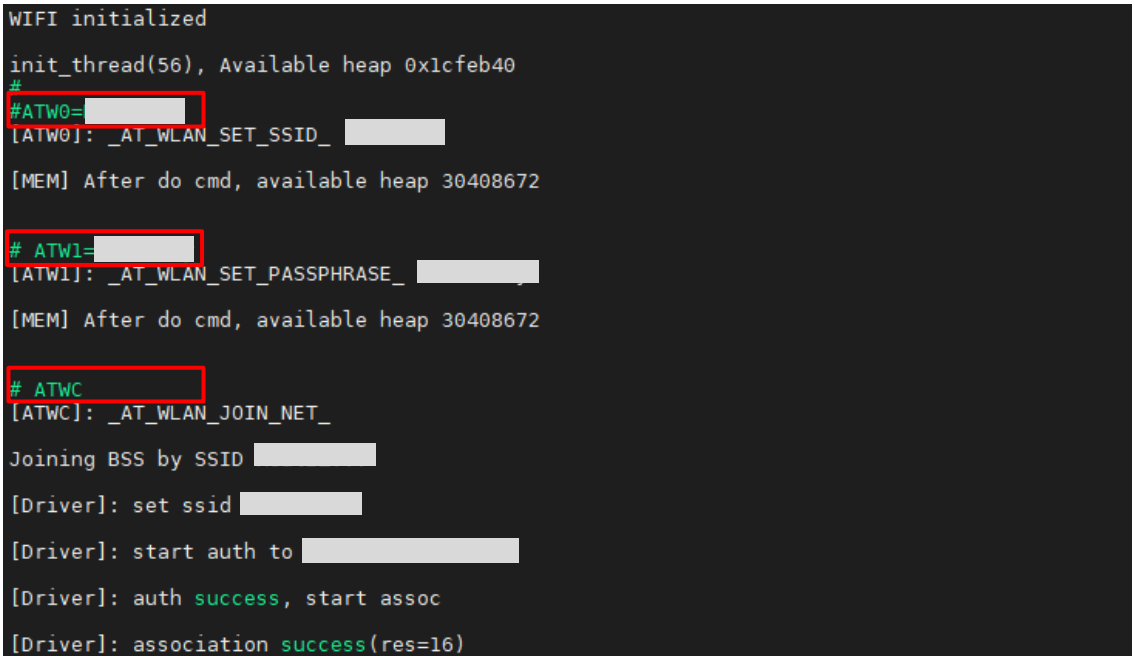
7.2 Run KVS Producer Demo

Default setting of SDK may not enable Producer demo, so please refer to Ch 3.3 to check whether KVS Producer Demo is enabled. Once the AmebaPro EVB has rebooted, the application will automatically start run Producer demo and put media to KVS.

7.2.1 Connect to WIFI AP

In order to run the example, AmebaPro should connect to the network. It can be achieved by run the AT command in uart console. Please refer to the steps below:

ATW0=<WiFi_SSID> : Set the WiFi AP to be connected
ATW1=<WiFi_Password> : Set the WiFi AP password
ATWC : Initiate the connection

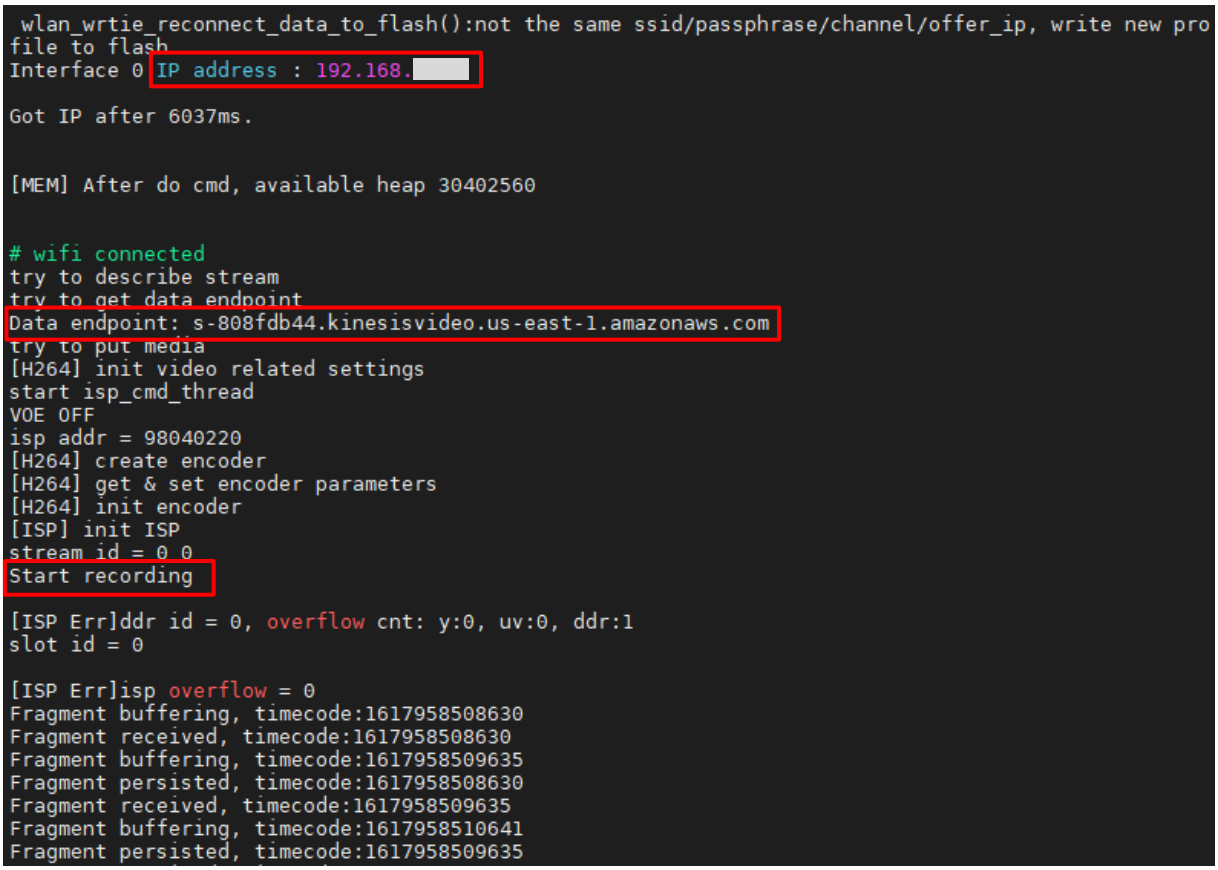


more information about RTK AT command:
https://github.com/HungTseLee/KVS_WebRTC_on_AmebaPro/blob/main/doc/AN0025%20Realtek%20at%20command.pdf

Note:
AmebaPro has the wifi fast connection after rebooting. Therefore, it is not required to run the command above again if you want AmebaPro connecting to the same AP.

7.2.2 Running Producer and Put Media to Stream

After connecting to the wifi and get an IP address, the AmebaPro will run the example and put the media to KVS…



Now the AmebaPro have started to out the media (video/audio) to the kinesis video streams.

7.3 Verify the Demo

There are two method can be used to verify demo easily, which are **Media Viewer Webpage** and **AWS KVS Console**.

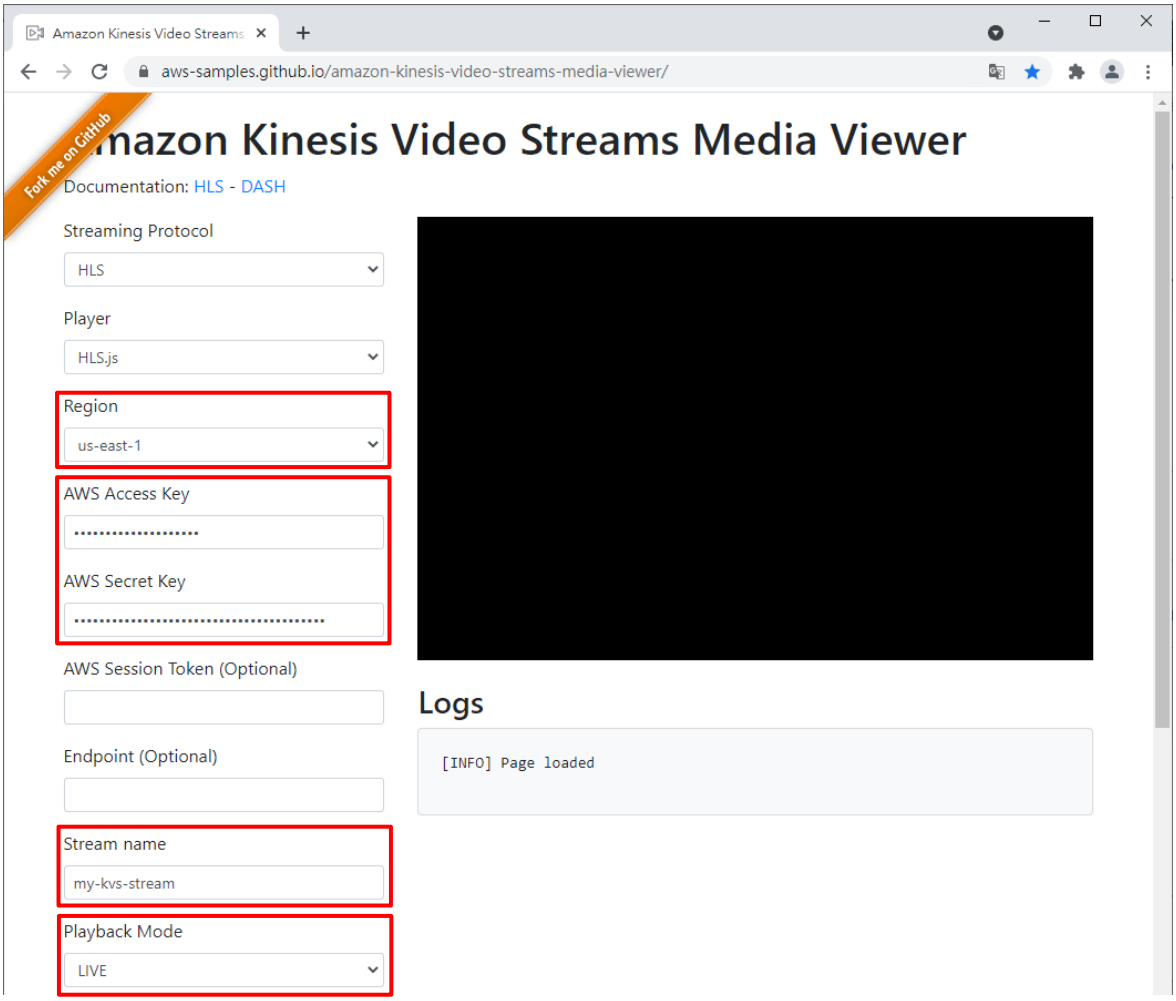
7.3.1 Use Media Viewer Webpage to Verify

We can use an on-line test page to verify the media is being sent to the video streams.

step 1. Go to test page: https://aws-samples.github.io/amazon-kinesis-video-streams-media-viewer

step 2. Set the **AWS region**, **Credential key**, **Stream name**

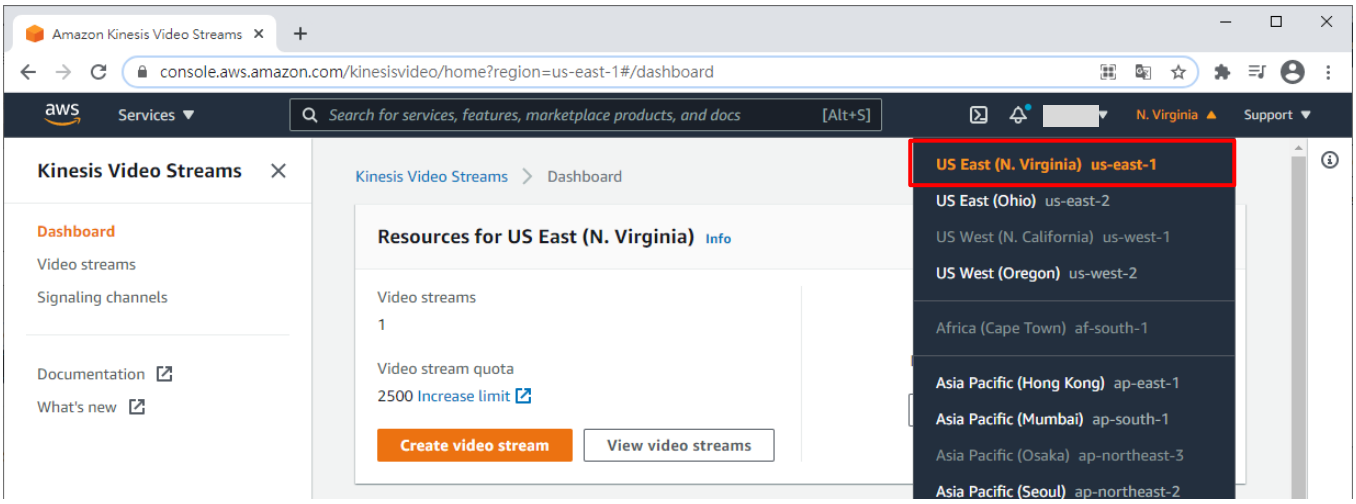
step 3. Select the "LIVE" playback mode and press the "Start Playback" button to validate the result.



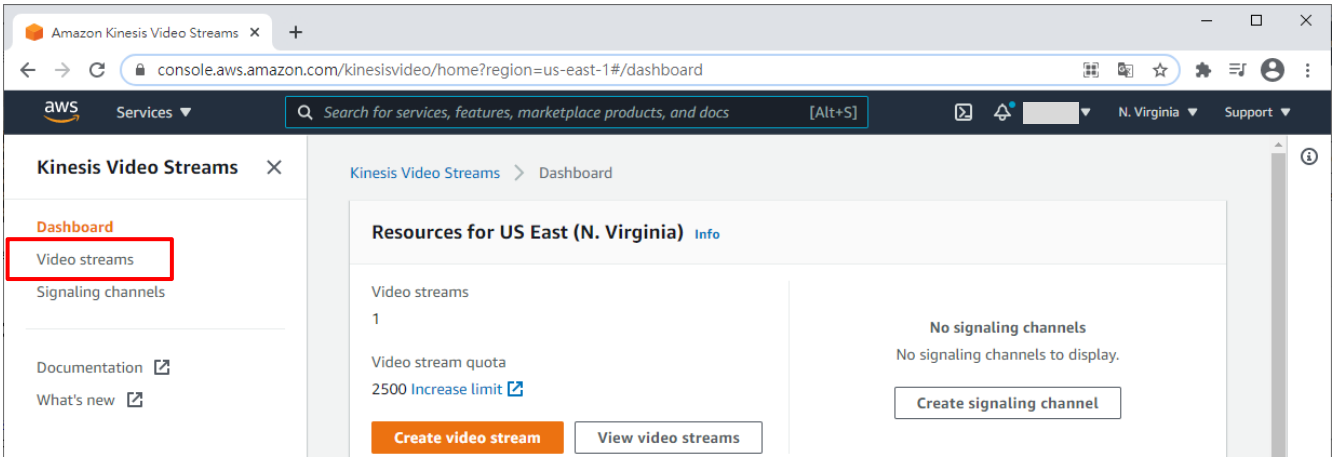
7.3.2 Use AWS KVS Console to Verify

We can use the AWS KVS console to verify the media is being sent to the video streams.

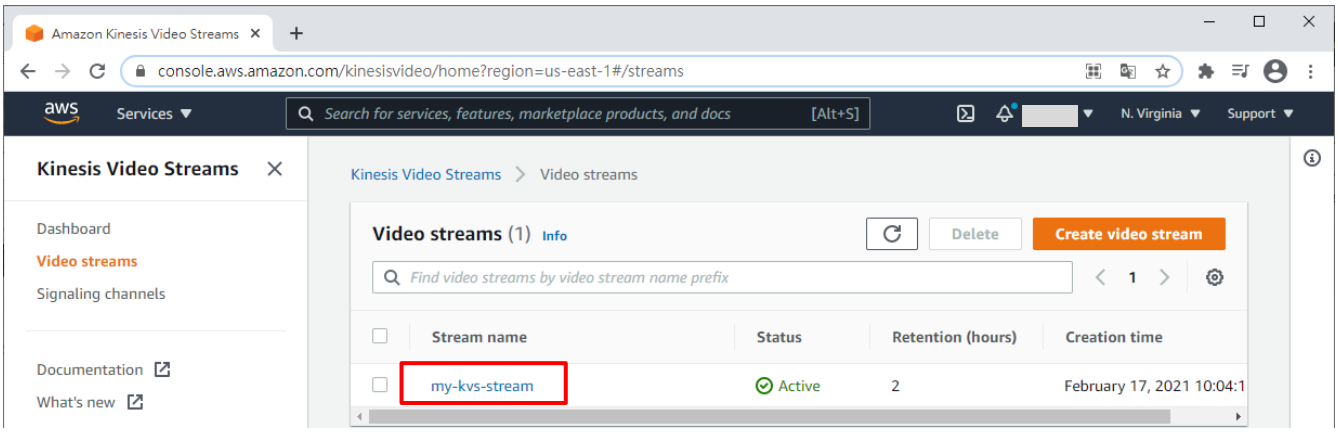
step 1. Go to AWS KVS console: https://console.aws.amazon.com/kinesisvideo/home?region=us-east-1#/streams
step 2. Choose the AWS region that you create the stream (default: us-east-1)



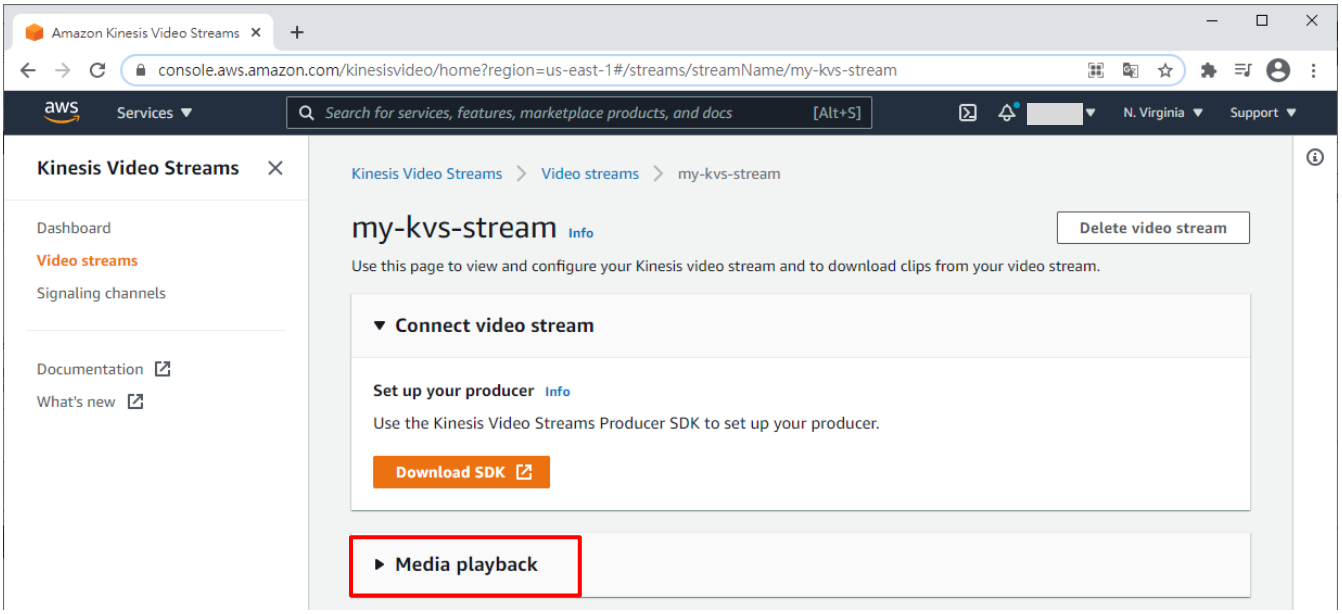
step 3. Click the **Video streams** on the left and choose the stream that created by AmebaPro



...



step 4. Using **Media playback** to playback the video



If success, you will get the video/audio from AmebaPro on the console…

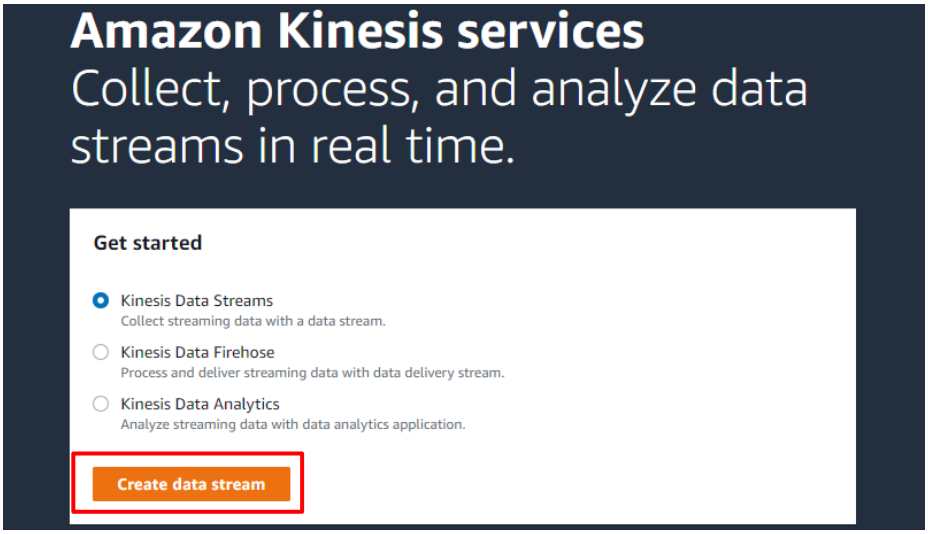
8 KVS Producer + Rekognition Example

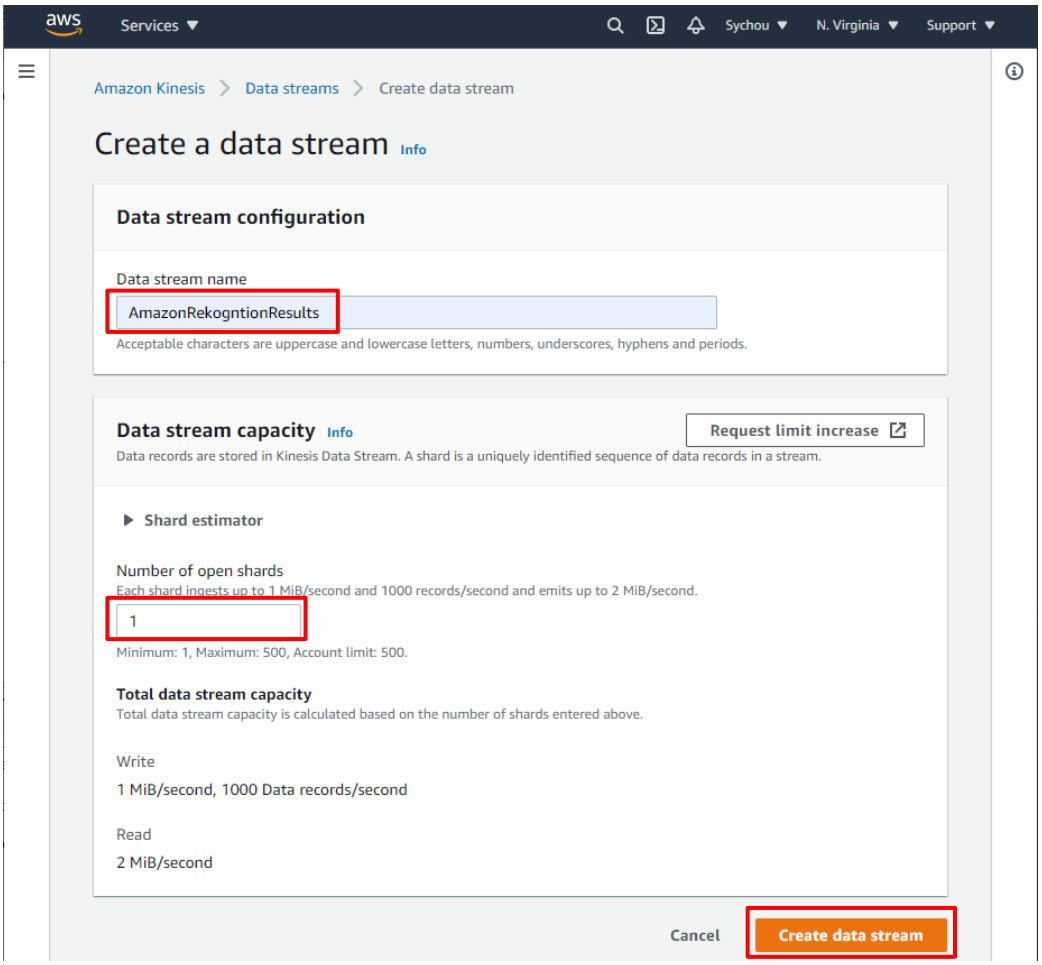
There are several cloud services on AWS, including Rekognition(face detection) or other machine learning framework(Apache MxNet, TensorFlow, and OpenCV). In this section, we will demonstrate how to feed the AmebaPro's KVS data as the input of Rekognition sevice, and then get the face detection result from the Kinesis Data Stream, the output of Rekognition.
We refer an article in *AWS Machine Learning Blog* to perform the demo on AmebaPro: https://aws.amazon.com/tw/blogs/machine-learning/improve-your-customer-service-using-amazon-kinesis-video-streams-and-amazon-rekognition-video/
For more information of Rekognition working with streaming video, see https://docs.aws.amazon.com/rekognition/latest/dg/streaming-video.html.

8.1 Create a Kinesis Video Stream and Kinesis Data Stream

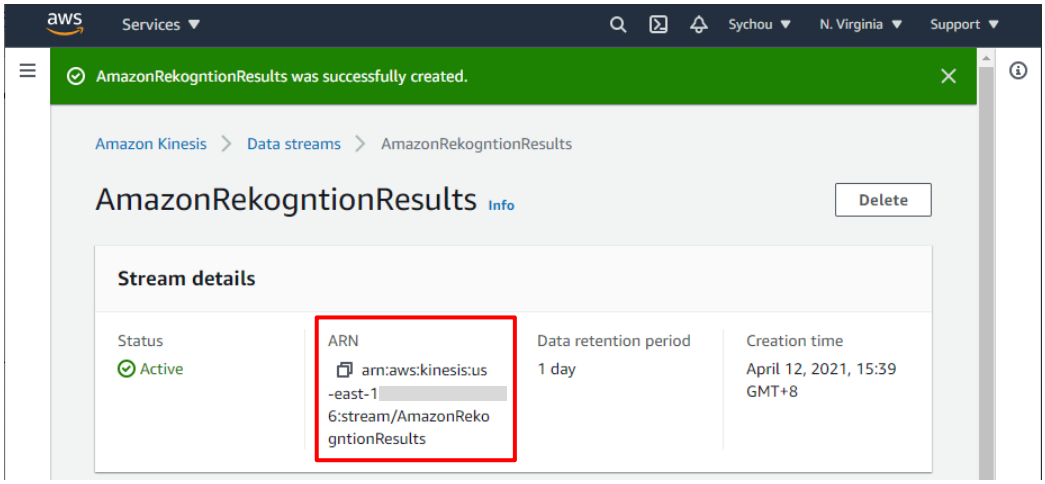
Kinesis Video Stream will automatically be created by AmebaPro, and you can check the existence of video stream in AWS Management Console. All you need to do in this section is to create a Data Stream where Rekognition will output its results.

Firstly, use the AWS Management Console to create a new Kinesis Video Stream.
step 1. Go to Amazon Kinesis services page: https://console.aws.amazon.com/kinesis/home?region=us-east-1#/home
step 2. Choose **Kinesis Data Streams** and click the **Create Kinesis stream** button.
step 3. Naming the stream as "AmazonRekogntionResults"
step 4. For data stream capacity, select 1 shards for the stream.
step 5. Create data stream





Use the AWS Management Console and click the data stream you create (AmazonRekogntionResults), then record the ARN number.



8.2 Start the Video Streaming

Please follow the steps from Chapter 2 to Chapter 7 to setup your AmebaPro, the video can then be put on the Kinesis Video Streams correctly. After a few seconds, you can view video frames from AmebaPro in the AWS Management Console for your Kinesis Video Stream. Now, we need to analyze the content in the streaming video.

8.3 Create Stream Processor for Rekognition

We can use the *stream processor* provided by *Rekognition Video service* to manage the analysis of content in Kinesis Video Stream. The permissions are required to create the stream processor:
step 1. go to the *IAM console* and choose to create a new IAM role
step 2. select "Rekognition" as the service that will use the role and attach the "AmazonRekognitionServiceRole" managed policy.
Now, the Rekognition has the rights to read from any Kinesis Video Stream and write to any Kinesis Data Stream prefixed with AmazonRekognition. In addition, record the ARN of your IAM role.

Note:
the basic settings should be configured to interact with AWS locally (like AWS CLI, local python code…). These include your **security credentials**, the default output format, and the **default AWS Region**, see:
https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html
https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html

Create a face collection: a private database that stores facial information of known people that you want Rekognition to detect. Using the AWS Command Line Interface(CLI) to create a face collection called my-collection:

$ aws rekognition create-collection --collection-id my-collection

Collect a photo of the person you want to detect, upload the image with a person to Amazon S3, and add them to your face collection with the CLI command:

$ aws rekognition index-faces --collection-id my-collection --image '{"S3Object":{"Bucket":"<bucket>","Name":"<key>"}}' --external-image-id <name>

Parameter description:
- <bucket> with the name of the Amazon S3 bucket
- <key> with key to the Amazon S3 object (a PNG or JPEG file)
- <name> with the name of the person

Create your stream processor with the following CLI command:

$ aws rekognition create-stream-processor --input '{"KinesisVideoStream":{"Arn":"<video stream ARN>"}}' --name store-processor --role-arn <role ARN> --stream-processor-output '{"KinesisDataStream":{"Arn":"<data stream ARN>"}}' --settings '{"FaceSearch":{"CollectionId":"my-collection ", "FaceMatchThreshold": 85.5}}'

Or you can create stream processor in Python:

```
create_response = self.rekognition_client.create_stream_processor(
    Input={
        'KinesisVideoStream': {
            'Arn': KinesisVideoStreamArn
        }
    },
    Output={
        'KinesisDataStream': {
            'Arn': KinesisDataStreamArn
        }
    },
```

```
        Name=Rekognition_stream_processor_name,
        Settings={
                'FaceSearch': {
                        'CollectionId': 'MyCollection',
                        'FaceMatchThreshold': 0
                }
        },
        RoleArn=AWSroleArn
)
```

Parameter description:

- <video stream ARN> with the ARN of the Kinesis Video Stream
- <role ARN> with the ARN of the IAM role you created
- <data stream ARN> with the ARN of the Kinesis Data Stream

The CLI command returns the ARN of the newly created stream processor. You can start this by running the command:

$ aws rekognition start-stream-processor --name store-processor

Or you can start the stream processor in Python:

```
# start stream processor
start_response = self.rekognition_client.start_stream_processor(
        Name=Rekognition_stream_processor_name
)
```

After starting the stream processor, the Rekognition will analysis the video from KVS and send the face detection results to Kenesis data streams.

8.4 Get the Video from Kinesis Video Stream Locally

AWS provide the get_media() API to retrieve media content from a Kinesis video stream. However, we use **get_hls_streaming_session_url()** to retrieve HTTP Live Streaming (HLS) URL for the stream, and then use **cv2.VideoCapture** API in *OpenCV* to capture the frames from URL easily.

Get HLS streaming URL in Python:

```
url = kvam.get_hls_streaming_session_url(
        StreamName = KinesisVideoStream_name,
        PlaybackMode = "LIVE",
        HLSFragmentSelector = {
                'FragmentSelectorType': 'PRODUCER_TIMESTAMP'
        },
        ContainerFormat = 'MPEG_TS',  # FRAGMENTED_MP4
        DiscontinuityMode='ALWAYS',
        DisplayFragmentTimestamp = 'NEVER'
)['HLSStreamingSessionURL']
```

Capture and read the video frame from the URL in Python:

```
self.capture = cv2.VideoCapture(URL)
self.capture.set(cv2.CAP_PROP_BUFFERSIZE, 2)
self.frame_width = self.capture.get(cv2.CAP_PROP_FRAME_WIDTH)
self.frame_height = self.capture.get(cv2.CAP_PROP_FRAME_HEIGHT)

self.status, self.Frame = self.capture.read()  # Read KVS video from URL
```

8.5 Get the Result from Kinesis Data Stream Locally

Rekognition can now analyze the video stream from AmebaPro and put the results of its analysis onto a Kinesis Data Stream. For each frame it analyses, Rekognition Video may find many faces and each face may have many potential matches. This information is detailed in JSON documents that Rekognition Video puts onto the Kinesis Data Stream.

Get records from data stream in Python:

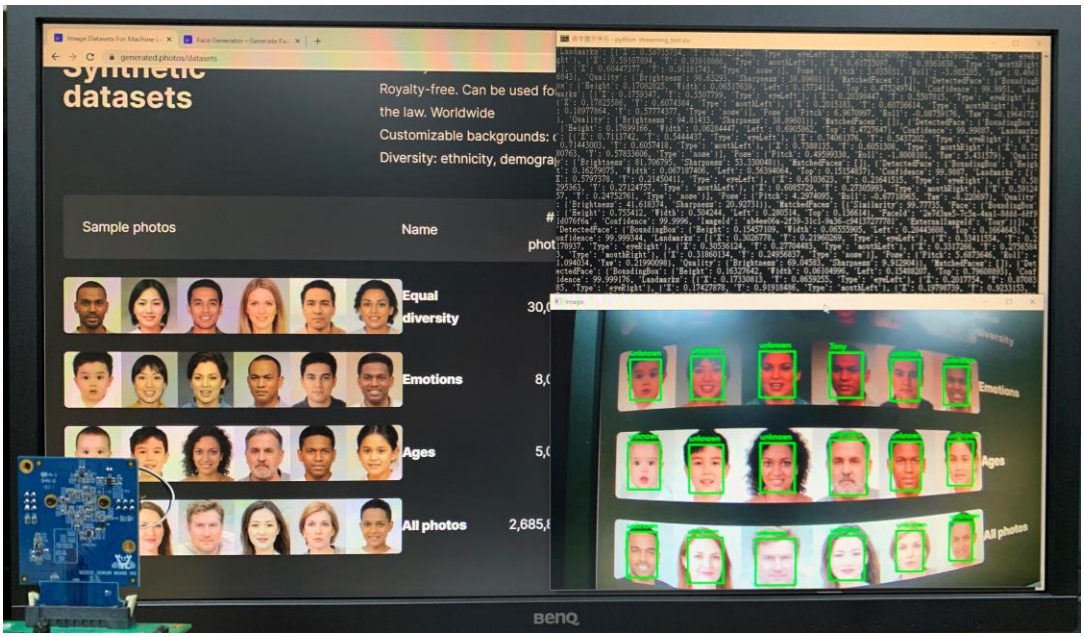
```
shard_iterator_response = self.kinesis_client.get_shard_iterator(
        StreamName=KinesisDataStream_name,
        ShardId='shardId-000000000000',
        ShardIteratorType='LATEST'
)

get_response = self.kinesis_client.get_records(
        ShardIterator=shard_iterator_response['ShardIterator']
)
```

8.6 Display Video frames Rendered with Bounding Boxes

The Rekognition results will provide the bounding boxes' position of the detected face in the frame. If there are lots of faces in the video, you can display all detected results returned from the get_records() API.

Following are the scenario of our demo:

- AmebaPro monitors PC's left screen, the sample photos are generated completely by AI (https://generated.photos)
- AmebaPro put the video to KVS
- Feed KVS as input of Rekognition
- Rekognition's result will be sent to Data Stream
- Run a python code locally to get the video frame from KVS and Rekognition result from Data Stream (Python packages: AWS Boto3 + OpenCV)
- Display the video frame rendered with bounding boxes



8.7 The Simple Demo Code for Getting the Results Locally (Python)

A simple python code “producer_rekognition_test.py” is given in “component/common/example/kvs_producer”. We can use it to perform the demo with AmebaPro.
Before running the python code, you should install the necessary packages (like boto3, opencv…). In addition, remember to configure basic settings that the AWS CLI and python code uses to interact with AWS. These include your **security credentials**, the default output format, and the **default AWS Region**, see:

https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html
https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html

The step of **creating a face collection** is **not** included in python, it make it more flexible to add the person id by using command line.

Setting the information of streams in producer_rekognition_test.py*:*

```
#set steam information
AWSroleArn = 'arn:aws:iam::XXXXXXXXXX:role/Rekognition-producer'
KinesisVideoStreamArn = 'arn:aws:kinesisvideo:us-east-1:XXXXXXXXXX:stream/my-kvs-stream/XXXXXXXXXX'
KinesisVideoStream_name = 'my-kvs-stream'
KinesisDataStreamArn = 'arn:aws:kinesis:us-east-1:XXXXXXXXXX:stream/AmazonRekogntionResults'
KinesisDataStream_name = 'AmazonRekogntionResults'
Rekognition_stream_processor_name = 'my-stream-processor'
```

Create a face collection:

$ aws rekognition create-collection --collection-id my-collection

Add person ID to collection:

$ aws rekognition index-faces --collection-id my-collection --image '{"S3Object":{"Bucket":"<bucket>","Name":"<key>"}}' --external-image-id <name>

Run the python code:

$ python producer_rekognition_test.py

After finishing the python code, you should check the stream processor is stop and deleted. It can avoid unnecessary expense:

$ aws rekognition stop-stream-processor --name my-stream-processor
$ aws rekognition delete-stream-processor --name my-stream-processor

9 Troubleshooting and More Information

If these steps don't work, look at the device log in the serial terminal. You may see the printing log in detail, and it will indicate the cause of the problem.

For general troubleshooting information about Getting Started with FreeRTOS, see Troubleshooting getting started.

9.1 Modify the Video Parameter

You may modify the video related parameter in **example_kvs_producer.h** and **example_kvs_producer.c**, including

example_kvs_producer.h: video resolution and FPS…

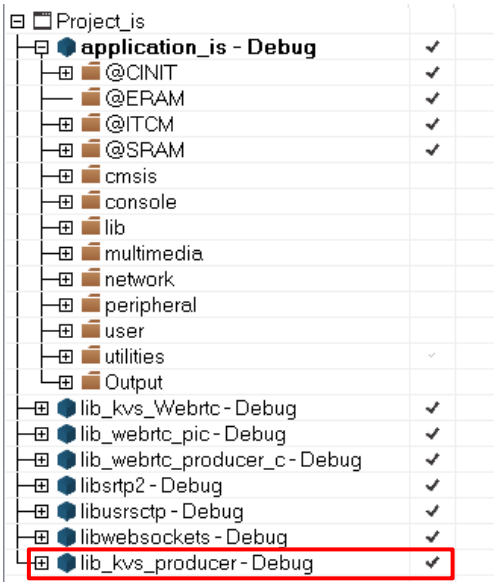
```
/* https://www.matroska.org/technical/codec_specs.html */
#define VIDEO_CODEC_NAME                        "V_MPEG4/ISO/AVC"
#define VIDEO_FPS                               30
//#define VIDEO_WIDTH                             1920
//#define VIDEO_HEIGHT                            1080
#define VIDEO_WIDTH                             1280
#define VIDEO_HEIGHT                            720
#define VIDEO_NAME                              "my video"
```

example_kvs_producer.c: H264 GoP, bitrate…

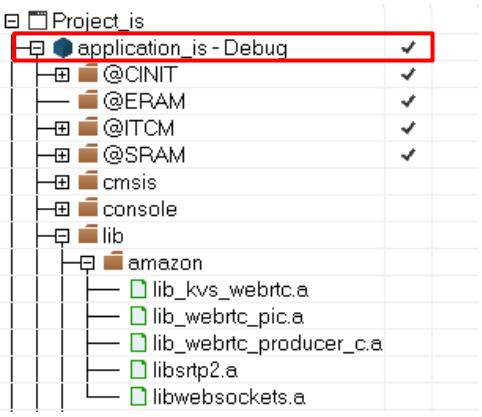
```
    h264_parm.height = VIDEO_HEIGHT;
    h264_parm.width = VIDEO_WIDTH;
    h264_parm.rcMode = H264_RC_MODE_CBR;
    h264_parm.bps = 1024 * 1024;
    h264_parm.ratenum = 30;
    h264_parm.gopLen = 30;
```

9.2 The Modified Library Content Cannot be Linked Correctly

Method: Rebuild the library and compile the application project again (IAR)
If the source codes in library are modified, the corresponding library should be rebuild.

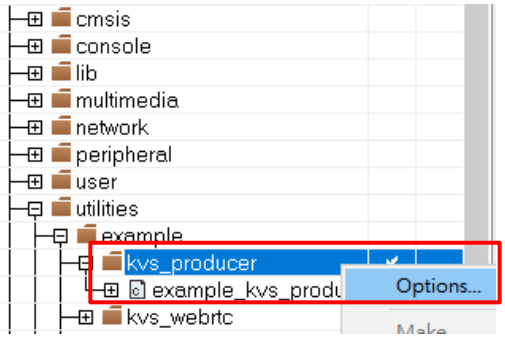


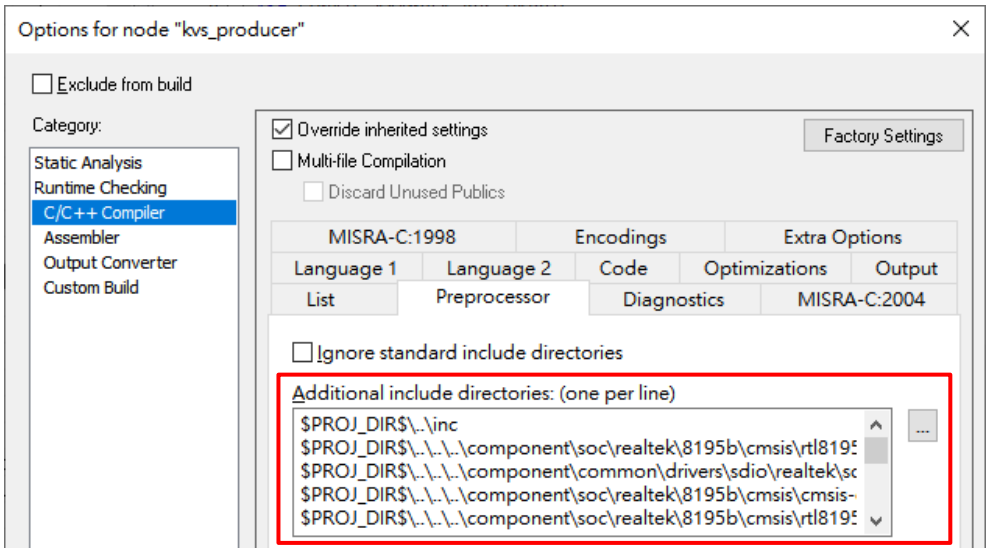
Then, go back to application_is project and "**Make**" again, the updated library can then be linked.



9.3 Cannot Find the Header File After Adding the Include Path

The additional include directories of KVS Producer example is temporarily independent. You can add additional include path by right clicking kvs_producer and choosing **options → C/C++Compiler → Preprocessor**





Note:
If the destination path of the header is too long, IAR may occur compile error. You can use the absolute path instead of relative path to deal with the issue. Or just move your header file to a shorter destination path.

9.4 Failed to Connect to AWS Server

Please check **AWS Access Key, AWS Secret Key**, **Stream name** and **Region** are provided in **example_kvs_producer.h**.

```
#define AWS_ACCESS_KEY          "xxxxxxxxxxxxxxxxxxx"
#define AWS_SECRET_KEY          "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

#define DEVICE_NAME             "DEV_12345678"

#define KVS_DATA_ENDPOINT_MAX_SIZE   ( 128 )

#define AWS_KVS_REGION       "us-east-1"
#define AWS_KVS_SERVICE      "kinesisvideo"
#define AWS_KVS_HOST         AWS_KVS_SERVICE "." AWS_KVS_REGION ".amazonaws.com"
#define HTTP_AGENT           "myagent"
#define KVS_STREAM_NAME      "my-kvs-stream"
//#define KVS_STREAM_NAME          "kvs_example_camera_stream"
#define DATA_RETENTION_IN_HOURS ( 2 )
```

9.5 Failed to Initialize the Image sensor

Please check **image sensor module name** is correct in "**sensor.h"** located in **\project\realtek_amebapro_v0_example\inc**.
For example, if I use the sensor model IMX307, the SENSOR_USE should be defined as SENSOR_IMX307.

```
#define ISP_FW_FLASH    0x00
#define ISP_FW_DRAW     0x01

#define ISP_FW_INTERNAL  0x00
#define ISP_FW_USERSPACE 0x01

#define ISP_AUTO_SEL_DISABLE 0X00
#define ISP_AUTO_SEL_ENABLE  0X01

#define SENSOR_USE          SENSOR_IMX307
#define SENSOR_AUTO_SEL         ISP_AUTO_SEL_ENABLE //Enalbe Auto select
```

9.6 Failed to Boot Up After Starting with a New Project

If your Amebapro cannot boot up correctly after downloading a new image and you find that the image size of **flash_is.bin** is smaller than expected. You may not compile the small CPU before the big CPU.

Method: Build the small CPU and then build the big CPU again, the correct image can be obtained.