AGL_test_FW_guider

Ver 0.0.1

2020/4/13

Reviewer	Executor
Li Xiaoming 2020/4/13	Qiu Tingting, Zhong Lu 2020/4/13

	Revision History						
Version	Updated Date	Contents	Remarks				
Ver 0.0.1	2020/4/13	Init version of AGL TEST FW GUIDER					

Page 2 of 11

Construction of test framework

■ Contents

History HW Construction of Fuego Construction of LAVA

AGL Image and SDK



m3ulcb	
target board	
8.10.20	

Construction of Fuego

1. introduction of Fuego

Fuego is the official automated test framework for the LTSI project. http://fuegotest.org/ The project's web site is at:

2. prepare for Fuego

To retrieve the Fuego software and create the docker image for it, you need to have git and docker installed on your system.

-	_
\$ sudo apt-get install git	
\$ sudo apt-get install docker io	

3. Installing Fuego to a docker container

Code for the test framework is available in 2 git repositories:

fuego

fuego-core

3-1. download the repository of Fuego v1.5.0 from bitbucket

\$ git clone https://bitbucket.org/fuegotest/fuego.git

\$ cd fuego

3-2. Run install.sh from the top directory.

This launches the ``docker build" command to create a docker image - named 'fuego' by default.

\$./install.sh

It also creates a container from this image, called 'fuego-container'. This container's port is 8090.

You can change the name of the docker image and the port used by Jenkins by passing the corresponding parameters to install.sh.

\$./install.sh fuego-8092 8092

3-3. Update "fuego-core"

The repository fuego-core is mounted as a external docker volume.

The fuego-core repository should be auto installed inside the fuego directory, at the top level of that repository's directory structure (parallel to fuego-ro and fuego-rw).

CHANGELOG	Dockerfile	Dockerfile.test	frontend-ins	tall	fuego-host-	scripts	fuego-rw	install.sh	LICENSE	start.sh
CREDITS	Dockerfile.nojenkins	docs	fuego-core		fuego-ro		install-debian.sh	jenkins	README	VERSION

4. Start "fuego-container"

To start your Fuego container, issue the following command top directory:

\$./start.sh fuego-container

(or, use the name of whatever container was created in the install step. For example, ./start.sh fuego-8092-container).

Jenkins and network are automatically started in docker container.

- [ok] Starting Jenkins Automation Server: jenkins.
- [ok] Starting network benchmark server.

5. Access the Fuego Jenkins web interface



6. Add a board

6-1. Set up communication to the target board

In order for Fuego to test a board, it needs to communicate with it from the host machine where Fuego is running. The most common way to do this is to use 'ssh' access over a network connection.

The target board needs to run an ssh server, and the host machine connects to it using the 'ssh' client.

For example:



6-2. Decide on user account for testing (creating one if needed)

On your target board, a user account is required in order to run tests.

For example:

use the root account

6-3. Create test directory on target

First, log in to your target board, and create a directory where tests can be run.

Usually, you do this as root, and a commonly used directory for this is "/home/fuego".

For example:



6-4. Create board file

Create board file by copy an existing one.

\$ cd fuego-ro/boards
\$ cp template-dev.board m3ulcb.board
\$ vi m3ulcb.board

A board file has parameters which define how Fuego interacts with your board.

For example:

IPADDR="192.168.10.20"		ip of target board(See 6-1)
LOGIN="root"		user account for testing (See 6-2)
PASSWORD=""		password of user account for testing (See 6-2)
BOARD_TESTDIR="/home/fuego)"	<- test directory on target(See 6-3)
TOOLCHAIN="m3ulcb"		toolchain of target(See 7-3)
TRANSPORT="ssh"		
ARCHITECTURE="arm64"		the architeture of target(See 6-1)

6-5. Add node to Jenkins interface

In the Jenkins interface, boards are referred to as "Nodes"



7. Add a toochain for board

You can skip this step when there is no cross-compiling in testsuites.

7-1. Install the SDK in the docker container

To allow fuego to use the SDK, you need to install it into the fuego docker container.

For example:

Copy m3ulcb's SDK into docker container.

 SDK:
 /poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-aarch64-toolchain-9.0.1.sh

 Compile m3ulcb's SDK:
 AGL Image and SDK

Install SDK:

\$ sudo docker exec -it fuego-container bash root@ubuntu-ESPRIMO-P720:/# ls /fuego-rw/poky-agl-glibc-x86 64-agl-demo-platform-crosssdk-aarch64-toolchain-9.0.1.sh /fuego-rw/poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-aarch64-toolchain-9.0.1.sh root@ubuntu-ESPRIMO-P720:/# sh /fuego-rw/poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-aarch64-toolchain-9.0.1.sh /bin/sh: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8) Automotive Grade Linux SDK installer version 9.0.1 Enter target directory for SDK (default: /opt/agl-sdk/9.0.1-aarch64): You are about to install the SDK to "/opt/agl-sdk/9.0.1-aarch64". Proceed[Y/n]? Y Extracting SDK.....done Setting it up.../bin/sh: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8) done /bin/sh: warning: setlocale: LC_ALL: cannot change locale (en US.UTF-8) SDK has been successfully set up and is ready to be used. Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g. \$./opt/agl-sdk/9.0.1-aarch64/environment-setup-aarch64-agl-linux root@ubuntu-ESPRIMO-P720:/# ls /opt/agl-sdk/9.0.1-aarch64/ environment-setup-aarch64-agl-linux sit -config-aarch64-agl-linux sys oots version-aarch64-agl-linux

7-2. Create a '*-tools.sh' file for the toolchain

Now, fuego needs to be told how to interact with the toolchain.

During test execution, the fuego system determines what toolchain to use based on the value of the TOOLCHAIN variable in the board file for the target under test. The TOOLCHAIN variable is a string that is used to select the appropriate '<TOOLCHAIN>-tools.sh' file in /fuego-ro/toolchains.

For example:m3ulcb-tools.sh

\$ cd fuego-ro/toolchains	
\$ vi m3ulcb-tools.sh	
Inside the -tools.sh file, you execute instructions that will set the environment variables r	eeded to build software with that SDI
SDKROOT=/opt/agl-sdk/9.0.1-aarch64/sysroots/aarch64-agl-linux <- th	e dir in docker container
ORIG_PATH=\$PATH	
PREFIX=aarch64-agl-linux	

source /opt/agl-sdk/9.0.1-aarch64/environment-setup-aarch64-agl-linux HOST=aarch64-agl-linux

<- the file in docker container

8. Add test jobs

There are two ways of adding test jobs, individually, and using testplans. In both cases, you use the 'ftc add-jobs' command. the 'ftc' command should be run in docker container.

For example:



9.Run test jobs

click 🔊 button

😌 Jenkins					6 🔍 se	arch		2
enkins >							ENABLE AUTO R	EFRESH
🚔 New Item		All	+				Madd o	lescriptio
Build History		S	w	Name ↓	Last Success	Last Failure	Last Duration	
Manage Jenkins			*	m3ulcb.bc-add.Functional.bc	N/A	N/A	N/A	\bigotimes
New View			*	m3ulcb.default.Functional.hello_world	N/A	N/A	N/A	Ð
Build Queue	_		*	m3ulcb.testplan_bc_add.batch	N/A	N/A	N/A	\bigotimes
No builds in the queue.		Icon: S	<u>5 M</u> L	Legend	I 🔊 RSS for all 🔊	RSS for failures	RSS for just lates	<u>builds</u>
Build Executor Status	-					6		
💻 master								
1 Idle								

10. Check test result.



refer to

1. http://fuegotest.org/wiki/FrontPage

2. https://bitbucket.org/fuegotest/fuego/src/master/

Construction of LAVA

1. introduction of LAVA

An automated validation architecture primarily aimed at testing deployments of systems based around the Linux kernel on ARM devices The project's web site is at: <u>https://validation.linaro.org/</u>

2. prepare for Fuego

To retrieve the Fuego software and create the docker image for it, you need to have git and docker installed on your system.



3 Installing LAVA with docker

3-1. Git c	lone this repo "https://github.com/kernelci/lava-docker"
	\$ git clone https://github.com/kernelci/lava-docker.git
3-2. Gene	rate configuration files for LAVA, udev, serial ports, etc. from boards.yaml
	\$ sudo apt-get install python-yaml
	\$./lavalab-gen.py
3-3 Go to	output/local directory
	\$ cd output/local/
2 4 0 11	1 1 ' '

3-4 Build docker images via \$ sudo apt install docker-compose \$ sudo docker-compose build

3-5 Start all images via \$ sudo docker-compose up -d

4. Access the LAVA web interface

4-1 You can access the LAVA web interface via "http://localhost:10080/". With the default users, you can login with "admin:admin".

4-2 If you want to use "http://your_ip:10080/"(like http://xxx.xxx.156:10080/) to accsee ILAVA web interface, use following cmds:



Welcome to LAVA

LAVA is an automated validation architecture primarily aimed at testing deployments of systems based around the Linux kernel on ARM devices, specifically ARMv7 and later. The current range of boards (device types) supported by this LAVA instance can be seen on the scheduler status page which includes details of how many boards of each type are available for tests and currently running jobs.

LAVA components

- ...I Results viewing results of tests run by you or others.
 Scheduler jobs are scheduled on available devices and the scheduler pages allow you to view current and past jobs as well as submit new jobs.
- **F** API information on how to interact with LAVA and export data from LAVA using XMLRPC.
- O Help documentation on using LAVA, worked examples and use cases, developing your own tests and how to adminster a LAVA instance of your own.
 Profile you are logged in as admin. Your profile provides access to jobs you have submitted or marked as favourites and details of devices owned by you.

Guides to LAVA Test using LAVA

Introduction to LAVA	More about LAVA & Linaro	O Use cases and worked examples.	• Writing a LAVA test definition.
O Administering a LAVA instance	O Developing LAVA	Construction a LAVA device.	
Your submissions		Your results	
Your jobs		Your results	al
Your favourite jobs			

5 Add user to LAVA

The LAVA frontend is developed using the Django web application framework and user authentication and authorization is based on the standard Django auth subsystems.

Local django user accounts can be created with the manage users command(all commands are run in the master docker):

5-1 If you want to add a new user.

\$ sudo lava-server manage users add <username> --passwd <password>

5-2 Set superuser rights

For example: username : admin

\$ sudo lava-server manage authorize_superuser --username admin User "admin" granted superuser rights

6 Add auth token

6-1 Find an unused token(e.g. dfjdfkfkdjfkdsjfsl) from the following url:

http://xxx.xxx.156:10080/admin/linaro_django_xmlrpc/authtoken/add/

OR http://xxx.xxx.156:10080/api/tokens/ Q LAVA 🕈 Home 📶 Results 🗸 🛗 Scheduler 🗸 🗲 API 🗸 🚱 Help

Instance: default

💄 admin 👻

LAVA / API tokens

Authentication Tokens

Authentication tokens allow scripts using lavacli and other scripts based on XML-RPC to securely access LAVA resources. You can create and use any number of tokens simultaneously. If you believe a token is compromised you can quickly remove it. Anyone using that token will no longer be able to authenticate as You in the system.

new 📀



Most recently created tokens shown first

Delete the 1 unused token 💼						
No.	Description	Created on	Last used			
1	empty	April 14, 2020	2 hours, 15 minutes			
2	no description	April 14, 2020	It was not used vet			

6	
	Hash for security token 2
Actions	no description dfjdfkfkdjfkdsjfs1

6-2 Add auth token for user admin(all commands are run in the master docker)

\$ apt install lava-tool \$ lava-tool auth-add http://admin@xxx.xxx.156:10080/RPC2/ Paste token for http://admin@xxx.xxx.156:10080/RPC2/:dfjdfkfkdjfkdsjfsl Token added successfully for user admin.

\$ lava-tool auth-list

Endpoint URL: http://xxx.xxx.156:10080/RPC2/ Tokens found for users: admin

6-3 Now set the user for this authentication as the default user for this endpoint, and set a shortcut for http://admin@xxx.xxx.156:10080/RPC2/ as admin



7 Use 'ser2net' daemon

ser2net provides a way for a user to connect from a network connection to a serial port, usually over telnet. http://ser2net.sourceforge.net/ ser2net is a dependency of lava-dispatcher, so will be installed automatically. Example config as below(in /etc/ser2net.conf): \$/etc/ser2net.conf

7001:telnet:600:/dev/ttyUSB0:115200 8DATABITS NONE 1STOPBIT banner

- 7001 is TCP/IP port to be monitored by the TCP server, binded to m3ulcb. - /dev/ttyUSB0 is the serial port name of the m3ulcb board connected to the LAVA server - 115200 is the serial baud rate.

8-1 Using the command line to list all known device types:

\$ lava-server manage device-types list --all

8-2 Add new known device types:

\$ lava-server manage device-types add r8a7796-m3ulcb

8-3 Add a device using a known device type

<worker-name>: "lab-slave-0"

<device name> : "m3ulcb"

\$ lava-server manage devices add --device-type r8a7796-m3ulcb --worker lab-slave-0 m3ulcb

(1)"lab-slave-0" is slaver name, you can see from web interface.



(2)"m3ulcb" is the device name will be displayed in LAVA interface. We can find that the status of m3ulcb is "bad". Click "m3ulcb" to see details. The reason is that device "m3ulcb" missing a template called "m3ulcb.jinja2" file in "/etc/lava-server/dispatcher-config/devices" in master docker. Q LAVA ↑ Home II Results ▼ Scheduler ▼ ↑ API ▼ O Help

-						
	Status					
Welcome to LAVA	Reports	Q LAVA A Home	📶 Results 👻 🛗 Schedule	er 🗸 🔸 API 🗸 🔞 Help		Instance: default
LAVA is an automated validation architecture	Device Types	LAVA / Scheduler / Devi	ices			
current range of boards (device types) support for tests and currently running jobs.	Devices					
	Jobs	All Devices				
LAVA components	Workers				fut pt	
Il Results - viewing results of tests ru Scheduler - jobs are scheduled on	Submit scheduler pages all		₩ All Devices	Active Devices	evices Maintenance Devices	
• # API - information on how to interact v	with LAVA and export data from LAVA using >	Show 25 v entries			Search	?
 Help - documentation on using LAVA Profile - you are logged in as admin 	 A, worked examples and use cases, developing Your profile provides access to jobs you have 	Hostname 🗄	Worker Host ‡†	Device Type 11	state 11 Health	↓† Tags ↓†
	, , , , , , , , , , , , , , , , , , , ,	m3ulcb	lab-slave-0	r8a7796-m3ulcb	Bad	
Guides to LAVA						
Introduction to LAVA	More about LAVA & Linaro	Output Use cases and wold wold wold wold wold wold wold wol		a LAVA test definition.		
O Administering a LAVA instance	Developing LAVA		Deculto - 🗮 Schodulor -			Inctanço: dofault
Your submissions		LAVA / Scheduler / Devices	s / m3ulcb			Help
	_	Configuration Error: missing	g or invalid template.			×
Your jobs	8	Jobs requesting this device ty	rpe (r8a7796-m3ulcb) will not b	e able to start until a template is available on	the master.	
		The device has no health ch	neck job Consider adding a be	ealth check definition		×
		← Previous device →				Next Device ►
		Hostname m3	sulch 🗲 📶	State Idle	Physical ac	cess
		Tags		Worker diotionany	Ver	rsion
				Device dictionary		
8-4 Adding a template(*.jinja2) to th	ne device m3ulcb					
More abort the templete, refer to	o https://validation.linaro.org/static/	/docs/v2/first-devices.html				
	https://validation.linaro.org/st	tatic/docs/v2/simple-admin.ht	ml#overriding-device-	configuration		
\$ cd /etc/lava-server/disp	oatcher-config/devices					
$v_1 m_3 ulcb_j n_j a^2$	4-90.7706 m ² which $3-10/2$					
{% extends	r_{6a}/r_{96} = r_{6a}/r_{96}	1 7001'%		#use "telpet 172 17 0 1 7	001" to connect board	
		.1 /001 /05		"7001" is setted in step	7 and "172.17.0.1" is ipaddr o	of test PC
{% set hard	reset command = '/bin/sh -c /newd	lisk/LAVA/relay serial/reboo	t m3.sh' %}	#reset board command		
{% set powe	er_on_command = '/bin/sh -c /newdi	isk/LAVA/relay_serial/power		#power on board comma	nd	
{% set powe	er_off_command = '/bin/sh -c /newd	lisk/LAVA/relay_serial/power	r_off_m3.sh' %}	#power off board comma	nd	
{% set uboo	t_ipaddr_cmd = 'setenv serverip 192	2.168.10.10; setenv ipaddr 19	2.168.10.20' %}	#set board ipaddr=192.16	8.10.20	
				set lava server ipaddr=19	02.168.10.10	
the status of m3ulcb con	Results - Scheduler - 4			Instance: default	👤 admin 👻	
				mstance. delaut		
LAVA / Scheduler / Dev	vices / m3ulcb				Help	
				Next	Device 🍉	
Liestner	m2ulch 🖌 🖢	Céséa Idla)	Physical access		
Device-type	r8a7796-m3ulcb	Health Good X		Description		
Tags		Worker lab-slave-0		Version		
	1	Device dictionary	/			

Recent jobs on	n m3ulcb				Search	2
ID Actions	State	Description 11	Submitter 11	Submit Time ↓7	End Time ‡†	Duration
Transitions Show 25 • entries						
Action Time 17		User 11	Reason			
April14, 8:26a.m.		admin	$Bad \to Good$			
April14, 8:12a.m.		lava-health	Good \rightarrow Bad (Inva	alid device configuration)		

9 Submit a job



job.name: AGL-test kernel.tree: AGL-yocto kernel.version: change-22746-1 kernel.defconfig_base: defconfig kernel.defconfig: AGL-m3ulcb-nogfx platform.mach: renesas platform.name: r8a7796-m3ulcb test.plan: agl-testplan device.type: r8a7796-m3ulcb job.build_environment: AGL-yocto device_type: r8a7796-m3ulcb job_name: AGL-test timeouts: job: minutes: 120 action: minutes: 15 connection: minutes: 10

9-3 Yaml file is the job definition, more details refer to https://validation.linaro.org/static/docs/v2/first-job.html agl-test.yaml: metadata:

> request: set_port kernel: url: http://download.automotivelinux.org/AGL/release/icefish/9.0.0/m3ulcb-nogfx/deploy/images/m3ulcb/Image type: image initrd: url: http://download.automotivelinum.org/AGL/release/icefish/9.0.0/m3ulcb-nogfx/deploy/images/m3ulcb/Image

- action: nbd-deploy

failure_retry: 2 protocols: lava-xnbd:

git.branch: halibut

job.arch: arm64

connections: lava-test-shell: minutes: 10 priority: medium visibility: public protocols: lava-xnbd: port: auto actions: - deploy: timeout: minutes: 60 to: nbd os: oe

git.commit: F8d626c53-L4deb709e image.type: AGL-release-9.0.0

url: http://download.automotivelinux.org/AGL/release/icefish/9.0.0/m3ulcb-nogfx/deploy/images/m3ulcb/initramfs-netboot-image-m3ulcb.ext4.gz allow_modify: false

nbdroot:

url: http://download.automotivelinux.org/AGL/release/icefish/9.0.0/m3ulcb-nogfx/deploy/images/m3ulcb/agl-image-ivi-crosssdk-m3ulcb.ext4.xz compression: xz dtb:

url: http://download.automotivelinux.org/AGL/release/icefish/9.0.0/m3ulcb-nogfx/deploy/images/m3ulcb/r8a7796-m3ulcb.dtb

- 5001.
timeout:
minutes: 15
method: u-boot
prompts:
- "root@m3ulcb:~"
- "m3ulcb:~#"
- '/ #'
auto_login:
login_prompt: "login:"
username: root
commands: nbd
type: booti
transfer_overlay:
download_command: ifconfig eth0 192.168.10.22 ; wget
unpack_command: tar -C / -xvpf
- test:
timeout:
minutes: 15
definitions:
- repository:
metadata:
format: Lava-Test Test Definition 1.0
name: inline-test-host
description: "Inline test to validate test framework health"
OS:
- debian
scope:
- functional
run:
steps:
- ls /
- uname -a
- df -h
from: inline
name: health-test
path: inline/health-test.yaml

10. Check test result.



[nbd-deploy] Checking protocol data for lava-xnbd
Get a port from pool
Set_port 61995
start: 1.1 download-retry (timeout 01:00:00) [common]
start: 1.1.1 file-download (timeout 01:00:00) [common]
downloading file:///home/image/m3ulcb/initramfs-netboot-image-m3ulcb.ext4.gz
saving as /var/lib/lava/dispatcher/tmp/28/nbd-deploy-b7gor3tr/initrd/initramfs-netboot-image-m3ulcb.ext4.gz
total size: 2053053 (1MB)
No compression specified
progress 1% (0HB)
progress 6% (OMB)
progress 11% (OMB)
progress 17% (OMB)
progress 22% (OMB)
progress 27% (OMB)
progress 33% (OMB)
progress 38% (OMB)
progress 43% (OMB)
progress 49% (OMB)
progress 54% (1/MB)
progress 59% (1MB)
progress 65% (1MB)
progress 70% (1MB)
progress 75% (1MB)
progress 81% (1MB)
progress 86% (1MB)
progress 92% (1MB)
progress 97% (1MB)
1MB downloaded in 0.02s (128.75MB/s)
end: 1.1.1 file-download (duration 00:00:00) [common]
case: file-download
case id: 257

refer to

https://validation.linaro.org/static/docs/v2/contents.html
 https://github.com/kernelci/lava-docker

		AGL Image and SDK
env		
	board:	Renesas R-Car M3
	AGL version:	Itchy Icefish v9.0.1

overview

The AGL image development workflow consists of setting up the system (i.e. the build host) that builds the image and finishes with using the Yocto Project to create an image targeted towards specific hardware. The following figure and list overview the AGL image development process.



1. Prepare your build host to be able to use the tools needed to build your image.

Preparing your build host so that it can build an AGL image means making sure your system is set up to use the Yocto Project OpenEmbedded build system

- -1)Use a Supported Linux Distribution
- -2)Be Sure Your Build Host Has Enough Free Disk Space: Your build host should have at least 50 Gbytes.
- -3)Be Sure Tools are Recent: You need to have recent versions for the following tools:
- Git 1.8.3.1 or greater
- Tar 1.27 or greater
- Python 3.4.0 or greater
- -4)Install Essential, Graphical, and Eclipse Plug-in Build Host Packages

Usually, a normal ubuntu environment is fine

2. Download the AGL software into a local Git repository on your build host. Renesas R-Car ivi series need to build manually, others can be downloaded from following: https://download.automotivelinux.org/AGL/release/ Download the release image/sdk. If do this you can skip all steps below. $(e \neq IIP2 \text{ icefish } 9.0.1)$

\$ wget https://download.automotivelinux.org/AGL/release/icefish/9.0.1/intel-corei7-64/deploy/images/intel-corei7-64/agl-demo-platform-crosssdk-intel-corei7-64.wic.xz	motivelinux.org/AGL/release/icefish/9.0.1/intel-corei7-64/deploy/images/intel-corei7-64/agl-demo-platform-crosssdk-intel-corei7-64.wic.xz
Usually, download *.wic.xz	
SDK (e.g. UP2 icefish 9.0.1)	$(\mathcal{Y}.0.1)$
\$ wget https://download.automotivelinux.org/AGL/release/icefish/9.0.1/intel-corei7-64/deploy/sdk/poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-corei7-64-toolchaited and the set of the	motivelinux.org/AGL/release/icefish/9.0.1/intel-corei7-64/deploy/sdk/poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-corei7-64-toolchain-9.0.1.sh
Usually, download *.sh	

3 Download the latest source code from git

3-1.Prepare Repo:
\$ mkdir ~/bin
\$ export PATH=~/bin:\$PATH
\$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
\$ chmod a+x ~/bin/repo

3-2.download code from master: \$ export AGL_TOP=/work/agl-test/icefish-9.0.1 \$ cd \$AGL_TOP \$ repo init -b icefish -m icefish_9.0.1.xml -u https://gerrit.automotivelinux.org/gerrit/AGL/AGL-repo \$ repo sync

4. Run the build environment script to initialize variables and paths needed for the build.

For example: m3ulcb \$ export XDG_DOWNLOAD_DIR=\$HOME/Downloads \$ export MACHINE=m3ulcb \$ cd \$AGL TOP \$ source meta-agl/scripts/aglsetup.sh -m \$MACHINE -b build agl-devel agl-demo agl-netboot agl-appfw-smack agl-localdev \$ grep -w -e "^MACHINE =" \$AGL_TOP/build/conf/local.conf

5. Make sure your build configuration is defined exactly how you want it for your build. usually, we need to do nothing during this step

6. Build Images.

\$ bitbake agl-demo-platform

The build process puts the resulting image in the Build Directory: For example: \$AGL_TOP/build/tmp/deploy/images/\$MACHINE/

7. Build SDK

Download proprietary drivers from the R-Car H3/M3 Software library and Technical document site into '/Downloads'. Refer to the following url to download Proprietary Drivers https://docs.automotivelinux.org/docs/en/master/getting_started/reference/getting-started/machines/renesas.html

Build

\$ bitbake agl-demo-platform-crosssdk

The SDK installer file (*.sh) is placed in the build directory. For example:

 $\label{eq:add_top} AGL_TOP/build/tmp/deploy/sdk/poky-agl-glibc-x86_64-agl-demo-platform-crosssdk-aarch64-toolchain-9.0.1.sh$

refer to

https://wiki.automotivelinux.org/agl-distro/release-notes?s[]=repo&s[]=init#itchy_icefish_v901 https://docs.automotivelinux.org/docs/en/master/getting_started/reference/getting-started/machines/renesas.html https://docs.automotivelinux.org/docs/en/master/getting_started/reference/getting-started/app-workflow-sdk.html https://docs.automotivelinux.org/docs/en/master/getting_started/reference/getting-started/image-workflow-intro.html