AVNET
MANUFACTURING SERVICES

MaaXBoard

(EM-MC-SBC-IMX8M)

Linux Software

Development Guide

V1.0

Copyright Statement:

- The MaaXBoard single board computer (partnumber: EM-MC-SBC-IMX8M) and its related intellectual property are owned by Avnet Manufacturing Services.
- Avnet Manufacturing Services has the copyright of this document and reserves all rights. Any part of the document should not be modified, distributed or duplicated in any approach and form with the written permission issued by Avnet Manufacturing Services.

Disclaimer:

- Avnet Manufacturing Services does not take warranty of any kind, either expressed or implied, as to the program source code, software and documents provided along with the products, and including, but not limited to, warranties of fitness for a particular purpose; The entire risk as to the quality or performance of the program is with the user of products.

Regulatory Compliance:

- MaaXBoard single board computer has passed the CE & FCC certification.

Revision History

Version	Note	Author	Release Date
V1.0	Initial version	Sandy	20190411

Catalog

Chapter 1 Build Environment Setup

1.1 Setup Build Environment

To setup the build environment need:

- Hardware: At least 20GB of disk space and 2GB of RAM
- Software: Ubuntu 64bit OS, 14.04 LTS version or later LTS version (Ubuntu Desktop or Ubuntu Server version). You could also run the Ubuntu 64 bit OS on virtual machine.

The following packages are required for the development environment. The required packages can be installed using the bash script below:

```
sudo apt install git
sudo apt install kpartx
sudo apt install zlib1g-dev
sudo apt install device-tree-compiler
```

Note: You may need to run **sudo apt update** first if the installation failed or can't find specific packages.

1.2 Configure Build Environment

Copy all the content of 02Linux under release folder to the $HOME path in Ubuntu, extract the rar files. The compilation tool gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz locate under path $HOME/03LinuxTools. Use the following instructions to extract it:

```
$tar -Jxvf gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
```

Import the environment variable:

```
$export
CROSS_COMPILE=$HOME/03LinuxTools/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/
aarch64-linux-gnu-
$export ARCH=arm64
```

Note: the environment variable need to be imported before each build.

Chapter 2 Compile Source Code

2.1 UBOOT

2.1.1 Get the U-Boot Source Code

U-boot source code locates under path $HOME/01LinuxSourceCode, extract the u-boot*.tar.gz:

```
$ cd $HOME/01LinuxSourceCode
$ tar -zxvf u-boot*.tar.gz
```

2.1.2 Compile U-Boot Image

```
$ make em_sbc_imx8m_defconfig
$ make
$ cp -f spl/u-boot-spl.bin tools/imx-boot/iMX8M/
$ cp -f u-boot-nodtb.bin tools/imx-boot/iMX8M/
$ cp -f arch/arm/dts/em-sbc-imx8m.dtb tools/imx-boot/iMX8M/
$ cd tools/imx-boot/
$ make clean
$ make SOC=iMX8M flash_ddr4_em
$ cd ../../
$ cp -f ./tools/imx-boot/iMX8M/flash.bin    u-boot.imx
```

When the compilation finished, it will generate a **u-boot.imx** under path $HOME/01LinuxSourceCode/u-boot. Copy this file to output/ for further use.

2.2 Kernel

2.2.1 Get the Kernel Source Code

Kernel source code locates under path $HOME/01LinuxSourceCode, extract the linux*.tar.gz:

```
$ tar -zxvf linux*.tar.gz
```

2.2.2 Compile Image

```
$ cd $HOME/01LinuxSourceCode/linux
$ make distclean
$ make em-sbc-imx8m_defconfig
$ make
```

When the compilation finished, it will generate

- Under path $HOME/01LinuxSourceCode/linux/arch/arm64/boot: Image
- Under path $HOME/01LinuxSourceCode/linux/arch/arm64/boot/dts/freescale/:
  - em-sbc-imx8m.dtb
  - em-sbc-imx8m-dcss-dsi.dtb
  - em-sbc-imx8m-dcss-lvds.dtb
  - em-sbc-imx8m-usb0-device.dtb

Copy these files to output path for further use.

2.2.3 Compile Modules

After Compile Image, you need to compile some module drivers, eg:

```
$ make modules_install INSTALL_MOD_PATH=~/output/modules/
```

This operation will generate a lib folder under path ~/output/modules/, we will use these files in burning process.

Chapter 3 Burn the Image

The default version of MaaXBoard support SD Card. Avnet Manufacturing Services also provide eMMC version for users to customize. To burn the system image to the eMMC, refer to MaaXBoard EMMC Burning Guide.

3.1 Partition

Before burn a blank SD Card (or eMMC), we need to partition the storage device. If your SD Card (or eMMC) has already been burned the shipment Image, you can jump this process.

3.1.1 Partition the SD Card

Connect SD card to Linux system, if the SD card is automatically mounted, enter the following instruction to umount it: (the value **sdb** should be replaced by the actual node of SD Card under Linux system)

```
$ sudo umount /dev/sdb*
```

Partition the SD card use following instructions:

```
$ echo -e "o\nn\np\n1\n20480\n+256M\na\nt\nc\nn\np\n2\n544768\n\nw\n" | sudo fdisk /dev/sdb
$ sudo mkfs.vfat /dev/sdb1
$ sudo mkfs.ext4 /dev/sdb2
```

3.2 Burn U-boot Image

3.2.1 Burn Image to SD Card

Connect SD card to Linux system, then enter the following instruction to burn the U-boot image: (the value **sdb** should be replaced by the actual node of SD Card under Linux system)

```
$ sudo dd if=u-boot.imx of=/dev/sdb bs=1k seek=33 conv=fsync
```

3.3 Burn Kernel Image

3.3.1 Burn Image to SD Card

Copy the Image and dtb files to the first partition of SD Card.

3.4 Copy uEnv.txt

The uEnv.txt is the configuration file for system boot

Refer to the operation of burn kernel image, copy this file to the first partition of SD Card.

Content of uEnv.txt:

```
fdt_file=em-sbc-imx8m.dtb
#fdt_file=em-sbc-imx8m-dcss-lvds.dtb
#fdt_file=em-sbc-imx8m-dcss-dsi.dtb
#fdt_file=em-sbc-imx8m-usb0-device.dtb
console=ttymxc0,115200 console=tty1 fbcon=rotate:0
```

The value of fdt_file should be modified according to the factual boot up situation.

3.5 Burn File System

3.5.1 Burn File System to SD Card

Burn the img file of file system to the second partition of SD Card use dd command, for example:

```
dd if= debian-9.x-arm64-2019-02-28.img of=/dev/sdb2
```

3.5.2 Copy Modules

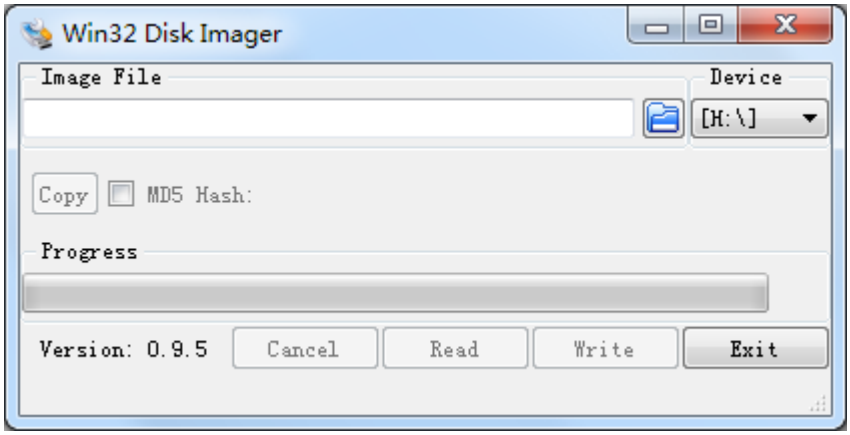
In previous process, we have compile the modules to /output/modules/ folder, now we need to copy them to the file system, execute the following instructions:

```
$ mkdir ~/fs_mountpoint
$ sudo mount /dev/sdb2 ~/fs_mountpoint
$ sudo cp -af ~/output/modules/* ~/fs_mountpoint/
$ sudo sync
$ sudo umount /dev/sdb2
```

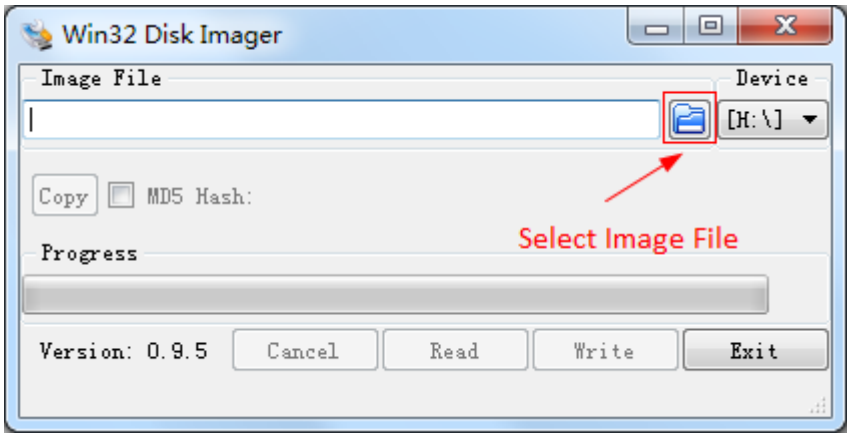
3.6 Read Entire Image

3.6.1 Read the Image from SD Card

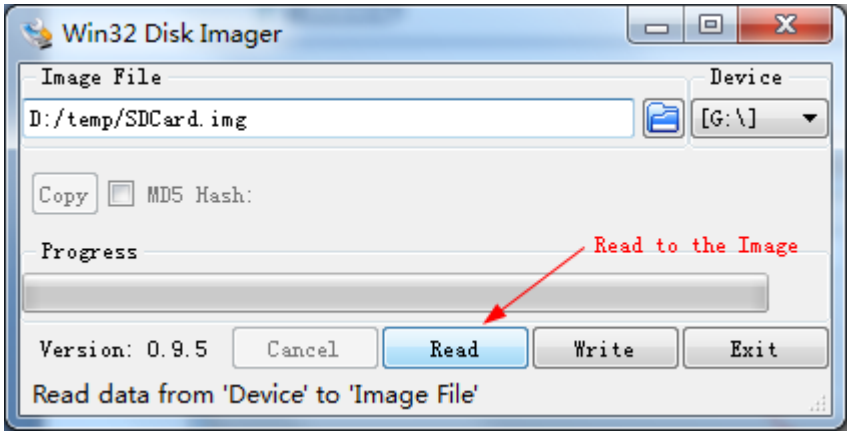
1. Connect the SD Card to Windows system, then run Win32 Disk Imager.



2. Select the destination of image file, such as: D:/temp/SDCard.img.



3. Click “Read” button to read the content of SD Card to img file.



When the progress finished successfully, you will get an entire SD Card Image.

Chapter 4 User Defined Module (TBD)

4.1 Compile a Hello world Project

This chapter will introduce how to write and compile a simple helloworld project.

4.1.1 Test Code

```
embest@compiler:~/compile/test $ cat hello_world.c
#include <stdio.h>

void main(void)
{
        printf("hello world!\n");
      }
```

4.1.2 Makefile

```
#cross compile tools path :
CROSS_COMPILE:=/home/embest/tool/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
ARCH:= arm64
CC:=$(CROSS_COMPILE)gcc
LD:=$(CROSS_COMPILE)ld
#.c files
objects = hello_world.o
# output executable file
hello_world: $(objects)
            $(CC) -O2 -static -o hello_world $(objects)
clean:
            @rm -vf *hello_world *.o *~
```

4.1.3 Compile and Test

Execute **make** command to compile the file.

```
embest@compiler:~/compile/test$ make
/home/embest/tool/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc -c -o hello_world.o hello_world.c
```

```
/home/embest/tool/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc -O2 -static -o hello_world hello_world.o
```

When compile finished, you will get two project, .o for compile object file, hello_world for executable file.

```
embest@compiler:~/compile/test$ ls hello_world*
hello_world    hello_world.c    hello_world.o
embest@compiler:~/compile/test$ file hello_world
hello_world: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), statically linked, for GNU/Linux 3.7.0, BuildID[sha1]=c5fa2e21b75b6a8afdaa0e4f581bfa9c807bd9bb, with debug_info, not stripped
```

Chapter 5 Appendix

5.1 Hardware

For the detail hardware introduction, please refer to MaaXBoard Hardware user manual.

Chapter 6 Technical Support and Warranty

6.1 Technical Support

Avnet Manufacturing Services provides its product with one-year free technical support including:

- Providing software and hardware resources related to the embedded products of Avnet Manufacturing Services;
- Helping customers properly compile and run the source code provided by Avnet Manufacturing Services;
- Providing technical support service if the embedded hardware products do not function properly under the circumstances that customers operate according to the instructions in the documents provided by Avnet Manufacturing Services;
- Helping customers troubleshoot the products.
- The following conditions will not be covered by our technical support service. We will take appropriate measures accordingly:
  - Customers encounter issues related to software or hardware during their development process;
  - Customers encounter issues caused by any unauthorized alter to the embedded operating system;
  - Customers encounter issues related to their own applications;
  - Customers encounter issues caused by any unauthorized alter to the source code provided by Avnet Manufacturing Services.

6.2 Warranty Conditions

- 12-month free warranty on the PCB under normal conditions of use since the sales of the product;
- The following conditions are not covered by free services; Avnet Manufacturing Services will charge accordingly:
  - Customers fail to provide valid purchase vouchers or the product identification tag is damaged, unreadable, altered or inconsistent with the products;
  - Not according to the user's manual operation causes damage to the product;
  - Products are damaged in appearance or function caused by natural disasters (flood, fire, earthquake, lightning strike or typhoon) or natural aging of components or other force majeure;
  - Products are damaged in appearance or function caused by power failure, external forces, water, animals or foreign materials;
  - Products malfunction caused by disassembly or alter of components by customers or, products disassembled or repaired by persons or organizations unauthorized by Avnet Manufacturing

Services, or altered in factory specifications, or configured or expanded with the components that are not provided or recognized by Avnet Manufacturing Services and the resulted damage in appearance or function;

- Product failures caused by the software or system installed by customers or inappropriate settings of software or computer viruses;
- Products purchased from unauthorized sales;
- Warranty (including verbal and written) that is not made by Avnet Manufacturing Services and not included in the scope of our warranty should be fulfilled by the party who committed. Avnet Manufacturing Services has no any responsibility.

- Within the period of warranty, the freight for sending products from customers to Avnet Manufacturing Services should be paid by customers; the freight from Avnet Manufacturing Services to customers should be paid by us. The freight in any direction occurs after warranty period should be paid by customers;
- Please contact technical support if there is any repair request.

- **Avnet Manufacturing Services will not take any responsibility on the products sent back without the permission of the company.**

Chapter 7 Contact Information

- Tel: +86-755-33190846/33190847/33190848
- E-mail:
  - Technical support: support@embest-tech.com
  - Sales contact: chinasales@embest-tech.com
- Fax: +86-755-25616057
- Website: http://www.embest-tech.com/
- Address: Tower B 4/F, Shanshui Building, Nanshan Yungu Innovation Industry Park, Liuxian Ave.No.4093,Nanshan District, Shenzhen, Guangdong, China