**CM-AM62x / CX-AM62x Manual**

**Revision 1.4**

**2024. 1. 16**

[**CRZ Technology**](http://www.crz-tech.com/)

**Document History**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Change note** |
| **1.0** | **2023.08.14** | **Created** |
| **1.1** | **2023.10.11** | **Updated** |
| **1.2** | **2023.11.22** | **Removed unnecessary files** |
| **1.3** | **2024.01.05** | **Removed SDK PATH configuration** |
| **1.4** | **2024.01.16** | **Added figure of debug cable connection** |

**Index**

[1. 개요 4](#_Toc156318540)

[2. 호스트 빌드 환경 4](#_Toc156318541)

[3. U-boot 빌드 4](#_Toc156318542)

[4. Kernel 빌드 4](#_Toc156318543)

[5. Root file system 빌드 4](#_Toc156318544)

[6. SD Boot 5](#_Toc156318545)

[7. NAND Boot 6](#_Toc156318546)

[8. eMMC Boot 6](#_Toc156318547)

[9. U-Boot & Kernel 7](#_Toc156318548)

[10. Led 테스트 7](#_Toc156318549)

[11. Key 테스트 7](#_Toc156318550)

[12. USB 디바이스 테스트 8](#_Toc156318551)

[13. SOUND 테스트 8](#_Toc156318552)

[14. WIFI 테스트 8](#_Toc156318553)

[15. GPU 테스트 10](#_Toc156318554)

[16. LCD 지원 11](#_Toc156318555)

[17. eMMC 이미지 Update 12](#_Toc156318556)

# **개요**

이 문서에서는 CM-AM62x / CX-AM62x 보드를 위한 리눅스 빌드 방법과 디바이스 테스트 방법을 기술한다.

# **호스트 빌드 환경**

호스트 빌드 환경은 Ubuntu 22.04 LTS를 이용한다.

아래 링크에서 2가지 툴체인을 가져와서 /opt/에서 압축을 해제한다.

|  |
| --- |
| [arm tool chain](https://drive.google.com/drive/folders/1LGdMmywEq-KxeDoMK93fZXkjJ8l9Pj-e?usp=sharing) |

아래 링크에서 AM62x 리눅스 빌드용 소스를 가져온다.

|  |
| --- |
| [am62x\_linux\_src\_20231122.tar.gz](https://drive.google.com/file/d/13AFftOa9z3OSzfMH_CL-M0MoFeEUA2H-/view?usp=sharing) |

아래 링크에서 AM62x 리눅스 prebuilt 이미지를 가져올 수 있다.

|  |
| --- |
| [am62x\_linux\_img\_20231122.tar.gz](https://drive.google.com/file/d/13K6bBMerN0LXfDId8ePNfqa8NYvp4ljT/view?usp=sharing) |

# **U-boot 빌드**

am62x\_linux\_src\_20231122.tar.gz 소스를 압축해제한다.

u-boot를 빌드하기 위해서 아래 명령을 입력한다.

|  |
| --- |
| $ ./build.sh uboot |

# **Kernel 빌드**

Linux kernel을 빌드하기 위해서 아래 명령을 입력한다.

|  |
| --- |
| $ ./build.sh kernel |

# **Root file system 빌드**

Root file system을 빌드하기 위해서 아래 명령을 입력한다.

|  |
| --- |
| $ ./build.sh rootfs |

프롬프트가 나오면 아래 명령을 입력한다.

|  |
| --- |
| # ./create\_ubuntu20.04\_arm.sh |

타임존 화면에서 Asia / Seoul을 선택한다.

Keyboard에서 Korean/Korean을 선택한다.

Dash 설정화면에서 “NO”를 선택한다.

프롬프트가 나오면 “exit”을 입력한다.

|  |
| --- |
| # exit |

u-boot / kernel / root file system을 한번에 빌드하기 위해서는 아래 명령을 입력한다.

|  |
| --- |
| $ ./build.sh |

u-boot / kernel을 clean하기 위해서는 아래 명령을 입력한다.

|  |
| --- |
| $ ./build.sh clean |

# **SD Boot**

uSD 카드(8GB 이상)를 uSD 카드 어댑터를 통하여 호스트 Ubuntu에 연결한다.

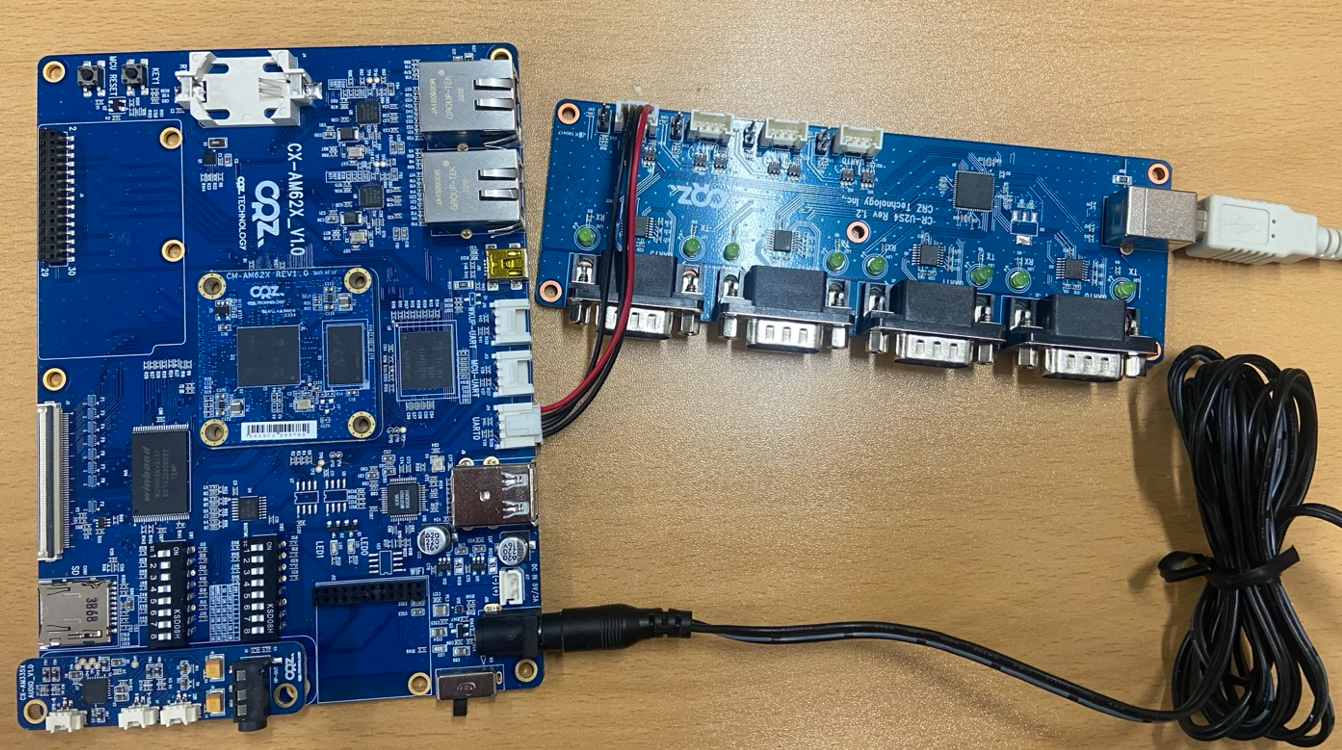
부팅가능한 uSD 카드를 만들기 위해서 image 폴더에서 “sudo ./create-sdcard\_ubuntu.sh”를 실행한다.

|  |
| --- |
| $ cd image  $ sudo ./create-sdcard\_ubuntu.sh |

SW1을 ON-ON-ON-OFF(1-2-3-4)로 설정한다.

uSD를 슬롯에 넣고 보드에 5V 파워를 인가하고 시리얼 터미널에 로그가 나오는지 확인한다.

Debug 케이블은 J14 UART0에 연결한다. 아래는 Debug 케이블 연결 사진이다.



PC와 연결시 COM 포트가 4개 생성이 되는데 로그가 나오는 포트가 있는지 확인한다.

# **NAND Boot**

SD boot를 확인한 uSD 카드를 다시 호스트에 연결한다.

아래 명령을 실행하여 NAND boot를 가능하게 해 주는 uSD를 만든다.

|  |
| --- |
| $ sudo ./make\_auto\_NAND\_fuse\_SD.sh |

SW1을 ON-ON-ON-OFF(1-2-3-4)로 설정한다.

uSD를 슬롯에 넣고 보드에 5V 파워를 인가하고 로그인하면 NAND가 설정된다.

보드 파워를 끄고 uSD 카드를 제거한다.

SW1을 OFF-OFF-ON-OFF(1-2-3-4)로 설정한다.

보드에 5V 파워를 인가하고 NAND boot가 되는지 확인한다.

# **eMMC Boot**

SD boot를 확인한 uSD 카드를 다시 호스트에 연결한다.

아래 명령을 실행하여 eMMC boot를 가능하게 해 주는 uSD를 만든다.

|  |
| --- |
| $ sudo ./make\_auto\_eMMC\_fuse\_SD.sh |

SW1을 ON-ON-ON-OFF(1-2-3-4)로 설정한다.

uSD를 슬롯에 넣고 보드에 5V 파워를 인가하고 로그인하면 eMMC가 설정된다.

보드 파워를 끄고 uSD 카드를 제거한다.

SW1을 OFF-ON-ON-OFF(1-2-3-4)로 설정한다.

보드에 5V 파워를 인가하고 eMMC boot가 되는지 확인한다.

# **U-Boot & Kernel**

Ubuntu 호스트에 NFS가 설정된 경우, u-boot / kernel을 콘솔에서 쉽게 교체할 수 있다.

SD boot의 경우 u-boot / kernel을 교체하기 위해 아래 명령을 수행한다.

|  |
| --- |
| $ ./tmp\_uboot\_update\_SD.sh  $ ./tmp\_kernel\_update\_SD.sh |

NAND boot의 경우 u-boot / kernel을 교체하기 위해 아래 명령을 수행한다.

|  |
| --- |
| $ ./tmp\_uboot\_update\_NAND.sh  $ ./tmp\_kernel\_update\_NAND.sh |

eMMC boot의 경우 u-boot / kernel을 교체하기 위해 아래 명령을 수행한다.

|  |
| --- |
| $ ./tmp\_uboot\_update\_eMMC.sh  $ ./tmp\_kernel\_update\_eMMC.sh |

# **Led 테스트**

아래 명령을 수행하여 LED1이 동작하는지 확인한다.

|  |
| --- |
| root@mango:~# ./led1\_test.sh |

# **Key 테스트**

아래 명령을 수행하고 KEY1을 눌려 키가 인식되는지 확인한다.

|  |
| --- |
| root@mango:~# evtest  No device specified, trying to scan all of /dev/input/event\*  Available devices:  /dev/input/event0: TSC2007 Touchscreen  /dev/input/event1: gpio-keys  Select the device event number [0-1]: 1  Input driver version is 1.0.1  Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100  Input device name: "gpio-keys"  Supported events:  Event type 0 (EV\_SYN)  Event type 1 (EV\_KEY)  Event code 256 (BTN\_0)  Key repeat handling:  Repeat type 20 (EV\_REP)  Repeat code 0 (REP\_DELAY)  Value 250  Repeat code 1 (REP\_PERIOD)  Value 33  Properties:  Testing ... (interrupt to exit)  Event: time 1690183095.082755, type 1 (EV\_KEY), code 256 (BTN\_0), value 1  Event: time 1690183095.082755, -------------- SYN\_REPORT ------------  Event: time 1690183095.222793, type 1 (EV\_KEY), code 256 (BTN\_0), value 0  Event: time 1690183095.222793, -------------- SYN\_REPORT ------------ |

# **USB 디바이스 테스트**

SD boot를 확인한 uSD 카드를 삽입하고 J10에 mini USB type B 단자를 연결하고 USB type A 단자는 Windows 호스트에 연결한다.

아래 명령을 실행하여 Windows 호스트에서 USB mass storage가 인식되는지 확인한다.

|  |
| --- |
| root@mango:~# ./usb\_device\_test.sh |

# **SOUND 테스트**

아래 명령을 실행하여 SOUND가 잘 동작하는지 확인한다.

|  |
| --- |
| root@mango:~# ./sound\_test.sh |

# **WIFI 테스트**

아래 명령을 실행하여 USB WIFI가 인식되는지 확인한다.

|  |
| --- |
| root@mango:~# lsusb  Bus 001 Device 003: ID 0bda:8179 Realtek Semiconductor Corp. RTL8188EUS 802.11n Wireless Network Adapter |

Wlan이 인식되었는지 확인한다.

|  |
| --- |
| root@mango:~# ifconfig -a  eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  inet 192.168.1.7 netmask 255.255.255.0 broadcast 192.168.1.255  inet6 fe80::1e63:49ff:fe1a:6a50 prefixlen 64 scopeid 0x20<link>  ether 1c:63:49:1a:6a:50 txqueuelen 1000 (Ethernet)  RX packets 519 bytes 624707 (624.7 KB)  RX errors 0 dropped 0 overruns 0 frame 0  TX packets 319 bytes 23825 (23.8 KB)  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  inet 192.168.1.36 netmask 255.255.255.0 broadcast 192.168.1.255  inet6 fe80::2805:39ff:fe57:c74a prefixlen 64 scopeid 0x20<link>  ether 2a:05:39:57:c7:4a txqueuelen 1000 (Ethernet)  RX packets 105 bytes 8308 (8.3 KB)  RX errors 0 dropped 0 overruns 0 frame 0  TX packets 74 bytes 5765 (5.7 KB)  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  inet 127.0.0.1 netmask 255.0.0.0  inet6 ::1 prefixlen 128 scopeid 0x10<host>  loop txqueuelen 1000 (Local Loopback)  RX packets 102 bytes 8188 (8.1 KB)  RX errors 0 dropped 0 overruns 0 frame 0  TX packets 102 bytes 8188 (8.1 KB)  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  wlx0432f41f15b2: flags=4098<BROADCAST,MULTICAST> mtu 1500  ether 04:32:f4:1f:15:b2 txqueuelen 1000 (Ethernet)  RX packets 0 bytes 0 (0.0 B)  RX errors 0 dropped 0 overruns 0 frame 0  TX packets 0 bytes 0 (0.0 B)  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 |

“wlx0432f41f15b2” device가 인식된 것을 확인한다.

|  |
| --- |
| ifconfig wlx0432f41f15b2 up  iwlist wlx0432f41f15b2 scanning => 공유기 essid 있는지 검색  iwconfig wlx0432f41f15b2 essid <name>  ex) iwconfig wlx0432f41f15b2 essid CRZ\_icanjji  udhcpc -i wlx0432f41f15b2=> ip 할당 받기 |

IP 할당 받으면 된다.

|  |
| --- |
| root@mango:~# ifconfig -a  wlx0432f41f15b2 Link encap:Ethernet HWaddr 04:32:F4:1F:12:81  inet addr:192.168.1.38 Bcast:192.168.100.255 Mask:255.255.255.0  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  RX packets:48 errors:0 dropped:6 overruns:0 frame:0  TX packets:20 errors:0 dropped:0 overruns:0 carrier:0  collisions:0 txqueuelen:1000  RX bytes:8859 (8.6 KiB) TX bytes:2556 (2.4 KiB) |

|  |
| --- |
| root@mango:~# ping 192.168.1.16  PING 192.168.1.16 (192.168.1.16) 56(84) bytes of data.  64 bytes from 192.168.1.16: icmp\_seq=1 ttl=64 time=62.4 ms  64 bytes from 192.168.1.16: icmp\_seq=2 ttl=64 time=7.60 ms  64 bytes from 192.168.1.16: icmp\_seq=3 ttl=64 time=3.24 ms  64 bytes from 192.168.1.16: icmp\_seq=4 ttl=64 time=2.89 ms  64 bytes from 192.168.1.16: icmp\_seq=5 ttl=64 time=5.65 ms |

# **GPU 테스트**

CM-AM6252 보드에서 PowerVR AXE-1-16 GPU의 성능을 아래 명령을 실행하여 측정할 수 있다.

|  |
| --- |
| root@mango:~# glmark2-es2-drm --off-screen  =======================================================  glmark2 Score: 46  =======================================================  root@mango:~# glmark2-es2-wayland --off-screen  =======================================================  glmark2 Score: 38  ======================================================= |

# **LCD 지원**

innolux-at070tn92 / innolux-m236hjj-l31 / bsd101wx1-300 세가지 LCD를 지원할 수 있다.

Uboot에서 아래와 같은 명령으로 LCD를 선택한다.

dtbo\_idx=0 /\* no overlay \*/

|  |
| --- |
| => setenv dtbo\_idx 0  => saveenv |

dtbo\_idx=1 /\* k3-am625-sk-innolux-at070tn92.dtbo \*/

|  |
| --- |
| => setenv dtbo\_idx 1  => saveenv |

dtbo\_idx=2 /\* k3-am625-sk-innolux-m236hjj-l31.dtbo \*/

|  |
| --- |
| => setenv dtbo\_idx 2  => saveenv |

dtbo\_idx=3 /\* k3-am625-sk-bsd101wx1-300.dtbo \*/

|  |
| --- |
| => setenv dtbo\_idx 3  => saveenv |

소스에서는 board-support\ti-u-boot\board\ti\am62x\am62x.env에서 원하는 값으로 수정한다.

|  |
| --- |
| dtbo\_idx=0  #if 0  dtbo\_idx=0 /\* no overlay \*/  dtbo\_idx=1 /\* k3-am625-sk-innolux-at070tn92.dtbo \*/  dtbo\_idx=2 /\* k3-am625-sk-innolux-m236hjj-l31.dtbo \*/  dtbo\_idx=3 /\* k3-am625-sk-bsd101wx1-300.dtbo \*/  #endif |

# **eMMC 이미지 Update**

eMMC boot mode에서 sd 카드나 usb mass storage를 이용하여 eMMC 이미지를 업데이트할 수 있다.

update\_image 폴더에서 스크립트를 실행하여 ramdisk를 생성한다.

|  |
| --- |
| user@ubuntu:~/am62x\_20231011/update\_image$ ./create\_ubuntu20.04.sh |

update\_image/images 폴더에 ramdisk.cpio.xz 파일이 생성된다. 이때 uboot image도 수정이 되어야 한다면 ramdisk 빌드 전에 uboot 이미지를 update\_image/custom\_datafs/boot-images 폴더에 복사한다.

Sd 카드나 usb mass storage에 /update\_image 폴더를 생성한 다음 ramdisk.cpio.xz와 함께 Image / k3-am625-sk.dtb / rootfs.tar.gz / custom\_datafs.tar.gz를 복사한다.

Sd 카드나 usb mass storage를 장착 후 부팅하여 eMMC boot mode에서 eMMC 이미지가 업데이트되는지 확인한다.