Migrating from the Moxa ART platform to the Moxa New Generation RISC Platform

Moxa Technical Support Team
support@moxa.com

Contents

Released on February 24, 2020

About Moxa

Moxa is a leading provider of edge connectivity, industrial computing, and network infrastructure solutions for enabling connectivity for the Industrial Internet of Things (IIoT). With over 30 years of industry experience, Moxa has connected more than 57 million devices worldwide and has a distribution and service network that reaches customers in more than 70 countries. Moxa delivers lasting business value by empowering industries with reliable networks and sincere service. Information about Moxa's solutions is available at www.moxa.com.

How to Contact Moxa

Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

1 Introduction

Moxa provides a wide portfolio of RISC-based Linux embedded computers for industrial applications, and has developed a new generation of computing platforms (using the Cortex A8 processor), the UC-2100/3100/5100/8100 series, which are designed to replace the UC-7100/IA-240/IA-260 series computers. Moxa will phase out the old products, listed below in Table 1, in 2020. Users can refer to Table 1 to choose the proper successor products. The specifications of most of the successor products include the specifications of the old products. For more details, please visit Moxa's official website.

Table 1. Successor Product Models

Old product Model name **(MOXA ART platform)**	**Successor** **(Cortex A8 platform)**
UC-7101-LX	UC2101-LX
UC-7112-LX/UC-7112-LX Plus	UC-2111-LX/UC-2112-LX
IA240-LX/ IA260	UC-5100 series

In the following paragraphs, we give users some pointers on how to migrate programs originally used on old computers to the newer computers.

Before migration, users must have a good understanding of the differences between the two software environments, device node definitions (i.e., peripherals), and API definitions.

Table 2. Differences Between Software Development Environments

	From		**To**
Platform	MOXA ART platform		Cortex A8 platform
OS	µClinux 2.6.19	Standard Linux 2.6.9	Debian ARM 8 or 9
Tool Chain	arm-elf-toolchain-1.6.sh (built for 32-bit environment)	arm-linux_1.3.sh (built for 32-bit environment)	arm-linux-gnueabihf_6.3_Build_amd64_20171114.sh (built for 64-bit environment)
Native Compiler	N/A	N/A	Yes
GCC version	2.95.3	3.3.2	4.9.2
Glibc version	0.9.26	2.2.5	2.19
$PATH	/usr/local/bin	/usr/local/arm-linux/bin	/usr/local/arm-linux-gnueabihf-6.3/usr/bin/
Console port baudrate	19200	115200	115200
Auto start program when boot up	/etc/rc	/etc/rc.local or /etc/init.d	systemd

	From	To
Reset Button	At least 5 seconds required to load factory defaults	Press and hold the reset button for 7 to 9 seconds to reset the computer to the factory default settings.

As above, the differences in the software environment are the tool-chain, versions of the kernel and Gcc, $PATH (environment variable), etc. The differences in the peripherals lie in the SD, RTC, Buzzer, and USB definitions. Refer to Table 3 for detailed relevant differences of the Device Node Definitions (Peripherals).

Table 3. Differences Between Device Node Definitions (Peripherals)

Model	UC7101 UC-7112-LX UC-7112-LX Plus	IA240-LX IA260-LX IA262-LX	Cortex A8 platform
Serial Port	/dev/ttyM0~1	/dev/ttyM0~4	/dev/ttyM0~1
USB	N/A	/mnt/usbstorage	/media/usb0
SD	/mnt/sd		/media/sd-mmcblk1p
RTC	/dev/rtc		/dev/rtc0
Buzzer	/dev/console		N/A(encapsulate in API)

The Cortex-A8 platform adopts new API calls. If you have used the MoxaART platform's APIs, refer to the following section, modify the original source code, and then recompile the code for use with the new platform.

UART Mode API

Operation	MoxaART platform w/ uC linux	MoxaART platform w/ standard linux	Cortex-A8 platform
Initialize Moxa UART control library	N/A	N/A	int mx_uart_init(void);
Set mode of target UART port	int ioctl(fd, MOXA_SET_OP_MODE, &mode)	int ioctl(fd, MOXA_SET_OP_MODE, &mode)	int mx_uart_set_mode(int port, int mode);
Get mode of target UART port	int ioctl(fd, MOXA_GET_OP_MODE, &mode)	int ioctl(fd, MOXA_GET_OP_MODE, &mode)	int mx_uart_get_mode(int port, int *mode);
			For more details, refer to moxa-arm-based-computer-linux-user-manual-for-debian-9-manual

Watchdog API

Operation	MoxaART platform w/ uC linux	MoxaART platform w/ standard linux	Cortex-A8 platform
Open watchdog device	int swtd_open(void)	int swtd_open(void)	int open(“/dev/watchdog”,O_RDWR)
Keep alive	int swtd_ack(int fd)	int swtd_ack(int fd)	int ioctl(fd, WDIOC_KEEPALIVE, 0)
Set timeout	int swtd_enable(int fd, unsigned long time)	int swtd_enable(int fd, unsigned long time)	ioctl(fd, WDIOC_SETTIMEOUT, &timeout)
Get timeout	int swtd_get(int fd, int *mode, unsigned long *time)	int swtd_get(int fd, int *mode, unsigned long *time)	ioctl(fd, WDIOC_GETTIMEOUT, &timeout)

Disable ack	int swtd_disable(int fd)	int swtd_disable(int fd)	int options = WDIOS_DISABLECARD;
			ioctl(fd, WDIOC_SETOPTIONS, &options);

Close watchdog file handle	int swtd_close(int fd)	int swtd_close(int fd)	
			For more details, refer to moxa-arm-based-computer-linux-user-manual-for-debian-9-manual

Buzzer API

Operation	MoxaART platform w/ uC linux	MoxaART platform w/ standard linux	Cortex-A8 platform
Init buzzer control	N/A	N/A	int mx_buzzer_init(void)
Play buzzer	ioctl(fd, KDMKTONE, unsigned int arg);	ioctl(fd, KDMKTONE, unsigned int arg);	int mx_buzzer_play_sound(unsigned long duration)
Stop buzzer	N/A	N/A	int mx_buzzer_stop_sound(void)
			For more details, refer to moxa-arm-based-computer-linux-user-manual-for-debian-9-manual

DIO API

platform	MoxaART platform w/ uC linux	MoxaART platform w/ standard linux	Cortex-A8 platform
Init DIO control	N/A	N/A	int mx_dio_init(void)
Set DO state	N/A	int set_dout_state(int doport, int state)	int mx_dout_set_state(int doport, int state)
Get DO state	N/A	int get_dout_state(int doport, int *state)	int mx_dout_get_state(int doport, int *state)
Get DI state	N/A	int get_din_state(int diport, int *state)	int mx_din_get_state(int diport, int *state)
Set DI event	N/A	int set_din_event(int diport, void (*func)(int diport), int mode, unsigned long duration)	int mx_din_set_event(int diport, void (*func)(int diport), int mode, unsigned long duration)
Get DI event	N/A	int get_din_event(int diport, int *mode, unsigned long *duration)	int mx_din_get_event(int diport, int *mode, unsigned long *duration)
			For more details, refer to moxa-arm-based-computer-linux-user-manual-for-debian-9-manual

Other operations

platform	MoxaART platform w/ uC linux	MoxaART platform w/ standard linux	Cortex-A8 platform
Default user	root	root	moxa
upramdisk/ downramdisk	Yes	Yes	No.
System commands	busybox	busybox	Standard Debian commands
init system	SysV	SysV	systemd
Shell	sh	sh	dash/bash
network interface configuration	/etc/motd	/etc/network/interfaces	/etc/network/interfaces

2 Steps for Migrating to the Cortex-A8 Platform

In this section, we give the steps you should follow to migrate a program from the MoxaART platform to the Cortex-A8 platform.

Step1: Set up the Cortex-A8 platform's development environment, including installing the Cortex-A8 platform's native compiler or tool chain. For more details, refer to moxa-arm-based-computer-linux-user-manual-for-debian-9-manual.

Step2: Develop or move the old code to the new development environment.

Step3: Check the information provided in the above tables to modify the code and then compile it.

Step4: If you receive any error messages during compilation, return to Step 3.

Step5: Download the program to the Cortex-A8 platform via SSH.

Step6: If needed, set up the systemd script for your program.

Step7: Test the program.
→If bugs are found, return to Step 3.
→If no bugs are found, continue with Step 8.

Step 8: Distribute the program to additional Cortex-A8 platform units if needed.

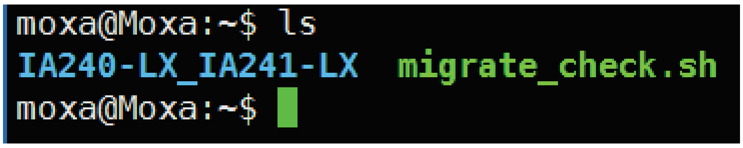
3 Using the Migration Checking Tool

To help users migrate old programs to new platforms more efficiently, Moxa provides a migration tool for listing the file names and line numbers of source code using old definitions. Users can refer to the resulting information to modify only those portions of the source code that require modification.

You may download the tool from Moxa's product website.

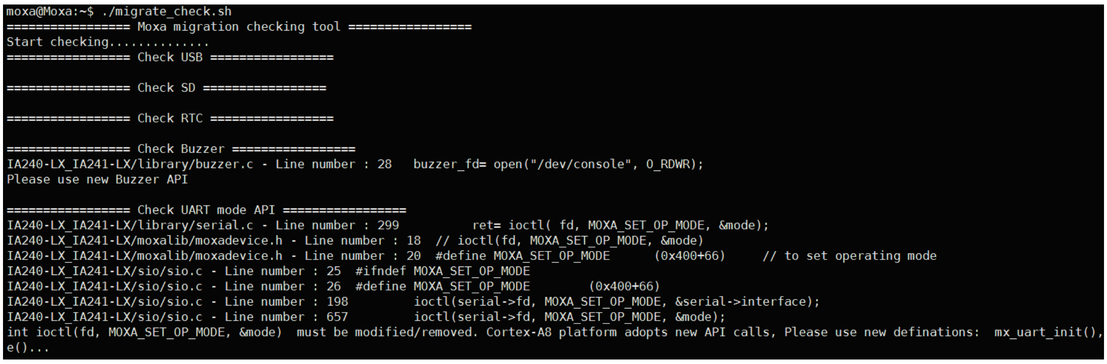
Usage:

1. Upload your MoxaART platform's source code to the new Moxa Cortex-A8 platform.
2. Put the migrate_check.sh in the same directory as your source code.



```
moxa@Moxa:~$ ls
IA240-LX_IA241-LX  migrate_check.sh
moxa@Moxa:~$
```

3. Execute migrate_check.sh. If the tool finds old definitions in your source code, it will show file names and line numbers of that source code. Please modify the code using the new definitions. For new definitions, refer to moxa-arm-based-computer-linux-user-manual-for-debian-9-manual.



```
moxa@Moxa:~$ ./migrate_check.sh
================= Moxa migration checking tool =================
Start checking.............
================= Check USB =================

================= Check SD =================

================= Check RTC =================

================= Check Buzzer =================
IA240-LX_IA241-LX/library/buzzer.c - Line number : 28   buzzer_fd= open("/dev/console", O_RDWR);
Please use new Buzzer API

================= Check UART mode API =================
IA240-LX_IA241-LX/library/serial.c - Line number : 299          ret= ioctl( fd, MOXA_SET_OP_MODE, &mode);
IA240-LX_IA241-LX/moxalib/moxadevice.h - Line number : 18  // ioctl(fd, MOXA_SET_OP_MODE, &mode)
IA240-LX_IA241-LX/moxalib/moxadevice.h - Line number : 20  #define MOXA_SET_OP_MODE      (0x400+66)     // to set operating mode
IA240-LX_IA241-LX/sio/sio.c - Line number : 25  #ifndef MOXA_SET_OP_MODE
IA240-LX_IA241-LX/sio/sio.c - Line number : 26  #define MOXA_SET_OP_MODE        (0x400+66)
IA240-LX_IA241-LX/sio/sio.c - Line number : 198         ioctl(serial->fd, MOXA_SET_OP_MODE, &serial->interface);
IA240-LX_IA241-LX/sio/sio.c - Line number : 657         ioctl(serial->fd, MOXA_SET_OP_MODE, &mode);
int ioctl(fd, MOXA_SET_OP_MODE, &mode)  must be modified/removed. Cortex-A8 platform adopts new API calls, Please use new definations:  mx_uart_init(),
e()...
```

4. Note that you may need to modify the logic of the program after using the new API.

4 Reporting Migration Problems

Before reporting migration problems to the Moxa Technical support team, please prepare the following items:

1. Old product model name, firmware version (kversion)
2. New product model name, firmware version (kversion)
3. Error message.
4. How to reproduce the error.

5 Revision History

Version	Release Notes	Release Date
v1.0	---	2020-03-03