NuMicro® Family

Arm® Cortex®-A35-based Microprocessor

NuMicro® Family
MA35 Series SWriter
User Manual

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 OVERVIEW

The MA35 series microprocessor supports booting from USB, SPI NOR/NAND, NAND, and SD/eMMC. No matter which boot mode is selected, the boot images should be written to boot storages combined with MA35 proprietary image headers, which can tell MA35 IBR (Internal Boot ROM) how to load and run the boot images. Nuvoton provides a NuWriter tool to help users pack boot images into IBR acceptable format. The NuWriter tool is written in Python and can run under Windows or Linux PC to connect with MA35 target boards via USB connection. The NuWriter can download and program packed boot images to the boot storages on the MA35 target board, for example, NAND Flash. For mass production, the NuWriter can connect up to 8 target boards from one Windows PC.

This document introduces SWriter, an on-board programming tool for the MA35 series. The SWriter provides a standalone proposal for programming boot storages. By making a bootable SD card or USB Pen Drive, it can program the NAND Flash, SPI NAND Flash, and eMMC. As compared to NuWriter, the SWriter does not require a Windows PC. The SWriter itself is an ARM64 code and can run on the MA35 target board, which intends to provide a simple, low-cost, and convenient method for mass production.

The bootable SD card / USB Pen Drive includes writer firmware in physical sectors and BSP binary files in FAT system. User can update BSP binary files to program different storage.

2 MAKE A SPECIFIC SD CARD

MA35 SWriter boots from an SD card with the objective of programming other storage devices, including NAND, SPI-NAND, and eMMC. Nuvoton has released the writer binary to facilitate the creation of a bootable SD card. There are two methods for creating a bootable SD card: using NuWriter or employing the Linux dd command.

2.1 NuWriter

In make-sdcard directory, there are five files – header.json, pack.json, sd_writer.bin, sd_writer_MBR.bin and otp_writer.bin. Copy these files to NuWriter directory. Use the following command to generate a pack binary file. The DDR image should depend on the target board.

- sd_writer.bin – Standalone writer firmware for SD boot.
- sd_writer_MBR.bin – Master Boot Record for FAT use.
- otp_writer.bin – OTP writer firmware.

Released setting is for NuMaker-HMI-MA35D1, NuMaker-HMI-MA35H04F70, and NuMaker-IOT-MA35D03F80 boards.

```
$ nuwriter.py –c header.json
$ nuwriter.py –p pack.json
```

After generating the pack.bin, use NuWriter to program the SD card. Follow the steps below:

1. Set the power-on-setting to USBD boot. Then power on the system.

2. Connect USB cable to PC.

3. Make NuWriter to “attach” the device.

```
$ nuwriter.py –a ddrimg/enc_ddr3_winbond_256mb.bin
```

4. Use NuWriter to write the pack.bin to SD card.

```
$ nuwriter.py –w sd pack.bin
```

Once the programming is completed, a bootable SD card is ready. The SD card still needs to be formatted as FAT format. Insert the SD card to PC, format the SD card as FAT file system, and copy the related files to SC card.

2.2 Linux dd Command

In make-sdcard directory, there are five files – header.json, pack.json, sd_writer.bin, sd_writer_MBR.bin and otp_writer.bin. Copy these files to NuWriter directory. Use the following command to generate a pack binary file. The DDR image should depend on the target board. Released setting is for NuMaker-HMI-MA35D1, NuMaker-HMI-MA35H04F70, and NuMaker-IOT-MA35D03F80 boards.

```
$ nuwriter.py –c header.json
$ nuwriter.py –o stuff –p pack.json
```

After generating the pack.bin, use Linux dd command to program the SD card.

```
$ dd if=pack.bin of=/dev/sdb
```

A bootable SD card is ready now. Insert the SD card to PC to format the SD card as FAT file system and copy the relative files to SD card. For more details, refer to Chapter 5 DISK Content.

2.3 How to Use SWriter

For example, use the NuMaker-HMI-MA35D1 board SD0 to program NAND Flash.

1. Insert the SD card specific to SWriter.
2. Set power on setting to be boot from SD0. PG0 on, PG2 on, and PG3 off.
3. Power on the system.
4. SWriter will follow the ‘config’ file to program/verify the NAND Flash. Please refer to Chapter 5 DISK Content for more details.

3 MAKE A SPECIFIC USB PEN DRIVE

USBH SWriter is used by MA35 USBH boot and program other storages, including NAND, SPI-NAND, and eMMC. Nuvoton released the writer binary to make a bootable USB Pen Drive.

3.1 Linux dd Command

In make-usbh-msc directory, there are five files – header.json, pack.json, usbh_writer.bin, writer_MBR.bin and otp_writer.bin. Copy these files to NuWriter directory. Use the following command to generate a pack binary file. The DDR image should depend on the target board.

- usbh_writer.bin – Standalone writer firmware for USBH boot.
- writer_MBR.bin – Master Boot Record for FAT use.
- otp_writer.bin – OTP writer firmware.

Released setting is for NuMaker-HMI-MA35D1, NuMaker-HMI-MA35H04F70, and NuMaker-IOT-MA35D03F80 boards.

```
$ nuwriter.py –c header.json
$ nuwriter.py –o stuff –p pack.json
```

After generating the pack.bin, use Linux dd command to program the USB Pen Drive.

```
$ dd if=pack.bin of=/dev/sdb
```

A bootable USB Pen Drive is ready now. Plug the USB pen drive to PC, format the pen drive as FAT file system, and copy the relative files to the pen drive. For more details, refer to Chapter 5 DISK Content.

3.2 How to Use SWriter

For example, use NuMaker-HMI-MA35D1 board USBH1 to program NAND Flash.

1. Insert the USB Pen Drive specific to SWriter.
2. Set power on setting to be boot from USBH1. Turn the power-on setting pins PG 0, 2, 3, 4, 5 on and PG.6 off.
3. Power on the system.
4. SWriter will follow the ‘config’ file to program/verify the NAND Flash. Please refer to Chapter 5 DISK Content for more details.

3.3 Board Limitations

For MA35 USBH boot, there are some limitations on USB circuit design. Figure 3-1 is a schematic of USB power.

The relative relationship between the boot process and power control is listed below:

- Insert the bootable USB Pen Drive.
- Power on: RTC power and V_{BUS} 5V.
- Press RTC_nRWAKE
- MA35 Boot
- Start USBH writer

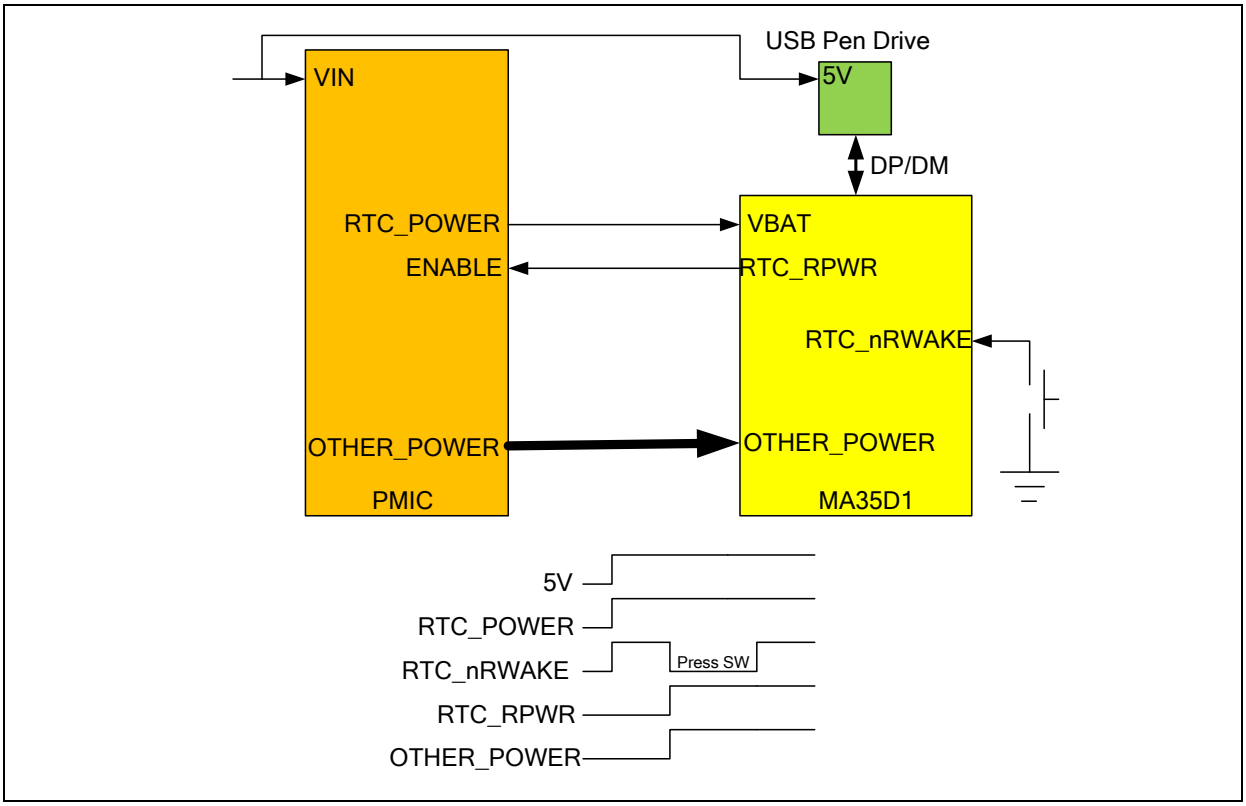

Figure 3-1 USB Power Reference Circuit

4 BUILD AN OTP FILE

MA35 NuWriter_MA35_UI has a friendly user interface. User can select self OTP setting by NuWriter and export it. Please convert otp.json before putting it into disk.

NuWriter_MA35: Use the following command to convert the otp.json to ma35_otp.bin.

```
$ nuwriter.py –o otp –c otp.json
```

NuWriter_MA35_UI:

- Step 1: Select “**Develop mode**”.
- Step 2: Click “**Convert**”.
- Step 3: Click “**Convert otp.json**”.
- Step 4: Browse otp.json file.
- Step 5: Click “**Convert**”.

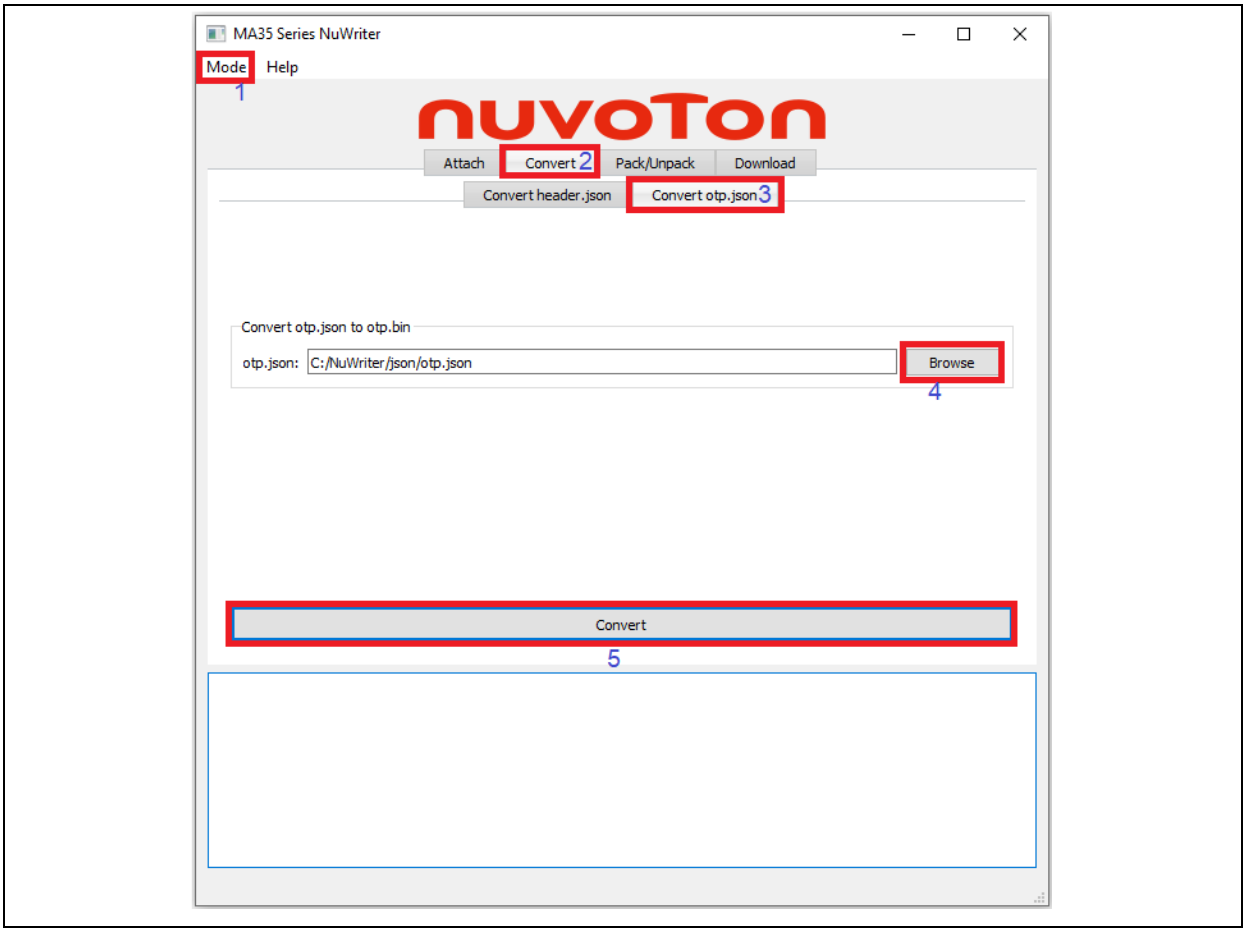

Figure 4-1 NuWriter_MA35_UI Convert OTP

Note: OTP and KEY are one-time-program only. **Unrecoverable damage can occur if the wrong OTP is flashed to the target board!** Please take care the OTP content.

5 DISK CONTENT

The files should depend on programming storage. User should modify the 'Config' file to match the storage.

5.1 NAND Flash Configuration

The following shows NAND Flash setting.

```
[TARGET]
//1. NUC972 2. NUC980 3. MA35 Series
3
[TYPE]
//NUC972, NUC980: 1. nand-normal 2. nand-pack
//MA35 Series: 1. normal
1
[FLASH]
//MA35 Series: 1. NAND 2. SPI-NAND 3. eMMC
1
[OTP]
//Format: file name
ma35_otp.bin
[Data0]
//Format: file name, offset
header.bin, 0x0
[Data1]
//Format: file name, offset
bl2.dtb, 0xC0000
[Data2]
//Format: file name, offset
bl2.bin, 0xE0000
[Data3]
//Format: file name, offset
fip.bin-nand, 0x100000
[Data4]
//Format: file name, offset
uboot-env.bin-nand, 0x300000
[Data5]
//Format: file name, offset
Image.dtb, 0x3C0000
[Data6]
//Format: file name, offset
Image, 0x400000
```

```
[Data7]
//Format: file name, offset
rootfs.ubi, 0x1C00000
[UserDefine]
EraseAll=1, Verify=1
[NAND_FLASH]
//ECC: 0. T24, 1. T4, 2. T8, 3. T12, 4. T15, 5, IGNORE
//PageSize=2048, PagePerBlock=64, BlockPerFlash=8192, SpareSize=64, ECC=2
[SD]
//Port=1
[LED]
//Port=9, Pin=14
```

Copy the following NAND needed files to SD card or USB Pen Drive to program the NAND Flash.

- Config
- bl2.bin
- bl2.dtb
- fip.bin-nand
- header.bin
- Image
- Image.dtb
- rootfs.ubi
- uboot-env.bin-nand
- ma35_otp.bin (optional)

5.2 SPI-NAND Flash Configuration

The following shows SPI-NAND Flash setting.

```
[TARGET]
//1. NUC972 2. NUC980 3. MA35 Series
3
[TYPE]
//NUC972, NUC980: 1. nand-normal 2. nand-pack
//MA35 Series: 1. normal
1
[FLASH]
//MA35 Series: 1. NAND 2. SPI-NAND 3. eMMC
2
[OTP]
```

```
//Format: file name
ma35_otp.bin
[Data0]
//Format: file name, offset
header.bin, 0x0
[Data1]
//Format: file name, offset
bl2.dtb, 0xC0000
[Data2]
//Format: file name, offset
bl2.bin, 0xE0000
[Data3]
//Format: file name, offset
fip.bin-spinand, 0x100000
[Data4]
//Format: file name, offset
uboot-env.ubi-spinand, 0x300000
[Data5]
//Format: file name, offset
Image.dtb, 0x3C0000
[Data6]
//Format: file name, offset
Image, 0x400000
[Data7]
//Format: file name, offset
rootfs.ubi, 0x1C00000
[UserDefine]
EraseAll=1, Verify=1
[NAND_FLASH]
//ECC: 0. T24, 1. T4, 2. T8, 3. T12, 4. T15, 5, IGNORE
//PageSize=2048, PagePerBlock=64, BlockPerFlash=8192, SpareSize=64, ECC=2
[SD]
//Port=1
[LED]
//Port=9, Pin=14
```

Copy the following SPI-NAND needed files to SD card or USB Pen Drive to program the SPI-NAND Flash.

- Config
- bl2.bin

- bl2.dtb
- fip.bin-spinand
- header.bin
- Image
- Image.dtb
- rootfs.ubi
- uboot-env.ubi-spinand
- ma35_otp.bin (optional)

5.3 eMMC Configuration

The following shows eMMC setting.

```
[TARGET]
//1. NUC972 2. NUC980 3. MA35 Series
3
[TYPE]
//NUC972, NUC980: 1. nand-normal 2. nand-pack
//MA35 Series: 1. normal
1
[FLASH]
//MA35 Series: 1. NAND 2. SPI-NAND 3. eMMC
3
[OTP]
//Format: file name
ma35_otp.bin
[Data0]
//Format: file name, offset
MBR.sdcard.bin, 0x0
[Data1]
//Format: file name, offset
header.bin, 0x400
[Data2]
//Format: file name, offset
bl2.dtb, 0x20000
[Data3]
//Format: file name, offset
bl2.bin, 0x30000
[Data4]
//Format: file name, offset
```

```
uboot-env.bin-sdcard, 0x40000
[Data5]
//Format: file name, offset
fip.bin-sdcard, 0xC0000
[Data6]
//Format: file name, offset
Image.dtb, 0x2C0000
[Data7]
//Format: file name, offset
Image, 0x300000
[Data8]
//Format: file name, offset
rootfs.ext4, 0x2400000
[UserDefine]
EraseAll=0, Verify=1
[NAND_FLASH]
//ECC: 0. T24, 1. T4, 2. T8, 3. T12, 4. T15, 5, IGNORE
//PageSize=2048, PagePerBlock=64, BlockPerFlash=8192, SpareSize=64, ECC=2
[SD]
Port=1
[LED]
//Port=9, Pin=14
```

Copy the following eMMC needed files to SD card or USB Pen Drive to program the eMMC.

- Config
- bl2.bin
- bl2.dtb
- fip.bin-sdcard
- header.bin
- Image
- Image.dtb
- rootfs.ext4
- uboot-env.bin-sdcard
- MBR.sdcard.bin
- ma35_otp.bin (optional)

6

6 REVISION HISTORY

Date	Revision	Description
2024.05.02	1.03	Add SD port setting. Add LED setting.
2023.12.12	1.02	Change MA35D1 to MA35 Series. Add support for MA35D0 and MA35H0 platform.
2023.04.28	1.01	Add UI support for ‘build an OTP file’ function.
2023.02.23	1.00	Initial version.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.