



Joy of innovation
nuvoTon

NuMaker - MA35D1 Quick Start

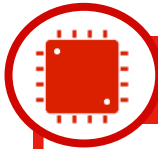
| Agenda

- Target Platform Introduction
- Overview
- Environment Setup
- Build Image by Yocto
- Image Programming and System Boot
- Fast Application Development

Target Platform Introduction



MA35D1 Application Processor



System

- Dual 64/32-bit Arm® Cortex®-A35 core running up to 1 GHz with 32/32 KB L1 I/D cache each one, 512 KB shared L2 cache, Arm® NEON™ and Arm® Trust Zone®
- A 32-bit Arm® Cortex®-M4 core running up to 200 MHz with 16/16 KB I/D cache, FPU/MPU
- Built-in 128/256 MB DDR2/DDR3L SDRAM in LQFP/BGA package



Features

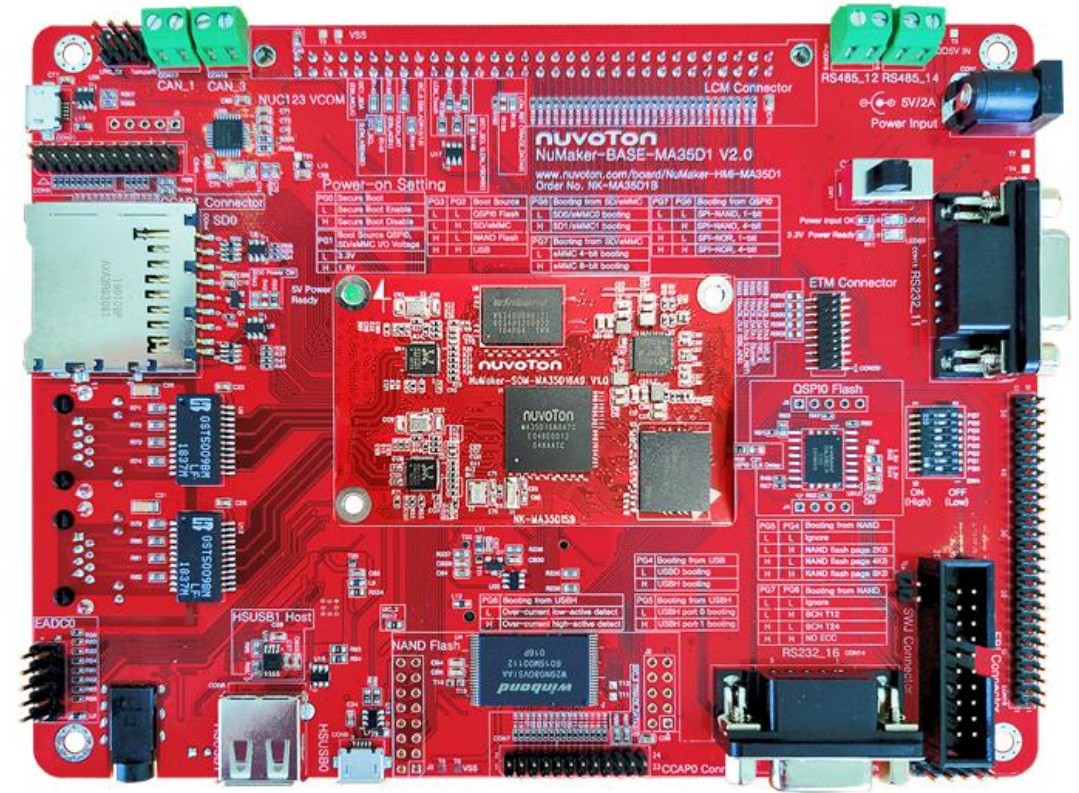
- Supports LCD Display controller, 2D Graphic Engine, H.264 decoder for HMI display
- Two CCIR656/601 Camera interfaces
- Two 10/100/1000 Mbps Ethernet MACs
- USB 2.0 high-speed host/device
- Supports AES-256, SHA-512, ECC-571, RSA-4096
- CAN-FD interfaces
- UART, ISO-7816 interfaces, Quad-SPI, SPI, I2C, I2S
- EPWM, 12-bit SAR ADC, 32-bit timers
- RTC (Real Time Clock), 32.768 kHz Oscillator
- WDT (Watchdog Timer), WWDT(Window)
- True Random Number Generator (TRNG)

NuMaker-HMI-MA35D1 Evaluation Board Features



Features

- MA35D16A81 (BGA312) MCP package with DDR3L (256 MB)
- 7-inch TFT LCD (1024x600) with touch daughter board
- An on-board eMMC (SD3.0) Flash memory device (16 GB)
- An on-board Quad SPI NAND Flash device (512 MB)
- An on-board NAND Flash device (1 GB)
- Standard-SD (SD2.0) memory card slot
- 2 x Giga Ethernet
- 2 x High Speed USB
- 2 x Camera Capture (CMOS sensor) header connectors
- 1 x Audio codec (NAU88C22)
- 2 x UART, 2 x RS485, 2 x CAN-FD, 8 x EADC channels

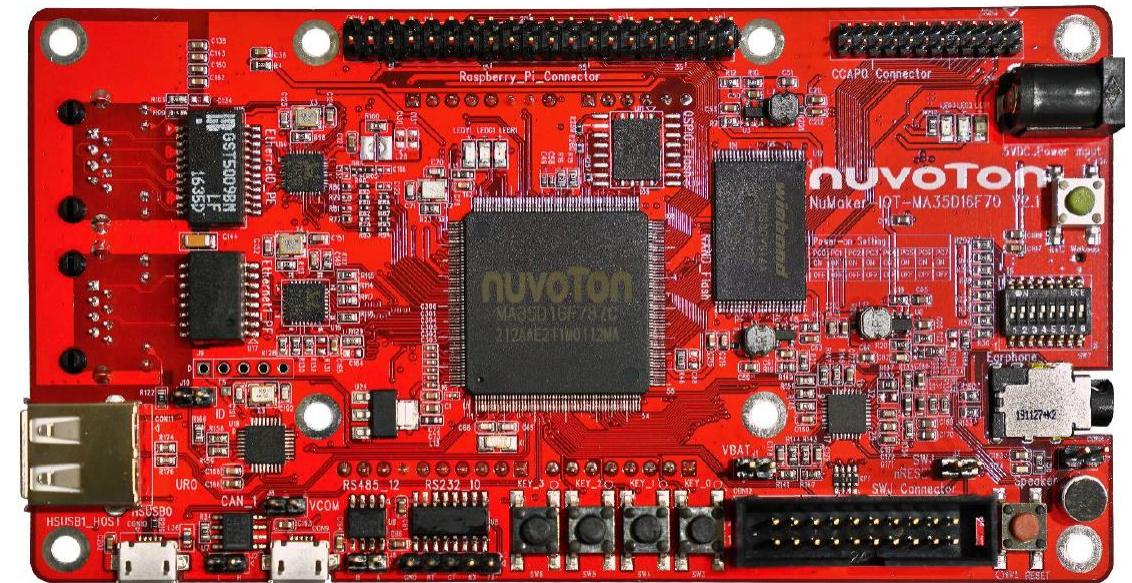


NuMaker-IoT-MA35D1 Evaluation Board Features



Features

- MA35D16F987C (LQFP216) MCP package with DDR (512MB)
- Run up to 800MHz operating speed
- An on-board Quad SPI NAND Flash device (512 MB)
- An on-board NAND Flash device (1 GB)
- Standard-SD memory card (SD1, supports SD2.0)
- 1 x Giga Ethernet, 1 x 10/100 Ethernet
- 2 x High Speed USB
- 1 x Camera Capture (CMOS sensor) header connectors
- 1 x Audio codec (NAU88C22)
- 1 x SIM Card slot
- 2 x UART, 2 x RS485, 2 x CAN-FD, 8 x EADC channels



| Quick Start Material

- Level 1 Quick Start
 - [MA35D1 Application Processor](#)
 - [Evaluation Board and Demo](#)
 - [NuWriter_MA35 Programming](#)
- Level 2 Development Environment
 - [Development Environment set up](#)
 - [GUI development environment set up](#)
 - [Hardware Design Notice](#)
- Level 3 Key Features
 - [TF-A](#)
 - [OP-TEE](#)
 - [Real Time Processor](#)
 - [VC8000 Video Decoder](#)

| Document

- Nuvoton provides many documents to help users develop quickly
- User Manual
 - UM_NuMaker_HMI_MA35D1_S1
 - UM_NuMaker_IoT_MA35D1_A1
- Technical Reference Manual
- Schematic, PCB and Gerber file
 - NuMaker-IoT-MA35D1-A1 Schematic, PCB , Gerber file & BOM
- Datasheet
 - MA35D1 Series Datasheet

| Document

- Software User Manual
 - UM_EN_MA35D1_BSP
 - UM_EN_MA35D1_Buildroot
 - UM_EN_MA35D1_Linux_BSP
 - UM_EN_MA35D1_NAND
 - UM_EN_MA35D1_NuWriter
 - UM_EN_MA35D1_OP-TEE
 - UM_EN_MA35D1_RTP
 - UM_EN_MA35D1_TF-A
 - UM_EN_MA35D1_U-boot
 - UM_EN_MA35D1_Yocto

| Document

- Application Note
 - MA35D1_NAND Chip Support_EN
 - MA35D1_Coprocessor_Management_EN
 - MA35D1_Hardware_Design_Guide_EN
 - MA35D1_PMIC_DA9062-3A_EN
 - MA35D1_Secure_Boot_EN
 - MA35D1_TSI_EN
- FAQ
 - FAQ_MA35D1_Extend Boot Space
 - FAQ_MA35D1_NuWriter Issue
 - FAQ_MA35D1_Read MAC Address from OTP
 - FAQ_MA35D1_Show Logo when booting by Yocto
 - FAQ_MA35D1_Switch booting source

Overview

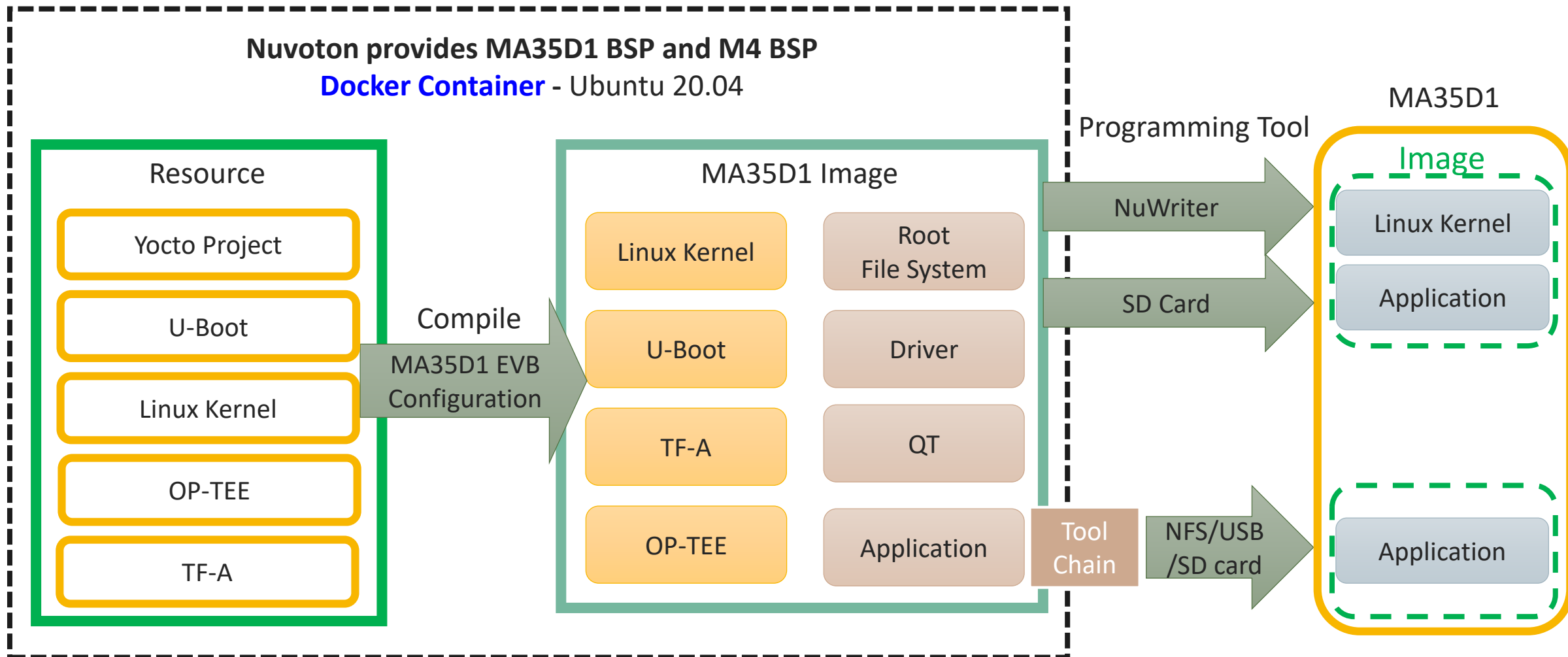


MA35D1 Development Environment

- Development environment
 - Docker container – Ubuntu 20.04 : virtual machine in Linux OS
- Development Tools
 - Yocto Project – Build a Linux distribution (MA35D1 Kernel) or an application
 - Buildroot – Build a Linux distribution (MA35D1 Kernel) or an application
 - OpenWRT – Build a Linux distribution (MA35D1 Kernel) or an application
 - NuWriter – Nuvoton provides a tool to program image to storage media
 - NuTool-PinConfigure – Nuvoton provides a tool to define pin Configure
- Image
 - Kernel – Linux 5.4.y
 - U-Boot v2020.07
 - OPTEE v3.9.0
 - ARM Trusted Firmware v2.3

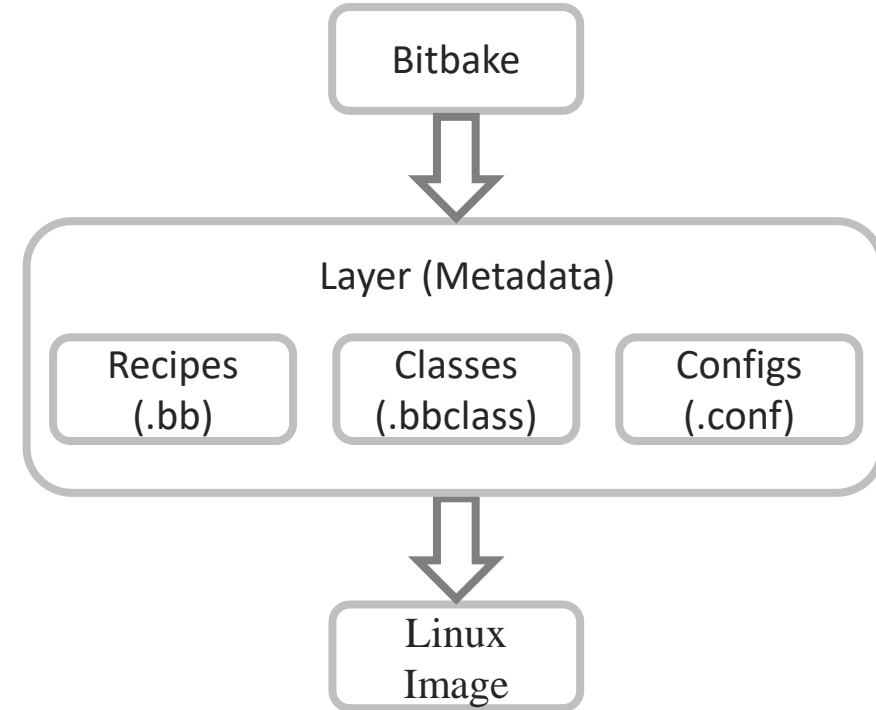
MA35D1 Development Scheme

Development Environment



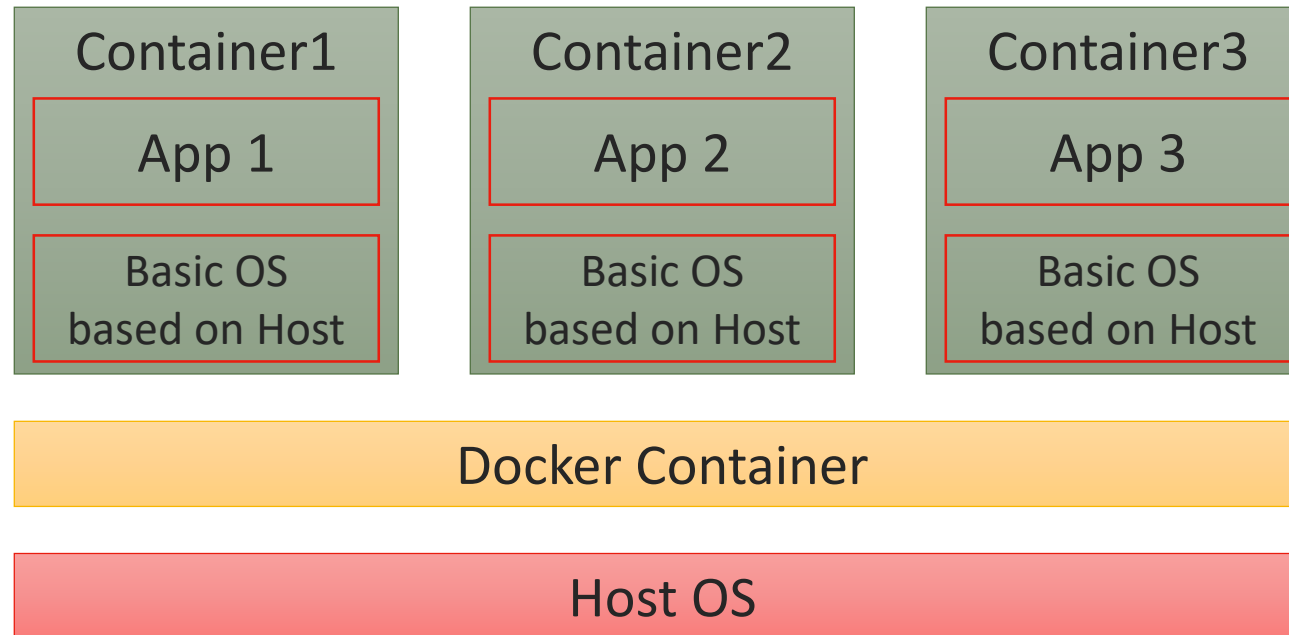
MA35D1 Linux Development Tools - Yocto

- The Yocto Project is an open source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture
- MA35D1 Yocto include the following Metadata
 - Meta-ma35d1
 - Meta-qt5
 - Meta-virtualization
 - Meta-Pocky



Yocto Development Environment - Docker

- Docker can pack up code and its dependencies as a container
- Every container is independent and based on Host OS so they won't affect each other and run faster than virtual machine



MA35D1 Linux Board Support Package

- This BSP supports Linux operating system for MA35D1. The peripheral drivers are also included in the BSP allowing applications to access them

Component	Description
Yocto	Version 3.1.3 (Dunfell). A Linux Foundation collaborative open source project to create the Linux distributions
Buildroot	Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation
OpenWrt	The OpenWrt Project is a Linux operating system targeting embedded devices
Linux	Version 5.4. An open source operating system based on GPLv2 license
U-Boot	Version 2020.07. An open source bootloader based on GPLv2+ license
OP-TEE	Version 3.9.0. An open source trusted execution environment
TF-A	Version 2.3. A BSD-3-Clause license reference implementation of secure world software
M4 BSP	CMSIS library 4.5.0 and standard driver for RTP BareMetal/FreeRTOS firmware development
NuWriter	A GUI and command line tool supports firmware update and OTP programming for MA35D1

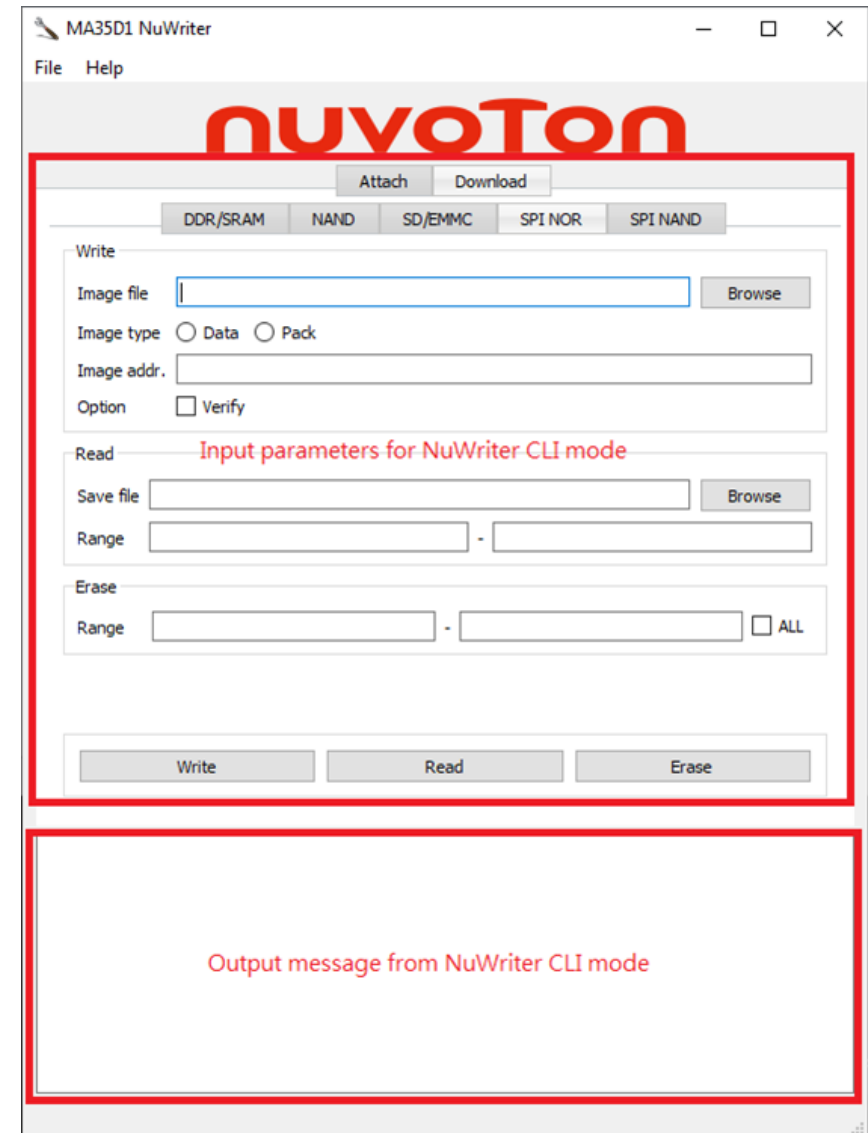
| MA35D1 Programming Tools

- Nuvoton provides several programming tools to help users program their own Image to devices

Name	Description
NuWriter_MA35	PC use USB interface program Image to <ol style="list-style-type: none">1. SD/eMMC2. SPI-NAND/NOR3. NAND
AutoWriter	MA35 use SD card program Image to <ol style="list-style-type: none">1. SD/eMMC2. SPI-NAND3. NAND
NANDWriter	NUC980 use SD card program Image to MA35D1 <ol style="list-style-type: none">1. NAND

MA35D1 Programming Tool - NuWriter

- Download NuWriter for MA35D1 :
 - https://github.com/OpenNuvoton/MA35D1_NuWriter
- The NuWriter is a programming tool for the MA35D1. The NuWriter application and firmware code are open sourced, and users can add new features or develop new user interfaces per user's application.
- USB host (mass storage) is also supported in mass production version
- The MA35D1 supports the following four system boot-up conditions:
 1. Boot from USB
 2. Boot from SD/eMMC
 3. Boot from NAND
 4. Boot from SPI Flash



Environment Setup



| Environment Setup

- PC Spec:
- CPU i5-10400 、 16GB DDR 、 1TB SSD
- The Yocto project needs at least 150 GB storage media but we recommend 200 GB up
- Nuvoton provides VMware Image which have been installed related packages to build MA35D1 Image
- Download [VMware Image](#)

After downloading the virtual machine, you can skip these following steps to [Start up with VMware](#)

This VMware Image also can be use to build Image by Buildroot

| Environment Setup (1/5)

- The necessary packages must be installed before building
- Ubuntu and Debian

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \  
build-essential chrpath socat cpio python python3 python3-pip python3-pexpect \  
xz-utils debianutils iputils-ping libsdl1.2-dev xterm curl
```

| Environment Setup (2/5)

1. This demo is under Ubuntu distribution. If you use virtual machine, ensure your RAM at least 5GB
2. Update existing list of packages

```
$ sudo apt-get update
```

3. Install a few prerequisite packages which let apt use packages over HTTPS

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

4. Add Docker's official GPG key for the official Docker repository to your system

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

5. Set up the stable repository, add the Docker repository to APT sources

```
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

| Environment Setup (3/5)

6. Update the package database with the Docker packages from the newly added repo

```
$ sudo apt-get update
```

7. Install Docker

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

8. Download the Docker Script for MA35D1

```
$ git clone https://github.com/OpenNuvoton/MA35D1_Docker_Script.git
```

```
nuvoton@user:~/docker-yocto$ ls  
build.sh  Dockerfile  join.sh  README.md
```

| Environment Setup (4/5)

9. Enter docker-yocto folder, build docker image. It may take one hour to get about 710 files

```
$ ./build.sh
```

10. Enter docker image, and your command line head will be like `nuvoton@a24d9e06abe3:~$`

```
$ ./join.sh  
ma35d1_user  
nuvoton@a24d9e06abe3:~$
```

11. Create a folder name `yocto` under `/share`

```
nuvoton@a24d9e06abe3:~/share$ mkdir yocto
```

| Environment Setup (5/5)

12. The first time you use repo, you need to set up the GIT environment

```
nuvoton@a24d9e06abe3:~/shared/yocto$ git config --global user.email "test@test.test.test"  
nuvoton@a24d9e06abe3:~/shared/yocto$ git config --global user.name "test"  
nuvoton@a24d9e06abe3:~/shared/yocto$ git config --global http.sslverify false
```

13. Go to /share/yocto to setup repo path

```
nuvoton@a24d9e06abe3:~/share/yocto$ repo init -u  
https://github.com/OpenNuvoton/MA35D1_Yocto-v3.1.3.git -m meta-ma35d1/base/ma35d1.xml
```

14. Download the yocto project and update ma35d1 source code

```
nuvoton@a24d9e06abe3:~/share/yocto$ repo sync
```

| Start up with VMware

- This VMware Image provides a MA35D1 Linux development environment

User Name: user

Password: user

- Yocto:

```
$ cd ~/yocto  
~/yocto$ repo sync
```

- Buildroot:

```
$ cd ~/buildroot/MA35D1_Buildroot  
~/MA35D1_Buildroot$ git pull
```


Build Image by Yocto



Build Image by Yocto (1/3)

1. Setup building configuration. The DISTRO option we usually use nvt-ma35d1-directfb

```
~/yocto$ DISTRO=nvt-ma35d1-directfb MACHINE=numaker-som-ma35d16a81 source sources/init-build-env build
```

After typing this command and if you want to change this setting, [please modify /build/conf/local.conf](#)

If you exit the docker container and join the docker container again, source the environment variables

```
~/yocto$ source sources/init-build-env build/
```

- Usage:
 - MACHINE=<machine> DISTRO=<distro> source sources/init-build-env <build-dir>
<machine> machine name <distro> distro name <build-dir> build directory
- Choose which DISTRO configuration you want to build
 - nvt-ma35d1-directfb (sources/meta-nua3500/conf/distro/nvt-ma35d1-directfb.conf)
- Choose which machine configuration you want to build
 - numaker-som-ma35d16a81 (sources/meta-ma35d1/conf/machine/ numaker-som-ma35d16a81)
 - numaker-iot-ma35d16f70 (sources/meta-ma35d1/conf/machine/ numaker-iot-ma35d16f70)
 - numaker-iot-ma35d16f90 (sources/meta-ma35d1/conf/machine/ numaker-iot-ma35d16f90)

| Build Image by Yocto (2/3)

1. Choose what Image you want to build

Image name	Target	Layer
core-image-minimal	A small image that only allows a device to boot.	Poky
nvt-image-qt5	Builds ma35d1 image	meta-nua3500

2. Here we choose nvt-image-qt5 to build image
3. This step take about 3hrs first time (download and compile)

```
$ bitbake nvt-image-qt5
```

4. After compiling completed, you can see image at

```
~/yocto/build/tmp-glibc/deploy/images/ma35d1-som-ma35d16a81/
```

| Build Image by Yocto (3/3)

- Update Yocto Project

```
nuvoton@a24d9e06abe3:~/share/yocto$ repo sync
```

- Update Linux
- Notice that, the command below will delete all Linux source code and download the newest source code and compile.

```
nuvoton@a24d9e06abe3:~/share/yocto/build$ bitbake linux-ma35d1 -c cleanall && bitbake linux-ma35d1
```

- Clean the old Image and build the newer Image

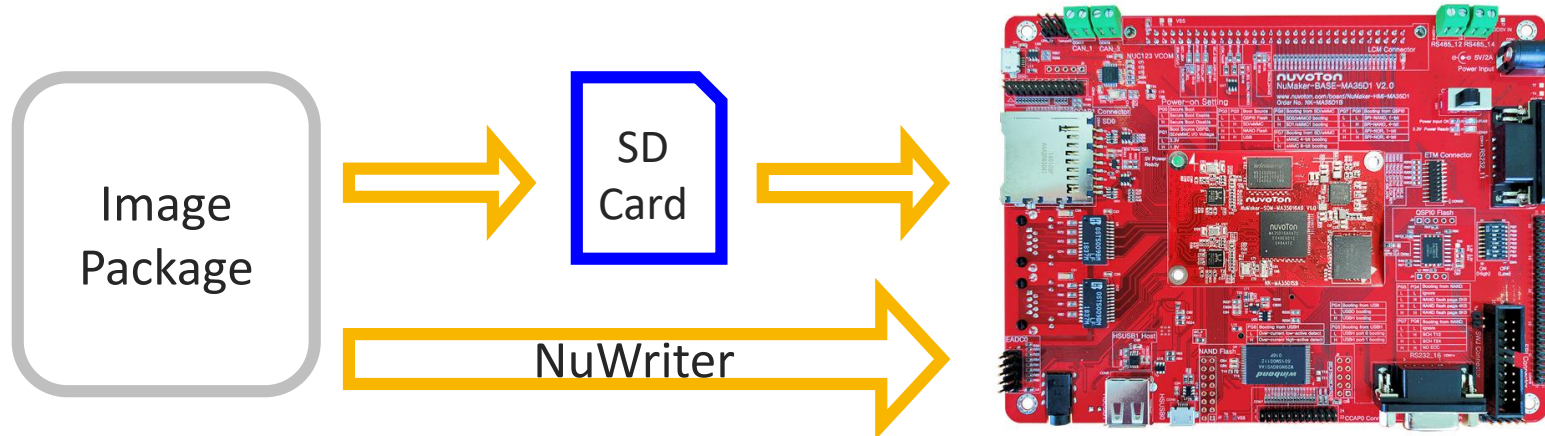
```
$ bitbake nvt-image-qt5 -c cleanall && bitbake nvt-image-qt5
```

Image Programming and System Boot



| Image Programming and System Boot

- Nuvoton provide two method to program Image to evaluation board
 1. Program image to SD card
 2. Program image to any storage in evaluation board with NuWriter
- First way is a quick way to evaluate application because of programming time less than second way
- The power on setting could be referred to MA35D1 user manual



| Image programming and Boot from SD Card

- Leave the Docker container or create a new command window

```
$ nuvoton@a24d9e06abe3:~$ exit
```

- Format your SD card first and search the SD card number

```
$ sudo fdisk -l
```

```
Disk /dev/sdb: 14.4 GiB, 15489564672 bytes, 30253056 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000
```

- Copy the image to SD card

```
$ sudo dd if=nvt-image-qt5-numaker-som-ma35d16a81.sdcard of=/dev/sdb
```

- After copying, insert the SD card to evaluation board, switch power setting to SD Booting, and boot
- Login password : root

NuWriter Setting and Image Programming (1/5)

- Connect the USB port and switch Power on Setting to USB booting
- Open nuwriterUI.exe or nuwriterUI.py, and attach the DDR parameter
- Step:
 1. Boot from USB
 2. Browse DDR File
 3. Attach
- Notice:

Remember install WinUSB4NuVCOM.exe and close virtual machine (VMware)



NuWriter Setting and Image Programming (2/5)

~/yocto/build/tmp-glibc/deploy/images/numaker-som-ma35d16a81/

- NAND:
 - pack-core-image-minimal-numaker-som-ma35d16a81-nand.bin
- SPI-NAND:
 - pack-core-image-minimal-numaker-som-ma35d16a81-spinand.bin
- SD:
 - pack-core-image-minimal-numaker-som-ma35d16a81-sdcard.bin
- Erase the flash you want to program to (NAND)
 - Choose NAND
 - Erase



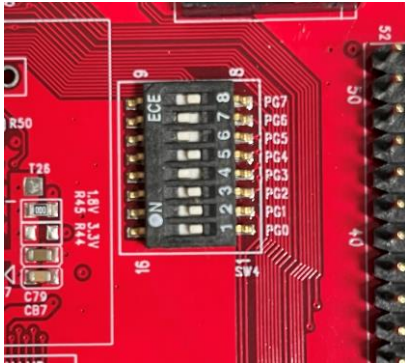
NuWriter Setting and Image Programming (3/5)

- Program NAND flash
 - Choose Download
 - Choses NAND
 - Browse NAND Package
 - Image : Pack
 - Write



NuWriter Setting and Image Programming (4/5)

- Switch the power setting to boot from NAND flash



- Push the reset button, and MA35D1 will boot from NAND flash

```
Nuvoton Release Distro 5.5-dunfell numaker-som-ma35d16a81 ttyS0
numaker-som-ma35d16a81 login: root
```

| NuWriter Setting and Image Programming (5/5)

- This method recommend to the advanced developer
- If you want to replace part of Image package like Linux kernel or DTB, refer to below files

```
~/yocto/build/tmp-glibc/deploy/images/numaker-som-ma35d16a91/nuwriter
```

- pack-nand.json
- pack-sdcard.json
- pack-spinand.json

- The three files show the details of every part of Linux package

You can replace the part of Image by NuWriter

Fast Application Development



| Setup Compiler by Yocto (1/2)

- Set up cross-compile environment can reduce some developing time, because you don't need to use Yocto re-build the whole Linux image and program it. After use MA35D1 toolchain to compile and program it to evaluation board, you can execute it directly on evaluation board.
- Make a toolchain installer, and it may take about 1 hour

```
$ bitbake nvt-image-qt5 -c populate_sdk
```

- Go to the following path and execute the shell file

```
~build/tmp-glibc/deploy/sdk $ ./oecore-x86_x64-aarch64-toolchain-5.5-dunfell.sh
```

Nuvoton Release Distro SDK installer version 5.5-dunfell

Enter target directory for SDK (default: /usr/local/oecore-x86_64):

You are about to install the SDK to "/usr/local/oecore-x86_64", P[Y/n]?

Extracting SDK done

Setting it up ...

Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.

| Setup Compiler by Yocto(2/2)

- Add toolchain to environment variables

```
$ source /usr/local/oe-core-x86_64/environment-setup-aarch64-poky-linux
```

- Create the source code file for this example: helloworld.c

```
#include <stdio.h>
int main() {
    // printf() displays the string inside console
    printf("Hello, World!\n");
    return 0;
}
```

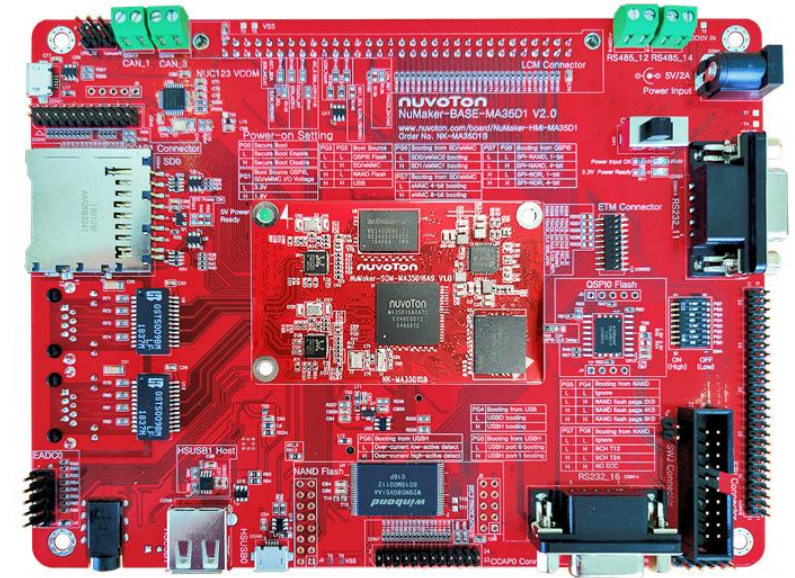
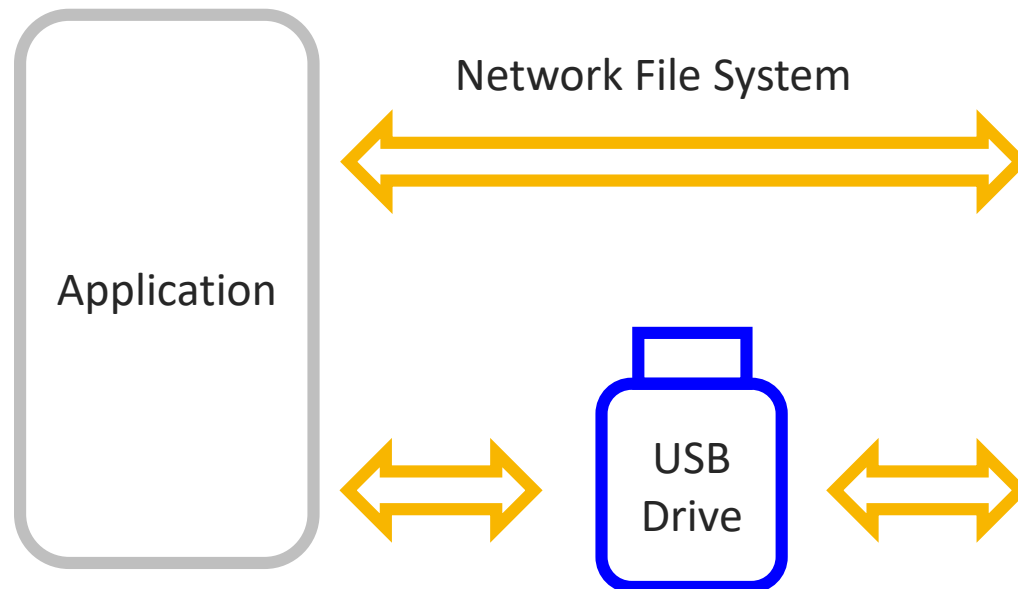
- Compile it

```
$ $CC helloworld.c -o helloworld
```

- Move the binary file to MA35D1 evaluation board and execute it

MA35D1 – Application Programming

- There are many ways to program application to evaluation board. Here Nuvoton demonstrates two methods
 - Network File System Programming
 - USB Flash Drive Programming



| Network File System Programming (1/3)

- This can cross-compile the application on PC and execute it on device through NFS rather than build the whole Linux image or use another way to send the application to device.
- The following demo is operate on Ubuntu 20.04 not in the Docker container
- First, install the network file system server on host OS

```
$ sudo apt-get install nfs-kernel-server nfs-common
```

- Create a folder to put your application code and shared with device
- Modify the network file system setting. Add the following statement in exports

```
$ sudo gedit /etc/exports
```

```
Add " *(The folder path you want to share)*(EVB's IP ADDRESS)(rw,sync,no_root_squash, no_subtree_check) "
```

```
/home/user/yocto/helloworld 192.168.0.100(rw,sync,no_root_squash,no_subtree_check)
```

- Restart the network file system service

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

Network File System Programming (2/3)

- Enable NFS client in Linux kernel

```
~/build$ bitbake linux-ma35d1 -c devshell
```

```
~/build/tmp-glibc/work-shared/ma35d1-evb/kernel-source# make menuconfig
```

```
File systems --->
```

```
[*] Enable POSIX file locking API
```

```
[*] Network File Systems --->
```

```
<*> NFS client support
```

```
<*> NFS client support for NFS version 2
```

```
<*> NFS client support for NFS version 3
```

```
[*] NFS client support for the NFSv3 ACL protocol extension
```

```
<*> NFS client support for NFS version 4
```

- Leave the kernel setting

```
~/build/tmp-glibc/work-shared/ma35d1-evb/kernel-source# exit
```

- Add nfs-utils to image and add the command to /build/conf/local.conf

```
IMAGE_INSTALL_append = "nfs-utils"
```

- Re-compile image and program to device

```
~/build$ bitbake linux-ma35d1 -C compile
```

```
~/build$ bitbake nvt-image-qt5
```


Network File System Programming (3/3)

- Create a folder in device terminal

```
root@ma35d1-evb:~# mkdir -p /mnt/nfs
```

- Check the IP address if device and host are in the same internet domain
- Mount the NFS on device

```
mount -o nolock -t nfs 192.168.0.103:/home/user/yocto/build/helloworld /mnt/nfs/
```

- Now, you can find the folder shared with host helloworld folder

```
root@ma35d1-evb:~# ifconfig eth0 192.168.0.100
root@ma35d1-evb:~# cd /mnt/nfs/
root@ma35d1-evb:/mnt/nfs# ls
root@ma35d1-evb:/mnt/nfs# cd ..
root@ma35d1-evb:/mnt# mount -o nolock -t nfs 192.168.0.103:/home/user/yocto/build/helloworld /mnt/nfs/
[ 64.473803] NFS: bad mount option value specified: minorversion=1
root@ma35d1-evb:/mnt# cd nfs/
root@ma35d1-evb:/mnt/nfs# ls
hello    hello.c
root@ma35d1-evb:/mnt/nfs# ./hello
Hello World!!
```

| USB Dongle Programming

- After cross-compile the application, you can copy the binary file to USB storage and execute on evaluation board
- Copy the application to USB drive, insert it to evaluation board, and confirm USB device number

```
$ fdisk -l
```

- Create a folder named usb for USB device under mnt folder, and mount on USB device

```
$ mount /dev/sda1 /mnt/usb
```

- Execute the application

```
$ ./hello
```

```
root@ma35d1-evb:/mnt/usb/Helloworld Sample# ls
hello
root@ma35d1-evb:/mnt/usb/Helloworld Sample# ./hello
Hello World!!
```

Joy of innovation
nuvoTon

Thank You

Danke

Merci

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

لكل اركش

הודות