MA35D1 Yocto Quick Start

Agenda

- Overview
- Docker Environment Setup
- Build Image by Yocto
- Programming and System Boot
- Fast Application Development

Overview

- This slide provides instructions on how to quickly build an MA35D1 image
- PC specification standard
  - CPU: Intel i5-10400
  - Memory: 16 GB DDR RAM
  - Storage: 1 TB SSD Disk (200 GB of which is empty space)
  - Operation System: Linux OS or Linux Virtual Machine (VMware provide by Nuvoton)
  - A MA35D1 Docker container

    Refer to MA35D1 Quick Start – Environment Setup section

    If you used VMware provide by Nuvoton, you have already created a MA35D1 Docker container
- Software Tool
  - Programming – NuWriter

VMware Setting

Start up with VMware

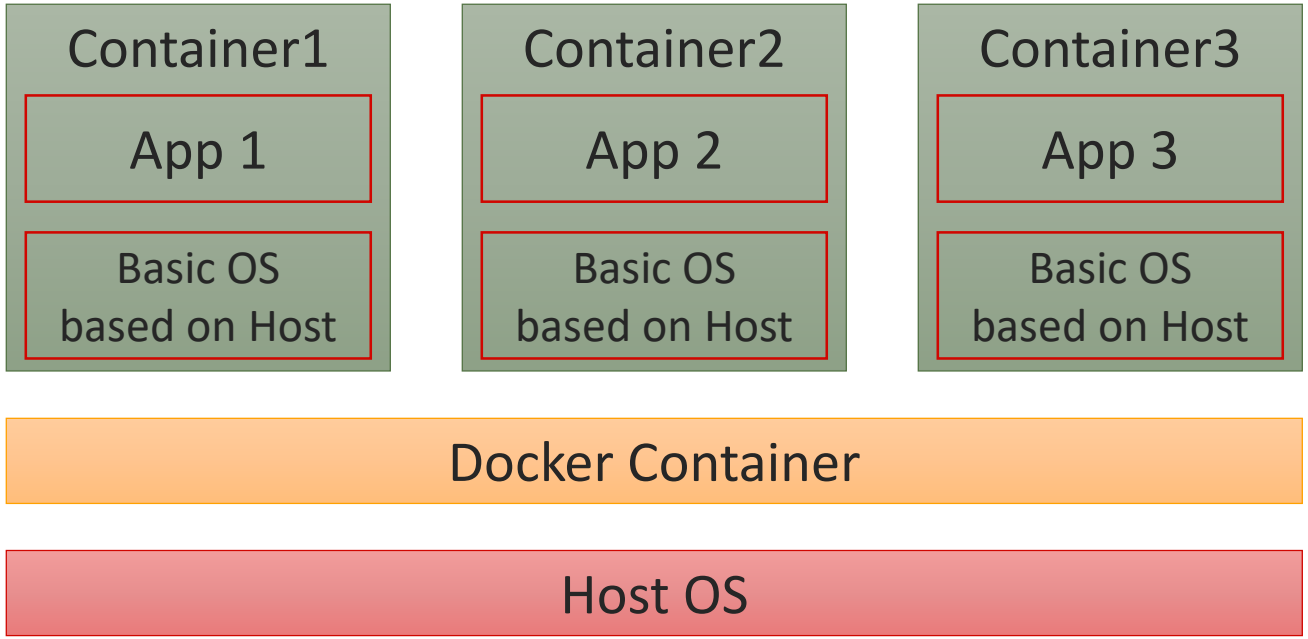
- This VMware Image offers a Linux development environment for MA35D1. If you utilize the VMware Image, you can bypass the Docker steps for building the Image
- User Name: user
  Password: user
- Execute the following command and skip to page 15

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

Docker Environment Setup

Development Environment – Docker

- Docker enables the packaging of code and its dependencies into containers.
- Each container is independent and based on the host OS, ensuring they operate in isolation without impacting each other. Containers run more efficiently than virtual machines, resulting in faster performance



Environment Setup (1/4)

- 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 (2/4)

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

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

- Install a few prerequisite packages which let apt use packages over HTTPS

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

- 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
```

- 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 (3/4)

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

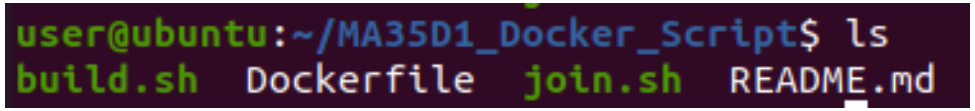
```
$ sudo apt-get update
```

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
```



Environment Setup (4/4)

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

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

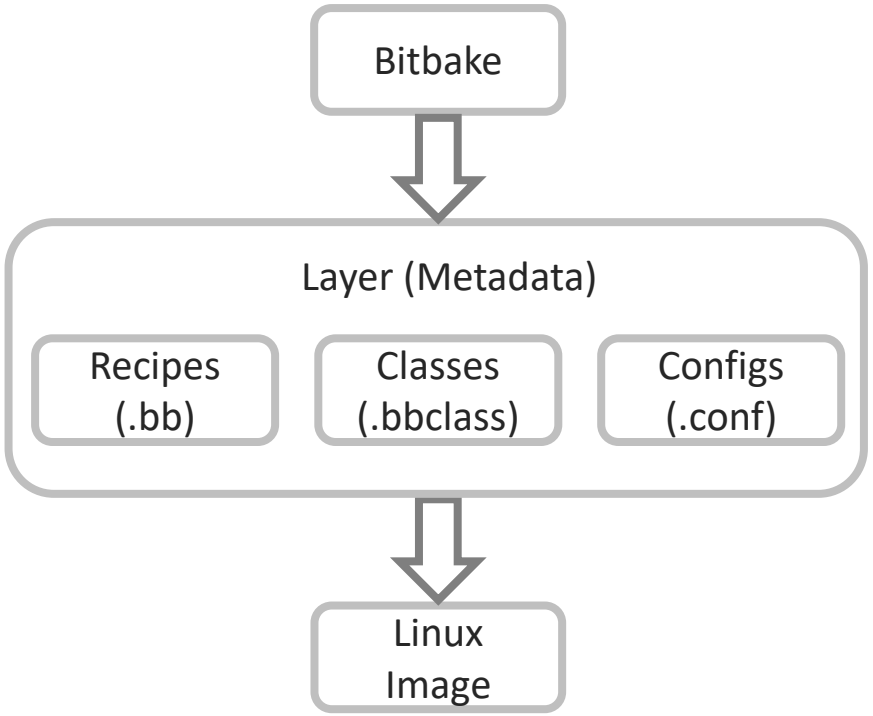
10. Enter docker image, and your command line head will be like nuvoton@a24d9e06abe3:~$

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

Build Image by Yocto

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 includes the following Metadata
  - Meta-ma35d1
  - Meta-qt5
  - Meta-virtualization
  - Meta-Pocky



Environment Setup

- Create a folder name yocto under /shared

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

- 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
```

- 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
```

- Download the yocto project and update ma35d1 source code

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

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)

- 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

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

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

- 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
```

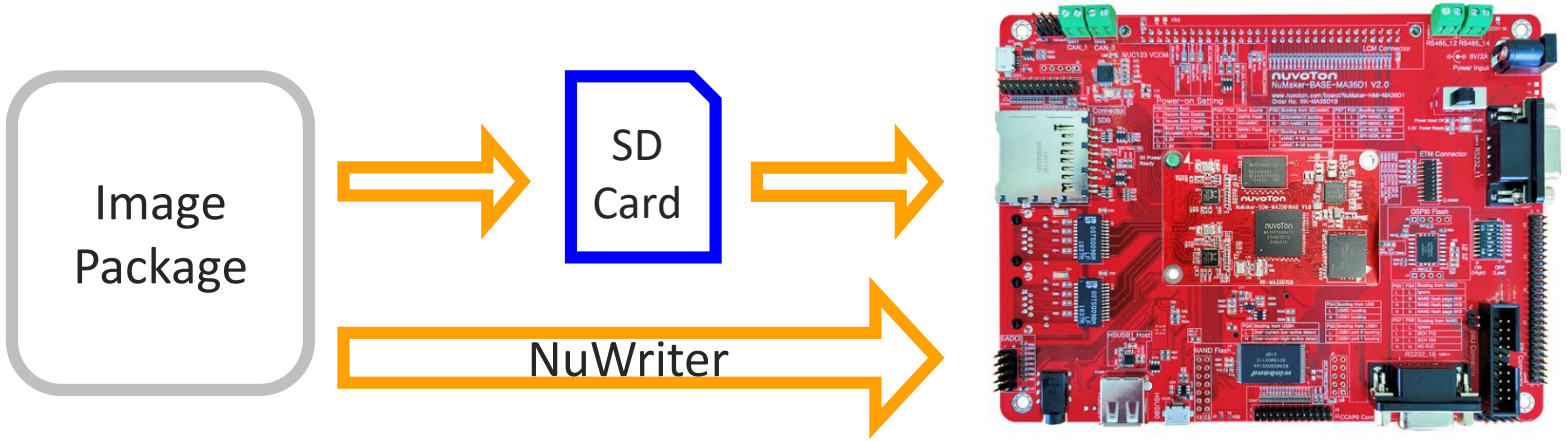
- Clean the old Image and build the newer Image

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

Programming and System Boot

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



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

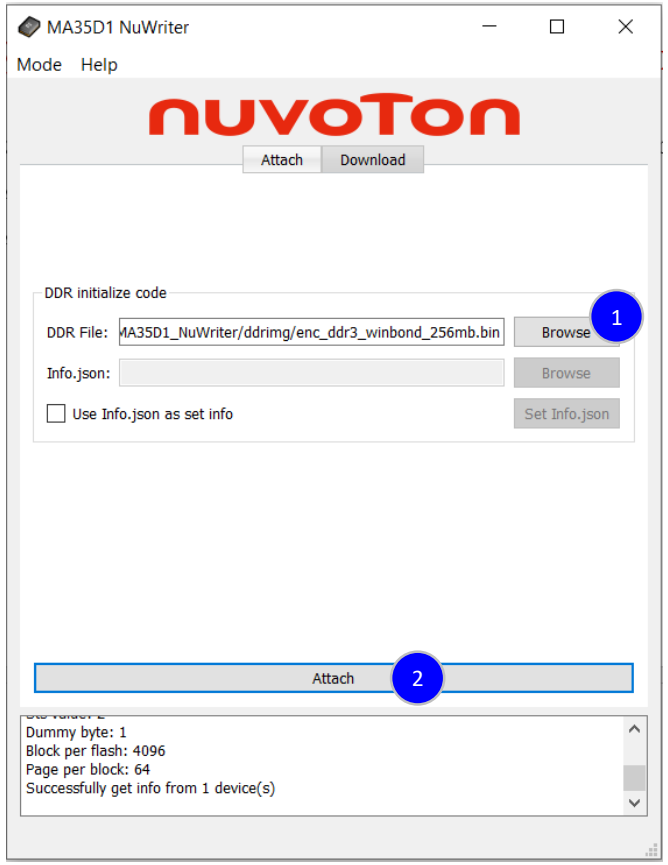
```
$ sudo dd if=nvt-image-qt5-numaker-som-ma35d16a81.sdcard of=/dev/sdb
```

- 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:

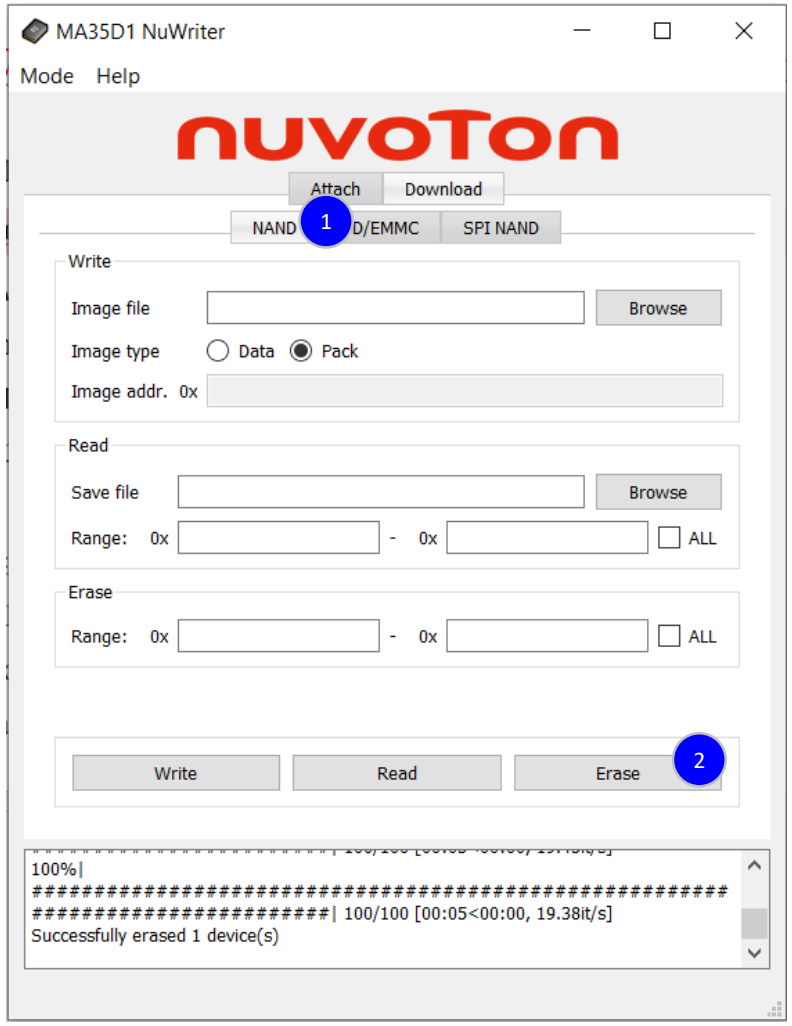
  Remember install WinUSB4NuVCOM.exe



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

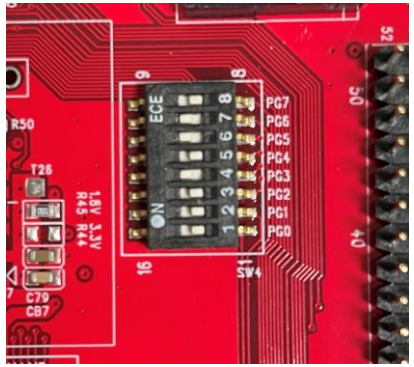


NuWriter Setting and Image Programming (3/5)

- 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-toolcahin-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/oecore-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;
}
```

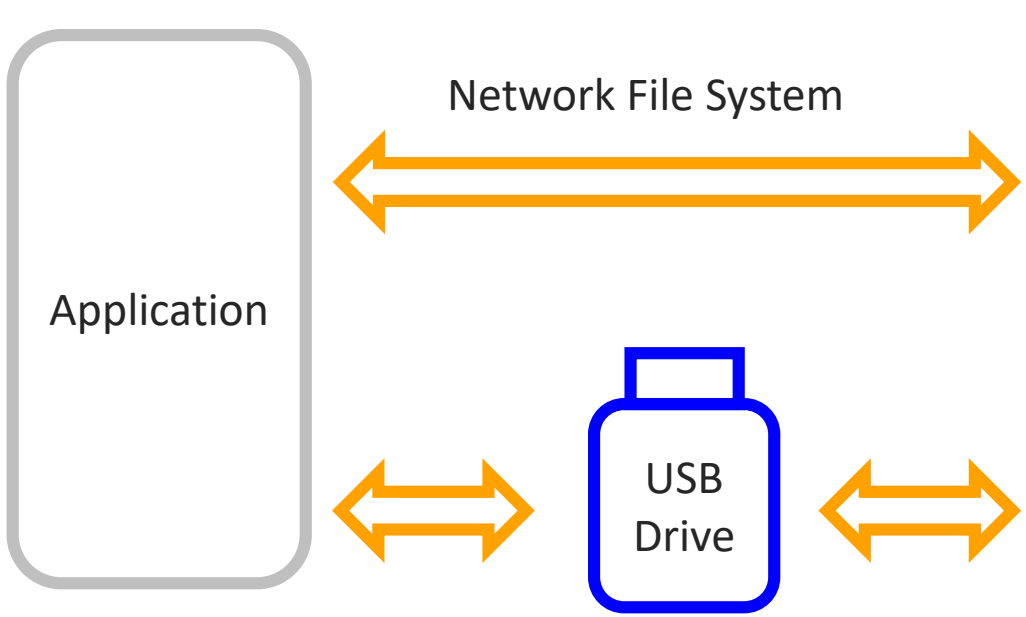
- Compile it

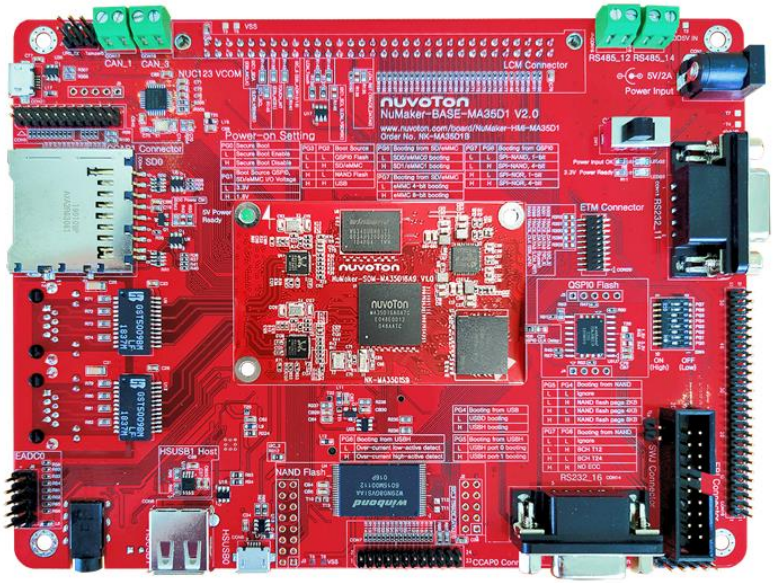
```
$ $CC helloworld.c -o helloworld
```

- 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/buil
d/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

谢谢

謝謝

Děkuji

Bedankt

Thank you

Kiitos

Merci

Danke

Grazie

ありがとう

감사합니다

Dziękujemy

Obrigado

Спасибо

Gracias

Teşekkür ederim

Cảm ơn