



感谢您选择英创 ESM7400 系列工控主板。

为了让您能够尽快地使用好我们的产品，英创公司编写了这篇《使用必读》，我们建议每一位使用英创产品的用户都浏览一遍。我们本着通俗易懂的原则，按照由浅入深的顺序，采用了大量图片和浅显的文字，以便于用户能边了解、边动手，轻松愉快地完成产品的开发。

在使用英创产品进行应用开发的过程中，如果您遇到任何困难需要帮助，都可以通过以下三种方式寻求英创工程师的技术支持：

- 1、直接致电 **028-86180660 85329360**
- 2、发送邮件到技术支持邮箱 support@emtronix.com
- 3、登录英创网站 www.emtronix.com，在技术论坛上直接提问

另，本手册以及其它相关技术文档、资料均可以通过英创网站下载。

注：英创公司将会不断完善本手册的相关技术内容，请客户适时从公司网站下载最新版本的手册，恕不另行通知。

再次感谢您的支持！

目 录

目 录.....	2
1 ESM7400 简介	3
2 搭建硬件开发平台	3
2.1 ESM7400 开发评估套件说明.....	3
2.2 必要的准备	4
2.3 开发环境的硬件连接和安装	5
3 配置开发环境	9
3.1 配置串口工具	9
3.2 编辑 userinfo.txt 文件.....	10
3.3 搭建开发环境	12
3.4 设置 NFS 文件系统挂载	12
4 基于 vscode 搭建应用软件编译环境.....	16
4.1 下载并安装 vscode,	16
4.2 安装英创公司提供的交叉编译工具链	16
4.3 配置环境变量	17
4.4 启动 vscode	17
4.5 在 vscode 中安装插件	17
4.6 配置 vscode 编译工具	18
4.7 配置 CMake configure Args.....	19
4.8 生成 CMake 编译配置文件.....	20
4.9 编译应用程序	21
4.10 工程中新添源文件	21
4.11 工程中增加线程支持库	21
附录 1 版本信息管理表	22

1 ESM7400 简介

感谢您购买英创信息技术有限公司的产品：ESM7400 系列工控主板。

ESMARC 是由英创公司设计开发的一套嵌入式主板与应用底板的连接规范，意为英创智能模块架构（Emtronix Smart Module Architecture，以下简称 ESMARC），ESM7400 系列工控主板是结构上符合 ESMARC 规范的一款主板产品。

ESM7400 系列主板是面向工业领域的高性价比、全国产嵌入式主板，以全志的四核 Cortex®-A7 芯片 A40i 为其硬件核心，ESM7400 通过预装完整的操作系统及接口驱动，为用户构造了可直接使用的通用嵌入式核心平台。目前 ESM7400 预装了 Linux-5.10.180 系统，用户应用程序开发方面，可采用 vscode 或者 QtCreator 开发环境，其编译生成的程序可直接运行于 ESM7400。英创公司针对 ESM7400 提供了完整的接口低层驱动以及丰富的应用程序范例，用户可在此基础上方便、快速地开发出各种工控产品。

ESM7400 开发的基本文档包括：

- 《ESM7400 工控主板使用必读》—ESM7400 快速入门手册，建议新客户都浏览一遍
- 《ESM7400 工控主板数据手册》—ESM7400 接口信号定义、电气特性以及各项技术指标
- 《ESM7400 工控主板编程参考手册》—ESM7400 功能接口使用方法及软件操作说明
- 《ESMARC 通用评估底板数据手册》—符合 ESMACR 规范主板的评估底板使用说明

ESM7400 的更多资料和说明请参考 ESM7400 开发光盘或登录我们的网站：

<http://www.emtronix.com/product/ESM7400.html>。

2 搭建硬件开发平台

ESM7400 工控主板使用必读的硬件环境，是以 ESM7400 开发评估套件为基础进行描述。

2.1 ESM7400 开发评估套件说明

ESM7400 开发评估套件，包含下列硬件或线材：

- **ESM7400 工控主板一张**：全志 A40i 处理器，预装嵌入式 Linux-5.10.180 实时多任务操作系统，接口资源丰富
- **ESMARC-EVB 通用开发评估底板一块**：搭载 ESM7400 并引出其板载资源。底板上提供了 ESM7400 所有板载资源的标准接口，既方便用户对 ESM7400 进行评估和开发，又为用户的外围硬件开发提供一定的参考
- **ETA312 模块**：从 ESMARC-EVB 底板的引出 ESM7400 的以太网接口 ETH2\ETH3，这是 ESM7400 的 4 路网络接口中的两路 100Mbps 以太网接口
- **USB-RS232 串口线一条**：用于输出调试信息，并作为控制终端通讯使用
- **以太网连接线一条**：直连方式，用于进行目标机系统的管理维护以及开发网络方面的应用功能
- **直流电源线一条**：红黑双色，+5V，用于为系统供电
- **DC5V/4A 电源适配器一只**：可用于系统评估测试，如果功率不够，需用户自行准备符合功率更大的 DC5V 电源
- **开发资料光盘一张**：为用户的开发提供丰富翔实的软硬件资料

根据客户所开发的产品不同的需求，除了以上一些客户开发的必要配备外，客户可能还有一些其它开发附件，如：

- 各种尺寸的彩色显示屏，如 10.1 寸（1024×600）、12 寸（1280×800）等
- 常用通讯模块（如：4G，Wifi 等）
- 客户所需要的其它附件

这些附件的配套使用方法，请参考该产品的使用说明或手册。

2.2 必要的准备

用户要利用 ESM7400 进行开发，需要作如下一些必要准备：

- 准备一台带以太网接口、USB 接口的 PC 机作为开发主机，该 PC 机需安装 Linux/Ubuntu 操作系统，对于不熟悉 Linux 系统的客户，建议选用 Ubuntu 系统，文章中涉及到 PC 发行版 Linux 系统的时候，会以 Ubuntu 系统为例讲解。

注：由于交叉编译工具链时 Linux 64 位版的，所以用户必需使用 64 位 Linux 系统，可以使用 Windows 虚拟机安装 Linux 系统。

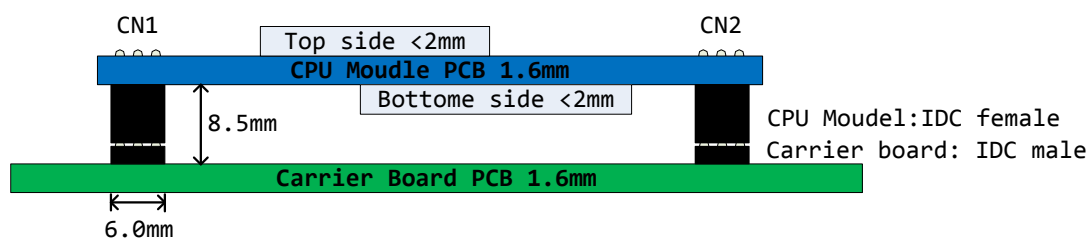
- 准备一台网络连接设备（集线器/交换机/路由器）。
- 准备一只可供临时存储数据的 U 盘。
- 串口超级终端软件，在英创提供的光盘->工具中，有基于 Windows 版本的 teraterm-4.76 串口超级终端软件

2.3 开发环境的硬件连接和安装

在以上条件准备好以后，就可以按照如下顺序进行开发环境的硬件连接了。

1、ESM7400 两侧有两个三排母座（CN1 和 CN2），这两个母座将 ESM7400 的板载接口资源引出，而开发评估底板上安装有相对应的两个三排插针（CN1 和 CN2），ESM7400 就象一个大芯片一样对插在开发评估底板上，从而构成一套较完整的开发系统，如下图所示。

注：在用户收到的开发评估套件中，ESM7400 已经插在开发评估底板上，开发过程中用户如需进行插拔，请注意插针和插座的序号对应。



英创工控主板与开发评估底板的连接关系

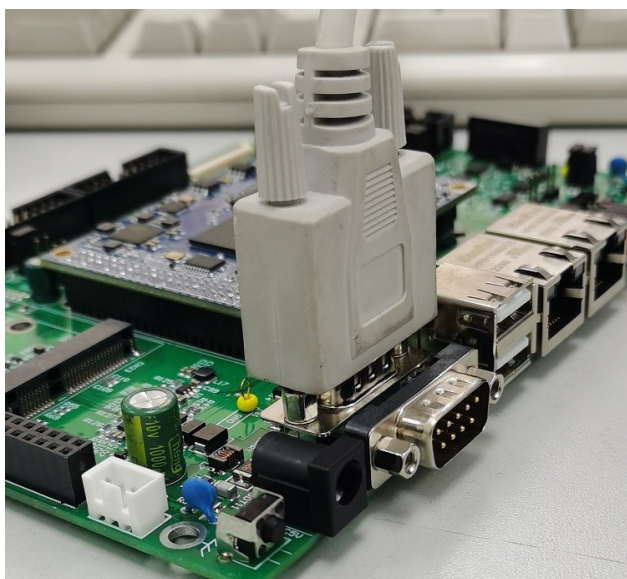
2、ESM7400 有两种工作模式：调试模式和运行模式。

- **调试模式**:是指开机以后系统处于调试状态，此时用户可以通过超级终端来操作 ESM7400，实现应用程序下载调试、文件管理等功能。在开发阶段，系统总是处于这种状态。
- **运行模式**:是指开机以后系统自动开始执行用户指定的程序。开发完成，进入实际

应用时系统总是处于这种状态。

ESM7400 工作于上述的哪一种模式，是通过开发评估底板上的跳线器 JP1 来选择的。JP1 短接，则工作于调试模式；JP1 断开，则工作于运行模式。

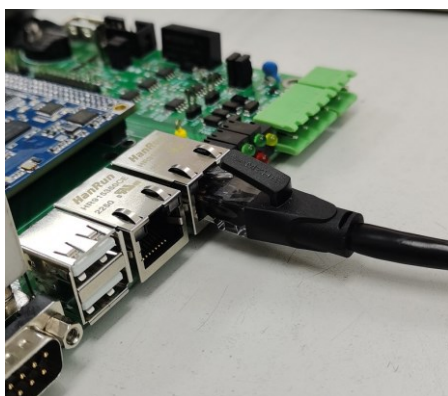
3、将套件中串口连接线的两端分别接入开发主机的串口和 ESM7400 开发评估底板的控制台串口，如下图所示。



连接调试串口

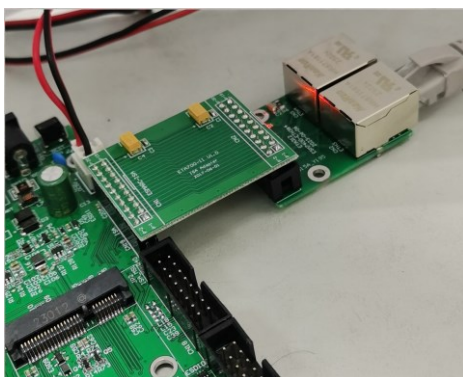
4、用户可以用交换机/路由器/集线器将主机和 ESM7400 接入同一个网络，如下图所示。这样开发主机和 ESM7400 就能够建立起网络连接。

注：ESM7400 的 ip 地址一定要与开发主机的 ip 地址设置在同一网段内。



将开发主机和 ESM7400 接入以太网

如果需要同时测试 ESM7400 的 ETH2、ETH3，则需要使用 ETA312 模块，从 ESMARC EVB 底板的 ISA 接口，转接出 ETH2、ETH3。



利用 ETA312 模块，引出 ETH2、ETH3

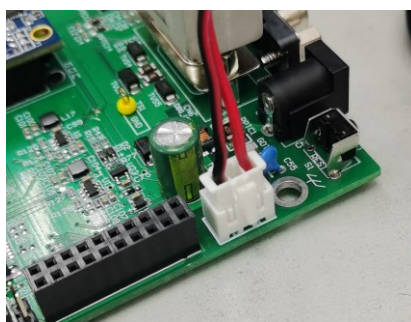
5、如果用户在英创购买了显示屏，可以将 LCD 显示屏的排线直接连接到 ESM7400 评估底板的 CN20—LVDS 显示接口，具体的连接方法可以参考英创公司网站的文章[《彩色 TFT LCD 的连接方法》](#)。

至此，ESM7400 运行的基本硬件环境已搭建完成。

现在可以给 ESM7400 通电，即将+5V 直流电源线接头插在底板上的电源插头（注意正负方向）。如果使用电源适配器接头，则电源极性是：内正、外负



适配器电源接头极性



插针电源连接极性：红色为正 5V，黑色为 GND

此时，ESM7400 上的红色电源 LED 指示灯亮，蓝色 LED 灯在一段时间的闪烁，表示 ESM7400 系统启动正常。如果连接了串口到 PC 机并打开了 PC 端的超级终端，则会看到

ESM7400 的启动信息。

ESM7400 板载嵌入式 Linux-5.10.180 实时多任务操作系统，可以支持多种高级应用，比如 qt-5.10, mysql 以及 java 等。用户需要在 PC 上使用 Linux 操作系统进行应用程序的开发。下面将介绍基于 Ubuntu 操作系统，利用 vscode 快速搭建起 ESM7400 的软件开发平台。

3 配置开发环境

3.1 配置串口工具

ESM7400 的运行信息会通过串口工具显示在开发主机的显示屏上；用户想要对 ESM7400 的文件系统进行操作也需通过超级终端以命令行方式进行。ESM7400 的调试串口默认使用 115200， 8N1， no hardware flow control(无硬件流控)格式协议。Linux 下常用的串口软件为 minicom，在 Ubuntu 系统中使用以下命令安装：

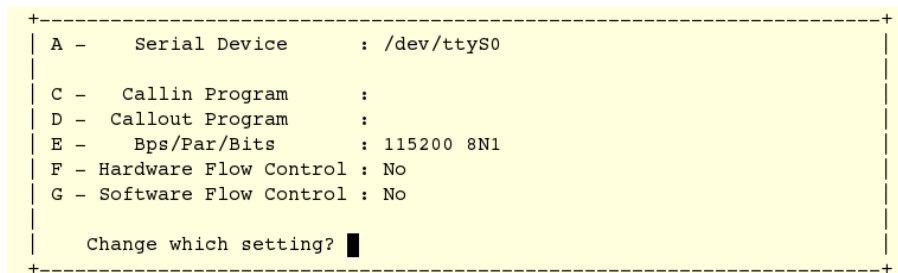
```
sudo apt-get install minicom
```

安装完成后，按照如下步骤设置 minicom：

- 1、运行命令进入设置界面: `sudo minicom -s`
- 2、按上下键选中 **Serial port setup**，按回车键进入串口配置界面：



3、按每行前面的大写字母对应的按键配置每一项，如按 **A** 配置串口设备，除了 **Serial Device** 需要根据自己的电脑来设置，其他项都需要设置成于下图相同：



参数配置

4、完成以后给 ESM7400 上电，超级终端将显示出 ESM7400 的开机启动信息。启动成功以后回车进入命令行，此时可以通过超级终端使用 Linux 的命令对 ESM7400 进行操作。

```
J-Boot SPL 2019.07 (Jul 24 2023 - 17:27:19 +0800)
DRAM: 512 MiB
Trying to boot from MMC2

J-Boot 2019.07 (Jul 24 2023 - 17:27:19 +0800) Allwinner Technology
CPU: Allwinner R40 (SUN8I 1701)
Model: ESM7400 - A40i
I2C: ready
DRAM: 512 MiB
MMC: Device 'mmc@1c11000': seq 1 is in use by 'mmc@1c10000'
mmc@1c0f000: 0, mmc@1c10000: 2, mmc@1c11000: 1
Loading Environment from MMC... OK
In: serial@1c28000
Out: serial@1c28000
Err: serial@1c28000
readsplashbmp: -> 800*480
Net: phy interface7
eth0: ethernet@1c50000
```

ESM7400 部分启动信息

3.2 编辑 userinfo.txt 文件

userinfo.txt 文件有三个作用：

- 1、配置 ESM7400 的网络参数，让 ESM7400 与开发主机处于同一网段
- 2、配置 NFS 挂载参数，让开发主机的指定目录能挂载到 ESM7400 的指定目录下
- 3、配置应用程序参数，这样开发完成以后 ESM7400 将自动根据该参数执行应用程序

userinfo.txt 文件的内容及格式如下（双斜线后不同字体和颜色的文字为加注的说明文字，并不包括在 userinfo.txt 文件中，‘=’号前后没有空格）：

```
[LOCAL_MACHINE] // ESM7400 信息
DHCP="0" // 配置 DHCP 客户端信息。设为“0”则 DHCP 关
// 闭，用户需手动设置网关、IP 地址、子网掩码；
// 设为“1”则 DHCP 开启，ESM7400 将自行获取
// 上述网络参数
DefaultGateway="192.168.201.20" // 默认网关，根据用户所在的实际运行网络设置
IPAddress="192.168.201.90" // ESM7400 的 IP 地址，由用户自行设置
```

```
SubnetMask="255.255.255.0" // 子网掩码，根据用户所在的实际运行网络填写
[NFS_SERVER] // NFS 挂载信息
IPAddress="192.168.201.85" // 开发主机 IP 地址，根据用户所在的实际运行
// 网络设置
Mountpath="/d/public" // 开发主机上被挂载的文件夹名，本文中以
// "public"为例，用户可自行选择任意文件夹，
// 需注意的是必须带上文件夹路径
[USER_EXE] // 用户程序信息
Name="/mnt/mmc/hello" // 系统开机自动执行的程序及其存储路径。开发
// 完成以后用户将自己的应用程序文件名填在
// 双引号之间取代目前的默认文件名，开机即可
// 自动运行（注意，用户也可以在
// /mnt/nandflash/下建立子目录存放应用程序，
// 配置此项参数的时候一定要带上绝对路径）
Parameters="" // 系统开机自动执行的程序的参数配置。开发完
// 成以后在此处填入实际应用程序的参数，如果
// 没有则不填，但必须保留双引号
```

根据用户的实际网络参数编辑好 `userinfo.txt`，存入 U 盘，将 U 盘接入 ESM7400 开发评估底板的 USB 接口，短接 JP1 使 ESM7400 处于调试模式，然后上电。系统将自动搜索 USB 接口，将读到的 `userinfo.txt` 文件存放到 `/mnt/mmc` 目录中，并按照其内容配置 ESM7400 的网络参数。启动完成以后，可以通过超级终端使用 `ifconfig` 命令查看是否配置完成。

`userinfo.txt` 写入 ESM7400 以后，系统每次开机都会自动读取该文件并按照文件内容进行配置。如果其中任何参数需要重新配置，可编辑好 `userinfo.txt` 并重复执行上述步骤。

如果要让系统开机自动挂载 NFS，则 ESM7400 上电启动之前必须先启动相应的 NFS 服务器。挂载成功后，挂载的路径在 ESM7400 的 `/mnt/nfs` 目录下。

如果 ESM7400 处于运行模式，则开机以后会自动执行 `Name="/mnt/mmc/*"` 中设置的

应用程序。英创为用户分配的存储地址固定在 `/mnt/mmc` 文件夹下，用户可以将应用程序直接存在这个目录中，也可以在此目录下建立子目录存放应用程序。用户配置该项参数的时候要带上绝对路径，否则系统无法找到执行文件。

注：Linux 操作系统严格区分大小写，因此此处的用户应用程序名称必须与实际的程序名称完全一样，包括大小写字母。

3.3 搭建开发环境

如果客户需要开发 Qt 图形界面，那么建议使用 Qtcreator 进行开发，QtCreator 提供了一套便于操作的 IDE 界面，搭建 Qtcreator 开发环境的方法可以参考《[ubuntu+Qt 开发环境搭建](#)》。

如果不需要使用 Qt，客户也可以选用在 Ubuntu 系统中使用 vscode 来进行开发。详细流程，请见：[4 基于 vscode 搭建应用软件编译环境](#)

3.4 设置 NFS 文件系统挂载

用户在开发主机中完成的应用程序必须通过一定的方法下载到 ESM7400 的存储器中，才能进行运行测试。这种文件复制的方法有很多，英创公司推荐使用 NFS 文件系统挂载，此方法可以将开发主机中用户指定的某一个目录挂载到 ESM7400 的 Linux 目录中。这样，用户在开发主机中完成的应用程序就可以直接放在该目录下，然后通过超级终端让其在 ESM7400 上进行运行测试。

1、NFS 服务器安装

NFS 的安装是非常简单的，只需要两个软件包即可，而且在通常情况下，是作为系统的默认包安装的，如果没有请按照自己所用 Linux 版本说明进行安装。

`nfs-utils-*`：包括基本的 NFS 命令与监控程序

`portmap-*`：支持安全 NFS RPC 服务的连接

在 Ubuntu 系统中，可以通过以下命令安装：

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

2、配置 NFS 服务器的工作目录

NFS 服务器的配置相对比较简单,只需要在相应的配置文件中设置,然后启动 NFS 服务器即可。

NFS 服务的配置文件为 `/etc/exports`,这个文件是 NFS 的主要配置文件,不过系统并没有默认值,所以这个文件不一定会存在,可能要使用 `vim` 手动建立,然后在文件里面写入配置内容。

`/etc/exports` 文件内容格式:

<输出目录> [客户端 1 选项 (访问权限,用户映射,其他)] [客户端 2 选项 (访问权限,用户映射,其他)]

a. 输出目录:

输出目录是指 NFS 系统中需要共享给客户机使用的目录;

b. 客户端:

客户端是指网络中可以访问这个 NFS 输出目录的计算机

客户端常用的指定方式

指定 ip 地址的主机: `192.168.0.200`

指定子网中的所有主机: `192.168.0.0/24 192.168.0.0/255.255.255.0`

指定域名的主机: `david.bsmart.cn`

指定域中的所有主机: `*.bsmart.cn`

所有主机: `*`

c. 选项:

选项用来设置输出目录的访问权限、用户映射等。

NFS 主要有 3 类选项:

访问权限选项

设置输出目录只读: `ro`

设置输出目录读写: `rw`

用户映射选项

`all_squash`: 将远程访问的所有普通用户及所属组都映射为匿名用户或用户组 (`nfsnobody`);

`no_all_squash`: 与 `all_squash` 取反 (默认设置);

root_squash: 将 root 用户及所属组都映射为匿名用户或用户组（默认设置）;

no_root_squash: 与 rootsquash 取反;

anonuid=xxx: 将远程访问的所有用户都映射为匿名用户，并指定该用户为本地用户（UID=xxx）;

anongid=xxx: 将远程访问的所有用户组都映射为匿名用户组账户，并指定该匿名用户组账户为本地用户组账户（GID=xxx）;

其它选项:

secure: 限制客户端只能从小于 1024 的 tcp/ip 端口连接 nfs 服务器（默认设置）;

insecure: 允许客户端从大于 1024 的 tcp/ip 端口连接服务器;

sync: 将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性;

async: 将数据先保存在内存缓冲区中，必要时才写入磁盘;

wdelay: 检查是否有相关的写操作，如果有则将这些写操作一起执行，这样可以提高效率（默认设置）;

no_wdelay: 若有写操作则立即执行，应与 sync 配合使用;

subtree: 若输出目录是一个子目录，则 nfs 服务器将检查其父目录的权限(默认设置);

no_subtree: 即使输出目录是一个子目录，nfs 服务器也不检查其父目录的权限，这样可以提高效率;

配置好之后启动 NFS 服务:

```
#sudo service portmap start
```

```
#sudo service nfs start
```

注: 还需自行设置防火墙, 由于 Linux 桌面版本很多, 每个设置都有一定区别, 如果 NFS 搭建有问题可以在网上查找更多详细资料。

3、利用 userinfo.txt 配置 ESM7400 的 IP 地址

确认 userinfo.txt 文件已准备好, 并存入 U 盘。将 U 盘接在 ESMARC EVB 的任意 USB 接口上, 然后为系统上电。英创在 ESM7400 上为开发主机指定的挂载点是 /mnt/nfs, 因此, 在超级终端中使用命令 `cd /mnt/nfs` 进入 nfs 文件夹, 使用命令 `ls` 查看, 可以看到开发主机上 public 文件夹下的内容, 如下图所示, 表示挂载成功。

```
[root@ESM7400 ~]#cd /mnt/nfs
[root@ESM7400 /mnt/nfs]#ls
ESM7400
Linux-Kernel
Software
```

查看挂载到 Linux 目录下开发主机中的文件夹

4、如果开机挂载没有成功或者使用中连接中断，建议检查网络连接，然后手动键入命令进行挂载：

```
mount -t nfs -o nolock,tcp 192.168.201.85:/d/public /mnt/nfs
```

上述命令中的红字部分仅为示例，用户应填写自己实际的开发主机 IP 地址和挂载文件夹目录。

5、如果挂载仍然失败，建议检查 NFS 服务器端防火墙、NSF 服务器运行状态、NFS 配置、ESM7400 网络（可以 ping），再重启工控主板，然后再次尝试

经过这两章的介绍，ESM7400 的软硬件开发环境搭建均已完成，接下来用户可以进行应用程序的开发了。

4 基于 vscode 搭建应用软件编译环境

为了方便下面的操作流程的描述，这里的用户名为： em

(注：4.1、4.2、4.5 这三个步骤，只需要在第一次搭建应用软件编译环境时，操作一次即可，无需重复操作)

4.1 下载并安装 vscode，

例如文件名为： code_1.80.1-1689183569_amd64.deb

```
sudo dpkg -i code_1.80.1-1689183569_amd64.deb
```

4.2 安装 esm7400 的交叉编译工具链

将 ESM7400-toolchain_x86.sh 文件复制到开发主机，然后调用 chmod 命令修改文件属性为可执行文件：

```
em@em:~$ chmod +x ESM7400-toolchain_x86.sh
em@em:~$
```

接着直接运行，便开始安装，这里需要输入安装路径(或使用默认安装路径)，然后输入‘y’ 进行安装：

```
em@em:~$ ./ESM7400-toolchain_x86.sh
Emtronix ESM7400 SDK installer version gcc-7.4.1 & qt-5.9.0
=====
Enter target directory for SDK (default: /opt/esm7400/toolchain): ~/esm7400/toolchain
```

这里，安装路径为“~/esm7400/toolchain”，即当前用户的 esm7400/toolchain 目录下。安装完成后，进入该目录，用 ls 进行查看以确认，在 toolchain 目录下存在 4 个项目。

```
em@em:~$ mkdir esm7400
em@em:~$ cd esm7400
em@em:~/esm7400$ mkdir toolchain
em@em:~/esm7400$ cd toolchain/
em@em:~/esm7400/toolchain$ ls -l
total 108356
-rw-rw-r-- 1 em em 1042 Jul 26 15:01 environment-setup-arm-linux-gnueabihf
-rw-r--r-- 1 em em 110949328 Jul 14 17:49 gcc-llvm-7.4.1-2019.02-x86_64_arm-linux-gnueabihf.tar.xz
em@em:~/esm7400/toolchain$
```

4.3 配置环境变量

在~/esm7400/toolchain 目录下，运行 source 指令，配置当前窗口环境变量，并查看环境变量是否成功：

source environment-setup-arm-linux-gnueabi

```
em@em:~/esm7400/toolchain$ source environment-setup-arm-linux-gnueabi
em@em:~/esm7400/toolchain$ arm-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/home/em/esm7400/toolchain/gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi/bin/../libexec/gcc/arm-linux-gnueabi/7.4.1/lto-wrapper
Target: arm-linux-gnueabi
Configured with: '/home/tcwg-buildslave/workspace/tcwg-make-release_0/snapshots/gcc-git-linaro-7.4-2019.02/configure' SHELL=/bin/bash --with-mpcc=/home/tcwg-buildslave/workspace/tcwg-make-release_0/build/builds/destdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release_0/build/builds/destdir/x86_64-unknown-linux-gnu --with-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release_0/build/builds/destdir/x86_64-unknown-linux-gnu --with-gnu-as --with-gnu-ld --disable-libmudflap --enable-lto --enable-shared --without-included-gettext --enable-nls --with-system-zlib --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id --disable-libstdcxx-pch --enable-c99 --enable-clocales-gnu --enable-libstdcxx-debug --enable-gnu-unique-object --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --disable-multilib --with-float=hard --with-fpu=vfpv3-d16 --with-mode=thumb --with-tune=cortex-a9 --with-arch=armv7-a --enable-threads-posix --enable-multiarch --enable-libstdcxx-time=yes --enable-gnu-ld-direct-function --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release_0/build/builds/destdir/x86_64-unknown-linux-gnu --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release_0/build/builds/destdir/x86_64-unknown-linux-gnu/arm-linux-gnueabi/libc --enable-checking=release --disable-bootstrap --enable-languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=arm-linux-gnueabi --prefix=/home/tcwg-buildslave/workspace/tcwg-make-release_0/build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 7.4.1 20181213 [linaro-7.4-2019.02 revision 56ec6f6b99cc167ff0c2f8e1a2eed33b1edc85d4] (Linaro GCC 7.4-2019.02)
em@em:~/esm7400/toolchain$
```

输入 arm-linux-gnueabi-gcc -v，可以看到 gcc 版本为 7.4.1，则环境变量设置成功。

注意：在当前命令窗口配置的环境变量，仅对当前窗口有效，即：运行 source 的窗口不能关闭，并且后续启动 vscode 也必须在该窗口下，否则会丢失已配置的环境变量。

4.4 启动 vscode

如需要创建应用程序 test_hello,并放置在/home/em/esm7400/test_hello 目录中，则先在 esm7400 目录下创建 test_hello 文件夹，并进入该目录，再启动运行 vscode:

```
em@em:~/esm7400/toolchain$ cd ~/esm7400/
em@em:~/esm7400$ mkdir test_hello
em@em:~/esm7400$ cd test_hello/
em@em:~/esm7400/test_hello$ code .
em@em:~/esm7400/test_hello$
```

启动 vscode 的指令为：code .。

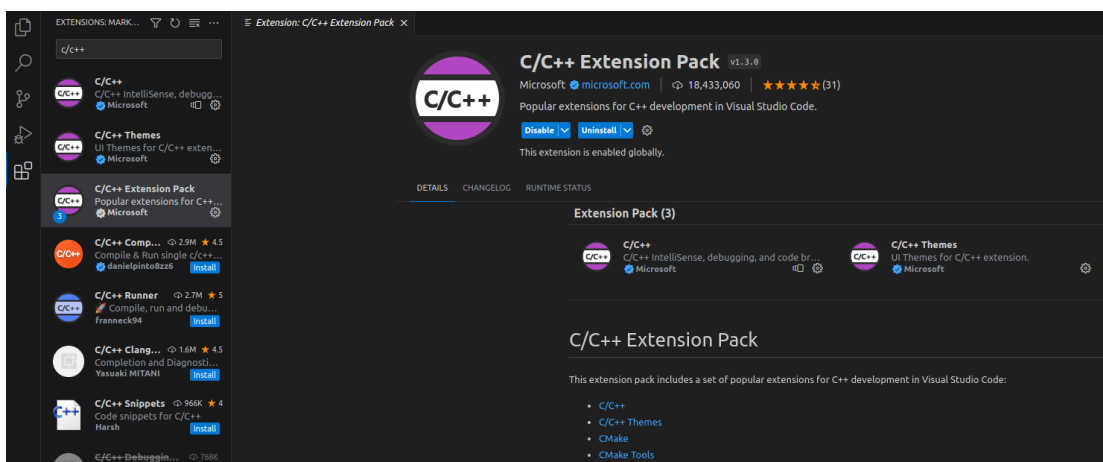
注意：vscode 运行后，启动 vscode 的指令窗口不能关闭，必须保持打开，直到退出 vscode。

4.5 在 vscode 中安装插件

启动 vscode 后，点击左侧 Extension 图标，并搜索 C/C++ Extension Pack，该插

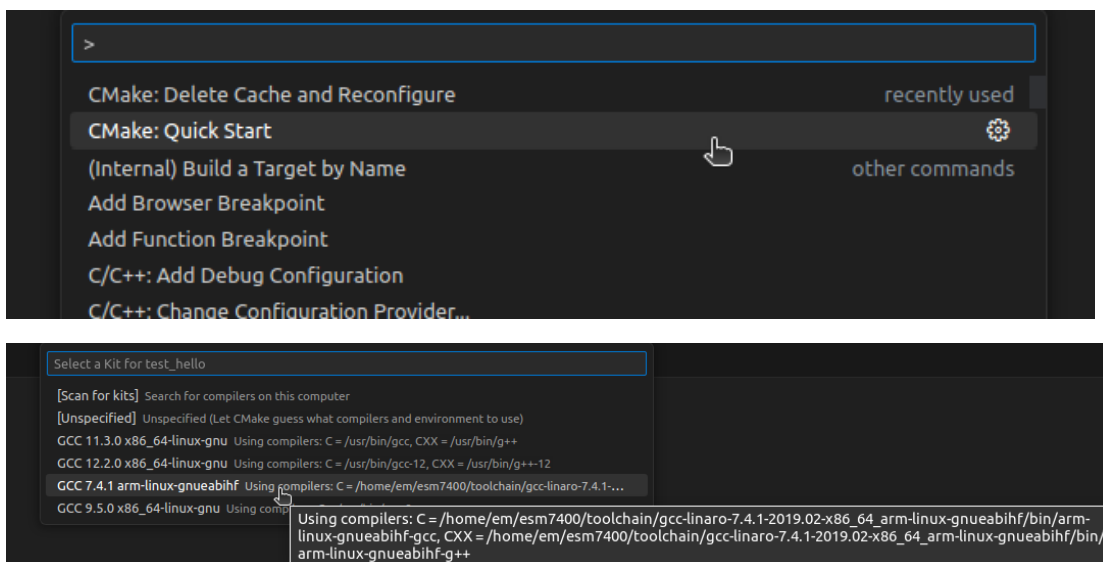
件会一次性安装 3 个插件：C/C++、C/C++ Themes、CMake Tools。

安装插件，仅仅是针对才安装的 vscode，如果插件已安装过，跳过该步骤。



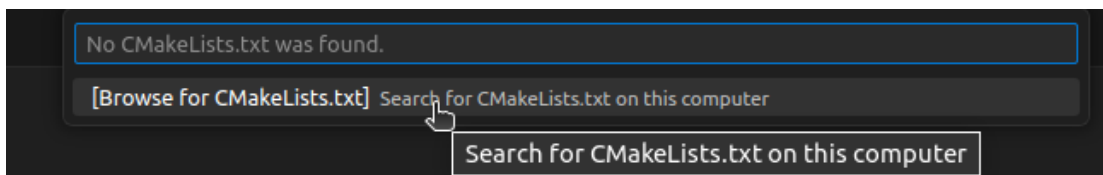
4.6 配置 vscode 编译工具

在 vscode 界面，按组合键：ctrl+shift+p，在 vscode 上方弹出的选项中，选择 CMake: Quick Start，再选择 GCC7.4.1 arm-linux-gnueabi

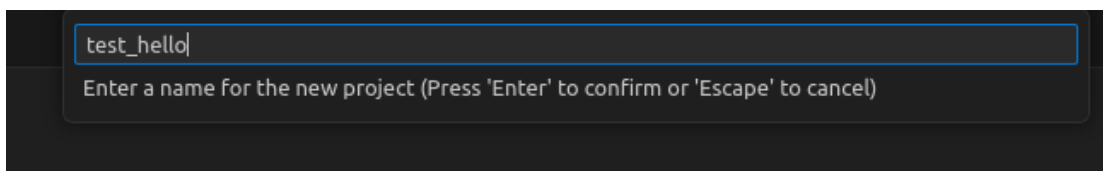


如果这里没有 GCC7.4.1 arm-linux-gnueabi 选项，则说明环境变量设置没有成功，可以选择 Scan for kits 扫描一下，再重试一遍。

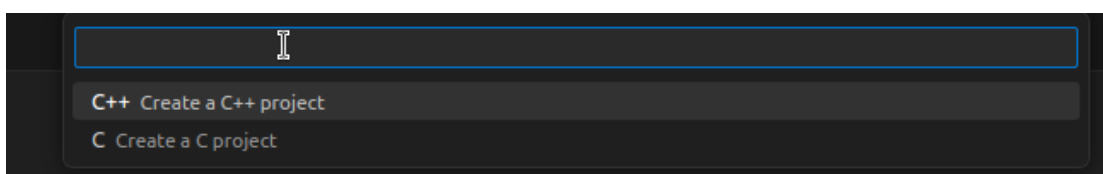
当弹出下面的选项时，鼠标单击选择，在弹出的窗口中，鼠标单击左上角的按钮返回。



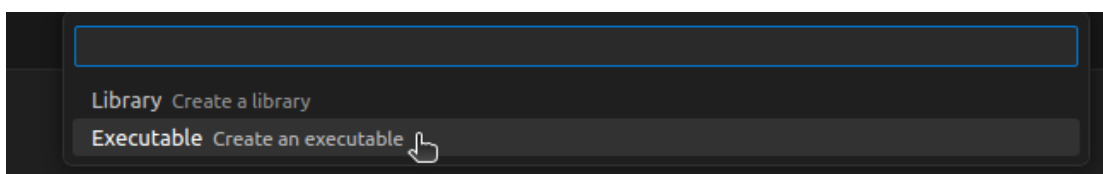
输入工程名: test_hello



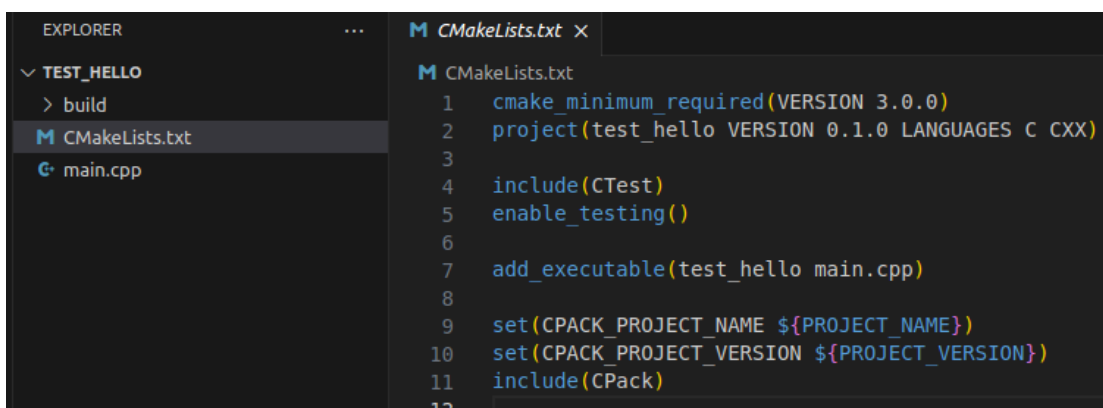
选择创建 C++工程:



选择创建可执行文件:



到此，应用程序 test_hello 的工程创建完成，并自动生成 CMakeLists.txt 文件。



从文件中可以看出，该应用程序工程中，只有一个源文件：main.cpp

4.7 配置 CMake configure Args

将 esm7400_toolchain.cmake 文件复制到应用程序的文件夹下

```

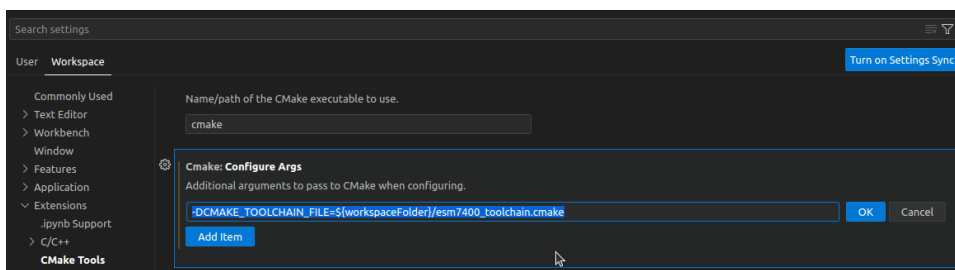
total 16
drwxrwxr-x 5 em em 4096 Jul 27 12:00 build
-rw-rw-r-- 1 em em 260 Jul 27 12:00 CMakeLists.txt
-rw-rw-r-- 1 em em 1872 Jul 26 15:05 esm7400_toolchain.cmake
-rw-rw-r-- 1 em em 92 Jul 27 12:00 main.cpp

```

然后将 esm7400_toolchain.cmake 连接到 CMake configure Args:

通过 vscode 的菜单, File -> Preferences -> Settings, 在弹出的窗口中, 选择 Workspace -> Extensions -> CMake Tools, 找到 CMake: Configure Args, 并点击 Add Item 按钮, 然后输入:

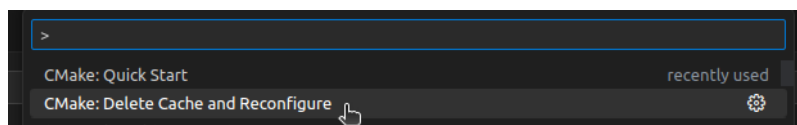
`-DCMAKE_TOOLCHAIN_FILE=${workspaceFolder}/esm7400_toolchain.cmake`



4.8 生成 CMake 编译配置文件

在 vscode 界面, 按组合键: ctrl+shift+p, 在 vscode 上方弹出的选项中, 选择:

CMake:Delete Cache and Reconfigure



如果配置正确, 会生成编译配置文件, 在 vscode 下方的 output 窗口会输出如下的

CMake 的编译配置信息:

```

[main] Configuring project: test_hello
[proc] Executing command: /usr/bin/cmake --no-warn-unused-cli -DCMAKE_TOOLCHAIN_FILE=/home/em/esm7400/test_hello/esm7400_toolchain.cmake -DCMAKE_EXPORT_COMPILE_COMMANDS=BOOL=TRUE -DCM
esm7400/toolchain/gcc-linaro-7.4.1-2019.02-x86_64-arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc -DCMAKE_CXX_COMPILER_FILEPATH=/home/em/esm7400/toolchain/gcc-linaro-7.4.1-2019.02-x86
test_hello -B/home/em/esm7400/test_hello/build -G "Unix Makefiles"
[cmake] Not searching for unused variables given on the command line.
[cmake] -- The C compiler identification is GNU 7.4.1
[cmake] -- The CXX compiler identification is GNU 7.4.1
[cmake] -- Detecting C compiler ABI info
[cmake] -- Detecting C compiler ABI info - done
[cmake] -- Check for working C compiler: /home/em/esm7400/toolchain/gcc-linaro-7.4.1-2019.02-x86_64-arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc - skipped
[cmake] -- Detecting C compile features
[cmake] -- Detecting C compile features - done
[cmake] -- Detecting CXX compiler ABI info
[cmake] -- Detecting CXX compiler ABI info - done
[cmake] -- Check for working CXX compiler: /home/em/esm7400/toolchain/gcc-linaro-7.4.1-2019.02-x86_64-arm-linux-gnueabihf/bin/arm-linux-gnueabihf-g++ - skipped
[cmake] -- Detecting CXX compile features
[cmake] -- Detecting CXX compile features - done
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /home/em/esm7400/test_hello/build

```

说明： 步骤 4.6、4.7、4.8，每一个新建的应用程序工程，都需要进行设置

4.9 编译应用程序

点击 vscode 最下方的 build，即可对程序进行编译：

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[main] Building folder: test_hello
[build] Starting build
[proc] Executing command: /usr/bin/cmake --build /home/em/esm7400/test_hello/build --config Debug --target all -j 10 --
[build] [ 50%] Building CXX object CMakeFiles/test_hello.dir/main.cpp.o
[build] [100%] Linking CXX executable test_hello
[build] [100%] Built target test_hello
[driver] Build completed: 00:00:00.399
[build] Build finished with exit code 0
GCC 7.4.1 arm-linux-gnueabihf Build [all] Run CTest
    
```

编译成功后，会在应用程序工程目录下的 build 目录中，生成运行文件 test_hello

```

em@em: ~/esm7400/test_hello/build$ ls
CMakeCache.txt  cmake_install.cmake  CPackConfig.cmake  CTestTestfile.cmake  Makefile  Testing
CMakeFiles     compile_commands.json  CPackSourceConfig.cmake  DartConfiguration.tcl  test_hello
em@em: ~/esm7400/test_hello/build$
    
```

4.10 工程中新添源文件

如果在应用程序工程中，要增加新的源文件，则需要编辑 CMakeLists.txt 文件，例如要增加文件 mat.cpp，则在 add_executable()内增加相应的文件名即可：

```

cmake_minimum_required(VERSION 3.0.0)
project(test_hello VERSION 0.1.0 LANGUAGES C CXX)

include(CTest)
enable_testing()

add_executable(test_hello main.cpp mat.cpp)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
    
```

4.11 工程中增加线程支持库

在应用程序中，要使用线程，则需要编辑 CMakeLists.txt 文件，增加 pthread 库的链接

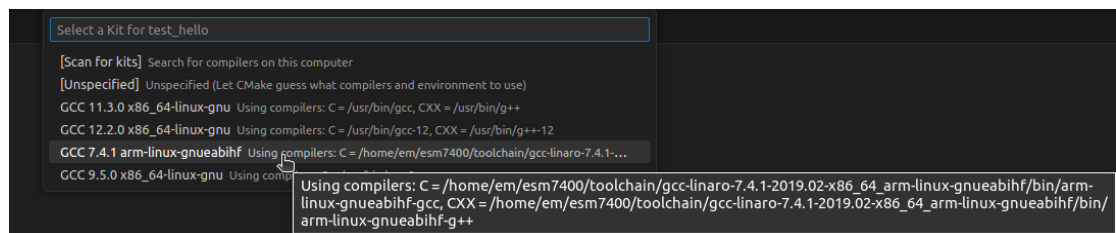
```
target_link_libraries(test_hello pthread)
```

```
set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
target_link_libraries(test_hello pthread)
```

5 关于英创公司提供的例子程序的编译

由于英创公司使用的系统用户名与客户不一致，所以 vscode 生成的 CMake 编译配置参数不适用于客户的环境。因此客户需要按如下步骤操作，才能顺利地对应用程序进行编译：

1. 按第 4 节中的步骤，完成 4.1、4.2、4.3 的操作。(注：4.1、4.2 步骤只需操作一次即可)
2. 进入应用程序目录
3. 删除原来的 build 文件夹
4. 点击 vscode 下面的 build 按钮，这时会在上方弹出下拉列表，在列表中选择 GCC7.4.1 arm-linux-gnueabi，如下图：



之后则会重新配置 vscode 的编译配置文件，并对应用程序进行编译，生成可执行文件。以后就可以直接点击 vscode 下方的 build 按钮进行编译，或在 vscode 的 terminal 窗口中，进入 build 目录，执行 make 进行编译。

附录 1 版本信息管理表

日期	版本	简要说明
2023 年 07 月	V1.0	创建 ESM7400 工控主板使用必读