

1 简介

在这篇应用笔记中，我们以在 i.MX RT1060 EVK 开发板上的工作为例，介绍了对开源机器视觉项目 OpenMV 的移植和适配。在编程模型上，OpenMV 通过与 Micropython 的结合，使用户可以使用 Python 语言来极简地开发机器视觉的应用。读者既可以在这个开发板上使用 Python 快速评估和使用 OpenMV 的功能，也可以在此基础上定制出自己的视觉处理模块，并与系统中其它模块通信。对于更加熟悉 Micropython 和 OpenMV 软件架构的读者，还可以做更进一步的定制，例如添加新的功能，或者剥离 Micropython 系统，在纯 C 环境中开发机器视觉应用。OpenMV 原生的项目管理和构建环境是在 Linux 下基于 GCC 和 Make 的，本文为了方便大多 MCU 嵌入式工程师的开发习惯，把开发环境也移植到了 KEIL MDK5 上。

本文假定读者有使用 KEIL MDK 开发的基本经验，比如了解 CMSIS-Pack，了解 KEIL MDK 工程中的 Target 概念以及切换方法等。

注

如果您对如何在 i.MX RT1050/1060 上使用 Micropython 还不熟悉，可以参考在 *i.MX RT1050/1060 上使用 KEIL 构建 Micropython 系统并体验 Python 开发* (document AN13242)。

i.MX RT1050/60 是一个基于 Arm® Cortex®-M7 核心的处理器，运行速度可达 600 MHz。强大的处理能力、实时特性以及丰富的外围设备，使 i.MX RT1050/60 成为众多高性能应用的理想选择，如工业计算、电机控制、电能转换、智能消费产品、高端音频系统、家庭和楼宇自动化。

2 硬件平台

2.1 i.MX RT1050/60 跨界处理器

NXP 公司提供的 **i.MX RT1050/60** 是基于 Arm Cortex-M7 内核的处理器，可以在高达 600 MHz 的速度下运行。它具有 512 KB 片上 RAM，可灵活配置为核心紧耦合内存 (TCM) 或通用 RAM。此外，还有专门的 512 KB OCRAM。

i.MX RT1050/60 提供：

- 各种接口用于连接各种外部存储器
- 各种串行通信接口，如 USB、以太网、SDIO、CAN、UART、I2C 和 SPI
- 丰富的音频和视频功能，包括 LCD 显示、基本的 2D 图形、摄像头接口、SPDIF 和 I2S 音频接口
- 其他值得注意的功能包括各种模块的安全，电机控制，模拟信号处理和电源管理

2.2 imx_rt105060_evk_board

目录

| | | |
|-----|---|----|
| 1 | 简介..... | 1 |
| 2 | 硬件平台..... | 1 |
| 2.1 | i.MX RT1050/60 跨界处理器..... | 1 |
| 2.2 | imx_rt105060_evk_board..... | 1 |
| 3 | 机器视觉与 OpenMV 项目..... | 2 |
| 3.1 | 机器视觉简介..... | 2 |
| 3.2 | OpenMV 项目简介..... | 2 |
| 3.3 | OpenMV 项目的组成部分..... | 2 |
| 4 | 在自己的 i.MX RT1050/1060 EVK 上构建并运行 OpenMV 固件..... | 3 |
| 4.1 | 下载源代码..... | 3 |
| 4.2 | 打开 KEIL 工程并生成固件..... | 3 |
| 4.3 | 准备文件系统..... | 5 |
| 4.4 | 下载和运行..... | 5 |
| 5 | 在 PC 上访问 Micropython 文件系统..... | 6 |
| 6 | 通过 OpenMV IDE 体验 OpenMV 的功能..... | 7 |
| 7 | OpenMV 机器视觉应用骨架..... | 8 |
| 8 | 图像采集配置..... | 8 |
| 8.1 | 图像色彩..... | 9 |
| 8.2 | 图像分辨率与窗口剪裁..... | 9 |
| 8.3 | 图像亮度控制..... | 9 |
| 9 | Image 模块 - OpenMV 处理机器视觉的“大脑”..... | 10 |
| 9.1 | 绘图..... | 10 |
| 9.2 | 图像处理/滤波..... | 11 |
| 9.3 | 抓拍..... | 11 |
| 9.4 | 录制小视频..... | 12 |
| 9.5 | 人脸与瞳孔检测..... | 12 |
| 9.6 | 特征检测..... | 12 |
| 9.7 | 色彩追踪..... | 12 |
| 9.8 | 条码与 QR 码检测..... | 12 |
| 9.9 | 基于 AprilTag 的方向、位置与姿态检测..... | 12 |
| 10 | 获取更多帮助..... | 12 |
| 11 | 参考资料..... | 12 |
| 12 | 版本历史..... | 13 |



3 机器视觉与 OpenMV 项目

3.1 机器视觉简介

机器视觉是通过光学的装置和非接触的传感器自动地接收和处理一个真实物体的图像，利用预设的算法，分析图像的颜色特征、明暗特征、形状特征、边界特征、长度特征、面积特征等，以获得所需信息或用于控制机器人运动的技术，代替人眼来做测量与决策。传统的机器视觉常常用于工作环境边界清晰，场景内容比较单纯可控的条件下，比如工厂生产线的一角，固定监控的设备等。而最近随着深度学习的兴起，基于深度神经网络的视觉处理系统正在不断拓宽机器视觉的应用领域。

机器视觉技术在工业界的应用由来已久。由于计算量较大，一直以工控电脑为主力运算部件。但是随着近年来 MCU 器件运算能力的持续加强，已经有在 MCU 上部署机器视觉技术的项目。OpenMV 就是其中一个优秀的 MCU 端机器视觉项目。

3.2 OpenMV 项目简介

OpenMV 代表了“Open Machine Vision”，即开放的机器视觉。它既是一个公司名，也是一个项目名。OpenMV 项目致力于创建能在 MCU 上运行的低成本、可扩展、由 Python 驱动的可编程机器视觉模块，旨在成为“机器视觉的 Arduino”，使得机器视觉算法更接近制造者和爱好者。OpenMV 团队已经完成了困难和耗时的算法工作，使用户可以集中精力在自己的应用领域上。

3.3 OpenMV 项目的组成部分

OpenMV 项目由多个软件和硬件资源组成。

- 主要硬件 - OpenMV Cam

这是 OpenMV 功能的硬件载体，已经发展了四代。它的主控芯片是一颗 MCU，从最早的 168 MHz 的 Cortex-M4 升级到了 480 MHz 的 Cortex-M7，软硬件向下兼容。它们的外形非常小巧玲珑，就像是一个大一点的 M12 摄像头模块。我们在移植和适配 OpenMV 的到 i.MX RT1060 EVK 之后，也有三方合作伙伴设计了一款与 OpenMV Cam 第三代硬件在外形上相似的视觉模块。



图 1. 移植了 OpenMV MCU 端软件的 i.MX RT1060 视觉模块

- OpenMV MCU 端软件

在 MIT 开源许可证下开放源代码，构建后会生成在 OpenMV Cam 的主控 MCU 上运行的固件映像。它是 OpenMV 项目软件功能的主要载体，实现在 MCU 上执行机器视觉算法，也能驱动 MCU 来控制外围设备。本文介绍的移植和适

配也是在 OpenMV 的 MCU 端软件上进行的。由于 OpenMV MCU 端软件原先运行的 MCU 平台和 i.MX RT1060 有很大的不同，这部分的工作量是最大的。OpenMV 的 MCU 端软件也是一个定制化的 Micropython，它的基本功能通过 sensor 模块和 image 模块来提供。

— OpenMV IDE

这是 OpenMV 团队为开发者专门打造的集成开发环境，既是一个跨平台的 Python 代码编辑器，又可以通过专门的调试协议，通过 USB 或者 WiFi 与 OpenMV 的固件通信，预览 OpenMV 固件中当前的帧缓冲，并控制 Python 脚本的启动和停止。OpenMV IDE 还自带了大量的样例，并集成了常用的机器视觉小工具。

— OpenMV camera module

这是在 OpenMV Cam 第 4 代中的新设计。在过去的 OpenMV 中，相机芯片是焊接到电路板上的。但从 OpenMV4 开始，相机被制作成单独的可拆卸模块。

— OpenMV expansion boards

通过 OpenMV Cam 有两排扩展口连接，类型比较丰富，如 LCD 板，伺服板，云台板，电视板等。

4 在自己的 i.MX RT1050/1060 EVK 上构建并运行 OpenMV 固件

4.1 下载源代码

4.1.1 下载为本文特制的版本

从 [source code](#) 下载 source code (zip)，并解压缩。请牢记使用 `an_mpy1050_rev1` 标签，这是一个与本文配套的版本，下文的介绍也是基于这个版本。

4.1.2 以 git clone 方式下载

如果您熟悉 git 的使用，可以打开一个命令窗口，导航到自己打算放入的目录中，然后执行以下命令：

```
git clone https://github.com/RockySong/micropython-rocky.git
```

这会默认克隆“omv_initial_integrate”分支。为了保证和本文所叙述的内容完全相同，建议 check out 到“an_mpy_rt1050_60”分支。

注

如果使用最新版本的代码遇到问题，最好切换到带 `an_mpy1050_rev1` 标记 (tag) 的版本。

4.2 打开 KEIL 工程并生成固件

基于 `an_mpy1050_rev1` 版本，找到 `ports\prj_keil_rt1060\mpyrt1060.uvprojx` 并且使用 KEIL 5.0 或更高版本打开它。

注

这个工程依赖 NXP.MIMXRT1062_DFP.12.1.0 或更高版本，这是 CMSIS pack 包。

这个工程里有多目标配置 (KEIL 称它们为“Targets”)，如 [图 2](#) 所列。

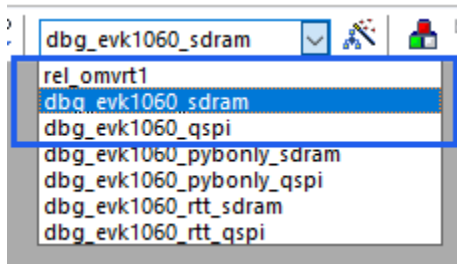



图 2. Multiple targets

这里前三个目标用于生成带有 OpenMV 的映像。

| | |
|----------------------------------|--|
| rel_omvrt1 | 为在 i.MX RT1060 视觉模块构建固件。 注 不能在 RT1060 EVK 上使用此目标构建固件！ |
| dbg_evk1060_sdram (调试验验之首选) | 为 i.MX RT1060 EVK 构建可以在 SDRAM 中运行和调试固件。下载快，不需要烧写 Flash，但断电或复位后需要重新下载。 |
| dbg_evk1060_qspi | 为 i.MX RT1060 EVK 构建可以在从 QSPI Flash 中运行和调试固件，适合于演示目的，断电和复位后无需重新下载，但烧写 Flash 比较久。 |


这三个构建目标也同样会使用 i.MX RT1060 的片上 ITCM，DTCM 和 OCRAM。合理地把性能关键的代码和数据放入它们之中，对提高机器视觉算法性能有至关重要的作用。

选择有 **dbg_evk1060_sdram** 关键字的 target，因为在 SDRAM 中调试最方便，建议一般先选择在 SDRAM 中调试。无论选哪一种，都可以点击  或按下 **F7** 来生成整个工程。

如果出现以下的报错：

```
Build Output
*** Target 'dbg_evk1060_pybonly_sdram' uses ARM-Compiler 'V5.06 update 6 (build 750)' which is not available.
*** Please review the installed ARM Compiler Versions:
'Manage Project Items - Folders/Extensions' to manage ARM Compiler Versions.
'Options for Target - Target' to select an ARM Compiler Version for the target.
*** Build aborted.
```

图 3. 报错信息

则表示没有指定的 AC5 编译器的版本，可以点击 

或按 **Alt+F7** 来呼出 **Options for Target** 对话框。切换到 **Target** 页面，在 **Code Generation** 框的 **ARM Compiler** 下拉菜单中选择 **Use default compiler 5**，确定后再重新编译。

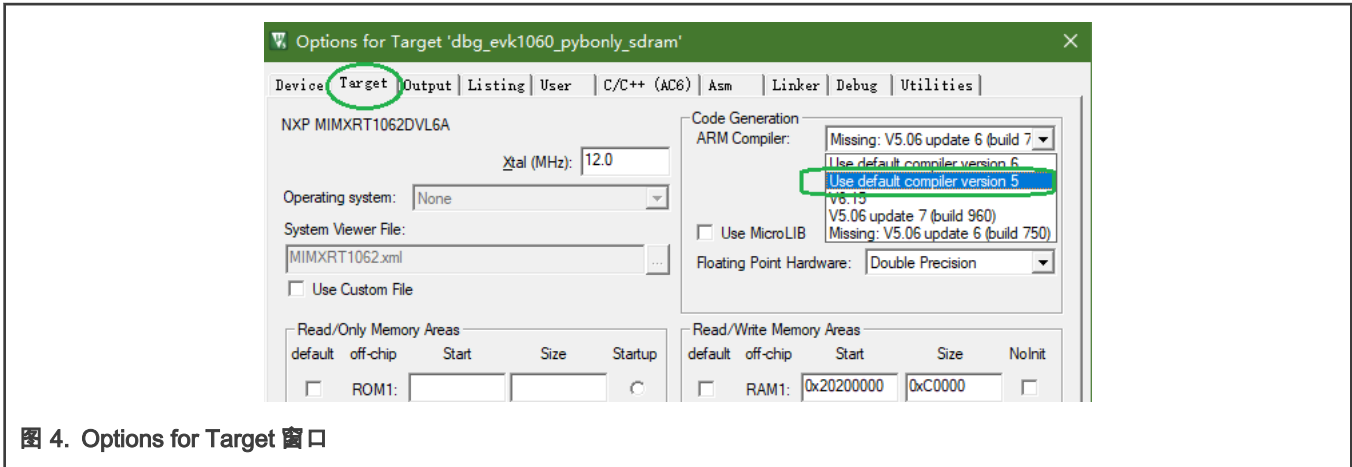


图 4. Options for Target 窗口

4.3 准备文件系统

OpenMV MCU 端固件的功能是基于在 Micropython 系统之上的，而在 Micropython 中，文件系统的概念极为重要，所有的 Python 脚本都是存储在文件系统上的。在我们移植的 OpenMV 系统中，支持 TF/microSD 卡上的文件系统和程序 Flash 中的文件系统。在启动期间，系统先检测是否插入了 TF 卡：

- 如果有就挂载 TF 卡到根文件系统下。
- 如果没有，则挂载从 QSPI Flash 中分配的一块约 2 MB 的空间作为文件系统。如果是首次使用，则系统会自动格式化 QSPI Flash 中的这块区域。

Flash 文件系统使得在缺失 TF 卡的时候仍然可以运行 Micropython。然而，它的写入性能极差（约 10 KB/s 左右），并且没有包含磨损平衡，一般建议只放一些不常改动的文件，比如配置文件、已调试好的脚本等。除此之外，在往 Flash 文件系统写入数据时，会关闭中断，影响实时性。因此，**强烈建议在板上插入一个使用 FAT/FAT32 格式化过的 TF 卡。**

4.4 下载和运行

按下列步骤进行下载：

1. 首先确定自己使用的调试器。i.MX RT1060 EVK 上自带了 CMSIS-DAP，但是由于本工程生成的固件比较大，下载慢一些。也可以使用 J-Link。在 i.MX RT1060 EVK 上，若使用 J-Link，需要断开 J47 和 J48；若使用板载 CMSIS-DAP 兼容调试器则短接它们。在 KEIL 下，J-Link 的程序下载远比板载的 CMSIS-DAP 要快。Micropython 固件在包含 OpenMV 后增大了数倍，因此强烈建议使用 J-Link 来下载。

注

如果使用 J-Link，那么在下载到 SDRAM 中后，在执行程序之前，最好再点击一下复位按钮，否则可能会意外进入 hard fault。

2. 按图 5 所示连接 i.MX RT1060 EVK 开发板。

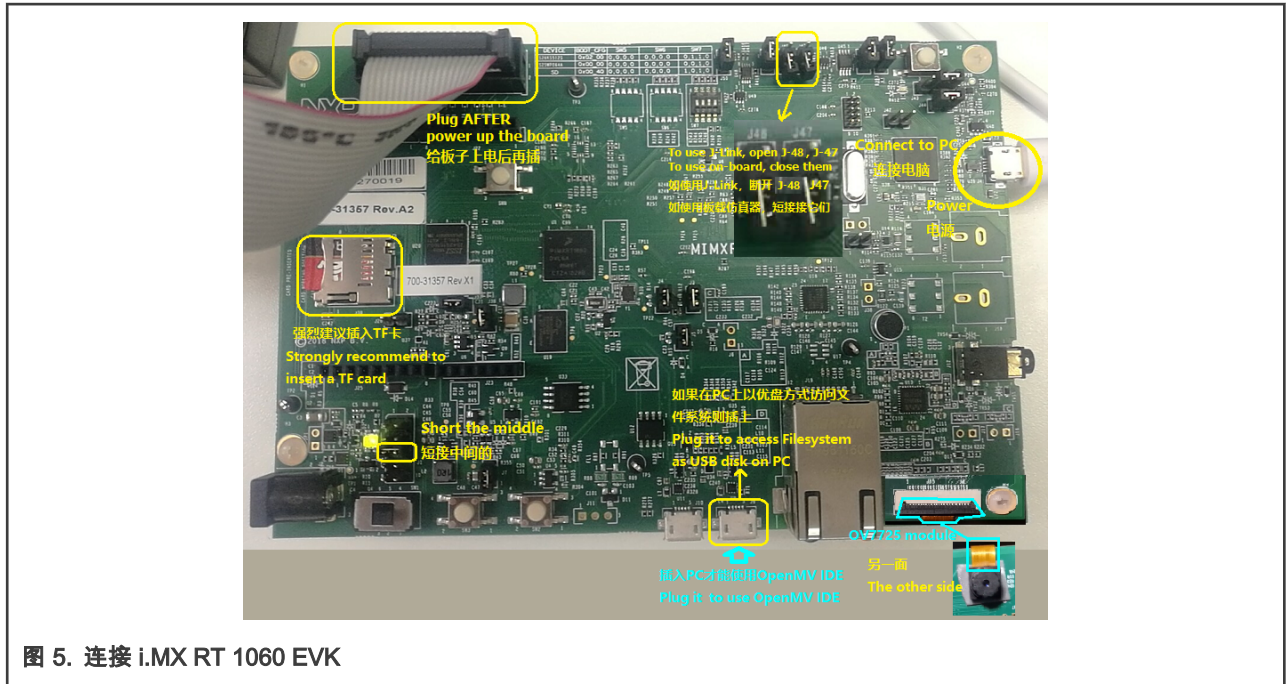


图 5. 连接 i.MX RT 1060 EVK



注

右下角连接的一个 OV7725 模块，它是视频信号的输入源。如果您手上还有为 RT1060evk 配套的 LCD 模块，也可以一并连接，这样就可以在 LCD 模块上预览图像，以及在 OpenMV 处理之后叠加的标注。

还需要注意靠近以太网的 USB 接口。OpenMV IDE 通过此接口与开发板通信。

3. 打开一个串口终端，连接开发板的板载调试器所呈现的虚拟串口，波特率设置为 115200。

按下列步骤来运行 i.MX1060 EVK：

- 选择在 SDRAM 中执行：
 1. 在生成完毕后点击  或按 **Ctrl-F5**。
 2. 等待程序下载完毕并进入调试界面。
 3. 按 **F5** 全速运行。
- 选择在 Flash 中(XIP)执行：
 1. 点击  或按 **F8** 把生成的固件烧写到开发板的 QSPI Flash 中。
- 选择在 Flash 中执行：
 1. 无需进入 KEIL 的调试会话，直接复位开发板即可运行。

5 在 PC 上访问 Micropython 文件系统

如果希望在 PC 上访问文件系统的内容，请在靠近以太网口的 USB 口上连接到电脑，这会在电脑上呈现出一个可移动磁盘。

- 当没有插入 TF 卡时，可移动磁盘中显示的是 Flash 文件系统的内容。
- 当插入了 TF 卡时，可移动磁盘里显示的是 TF 卡中的内容。

当访问的是 Flash 文件系统时，写入速度只有 10 KB/s 左右。当访问的是 TF 卡时，一般读写速度均在 10 MB/s 左右。OpenMV 有录制 MJPEG 视频和存储图片到文件系统中的功能，在使用这些功能时，请务必插入 TF 卡。

注

不要在 TF 卡中包含名为“flash”（区分大小写）的目录，这会导致在 Python 脚本访问“/flash”时仍然访问 Flash 文件系统，而在 PC 上看到的却是 TF 卡中的内容。

6 通过 OpenMV IDE 体验 OpenMV 的功能

OpenMV 的运行环境是 Micropython，所以自然可以使用 Micropython 的开发工具来开发 OpenMV 应用。不过，面向机器视觉应用有一些特殊需求，例如，实时预览系统采集的和标注后的图像，分析图像的基本统计特征、实时控制脚本的启停、集成机器视觉常用工具等。这部分既有低速率的控制流，也有高速率的图像数据流。

为了满足这个新增的需求，OpenMV 项目在提供核心的机器视觉功能之外，还新开发了一整套开发与调试机制，可以让 PC 通过 USB 虚拟串口来监视与控制 OpenMV 固件的运行状态。并且还开发了一个配套的 IDE 环境，名为 **OpenMV IDE**。我们在适配 OpenMV MCU 端软件到 i.MX RT1060 平台时保留了与 OpenMV IDE 通信的部分，所以仍然可以使用 OpenMV IDE 来监控 OpenMV MCU 端软件在 i.MX RT1060 上的运行状态。

OpenMV IDE 可以从 <https://openmv.io/pages/download> 上下载，基于 Qt5，支持 Windows, OSX, Ubuntu 和树莓派平台。图 6 是 OpenMV IDE 的界面截图，包含代码编辑窗口、MCU 中帧缓冲当前内容监视窗口、串口终端窗口，以及图像直方图分析窗口，如图 6 所示。

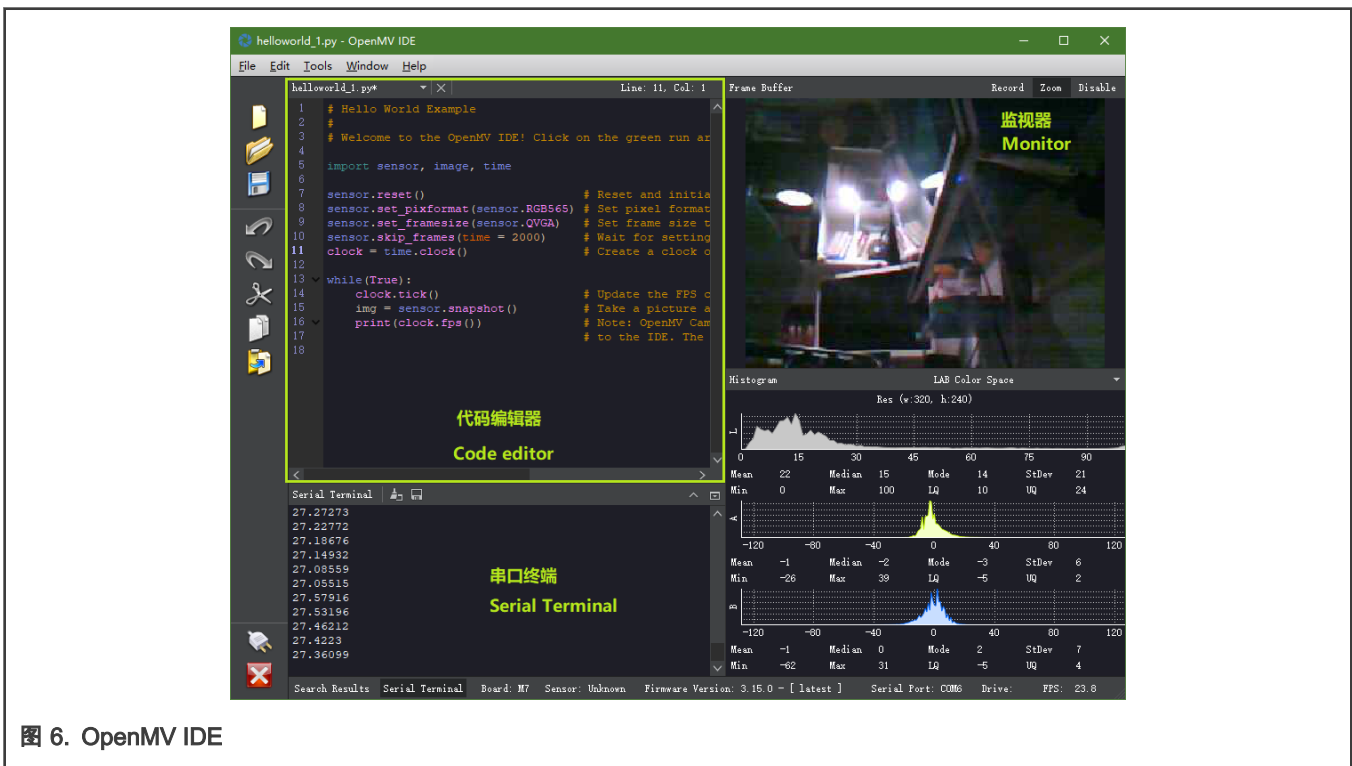


图 6. OpenMV IDE

注

在下方状态栏的右边实时刷新帧缓冲监视窗口的刷新率，常常小于图像处理主循环的循环频率。并且由于图像压缩与上传也占用了一部分 MCU 端系统资源，如果 Disable 图像监视器窗口，图像处理频率还会上升。

有关 OpenMV IDE 的更详细文档，参见 [OpenMV IDE Overview](#)。

为了与 OpenMV IDE 通信，在 OpenMV MCU 端软件上也有一个专用的模块为 usbdbg。

在我们移植的 OpenMV MCU 端软件中，usbdbg 与 REPL 复用同一个 USB 虚拟串口，使用前需要把 i.MX RT1060 EVK 上靠近以太网口的 USB 接口(J9)通过 USB 电缆连接到电脑上。在与 OpenMV IDE 建立连接期间，无法使用 REPL。

7 OpenMV 机器视觉应用骨架

按下列步骤运行机器视觉的算法：

1. 准备好相机，然后循环取到图像数据。

以实现这个“空转”功能的小程序为例子，这也可以当作是 OpenMV 机器视觉中的“Hello world”程序。

- a. 导入需要的模块。

```
import sensor, image, time
```

sensor 和 **image** 是 OpenMV 中最重要的两个模块。sensor 模块封装了相机操作，image 模块封装了机器视觉的运算。time 模块可用于计量帧率。

- b. 准备好相机。

```
sensor.reset() # Reset and initialize the sensor.
```

sensor.reset() 方法用于复位并初始化相机的控制芯片，例如 OV7725。根据不同的感光芯片，复位时间也有所不同。

- c. 相机初始化后，给出所需的分辨率、彩色/灰度模式设置。在这个简单的示例里，使用 QVGA @ RGB565，这也是 OpenMV 中多数示例使用的设置。

```
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565
sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
```

- d. 上述的 **set_pixFormat()** 方法可用于设置像素的格式，常用的彩色格式是 RGB565。如果需要灰度图，可以把 **sensor.RGB565** 改为 **sensor.GRAYSCALE**。

- e. 感光芯片在工作后，常常需要 1-2 秒钟来动态适应光照环境，调整曝光时间、白平衡等参数。在 OpenMV 中，可以选择性地跳过这一过渡过程中图像。

```
sensor.skip_frames(time = 2000) # Wait for settings take effect.
```

- f. 上述的 **skip_frames()** 方法可用于跳过一段时间的图像，示例中设置为比较长的 2000 毫秒。

通过以上的操作，相机已经正常工作了。

2. 进入主工作循环。

假设这个系统是一直工作的，所以使用一个无限循环。每循环一次，先拍摄一张照片，然后在这个照片上做真正所需的处理。

```
while(True):
    clock.tick() # Update the FPS clock.
    img = sensor.snapshot() # Take a picture and return the image.
    <do further image processing>
    print(clock.fps()) # Note: OpenMV Cam runs slower when connected to the IDE. The FPS should increase once disconnected.
```

上述的 **sensor.snapshot()** 方法可用于拍摄并返回一张照片。

以上就是一个通过 OpenMV 实现的机器视觉应用的“Hello world”，可以看作是添加后续机器视觉逻辑的起点。

从上面的主循环还可以发现，在 OpenMV 中，图像是一张一张独立处理的。

8 图像采集配置

图像采集是机器视觉系统的眼睛，高质量的采集效果为机器视觉算法结果的正确性提供有力的保证。开发者需要根据机器视觉任务的需要，合理设置图像的分辨率、色彩模式、颜色和亮度控制等参数。

8.1 图像色彩

对于图像的采集，OpenMV 支持 `sensor.RGB565` 作为 RGB565 的彩色度与 `sensor.GRAYSCALE` 作为灰度。

当需要使用二值图时，也可以使用 `image` 对象的“`binary()`”方法转换成二值图。

8.2 图像分辨率与窗口剪裁

虽然在电脑上大家已经习惯了 1920×1080 甚至是 3840×2160 (4K) 的极高分辨率，但是在 OpenMV 机器视觉应用中，最常用的还是 320×240 (QVGA) 甚至是更低的。表 1 给出了常见的一些图像分辨率。

表 1. 图像分辨率

| 分辨率 | 简称 |
|-----------|---------------------------|
| 128 × 128 | B128 × 128 |
| 160 × 120 | QQVGA |
| 320 × 240 | QVGA |
| 352 × 288 | CIF |
| 480 × 272 | i.MX RT1060 EVK 的 LCD 分辨率 |
| 640 × 480 | VGA |

通过 `sensor.set_windowing()` 方法，可以从大分辨率的输入上剪裁出更小的窗口。例如， 480×272 不是 OpenMV 支持的原生分辨率，但是可以从 VGA 分辨率下剪裁出来。

在我们移植的相机驱动中，剪裁窗口是在收图的同时进行的，并且不会让原图额外占用一份帧缓冲的空间。`sensor.set_windowing()` 既支持由 4 元组给出的目标矩形位置的大小 (左,上,长,宽)，也支持由 2 元组只给出目标矩形大小并且裁剪出和原图“同中心”的矩形窗口。例如，我们可以如下从原图中心剪裁出一个能充满 i.MX RT1060 EVK 的 LCD 屏幕的窗口。

```
sensor.set_framesize(sensor.VGA)
sensor.set_windowing((480,272))
```

注

(480,272)外面的括号是必需的，表示以 tuple 类型传递参数，而 `set_windowing()` 方法根据 tuple 中的元素是 2 个还是 4 个来判断是否只提供矩形大小。

8.3 图像亮度控制

图像的亮度对机器视觉算法的质量有重大的影响，而相机的曝光时间和环境光照的配合非常影响成像的亮度，感光芯片也一般提供多种可编程的亮度直接或间接控制方式。在 OpenMV 中，通过 Python API 绑定提供了多种能直接/间接控制亮度的方法，这里简单介绍一下。

- 直接调节亮度

通过 `sensor.set_brightness()` 方法，可以直接控制感光元件的亮度处理逻辑。可设置的范围是 -3 至 +3 的整数。

- 增加曝光时间

通过 `sensor.set_auto_exposure(False, 整数)` 可以手动控制曝光时间，范围在 0-10000。曝光时间越长亮度越高。

- 提高设置增益上限

在 (默认的) 自动增益下，通过 `sensor.set_gainceiling(2/4/8/16/32/64/128)` 来增加增益上限，每次提高一倍，最高 128 倍。增益越高，图像越亮。

- 手动指定增益

通过 `sensor.set_auto_gain (False, 2/4/8/16/32/64/128)` 来手动指定增益。

- 降低感光芯片主时钟的频率

通过 `sensor.set_framerate()` 方法，可以调节感光芯片主时钟频率，调低常常能起到增加曝光时间提高亮度的效果。在我们移植的系统中，可以由用户选择时钟源和分频因子，编码如下：

```
{时钟源代码}<<9 | {分频因子+1}<<11
```

其中，时钟源代码可以是

- 0 : 24 MHz
- 2 : 120 MHz

分频因子可以是 0-7，分别对应 1-8 分频。

在 i.MX RT1060 EVK 上使用的感光芯片是 OV7725，**最大输入时钟频率为 15 MHz**，在内部被 4 倍频。通过把时钟源改为 0 (24 MHz)，并使分频因子大于 1，可降低 OV7725 的输入时钟，增加亮度。

注

注意：使用此法时，不要使输入时钟大于 15 MHz！

- 使用 VGA 作为原始分辨率并剪切图像

OV7725 在以 VGA 分辨率下采集图像时亮度较高，不过高分辨率但是对资源的开销也扩大了。我们可以剪裁窗口来解决，通过以下命令来模拟 `sensor.set_framesize(sensor.QVGA)` 的效果。只是这样成像区域也缩小了一半。

```
sensor.set_framesize(sensor.VGA)
sensor.set_windowing((320,240))
```

上述方法的效果可以叠加，使图像成像亮度有非常大的变化范围。图 7 显示了同一暗室内的成像效果（光源是 3W 白光 LED 带状灯）。

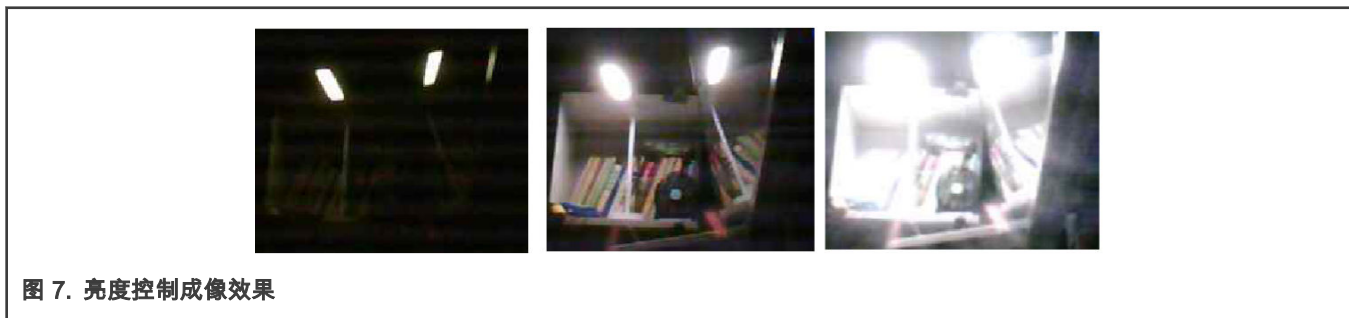


图 7. 亮度控制成像效果

还可以看出，当亮度过高或过低时会影响成像的色彩。

9 Image 模块 - OpenMV 处理机器视觉的“大脑”

OpenMV MCU 端软件包含了丰富的机器视觉算法的实现。这些算法封装到了 image 模块中，负责呈现各项 OpenMV 的机器视觉功能，它们的种类非常的丰富，博大精深。如果说前面介绍的 sensor 模块是 OpenMV 的“眼睛”，那么 image 模块就是 OpenMV 的“大脑”。为了演示这个大脑的功能，OpenMV IDE 中自带了众多的样例程序。接下来，我们结合示例，展示一下 image 模块中比较常用的机器视觉功能。

9.1 绘图

绘制箭头、线条、十字、圆形、椭圆、关键点，以及输出字符串。这些功能与机器视觉没有关系，但是可以作为处理结果的标注和制作简单的人机界面。

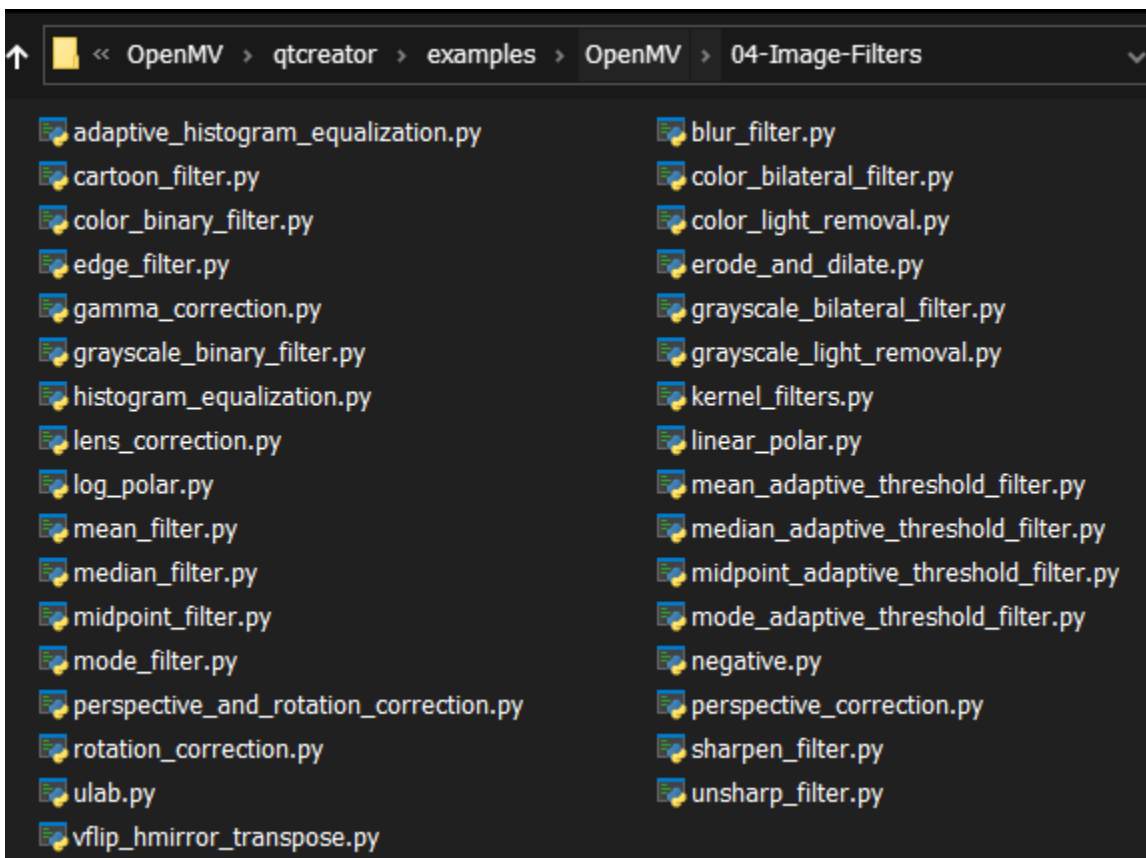


9.2 图像处理/滤波

这部分包含了非常丰富的图像处理算法，包括：自适应直方图，模糊，cartoon，双边滤波，二值化滤镜，光线移除，边缘检测、腐蚀与膨胀、伽玛校正、摄像机标定、线性核函数和形态滤波-极坐标、对数-极坐标、平均自适应阈值滤波、自适应阈值平均滤波和中值滤波、中值滤波、中点自适应阈值滤波、中点滤波，否定，透视和旋转校正，锐化，翻转和反射等。

用于演示图像处理/滤波的样例位于\OpenMV IDE\share\QtCreator\Examples\OpenMV\04-image-filters。

可以在 OpenMV IDE 里通过 File -> Examples -> OpenMV -> Image-Filters 下的菜单项一次打开一个，也可以通过文件资源管理器来浏览。



9.3 抓拍

OpenMV 支持条件触发和时间触发的抓拍方式。具体示例位于 .\OpenMV IDE\share\qtcreator\examples\OpenMV\05-Snapshot.

9.4 录制小视频

OpenMV 可以录制 gif 和 mjpeg 格式的小视频，附带的示例还演示了由人脸检测事件和运行检测事件来触发视频录制。具体示例位于 [.|OpenMV IDE|Share|QtCreator|Examples|OpenMV106-video-recording](#)。

9.5 人脸与瞳孔检测

OpenMV 包含了使用经典机器学习 HAAR-Cascade 方法训练的人脸和瞳孔检测模型。具体示例位于 [.|OpenMV IDE|Share|QtCreator|Examples|OpenMV107-Face-Detection](#) 和 [.|OpenMV IDE|Share|QtCreator|Examples|OpenMV108-Eye-Tracking](#)。

9.6 特征检测

OpenMV 包含了多种图像特征检测功能，包括边缘检测、圆形检测、线段检测、矩形检测、方向梯度直方图 (HOG)、关键点、线性回归、局部二值模式 (LBP)。

具体示例位于 [.|OpenMV IDE|Share|QtCreator|Examples|OpenMV109-Feature-Detection](#)。

9.7 色彩追踪

OpenMV 包含了多种色块和线条追踪功能，既包含灰度块追踪也包含彩色色块追踪，并且可同时追踪多种色彩。此外，还包含了沿着黑色引导线循迹的示例。

具体示例位于 [.|OpenMV IDE|Share|QtCreator|Examples|OpenMV110-Color-Tracking](#)。

9.8 条码与 QR 码检测

OpenMV 移植了“zbar”库，可以使用 Python 接口检测图像视野中的单个与多个条码与 QR 码，并且可以在检测之前先行校正镜头的畸变。

具体示例位于 [.|OpenMV IDE|Share|QtCreator|Examples|OpenMV116-Codes](#)。

9.9 基于 AprilTag 的方向、位置与姿态检测

OpenMV 支持多个家族的 AprilTag 检测，包括码名、距离信息、方位角信息，并且支持在同一个视野中检测多个 AprilTag。

OpenMV IDE 还集成了生成 AprilTag 的功能。

OpenMV 支持 AprilTag 家族有：16H5, 25H7, 25H9, 36H10, 36H11。

具体示例位于 [.|OpenMV IDE|Share|QtCreator|Examples|OpenMV126-April-Tags](#)。

10 获取更多帮助

OpenMV 官方网站有非常完备的文档系统，位于 [MicroPython documentation](#)。

这里不但包含 OpenMV 有关的文档，也包括对 Micropython 语言本身、库、以及一些开发板的功能的介绍，可做参考。

使用中文的读者还可以前往 [Singtown](#) 参考星瞳科技翻译的 OpenMV 文档和制作的大量教学视频。

11 参考资料

下列资料可提供更多信息：

- [MicroPython documentation](#)
- [MicroPython](#)
- [micropython-rocky](#)
- [OpenMV](#)
- [Singtown](#)

12 版本历史

| 版本历史 | 日期 | 重大更新 |
|------|--------------|------|
| 0 | 10 May, 2021 | 初始发布 |

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2021.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 10 May, 2021

Document identifier: AN13243

