基于RISC-V的CH32V307及D1芯片构件GEC生态系统

浙江海洋大学信息工程学院 陈宏铭教授
苏州大学计算机科学与技术学院 王宜怀教授

2022年6月28日

目 录

RISC-V的来龙去脉

名词梳理

- **复杂指令集计算机**（Complex Instruction Set Computer，CISC），其指令系统复杂、庞大，一般在200条以上；寻找方式较多，一般大于4；代表着复杂的CPU设计，典型代表为Intel公司X86。
- **精简指令集计算机**（Reduced Instruction Set Computer，RISC），其指令系统简单、精简，一般在100条以下；寻找方式较少，一般小于4；代表着简约的CPU设计，典型代表为Apple公司的Macintosh、ARM、RISC-V。

为什么会出现RISC?

- 1979年，美国加州大学伯克莱分校的帕特逊教授团队分析CISC存在的主要缺点：
  - CISC中各种指令的使用率相差悬殊：一个典型程序的运算过程所使用的80%指令，只占一个处理器指令系统的20%；
  - 复杂的指令系统带来结构的复杂性，增加了设计的时间与成本，也容易造成设计失误；
  - 很难把CISC的全部硬件做在一个芯片上，影响了单片计算机的发展。
- 为了克服这些缺点，帕特逊等人提出了精简指令的设想，即指令系统应当只包含那些使用频率很高的少量指令，并提供一些必要的指令以支持操作系统和高级语言，按照这个原则发展而成的计算机被称为精简指令集计算机RISC。

RISC-V的由来

1980年，美国加州大学伯克莱分校的帕特逊教授团队设计了RISC-I。2010年，设计了RISC-V指令集架构之前，该团队已经具有四代RISC指令集架构设计经验，正因为有相关的技术沉淀，该团队才能在短期内设计出了RISC-V。

RISC-V的简明描述

- RISC-V是2010年美国加州大学伯克利分校基于RISC原则而设计一个开源指令集架构（Instruction Set Architecture，ISA）。它不是一种处理器或芯片，而是一套指令集架构规范（Specification）。
- 其中的“V”有两层含义：
  - 发明者美国加州大学伯克利分校团队着眼于RISC开始而设计的第五代指令集架构；
  - 代表了变化（Variation）和向量（Vectors）。RISC-V三个重要特点：开源免费、设计成本低，可设计出功耗低体积小的芯片。

RISC生态建设的开端

- 帕特逊团队属于高校，在设计RISC-V指令集之后，研究团队决定将它彻底开放，使用BSD License开源协议。任何人都可以基于RISC-V指令集进行芯片设计和开发，而不需要支付授权费用，由此，大批公司开始加入对RISC-V的研究和二次开发之中。
- 从2011年开始，SiFive公司的创始人Andrew Waterman、Yunsup Lee等设计出两款RISC-V芯片并成功流片，验证了RISC-V架构的可行性。随着这两款芯片提供给全世界开发者，成为RISC-V生态建设的开端。
- SiFive是2015年Andrew Waterman等创立的一家专门推动RISC-V商业化的公司，是SaaS（Software-as-a-Service）服务商，提供基于RISC-V指令架构的商用化处理器IP，开发工具和芯片解决方案。

RISC-V基金会

- 2015年，为了确保RISC-V成功运营，非盈利性组织RISC-V基金会（RISC-V Foundation）创立。
- 从开始成立时的29个会员，发展到2021年12月在 70 多个国家/地区拥有2000 多名成员。
- 短短几年的时间里，包括谷歌、华为、IBM、镁光、英伟达、高通、三星、西部数据等商业公司，以及加州大学伯克利分校、麻省理工学院、普林斯顿大学、ETH Zurich、印度理工学院、洛伦兹国家实验室、新加坡南洋理工大学以及中科院计算所等学术机构，都纷纷加入RISC-V基金会。

RISC-V在中国

- 2018年8月，SiFive将业务扩展到中国，注册赛昉科技（SaiFan）为国内客户提供IP授权服务。
- 2018年10月，中国RISC-V产业联盟（CRVIC）在上海成立，同年11月，中国开放指令生态联盟（CRVA）宣布成立。
- 随后的一年，华米黄山一号、阿里巴巴玄铁910，沁恒青稞V3，青稞V4等基于RISC-V开发的处理器相继发布，中国科技公司在对RISC-V的开发中渐渐找到各自的方式。

RISC-V的未来发展场景

- 随着中美贸易战的发生，以及对潜在贸易限制的担忧，总部位于美国的芯片技术非营利组织RISC-V基金会总部已于2020年3月已经正式迁移至瑞士，其法律实体过渡至瑞士，并将从美国迁出。
- 官方表示这是为了防止任何国家、公司或个人对RISC-V进行任何限制或禁运，这也将成为RISC-V的另一个竞争优势。

基于RISC-V的构件通用嵌入式计算机

智能终端开发方式存在的问题

- 微控制器（Microcontroller Unit，MCU）或应用处理器（Application Processor，AP）是智能终端（Ultimate-Equipment，UE）的核心，承担着传感器采样、滤波处理、边缘计算、融合计算、嵌入式人工智能算法、通信、控制执行机构等功能。
- 芯片生产厂家往往配备一本厚厚的参考手册，少则几百页，多则可达近千页。许多厂家也给出庞大软件开发包（Software Development Kit，SDK），但设计人员需要花许多精力从中析出个体需要。
- 智能终端开发人员通常花费太多的精力在基于芯片级硬件设计及基于寄存器级的底层驱动设计上，开发方式存在软硬件开发颗粒度低、可移植性弱等问题。

解决办法 —— 三个提高

- **提高硬件设计颗粒度**
  - 若能将MCU硬件最小系统及面向领域应用的共性电路封装成一个整体，则可提高UE的硬件设计颗粒度。硬件设计也可从元件级过度到硬件构件级为主的设计场景。
- **提高软件编程颗粒度**
  - 若能设计出基于模块共性知识要素的底层驱动构件，屏蔽寄存器级编程之差异，则可把软件编程颗粒度从寄存器级提高到知识要素级。
- **提高软硬件可移植性**
  - 若能从共性知识要素角度研究可移植性问题，屏蔽芯片生产厂家、编译器类型等因素，遵循软硬件构件的设计原则，则可提高软硬件的可移植性。

通用嵌入式计算机GEC的定义

- 一个具有特定功能的通用嵌入式计算机（General Embedded Computer，GEC），体现在硬件与软件两面。
  - 在硬件上，把MCU硬件最小系统及面向具体应用的共性电路封装成一个整体，为用户提供SoC级芯片的可重用的硬件实体，并按照硬件构件要求进行原理图绘制、文档撰写及硬件测试用例设计。
  - 在软件上，把嵌入式软件分为BIOS程序与User程序两部分。BIOS程序先于User程序固化于MCU内的非易失存储器（如Flash）中，启动时，BIOS程序先运行，随后转向User程序。BIOS提供工作时钟及面向知识要素的底层驱动构件，并为User程序提供函数原型级调用接口。
- 与MCU对比，GEC具有**硬件直接可测性、用户软件编程快捷性与可移植性**三个基本基本特点。

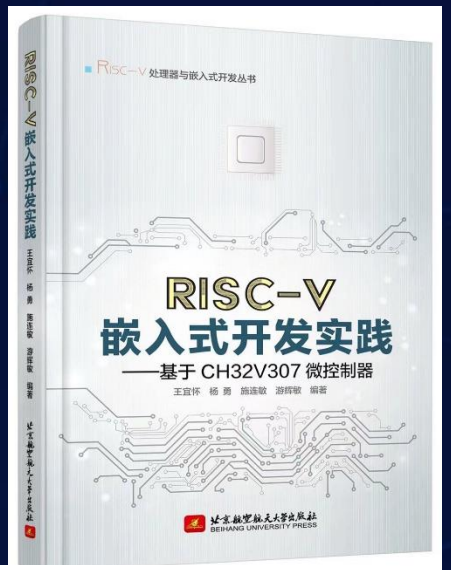
什么是GEC的生态系统

- 普适意义上的生态系统是指可以持续发展一系列要素集合。
- GEC生态系统是指GEC的硬件、软件开发工具、底层驱动构件、操作系统、测试样例、应用样例、文档等集合，可以实现GEC快速规范的应用开发。
- 目前基于RISC-V的GEC基本生态系统包括：
  （1）硬件开发板：AHL-XXX
  （2）AHL-GEC-IDE
  （3）RT-Thread实时操作系统的驻留
  （4）标准工程框架
  （5）基础构件
  （6）软件及文档电子资源
  （7）配套书籍
  （8）应用案例

基于RISC-V的CH32V307构件GEC生态系统

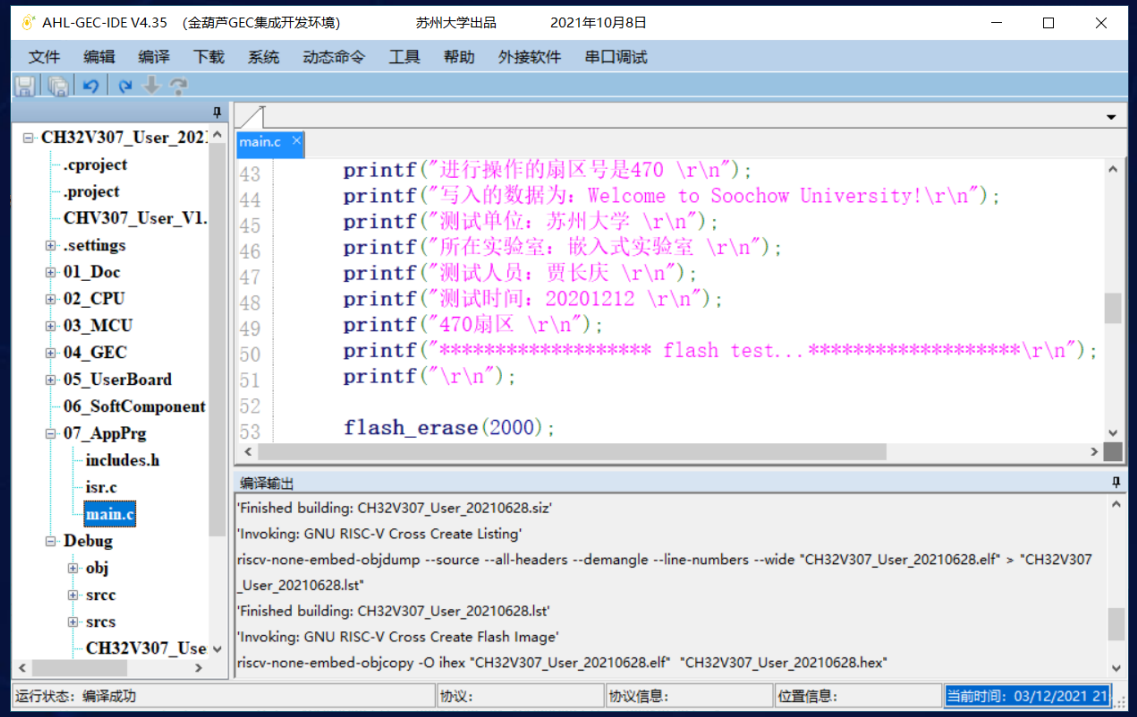
硬件开发板：AHL-CH32V307

配套书籍：RISC-V嵌入式开发实践



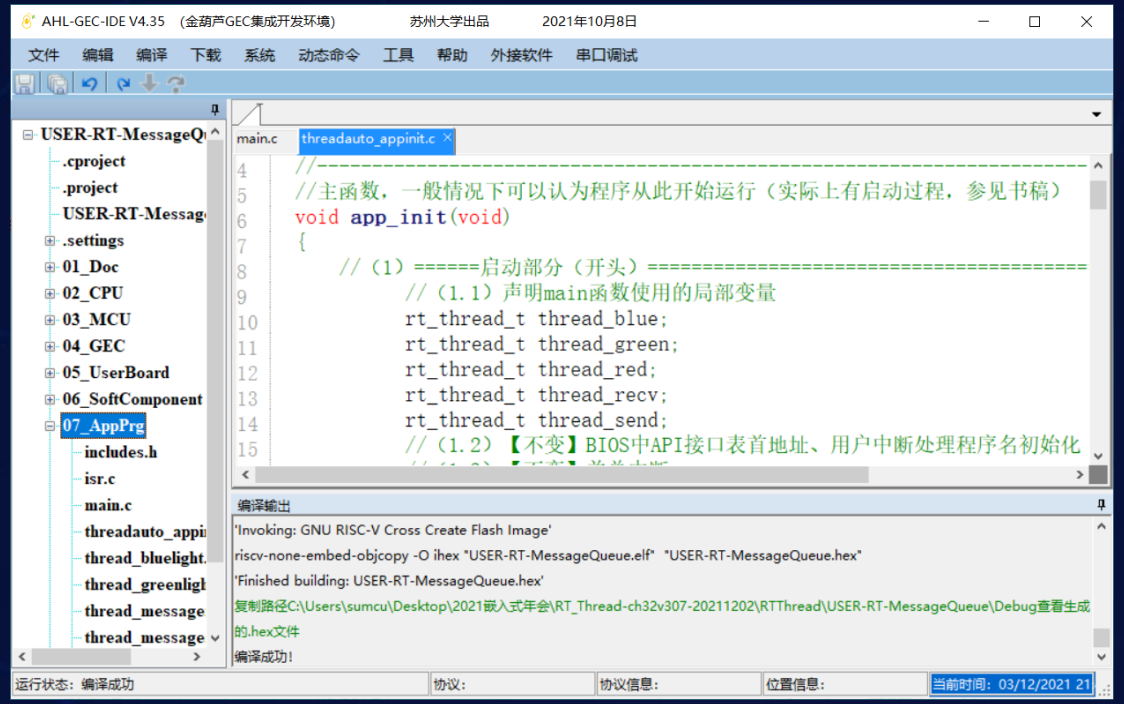
序号	名称	描述
1	芯片引脚	南京沁恒 2021 年推出的 CH32V307VCT6：144 引脚
2	主频	工作频率最高为 144MHz，本系统初始化运行频率为 72MHz。
3	程序空间 Flash	Flash 存储器，480KB，地址范围：0x0800_0000~0x0807_7FFF，分为 1920 个扇区，每个扇区大小为 256 字节
4	RAM 空间	SRAM 存储器，64KB，地址范围：0x2000_0000-0x2001_0000
5	内部主要硬件模块	GPIO、UART、SysTick、Timer、PWM、RTC、WDG、12 位 A/D、SPI、I2C、TKEY、CAN、USB、以太网等。

AHL-GEC-IDE（for AHL-CH32V307）



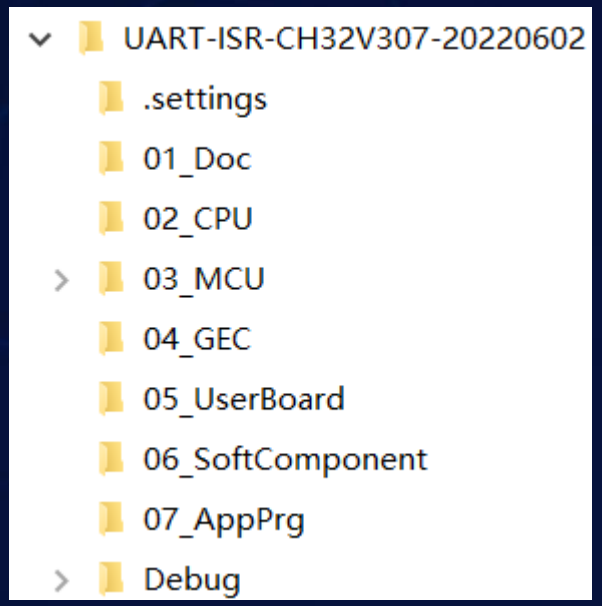
与AHL-GEC结合，具备直接直接下载运行，prinft打桩调试功能，实现GEC的类PC开发模式

RT-Thread驻留在AHL-CH32V307的BIOS中



RTOS驻留于AHL-CH32V307的BIOS内部，研究了RTOS下编程的统一接口，可以实现不同RTOS编程的可移植与可复用

标准工程框架



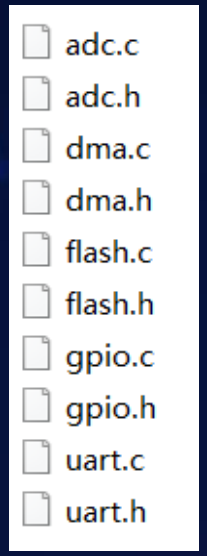
做到“分门别类，各有归处”

main之前的运行流程

- 第一阶段，寻找第一句被执行的指令。
  - 在“..\03_MCU\Linker_file\CH32V307.ld”文件中，“ENTRY(_start)”设定芯片执行的第一条指令存放在标签“_start”处。“_start”在启动文件中定义，其第一条指令为“j handle_reset”，即进入启动流程的第二阶段——执行复位服务程序。
- 第二阶段，执行复位服务程序。
  - 复位服务程序handle_reset在“..\03_MCU\startup\system_ch32v30x.c”文件中，首先初始化全局指针（Global Pointer，GP），GP可以优化±2KB内全局变量的访问；
  - 其次初始化堆栈指针（Stack Pointer，SP），把_estack的值赋给SP；接着对数据初始化：
  - 将数据段内容从Flash复制到RAM中，清零未初始化BSS数据段，使用机器模式并打开中断；
  - 最后对芯片的中断向量表、系统时钟及看门狗进行初始化，对于user程序，还要判断其是否需要使用BIOS的中断服务程序完成了这些工作，芯片就可以跳转到main函数中运行了。

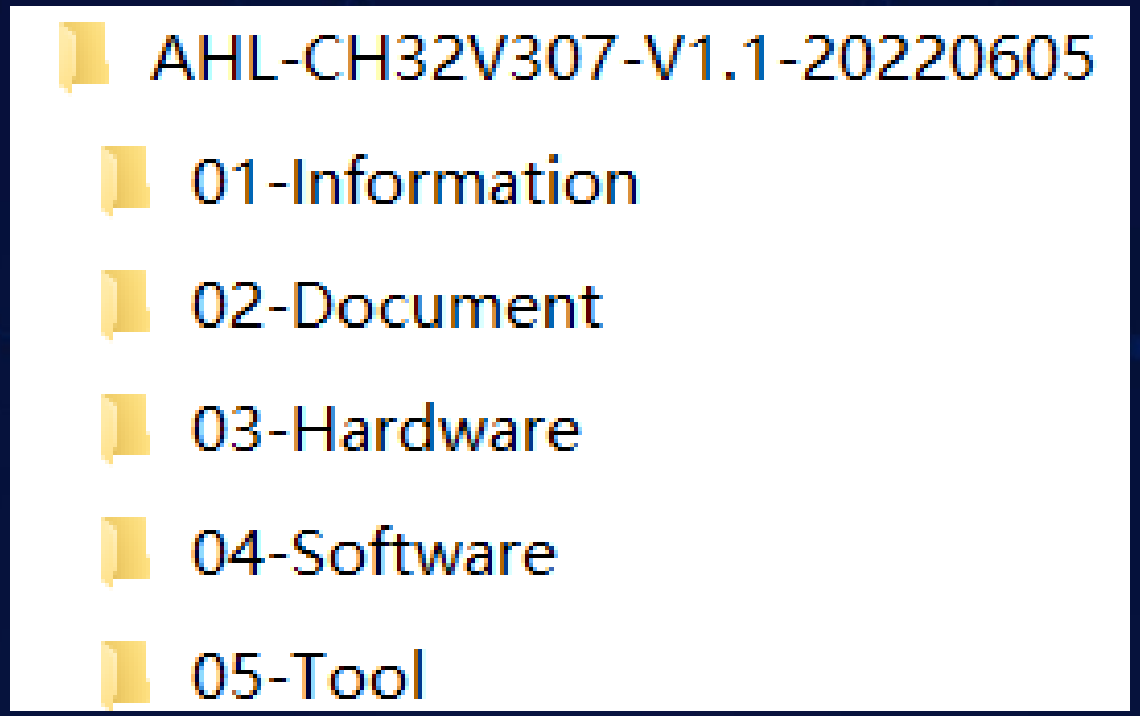
基础构件

基础构件是面向芯片级的硬件驱动构件，是符合软件工程封装规范的芯片硬件驱动程序。其特点是面向芯片，以知识要素为核心，以模块独立性为准则进行封装。



务必做到：一个构件由一个.h与一个.c文件构成，而不是多个文件

软件及文档电子资源

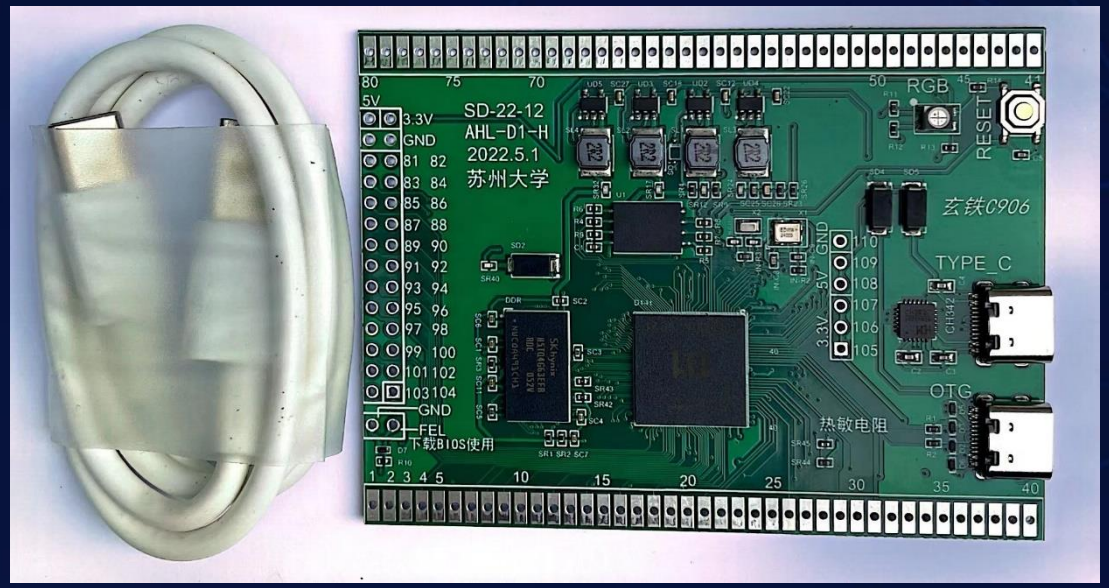

AHL-CH32V307-V1.1-20220605
01-Information
02-Document
03-Hardware
04-Software
05-Tool

其他型号

AHL-CH32V307-NB-IoT
AHL-CH32V307-WiFi
AHL-CH32V307-Cat1
AHL-CH32V307-EORS

基于RISC-V的D1-H构件GEC生态系统

硬件：AHL-D1-H

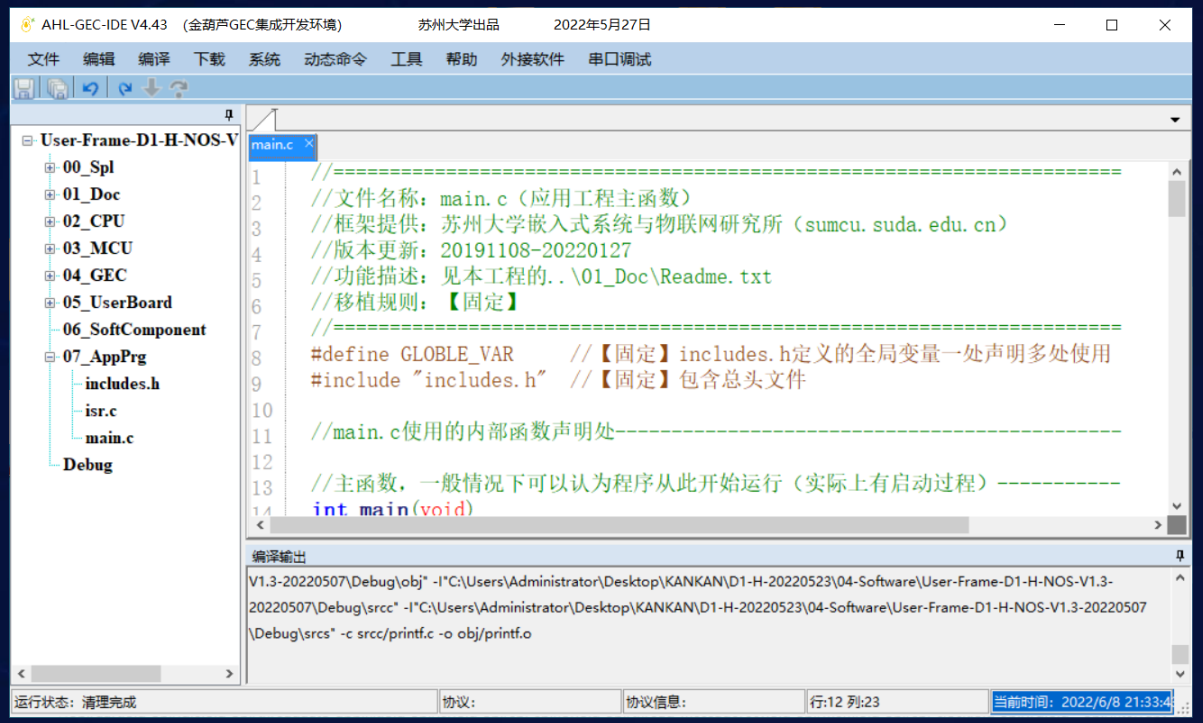


芯片的基本资源

名称	容量	寻址范围
DDR	512MB	0x4000_0000~0x6000_0000
SPI Nand Flash	256MB	0x0~0x1000_0000

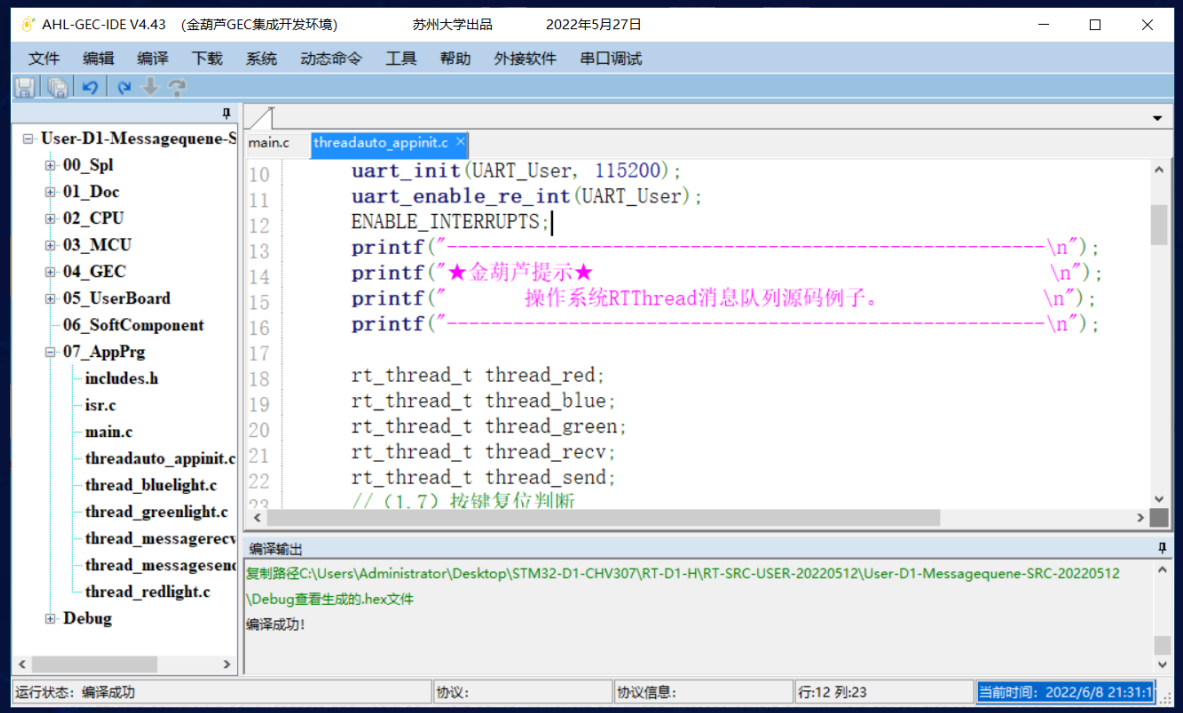
端口号	引脚数	引脚名	硬件最小系统复用引脚
B	13	PB[0-12]	PB12、PB9、PB8、PB5、PB4、PB1、PB0
C	8	PC[0-7]	PC2、PC3、PC4、PC5、PC6、PC7
D	23	PD[0-22]	PD16、PD21
E	18	PE[0-17]	—
F	7	PF[0-6]	—
G	19	PG[0-18]	—

AHL-GEC-IDE（for AHL-D1-H）



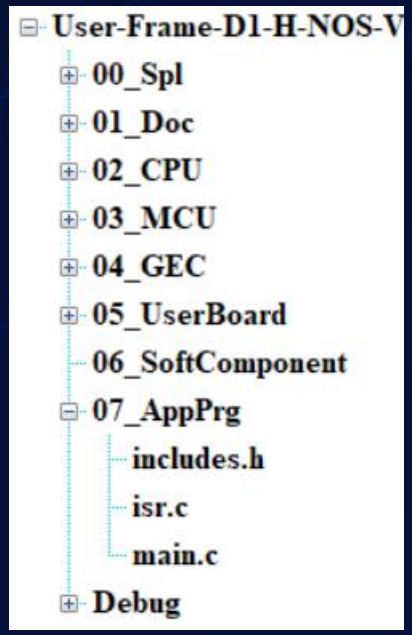
与AHL-GEC结合，具备直接直接下载运行，prinft打桩调试功能，实现GEC的类PC开发模式

RT-Thread驻留在AHL-D1-H的BIOS中



RTOS驻留于AHL-D1-H的BIOS内部，研究了RTOS下编程的统一接口，可以实现不同RTOS编程的可移植与可复用

标准工程框架



做到“分门别类，各有归处”

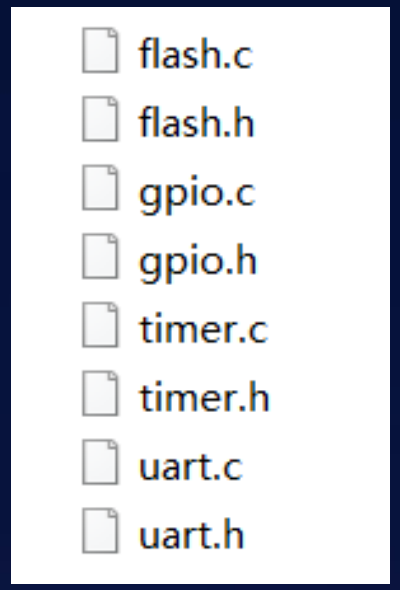
main之前的运行流程

- 第一阶段，上电启动运行内部固化引导程序。
  - D1上电后，首先运行内置ROM中引导程序，这个内置ROM被称为Boot ROM，简称BROM。
  - 在D1芯片中，其地址从0x0开始，功能为在外部存储器中寻找后续执行的程序。
  - 外部存储器的主要类型有NOR Flash、NAND Flash、SD card、eMMC等，例如，AHL-D1开发板的外部存储器是NOR Flash。
  - 在AHL-D1软件的工程框架中，在“..\03_MCU\startup\start.S”中可以看到给BROM配置的引导头信息，包含第二阶段程序加载器（Second Program Loader，SPL）的基本信息。

- **第二阶段**，片内SDRAM运行SPL程序。
  - 当BROM启动完成后，接下来根据引导头的信息加载SPL程序到片内SDRAM（同步动态随机存取内存，Synchronous Dynamic Random-Access Memory，简称SDRAM）中，从地址0x00020000开始运行，SPL具有初始化时钟、串口、片外DDR（Double Data Rage RAM，双数据率RAM）等功能。
  - 在AHL-D1软件的工程框架中，之后会调用“..\00_boot0\sys-copyself.c”中的sys_copyself函数，将所有程序读取到片外DDR中，并将PC指向0x40000000即可跳转到外接DDR中。
- **第三阶段**，片外DDR运行程序。
  - 程序根据PC指向地址跳转到片外DDR中运行，此时会再去运行一遍start.S，进行一些配置操作。
  - 其中对于串口、时钟以及外接ddr的初始化，通过位置无关代码来判断当前PC地址与链接文件中的地址是否相同从而避免重复运行，然后进行堆栈的初始化，开机器模式中断，最后跳转到main函数中运行。

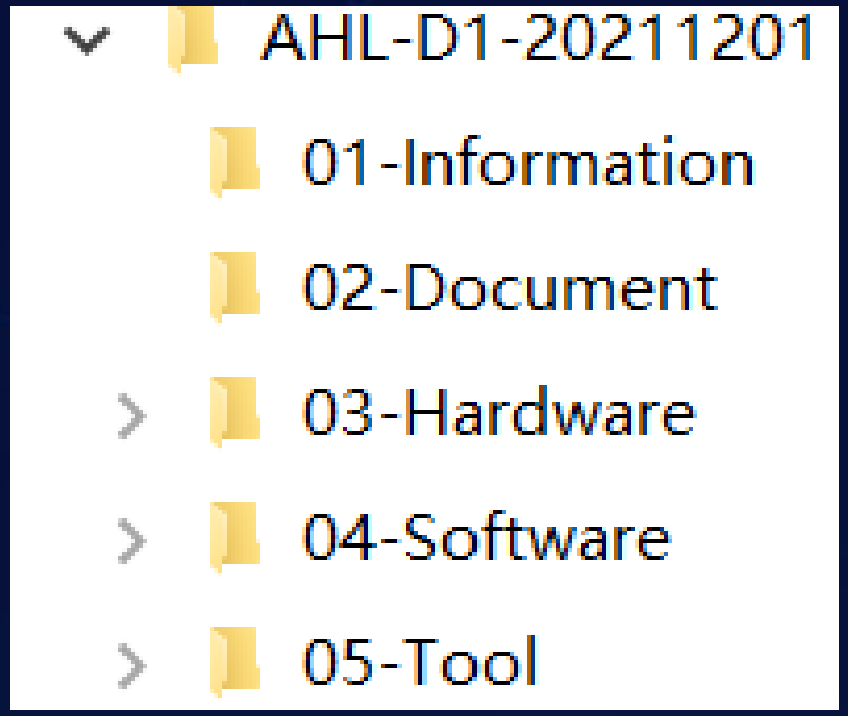
基础构件

- 基础构件是面向芯片级的硬件驱动构件，是符合软件工程封装规范的芯片硬件驱动程序。其特点是面向芯片，以知识要素为核心，以模块独立性为准则进行封装。



务必做到：一个构件由一个.h与一个.c文件构成，而不是多个文件

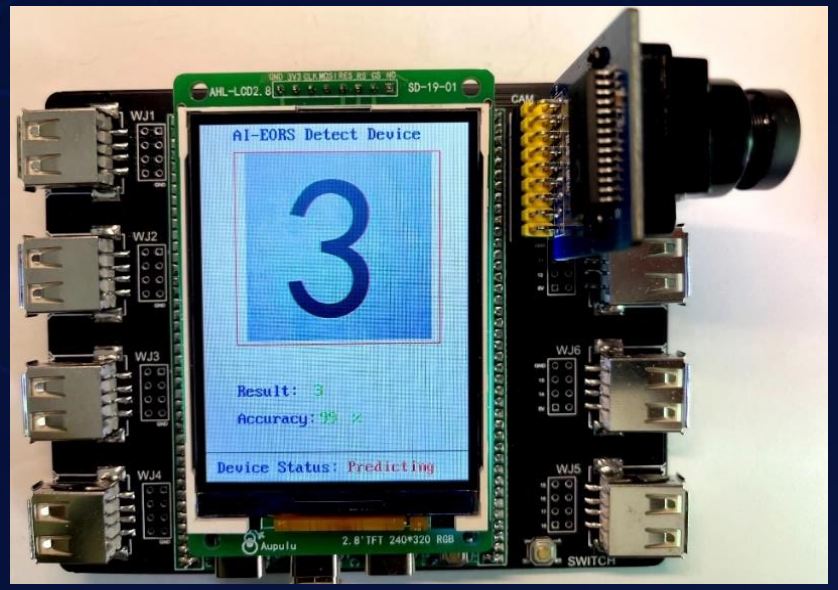
软件及文档电子资源


AHL-D1-20211201
01-Information
02-Document
03-Hardware
04-Software
05-Tool

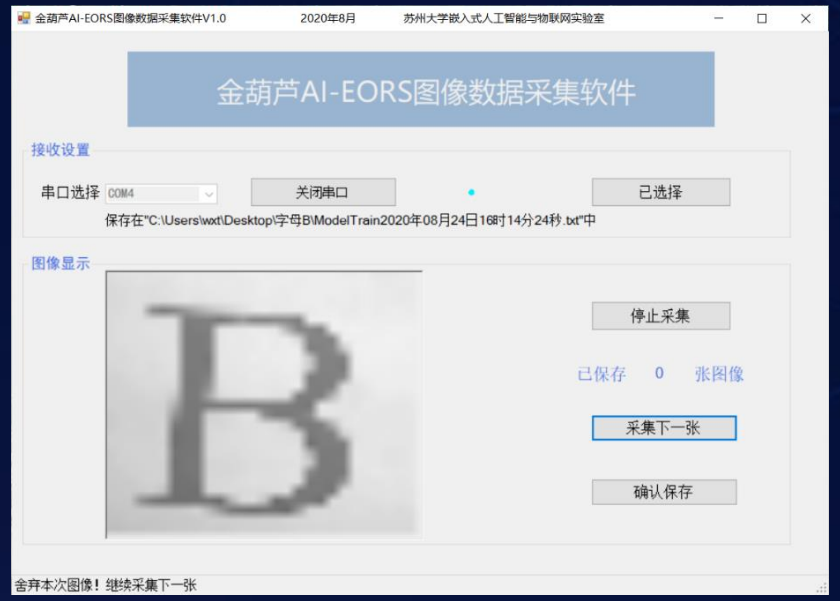
嵌入式人工智能：物体认知系统

- 基于图像识别的嵌入式物体认知系统（Embedded Object Recognition System，EORS）是利用嵌入式计算机通过摄像头采集物体图像，利用图像识别相关算法进行训练、标记，训练完成后，可进行推理完成对图像的识别。
- EORS主要目标用于嵌入式人工智能入门教学，试图把复杂问题简单化，利用清晰的流程体现人工智能中“标记、训练、推理”的基本知识要素。
- 同时期望达到“学习汉语拼音从啊（a）、喔（o）、鹅（e）开始，学习英语从A、B、C开始，学习嵌入式人工智能从EORS开始”的效果。
- 利用前文已建立的D1-H的GEC生态系统，辅以摄像头和LCD传感器等，设计一套原理清晰、简单易用、可对简单物体进行识别的基于图像处理的嵌入式物体认知系统。

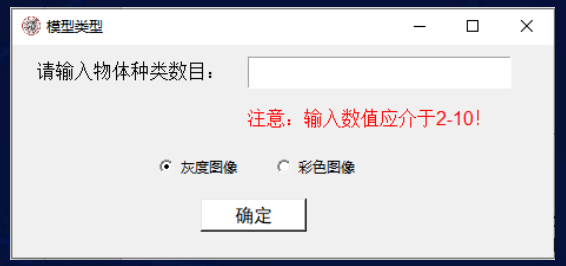
硬件实物

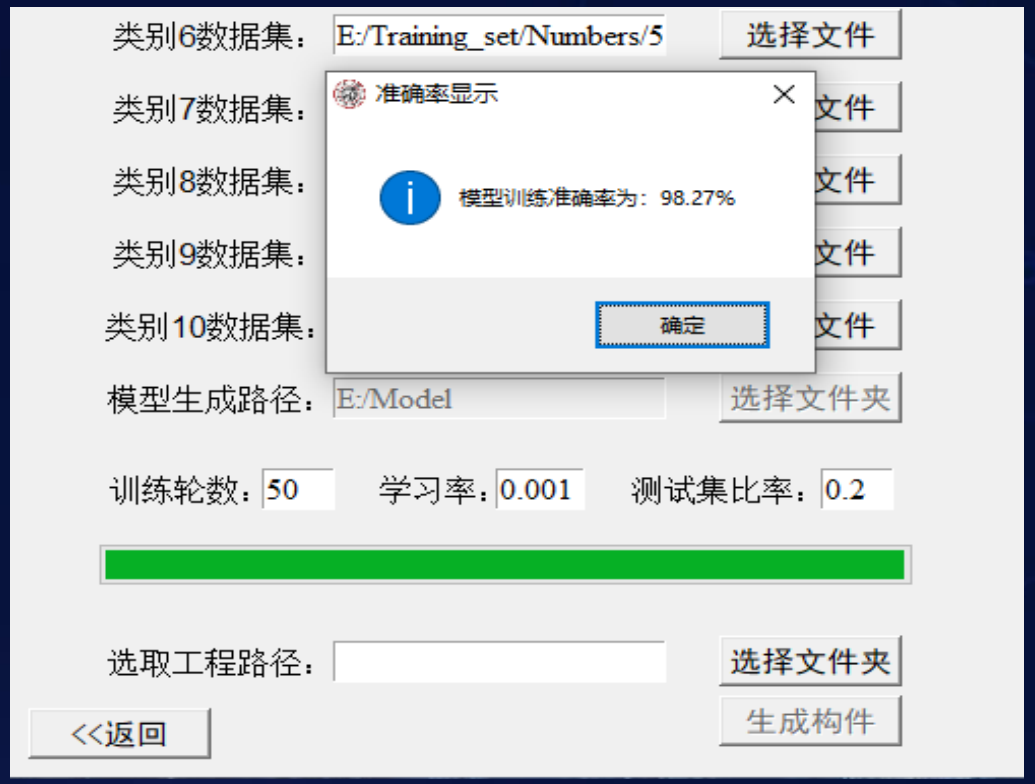

AML-LCD2.8
SD-19-01
CAM
AI-EORS Detect Device
3
Result: 3
Accuracy: 95 %
Device Status: Predicting
WJ1
WJ2
WJ3
WJ4
WJ5
WJ6
Aupulu
2.8'TFT 240*320 RGB
SWITCH

图像采集软件

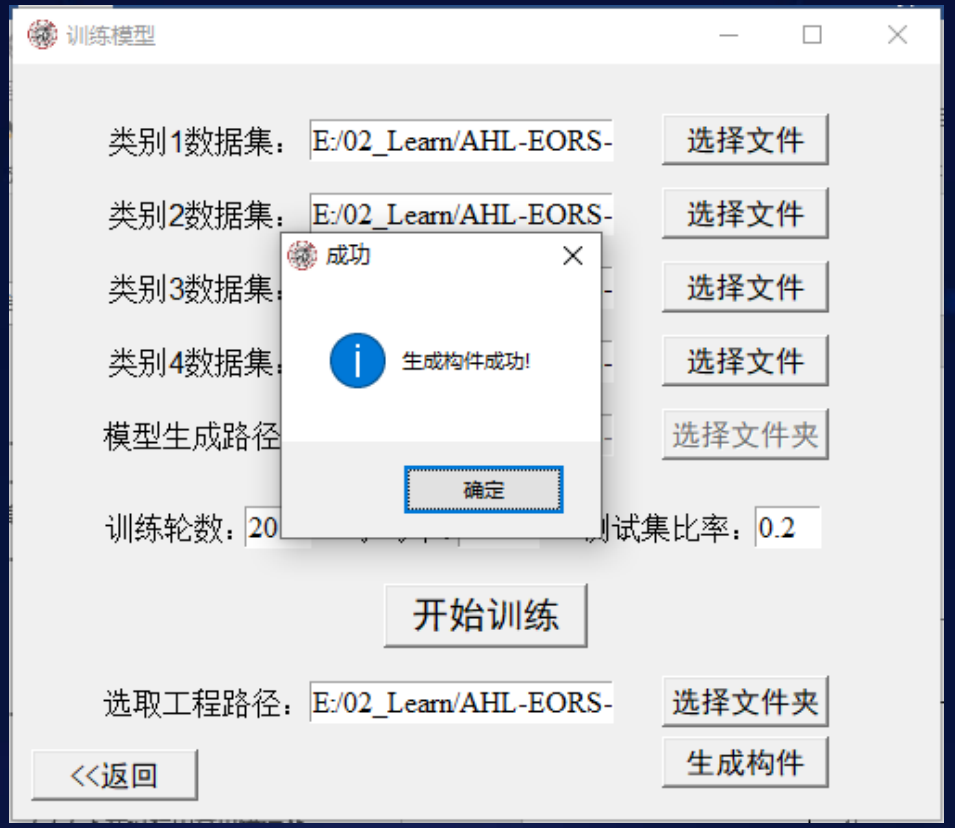

金葫芦AI-EORS图像数据采集软件V1.0
2020年8月
苏州大学嵌入式人工智能与物联网实验室
金葫芦AI-EORS图像数据采集软件
接收设置
串口选择
COM4
关闭串口
已选择
保存在"C:\Users\wxt\Desktop\字母B\ModelTrain2020年08月24日16时14分24秒.txt"中
图像显示
B
停止采集
已保存 0 张图像
采集下一张
确认保存
舍弃本次图像！继续采集下一张

PC机模型训练界面

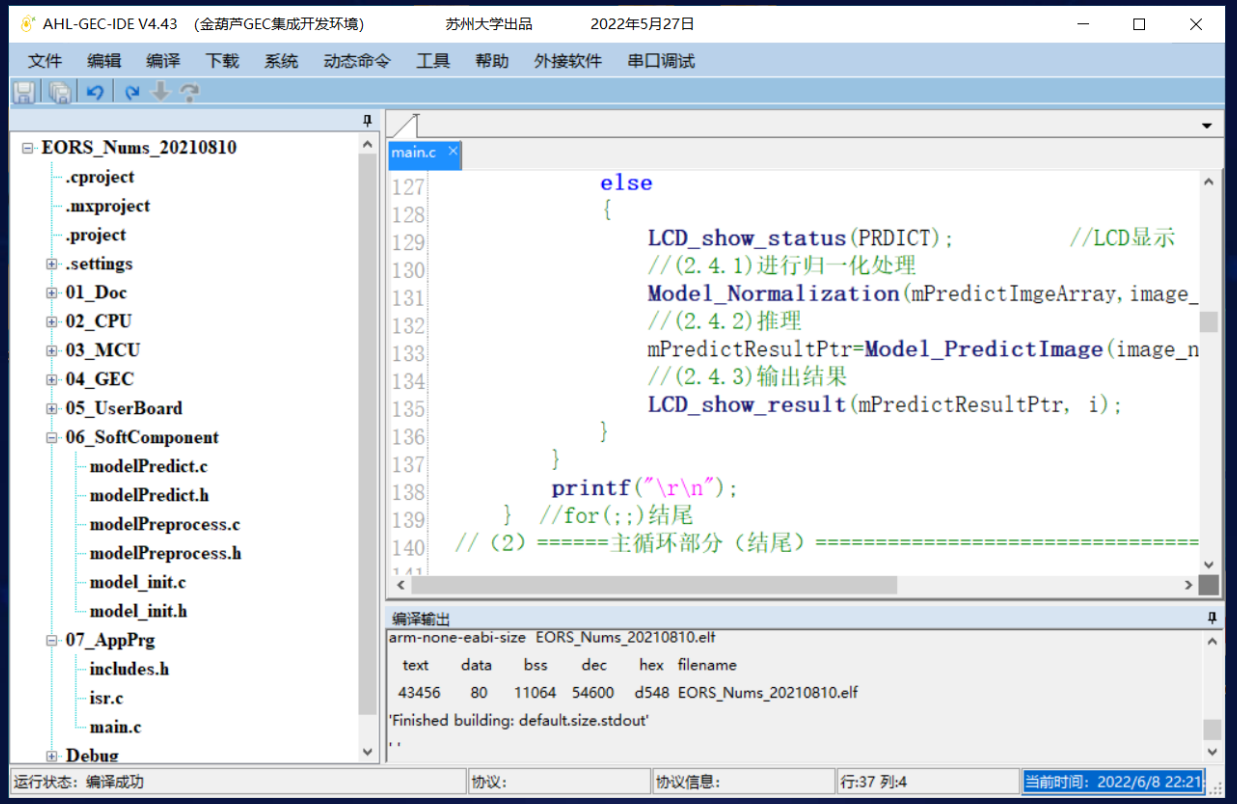

模型类型
请输入物体种类数目：
注意：输入数值应介于2-10!
灰度图像
彩色图像
确定


类别6数据集：
E:/Training_set/Numbers/5
选择文件
类别7数据集：
类别8数据集：
类别9数据集：
类别10数据集：
文件
准确率显示
模型训练准确率为：98.27%
确定
模型生成路径：
E:/Model
选择文件夹
训练轮数：50
学习率：0.001
测试集比率：0.2
选取工程路径：
选择文件夹
生成构件
<<返回

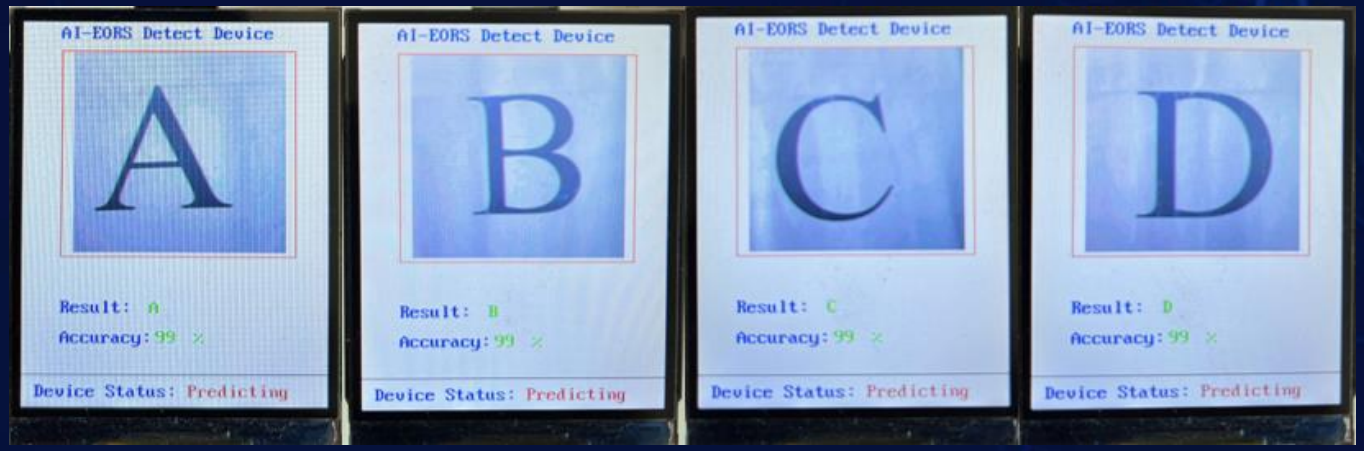
生成推理构件


训练模型
类别1数据集：
E:/02_Learn/AHL-EORS-
选择文件
类别2数据集：
E:/02_Learn/AHL-EORS-
选择文件
类别3数据集：
选择文件
类别4数据集：
选择文件
模型生成路径
选择文件夹
成功
生成构件成功!
确定
训练轮数：20
试集比率：0.2
开始训练
选取工程路径：
E:/02_Learn/AHL-EORS-
选择文件夹
<<返回
生成构件

生成嵌入式推理工程


AHL-GEC-IDE V4.43 (金葫芦GEC集成开发环境) 苏州大学出品 2022年5月27日
文件 编辑 编译 下载 系统 动态命令 工具 帮助 外接软件 串口调试
EORS_Nums_20210810
.cproject
.mxproject
.project
.settings
01_Doc
02_CPU
03_MCU
04_GEC
05_UserBoard
06_SoftComponent
modelPredict.c
modelPredict.h
modelPreprocess.c
modelPreprocess.h
model_init.c
model_init.h
07_AppPrg
includes.h
isr.c
main.c
Debug
main.c
else
{
LCD_show_status(PRDICT); //LCD显示
//(2.4.1)进行归一化处理
Model_Normalization(mPredictImgeArray,image_
//(2.4.2)推理
mPredictResultPtr=Model_PredictImage(image_n
//(2.4.3)输出结果
LCD_show_result(mPredictResultPtr, i);
}
}
printf("\r\n");
} //for(;;)结尾
// (2) ======主循环部分（结尾）==========
编译输出
arm-none-eabi-size EORS_Nums_20210810.elf
text data bss dec hex filename
43456 80 11064 54600 d548 EORS_Nums_20210810.elf
'Finished building: default.size.stdout'
运行状态：编译成功
协议:
协议信息:
行:37 列:4
当前时间：2022/6/8 22:21

识别示例


AI-EORS Detect Device
A
Result: A
Accuracy: 99 %
Device Status: Predicting
AI-EORS Detect Device
B
Result: B
Accuracy: 99 %
Device Status: Predicting
AI-EORS Detect Device
C
Result: C
Accuracy: 99 %
Device Status: Predicting
AI-EORS Detect Device
D
Result: D
Accuracy: 99 %
Device Status: Predicting


AI-EORS Detect Device
Result: Circle
Accuracy: 97 %
Device Status: Predicting
AI-EORS Detect Device
Result: Rectangle
Accuracy: 99 %
Device Status: Predicting
AI-EORS Detect Device
Result: Triangle
Accuracy: 99 %
Device Status: Predicting

谢谢