# Homemada Script Luaguage Note

## 自制脚本语言 笔记

There is no good end for the fuck thieves country.

整理：LMin

整理时间：April 12, 2018

Email: luomin5417@gmail.com

Version: 0.1

# 目　　录

# 第 1 章 语言处理器基本概

学而不思则罔，思而不学则怠！

—孔子

## 1.1 机器语言与汇编语言

本节主要介绍了机器语言与汇编语言：

- 机器语言是可以由机器直接解释执行的语言，一般才用二进制形式 [?]。
- 汇编语言一般是相对于机器语言更易理解的语言，但是又因不同的体系结构不同具有不同的指令集。

## 1.2 解释器与编译器

本节主要介绍了解释器和编译器的区别：

- 解释器根据程序中的算法执行运算。
- 编译器将某种语言写成的程序转换为另一种语言的程序。

## 1.3 开发语言处理器

本节主要介绍语言处理器，综合了解释器和编译器的优点。

## 1.4 语言处理器结构与本书框架

语言处理器的基本结构，如图 1.1 所示，画出了语言处理器的基本处理结构，源程序通过词法分析进行单词排列，然后经过语法分析生成一个抽象语法树，然后根据是编译器还是解释器执行不同的步骤，如果是编译器则直接生成机器语言，如果是解释器则一边分析抽象语法树，一边输出执行结果。

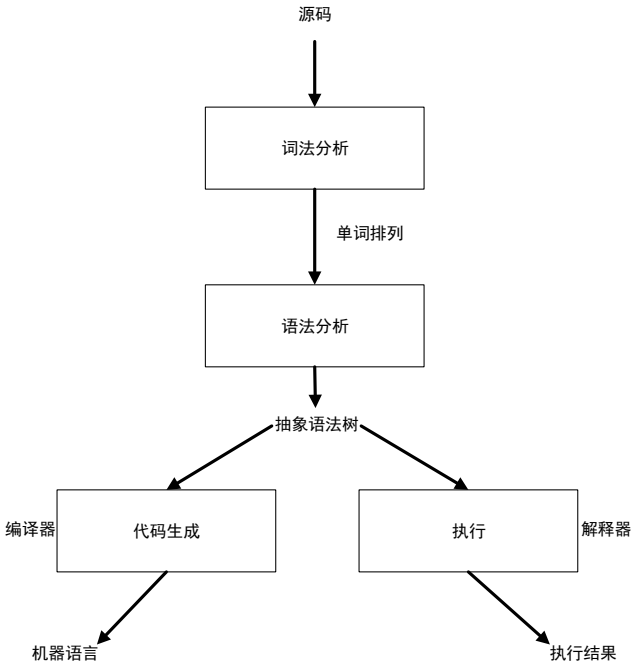


**图 1.1:** 语言处理器结构

无论是编译器还是解释器的结构都是大同小异的，第一步都是要先进行词法分析，由一长串字符分解成为多个更小的字符串单元，然后执行语法分析，把单词排列成为抽象语法树，到此解释器与编译器的行为都是一致的，在这之后编译器将转换为另外的语言，而解释器则将一边分析语法树一边执行。

# 第 2 章 设计脚本语言

故不登高山，不知天之高也；不临深溪，不知地之厚也; 不闻先王之遗言，不知学问之大也。

—荀子

## 2.1 麻雀虽小五脏俱全

设计脚本语言的基本元素如图 2.1 所示：

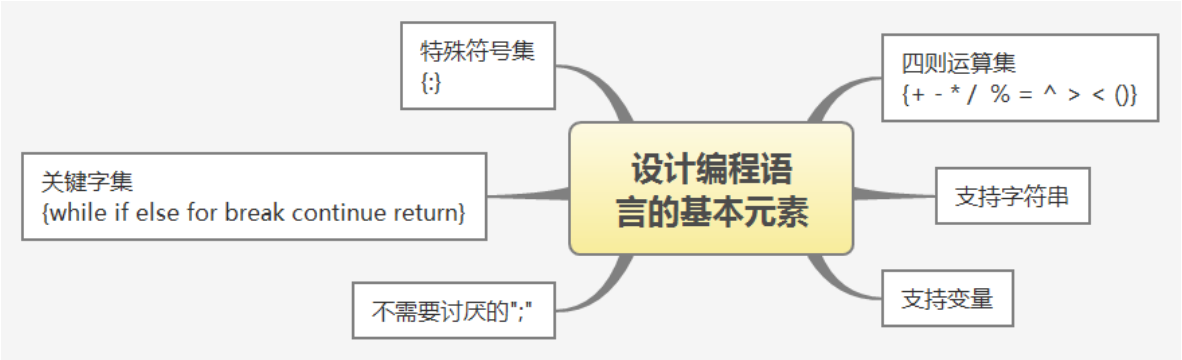


**图 2.1:** 脚本语言构成要素

只有具备这些基本元素才能算为一个完备的脚本语言, 暂定这些基本元素，后期在有需要再进行进一步的修改：

- 支持四则运算
- 支持字符串
- 支持变量
- 一些其它特性, 如不支持‘;’
- 支持基本逻辑关键字

# 第 3 章 分割单词

为往圣继绝学，为万世开太平！

—张载

## 3.1 Token 对象

# 第 4 章 全志 A83t Camera 开发

纸上得来终觉浅, 绝知此事要躬行。

—傅玄

## 4.1 总线与硬件接口

根据全志官方提供的《A83t_User_Manual》和《A83tDatasheet》两个文档，可以知道 A83t 支持 MIPI-CSI 总线和 CSI 总线，且 A83t 内部支持 ISP，A83t 体系架构图如图 4.1 所示。

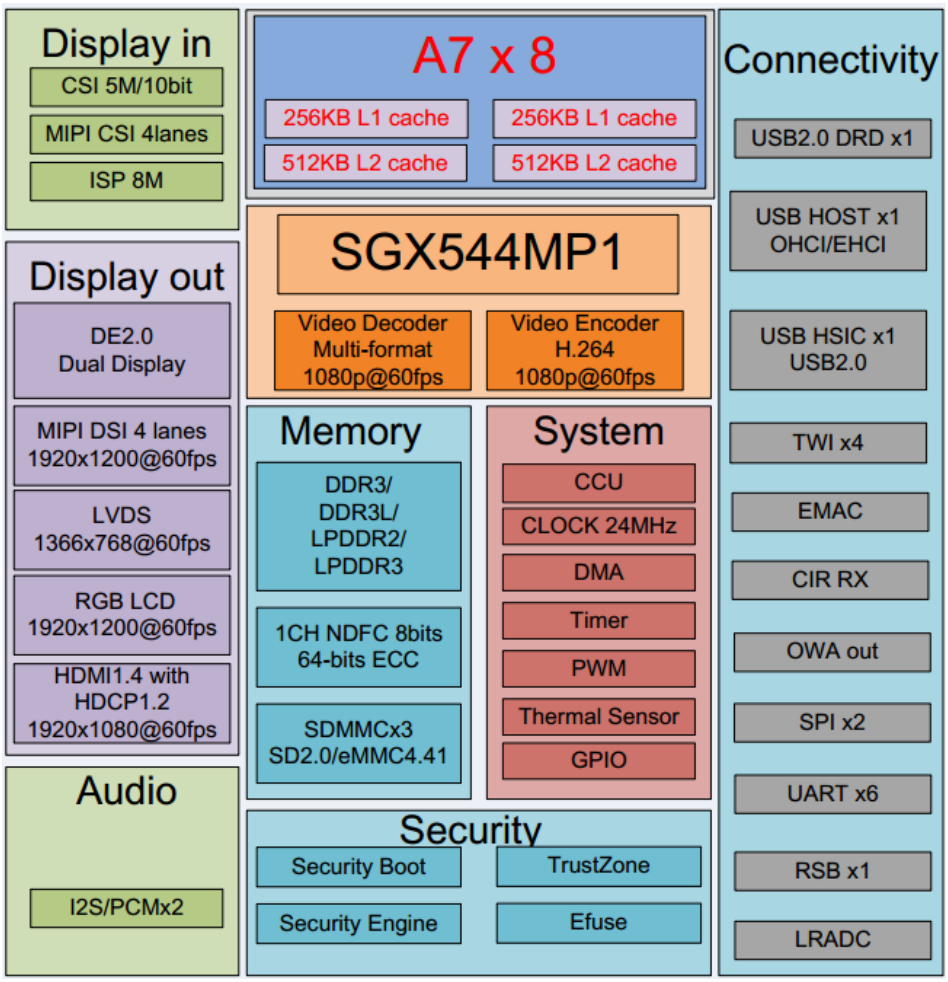


**图 4.1:** A83t 体系架构

CSI 接口的特性如图 4.2，

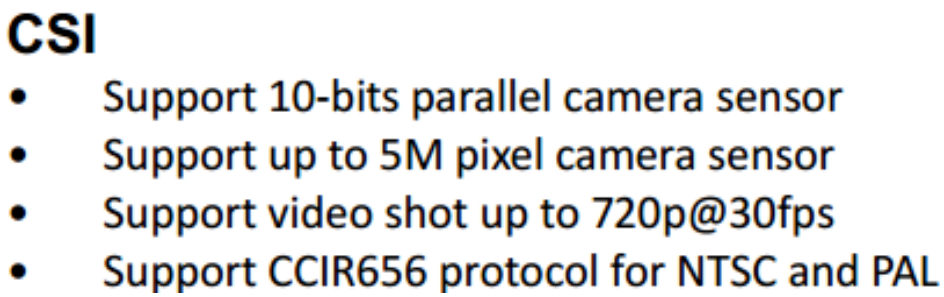


**图 4.2:** CSI 接口特性

MIPI-CSI 接口的特性如图 4.3，

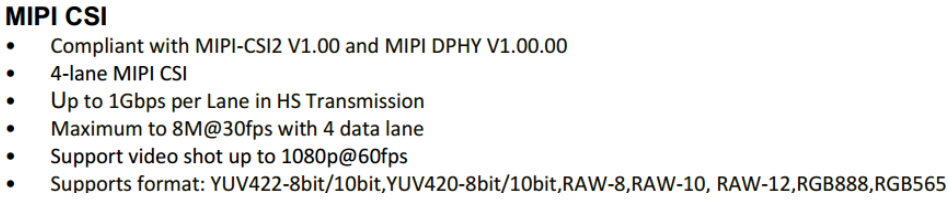


**图 4.3:** MIPI-CSI 接口特性

## 4.2 硬件原理图

在调试的开发板上采用的是 gc2145+gc0329 连接到 A83t 的 CSI0 接口,gc2145 器件原理图如图 4.4 所示，

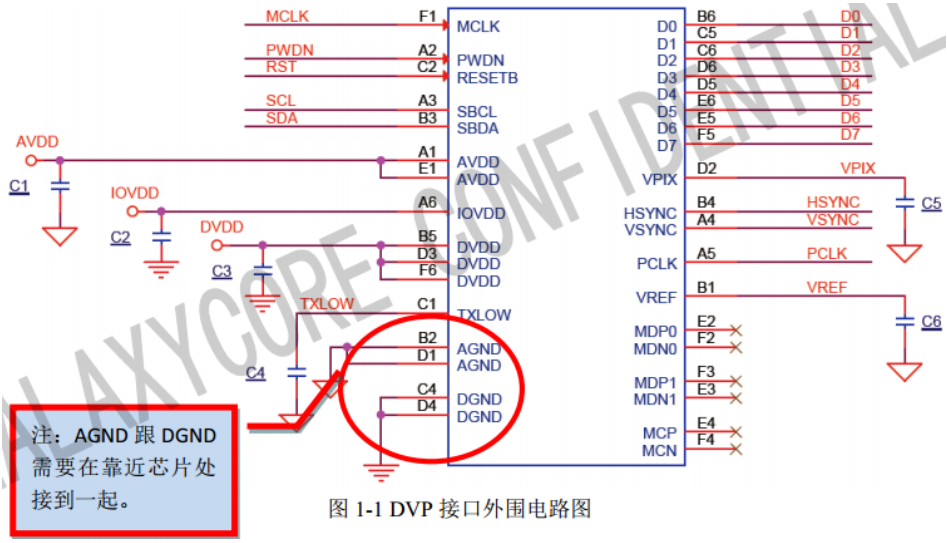


**图 4.4:** GC2145 器件原理图

## 4.3 驱动源码

## 4.4 内核与驱动配置

调试 camera 需要配置内核与 sysconfig.fex 文件，内核配置如图 4.6 所示。

1. Device Drivers-><*>Multimedia support-><*>Video For linux
2. Device Drivers-><*>Multimedia support->[*]Video capture adapters ->[*]V4L platform devices-><M>sunxi video front end(camera and etc)driver
3. Device Drivers-><*>Multimedia support->[*]Video capture adapters ->[*]V4L platform devices-><M>v4l2 driver for SUNXI

**图 4.5:** make menuconfig 配置

除了基本的内核对 camera 的支持，还需要修改内核 Makefile 添加爱 gc2145 驱动的编译，修改文件路径: linux-3.4/drivers/media/video/sunxi-vfe/device/Makefile, 修改内容如图 4.7 所示，在 24 行添加了 gc2145sensor 模组的驱动编译，

```
20 obj-m += gc0307.o
21 obj-m += gc0308.o
22 obj-m += gc0328.o
23 obj-m += gc0329.o
24 obj-m += gc2145.o
25 obj-m += gc0311.o
26 obj-m += hi253.o
27 obj-m += sp2518.o
28 obj-m += sp2519.o
```

**图 4.6:** 添加 GC2145 驱动

完成内核配置之后还需要对 android 系统进行配置，让 android 系统在启动时加载对应的驱动模块, 对应的修改目录：android/device/softwinner/octopus-f1/init.sun8i.rc, 修改内容如图 4.8 所示, 添加了加载 gc2145.ko 模块，去掉了 ov5640 模块的加载。

```
83 #csi module
84      insmod /system/vendor/modules/videobuf-core.ko
85      insmod /system/vendor/modules/videobuf-dma-contig.ko
86      insmod /system/vendor/modules/cci.ko
87      insmod /system/vendor/modules/actuator.ko
88      insmod /system/vendor/modules/vfe_os.ko
89      insmod /system/vendor/modules/vfe_subdev.ko
90      insmod /system/vendor/modules/dw9714_act.ko
91 #    insmod /system/vendor/modules/ov5640.ko
92      insmod /system/vendor/modules/gc2145.ko
93      insmod /system/vendor/modules/gc0308.ko
94      insmod /system/vendor/modules/vfe_v4l2.ko
```

图 4.7: 加载 GC2145 驱动

## 4.5 调试问题记录

将调试中遇到的一些问题记录如下：

1）将 **sysconfig.fex** 文件配置好之后，编译固件下载运行，在 **linux** 加载时出现卡住无法启动的问题。

```
2473      case CSI_SUBDEV_PWR_ON:
2474        vfe_dev_dbg("CSI_SUBDEV_PWR_ON\n");
2475        //power on reset
2476        vfe_gpio_set_status(sd,PWDN,1);//set the gpio to output
2477        vfe_gpio_set_status(sd,RESET,1);//set the gpio to output
2478        //active mclk before power on
2479        vfe_set_mclk_freq(sd,MCLK);
2480        vfe_set_mclk(sd,ON);
2481        //standby off io
2482        vfe_gpio_write(sd,PWDN,CSI_STBY_OFF);
2483        //reset on io
2484        vfe_gpio_write(sd,RESET,CSI_RST_ON);
2485        //power supply
2486        vfe_gpio_write(sd,POWER_EN,CSI_PWR_ON);
2487        vfe_set_pmu_channel(sd,IOVDD,ON);
2488        vfe_set_pmu_channel(sd,AVDD,ON);
2489        vfe_set_pmu_channel(sd,DVDD,ON);
2490        vfe_set_pmu_channel(sd,AFVDD,ON);
2491        usleep_range(10000,12000);
2492        //reset after power on
2493        vfe_gpio_write(sd,RESET,CSI_RST_OFF);
2494        //usleep_range(30000,31000);
2495        vfe_dev_dbg("CSI_SUBDEV_PWR_ON END\n");
2496        break;
```

图 4.8: 定位代码

解决方法：在各个驱动里面添加打印，定位具体是什么位置导致了 CPU 死掉，初步判断是有空指针之类的问题，但是内核栈没有更多的信息打印出来。还想到一个办法，便是通过将 android 中设置的自动加载 camera 各个驱动，换成手动加载驱动，看看是如何挂掉的。添加打印后和换成手动加载驱动之后定位到挂掉的代码位置，代码路径 linux-3.4/drivers/media/video/sunxi-vfe/device/gc2145.c, 如图 4.8 所示, 最后定位到了 vfe_set_pmu_channel(sd, IOVDD, ON) 这一句，原来是在 sys_config.fex 中 iovdd-csi 位置设置错误，放到了 aldo2 下面，aldo2 是控制 sdram 和 pll 的电源。

完成电源配置之后，摄像头便可以正常启动了, 此时的 sys_config.fex 的电源分配如图 4.9 所示, 将 iovdd-csi 分配到了 axp81x_dldo3, 将 dvdd-csi-18 分配在了 axp81x_eldo1,avdd-csi 分配在了 axp81x_dldo4。

```
;-------------------------------------------------------------------------------------------------------
;pmu1 is pmu81x
;regulator tree
;-------------------------------------------------------------------------------------------------------
[pmu1_regu]
regulator_count = 23
regulator1      = "axp81x_dcdc1 none vcc-nand vcc-nor vcc-usb0-33 vcc-card vcc-io vcc-wifi-io vcc-io1 vcc-io2 vcc-sensor vcc-camera-33"
regulator2      = "axp81x_dcdc2 none vdd-cpua"
regulator3      = "axp81x_dcdc3 none vdd-cpub"
regulator4      = "axp81x_dcdc4 none vdd-gpu"
regulator5      = "axp81x_dcdc5 none vcc-dram"
regulator6      = "axp81x_dcdc6 none vdd-sys vdd-usb0-09 vdd-hdmi-09"
regulator7      = "axp81x_dcdc7 none"
regulator8      = "axp81x_rtc none"
regulator9      = "axp81x_aldo1 none vcc-dsi-18  vcc-hdmi-18"
regulator10     = "axp81x_aldo2 none vdd-drampll vdd-lpddr-18 vcc-pll vcc-adc vcc-cpvdd vcc-ldoin"
regulator11     = "axp81x_aldo3 none vcc-avcc vcc-pl"
regulator12     = "axp81x_dldo1 none vcc-vibrator dvdd-ephy"
regulator13     = "axp81x_dldo2 none"
regulator14     = "axp81x_dldo3 none iovdd-csi afvcc-csi"
regulator15     = "axp81x_dldo4 none vcc-ephy-io vcc-pd avdd-csi"
regulator16     = "axp81x_eldo1 1  dvdd-csi-18"
regulator17     = "axp81x_eldo2 1  vcc-csi2-18 vcc-lvds-18 vcc-efuse-18"
regulator18     = "axp81x_eldo3 1  dvdd-csi3-18"
regulator19     = "axp81x_fldo1 none avdd-ephy vcc-hsic-12"
regulator20     = "axp81x_fldo2 none vdd-cpus"
regulator21     = "axp81x_gpio0ldo none vcc-ctp"
regulator22     = "axp81x_gpio1ldo none"
regulator23     = "axp81x_dc1sw 1 vcc-lcd-0"
```

图 **4.9:** sys_config 电源分配

iovdd 是摄像头模块 io 接口电源,avdd 是摄像头模块模拟电路电源,dvdd 是摄像头模块数字电路电源, 具体电源配置电压如图 4.10 所示,

2) 驱动挂死

解决方法：

## 4.6 总结

```
43 ;----------------------------------------------------------------------------------
44 ;
45 ;  各路电压输出语法说明:
46 ;
47 ;  电压名称  = 100XXXX  : 表示把该路电压设置为XXXX指定的电压值，同时打开输出开关
48 ;  电压名称  = 000XXXX  : 表示把该路电压设置为XXXX指定的电压值，同时关闭输出开关，当有需要时由内核驱动打开
49 ;  电压名称  = 0        : 表示关闭该路电压输出开关，不修改原有的值
50 ;  Note:eldo2 before dldo4 > 100us; first off dldo4 then eldo2
51 ;----------------------------------------------------------------------------------
52 [power_sply]
53 dcdc1_vol                = 1003100
54 dcdc2_vol                = 1000900
55 dcdc3_vol                = 1000900
56 dcdc4_vol                = 1000900
57 ;dcdc5_vol                = 1001500
58 dcdc6_vol                = 1000900
59 aldo1_vol                = 1002800
60 aldo2_vol                = 1001800
61 aldo3_vol                = 1003000
62 eldo1_vol                = 1001800
63 eldo2_vol                = 1001800
64 eldo2_vol                = 1001800
65 dldo1_vol                = 3100
66 dldo2_vol                = 0003100
67 dldo3_vol                = 1002800
68 dldo4_vol                = 1002800
69 fldo2_vol                = 1000900
70 gpio0_vol                = 1003100
71 gpio1_vol                = 3100
```

图 **4.10:** sys_config 电源电压配置

# 第 5 章 多变量线性回归