使用产品之前请仔细阅读产品说明书

JW-MVS01 整机说明书

版本：v1.0

版本更新表

版本	版本特征	撰写人	Date
V1.0	First Version		2017/11/20

目录

1 注意事项

商标

本手册所提及的商标与名称都归其所属公司所有。

注意

1. 使用前，请先详细阅读说明书，避免误操作导致产品损坏；
2. 请将此产品放置在 0℃<=工作环境<=60℃、95%RH 的环境下，以免因过冷、热或受潮导致产品损坏；
3. 请勿将此产品做强烈的机械运动，以及在没有作好静电防护之前对此产品操作；
4. 请确保输入电压在 9~36V 范围以内，以免造成不可预测的后果；
5. 在安装任何外接卡或模组之前，請先关闭电源；
6. 禁止对机箱内主板产品进行私自更改、拆焊，对此所导致的任何后果我司不承担任何责任；

2 产品概述

2.1 产品特点介绍

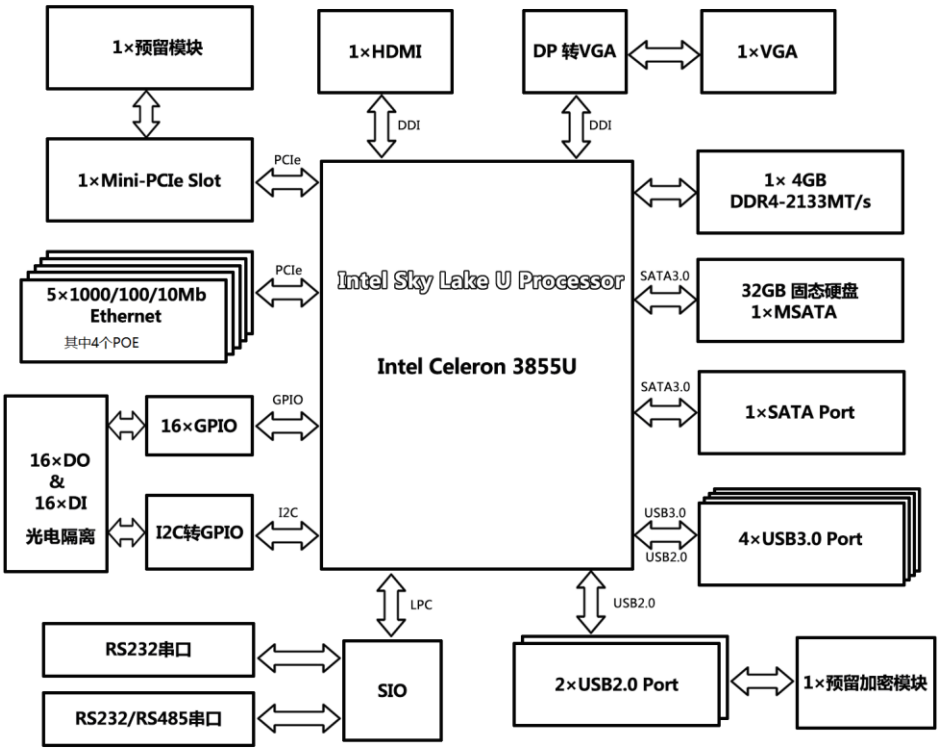
本产品基于 Intel Sky Lake 第六代处理器平台。产品采用 Intel 3855U/i3-6100U 等处理器，配备 1 个 VGA 显示输出接口；配备 5 个千兆网口（其中 4 个 POE）、4 个 USB3.0、2 个 RS-232、16 路输入/16 路输出隔离 DIO 接口。

产品外壳采用散热性能优良的铝型材主体而设计，表面采用黑色喷砂氧化处理工艺，产品结构简洁，外形美观； 产品基于Intel Sky Lake U系列高能效处理器平台，具备丰富的IO扩展，是一款为机器视觉、工业网关等应用而设计的无风扇工控电脑产品。

2.2 基本信息表

整机参数	
处理器	Intel® 赛扬 3855U/i3-6100U/i5-6200U/i7-6500U
	双核 1.6GHz / 双核 2.4GHz
内存	4GB DDR4-2400MT/s
存储	32GB SSD 固态硬盘
显卡	Intel® HD graphics 510/520
网络	5 个千兆网口(4*POE-802.3af)
	Intel I211AT
扩展特性	
IO 接口	1 个 VGA
	5 个 RJ45 网口
	4 个 USB3.0,1 个 USB2.0
	2 个 RS232
	16 个 DI & 16 个 DO
预留扩展	1 个 mini-PCIe 扩展槽
	1 个 mSATA 接口
整机特性	
系统支持	Win7/Win8/Win10
	Linux/ Unix
供电	9~36V DC 宽压输入
机箱特征	
外形	定制机箱
机箱尺寸	210mm（长）×140mm （宽）×90mm （高）
运行环境	
温度	运行温度 0℃~+60℃ 存储温度 -40℃~+85℃
相对湿度	非运行时 95%，于 25℃至 30℃温度下不凝结

2.3 功能区块图


1×预留模块
1×HDMI
DP 转VGA
1×VGA
DDI
DDI
1×Mini-PCIe Slot
PCIe
Intel Sky Lake U Processor
Intel Celeron 3855U
1× 4GB
DDR4-2133MT/s
PCIe
SATA3.0
5×1000/100/10Mb
Ethernet
其中4个POE
32GB 固态硬盘
1×MSATA
SATA3.0
1×SATA Port
GPIO
16×GPIO
16×DO
&
16×DI
光电隔离
I2C
I2C转GPIO
USB3.0
USB2.0
4×USB3.0 Port
LPC
USB2.0
RS232串口
RS232/RS485串口
SIO
2×USB2.0 Port
1×预留加密模块

3 实物介绍

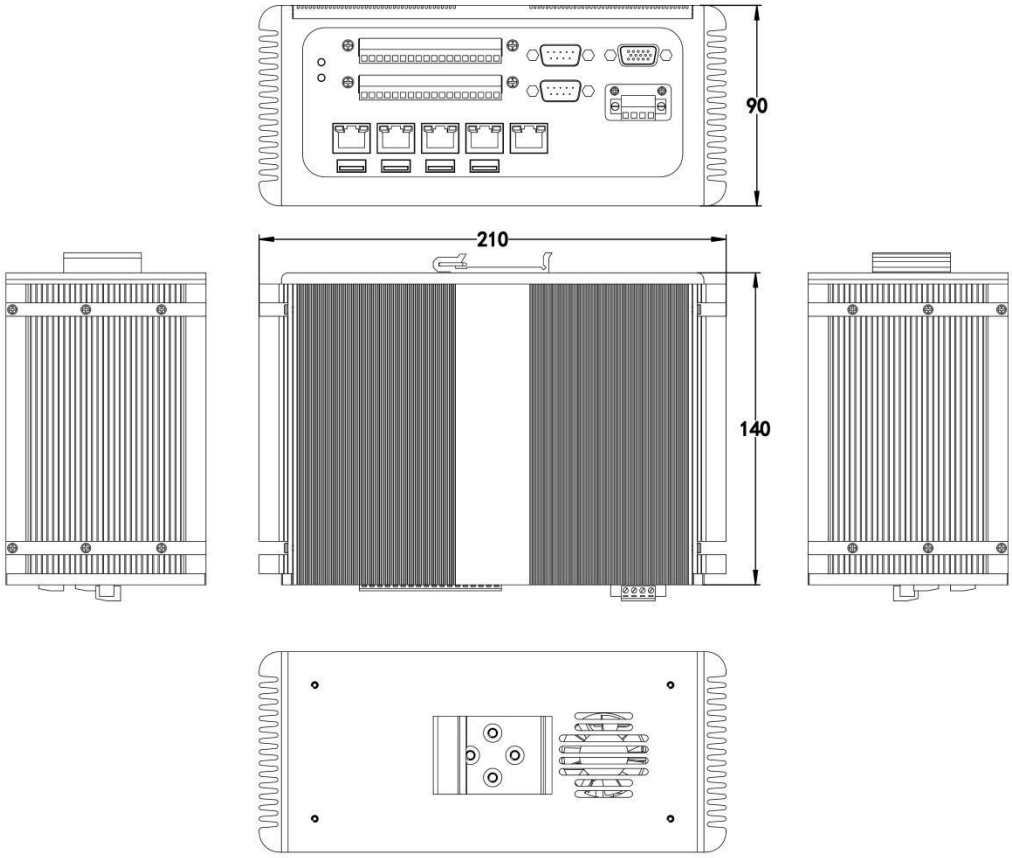
3.1 产品实物

整机侧视图

整机前视图

整机后视图

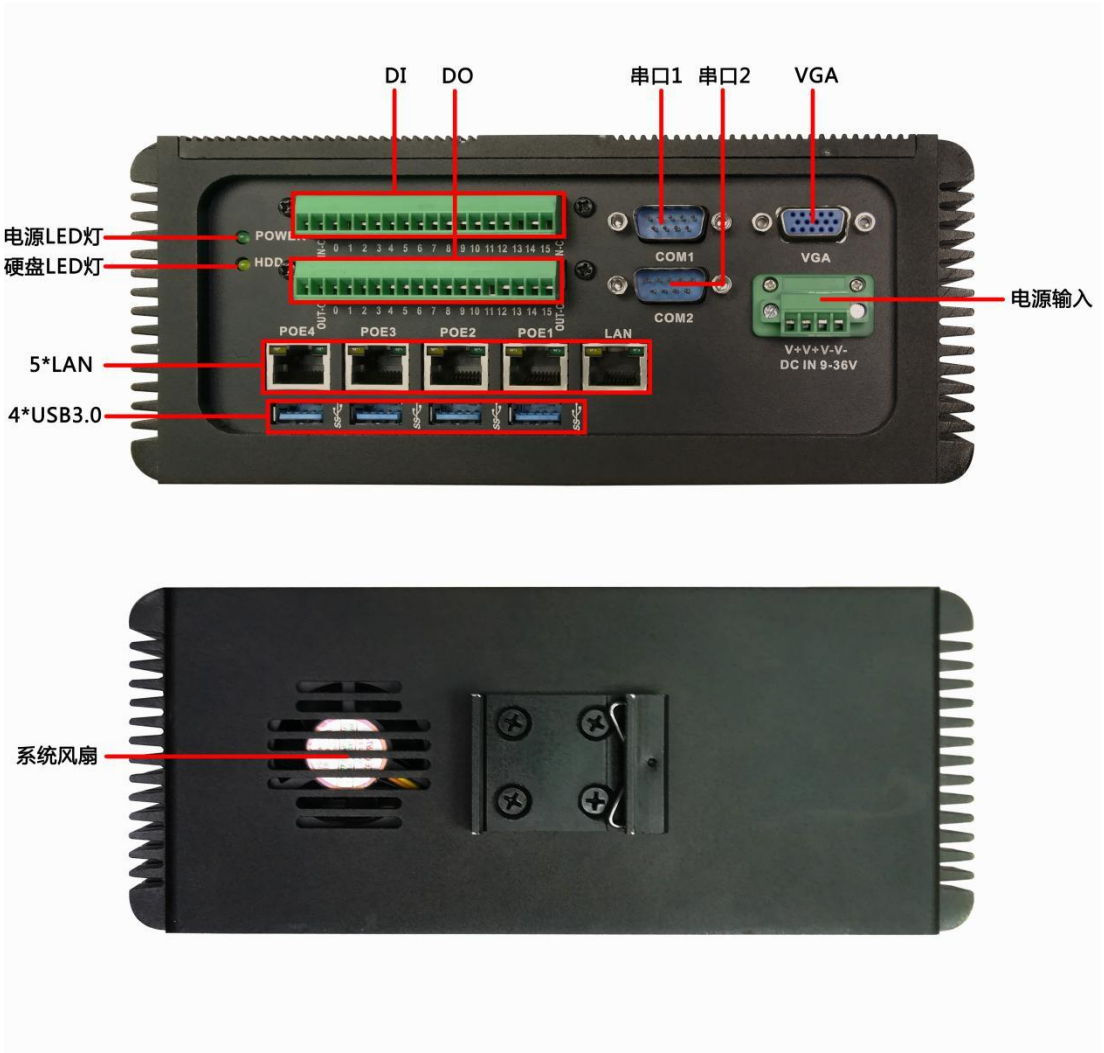
3.2 结构尺寸



注意：图中尺寸统一单位为毫米（mm）

4 接口介绍

4.1 功能接口



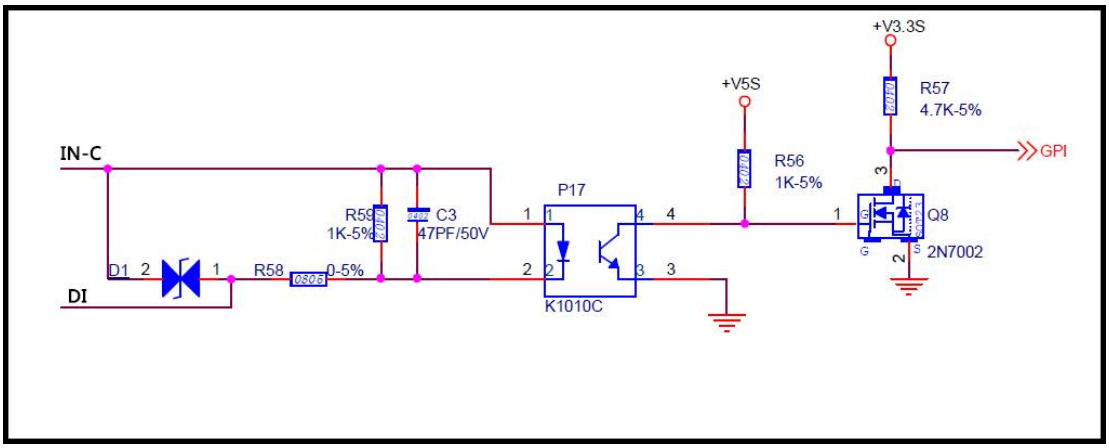
注：I5 ~ I7 CPU 可选安装系统风扇

4.2 接口描述

（1）DI 输入接口

1	2	3	4	5	6	7	8	9	10
IN-C	DI0	DI1	DI2	DI3	DI4	DI5	DI6	DI7	DI8
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**		
DI9	DI10	DI11	DI12	DI13	DI14	DI15	IN-C		

DI 接入参考电路图如下：



DI 输入的工作电压范围：12~30V

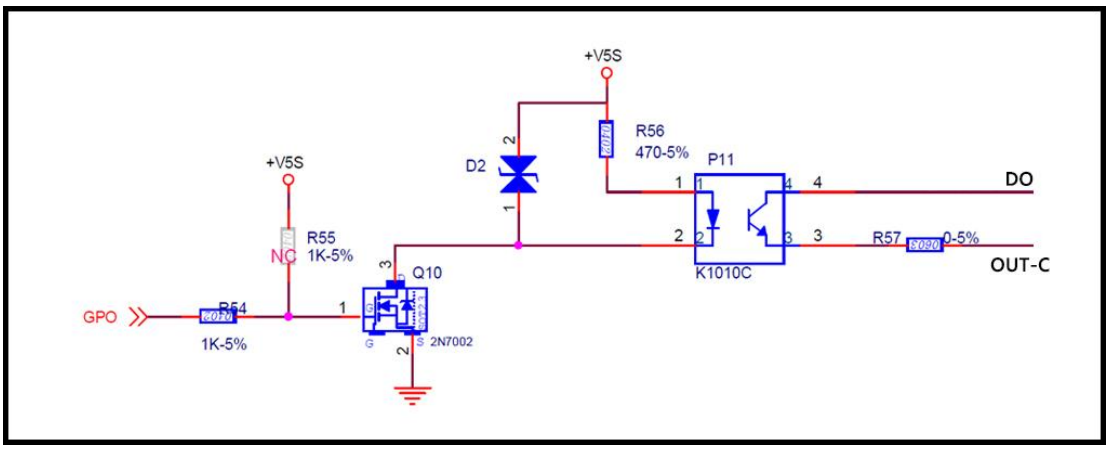
光隔离电压：直流 5000V

IN-C 引脚需接外部高电平

（2）DO 输出接口

1	2	3	4	5	6	7	8	9	10
OUT-C	DO0	DO1	DO2	DO3	DO4	DO5	DO6	DO7	DO8
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**		
DO9	DO10	DO11	DO12	DO13	DO14	DO15	OUT-C		

DO 接出参考电路图如下：



DO 最大输出电流小于 20mA

OUT-C 引脚需接地

（3）串口 COM1、COM2

COM1、COM2 为标准 RS232 9PIN，定义此处略

其中 COM2 可选 RS485，可通过 BOM 变更选择。

（4）网络接口

POE4/POE3/POE2/POE1/LAN：均为千兆 RJ45 网口，其中 POE4~POE1 支持 POE 供电输出，采用 802.1af 标准，单端口最大输出 15W。

（5）USB 接口

USB 接口均为标准 USB3.0 定义，此处略

（6）VGA 显示接口

VGA 接口均为标准 DB15 定义，此处略

（7）电源输入接口 DC-IN

1	2	3	4
V_IN	V_IN	GND	GND

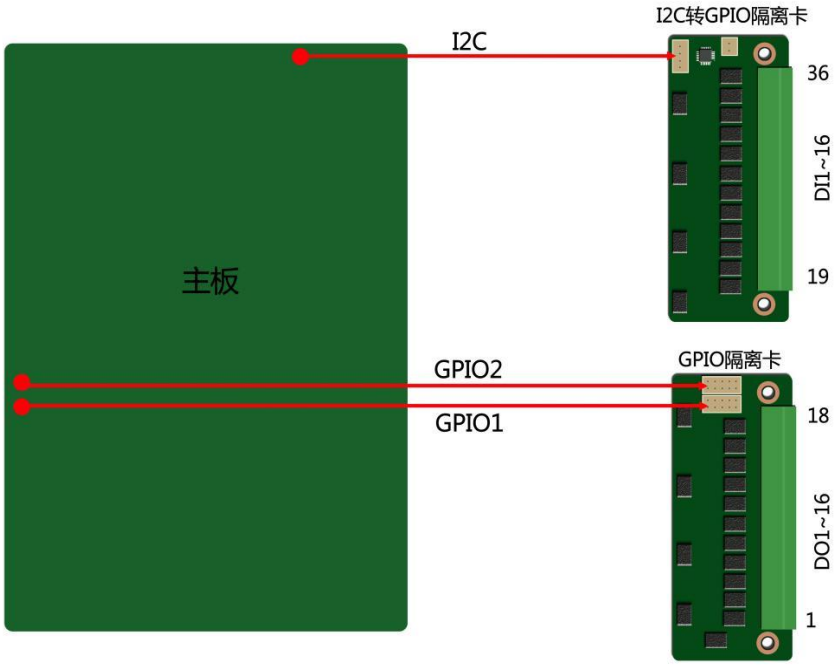
V_IN 输入电压范围：9~36V DC

防护：1.反接保护；2.过压保护；3.过流保护

***注意**：外接 POE 设备负载超过恳 20W，请使用 24V 电压供电*

5 DIO 参考例程

5.1 DIO 方案参考



DIO 方案图示

16 路 DO 接口: CPU 原生 GPIO 接口全部配置成 GPO 模式，外接光电隔离电路；

16 路 DI 接口: I2C 接口通过 GPIO 扩展 IC 得到的接口配置为 GPI 模式，外接光电隔离电路。

5.2 DI、DO 对应 GPIO 操作地址

DO 对应原生 GPIO 寄存器地址

序号	信号	寄存器地址
1	DO0	0x FDAE0450
2	DO1	0x FDAE0458
3	DO2	0x FDAE0440
4	DO3	0x FDAE0448
5	DO4	0x FDAE04A0
6	DO5	0x FDAE04A8
7	DO6	0x FDAE04B0
8	DO7	0x FDAE04B8
9	DO8	0x FDAE0460
10	DO9	0x FDAE0468
11	DO10	0x FDAE0470
12	DO11	0x FDAE0478
13	DO12	0x FDAF0490
14	DO13	0x FDAF0498
15	DO14	0x FDAF04A0
16	DO15	0x FDAF04A8

DI 对应 GPIO 转换 IC 操作地址

序号	信号	I2C 转 GPIO
1	DI0	I2C_bit0
2	DI1	I2C_bit1
3	DI2	I2C_bit2
4	DI3	I2C_bit3
5	DI4	I2C_bit4
6	DI5	I2C_bit5
7	DI6	I2C_bit6
8	DI7	I2C_bit7
9	DI8	I2C_bit8
10	DI9	I2C_bit9
11	DI10	I2C_bit10
12	DI11	I2C_bit11
13	DI12	I2C_bit12
14	DI13	I2C_bit13
15	DI14	I2C_bit14
16	DI15	I2C_bit15

5.3 DO 操作例程

处理器 GPIO 32 位寄存器说明

1，使能 GPIO

Bit10 ----写 0，使能 GPIO

2，output 模式设置

Bit[8,9]---写 01 表示设置成 GPO 模式

3，设置 GPO 的电平

bit 0----当作为 GPO 使用时，bit0 写 1，表示高电平，写 0 表示低电平

DOS 下操作例程如下：

```
/************************************************/
/*-----------sf701 change list-------------*/
//091108, w39v04 fireware flash programme;
/************************************************/

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <i86.h>
#include "io_fun.h"

#define IO_MODE_IN      0
#define IO_MODE_OUT     1
```

```
#define LEVEL_LOW      0
typedef unsigned int u32;
//Community
#define A_Community 0xAF
#define B_Community 0xAF
#define C_Community 0xAE
#define D_Community 0xAE
#define E_Community 0xAE
#define F_Community 0xAE
#define G_Community 0xAE
#define H_Community 0xAE
#define I_Community 0xAC

//Padcfgoffset
#define R_PCH_PCR_GPIO_GPP_A_PADCFG_OFFSET 0x400
#define R_PCH_PCR_GPIO_GPP_B_PADCFG_OFFSET 0x4c0
#define R_PCH_PCR_GPIO_GPP_C_PADCFG_OFFSET 0x400
#define R_PCH_PCR_GPIO_GPP_D_PADCFG_OFFSET 0x4c0
#define R_PCH_PCR_GPIO_GPP_E_PADCFG_OFFSET 0x580
#define R_PCH_H_PCR_GPIO_GPP_F_PADCFG_OFFSET 0x5e8
#define R_PCH_H_PCR_GPIO_GPP_G_PADCFG_OFFSET 0x6a8
#define R_PCH_H_PCR_GPIO_GPP_H_PADCFG_OFFSET 0x768
#define R_PCH_H_PCR_GPIO_GPP_I_PADCFG_OFFSET 0x400

#define PCH_PCR_BASE_ADDRESS 0xFD000000
#define GPIO_BASE(Community, PadCfgReg)        (PCH_PCR_BASE_ADDRESS |
((unsigned int)(Community) << 16) | (unsigned int)(PadCfgReg))

//PadCfgReg = 0x8 * PadNumber + GpioGroupInfo[GroupIndex].PadCfgOffset;
```

```
//C8 C9 C10 C11 C20 C21 C22 C23 D9 D10 D11 D12 C12 C13 C14 C15
//int port[]={8,9,10,11,22,21,20,23,9,10,11,12,12,13,14,15};    //padnumber
//int
community[]={0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE
,0xAE,0xAE,0xAE,0xAE,0xAE};
//int
padcfgoffset[]={0x400,0x400,0x400,0x400,0x400,0x400,0x400,0x400,0x4c0,0x4c0,0
x4c0,0x4c0,0x400,0x400,0x400,0x400};

int port[]={10,11,8,9,22,21,20,23,12,13,14,15,18,19,20,21};
int
community[]={0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE,0xAE
,0xAE,0xAF,0xAF,0xAF,0xAF};
int
padcfgoffset[]={0x400,0x400,0x400,0x400,0x400,0x400,0x400,0x400,0x400,0x400,
0x400,0x400,0x400,0x400,0x400,0x400};

int set_lvl(int index, int lvl);
void detect_input(int gpio_num);
void set_io_mode(int index,int mode);
void gpio_init(int gpio_num);

int main(void){
    char operate[25];
        int gpio_num=sizeof(port)/sizeof(port[0]);
        int index;
        int gpio_x;
        gpio_init(gpio_num);
```

```
while(1){
        printf("cmd>");
        scanf("%s",operate);

        /*function for input test*/
if(!strcmp(operate,"qtest")){
        printf("Please enter cmd 'i' or 'o':");
                scanf("%s",operate);
        if(!strcmp(operate,"i")){
          for(index=0;index<gpio_num;index++)
                    set_io_mode(index,IO_MODE_IN);
          printf("all gpios is in input mode!\n");
          detect_input(gpio_num);
            }
          else if(!strcmp(operate,"o"))    {
          for(index=0;index<gpio_num;index++)
                set_io_mode(index,IO_MODE_OUT);
          printf("set all gpios level(0 or 1): ");

          for(index=0;index<gpio_num;index++)
          {
            scanf("%d",&gpio_x);
                set_lvl(index,gpio_x);
            }
          }
        continue;
}
else if(!strcmp(operate,"q"))
```

```
            break;
        else
        {
        printf("Input invalid command!\n");
        continue;
        }
  wrong:
        printf("Input invalid pin num!\n");

    }
    printf("Programm exit normally!\n");
         return 0;
}

///GPIO init
void gpio_init(int gpio_num){
        int index;
        unsigned int PadCfgReg;
        unsigned int DW0;
        for(index=0;index<gpio_num;index++){
                PadCfgReg= 0x8 * port[index] + padcfgoffset[index];
//              printf("base address
is %4x\n",GPIO_BASE(community[index],PadCfgReg));
                DW0=read32(GPIO_BASE(community[index],PadCfgReg));
//              printf("value is %4x\n",DW0);
                DW0 &=0xFFFFFBFF;    //bit10 0 as gpio mode
                write32(GPIO_BASE(community[index],PadCfgReg),DW0);
//               DW0=read32(GPIO_BASE(community[index],PadCfgReg));
//               printf("value is %4x\n",DW0);
```

```
    }

}

void set_io_mode(int index,int mode){
    unsigned int DW0;
    unsigned int PadCfgReg;
    PadCfgReg= 0x8 * port[index] + padcfgoffset[index];
    DW0=read32(GPIO_BASE(community[index],PadCfgReg));
    if(mode==IO_MODE_IN){
        DW0 &=(~(1<<8 | 1<< 9));
        DW0 |=(1<<8);
    }
    else
    {
            DW0 &=(~(1<<8 | 1<< 9));
        DW0 |=(1<<9);
    }
    write32(GPIO_BASE(community[index],PadCfgReg),DW0);
}

int set_lvl(int index,int lvl)
{
    unsigned int DW0;
    unsigned int PadCfgReg;
    PadCfgReg= 0x8 * port[index] + padcfgoffset[index];
    DW0=read32(GPIO_BASE(community[index],PadCfgReg));
    if(lvl==LEVEL_LOW){
        DW0 &=0xFFFFFFFE;
```

```
    }
    else{
        DW0 |=0x00000001;
    }
    write32(GPIO_BASE(community[index],PadCfgReg),DW0);
    return 0;
}

void detect_input(int gpio_num){
    int i;
    int change[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    int olddata[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    int newdata[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    unsigned int DW0;
    unsigned int PadCfgReg;
    for(i=0;i<gpio_num;i++){
        PadCfgReg= 0x8 * port[i] + padcfgoffset[i];
        DW0=read32(GPIO_BASE(community[i],PadCfgReg));
        olddata[i]=(DW0 &0x02);
    }

    while(1){
        if(kbhit())
            if(getch()==27)    return;
            delay(50);
            for(i=0;i<gpio_num;i++){
                PadCfgReg= 0x8 * port[i] + padcfgoffset[i];
                DW0=read32(GPIO_BASE(community[i],PadCfgReg));
                newdata[i]=(DW0 &0x02);
```

```
}
for(i=0;i<gpio_num;i++){
    if(olddata[i] !=newdata[i])
        change[i]=1;
    else
        change[i]=0;
}

for(i=0;i<gpio_num;i++){
    if(change[i]==1){
        if(newdata[i]==0x02)
         {
            if(community[i]==0xAE)
              printf("GPIO-C%d input value is high\n",port[i]);
            if(community[i]==0xAF)
              printf("GPIO-A%d input value is high\n",port[i]);
         }
        else
         {
            if(community[i]==0xAE)
              printf("GPIO-C%d input value is low\n",port[i]);
            if(community[i]==0xAF)
              printf("GPIO-A%d input value is low\n",port[i]);
         }
    }

  }  //for
```

xy:

```
            for(i=0;i<gpio_num;i++){
                olddata[i]=newdata[i];
            }

        }//while
}
```

Linux 系统下 DO 操作例程如下：

```
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdbool.h>
#include <sys/io.h>
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>
#include <pthread.h>
#define     SMBUS_BASE      0xf040
#define A_GPIO_PHY_BASE 0xfdaf0000
#define C_GPIO_PHY_BASE 0xfdae0000

static unsigned int *A_GPIO_MEM_ADDR = NULL;
static unsigned int *C_GPIO_MEM_ADDR = NULL;

#define GPP_A18_ADDR      (unsigned int
```

```
*)((A_GPIO_MEM_ADDR)==NULL ?(A_GPIO_MEM_ADDR) :
(A_GPIO_MEM_ADDR)+18*2)
#define GPP_A19_ADDR      (unsigned int
*)((A_GPIO_MEM_ADDR)==NULL ?(A_GPIO_MEM_ADDR) :
(A_GPIO_MEM_ADDR)+19*2)
#define GPP_A20_ADDR      (unsigned int
*)((A_GPIO_MEM_ADDR)==NULL ?(A_GPIO_MEM_ADDR) :
(A_GPIO_MEM_ADDR)+20*2)
#define GPP_A21_ADDR      (unsigned int
*)((A_GPIO_MEM_ADDR)==NULL ?(A_GPIO_MEM_ADDR) :
(A_GPIO_MEM_ADDR)+21*2)

#define GPP_C8_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+8*2)
#define GPP_C9_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+9*2)
#define GPP_C10_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+10*2)
#define GPP_C11_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+11*2)
#define GPP_C12_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+12*2)
#define GPP_C13_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
```

```
(C_GPIO_MEM_ADDR)+13*2)
#define GPP_C14_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+14*2)
#define GPP_C15_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+15*2)
#define GPP_C20_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+20*2)
#define GPP_C21_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+21*2)
#define GPP_C22_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+22*2)
#define GPP_C23_ADDR      (unsigned int
*)((C_GPIO_MEM_ADDR)==NULL ?(C_GPIO_MEM_ADDR) :
(C_GPIO_MEM_ADDR)+23*2)

#define CQ_OUT0_ADDR      GPP_C10_ADDR
#define CQ_OUT1_ADDR      GPP_C11_ADDR
#define CQ_OUT2_ADDR      GPP_C8_ADDR
#define CQ_OUT3_ADDR      GPP_C9_ADDR
#define CQ_OUT4_ADDR      GPP_C20_ADDR
#define CQ_OUT5_ADDR      GPP_C21_ADDR
#define CQ_OUT6_ADDR      GPP_C22_ADDR
#define CQ_OUT7_ADDR      GPP_C23_ADDR
```

```
#define CQ_OUT8_ADDR       GPP_C12_ADDR
#define CQ_OUT9_ADDR       GPP_C13_ADDR
#define CQ_OUT10_ADDR      GPP_C14_ADDR
#define CQ_OUT11_ADDR      GPP_C15_ADDR
#define CQ_OUT12_ADDR      GPP_A18_ADDR
#define CQ_OUT13_ADDR      GPP_A19_ADDR
#define CQ_OUT14_ADDR      GPP_A20_ADDR
#define CQ_OUT15_ADDR      GPP_A21_ADDR

//set all Pins as GPIO function
//set all GPIO as GPO fucntion
//set all GPO output low level
void Init_Gpio(void)
{
        unsigned int val;

      val = *(CQ_OUT0_ADDR);
      val &= ~(0x1<<10); //bit10 set 0 gpio mode

      val &= ~(0x3<<8);
      val |= (0x02<<8);      //output mode
      val &= ~(0x3);           //low level
      *(CQ_OUT0_ADDR) = val;

      val = *(CQ_OUT1_ADDR);
      val &= ~(0x1<<10); //bit10 set 0 gpio mode

      val &= ~(0x3<<8);
      val |= (0x02<<8);      //output mode
```

```
val &= ~(0x3);          //low level
*(CQ_OUT1_ADDR) = val;


val = *(CQ_OUT2_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);     //output mode
val &= ~(0x3);          //low level
*(CQ_OUT2_ADDR) = val;


val = *(CQ_OUT3_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);     //output mode
val &= ~(0x3);          //low level
*(CQ_OUT3_ADDR) = val;


val = *(CQ_OUT4_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);     //output mode
val &= ~(0x3);          //low level
*(CQ_OUT4_ADDR) = val;
```

```
val = *(CQ_OUT5_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT5_ADDR) = val;


val = *(CQ_OUT6_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT6_ADDR) = val;


val = *(CQ_OUT7_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT7_ADDR) = val;

val = *(CQ_OUT8_ADDR);
```

```
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT8_ADDR) = val;

val = *(CQ_OUT9_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT9_ADDR) = val;

val = *(CQ_OUT10_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT10_ADDR) = val;

val = *(CQ_OUT11_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
```

```
*(CQ_OUT11_ADDR) = val;

val = *(CQ_OUT12_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT12_ADDR) = val;

val = *(CQ_OUT13_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT13_ADDR) = val;

val = *(CQ_OUT14_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode

val &= ~(0x3<<8);
val |= (0x02<<8);      //output mode
val &= ~(0x3);          //low level
*(CQ_OUT14_ADDR) = val;

val = *(CQ_OUT15_ADDR);
val &= ~(0x1<<10); //bit10 set 0 gpio mode
```

```
        val &= ~(0x3<<8);
        val |= (0x02<<8);      //output mode
        val &= ~(0x3);           //low level
        *(CQ_OUT15_ADDR) = val;
}

void set_lvl(int portNum,int level)
{
          unsigned int val;

          switch(portNum){
        case 0:
        {
             val = *(CQ_OUT0_ADDR);
             val &= ~(0x1);
             if(level)
                  val |= 0x1;
            *(CQ_OUT0_ADDR) = val;
        }
          break;
        case 1:
        {
             val = *(CQ_OUT1_ADDR);
             val &= ~(0x1);
             if(level)
                  val |= 0x1;
            *(CQ_OUT1_ADDR) = val;
        }
          break;
```

```
case 2:
{
    val = *(CQ_OUT2_ADDR);
    val &= ~(0x1);
    if(level)
        val |= 0x1;
   *(CQ_OUT2_ADDR) = val;
}
 break;
case 3:
{
    val = *(CQ_OUT3_ADDR);
    val &= ~(0x1);
    if(level)
        val |= 0x1;
   *(CQ_OUT3_ADDR) = val;
}
 break;
case 4:
{
    val = *(CQ_OUT4_ADDR);
    val &= ~(0x1);
    if(level)
        val |= 0x1;
   *(CQ_OUT4_ADDR) = val;
}
 break;
case 5:
{
```

```
        val = *(CQ_OUT5_ADDR);
        val &= ~(0x1);
        if(level)
            val |= 0x1;
      *(CQ_OUT5_ADDR) = val;
}
  break;
case 6:
{
        val = *(CQ_OUT6_ADDR);
        val &= ~(0x1);
        if(level)
            val |= 0x1;
      *(CQ_OUT6_ADDR) = val;
}
  break;
case 7:
{
        val = *(CQ_OUT7_ADDR);
        val &= ~(0x1);
        if(level)
            val |= 0x1;
      *(CQ_OUT7_ADDR) = val;
}
  break;
case 8:
{
        val = *(CQ_OUT8_ADDR);
        val &= ~(0x1);
```

```
        if(level)
            val |= 0x1;
       *(CQ_OUT8_ADDR) = val;
    }
     break;
    case 9:
    {
        val = *(CQ_OUT9_ADDR);
        val &= ~(0x1);
        if(level)
            val |= 0x1;
       *(CQ_OUT9_ADDR) = val;
    }
     break;
    case 10:
    {
        val = *(CQ_OUT10_ADDR);
        val &= ~(0x1);
        if(level)
            val |= 0x1;
       *(CQ_OUT10_ADDR) = val;
    }
     break;
    case 11:
    {
        val = *(CQ_OUT11_ADDR);
        val &= ~(0x1);
        if(level)
            val |= 0x1;
```

```
    *(CQ_OUT11_ADDR) = val;
}
 break;
case 12:
{
     val = *(CQ_OUT12_ADDR);
     val &= ~(0x1);
     if(level)
          val |= 0x1;
    *(CQ_OUT12_ADDR) = val;
}
 break;
case 13:
{
     val = *(CQ_OUT13_ADDR);
     val &= ~(0x1);
     if(level)
          val |= 0x1;
    *(CQ_OUT13_ADDR) = val;
}
 break;
case 14:
{
     val = *(CQ_OUT14_ADDR);
     val &= ~(0x1);
     if(level)
          val |= 0x1;
    *(CQ_OUT14_ADDR) = val;
}
```

```
        break;
      case 15:
      {
          val = *(CQ_OUT15_ADDR);
          val &= ~(0x1);
          if(level)
              val |= 0x1;
         *(CQ_OUT15_ADDR) = val;
      }
        break;
      default:
            break;
   }
}

void Test_GPO(void)
{
     int portNum;
   int gpo_level;
   for(portNum=0;portNum<16;portNum++)
   {
     printf("set gpo pin%d level(0 or 1): \n",portNum);
     scanf("%d",&gpo_level);
     set_lvl(portNum,gpo_level);
   }
}

void main(void)
{
```

```
    int fd;
    unsigned int *add1;
    unsigned int *add2;


  if (0 != iopl(3))
  {
        printf("gpio must run in root mode.\n");
        return;
  }

  fd = open("/dev/mem",O_RDWR|O_SYNC);
  if(fd<0)
  {
        printf("cannot open /dev/mem \n");
        return;
  }

  add1 = (unsigned int *)mmap(NULL,0x1000 & ~(sysconf(_SC_PAGE_SIZE) -
1),PROT_READ|PROT_WRITE,MAP_SHARED,fd,C_GPIO_PHY_BASE);
  add2 = (unsigned int *)mmap(NULL,0x1000 & ~(sysconf(_SC_PAGE_SIZE) -
1),PROT_READ|PROT_WRITE,MAP_SHARED,fd,A_GPIO_PHY_BASE);

  if(MAP_FAILED==add1)
  {
        perror("mmap faild:");
        printf("mmap faild, addr1=%x addr2=%x\n",add1,add2);
        return;
  }
```

```
    printf("mmap successful, addr1=%x addr2=%x\n",add1,add2);

    C_GPIO_MEM_ADDR = add1+0x100;
    A_GPIO_MEM_ADDR = add2+0x100;

    Init_Gpio();
    Test_GPO();
    printf("Soc GPIO Test End!!\n");
}
```

5.4 DI 操作例程

I2C GPIO 操作说明

I2C IC： TCA9555

Slave Addr： 0x40

SMBUS_BASE: 0xF040

访问方式： IO 访问

IC 寄存器说明：

IC 的 PD0—7 作为 GPO

IC 的 PD8—15 作为 GPI

0x00—0x01: input port register

读取 GPI 的电平状态 bit0 对应 PD0 …bit15 对应 DP15

0x02—0x03: output port register

设置 GPO 的电平状态 bit0 对应 PD0 …bit15 对应 DP15

1—表示高电平，0—表示低电平

0x04—0x05: Polarity Inversin register

反转寄存器。bit0 对应 PD0 …bit15 对应 DP15

置 1 时，对应的 GPIO 极性会反转。

bit0 对应 PD0 …bit15 对应 DP15

0x06---0x07: Configuration registers

配置作为 GPI or GPO 模式。bit0 对应 PD0 …bit15 对应 DP15

1—表示 GPI， 0—表示 GPO

GPI： 0 1 2 3 4 5 6 7 (IC: PD8 9 10 11 12 13 14 15)

GPO：0 1 2 3 4 5 6 7 (IC: PD0 1 2 3 4 5 6 7)

操作函数：

Write_Smbus(int command,int data)	往 TCA9555 IC 的 寄存器写数据 command: tca9555 寄存器 Data: 要写的数据
Read_Smbus(int command)	读取 TCA9555 IC 寄存器的值

1，配置 GPIO

//config PD0--7 as GPO

//set GPO Level high as Default

```
Write_Smbus(0x06,0x00);
Write_Smbus(0x04,0x00);
Write_Smbus(0x02,0xFF);
```

//config PD8--15 as GPI

//invert GPI

```
Write_Smbus(0x07,0xFF);
Write_Smbus(0x05,0xFF); //GPI 需要反转
```

2，设置 DP0—7 都为低电平

```
Write_Smbus(0x02,0x00);
```

3，读取 DP8—15 的电平状态

```
Read_Smbus(0x01);
```

```
void Write_Smbus(int command,int data)
{
        int status;
//clear smubs clear status
    outportb(SMBUS_BASE+0x00,0xFF);

    outportb(SMBUS_BASE+0x04,0x40);
    outportb(SMBUS_BASE+0x03,command);
    outportb(SMBUS_BASE+0x05,data);
    outportb(SMBUS_BASE+0x02,0x48);
    while(1)
    {
        status=inportb(SMBUS_BASE+0x00);
        status &=0x02;
        if(status==0x02) break;
      }

    outportb(SMBUS_BASE+0x00,0xFF);

}
int Read_Smbus(int command)
{
            int status;
            int data;
//clear smubs clear status
    outportb(SMBUS_BASE+0x00,0xFF);

    outportb(SMBUS_BASE+0x04,0x41);
    outportb(SMBUS_BASE+0x03,command);
```

```
    outportb(SMBUS_BASE+0x02,0x48);
    while(1)
    {
        status=inportb(SMBUS_BASE+0x00);
        status &=0x02;
        if(status==0x02) break;
      }

    data=inportb(SMBUS_BASE+0x05);
    //clear smubs clear status
    outportb(SMBUS_BASE+0x00,0xFF);
    return data;
}
```

DOS 下 DI 操作例程如下：

```
/*************************************************/
/*************************************************/
/******ACT9555 i2c to GPIO chip
/*************************************************/

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <i86.h>
```

```
#include "io_fun.h"


//PD0--7 as GPO
//PD8--15 as GPI

#define IO_MODE_IN      0
#define IO_MODE_OUT     1

#define LEVEL_LOW      0

#define port_mask(x)    (1<<(x))

#define SLAVE_ADDR 0x40;
#define SMBUS_BASE 0xF040                       //SMBUS BaseAddress

typedef unsigned int u32;

int set_lvl(int lvl);
void detect_input(void);
void gpio_init(void);

void Write_Smbus(int command,int data);
int Read_Smbus(int command);


int main(){
    char operate[25];
    int gpio_x;
```

```
/****set all pin as GPIO function****/
printf("GPIO_INIT\n");
gpio_init();

while(1){
      printf("cmd>");
      scanf("%s",operate);

      /*function for input test*/
if(!strcmp(operate,"qtest")){
      printf("Please enter cmd 'i' or 'o':");
            scanf("%s",operate);
      if(!strcmp(operate,"i")){
        detect_input();
          }
         else if(!strcmp(operate,"o"))    {
            printf("set all GPO level(0 or 1): ");
             scanf("%d",&gpio_x);
               set_lvl(gpio_x);
          }
       continue;
}
else if(!strcmp(operate,"q"))
      break;
else
{
printf("Input invalid command!\n");
continue;
}
```

```
  wrong:
      printf("Input invalid pin num!\n");

   }
   printf("Programm exit normally!\n");
       return 0;
}

void detect_input(void){
      int change,newdata,olddata;
      int i=0;

      olddata=Read_Smbus(0x01);
      while(1){
            if(kbhit())
                  if(getch()==27)    return;
                  delay(50);
                  newdata=Read_Smbus(0x01);
                  change=(olddata^newdata);
                  if(change){
                        for(i=0;i<8;i++){
                              if((change & (1<<i)))
                              {
                                    printf("GPI-%d input value
is %d\n",i,((newdata&(1<<i)))>>i);
                                    goto xy;
                              }
                        }  //for
xy:
```

```
                    ;

            }// if
            olddata=newdata;

      }//while
}

int set_lvl(int lvl)
{
      if(lvl==LEVEL_LOW){
       Write_Smbus(0x02,0x00);
      }
      else{
        Write_Smbus(0x02,0xFF);
      }
      return 0;
}

void gpio_init(void){
//init act9555 chip
//config PD0--7 as GPO
//set GPO Level high as Default
  Write_Smbus(0x06,0x00);
  Write_Smbus(0x04,0x00);
  Write_Smbus(0x02,0xFF);

//config PD8--15 as GPI
//invert GPI
```

```
  Write_Smbus(0x07,0xFF);
  Write_Smbus(0x05,0xFF);
}

int Read_Smbus(int command)
{
        int status;
        int data;
//clear smubs clear status
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                        //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                        //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                        //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                        //IO delay

    outportb(SMBUS_BASE+0x04,0x41);
    outportb(0xED,0xFF);                        //IO delay
    outportb(SMBUS_BASE+0x03,command);
    outportb(0xED,0xFF);                        //IO delay
    outportb(SMBUS_BASE+0x02,0x48);
    outportb(0xED,0xFF);                        //IO delay
    while(1)
    {
        status=inportb(SMBUS_BASE+0x00);
        outportb(0xED,0xFF);                    //IO delay
        status &=0x02;
```

```
            if(status==0x02) break;
        }

    data=inportb(SMBUS_BASE+0x05);
    //clear smubs clear status
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay

    return data;
}

void Write_Smbus(int command,int data)
{
        int status;
//clear smubs clear status
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                    //IO delay
```

```
    outportb(SMBUS_BASE+0x04,0x40);
    outportb(0xED,0xFF);                                    //IO delay
    outportb(SMBUS_BASE+0x03,command);
    outportb(0xED,0xFF);                                    //IO delay
    outportb(SMBUS_BASE+0x05,data);
    outportb(0xED,0xFF);                                    //IO delay
    outportb(SMBUS_BASE+0x02,0x48);
    outportb(0xED,0xFF);                                    //IO delay
    while(1)
    {
        status=inportb(SMBUS_BASE+0x00);
        outportb(0xED,0xFF);                                //IO delay
        status &=0x02;
        if(status==0x02) break;
      }

//clear smubs clear status
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                                    //IO delay
    outportb(SMBUS_BASE+0x00,0xFF);
    outportb(0xED,0xFF);                                    //IO delay

}
```

Linux 下 DI 操作例程如下：

```
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdbool.h>
#include <sys/io.h>
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>
#include <pthread.h>
#define     SMBUS_BASE     0xf040


static void Write_Smbus(int command,int data)
{
        int status; //clear smubs clear status
      int timeout = 100;
      outb(0xff,SMBUS_BASE+0x00);
      outb(0xff,SMBUS_BASE+0x00);
      outb(0x40,SMBUS_BASE+0x04);
      outb(command,SMBUS_BASE+0x03);
      outb(data,SMBUS_BASE+0x05);
      outb(0x48,SMBUS_BASE+0x02);

      while(timeout--)
      {
            status = inb(SMBUS_BASE+0x00);
```

```
        status &= 0x02;
        if(status==0x02)
            break;
        usleep(1);
    }
    outb(0xff,SMBUS_BASE+0x00);
    outb(0xff,SMBUS_BASE+0x00);
}

static int Read_Smbus(int command)
{
     int status; //clear smubs clear status
    int timeout = 100;
    outb(0xff,SMBUS_BASE+0x00);
    outb(0xff,SMBUS_BASE+0x00);
    outb(0x41,SMBUS_BASE+0x04);
    outb(command,SMBUS_BASE+0x03);
    outb(0x48,SMBUS_BASE+0x02);

    while(timeout--)
    {
        status = inb(SMBUS_BASE+0x00);
        status &= 0x02;
        if(status==0x02)
            break;
        usleep(1);
    }
    outb(0xff,SMBUS_BASE+0x00);
    outb(0xff,SMBUS_BASE+0x00);
```

```
        int data = inb(SMBUS_BASE+0x05);
        return data;
}

void detect_input()
{
       int change,newdata,olddata;
       int temp1;
       int temp2;
       int i=0;
       temp1=Read_Smbus(0x01);
    temp2=Read_Smbus(0x00);
       olddata=(temp1*0x100 +temp2);
       while(1)
       {
             usleep(500);
           temp1=Read_Smbus(0x01);
             temp2=Read_Smbus(0x00);
             newdata=(temp1*0x100 +temp2);
             change=(olddata^newdata);
             if(change)
             {
                   for(i=0;i<16;i++)
                   {
                         if((change & (1<<i)))
                           {
                             printf("GPI-%d input value
is %d\n",i,((newdata&(1<<i)))>>i);
```

```
                                goto xy;
                               }
                        }    //for
xy:
                        ;

            }// if
            olddata=newdata;
        }
}

void gpio_init()
{
      //init act9555 chip
//config PD0--7 as GPI
  Write_Smbus(0x06,0xFF);
  Write_Smbus(0x07,0xFF);
}

void main()
{
          if (0 != iopl(3))
       {
              printf("gpio must run in root mode.\n");
              return;
       }
      gpio_init();
      printf("please set the GPI high or low\n");
      detect_input();
```

```
}
```

6 BIOS 设置

在开机运行时，按下键盘上的＜F2＞键即可进入 BIOS 设定程序
设置结束后，需按 F10 或者通过 <Save & Exit>中的保存选项，当前设置才能生效

6.1 日期和时间设置

当你进入 BIOS 的设定界面时，所出现的第一个界面就可以设定日期和时间，如下所示：

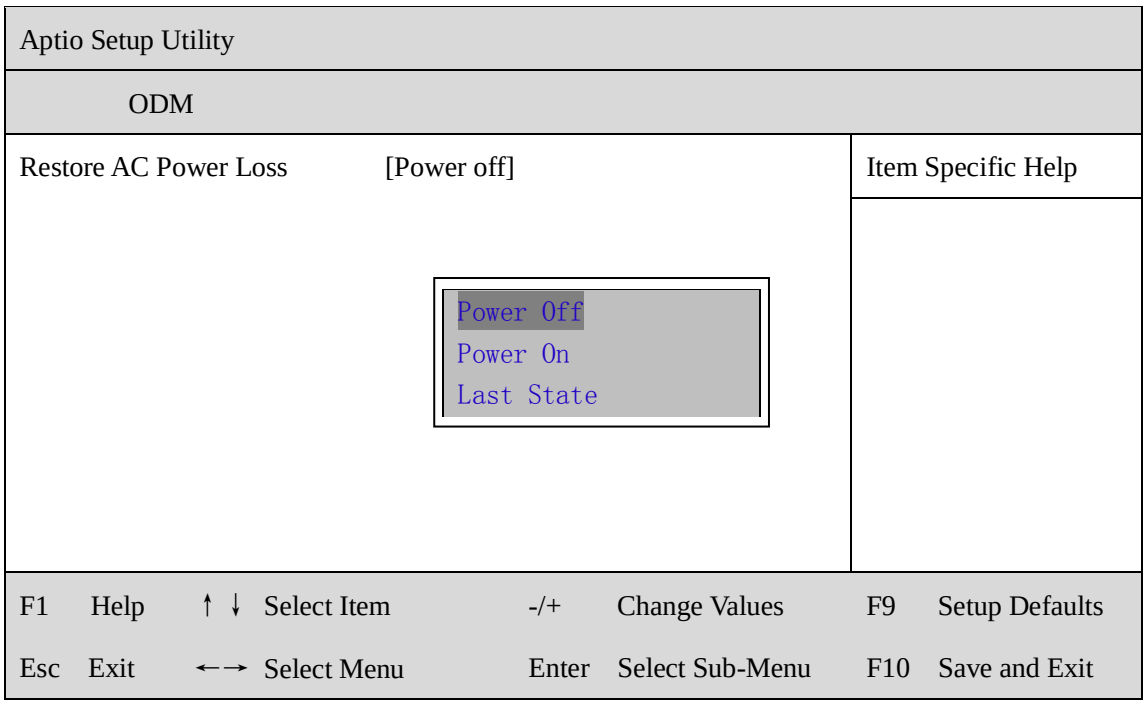
Aptio Setup Utility								
Main	ODM	Advanced	Chipset	Boot	Security	Save & Exit		
							Item Specific Help	
System Language:			[English]					
System Time:			[10:50:34]					
System Date:			[01/01/2012]					
F1	Help	↑ ↓	Select Item	-/+	Change Values		F9	Setup Defaults
Esc	Exit	←→	Select Menu	Enter	Select Sub-Menu		F10	Save and Exit

System Time：设置时间；
System Date：设置日期。

6.2 ODM 常用功能设置

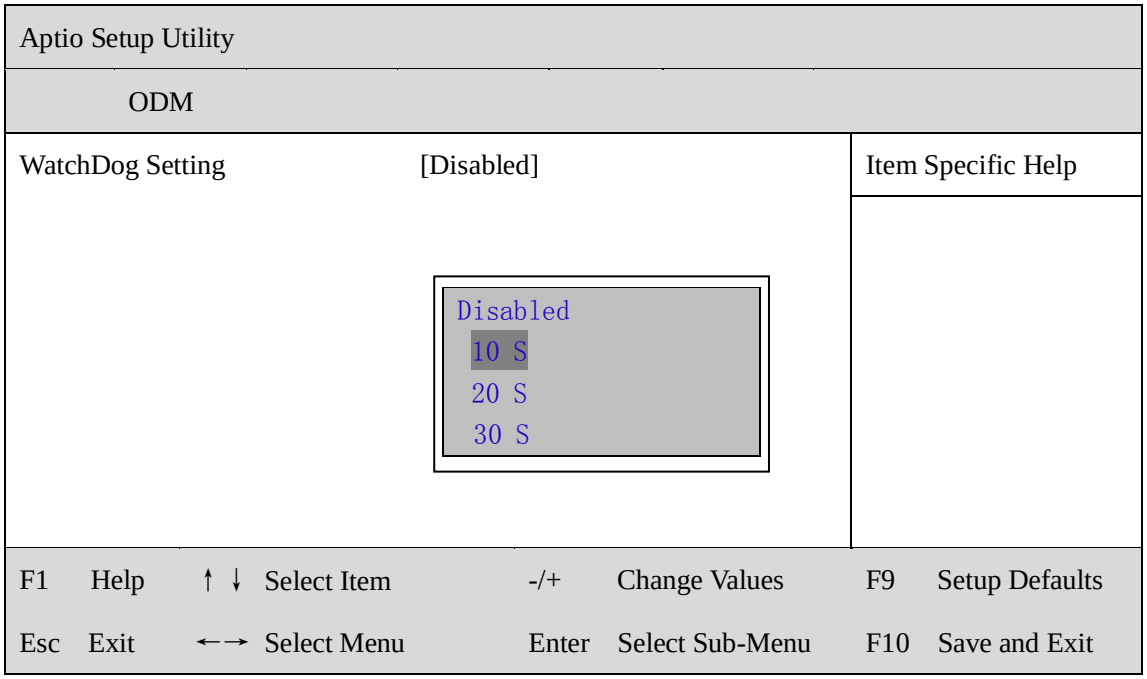
1.来电开机设置

进入 BIOS 设置界面，选择<ODM> → <AC Power Loss Setting>，对选项进行设置，选择 “Power ON”则启动来电开机功能，改为“Power Off”，则关闭来电开机功能。



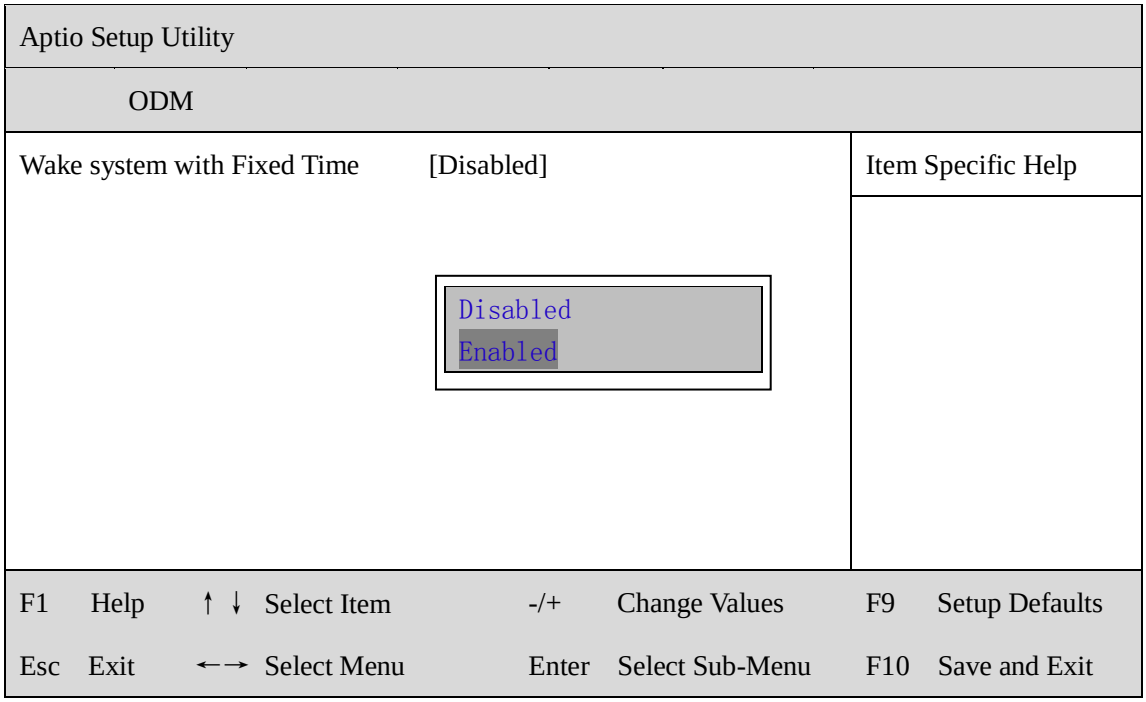
2.看门狗设置

进入 BIOS 设置界面，选择<ODM> → <Watchdog Setting>，根据自己的需要，对<Watchdog Setting>选项进行相关设置，如下图所示：

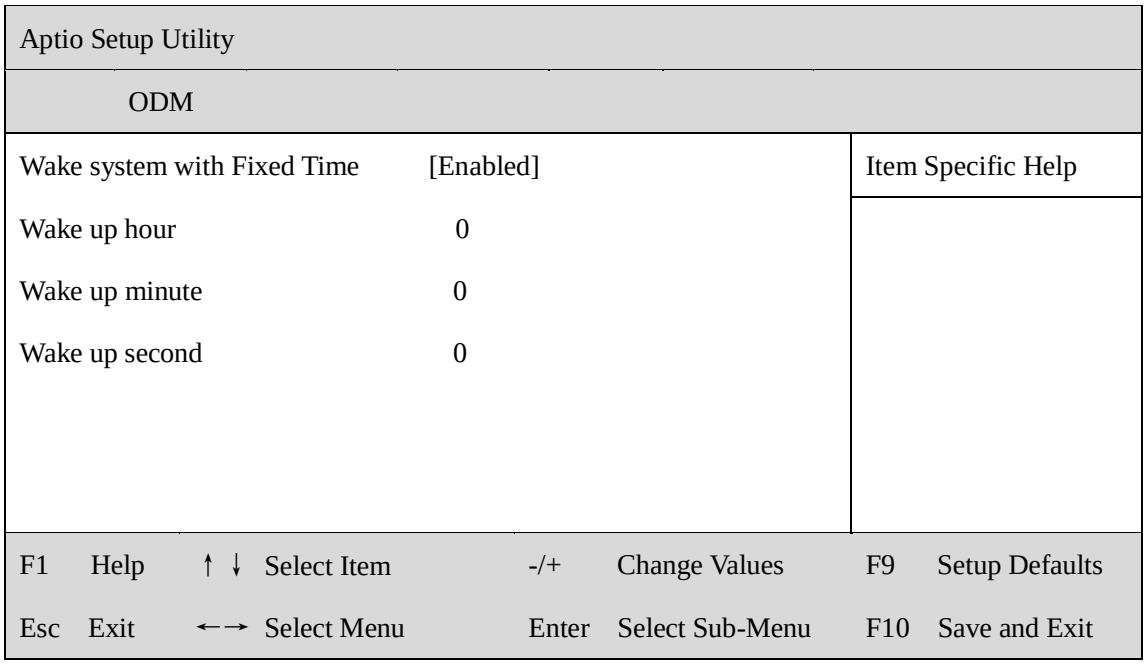


3.定时开机功能

进入 BIOS 设置界面，选择<ODM> → <S5 RTC Wake Setting> → <Wake system with Fixed Time>选项，将默认值设置为“Enable”之后，可根据自己的需要，设置定时开机时间，如下图所示：



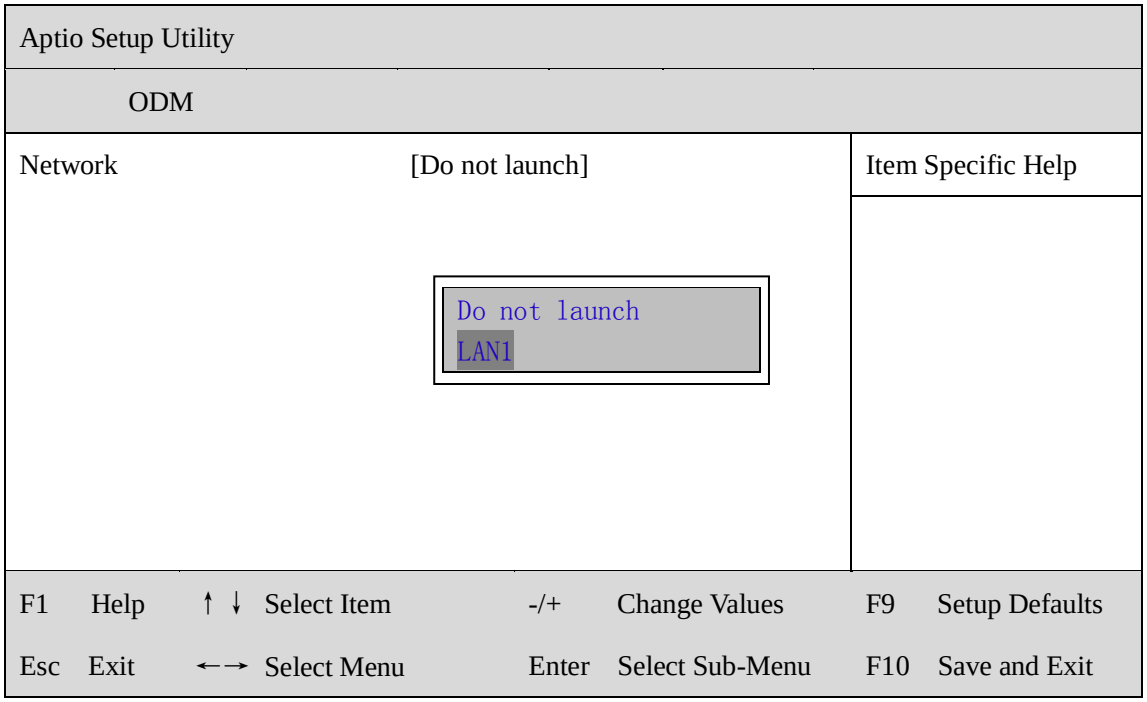
分别设置开机的时/分/秒，如 8:30:00



备注：设定 ok 后，表示每天这个时间，主板会自动开机

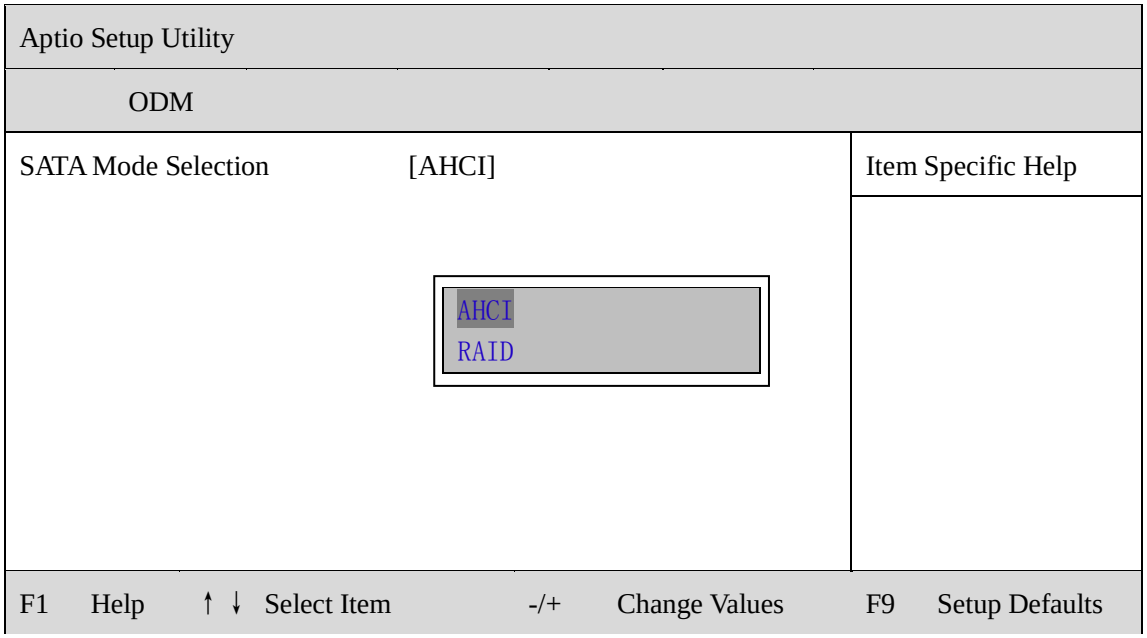
4.PXE 启动功能（无盘启动）

进入 BIOS 设置界面，选择<ODM> → <LAN PXE Setting> → <Network>选项，将默认值改成“LAN1”，完成 PXE 启动功能设置，如下所示：



5.SATA HDD 模式选择

进入 BIOS 设置界面中，选择<ODM> → <SATA Model Setting> ，对<SATA Mode Selection>项进行设置，如下所示：

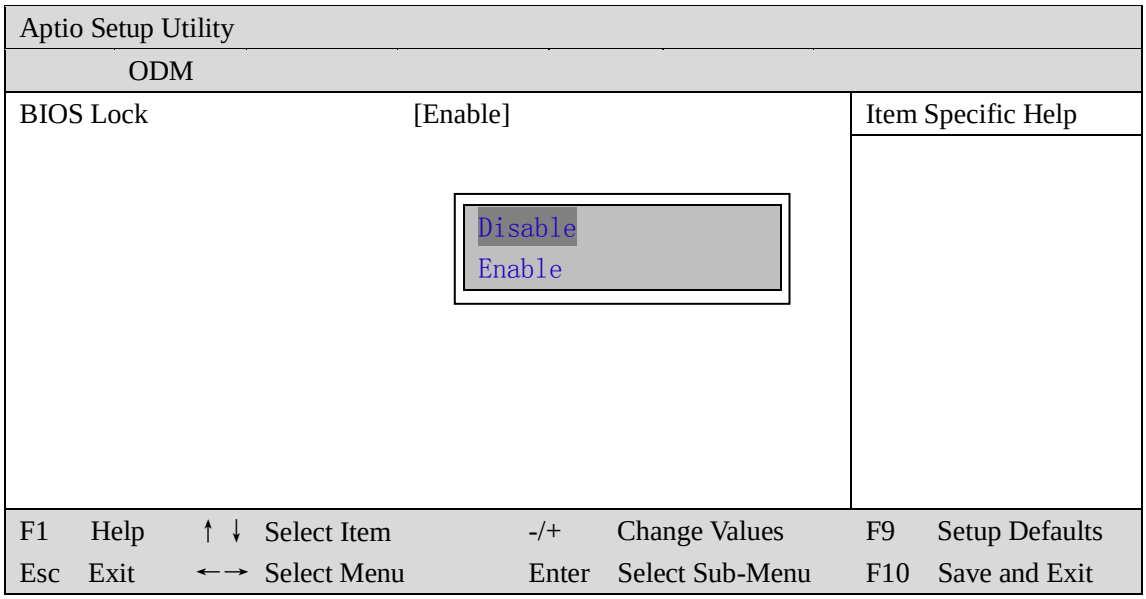


Esc	Exit	←→	Select Menu	Enter	Select Sub-Menu	F10	Save and Exit

6.bios 刷写关闭 bios 写保护功能

更新 bios 前，需先把 bios 写保护功能选项关闭才能执行，具体是：

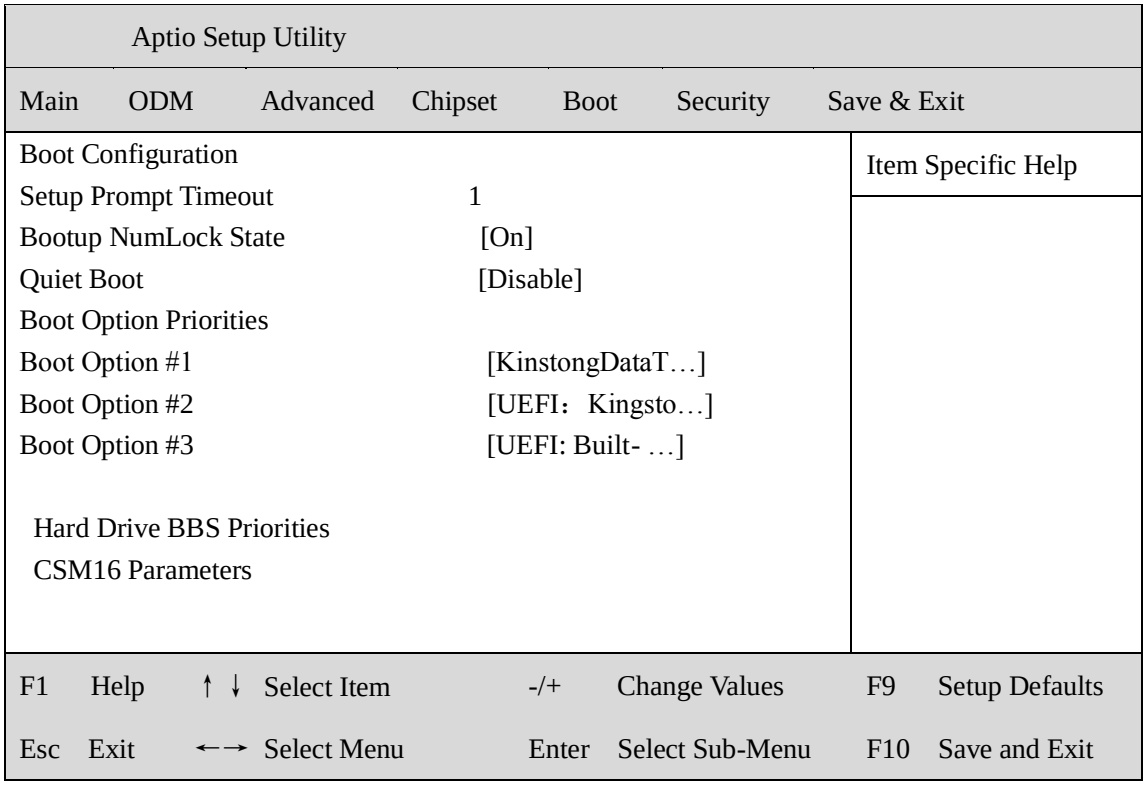
进入 BIOS 设置界面，选择<ODM> → <Special Setting> → < BIOS Lock>选项，将此选项设置为“Disable”，如下所示：



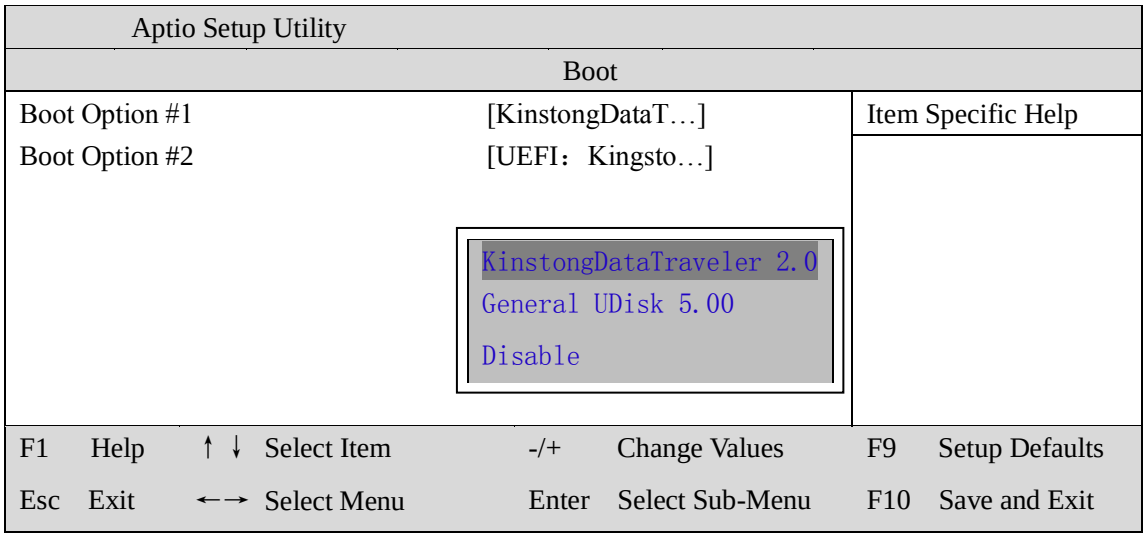
6.3 其他功能设置

1.boot 设置功能

进入 BIOS 设置界面中，选择<boot>选项，进入后，设置需要的启动顺序，如下所示：



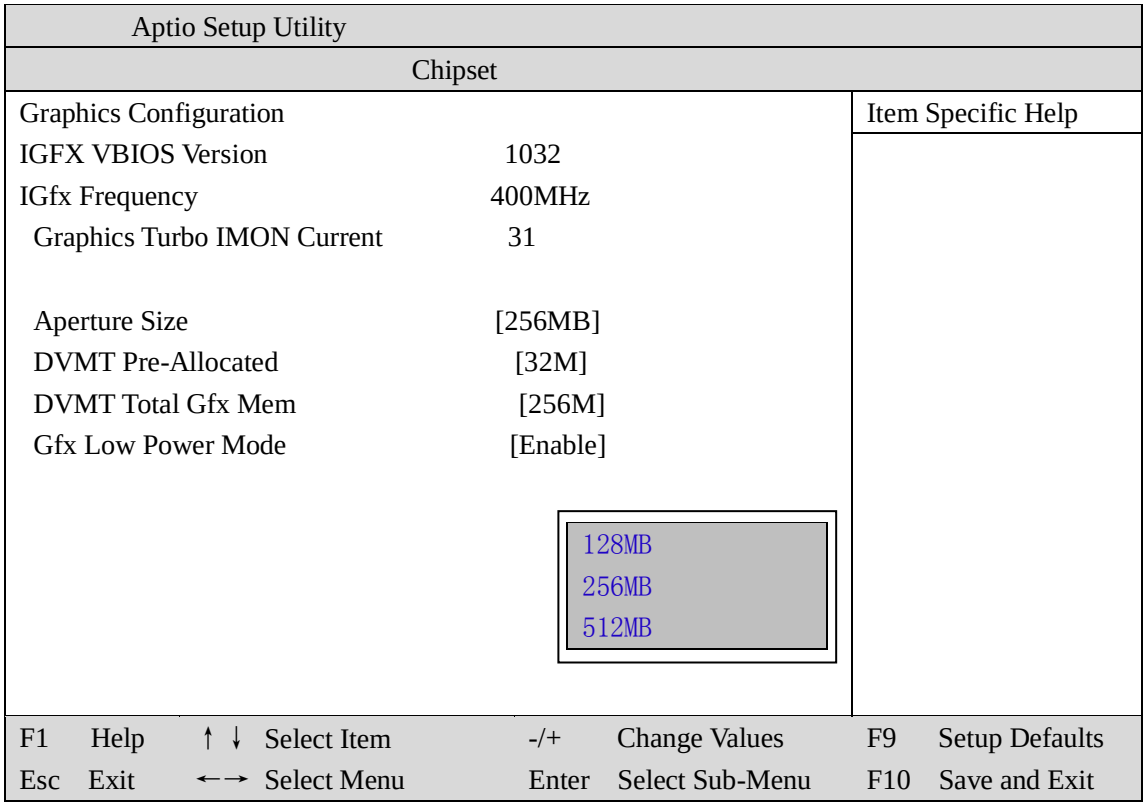
选择<Hard Driver BBS Priorities> → <Boot Option #1>，设置 Boot 启动首选项。



备注：可依次对后续选项进行设置，设定启动优先级顺序。

2.共享内存设置功能

进入 BIOS 设置界面，选择<Chipset> → <Systems Agent Configuration> → <Graphics Configuration>，进入后，设置 DVMT 功能，如下所示：



备注：选择“MAX”，最大可共享 1GB（使用的内存容量为 2GB 以上时）的内存用于图形处理

3.温度、电压和 FAN 转速侦测

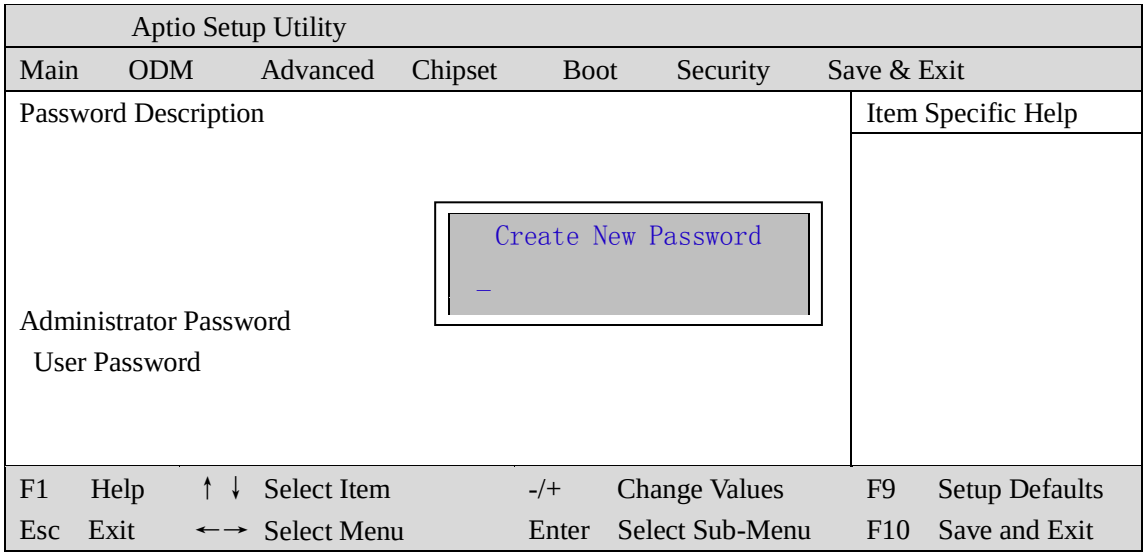
进入 BIOS 的 CMOS 置界面后，按选择<Advanced> → <Hardware Monitor>，进入此界面，可以看相关侦测值，如下所示：

Aptio Setup Utility						
Advanced						
Pc Health Status					Item Specific Help	
CPU Temp	: -51					
System Temp	: +37					
SYS_FAN Speed	: 2000 RPM					
VCORE	: +1.716 V					
+12V	: +12.60 V					
+ 5V	: +4.980 V					
VDIMM	: +1.210 V					
VSB3	: +3.264 V					
F1 Help	↑ ↓ Select Item	-/+	Change Values		F9	Setup Defaults
Esc Exit	←→ Select Menu	Enter	Select Sub-Menu		F10	Save and Exit

备注：此 bios 不显示 CPU 温度，显示 CPU 温度控制值（把 CPU 承受的最高温度值设为 0），控制值为显示数，是表示离 CPU 最高承受值的差值，如上图所示-51，表示离 CPU 最高承受温度（100℃）还有 51 度：

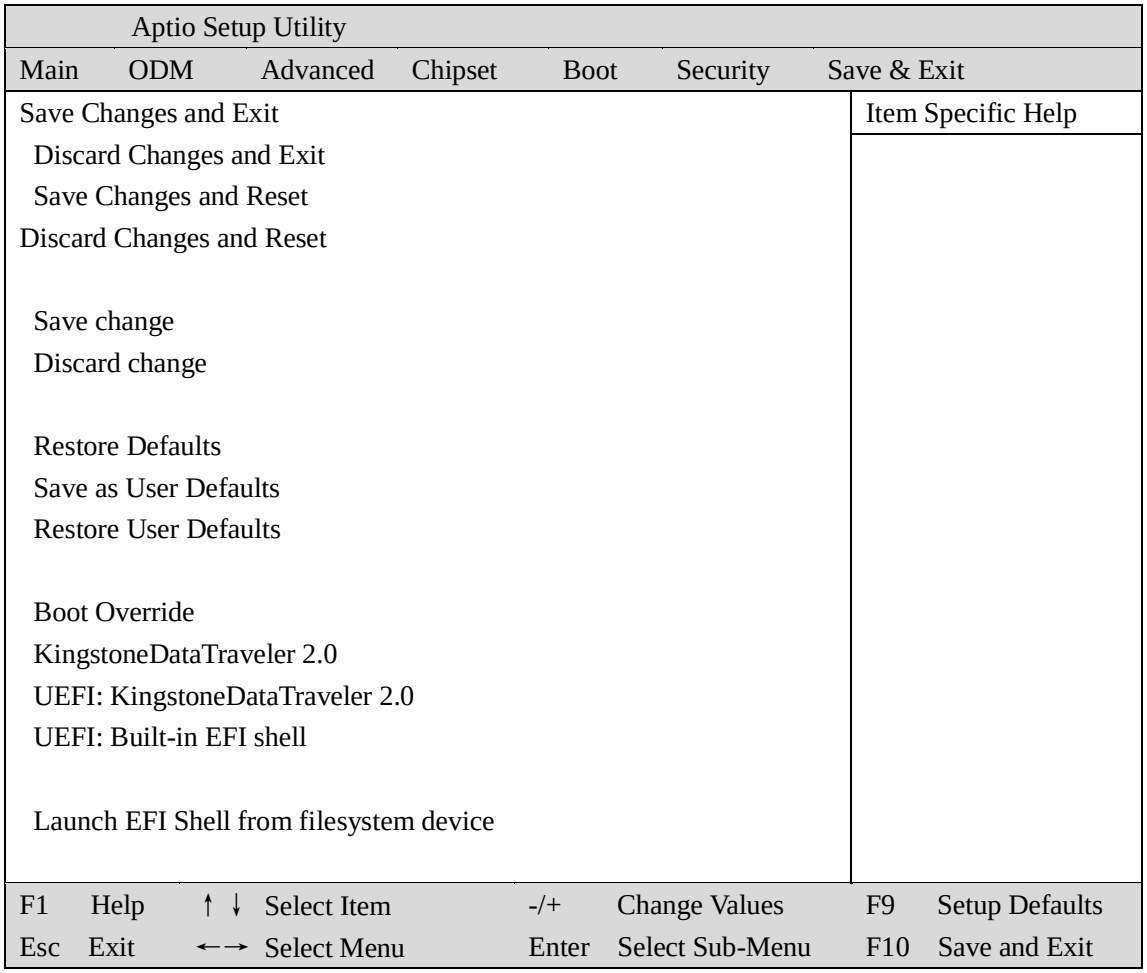
4.密码设置功能

进入 BIOS 设置界面中，选择<Security>选项，进入后，设置超级用户密码和普通用户密码，如下所示：



5.优化.保存设置功能

进入 BIOS 设置界面中，选择<Save & Exit>选项，进行优化.保存设置，如下所示：



Save changes and Exit： 保存当前设置，并退出 BIOS 设置界面，当前设置生效；

Discard changes and Exit： 不保存当前设置，并退出 BIOS 设置界面；

Save changes and Reset： 保存当前设置，并重启电脑，当前设置生效；

Save changes： 保存当前设置，不退出 BIOS 设置界面；

Discard changes： 放弃当前设置，回退到更改操作之前的设置；

Restore Defaults： 加载出厂默认设置为当前设置，需保存退出后生效；

Save as User Defaults： 当前设置保存为用户默认设置；

Restore User Defaults： 加载用户默认值作为当前设置，需要保存才能生效。