**附录B：下位机程序**

**1.1 下位机主程序**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Module Name: ov5640\_ethernet\_800600

//////////////////////////////////////////////////////////////////////////////////

module ov5640\_ethernet\_800600(

input clk\_50m,

input rst\_n,

//Camera接口信号

output cmos\_xclk, //cmos externl clock

output cmos\_reset,

output cmos\_pwnd,

input cmos\_pclk, //cmos pxiel clock

input cmos\_href, //cmos hsync refrence

input cmos\_vsync, //cmos vsync

input [7:0] cmos\_data, //cmos data

inout cmos\_scl, //cmos i2c clock

inout cmos\_sda, //cmos i2c data

//千兆以太网接口信号

output e\_reset,

output e\_mdc, //MDIO的时钟信号，用于读写PHY的寄存器

inout e\_mdio, //MDIO的数据信号，用于读写PHY的寄存器

output [3:0] rgmii\_txd, //RGMII 发送数据

output rgmii\_txctl, //RGMII 发送数据有效信号

output rgmii\_txc, //125Mhz ethernet rgmii tx clock

input [3:0] rgmii\_rxd, //RGMII 接收数据

input rgmii\_rxctl, //RGMII 接收数据有效信号

input rgmii\_rxc //125Mhz ethernet gmii rx clock

);

//上电软件复位

wire reset\_n =1'b1;

//定义GMII信号

wire [ 7:0] gmii\_txd /\*synthesis keep\*/;

wire gmii\_tx\_en /\*synthesis keep\*/;

wire gmii\_tx\_er;

wire gmii\_tx\_clk /\*synthesis keep\*/;

wire gmii\_crs;

wire gmii\_col;

wire [ 7:0] gmii\_rxd;

wire gmii\_rx\_dv;

wire gmii\_rx\_er;

wire gmii\_rx\_clk;

wire [ 1:0] speed\_selection; // 1x gigabit, 01 100Mbps, 00 10mbps

wire duplex\_mode; // 1 full, 0 half

wire clk\_out1 /\*synthesis keep\*/;

assign speed\_selection = 2'b10;

assign duplex\_mode = 1'b1;

assign e\_reset = rst\_n;

assign cmos\_xclk = clk\_out1;

//MDIO配置程序

miim\_top miim\_top\_m0(

.reset\_i (1'b0),

.miim\_clock\_i (gmii\_tx\_clk),

.mdc\_o (e\_mdc),

.mdio\_io (e\_mdio),

.link\_up\_o (), //link status

.speed\_o (), //link speed

.speed\_override\_i (2'b11) //11: autonegoation

);

//GMII转RGMII

util\_gmii\_to\_rgmii util\_gmii\_to\_rgmii\_m0(

.reset(1'b0),

.rgmii\_td (rgmii\_txd),

.rgmii\_tx\_ctl (rgmii\_txctl),

.rgmii\_txc (rgmii\_txc),

.rgmii\_rd (rgmii\_rxd),

.rgmii\_rx\_ctl (rgmii\_rxctl),

.rgmii\_rxc (rgmii\_rxc),

.gmii\_txd (gmii\_txd),

.gmii\_tx\_en (gmii\_tx\_en),

.gmii\_tx\_er (1'b0),

.gmii\_tx\_clk (gmii\_tx\_clk),

.gmii\_crs (gmii\_crs),

.gmii\_col (gmii\_col),

.gmii\_rxd (gmii\_rxd),

.gmii\_rx\_dv (gmii\_rx\_dv),

.gmii\_rx\_er (gmii\_rx\_er),

.gmii\_rx\_clk (gmii\_rx\_clk),

.speed\_selection (speed\_selection),

.duplex\_mode (duplex\_mode));

//////////////////////PLL时钟//////////////////////////////

wire cmos\_clk /\*synthesis keep\*/;

wire pll\_locked /\*synthesis keep\*/;

pll pll\_inst

(

.inclk0(clk\_50m), // IN

.c0(clk\_out1), // 24Mhz

.c1(cmos\_clk), // 24Mhz

.areset(~rst\_n),// IN

.locked(pll\_locked)); // OUT

//Camera初始化部分,Camera LED FLASH control//

wire reg\_conf\_done;

wire[9:0] lut\_index;

wire[31:0] lut\_data;

//I2C master controller

i2c\_config i2c\_config\_m0(

.rst (~i2c\_rst\_n ),

.clk (clk\_50m ),

.clk\_div\_cnt (16'd500),

.i2c\_addr\_2byte (1'b1),

.lut\_index (lut\_index),

.lut\_dev\_addr (lut\_data[31:24] ),

.lut\_reg\_addr (lut\_data[23:8] ),

.lut\_reg\_data (lut\_data[7:0]),

.error (),

.done ( reg\_conf\_done),

.i2c\_scl (cmos\_scl),

.i2c\_sda (cmos\_sda)

);

//configure look-up table

lut\_ov5640\_rgb565\_1024\_768 lut\_ov5640\_rgb565\_1024\_768\_m0(

.lut\_index (lut\_index),

.lut\_data (lut\_data)

);

wire cmos\_vsync\_delay /\*synthesis keep\*/;

wire cmos\_href\_delay /\*synthesis keep\*/;

wire [7:0] cmos\_data\_delay /\*synthesis keep\*/;

camera\_delay camera\_delay\_inst

(

.cmos\_pclk (cmos\_pclk), //cmos pxiel clock

.cmos\_href (cmos\_href), //cmos hsync refrence

.cmos\_vsync (cmos\_vsync), //cmos vsync

.cmos\_data (cmos\_data), //cmos data

.cmos\_href\_delay (cmos\_href\_delay), //cmos hsync refrence

.cmos\_vsync\_delay (cmos\_vsync\_delay), //cmos vsync

.cmos\_data\_delay (cmos\_data\_delay) //cmos data

) ;

//////////////////// CMOS FIFO///////////////////

wire [10 : 0] fifo\_data\_count;

wire [7:0] fifo\_data;

wire fifo\_rd\_en;

camera\_fifo camera\_fifo\_inst (

.aclr (cmos\_vsync), // input rst

.wrclk (cmos\_pclk), // input wr\_clk

.data (cmos\_data\_delay), // input [7 : 0] din

.wrreq (cmos\_href\_delay), // input wr\_en

.rdclk (gmii\_rx\_clk), // input rd\_clk

.rdreq (fifo\_rd\_en), // input rd\_en

.q (fifo\_data), // output [7 : 0] dout

.wrfull (), // output full

.rdempty (), // output empty

.rdusedw (fifo\_data\_count) // output [10 : 0] rd\_data\_count

);

wire eth\_rst\_n ;

wire i2c\_rst\_n ;

power\_on\_rst

#(

.CLK\_FRE(125),

.DELAY\_MS(50)

)

ethernet\_reset

(

.clk (rgmii\_rxc) ,

.rst\_n (rst\_n),

.power\_on\_rstn (eth\_rst\_n)

);

power\_on\_rst

#(

.CLK\_FRE(50),

.DELAY\_MS(10)

)

i2c\_reset

(

.clk (clk\_50m) ,

.rst\_n (pll\_locked),

.power\_on\_rstn (i2c\_rst\_n)

);

mac\_test mac\_test0

(

.gmii\_tx\_clk (gmii\_tx\_clk),

.gmii\_rx\_clk (gmii\_rx\_clk) ,

.rst\_n (eth\_rst\_n ),

.cmos\_vsync (cmos\_vsync),

.cmos\_href (cmos\_href),

.reg\_conf\_done (reg\_conf\_done),

.fifo\_data (fifo\_data), //FIFO读出的8bit数据/

.fifo\_data\_count (fifo\_data\_count), //(fifo\_rdusedw), //FIFO中的数据数量

.fifo\_rd\_en (fifo\_rd\_en), //FIFO读使能

.udp\_send\_data\_length (16'd1024),

.gmii\_rx\_dv (gmii\_rx\_dv),

.gmii\_rxd (gmii\_rxd ),

.gmii\_tx\_en (gmii\_tx\_en ),

.gmii\_txd (gmii\_txd )

);

endmodule

//////////////////////////////////////////////////////////////////////////////////

// Module Name: 摄像头配置程序

//////////////////////////////////////////////////////////////////////////////////

module lut\_ov5640\_rgb565\_1024\_768(

input[9:0] lut\_index, //Look-up table address

output reg[31:0] lut\_data //Device address (8bit I2C address), register address, register data

);

always@(\*)

begin

case(lut\_index)

10'd 0: lut\_data <= {8'h78 , 24'h310311};// system clock from pad, bit[1]

10'd 1: lut\_data <= {8'h78 , 24'h300882};// software reset, bit[7]// delay 5ms

10'd 2: lut\_data <= {8'h78 , 24'h300842};// software power down, bit[6]

10'd 3: lut\_data <= {8'h78 , 24'h310303};// system clock from PLL, bit[1]

10'd 4: lut\_data <= {8'h78 , 24'h3017ff};// FREX, Vsync, HREF, PCLK, D[9:6] output enable

10'd 5: lut\_data <= {8'h78 , 24'h3018ff};// D[5:0], GPIO[1:0] output enable

10'd 6: lut\_data <= {8'h78 , 24'h30341A};// MIPI 10-bit

10'd 7: lut\_data <= {8'h78 , 24'h303713};// PLL root divider, bit[4], PLL pre-divider, bit[3:0]

10'd 8: lut\_data <= {8'h78 , 24'h310801};// PCLK root divider, bit[5:4], SCLK2x root divider, bit[3:2] // SCLK root divider, bit[1:0]

10'd 9: lut\_data <= {8'h78 , 24'h363036};

10'd 10: lut\_data <= {8'h78 , 24'h36310e};

10'd 11: lut\_data <= {8'h78 , 24'h3632e2};

10'd 12: lut\_data <= {8'h78 , 24'h363312};

10'd 13: lut\_data <= {8'h78 , 24'h3621e0};

10'd 14: lut\_data <= {8'h78 , 24'h3704a0};

10'd 15: lut\_data <= {8'h78 , 24'h37035a};

10'd 16: lut\_data <= {8'h78 , 24'h371578};

10'd 17: lut\_data <= {8'h78 , 24'h371701};

10'd 18: lut\_data <= {8'h78 , 24'h370b60};

10'd 19: lut\_data <= {8'h78 , 24'h37051a};

10'd 20: lut\_data <= {8'h78 , 24'h390502};

10'd 21: lut\_data <= {8'h78 , 24'h390610};

10'd 22: lut\_data <= {8'h78 , 24'h39010a};

10'd 23: lut\_data <= {8'h78 , 24'h373112};

10'd 24: lut\_data <= {8'h78 , 24'h360008};// VCM control

10'd 25: lut\_data <= {8'h78 , 24'h360133};// VCM control

10'd 26: lut\_data <= {8'h78 , 24'h302d60};// system control

10'd 27: lut\_data <= {8'h78 , 24'h362052};

10'd 28: lut\_data <= {8'h78 , 24'h371b20};

10'd 29: lut\_data <= {8'h78 , 24'h471c50};

10'd 30: lut\_data <= {8'h78 , 24'h3a1343};// pre-gain = 1.047x

10'd 31: lut\_data <= {8'h78 , 24'h3a1800};// gain ceiling

10'd 32: lut\_data <= {8'h78 , 24'h3a19f8};// gain ceiling = 15.5x

10'd 33: lut\_data <= {8'h78 , 24'h363513};

10'd 34: lut\_data <= {8'h78 , 24'h363603};

10'd 35: lut\_data <= {8'h78 , 24'h363440};

10'd 36: lut\_data <= {8'h78 , 24'h362201}; // 50/60Hz detection 50/60Hz

10'd 37: lut\_data <= {8'h78 , 24'h3c0134};// Band auto, bit[7]

10'd 38: lut\_data <= {8'h78 , 24'h3c0428};// threshold low sum

10'd 39: lut\_data <= {8'h78 , 24'h3c0598};// threshold high sum

10'd 40: lut\_data <= {8'h78 , 24'h3c0600};// light meter 1 threshold[15:8]

10'd 41: lut\_data <= {8'h78 , 24'h3c0708};// light meter 1 threshold[7:0]

10'd 42: lut\_data <= {8'h78 , 24'h3c0800};// light meter 2 threshold[15:8]

10'd 43: lut\_data <= {8'h78 , 24'h3c091c};// light meter 2 threshold[7:0]

10'd 44: lut\_data <= {8'h78 , 24'h3c0a9c};// sample number[15:8]

10'd 45: lut\_data <= {8'h78 , 24'h3c0b40};// sample number[7:0]

10'd 46: lut\_data <= {8'h78 , 24'h381000};// Timing Hoffset[11:8]

10'd 47: lut\_data <= {8'h78 , 24'h381110};// Timing Hoffset[7:0]

10'd 48: lut\_data <= {8'h78 , 24'h381200};// Timing Voffset[10:8]

10'd 49: lut\_data <= {8'h78 , 24'h370864};

10'd 50: lut\_data <= {8'h78 , 24'h400102};// BLC start from line 2

10'd 51: lut\_data <= {8'h78 , 24'h40051a};// BLC always update

10'd 52: lut\_data <= {8'h78 , 24'h300000};// enable blocks

10'd 53: lut\_data <= {8'h78 , 24'h3004ff};// enable clocks

10'd 54: lut\_data <= {8'h78 , 24'h300e58};// MIPI power down, DVP enable

10'd 55: lut\_data <= {8'h78 , 24'h302e00};

10'd 56: lut\_data <= {8'h78 , 24'h430030};// YUV 422,YUYV

10'd 57: lut\_data <= {8'h78 , 24'h501f00};// YUV422

10'd 58: lut\_data <= {8'h78 , 24'h440e00};

10'd 59: lut\_data <= {8'h78 , 24'h5000a7}; // Lenc on, raw gamma on, BPC on, WPC on, CIP on // AEC target

10'd 60: lut\_data <= {8'h78 , 24'h3a0f30};// stable range in high

10'd 61: lut\_data <= {8'h78 , 24'h3a1028};// stable range in low

10'd 62: lut\_data <= {8'h78 , 24'h3a1b30};// stable range out high

10'd 63: lut\_data <= {8'h78 , 24'h3a1e26};// stable range out low

10'd 64: lut\_data <= {8'h78 , 24'h3a1160};// fast zone high

10'd 65: lut\_data <= {8'h78 , 24'h3a1f14};// fast zone low// Lens correction for

10'd 66: lut\_data <= {8'h78 , 24'h580023};

10'd 67: lut\_data <= {8'h78 , 24'h580114};

10'd 68: lut\_data <= {8'h78 , 24'h58020f};

10'd 69: lut\_data <= {8'h78 , 24'h58030f};

10'd 70: lut\_data <= {8'h78 , 24'h580412};

10'd 71: lut\_data <= {8'h78 , 24'h580526};

10'd 72: lut\_data <= {8'h78 , 24'h58060c};

10'd 73: lut\_data <= {8'h78 , 24'h580708};

10'd 74: lut\_data <= {8'h78 , 24'h580805};

10'd 75: lut\_data <= {8'h78 , 24'h580905};

10'd 76: lut\_data <= {8'h78 , 24'h580a08};

10'd 77: lut\_data <= {8'h78 , 24'h580b0d};

10'd 78: lut\_data <= {8'h78 , 24'h580c08};

10'd 79: lut\_data <= {8'h78 , 24'h580d03};

10'd 80: lut\_data <= {8'h78 , 24'h580e00};

10'd 81: lut\_data <= {8'h78 , 24'h580f00};

10'd 82: lut\_data <= {8'h78 , 24'h581003};

10'd 83: lut\_data <= {8'h78 , 24'h581109};

10'd 84: lut\_data <= {8'h78 , 24'h581207};

10'd 85: lut\_data <= {8'h78 , 24'h581303};

10'd 86: lut\_data <= {8'h78 , 24'h581400};

10'd 87: lut\_data <= {8'h78 , 24'h581501};

10'd 88: lut\_data <= {8'h78 , 24'h581603};

10'd 89: lut\_data <= {8'h78 , 24'h581708};

10'd 90: lut\_data <= {8'h78 , 24'h58180d};

10'd 91: lut\_data <= {8'h78 , 24'h581908};

10'd 92: lut\_data <= {8'h78 , 24'h581a05};

10'd 93: lut\_data <= {8'h78 , 24'h581b06};

10'd 94: lut\_data <= {8'h78 , 24'h581c08};

10'd 95: lut\_data <= {8'h78 , 24'h581d0e};

10'd 96: lut\_data <= {8'h78 , 24'h581e29};

10'd 97: lut\_data <= {8'h78 , 24'h581f17};

10'd 98: lut\_data <= {8'h78 , 24'h582011};

10'd 99: lut\_data <= {8'h78 , 24'h582111};

10'd100: lut\_data <= {8'h78 , 24'h582215};

10'd101: lut\_data <= {8'h78 , 24'h582328};

10'd102: lut\_data <= {8'h78 , 24'h582446};

10'd103: lut\_data <= {8'h78 , 24'h582526};

10'd104: lut\_data <= {8'h78 , 24'h582608};

10'd105: lut\_data <= {8'h78 , 24'h582726};

10'd106: lut\_data <= {8'h78 , 24'h582864};

10'd107: lut\_data <= {8'h78 , 24'h582926};

10'd108: lut\_data <= {8'h78 , 24'h582a24};

10'd109: lut\_data <= {8'h78 , 24'h582b22};

10'd110: lut\_data <= {8'h78 , 24'h582c24};

10'd111: lut\_data <= {8'h78 , 24'h582d24};

10'd112: lut\_data <= {8'h78 , 24'h582e06};

10'd113: lut\_data <= {8'h78 , 24'h582f22};

10'd114: lut\_data <= {8'h78 , 24'h583040};

10'd115: lut\_data <= {8'h78 , 24'h583142};

10'd116: lut\_data <= {8'h78 , 24'h583224};

10'd117: lut\_data <= {8'h78 , 24'h583326};

10'd118: lut\_data <= {8'h78 , 24'h583424};

10'd119: lut\_data <= {8'h78 , 24'h583522};

10'd120: lut\_data <= {8'h78 , 24'h583622};

10'd121: lut\_data <= {8'h78 , 24'h583726};

10'd122: lut\_data <= {8'h78 , 24'h583844};

10'd123: lut\_data <= {8'h78 , 24'h583924};

10'd124: lut\_data <= {8'h78 , 24'h583a26};

10'd125: lut\_data <= {8'h78 , 24'h583b28};

10'd126: lut\_data <= {8'h78 , 24'h583c42};

10'd127: lut\_data <= {8'h78 , 24'h583dce};// lenc BR offset // AWB

10'd128: lut\_data <= {8'h78 , 24'h5180ff};// AWB B block

10'd129: lut\_data <= {8'h78 , 24'h5181f2};// AWB control

10'd130: lut\_data <= {8'h78 , 24'h518200};// [7:4] max local counter, [3:0] max fast counter

10'd131: lut\_data <= {8'h78 , 24'h518314};// AWB advanced

10'd132: lut\_data <= {8'h78 , 24'h518425};

10'd133: lut\_data <= {8'h78 , 24'h518524};

10'd134: lut\_data <= {8'h78 , 24'h518609};

10'd135: lut\_data <= {8'h78 , 24'h518709};

10'd136: lut\_data <= {8'h78 , 24'h518809};

10'd137: lut\_data <= {8'h78 , 24'h518975};

10'd138: lut\_data <= {8'h78 , 24'h518a54};

10'd139: lut\_data <= {8'h78 , 24'h518be0};

10'd140: lut\_data <= {8'h78 , 24'h518cb2};

10'd141: lut\_data <= {8'h78 , 24'h518d42};

10'd142: lut\_data <= {8'h78 , 24'h518e3d};

10'd143: lut\_data <= {8'h78 , 24'h518f56};

10'd144: lut\_data <= {8'h78 , 24'h519046};

10'd145: lut\_data <= {8'h78 , 24'h5191f8};// AWB top limit

10'd146: lut\_data <= {8'h78 , 24'h519204};// AWB bottom limit

10'd147: lut\_data <= {8'h78 , 24'h519370};// red limit

10'd148: lut\_data <= {8'h78 , 24'h5194f0};// green limit

10'd149: lut\_data <= {8'h78 , 24'h5195f0};// blue limit

10'd150: lut\_data <= {8'h78 , 24'h519603};// AWB control

10'd151: lut\_data <= {8'h78 , 24'h519701};// local limit

10'd152: lut\_data <= {8'h78 , 24'h519804};

10'd153: lut\_data <= {8'h78 , 24'h519912};

10'd154: lut\_data <= {8'h78 , 24'h519a04};

10'd155: lut\_data <= {8'h78 , 24'h519b00};

10'd156: lut\_data <= {8'h78 , 24'h519c06};

10'd157: lut\_data <= {8'h78 , 24'h519d82};

10'd158: lut\_data <= {8'h78 , 24'h519e38};// AWB control // Gamma

10'd159: lut\_data <= {8'h78 , 24'h548001};// Gamma bias plus on, bit[0]

10'd160: lut\_data <= {8'h78 , 24'h548108};

10'd161: lut\_data <= {8'h78 , 24'h548214};

10'd162: lut\_data <= {8'h78 , 24'h548328};

10'd163: lut\_data <= {8'h78 , 24'h548451};

10'd164: lut\_data <= {8'h78 , 24'h548565};

10'd165: lut\_data <= {8'h78 , 24'h548671};

10'd166: lut\_data <= {8'h78 , 24'h54877d};

10'd167: lut\_data <= {8'h78 , 24'h548887};

10'd168: lut\_data <= {8'h78 , 24'h548991};

10'd169: lut\_data <= {8'h78 , 24'h548a9a};

10'd170: lut\_data <= {8'h78 , 24'h548baa};

10'd171: lut\_data <= {8'h78 , 24'h548cb8};

10'd172: lut\_data <= {8'h78 , 24'h548dcd};

10'd173: lut\_data <= {8'h78 , 24'h548edd};

10'd174: lut\_data <= {8'h78 , 24'h548fea};

10'd175: lut\_data <= {8'h78 , 24'h54901d};// color matrix

10'd176: lut\_data <= {8'h78 , 24'h53811e};// CMX1 for Y

10'd177: lut\_data <= {8'h78 , 24'h53825b};// CMX2 for Y

10'd178: lut\_data <= {8'h78 , 24'h538308};// CMX3 for Y

10'd179: lut\_data <= {8'h78 , 24'h53840a};// CMX4 for U

10'd180: lut\_data <= {8'h78 , 24'h53857e};// CMX5 for U

10'd181: lut\_data <= {8'h78 , 24'h538688};// CMX6 for U

10'd182: lut\_data <= {8'h78 , 24'h53877c};// CMX7 for V

10'd183: lut\_data <= {8'h78 , 24'h53886c};// CMX8 for V

10'd184: lut\_data <= {8'h78 , 24'h538910};// CMX9 for V

10'd185: lut\_data <= {8'h78 , 24'h538a01};// sign[9]

10'd186: lut\_data <= {8'h78 , 24'h538b98}; // sign[8:1] // UV adjust

10'd187: lut\_data <= {8'h78 , 24'h558006};// saturation on, bit[1]

10'd188: lut\_data <= {8'h78 , 24'h558340};

10'd189: lut\_data <= {8'h78 , 24'h558410};

10'd190: lut\_data <= {8'h78 , 24'h558910};

10'd191: lut\_data <= {8'h78 , 24'h558a00};

10'd192: lut\_data <= {8'h78 , 24'h558bf8};

10'd193: lut\_data <= {8'h78 , 24'h501d40};// enable manual offset of contrast// CIP

10'd194: lut\_data <= {8'h78 , 24'h530008};// CIP sharpen MT threshold 1

10'd195: lut\_data <= {8'h78 , 24'h530130};// CIP sharpen MT threshold 2

10'd196: lut\_data <= {8'h78 , 24'h530210};// CIP sharpen MT offset 1

10'd197: lut\_data <= {8'h78 , 24'h530300};// CIP sharpen MT offset 2

10'd198: lut\_data <= {8'h78 , 24'h530408};// CIP DNS threshold 1

10'd199: lut\_data <= {8'h78 , 24'h530530};// CIP DNS threshold 2

10'd200: lut\_data <= {8'h78 , 24'h530608};// CIP DNS offset 1

10'd201: lut\_data <= {8'h78 , 24'h530716};// CIP DNS offset 2

10'd202: lut\_data <= {8'h78 , 24'h530908};// CIP sharpen TH threshold 1

10'd203: lut\_data <= {8'h78 , 24'h530a30};// CIP sharpen TH threshold 2

10'd204: lut\_data <= {8'h78 , 24'h530b04};// CIP sharpen TH offset 1

10'd205: lut\_data <= {8'h78 , 24'h530c06};// CIP sharpen TH offset 2

10'd206: lut\_data <= {8'h78 , 24'h502500};

10'd207: lut\_data <= {8'h78 , 24'h300802}; // wake up from standby, bit[6]

10'd208: lut\_data <= {8'h78 , 24'h303531};// PLL

10'd209: lut\_data <= {8'h78 , 24'h303656};// PLL

10'd210: lut\_data <= {8'h78 , 24'h3c0708};// light meter 1 threshold [7:0]

10'd211: lut\_data <= {8'h78 , 24'h382047};// Sensor flip off, ISP flip on

10'd212: lut\_data <= {8'h78 , 24'h382121};// Sensor mirror on, ISP mirror on, H binning on, enable Compression

10'd213: lut\_data <= {8'h78 , 24'h381431};// X INC

10'd214: lut\_data <= {8'h78 , 24'h381531};// Y INC

10'd215: lut\_data <= {8'h78 , 24'h380000};// HS: X address start high byte

10'd216: lut\_data <= {8'h78 , 24'h380100};// HS: X address start low byte

10'd217: lut\_data <= {8'h78 , 24'h380200};// VS: Y address start high byte

10'd218: lut\_data <= {8'h78 , 24'h380304};// VS: Y address start high byte

10'd219: lut\_data <= {8'h78 , 24'h38040a};// HW (HE)

10'd220: lut\_data <= {8'h78 , 24'h38053f};// HW (HE)

10'd221: lut\_data <= {8'h78 , 24'h380607};// VH (VE)

10'd222: lut\_data <= {8'h78 , 24'h38079b};// VH (VE)

10'd223: lut\_data <= {8'h78 , 24'h380803};// DVPHO //800

10'd224: lut\_data <= {8'h78 , 24'h380920};// DVPHO

10'd225: lut\_data <= {8'h78 , 24'h380a02};// DVPVO //600

10'd226: lut\_data <= {8'h78 , 24'h380b58};// DVPVO

10'd227: lut\_data <= {8'h78 , 24'h380c07};// HTS //Total horizontal size

10'd228: lut\_data <= {8'h78 , 24'h380d68};// HTS

10'd229: lut\_data <= {8'h78 , 24'h380e03};// VTS //total vertical size

10'd230: lut\_data <= {8'h78 , 24'h380fd8};// VTS

10'd231: lut\_data <= {8'h78 , 24'h381306};// Timing Voffset

10'd232: lut\_data <= {8'h78 , 24'h361800};

10'd233: lut\_data <= {8'h78 , 24'h361229};

10'd234: lut\_data <= {8'h78 , 24'h370952};

10'd235: lut\_data <= {8'h78 , 24'h370c03};

10'd236: lut\_data <= {8'h78 , 24'h3a0217};// 60Hz max exposure, night mode 5fps

10'd237: lut\_data <= {8'h78 , 24'h3a0310};// 60Hz max exposure // banding filters are calculated automatically in camera driver

10'd238: lut\_data <= {8'h78 , 24'h3a1417};// 50Hz max exposure, night mode 5fps

10'd239: lut\_data <= {8'h78 , 24'h3a1510};// 50Hz max exposure

10'd240: lut\_data <= {8'h78 , 24'h400402};// BLC 2 lines

10'd241: lut\_data <= {8'h78 , 24'h300200};// enable JFIFO, SFIFO, JPEG

10'd242: lut\_data <= {8'h78 , 24'h3006ff};// enable clock of JPEG2x, JPEG

10'd243: lut\_data <= {8'h78 , 24'h471302};// JPEG mode 2

10'd244: lut\_data <= {8'h78 , 24'h440704};// Quantization scale

10'd245: lut\_data <= {8'h78 , 24'h460b35};

10'd246: lut\_data <= {8'h78 , 24'h460c22};

10'd247: lut\_data <= {8'h78 , 24'h483722}; // DVP CLK divider

10'd248: lut\_data <= {8'h78 , 24'h382402}; // DVP CLK divider

10'd249: lut\_data <= {8'h78 , 24'h5001a3}; // SDE on, scale on, UV average off, color matrix on, AWB on

10'd250: lut\_data <= {8'h78 , 24'h350300}; // AEC/AGC on

// 10'd251: lut\_data <= {8'h78 , 24'h503d80}; //reg\_data<=24'h503d80; test pattern selection control, 80:color bar,00: test disable

// 10'd252: lut\_data <= {8'h78 , 24'h474101}; //reg\_data<=24'h47401; test pattern enable, Test pattern 8-bit

// 10'd253: lut\_data <= {8'h78 , 24'h3c0708}; // lightmeter 1 threshold[7:0]

// 10'd254: lut\_data <= {8'h78 , 24'h382047}; // flip

// 10'd255: lut\_data <= {8'h78 , 24'h382101}; // mirror

// 10'd256: lut\_data <= {8'h78 , 24'h381431}; // timing X inc

// 10'd257: lut\_data <= {8'h78 , 24'h381531}; // timing Y inc

// 10'd258: lut\_data <= {8'h78 , 24'h380000}; // HS

// 10'd259: lut\_data <= {8'h78 , 24'h380100}; // HS

// 10'd260: lut\_data <= {8'h78 , 24'h380200}; // VS

// 10'd261: lut\_data <= {8'h78 , 24'h380304}; // VS

// 10'd262: lut\_data <= {8'h78 , 24'h38040a}; // HW (HE)

// 10'd263: lut\_data <= {8'h78 , 24'h38053f}; // HW (HE)

// 10'd264: lut\_data <= {8'h78 , 24'h380607}; // VH (VE)

// 10'd265: lut\_data <= {8'h78 , 24'h38079f}; // VH (VE)

// 10'd266: lut\_data <= {8'h78 , 24'h380803}; // DVPHO 800

// 10'd267: lut\_data <= {8'h78 , 24'h380920}; // DVPHO

// 10'd268: lut\_data <= {8'h78 , 24'h380a02}; // DVPVO 600

// 10'd269: lut\_data <= {8'h78 , 24'h380b58}; // DVPVO

// 10'd270: lut\_data <= {8'h78 , 24'h380c07}; // HTS

// 10'd271: lut\_data <= {8'h78 , 24'h380d68}; // HTS

// 10'd272: lut\_data <= {8'h78 , 24'h380e03}; // VTS

// 10'd273: lut\_data <= {8'h78 , 24'h380fd8}; // VTS

// 10'd274: lut\_data <= {8'h78 , 24'h381304}; // timing V offset

// 10'd275: lut\_data <= {8'h78 , 24'h361800};

// 10'd276: lut\_data <= {8'h78 , 24'h361229};

// 10'd277: lut\_data <= {8'h78 , 24'h370952};

// 10'd278: lut\_data <= {8'h78 , 24'h370c03};

// 10'd279: lut\_data <= {8'h78 , 24'h3a0202}; // 60Hz max exposure

// 10'd280: lut\_data <= {8'h78 , 24'h3a03e0}; // 60Hz max exposure

// 10'd281: lut\_data <= {8'h78 , 24'h3a0800}; // B50 step

// 10'd282: lut\_data <= {8'h78 , 24'h3a096f}; // B50 step

// 10'd283: lut\_data <= {8'h78 , 24'h3a0a00}; // B60 step

// 10'd284: lut\_data <= {8'h78 , 24'h3a0b5c}; // B60 step

// 10'd285: lut\_data <= {8'h78 , 24'h3a0e06}; // 50Hz max band

// 10'd286: lut\_data <= {8'h78 , 24'h3a0d08}; // 60Hz max band

// 10'd287: lut\_data <= {8'h78 , 24'h3a1402}; // 50Hz max exposure

// 10'd288: lut\_data <= {8'h78 , 24'h3a15e0}; // 50Hz max exposure

// 10'd289: lut\_data <= {8'h78 , 24'h400402}; // BLC line number

// 10'd290: lut\_data <= {8'h78 , 24'h300200}; // reset JFIFO, SFIFO, JPG

// 10'd291: lut\_data <= {8'h78 , 24'h3006ff}; // enable clock of JPEG2x, JPEG

// 10'd292: lut\_data <= {8'h78 , 24'h471302}; // JPEG mode 2

// 10'd293: lut\_data <= {8'h78 , 24'h440704}; // Quantization sacle

// 10'd294: lut\_data <= {8'h78 , 24'h460b37};

// 10'd295: lut\_data <= {8'h78 , 24'h460c20};

// 10'd296: lut\_data <= {8'h78 , 24'h483716}; // MIPI global timing

// 10'd297: lut\_data <= {8'h78 , 24'h382404}; // PCLK manual divider

// 10'd298: lut\_data <= {8'h78 , 24'h350300}; // AEC/AGC on

// 10'd299: lut\_data <= {8'h78 , 24'h301602}; //Strobe output enable

// 10'd300: lut\_data <= {8'h78 , 24'h3b070a}; //FREX strobe mode1

// 10'd301: lut\_data <= {8'h78 , 24'h3b0083}; //STROBE CTRL: strobe request ON, Strobe mode: LED3

// 10'd302: lut\_data <= {8'h78 , 24'h3b0000}; //STROBE CTRL: strobe request OFF

default:lut\_data <= {8'hff,16'hffff,8'hff};

endcase

end

endmodule

//////////////////////////////////////////////////////////////////////////////////////

//Module Name : 以太网传输

//Description :

//

//////////////////////////////////////////////////////////////////////////////////////

`timescale 1 ns/1 ns

module mac\_test

(

input rst\_n ,

input cmos\_vsync,

input cmos\_href,

input reg\_conf\_done,

// input vsync\_posedge,

// input [15:0] identify\_code,

input [15:0] udp\_send\_data\_length,

input [7:0] fifo\_data, //FIFO中的数据

input [10:0] fifo\_data\_count, //FIFO中的数据数量

output fifo\_rd\_en, //FIFO读使能

input gmii\_tx\_clk ,

input gmii\_rx\_clk ,

input gmii\_rx\_dv,

input [7:0] gmii\_rxd,

output reg gmii\_tx\_en,

output reg [7:0] gmii\_txd

);

reg gmii\_rx\_dv\_d0 ;

reg [7:0] gmii\_rxd\_d0 ;

wire gmii\_tx\_en\_tmp ;

wire [7:0] gmii\_txd\_tmp ;

reg [9:0] vsync\_posedge\_buf ;

reg [15:0] identify\_code\_d0 ;

wire udp\_tx\_req ;

wire arp\_request\_req ;

wire mac\_send\_end ;

reg write\_end ;

wire [7:0] udp\_rec\_ram\_rdata ;

reg [10:0] udp\_rec\_ram\_read\_addr ;

wire [15:0] udp\_rec\_data\_length ;

wire udp\_rec\_data\_valid ;

reg [31:0] wait\_cnt ;

wire button\_negedge ;

wire mac\_not\_exist ;

wire arp\_found ;

parameter IDLE = 10'b0000000001 ;

parameter ARP\_REQ = 10'b0000000010 ;

parameter ARP\_SEND = 10'b0000000100 ;

parameter ARP\_WAIT = 10'b0000001000 ;

parameter GEN\_REQ = 10'b0000010000 ;

parameter CHECK\_FIFO = 10'b0000100000 ;

parameter SEND = 10'b0001000000 ;

parameter WAIT = 10'b0010000000 ;

parameter CHECK\_ARP = 10'b0100000000 ;

parameter WAIT\_SYNC = 10'b1000000000 ;

reg [9:0] state ;

//reg [9:0] state ;

//

//always @(posedge gmii\_tx\_clk or negedge rst\_n)

//begin

// if (~rst\_n)

// state <= IDLE ;

// else

// state <= state ;

//end

always @(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if (~rst\_n)

state <= IDLE ;

else

begin

case(state)

IDLE : begin

if (wait\_cnt == 32'd1250000 ) //wait for 10ms

state <= ARP\_REQ ;

else

state <= IDLE ;

end

ARP\_REQ : state <= ARP\_SEND ;

ARP\_SEND : begin

if (mac\_send\_end)

state <= ARP\_WAIT ;

else

state <= ARP\_SEND ;

end

ARP\_WAIT : begin

if (arp\_found)

state <= WAIT\_SYNC ;

else if (wait\_cnt[27])// == 32'd125\_000\_000)

state <= ARP\_REQ ;

else

state <= ARP\_WAIT ;

end

WAIT\_SYNC : begin

if (vsync\_posedge\_buf[9])

state <= CHECK\_FIFO ;

else

state <= WAIT\_SYNC ;

end

CHECK\_FIFO : begin

if (fifo\_rdusedw[10])// == 3'b111)// > 11'd768)

state <= GEN\_REQ ;

else if (wait\_cnt[31])// == 32'd4\_125\_000)

state <= WAIT\_SYNC ;

else

state <= CHECK\_FIFO ;

end

GEN\_REQ : begin

state <= SEND ;

end

SEND : begin

if (mac\_send\_end)

state <= CHECK\_ARP ;

else

state <= SEND ;

end

// WAIT : begin

// if (wait\_cnt == 32'd5)

// state <= CHECK\_ARP ;

// else

// state <= WAIT ;

// end

CHECK\_ARP : begin

if (mac\_not\_exist)

state <= ARP\_REQ ;

else

state <= CHECK\_FIFO ;

end

default : state <= IDLE ;

endcase

end

end

always@(posedge gmii\_rx\_clk or negedge rst\_n)

begin

if(rst\_n == 1'b0)

begin

gmii\_rx\_dv\_d0 <= 1'b0 ;

gmii\_rxd\_d0 <= 8'd0 ;

end

else

begin

gmii\_rx\_dv\_d0 <= gmii\_rx\_dv ;

gmii\_rxd\_d0 <= gmii\_rxd ;

end

end

always@(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if(rst\_n == 1'b0)

begin

gmii\_tx\_en <= 1'b0 ;

gmii\_txd <= 8'd0 ;

end

else

begin

gmii\_tx\_en <= gmii\_tx\_en\_tmp ;

gmii\_txd <= gmii\_txd\_tmp ;

end

end

mac\_top mac\_top0

(

.gmii\_tx\_clk (gmii\_tx\_clk) ,

.gmii\_rx\_clk (gmii\_rx\_clk) ,

.rst\_n (rst\_n) ,

.source\_mac\_addr (48'h00\_0a\_35\_01\_fe\_c0) , //source mac address

.identify\_code (identify\_code\_d0 ),

.TTL (8'h80),

.source\_ip\_addr (32'hc0a80002),

.destination\_ip\_addr (32'hc0a80003),

.udp\_send\_source\_port (16'h1f90),

.udp\_send\_destination\_port (16'h1f90),

.fifo\_data (fifo\_data),

.fifo\_rd\_en (fifo\_rd\_en),

.ram\_wr\_data () ,

.ram\_wr\_en (),

.udp\_ram\_data\_req (),

.udp\_send\_data\_length (udp\_send\_data\_length),

.udp\_tx\_req (udp\_tx\_req),

.arp\_request\_req (arp\_request\_req ),

.mac\_send\_end (mac\_send\_end),

.mac\_data\_valid (gmii\_tx\_en\_tmp),

.mac\_tx\_data (gmii\_txd\_tmp),

.rx\_dv (gmii\_rx\_dv\_d0 ),

.mac\_rx\_datain (gmii\_rxd\_d0 ),

.udp\_rec\_ram\_rdata (udp\_rec\_ram\_rdata),

.udp\_rec\_ram\_read\_addr (udp\_rec\_ram\_read\_addr),

.udp\_rec\_data\_length (udp\_rec\_data\_length ),

.udp\_rec\_data\_valid (udp\_rec\_data\_valid),

.arp\_found (arp\_found ),

.mac\_not\_exist (mac\_not\_exist )

) ;

assign udp\_tx\_req = (state == GEN\_REQ) ;

assign arp\_request\_req = (state == ARP\_REQ) ;

reg [10:0] fifo\_rdusedw ;

always @(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if (!rst\_n)

fifo\_rdusedw <= 11'd0 ;

else

fifo\_rdusedw <= fifo\_data\_count ;

end

reg cmos\_vsync\_d0 ;

reg cmos\_vsync\_d1 ;

reg cmos\_vsync\_d2 ;

wire vsync\_posedge ;

reg cmos\_href\_d0 ;

reg cmos\_href\_d1 ;

reg cmos\_href\_d2 ;

reg [15:0] identify\_code ;

always @(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if (!rst\_n)

begin

cmos\_vsync\_d0 <= 1'b0 ;

cmos\_vsync\_d1 <= 1'b0 ;

cmos\_vsync\_d2 <= 1'b0 ;

end

else

begin

cmos\_vsync\_d0 <= cmos\_vsync ;

cmos\_vsync\_d1 <= cmos\_vsync\_d0 ;

cmos\_vsync\_d2 <= cmos\_vsync\_d1 ;

end

end

assign vsync\_posedge = ~cmos\_vsync\_d2 & cmos\_vsync\_d1 ;

always @(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if (!rst\_n)

begin

cmos\_href\_d0 <= 1'b0 ;

cmos\_href\_d1 <= 1'b0 ;

cmos\_href\_d2 <= 1'b0 ;

end

else

begin

cmos\_href\_d0 <= cmos\_href ;

cmos\_href\_d1 <= cmos\_href\_d0 ;

cmos\_href\_d2 <= cmos\_href\_d1 ;

end

end

always @(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if (!rst\_n)

identify\_code <= 16'd0 ;

else if (vsync\_posedge)

identify\_code <= 16'd0 ;

else if (cmos\_href\_d1 == 1'b1 && cmos\_href\_d2 == 1'b0)

identify\_code <= identify\_code + 1'b1 ;

end

always@(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if(rst\_n == 1'b0)

wait\_cnt <= 0 ;

else if (state == IDLE ||state == CHECK\_FIFO || state == WAIT || state == ARP\_WAIT)

wait\_cnt <= wait\_cnt + 1'b1 ;

else

wait\_cnt <= 0 ;

end

always@(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if(rst\_n == 1'b0)

vsync\_posedge\_buf <= 10'd0 ;

else

vsync\_posedge\_buf <= {vsync\_posedge\_buf[8:0], vsync\_posedge} ;

end

always@(posedge gmii\_tx\_clk or negedge rst\_n)

begin

if(rst\_n == 1'b0)

identify\_code\_d0 <= 10'd0 ;

else if (state == CHECK\_FIFO)

identify\_code\_d0 <= identify\_code ;

end

endmodule

**附录C：上位机程序**

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%图像处理子程序

%作者：胡一浪

%功能：识别一张图片

%参数：有效合理图片和显示句柄；

%返回：无

%版本：5.1 2018/4/23

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function RGB\_extract\_basic(src\_image,handles\_temp)

identify\_result = '';

identify\_result\_num = 0;%是否能识别标识

standard\_image=imresize(src\_image,[592 748]);%归一化，尺寸592:748，%防止图片过大过小

% figure;imshow(standard\_image);title('原始图像');

image\_height=size(standard\_image,1);%size函数图像的高

image\_width=size(standard\_image,2);%size函数图像的宽

DEAL\_NUM = 3; %红，蓝，黄各处理一次

for deal\_num = 1:DEAL\_NUM

%彩色转二值化

CA = zeros(image\_height,image\_width,'double');

for i=1:image\_height

for j=1:image\_width

if deal\_num == 1 %处理黄色情况

if standard\_image(i,j,1) < 50 && standard\_image(i,j,2) < 50 && standard\_image(i,j,3)< 50

CA(i,j)=standard\_image(i,j,3)+standard\_image(i,j,1);

end

end

if deal\_num == 2 %处理蓝色情况

if standard\_image(i,j,1) < 10

CA(i,j)=standard\_image(i,j,2);

end

end

if deal\_num == 3 %处理红色情况

if 1.6\*standard\_image(i,j,1)-standard\_image(i,j,2)-standard\_image(i,j,3)>=0

CA(i,j)=1.6\*standard\_image(i,j,1)-standard\_image(i,j,2)-standard\_image(i,j,3);

end

end

end

end

%边缘检测

if deal\_num == 3 %当前处理的是红色

BianYuan=edge(CA,'canny',0.5);%Canny算子边缘检测

else %当前处理的是黄和蓝色

BianYuan=edge(CA,'canny',0.4);%Canny算子边缘检测

end

% figure();imshow(BianYuan),title('Canny算子边缘检测后图像');

%图像预处理

se=strel('disk',6); %创建一个指定半径10的平面圆盘形的结构元素

BianYuan\_rgb=imclose(BianYuan,se);%闭操作；

% figure();imshow(BianYuan\_rgb),title('闭操作');

BianYuan\_rgb\_tc=bwfill(BianYuan\_rgb,'holes');%填充

% figure();imshow(BianYuan\_rgb\_tc),title('填充');

YuanShiLvBo=bwareaopen(BianYuan\_rgb\_tc,1000);%从对象中移除面积小于2000的小对象

% figure();imshow(YuanShiLvBo),title('移除');

YuanShiLvBo=YuanShiLvBo(:,:,1);

% figure();imshow(YuanShiLvBo),title('移除面积小于1000的后图像');

[~,num\_L\_1]=bwlabeln(YuanShiLvBo); %记录滤波后白区数目num\_L\_1；

% 边缘平滑处理

hsize = [10 10];

sigma = 1.5;

h = fspecial('gaussian', hsize, sigma);

YuanShiLvBo = imfilter(YuanShiLvBo, h, 'replicate');

% figure();imshow(YuanShiLvBo),title(strcat('边缘平滑处理',num2str(deal\_num)));

[L,num\_L]=bwlabeln(YuanShiLvBo); %对连通对象的各个分离部分进行标注,L中包含了连通对象的标注，num\_L平滑处理后白色区域。

% 第一次修正算法：根据平滑处理前后是否产生新的区域来判断

if num\_L\_1 ~= num\_L %如果平滑分离了白色区域，不满足条件，

% fprintf('%s\n','如果平滑分离了白色区域，不满');

continue;

end

%第二次修正算法：通过白色区域数目和最大白色区域占图片比例来判断

Max\_White\_area = 0; %最大白色面积

S=regionprops(L,'Area','boundingbox');

rects\_pro = cat(1,S.BoundingBox);

%最大面积

for i = 1:num\_L

if(S(i).Area >= Max\_White\_area)

Max\_White\_area = S(i).Area;

end

end

sum\_area\_ratio = Max\_White\_area/(image\_height\*image\_width);%最大的白色区域占图片的面积

if(sum\_area\_ratio >= 0.1 || sum\_area\_ratio == 0 || num\_L > 4) %面积过大，面积为0，或区域超过4个异常

% fprintf('%s\n','max\_sum\_area/Max\_White\_area <= 0.8');

continue;

end

for i = 1:num\_L

if S(i).Area/Max\_White\_area <= 0.2 %其他的面积小于最大面积的0.2，就把其他面积涂白

YuanShiLvBo = change\_black(YuanShiLvBo,round(rects\_pro(i, 1)),round(rects\_pro(i, 2)),rects\_pro(i, 3),rects\_pro(i, 4));

% figure();imshow(YuanShiLvBo),title('处理后的图像');

end

end

%第三次修正算法：根据涂白后面积是否变化来判断

[L,num]=bwlabeln(YuanShiLvBo); %对连通对象的各个分离部分进行标注,L中包含了连通对象的标注。默认

img\_reg = regionprops(L,'Area','boundingbox');

rects = cat(1,img\_reg.BoundingBox);

max\_sum\_area = 0;

for i = 1:num

if(S(i).Area >= max\_sum\_area)

max\_sum\_area = S(i).Area;

end

end

if(max\_sum\_area/Max\_White\_area <= 0.8) %过滤后面积变化，则不行

% fprintf('%s\n','max\_sum\_area/Max\_White\_area <= 0.8');

continue;

end

size\_num\_rects = size(rects, 1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%对定位出来的目标区域进行提取

for i = 1:size\_num\_rects

Iresult\_YuanShiLvBo=imcrop(YuanShiLvBo,rects(i, :));%提取出定位区域的黑白图像

% figure();imshow(Iresult\_YuanShiLvBo),title('切割出来的黑白区域');

%判断定位的区域是否符合标准

%形状上

[L\_test,num\_test]=bwlabeln(Iresult\_YuanShiLvBo); %对连通对象的各个分离部分进行标注,L中包含了连通对象的标注。默认

if num\_test > 1 || rects(i, 3)/rects(i, 4) < 0.7 || rects(i, 3)/rects(i, 4) > 1.5 %分割后应该是一个填充的白色区域，有两个说明不对

% msgbox({'分割后白色数量: ' num2str(num\_test) num2str(deal\_num) num2str(rects(i, 3)/rects(i, 4)) '不应该被处理'},'分割后白色数量');

continue;

end

%面积上

L\_test\_Area = regionprops(L\_test,'Area');

white\_black\_ratio = L\_test\_Area(1).Area/(size(Iresult\_YuanShiLvBo,1)\*size(Iresult\_YuanShiLvBo,2));

% msgbox({'分割后白色占总的面积比例:' num2str(white\_black\_ratio)},'分割后白色占总的面积比例');

if deal\_num == 1 %三角面积占这个图只有0.5-0.6

if(white\_black\_ratio <= 0.5 || white\_black\_ratio >= 0.6)

continue;

end

else

if(white\_black\_ratio <= 0.7 || white\_black\_ratio >= 0.8)

continue;

end

end

%处理定位的合理图片

Iresult\_I=imcrop(standard\_image,rects(i, :));%原图切割

% figure();imshow(Iresult\_I),title('原图切割');

Iresult\_change\_white = change\_white(Iresult\_I,Iresult\_YuanShiLvBo);%目标区域周围部分变白

% figure();imshow(BianYuan\_rgb\_qiege),title('闭操作切割图像');

BianYuan\_qiege=imcrop(BianYuan,rects(i, :));%canny切割

% figure();imshow(BianYuan\_qiege),title('定位之后，canny切割图像');

BianYuan\_qiege=imresize(BianYuan\_qiege,[60 60]);

% figure();imshow(BianYuan\_qiege),title('60x60归一化图像');

se=strel('disk',5); %创建一个指定半径10的平面圆盘形的结构元素

BianYuan\_qiege\_imclose=imclose(BianYuan\_qiege,se);%闭操作；

% figure();imshow(BianYuan\_qiege\_imclose),title('canny切割图像在闭操作');

identify\_result\_num = 1;

% figure();imshow(Iresult\_change\_white),title('目标区域周围部分变白');

e = template\_create(Iresult\_change\_white,deal\_num,BianYuan\_qiege\_imclose);%生成匹配的模板

% figure();imshow(e),title('生成匹配的模板');

[return\_template\_image,identify\_name] = template\_matching(e);%模板匹配

identify\_result = strcat(identify\_result,identify\_name);

if(size(rects, 1) ~= 1 && i ~= size(rects, 1))

identify\_result = strcat(identify\_result,',');%逗号隔开多个结果

end

end

end

axes(handles\_temp.original\_image);

imshow(src\_image);%初始图像

unidentification\_img = imread('unidentification.jpg');

if(identify\_result\_num == 1)

axes(handles\_temp.target\_image);

imshow(Iresult\_change\_white);%定位图像

axes(handles\_temp.standard\_image);

imshow(e);%提取图像

axes(handles\_temp.template\_image);

imshow(return\_template\_image);%模板图像

set(handles\_temp.identify\_result,'String',identify\_result);%识别结果

drawnow;%没有drawnow 只会显示最后结果

else

axes(handles\_temp.target\_image);

imshow(unidentification\_img);%定位图像

axes(handles\_temp.standard\_image);

imshow(unidentification\_img);%提取图像

axes(handles\_temp.template\_image);

imshow(unidentification\_img);%模板图像

set(handles\_temp.identify\_result,'String','未识别');%识别结果

drawnow;%没有drawnow 只会显示最后结果

end

% disp(['etime程序总耗时',num2str(etime(clock,t9))]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%图像处理子程序

%作者：胡一浪

%功能：生成用于匹配的图片

%参数：rgb是原图，deal\_num：当前按照哪种方式识别：红，蓝，黄，BianYuan\_rgb\_qiege：切割闭操作的图像；

%返回：用于匹配的图片

%版本：5.1 2018/4/23

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function e = template\_create(rgb,deal\_num,BianYuan\_rgb\_qiege)

% %去除内部干扰,修正颜色

image\_height=size(rgb,1);%size函数图像的高

image\_width=size(rgb,2);%size函数图像的宽

if(deal\_num == 3) %红色

for i = 1:image\_height

for j = 1:image\_width

if ((rgb(i,j,1) > 150 && rgb(i,j,1)<160 && rgb(i,j,2)>150&& rgb(i,j,2)<170&& rgb(i,j,3)>130&& rgb(i,j,3)<160)) || ((rgb(i,j,1) >= 0 && rgb(i,j,1)<50 && rgb(i,j,2)>30&& rgb(i,j,2)<100&& rgb(i,j,3)>100&& rgb(i,j,3)<200)) || (rgb(i,j,1) < 10 && rgb(i,j,2)>30&& rgb(i,j,2)<50&& rgb(i,j,3)>100&& rgb(i,j,3)<150 || ((rgb(i,j,1) > 5 && rgb(i,j,1)<50 && rgb(i,j,2)>20 && rgb(i,j,2)<60&& rgb(i,j,3)>80&& rgb(i,j,3)<160)))

rgb(i,j,1)=255;

rgb(i,j,2)=255;

rgb(i,j,3)=255;

end

end

end

end

if(deal\_num == 2) %蓝

for i = 1:image\_height

for j = 1:image\_width

if (rgb(i,j,1) > 20)

rgb(i,j,1)=255;

rgb(i,j,2)=255;

rgb(i,j,3)=255;

end

end

end

end

% figure();

% subplot(4,2,1);imshow(rgb);title('1.原图');

b=rgb2gray(rgb);

% subplot(4,2,2); imshow(b);title('2.灰度');

I1=imbinarize(b,0.6);

%修正算法，针对图片灰度处理不好,图片全黑

[L,num\_L]=bwlabeln(I1); %对连通对象的各个分离部分进行标注,L中包含了连通对象的标注。默认

sum\_area = 0;

S=regionprops(L,'Area');

for i = 1:num\_L

sum\_area = S(i).Area+sum\_area;

end

sum\_area = sum\_area/(image\_height\*image\_width);

if(sum\_area <= 0.35)

% msgbox(num2str(sum\_area),'图片全黑');

% fprintf('%s\n','图片全黑');

I1=imbinarize(b,0.4);

end

[L,num\_L]=bwlabeln(I1); %对连通对象的各个分离部分进行标注,L中包含了连通对象的标注。默认

sum\_area = 0;

S=regionprops(L,'Area');

for i = 1:num\_L

sum\_area = S(i).Area+sum\_area;

end

sum\_area = sum\_area/(image\_height\*image\_width);

if(sum\_area <= 0.35)

% msgbox(num2str(sum\_area),'图片全黑');

% fprintf('%s\n','图片仍然全黑');

e = BianYuan\_rgb\_qiege;

return;

end

% subplot(4,2,3);imshow(I1);title('3.二值化');

I1=~I1;

% subplot(4,2,4);imshow(I1);title('4.反色');

if(deal\_num == 3)

I1=bwareaopen(I1,90);%从对象中移除面积小于2000的小对象

else

I1=bwareaopen(I1,40);%从对象中移除面积小于2000的小对象

end

% subplot(4,2,5);imshow(I1),title('移除面积小于90');

I1=qiege(I1);

I1=imresize(I1,[60 60]);

% subplot(4,2,6);imshow(I1);title('归一化');

e = I1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%图像处理子程序

%作者：胡一浪

%功能：模板匹配

%参数：rgb是原图，deal\_num：当前按照哪种方式识别：红，蓝，黄，BianYuan\_rgb\_qiege：切割闭操作的图像；

%返回：匹配的模板和模板名

%版本：5.1 2018/4/23

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [return\_template\_image,identify\_name]= template\_matching(src)%flag\_result识别结果

src=imresize(src,[60 60]);

file\_path = '.\binaryzation\_image\_template\';% 图像文件夹路径

img\_path\_list = dir(strcat(file\_path,'\*.jpg'));%获取该文件夹中所有jpg格式的图像

img\_num = length(img\_path\_list);%获取图像总数量

if img\_num > 0 %有满足条件的图像

for j = 1:img\_num %逐一读取图像

image\_name = img\_path\_list(j).name;% 图像名

image = imread(strcat(file\_path,image\_name));

temp = 0;

image\_height=size(image,1);%size函数图像的高

image\_width=size(image,2);%size函数图像的宽

for image\_height\_temp=1:image\_height %图像高度

for image\_width\_temp=1:image\_width %图像宽度

if(src(image\_height\_temp,image\_width\_temp) -image(image\_height\_temp,image\_width\_temp) == 0)

temp = temp + 1;

end

end

end

% fprintf('%s %d %s\n',image\_name,j,num2str(temp/(image\_height \* image\_height)));% 显示正在处理的图像名

if(temp/(image\_height \* image\_height) > 0.821)

return\_template\_image = image;

s = regexp(image\_name,'\.','split');%读到的文件是xx.bmp，把bmp去掉

identify\_name = char(s(1));

%写入文件

% filename = fopen('C:\Users\HYL\Desktop\identify\_result.txt','wt');

% fprintf(filename,'%s\n',identify\_name);

% fclose(filename);

return;

end

end

identify\_name = '未识别';

return\_template\_image = imread('unidentification.jpg');

imwrite(src,strcat(datestr(now,'HHMMSS'),'.jpg'),'bmp');%保存未识别的图像

else

identify\_name = '模板库没有.jpg模板';

return\_template\_image = imread('unidentification.jpg');

end