# 【1】BOOL MDrv\_Tuner\_InitMxd1516(void)

{

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

UINT8 u8Data;

## 【2-0-1-000】mxd1516\_i2c\_read(0x01,&u8Data);

STTBX\_Print((L"mxd1516 chip ID verfication:%d \n",u8Data));

## 【1-0】mxd1516\_init();

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

return TRUE;

}

# 【2】UINT32 MDrv\_Tuner\_SetTunerMxd1516( UINT32 dwFreq, // KHz UINT8 ucBw // MHz)

{

UINT8 u8BW;

u8BW = 6+ucBw;

## 【2-0-1】tun\_mxd1516\_control(dwFreq,u8BW);

return (HDIC\_NO\_ERROR);

}

# 【3】BOOL MDrv\_Tuner\_PowerOnOff(BOOL bPowerOn)

{

## 【1-0-1】mxd1516\_i2c\_write( 0x41, 0x90);

mxd1516\_i2c\_write( 0x42, 0x02);【1-0-1】

mxd1516\_i2c\_write( 0x0a, 0x7e);【1-0-1】

mxd1516\_i2c\_write( 0x02, 0x01);【1-0-1】

//loop through -2dB gain

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

return TRUE;

}

# 【4】UINT32 tun\_mxd1516\_status(UINT32 tuner\_id, UINT8 \*lock)

{

UINT8 status;

status = mxd1516\_get\_status();

if (status==0)

{

\*lock=1;

}

else

{

\*lock=0;

}

return 0;

}

# 【5】unsigned short tun\_agc\_boundary[8][2]=

{{170,206},{206,247},{470,544},{544,600},{600,644},{644,692},{692,775},{775,870}};

# 【1-0】void mxd1516\_init( void )

{

// UINT8 u8Data;

unsigned int i=0;

//int status =0;

unsigned char init\_table[] =

// basic function reset

{

0x00, 0x00, // soft-reset all fsm

//PMU setup

0x07, 0xff, // set LPC control

0x08, 0xff, // set LPC control

0x09, 0xff, // set LPC control

0x0a, 0x7f, // turn on LTA\_LDO

0x0b, 0xe9, // rbus rfrssi range and rect\_tune

0x0c, 0x6d, // turn on LO buffer %ip2 improvement by ymiao //change in 20110823

0x0d, 0x1b, // set LPF clk 27MHz

0x2b, 0x06, // set LTALDO to 1.4V, LDOIF to 1.2V, LDORF to 1.4V

//Clock Setup

0x0e, 0x18, // set clock domain,clock out disable, sar\_clk to800kHz by ymiao

//LPF setup

0x24, 0x02, // set ACS LPF\_BW offset to 500kHz by ymiao

0x27, 0x0f, // set lpf bw

0x2a, 0x03, // enable ifrssi Vout, turn off test module

0x2d, 0x28, // bypass sc

//system bring-up

0x02, 0x00, // release init\_cfg

0x00, 0x0e, // release fd, sdm, sc fsm(FD starts to work after reset release)

0x03, 0x08, // start afc

//agc mode setup

0x39, 0x06, // 0x1E % ACS high

0x38, 0x04, // 0x19 % ACS low

0x37, 0x40, // ALT high by ymiao

0x36, 0x38, // ALT low by ymiao

0x31, 0x12,

0x32, 0xea,

0x33, 0x0e,

0x34, 0x41,

0x35, 0x43, // only CN disable pga adcoc

0x3a, 0x17, // tia\_gain\_acs\_low\_thd, can enter ACS mode by ymiao

0x3b, 0x17, // tia\_gain\_acs\_high\_thd, can not exit ACS mode by this condition by ymiao

0x3c, 0x01, // lna\_far\_thd, Far condition

0x3d, 0x00, // tia gain low thd

0x3e, 0x12, // tia gain far high thd by ymiao //change in 20110823

0x3f, 0x6a, // alpf\_voltage\_low\_thd by ymiao

0x40, 0x78, // alpf\_voltage\_high\_thd by ymiao //change in 20110925 V1.4

0x41, 0x91, // set AGC mode to normal with lta, inter-hold, far, disable alt,acs mode

0x42, 0x09, //

0x43, 0x77, //

0x46, 0x14, // set LNA cfg

0x47, 0x61, // disable the GSM mode, tia\_bw\_ctrl, tia\_buf\_cap\_en and lo\_filter\_mode by ymiao

0x48, 0x65, // turn off RSSI calibration

0x49, 0xdd, // agc rfrssi gain and bias

0x4a, 0xc5, // agc ifrssi gain = mid

//0x50, 0x2c, // set tia\_pout\_cn

//0x51, 0x2c, // set tia\_pout\_n1 by ymiao

//0x52, 0x2c, // set tia\_pout\_n2 % shuld be smaller than pout\_n1 by ymiao

//0x53, 0x34, // set tia\_pout\_far % shuld be equal to pout\_n2

0x54, 0x07, // set tia\_pout large range

0x55, 0x03, // set tia\_lock\_range to +/-1db by ymiao

0x56, 0x68, // set tia\_settle\_tw to 64us/256us 0x68

0x57, 0x1f, // TIA protect

0x58, 0x00, // TIA BW = 6M

0x59, 0x4a, // 0x00 for LPC, 0x4E % set rf\_settle\_tw to 16us/128\*8us=1ms 0x47

0x5a, 0xff, // LTA fail num = 15, LNA fial num =10

0xae, 0x92, // set saradc sample tw to 18us/2us

0xaf, 0x3d, // set rf\_sample\_num to 8/32 samples(320us\*4 = 1.2ms) 0x35

0xb0, 0x3a, // set tia\_sample\_num to 16/512 samples(10mS) 0x49

0xb2, 0x03, // set BBAGC to PWM mode, positive polarity; set PWM sample number to 1 //change in 20110823

0xb8, 0x27, // LTA tbl size = 4, LNA tbl size = 8

// AGC Table Setup

0xba, 0x1d, // mixer max gain //change in 20110810

0xbb, 0x17, // mixer max gain protect

0xbc, 0x0c, // mixer min gain

0xbd, 0x0b, // Mixer 1st gain

0xbe, 0x08, // LTA 1st gain

0xbf, 0x24, // LTA 2nd gain

0xc0, 0x12, // LTA 3rd gain

// AGC param setup

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

//Start AGC, FD

0x00, 0x0f, // release AGC\_FSM

0x03, 0x01, // start AGC

0x00, 0x07, // reset FD

0x00, 0x0F, // release FD

0x47, 0x60,

0xb3, 0x0f,

0xb7, 0xac,

0x48, 0x75,

0x03, 0x02,

0x47, 0x61

};

UINT32 err;

for(i=0;i<sizeof(init\_table);i+=2)

{

## 【1-0-1】err = mxd1516\_i2c\_write(init\_table[i], init\_table[i+1]);

if(0 != err)

STTBX\_Print((L"1516 init %d error!\n", i));

}

// #if 1 // debug only

// for(i=0;i<0xFF;i++)

// {

// mxd1516\_i2c\_read(i,&u8Data);

// printf("Tuner address %x, Tuner data %x \n", i, u8Data);

// }

// #endif

return;

}

# 【1-0-1】UINT32 mxd1516\_i2c\_write(UINT8 reg\_addr, UINT8 reg\_val )

{

UINT8 WriteData[2];

WriteData[0] = reg\_addr;

//WriteData[1] = &reg\_val;

WriteData[1] = reg\_val;

## 【API】Write\_I2C(MXD1516\_ADDRESS,2,WriteData);

//IIC\_WRITE(0xD4,&reg\_addr,1,&reg\_val,1);

return 0;

}

# 【2-0-1】UINT32 tun\_mxd1516\_control(UINT32 freq, UINT8 bandwidth)

{

UINT32 sym;

UINT32 result;

UINT8 retryCns=0; //Ben 100512#3

//TUN\_MXD1516\_PRINTF("%s--%d!\n",\_\_FUNCTION\_\_,\_\_LINE\_\_);

//printf("bandwidth=%d\n",bandwidth); //Ben 100511#2

switch(bandwidth)

{

case 8:

case 7:

case 6:

case 5:

sym=bandwidth\*1000;

break;

default:

sym=8000; // default is 7.56Mhz symbolrate

break;

}

//Ben 100512#3

## 【2-0-1-0】result = mxd1516\_set\_freq(freq\*1000,sym);

//printf("result=%d\n",result);

if(-1 == result)

{

while(retryCns < 5)

{

retryCns++;

## 【2-0-1-0】result = mxd1516\_set\_freq(freq\*1000,sym);

if(0 == result)

break;

}

}

return 0;

}

# 【2-0-1-0】UINT32 mxd1516\_set\_freq(unsigned int freq\_hz, unsigned int bandwidth\_khz)

{

unsigned int Flo; //KHz

unsigned int Fref= 27; // F\_ref( MHz )

unsigned char Scal= 18; // Bit Num of Fractional-N of PLL

unsigned char reg\_val=0;

unsigned char reg\_val\_11=0;

unsigned char reg\_val\_13=0;

unsigned int vco\_sel=0;

unsigned int Fvco=0; //KHz

unsigned char band\_type;

unsigned char nDiv2,nDiv1,frac3,frac2,frac1;

unsigned int quotientN,quatientFrac;

unsigned char agc\_sel = 0;

unsigned char agc\_pos = 0;

unsigned int freq\_offset=0;

unsigned int i=0;

//UINT8 u8Data;

Flo = (freq\_hz+freq\_offset)/1000; //KHz

/\*Set lpf bandwidth\*/

switch(bandwidth\_khz)//LPF BW (kHz);

{

case 8000:

band\_type = 0;

break;

case 430:

band\_type = 1;

break;

case 2000:

band\_type = 2;

break;

case 1300:

band\_type = 3;

break;

case 5000:

band\_type = 5;

break;

case 6000:

band\_type = 6;

break;

case 7000:

band\_type = 7;

break;

default:

band\_type = 0;

break;

}

/\*Set lpf bandwidth\*/

## 【2-0-1-000】mxd1516\_i2c\_read( 0x26,&reg\_val);

reg\_val = (reg\_val&0xf8);

reg\_val = reg\_val|band\_type;

## 【1-0-1】mxd1516\_i2c\_write( 0x26, reg\_val);

//=============initializing===========

mxd1516\_i2c\_write( 0x02,0x00 ); 【1-0-1】

mxd1516\_i2c\_write( 0x00,0x08 ); 【1-0-1】//reset sdm & afc change in 20110925 V1.4

mxd1516\_i2c\_write( 0x00,0x0e ); 【1-0-1】 //release sdm & afc change in 20110925 V1.4

mxd1516\_i2c\_write( 0x12,0x05 ); 【1-0-1】

mxd1516\_i2c\_write( 0x2d,0x28 ); 【1-0-1】

mxd1516\_i2c\_write( 0x1e,0x10 ); 【1-0-1】

/\*band sel\*/

if( (Flo<= 39840)&&(Flo>28120)) /\*VHF\*/

{

Fvco = 128\*Flo;

//Fref=Fref/128;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xee;

reg\_val\_13 = 0x01;

}

else if(( Flo<=56250 )&&( Flo>39840 )) /////// VHF //////////

{

Fvco = 96\*Flo;

//Fref=Fref/96;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xed;

reg\_val\_13 = 0x01;

}

else if( ( Flo<=79680 )&&( Flo>56250 )) /////// UHF //////////

{

Fvco = 64\*Flo;

//Fref=Fref/72;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xee;

reg\_val\_13 = 0x02;

}

else if( ( Flo<=112500 )&&( Flo>79680 )) /////// UHF //////////

{

Fvco = 48\*Flo;

//Fref=Fref/64;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xed;

reg\_val\_13 = 0x02;

}

else if( ( Flo<=159400 )&&( Flo>112500 ) )

{

Fvco = 32\*Flo;

//Fref=Fref/48;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xee;

reg\_val\_13 = 0x04;

}

else if( ( Flo<=225000 )&&( Flo>159400 ))

{

Fvco = 24\*Flo;

//Fref=Fref/36;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xed;

reg\_val\_13 = 0x04;

}

else if (( Flo<=318700 )&&( Flo>225000 ))

{

Fvco = 16\*Flo;

//Fref=Fref/32;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xee;

reg\_val\_13 = 0x10;

}

else if( ( Flo<=450000 )&&( Flo>318700 ))

{

Fvco = 12\*Flo;

//Fref=Fref/24;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xed;

reg\_val\_13 = 0x10;

}

else if( ( Flo<=637500 )&&( Flo>450000 ))

{

Fvco = 8\*Flo;

//Fref=Fref/18;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xee;

reg\_val\_13 = 0x08;

}

else if( Flo>637500 )

{

Fvco = 6\*Flo;

//Fref=Fref/8;

//reg\_val\_11 = 0x2f;

reg\_val\_11 = 0xed;

reg\_val\_13 = 0x08;

}

mxd1516\_i2c\_write( 0x11,reg\_val\_11); 【1-0-1】

mxd1516\_i2c\_write( 0x13,reg\_val\_13); 【1-0-1】

//icp config

vco\_sel = 5427;

if( ( Fvco<=363100 )&&( Fvco > 330000))

vco\_sel = 5427;

else if(( Fvco<=3796000 )&&( Fvco > 3631000))

vco\_sel = 5042;

else if(( Fvco<=3971000 )&&( Fvco > 3796000))

vco\_sel = 4643;

else if(( Fvco<=4293000 )&&( Fvco > 3971000))

vco\_sel = 4184;

else if(( Fvco<=4528000 )&&( Fvco > 4293000))

vco\_sel = 3501;

else if(( Fvco<=4777000 )&&( Fvco > 4528000))

vco\_sel = 3318;

else if(( Fvco<=5164000 )&&( Fvco > 4777000))

vco\_sel = 2946;

else if(( Fvco<=5431000 )&&( Fvco > 5164000))

vco\_sel = 2516;

else if(( Fvco > 5431000))

vco\_sel = 2351;

## 【2-0-1-000】 mxd1516\_i2c\_read( 0x10, &reg\_val );

reg\_val &= 0xf0;

reg\_val |= (0x0f & ((vco\_sel\*774+1000)/(Fref\*1000\*20) - 2));

//VCO 0x21 Setting

## 【1-0-1】mxd1516\_i2c\_write( 0x10,reg\_val);

// Fractional-N PLL Settings

## 【2-0-1-111】UTIL\_DivToUintScale( Fvco, Fref\*1000,&quotientN,&quatientFrac,Scal);

/\* 24bits quatientFrac \*/

quatientFrac += 1;

quatientFrac = quatientFrac>>1;

nDiv2 = (unsigned char)(quotientN>>8);

nDiv1 = (unsigned char)quotientN;

frac3 = (unsigned char) ((quatientFrac>>16) & 0xff );

frac2 = (unsigned char) (quatientFrac >> 8) & 0xff;

frac1 = (unsigned char) (quatientFrac & 0xff);

## 【1-0-1】mxd1516\_i2c\_write( 0x14,nDiv2);

mxd1516\_i2c\_write( 0x15,nDiv1); 【1-0-1】

mxd1516\_i2c\_write( 0x16,frac3); 【1-0-1】

mxd1516\_i2c\_write( 0x17,frac2); 【1-0-1】

mxd1516\_i2c\_write( 0x18,frac1); 【1-0-1】

mxd1516\_i2c\_write( 0x03,0x08); 【1-0-1】//start AFC\_FSM, system enters into operation mode

## 【2-0-1-0-00】mxd\_sleep(10);

mxd1516\_i2c\_write( 0x25, 0x01); 【1-0-1】// fd enable

mxd1516\_i2c\_write( 0x00, 0x06); 【1-0-1】 //change in 20110925 V1.4

mxd1516\_i2c\_write( 0x00, 0x0e); 【1-0-1】 //release fd changed in 20110925 V1.4

mxd1516\_i2c\_write( 0x5c, 0x00); 【1-0-1】

mxd1516\_i2c\_write( 0x5d, 0x00); 【1-0-1】

mxd1516\_i2c\_write( 0x3c, 0x06); 【1-0-1】// changed in 20110925 V1.4

mxd1516\_i2c\_write( 0x00, 0x0e); 【1-0-1】

mxd1516\_i2c\_write( 0x00, 0x0f); 【1-0-1】

// mxd1516\_i2c\_write( 0x03, 0x01);

// add lo&fd lock protectoin

for(i=0;i<50;i++) // add 5 times retry

{

## 【2-0-1-0-00】 mxd\_sleep(5);

## 【2-0-1-000】mxd1516\_i2c\_read( 0x6d, &reg\_val );

if(0xc0!=(reg\_val&0xc0))

{

if(0x80!=(reg\_val&0x80)) //restart afc

{

## 【1-0-1】mxd1516\_i2c\_write( 0x00,0x09); //reset afc )

mxd1516\_i2c\_write( 0x00,0x0f); 【1-0-1】//release afc

mxd1516\_i2c\_write( 0x03,0x08); 【1-0-1】//start AFC\_FSM, system

mxd1516\_i2c\_write( 0x02,0x0); 【1-0-1】

}

if(0x40!=(reg\_val&0x40))//restart fd

{

mxd1516\_i2c\_write( 0x0,0x07); 【1-0-1】//reset fd

mxd1516\_i2c\_write( 0x0,0x0f); 【1-0-1】//release fd

}

}

else

{

break;

}

}

## 【2-0-1-0-00】mxd\_sleep(10); //ms

## 【2-0-1-000】 mxd1516\_i2c\_read( 0x6d, &reg\_val );

if(0x80!=(reg\_val&0x80))

return -1;

for(agc\_sel=0;agc\_sel<(sizeof(tun\_agc\_boundary)/sizeof(tun\_agc\_boundary[0]));agc\_sel++)

{

if( (freq\_hz<=(tun\_agc\_boundary[agc\_sel][1])\*1000000)

&&(freq\_hz>=(tun\_agc\_boundary[agc\_sel][0]\*1000000)))

{

agc\_pos = agc\_sel;

break;

}

else

{

agc\_pos = 1;

}

}

## 【2-0-1-222】mxd1516\_set\_agc( agc\_pos );

// #if 1 // debug only

// for(i=0;i<0xFF;i++)

// {

// mxd1516\_i2c\_read(i,&u8Data);

// printf("Tuner address %x, Tuner data %x \n", i, u8Data);

// }

// #endif

## 【1-0-1】mxd1516\_i2c\_write( 0x0a, 0x7e); //changed in 20110921 V1.4

mxd1516\_i2c\_write( 0x03, 0x01); 【1-0-1】 //changed in 20110921 V1.4

## 【2-0-1-0-00】mxd\_sleep(100);

mxd1516\_i2c\_write( 0x0a, 0x7f); 【1-0-1】 //changed in 20110921 V1.4

mxd1516\_i2c\_write( 0x3c, 0x01); 【1-0-1】 //changed in 20110921 V1.4

return 0;

}

# 【2-0-1-000】UINT32 mxd1516\_i2c\_read(UINT8 reg\_addr, UINT8 \*reg\_val )

{

//IIC\_READ(0xD4,&reg\_addr,1,reg\_val,1);

## 【API】Write\_I2C(MXD1516\_ADDRESS,1,&reg\_addr);

## 【API】Read\_I2C(MXD1516\_ADDRESS,1,reg\_val);

return 0;

//return 0;

}

# 【2-0-1-222】void mxd1516\_set\_agc(unsigned char agc\_sel)

{

unsigned char agc\_table\_0[] = //VBAND

{

0x0f, 0x1f,

0x20, 0x3f,

0x21, 0x0f,

0x22, 0x00,

0x26, 0x07,

0xbe, 0x08,

0xbf, 0x24,

0xc0, 0x12,

0xE8, 0x53,

0xE9, 0x42,

0xEA, 0x46,

0xEB, 0x48,

0xEC, 0x3d,

0xED, 0x35,

0xEE, 0x38,

0xEF, 0x3e,

0xCE, 0x6f,

0xCF, 0x4b,

0xD0, 0x3c,

0xD1, 0x3c,

0xD3, 0x36,

0xD4, 0x1a,

0xD5, 0x25,

0xD6, 0x25,

0xF0, 0x14,

0xF1, 0x19,

0xF2, 0x10,

0xF3, 0x10,

0xF4, 0x0d,

0xF5, 0x0d,

0xF6, 0x07,

0xF7, 0x06,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x3b,

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x34,

0x5c, 0xa6,

0x5d, 0xe5,

0x5d, 0xe5,

};

unsigned char agc\_table\_1[] = //VBAND

{

0x0f, 0x1f,

0x20, 0x3f,

0x21, 0x0b,

0x22, 0x00,

0x26, 0x07,

0xbe, 0x08,

0xbf, 0x24,

0xc0, 0x12,

0xE8, 0x53,

0xE9, 0x42,

0xEA, 0x46,

0xEB, 0x48,

0xEC, 0x3d,

0xED, 0x35,

0xEE, 0x38,

0xEF, 0x3e,

0xCE, 0x6f,

0xCF, 0x4b,

0xD0, 0x3c,

0xD1, 0x3c,

0xD3, 0x36,

0xD4, 0x1a,

0xD5, 0x25,

0xD6, 0x25,

0xF0, 0x14,

0xF1, 0x19,

0xF2, 0x10,

0xF3, 0x10,

0xF4, 0x0d,

0xF5, 0x0d,

0xF6, 0x07,

0xF7, 0x06,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x3b,

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x34,

0x5c, 0xa6,

0x5d, 0xe5,

0x5d, 0xe5,

};

unsigned char agc\_table\_2[] = //474-538

{

0x0f, 0xff,

0x20, 0x3f,

0x21, 0x0c,

0x22, 0x0f,

0x26, 0x00,

0xE8, 0x5f,

0xE9, 0x4e,

0xEA, 0x4e,

0xEB, 0x51,

0xEC, 0x4d,

0xED, 0x47,

0xEE, 0x49,

0xEF, 0x54,

0xCE, 0x7c,

0xCF, 0x5d,

0xD0, 0x4e,

0xD1, 0x4e,

0xD3, 0x2e,

0xD4, 0x1b,

0xD5, 0x2d,

0xD6, 0x2d,

0xF0, 0x11,

0xF1, 0x1a,

0xF2, 0x10,

0xF3, 0x10,

0xF4, 0x11,

0xF5, 0x0d,

0xF6, 0x0b,

0xF7, 0x03,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x40,

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x2c,

0x5c, 0xa6,

0x5d, 0xe8,

0x5d, 0xe8,

};

unsigned char agc\_table\_3[] = //546-594

{

0x0f, 0xff,

0x20, 0x3f,

0x21, 0x0a,

0x22, 0x0c,

0x26, 0x00,

0xE8, 0x66,

0xE9, 0x55,

0xEA, 0x53,

0xEB, 0x56,

0xEC, 0x53,

0xED, 0x4b,

0xEE, 0x50,

0xEF, 0x5b,

0xCE, 0x81,

0xCF, 0x62,

0xD0, 0x51,

0xD1, 0x51,

0xD3, 0x2e,

0xD4, 0x18,

0xD5, 0x2d,

0xD5, 0x2d,

0xF0, 0x12,

0xF1, 0x1c,

0xF2, 0x0f,

0xF3, 0x10,

0xF4, 0x10,

0xF5, 0x0f,

0xF6, 0x0a,

0xF7, 0x03,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x40,

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x2c,

0x5c, 0xa6,

0x5d, 0xd9,

0x5d, 0xd9,

};

unsigned char agc\_table\_4[] = //602-642

{

0x0f, 0xff,

0x20, 0x3f,

0x21, 0x08,

0x22, 0x0a,

0x26, 0x00,

0xE8, 0x64,

0xE9, 0x59,

0xEA, 0x57,

0xEB, 0x61,

0xEC, 0x5b,

0xED, 0x56,

0xEE, 0x5b,

0xEF, 0x65,

0xCE, 0x84,

0xCF, 0x64,

0xD0, 0x54,

0xD1, 0x54,

0xD3, 0x2d,

0xD4, 0x16,

0xD5, 0x2b,

0xD6, 0x2b,

0xF0, 0x09,

0xF1, 0x16,

0xF2, 0x14,

0xF3, 0x0f,

0xF4, 0x10,

0xF5, 0x10,

0xF6, 0x10,

0xF7, 0x10,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x40,

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x2c,

0x5c, 0xa6,

0x5d, 0xd5,

0x5d, 0xd5,

};

unsigned char agc\_table\_5[] = //648-690

{

0x0f, 0xff,

0x20, 0x3f,

0x21, 0x06,

0x22, 0x08,

0x26, 0x00,

0xE8, 0x6b,

0xE9, 0x59,

0xEA, 0x5a,

0xEB, 0x5d,

0xEC, 0x5d,

0xED, 0x60,

0xEE, 0x60,

0xEF, 0x63,

0xCE, 0x85,

0xCF, 0x6b,

0xD0, 0x55,

0xD1, 0x55,

0xD3, 0x26,

0xD4, 0x1b,

0xD5, 0x2f,

0xD6, 0x2f,

0xF0, 0x10,

0xF1, 0x19,

0xF2, 0x10,

0xF3, 0x10,

0xF4, 0x0f,

0xF5, 0x0f,

0xF6, 0x0f,

0xF7, 0x09,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x40,

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x2c,

0x5c, 0xa6,

0x5d, 0xd9,

0x5d, 0xd9,

};

unsigned char agc\_table\_6[] = //698-770

{

0x0f, 0xff,

0x20, 0x3f,

0x21, 0x04,

0x22, 0x06,

0x26, 0x00,

0xE8, 0x70,

0xE9, 0x5e,

0xEA, 0x5f,

0xEB, 0x62,

0xEC, 0x5a,

0xED, 0x61,

0xEE, 0x65,

0xEF, 0x6b,

0xCE, 0x88,

0xCF, 0x68,

0xD0, 0x5e,

0xD1, 0x5e,

0xD3, 0x28,

0xD4, 0x18,

0xD5, 0x33,

0xD5, 0x33,

0xF0, 0x0a,

0xF1, 0x14,

0xF2, 0x10,

0xF3, 0x0e,

0xF4, 0x0e,

0xF5, 0x0e,

0xF6, 0x0e,

0xF7, 0x0e,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x40,

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x2c,

0x5c, 0xa6,

0x5d, 0xe0,

0x5d, 0xe0,

};

unsigned char agc\_table\_7[] = //778-858

{

0x0f, 0xff,

0x20, 0x3f,

0x21, 0x04,

0x22, 0x06,

0x26, 0x00,

0xE8, 0x7d,

0xE9, 0x68,

0xEA, 0x68,

0xEB, 0x6b,

0xEC, 0x72,

0xED, 0x6d,

0xEE, 0x72,

0xEF, 0x77,

0xCE, 0x8c,

0xCF, 0x6d,

0xD0, 0x64,

0xD1, 0x64,

0xD3, 0x25,

0xD4, 0x18,

0xD5, 0x34,

0xD6, 0x34,

0xF0, 0x08,

0xF1, 0x12,

0xF2, 0x10,

0xF3, 0x0d,

0xF4, 0x0d,

0xF5, 0x0d,

0xF6, 0x0d,

0xF7, 0x0d,

0xd8, 0x73,

0xd9, 0x66,

0xda, 0x56,

0xdb, 0x46,

0xdc, 0x33,

0xdd, 0x23,

0xde, 0x13,

0xdf, 0x00,

0xE0, 0x14,

0xE1, 0x33,

0xE2, 0x60,

0xE3, 0x60,

0xE4, 0x60,

0xE5, 0x60,

0xE6, 0x30,

0xE7, 0x30,

0xF8, 0x14,

0xF9, 0x33,

0xFA, 0x60,

0xFB, 0x60,

0xFC, 0x60,

0xFD, 0x60,

0xFE, 0x30,

0xFF, 0x30,

0x50, 0x42, //20110925 V1.4

0x51, 0x2c,

0x52, 0x2c,

0x53, 0x2c,

0x5c, 0xa6,

0x5d, 0xda,

0x5d, 0xda,

};

unsigned int i;

switch(agc\_sel)

{

case 0:

for(i=0;i<sizeof(agc\_table\_0);)

{

## 【1-0-1】mxd1516\_i2c\_write( agc\_table\_0[i],agc\_table\_0[i+1]);

i+=2;

}

break;

case 1:

for(i=0;i<sizeof(agc\_table\_1);)

{

mxd1516\_i2c\_write( agc\_table\_1[i],agc\_table\_1[i+1]); 【1-0-1】

i+=2;

}

break;

case 2:

for(i=0;i<sizeof(agc\_table\_2);)

{

mxd1516\_i2c\_write( agc\_table\_2[i],agc\_table\_2[i+1]); 【1-0-1】

i+=2;

}

break;

case 3:

for(i=0;i<sizeof(agc\_table\_3);)

{

mxd1516\_i2c\_write( agc\_table\_3[i],agc\_table\_3[i+1]); 【1-0-1】

i+=2;

}

break;

case 4:

for(i=0;i<sizeof(agc\_table\_4);)

{

mxd1516\_i2c\_write( agc\_table\_4[i],agc\_table\_4[i+1]); 【1-0-1】

i+=2;

}

break;

case 5:

for(i=0;i<sizeof(agc\_table\_5);)

{

mxd1516\_i2c\_write( agc\_table\_5[i],agc\_table\_5[i+1]); 【1-0-1】

i+=2;

}

break;

case 6:

for(i=0;i<sizeof(agc\_table\_6);)

{

mxd1516\_i2c\_write( agc\_table\_6[i],agc\_table\_6[i+1]); 【1-0-1】

i+=2;

}

break;

case 7:

for(i=0;i<sizeof(agc\_table\_7);)

{

mxd1516\_i2c\_write( agc\_table\_7[i],agc\_table\_7[i+1]); 【1-0-1】

i+=2;

}

break;

default:

break;

}

return ;

}

# 【2-0-1-111】void UTIL\_DivToUintScale( unsigned int inDividend,

unsigned int inDivisor,

unsigned int \* outQuotientN,

unsigned int \* outQuatientFrac,

unsigned char Scale)

{

unsigned int remainder;

unsigned int quotientN;

unsigned int quatientFrac;

unsigned char i;

unsigned int divisor = inDivisor;

unsigned int dividend = inDividend;

if (0 == divisor)

{

return ;

}

quotientN = dividend/divisor;

remainder = dividend - quotientN\*divisor;

/\* @here remainder < divisor \*/

if (remainder>=0x80000000)

{

remainder = remainder >> 1;

divisor = divisor >> 1;

}

/\* @here remainder <= divisor \*/

quatientFrac = 0;

if (remainder != divisor)

{

for (i=0;i<(Scale+1);i++)

{

remainder = remainder<<1; /\* remainder \*= 2; \*/

if ( remainder >= divisor )

{

quatientFrac += 1; /\* BS\_ (quatientFrac,bit0) \*/

remainder -= divisor;

}

quatientFrac = quatientFrac<<1;

}

if(0xffffff==quatientFrac )

{

quotientN++;

quatientFrac = 0;

}

else

{

quatientFrac = (quatientFrac+1)>>1;

}

}

else /\* remainder == divisor \*/

{

quotientN++;

quatientFrac = 0;

}

\*outQuotientN = quotientN;

\*outQuatientFrac = quatientFrac;

return ;

}

# 【2-0-1-0-00】void mxd\_sleep(UINT32 millisecond )

{

//UINT32 i=0;

//for(i=millisecond;i>0;i--);

Sleep(millisecond);

}