#include "input.h"

#include "const.h"

#include "configure.h"

#include "p18f25k80.h"

INT8U Test\_Point01; // for code test

SUPPLY\_VOLTAGE\_CONTROL IGN\_Feedback;

SUPPLY\_VOLTAGE\_CONTROL BATT\_Feedback;

void Init\_MCU\_AD(void)

{

PMD1bits.ADCMD = 0;

ADCON0 = 0x00;

ADCON1 =0X00; //右对齐

ADCON2 =0XA0;

PIR1bits.ADIF=0;

PIE1bits.ADIE=0;

ANCON0 =0X00;

ANCON1= 0X00;

ANCON0=MARK\_ADL;

ANCON1=MARK\_ADH;

}

INT16U Get\_AD\_Word(INT8U CH)

{

INT16U n;

ADCON0 = CH;

ADCON0bits.ADON = 1;

for(n=0;n<50;n++)

{

Nop();

}

ADCON0bits.GO = 1;

while(ADCON0bits.GO);

n = ADRESH\*256 + ADRESL;

ADCON0bits.ADON = 0;

n >>=2;

return n;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INT16U Feedback\_GetValue(INT8U Channel)

{

if (Channel==BATT\_FEEDBACK) return BATT\_Feedback.Value\_ECU;

else if (Channel==IGN\_FEEDBACK) return IGN\_Feedback.Value\_ECU;

else return 0 ;

}

INT8U Feedback\_GetSatus(INT8U Channel)

{

if (Channel==BATT\_FEEDBACK)

return BATT\_Feedback.Status;

else if (Channel==IGN\_FEEDBACK)

return IGN\_Feedback.Status;

else

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Input\_Sensor\_Proc(void)

{

IGN\_Feedback\_Detect();

// Test\_Point01=IGN\_Feedback.Status;

BATT\_Feedback\_Detect();

}

INT16U IGN\_Feedback\_AdtoVoltage(INT16U AdValue)

{ return AdValue; }

INT16U BATT\_Feedback\_AdtoVoltage(INT16U AdValue)

{ return AdValue; }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* IGN input and operating

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void IGN\_Feedback\_Init( )

{

IGN\_Feedback.Init=1;

IGN\_Feedback.Status=C\_IGN\_NORMAL ;

IGN\_Feedback.Interval=C\_AD\_MAX\_NUM \*C\_SYSTEM\_TIMER\_INTERVAL;

//IGN\_Feedback.Value\_ECU=0XFF; /\*\*TO think deeply\*\*\*\*\*/

}

void IGN\_Feedback\_Detect()

{

INT16U AD, DAT;

INT8U i;

AD=Get\_AD\_Word(CH\_IGN\_FB);

DAT= IGN\_Feedback\_AdtoVoltage(AD);

if(IGN\_Feedback.Init)

{

IGN\_Feedback.Init=0;

IGN\_Feedback.Status=C\_IGN\_NORMAL;

IGN\_Feedback.Interval=C\_AD\_MAX\_NUM\*C\_SYSTEM\_TIMER\_INTERVAL;

IGN\_Feedback.TimerOutCountOverVoltage =C\_TIMER\_OVERVOLTAGE;

IGN\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE ;

IGN\_Feedback.Value\_SUM=0;

IGN\_Feedback.Count=0;

for (i=0;i<C\_SMOOTHING\_COUNT;i++)

{

IGN\_Feedback.Buffer[i]=DAT;

IGN\_Feedback.Value\_SUM +=DAT;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IGN\_Feedback.Value\_SUM -= IGN\_Feedback.Buffer[IGN\_Feedback.Count];

IGN\_Feedback.Value\_SUM += DAT;

IGN\_Feedback.Buffer[IGN\_Feedback.Count]= DAT;

IGN\_Feedback.Count =(IGN\_Feedback.Count+1)%C\_SMOOTHING\_COUNT;

DAT=(INT16U)IGN\_Feedback.Value\_SUM/C\_SMOOTHING\_COUNT;

IGN\_Feedback.Value\_ECU=DAT;

i=0;

if(i) //when the head if off ,the voltage diagnostic is not applied

{

IGN\_Feedback.Status=C\_IGN\_NORMAL;

IGN\_Feedback.TimerOutCountOverVoltage =C\_TIMER\_OVERVOLTAGE;

IGN\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE ;

return ;

}

switch (IGN\_Feedback.Status)

{

case C\_IGN\_NORMAL:

if(IGN\_Feedback.Value\_ECU>C\_VOLTHREHOLD\_OVERVOLTAGE)

{

if( IGN\_Feedback.TimerOutCountOverVoltage> IGN\_Feedback.Interval )

{ IGN\_Feedback.TimerOutCountOverVoltage -=IGN\_Feedback.Interval; }

else

{ IGN\_Feedback.TimerOutCountOverVoltage =0; }

if(IGN\_Feedback.TimerOutCountOverVoltage ==0)

{

//!!!!!!speed is over a value the overtage is effect

IGN\_Feedback.Status=C\_IGN\_0VERVOLTAGE;

IGN\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_OVERVOLTAGERETURN ;

//system off

}

}

else

{

IGN\_Feedback.TimerOutCountOverVoltage=C\_TIMER\_OVERVOLTAGE;

}

if(IGN\_Feedback.Value\_ECU < C\_VOLTHREHOLD\_UNDERVOLTAGE)

{

if( IGN\_Feedback.TimerOutCountUnderVoltage> IGN\_Feedback.Interval )

{ IGN\_Feedback.TimerOutCountUnderVoltage -=IGN\_Feedback.Interval; }

else

{ IGN\_Feedback.TimerOutCountUnderVoltage =0; }

if(IGN\_Feedback.TimerOutCountUnderVoltage ==0)

{

//!!!!!!!!!!!!!!enginee is on, a value the overtage is effect

IGN\_Feedback.Status=C\_IGN\_UNDERVOLTAGE;

IGN\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_UNDERVOLTAGERETURN;

//system off

}

}

else

{

IGN\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE;

}

break;

case C\_IGN\_UNDERVOLTAGE:

if(IGN\_Feedback.Value\_ECU>=C\_VOLTHREHOLD\_UNDERVOLTAGERETURN)

{

if(IGN\_Feedback.TimerOutCountReturnVoltage>IGN\_Feedback.Interval)

{IGN\_Feedback.TimerOutCountReturnVoltage -= IGN\_Feedback.Interval;}

else {IGN\_Feedback.TimerOutCountReturnVoltage=0;}

if(IGN\_Feedback.TimerOutCountReturnVoltage==0)

{

IGN\_Feedback.Status=C\_IGN\_NORMAL;

//system on

IGN\_Feedback.TimerOutCountOverVoltage =C\_TIMER\_OVERVOLTAGE;

IGN\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE ;

//PutMsg\_Diag\_IGNStatus(0);

}

}

else

{

IGN\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_UNDERVOLTAGERETURN;

}

break;

case C\_IGN\_0VERVOLTAGE:

if(IGN\_Feedback.Value\_ECU<=C\_VOLTHREHOLD\_OVERVOLTAGERETURN)

{

if(IGN\_Feedback.TimerOutCountReturnVoltage>IGN\_Feedback.Interval)

{IGN\_Feedback.TimerOutCountReturnVoltage -= IGN\_Feedback.Interval;}

else{IGN\_Feedback.TimerOutCountReturnVoltage=0;}

if(IGN\_Feedback.TimerOutCountReturnVoltage==0)

{

IGN\_Feedback.Status=C\_IGN\_NORMAL;

//system on

IGN\_Feedback.TimerOutCountOverVoltage = C\_TIMER\_OVERVOLTAGE;

IGN\_Feedback.TimerOutCountUnderVoltage = C\_TIMER\_UNDERVOLTAGE ;

//PutMsg\_Diag\_IGNStatus(0);

}

}

else

{

IGN\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_OVERVOLTAGERETURN;

}

break;

default:

IGN\_Feedback.Status=C\_IGN\_NORMAL;

break;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* BATT input and operating

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void BATT\_Feedback\_Init( )

{

BATT\_Feedback.Init=1;

BATT\_Feedback.Status=C\_BATT\_NORMAL ;

BATT\_Feedback.Interval=C\_AD\_MAX\_NUM \*C\_SYSTEM\_TIMER\_INTERVAL;

//BATT\_Feedback.Value\_ECU=0XFF; /\*\*TO think deeply\*\*\*\*\*/

}

void BATT\_Feedback\_Detect()

{

INT16U AD, DAT;

INT8U i;

AD=Get\_AD\_Word(CH\_BAT\_FB);

DAT= BATT\_Feedback\_AdtoVoltage(AD);

if(BATT\_Feedback.Init)

{

BATT\_Feedback.Init=0;

BATT\_Feedback.Status=C\_BATT\_NORMAL;

BATT\_Feedback.Interval=C\_AD\_MAX\_NUM\*C\_SYSTEM\_TIMER\_INTERVAL;

BATT\_Feedback.TimerOutCountOverVoltage =C\_TIMER\_OVERVOLTAGE;

BATT\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE ;

BATT\_Feedback.Value\_SUM=0;

BATT\_Feedback.Count=0;

for (i=0;i<C\_SMOOTHING\_COUNT;i++)

{

BATT\_Feedback.Buffer[i]=DAT;

BATT\_Feedback.Value\_SUM +=DAT;

}

}

BATT\_Feedback.Value\_SUM -= BATT\_Feedback.Buffer[BATT\_Feedback.Count];

BATT\_Feedback.Value\_SUM += DAT;

BATT\_Feedback.Buffer[BATT\_Feedback.Count]= DAT;

BATT\_Feedback.Count =(BATT\_Feedback.Count+1)%C\_SMOOTHING\_COUNT;

DAT=(INT16U)BATT\_Feedback.Value\_SUM/C\_SMOOTHING\_COUNT;

BATT\_Feedback.Value\_ECU=DAT;

i=0;

if(i)

//when the head if off ,the voltage diagnostic is not applied

{

BATT\_Feedback.Status=C\_BATT\_NORMAL;

BATT\_Feedback.TimerOutCountOverVoltage =C\_TIMER\_OVERVOLTAGE;

BATT\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE ;

return ;

}

switch (BATT\_Feedback.Status)

{

case C\_BATT\_NORMAL:

if(BATT\_Feedback.Value\_ECU>C\_VOLTHREHOLD\_OVERVOLTAGE)

{

if( BATT\_Feedback.TimerOutCountOverVoltage> BATT\_Feedback.Interval )

{ BATT\_Feedback.TimerOutCountOverVoltage -=BATT\_Feedback.Interval; }

else

{ BATT\_Feedback.TimerOutCountOverVoltage =0; }

if(BATT\_Feedback.TimerOutCountOverVoltage ==0)

{

//!!!!!!speed is over a value the overtage is effect

BATT\_Feedback.Status=C\_BATT\_0VERVOLTAGE;

BATT\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_OVERVOLTAGERETURN ;

//system off

}

}

else

{

BATT\_Feedback.TimerOutCountOverVoltage=C\_TIMER\_OVERVOLTAGE;

}

if(BATT\_Feedback.Value\_ECU < C\_VOLTHREHOLD\_UNDERVOLTAGE)

{

if( BATT\_Feedback.TimerOutCountUnderVoltage> BATT\_Feedback.Interval )

{ BATT\_Feedback.TimerOutCountUnderVoltage -=BATT\_Feedback.Interval; }

else

{ BATT\_Feedback.TimerOutCountUnderVoltage =0; }

if(BATT\_Feedback.TimerOutCountUnderVoltage ==0)

{

//!!!!!!!!!!!!!!enginee is on, a value the overtage is effect

BATT\_Feedback.Status=C\_BATT\_UNDERVOLTAGE;

BATT\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_UNDERVOLTAGERETURN;

//system off

}

}

else

{

BATT\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE;

}

break;

case C\_BATT\_UNDERVOLTAGE:

if(BATT\_Feedback.Value\_ECU>=C\_VOLTHREHOLD\_UNDERVOLTAGERETURN)

{

if(BATT\_Feedback.TimerOutCountReturnVoltage>BATT\_Feedback.Interval)

{BATT\_Feedback.TimerOutCountReturnVoltage -= BATT\_Feedback.Interval;}

else {BATT\_Feedback.TimerOutCountReturnVoltage=0;}

if(BATT\_Feedback.TimerOutCountReturnVoltage==0)

{

BATT\_Feedback.Status=C\_BATT\_NORMAL;

//system on

BATT\_Feedback.TimerOutCountOverVoltage =C\_TIMER\_OVERVOLTAGE;

BATT\_Feedback.TimerOutCountUnderVoltage=C\_TIMER\_UNDERVOLTAGE ;

}

}

else

{

BATT\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_UNDERVOLTAGERETURN;

}

break;

case C\_BATT\_0VERVOLTAGE:

if(BATT\_Feedback.Value\_ECU<=C\_VOLTHREHOLD\_OVERVOLTAGERETURN)

{

if(BATT\_Feedback.TimerOutCountReturnVoltage>BATT\_Feedback.Interval)

{BATT\_Feedback.TimerOutCountReturnVoltage -= BATT\_Feedback.Interval;}

else{BATT\_Feedback.TimerOutCountReturnVoltage=0;}

if(BATT\_Feedback.TimerOutCountReturnVoltage==0)

{

BATT\_Feedback.Status=C\_BATT\_NORMAL;

//system on

BATT\_Feedback.TimerOutCountOverVoltage = C\_TIMER\_OVERVOLTAGE;

BATT\_Feedback.TimerOutCountUnderVoltage = C\_TIMER\_UNDERVOLTAGE ;

}

}

else

{

BATT\_Feedback.TimerOutCountReturnVoltage=C\_TIMER\_OVERVOLTAGERETURN;

}

break;

default:

BATT\_Feedback.Status=C\_BATT\_NORMAL;

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/