

```

#ifndef __HSK_OP_H

#define __HSK_OP_H

static char *hsk_op_SccsId = "@(#) hskop.h 1.6 6/14/93";

#include "defs.h"
#include "bitmask.h"

#define OUT_OF_RANGE 9999.

#define ALARM TRUE
#define ITSOK FALSE

#define RELEASED 2
#define INSERTED 1

/*-----*/

#define E_INT_NUMBER(value) (value)

/*-----*/

#define E_MASK_STATUS(value) ((value)&TWO_B)
#define E_BB_STATUS(value) ((SHIFT((value),-2))&TWO_B)

/* 12 bits : 0 -> -180 , 4096 -> 180 ) */

/* revision 18/5/92 sign of angle in 0x0800,
was on 0x0100
*/

#define ASTEP (360./4096)

/*
#define E_LS_ANGLE(value) ((value > 0) ? ((value >> 4) * ASTEP)\
: (-1 * (((-value) >> 4) * ASTEP)))
*/
#define E_LS_ANGLE(value) ((short)((value)&0xfff0)/16 *ASTEP)

/*-----*/

#ifdef REV_0
#define E_GAIN_BITS(value) (((value)&1)<<2)+(((value)&4)>>2)+ \
((value)&2))
#else
#define E_GAIN_BITS(value) (value)
#endif

#define E_SLOW_GAIN_0(value) ((value)&FOUR_B)
#define E_SLOW_GAIN_1(value) ((SHIFT((value),-4))&FOUR_B)
#define E_SLOW_GAIN_2(value) ((SHIFT((value),-8))&FOUR_B)
#define E_SLOW_GAIN_3(value) ((SHIFT((value),-12))&FOUR_B)

/*-----*/

#define E_MAX_PATH(value) (SHIFT((value),-4) * 0.032 )

/* laser off -> bit set -> (~(FALSE)) -> TRUE = ALARM */

#define E_LASER_STATUS(value) (~(_F_T((value),LASER_BIT)))

```

```

/* phi/f off -> bit clear -> TRUE = ALARM */

#define E_PHF_STATUS(value)  (__F_T((value),PHF_BIT))
#define E_PHI_STATUS(value)  (__F_T((value),PHI_BIT))

/*-----*/

/* following macros return ALARM (= TRUE) if bit is set
   but for CC e CCW which return ALARM when the bit is clear */

#define E_UNUSED_COMMAND(value) (~(__F_T((value),UN_ACCEPTED_BIT)))
#define E_CRASH(value)          (~(__F_T((value),CRASH_BIT)))
#define E_PENDING_COMMAND(value) (~(__F_T((value),PENDING_BIT)))
#define E_CCW_STOP(value)      (__F_T((value),NOT_ANTI_CLOCKWISE_BIT))
#define E_CW_STOP(value)       (__F_T((value),NOT_CLOCKWISE_BIT))
#define E_EM_STOP(value)       (~(__F_T((value),EMERGENCY_BREAK_BIT)))
#define E_IN_WIN(value)        (~(__F_T((value),IN_WINDOW_BIT)))
#define E_ZERO_SEARCH(value)   (~(__F_T((value),ZERO_SEARCH_BIT)))

/* following macro returns direction */

#define E_SHUTTLE_DIRECT(value) (((value)&DOWN_BIT) ? \
                                REVERSE : FORWARD)

/* following macros return TRUE when the bit is set */

#define E_IS_MOVING(value)      (~(__F_T((value),MOVING_BIT)))
#define E_SHUT_DOWN(value)     (~(__F_T((value),SHUT_DOWN_BIT)))
#define E_ACTIVE(value)        (~(__F_T((value),ACTIVE_BIT)))
#define E_START_UP(value)      (~(__F_T((value),START_UP_BIT)))
#define E_STAND_BY(value)      (~(__F_T((value),STAND_BY_BIT)))
#define E_BOOTSTRAP(value)     (~(__F_T((value),BOOTSTRAP_BIT)))

/*-----*/

/* CLOCK HAS NO DEFINE */

/*-----*/

/* linearly interpolate range 0 +20 A over bit 128 - 256 */

#define E_CEU_CURRENT(value)  LIN_INT(0.0,20.0,H_8_B,H_8_B, \
                                     ((value)&EIGHT_B))

/*-----*/

#define E_DIGITAL__1(value)  ((value)&EIGHT_B)

/*-----*/

/* the macro linearly interpolates for the temperature
range -51.34 , 86.23 assuming a 10 bits ADC with range 0 - 1023
(-10 +10 unipolar ADC)

range -50.00 , 50.00 assuming a 12 bits ADC with range 0 - 4095
(-10 +10 bipolar ADC) */

#ifndef REV_0
#define E_MASKX_TX(value)  LIN_INT(-51.34,86.23,F_10_B,0, \
                                   (SHIFT((value),-6)))
#else
#define E_MASKX_TX(value)  LIN_INT(-50.,50.,F_12_B,0, \

```

```

                                (SHIFT((value),-4))
#endif

#define E_MASK1_T1(value)    E_MASKX_TX(value)
#define E_MASK1_T2(value)    E_MASKX_TX(value)
#define E_MASK2_T1(value)    E_MASKX_TX(value)
#define E_MASK2_T2(value)    E_MASKX_TX(value)

/*-----*/

/*   the macro linearly interpolates for the temperature
range -24.20 , 75.70 assuming a 10 bits ADC with range 0 - 1023
      (-10 +10 unipolar ADC)

range -24.20 , 75.70 assuming a 12 bits ADC with range 2048 - 4095
      (-10 +10 bipolar ADC)          */

#define E_AD590_10_T(value)  LIN_INT(-24.2,75.7,F_10_B,0,\
      (SHIFT((value),-6)))

#define E_AD590_12_T(value)  LIN_INT(-24.2,75.7,H_12_B,H_12_B,\
      (SHIFT((value),-4)))

/*-----*/

#ifdef REV_0
#define E_M_MASK1_T(value)    E_AD590_10_T(value)
#define E_M_MASK2_T(value)    E_AD590_10_T(value)
#else
#define E_M_MASK1_T(value)    E_AD590_12_T(value)
#define E_M_MASK2_T(value)    E_AD590_12_T(value)
#endif
#endif

/*-----*/

#define E_SAP_DECK_T(value)    E_AD590_10_T(value)

/*-----*/

#define E_INCLIN_T(value)      E_AD590_10_T(value)

/*-----*/

#define E_SHA_ENC_T(value)     E_AD590_10_T((value))

/*-----*/

#define E_LASER_T(value)       E_AD590_10_T(value)

/*-----*/

#define E_COFFIN_T(value)      E_AD590_10_T(value)

/*-----*/

#define E_AIR_T(value)         E_AD590_10_T(value)

/*-----*/

#define E_BS_INSTR_T(value)    E_AD590_10_T(value)

/*-----*/

```

```

#define SWITCH_OFFSET      0x48e8

/* interpolate 0 - 360 degrees over bit range 0 - 65536 */

#define E_SHA_ENC(value)   (\
    (SWITCH_OFFSET-(unsigned)(value))\
    *360./65535.)

/*-----*/

/* interpolate 0 - 360 degrees over bit range 0 - 65536 */

#define E_ELEV_REGISTER(value) (\
    (SWITCH_OFFSET-(unsigned)(value))\
    *360./65535.)

/*-----*/

/* interpolate -60 - +60 degrees over bit range 0 - 4096 */

#define E_GYRO_COMP(value)  LIN_INT(-60.,60.,F_12_B,0, \
    (SHIFT((value),-4)))

/*-----*/

/* following macros return TRUE when bit is set */

#define E_SAP_FAST(value)   (~(__F_T((value),SAP_FAST_BIT)))
#define E_SAP_ZERO(value)  (~(__F_T((value),SAP_ZERO_BIT)))
#define E_SAP_MANUAL(value) (~(__F_T((value),SAP_MANUAL_BIT)))
#define E_SAP_NORMAL(value) (~(__F_T((value),SAP_NORMAL_BIT)))
#define E_SAP_SHTR(value)  (~(__F_T((value),SAP_SHUTTER_ON_BIT)))
#define E_SAP_HTR7(value)  (~(__F_T((value),SAP_HEATER7_ON_BIT)))
#define E_SAP_HTR6(value)  (~(__F_T((value),SAP_HEATER6_ON_BIT)))
#define E_SAP_HTR5(value)  (~(__F_T((value),SAP_HEATER5_ON_BIT)))
#define E_SAP_MOTOR(value) (~(__F_T((value),SAP__MOTOR_ON_BIT)))
#define E_SAP_GYROM(value) (~(__F_T((value),SAP_GYR_MOT_ON_BIT)))
#define E_SAP_INCL_HTR_S(value) (~(__F_T((value),SAP_INC_HTR_ON_BIT)))
#define E_SAP_INCL_HTR_E(value) (~(__F_T((value),SAP_INC_HTR_EN_BIT)))
#define E_SAP_ENC_HTR_S(value) (~(__F_T((value),SAP_ENC_HTR_ON_BIT)))
#define E_SAP_ENC_HTR_E(value) (~(__F_T((value),SAP_ENC_HTR_EN_BIT)))

/*-----*/

/* linearly interpolates in range -5 +5 V over 1024 bits */

#define E_GYRO_EXCIT(value)  LIN_INT(-5.0,5.0,F_10_B,0, \
    (SHIFT((value),-6)))

/*-----*/

/* linearly interpolates in range -125 +125 C over 1024 bits */

#define E_GYRO_T(value)     LIN_INT(-125.0,125.0,F_10_B,0, \
    (SHIFT((value),-6)))

/*-----*/

/* linearly interpolates in range -5 +5 V over 1024 bits */

#define E_M_GYRO_AC(value)  LIN_INT(-5.0,5.0,F_10_B,0, \
    (SHIFT((value),-6)))

```

```

/*-----*/
#define E_M_BRAKE_T(value)  E_AD590_12_T(value)
/*-----*/
#define E_BARATRON_T(value)  E_AD590_12_T(value)
/*-----*/
#define E_TELESCOPE_T(value)  E_AD590_12_T(value)
/*-----*/
#define E_SENSOTEC_T(value)  E_AD590_12_T(value)
/*-----*/
#define E_CEU_T(value)      E_AD590_12_T(value)
/*-----*/
/* linearly interpolates in range -10 +10 V over 4096 bits */
#define OP12(value)          LIN_INT(-10.0,10.0,F_12_B,0, \
                                   (SHIFT((value),-4)))
/*-----*/
/* zero is moved to offset 2.5 V ; if NORMAL_BIT is set
   datum is converted to 1 unit = 2.5 V , otherwise it
   is converted to 9.6 unit = 2.5 V */
#define E_GYRO_PICKOFF(value) ( E_SAP_NORMAL(value) ? \
                               (( OP12(value) - 2.5 ) / 2.5 ) : \
                               (( OP12(value) - 2.5 ) * (9.6/2.5) ) )
/*-----*/
/* zero is moved to offset 2.5 V ;
   datum is converted to 0.4 unit = 2.5 V */
#define E_GYRO_TORQUE(value) (( OP12(value) - 2.5 ) * (0.4 / 2.5))
/*-----*/
/* linearly interpolates in range -10 +10 V over 4096 bits */
#define E_INCLIN(value)      ( OP12(value) )
/*-----*/
/* zero is moved to offset 2.5 V ;
   datum is converted to 4 unit = 1.0 V */
#define E_FILT_INCLIN(value) (( OP12(value) - 2.5 ) * 4. )
/*-----*/
/* zero is moved to offset 2.5 V ;
   datum is converted to 1 unit = 1.0 V */

```

```

#define E_LVDT(value)      (( OP12(value) - 2.5 ))

/*-----*/

/* linearly interpolates in range -10 +10 V over 4096 bits */

#define E_VOLT_MON_5(value)  OP12(value)

/*-----*/

/* linearly interpolates in range -30 +30 V over 4096 bits */

#define E_VOLT_MON_P15(value) (OP12(value)*3)

/*-----*/

/* linearly interpolates in range -30 +30 V over 4096 bits */

#define E_VOLT_MON_M15(value) (OP12(value)*3)

/*-----*/

/* linearly interpolates in range -40 +40 V over 4096 bits */

#define E_VOLT_MON_P28(value) (OP12(value)*4)

/*-----*/

/* zero is moved to offset 2.5 V ;
   datum is converted to 1 unit = 1.0 V */

#define E_SAP_SERVO(value)  ( OP12(value) - 2.5 )

/*-----*/

/* useful conversion values */

/* revision of TOR_2_MB was 1014/760
   PSI_2_MB was 69.95
*/

#define TOR_2_MB      (1000./750.065)
#define PSI_2_MB      (1000./14.5038)
#define KPA_2_MB      (6.895*PSI_2_MB)

/*-----*/

/* revision to take into account half range work of ADC */

/* sensotec has max output at 0xbfe0 (shifted) corresponding
   to 5 V ; for overpressure we output out_of_range value ,
   otherwise we fit linearly between 0 and 1. psi over
   the output range and then convert to mbar */

#define MAX_SNS      (unsigned )0xbff0

#define E_SENSOTEC_P(value) (((value) > MAX_SNS) ? \
   OUT_OF_RANGE :\
   (LIN_INT(0.,1.,(0x0bff-0x0800),0x0800, \
   (SHIFT((value),-4))))*PSI_2_MB)

/*-----*/

```

```

/*  sensotec has max output at 0xffe0 (shifted) ; for over-
    pressure we output out_of_range value , otherwise we fit
    linearly between 0 and 10.000 torr over the output range
    and then convert to mbar */

#define MAX_BRT      (unsigned)0xffe0
#define E_BARATRON_P(value)  (((value) > MAX_BRT) ? \
    OUT_OF_RANGE :\
    (LIN_INT(0.,10.000,(0x0fff-0x0800),0x0800, \
    (SHIFT((value),-4))))*TOR_2_MB)

/*-----*/

/* linearly interpolates in range -10 0 Volts over 2048 bits
    conversion made assuming -10 V @ 0 mbar and
    0 V at 1034 mbar */

#define E_LASER_P(value)  ((2048.-((value)>>4))/2048*1034)

/*-----*/

/* linearly interpolates in range -20 +20 A over 4096 bits
    still requires conversion to pressure units */

#define E_M_SHUTTLE(value)  (2.*OP12(value))

/*-----*/

/* linearly interpolates in range -2 + 2 A over 4096 bits
    still requires conversion to pressure units */

#define E_M_LIMB_AMP(value)  (0.2*OP12(value))

/*-----*/

/*  fit -20 +20 degrees over range 0 - 4096 */
#define INCLIN_SCALE  10
#define NULL_OUTPUT  4.5
#define FULL_SCALE  (2*NULL_OUTPUT)
#define E_OUR_INCL_X(value)  ((-LIN_INT(-10.,10.,F_12_B,0,\
    (SHIFT((value),-4))) + NULL_OUTPUT) \
    * INCLIN_SCALE)
#define E_OUR_INCL_Y(value)  ((LIN_INT(-10.,10.,F_12_B,0,\
    (SHIFT((value),-4))) - NULL_OUTPUT) \
    * INCLIN_SCALE)

/*-----*/

/*  this is bourdon pressure
    range 0. 1000. millibar over half 12 bit range */

#define E_SPARE_AN_1(value)  LIN_INT(0.,1000.,H_12_B,H_12_B, \
    (SHIFT((value),-4)))

/*  line added 2 march 1993 */

#define E_SODEME(value)  E_SPARE_AN_1(value)

/*-----*/

/*  modified and added 2 lines 2 march 1993 */

```

```

#define E_SPARE_AN_2(value) (value)

#define E_OPTDSK_PR_A(value) (LIN_INT(0.,15.,H_12_B,H_12_B,\
    (SHIFT((value),-4))*PSI_2_MB)

#define E_OPTDSK_PR_D(value) (((value)>>2)&0x3f)

/*-----*/

/* modified and added 1 line 2 march 1993 */

#define E_SPARE_AN_3(value) (value)

#define E_MAGNETOMETER(value) ((1-SHIFT((value),-4)/4096.)*180)

/*-----*/

#define E_SPARE_AN_4(value) OUT_OF_RANGE

/*-----*/

/* preamp : T in K is volt * 100 */

/* nota

20 V -> 255 scalini => 1 scalino -> 20/255= .078 V

la calibrazione da .01 V per grado e cioe' 1 bit -> 8 gradi

*/

#define NIBBLE(value) (((float)(value)/EIGHT_B)*20.-10.)
#define E_PREAMP_T(value) (NIBBLE(value) * 100 - 273.4)

#define E_DET_VOLT(value) (((float)(value)/EIGHT_B)*40. - 20.)

#define E_PREAMP01_T(value) E_PREAMP_T(UPPER_BYTE(value))
#define E_PREAMP23_T(value) E_PREAMP_T(LOWER_BYTE(value))

#define E_PC_BIAS(value) (0.2*(LIN_INT(-10.,10.,EIGHT_B,0,\
    (LOWER_BYTE(value)))) -1.5)

#define E_DET0_VOLT1(value) E_DET_VOLT(UPPER_BYTE(value))
#define E_DET2_VOLT1(value) E_DET_VOLT(LOWER_BYTE(value))
#define E_DET0_VOLT2(value) E_DET_VOLT(UPPER_BYTE(value))
#define E_DET2_VOLT2(value) E_DET_VOLT(LOWER_BYTE(value))
#define E_DET1_VOLT1(value) E_DET_VOLT(UPPER_BYTE(value))
#define E_DET3_VOLT1(value) E_DET_VOLT(LOWER_BYTE(value))
#define E_DET1_VOLT2(value) E_DET_VOLT(UPPER_BYTE(value))
#define E_DET3_VOLT2(value) E_DET_VOLT(LOWER_BYTE(value))

#define E_DET0_BIAS(value) NIBBLE(UPPER_BYTE(value))
#define E_DET2_BIAS(value) NIBBLE(LOWER_BYTE(value))
#define E_DET1_BIAS(value) NIBBLE(UPPER_BYTE(value))
#define E_DET3_BIAS(value) NIBBLE(LOWER_BYTE(value))

/* offset value 16385 reevaluated from AEU tests made in
IROE, sec. 2.3.1 */

#define E_INTERF(value,aid) (((long)SHIFT((value),-1)-16385)<<\
    (3-(SHIFT((aid),-1)&TWO_B)))

```



```
#define E_OPTI_STATUS(value) ((value) & OPTI_WORD)
#define E_OPTI_IS_STDBY(value) ((value) & OPTI_STDBY)
#define E_OPTI_IS_RECOR(value) ((value) & OPTI_RECOR)
#define E_OPTI_IS_READ(value) ((value) & OPTI_READ)
#define E_OPTI_IS_DIAG(value) ((value) & OPTI_DIAG)

#endif
```

```
#ifndef __HSK_NOR_H
#define __HSK_NOR_H
static char *hsk_nor_Sccs_Id = "@(#) hsknor.h 1.1 3/16/93";
/*  this file normalizes offset in a frame making
    index as 'circular' buffer          */
/*  revised january 1993  */
#include "struct.h"
#define NORMALIZE(frame_shift,this_offset)  \
    (((frame_shift)+(this_offset))%FRA_IN_FOR)
#endif
```