valmet
A member of Metso Corporation

2000-12-13

OptiLoad-10 multinip kalander **BM 1 (HUSUMBM1)**

Handbok för
OptiLoad-10
multinip kalander **Tekniska data**

1. OPTILOADKALANDERNS DIMENSIONERINGSVÄRDEN

Maskintyp	OL 10-6700
Layouttyp	U
Papperssort	WFC
Papperets ytvikt	80 - 170 g/m²
Banbredd vid avrullningen	max. 6700 mm
Avrullningsdiameter	max. 3700 mm
- automatisk hopfogning	min. 2500 mm
Upprullningsdiameter	max. 3400 mm
Antalet valsar i valssystemet	10
Stativet dimensioneras för	10 valsar
Dimensioneringshastighet	1500 m/min
Körhastighet	1400 m/min
Spetsföringshastighet	max. 15 m/min
Hopfogningshastighet	max. 40 m/min
Linjetryck	max. 400 kN/m
- stativ	min. 80 kN/m
- beklädnad	max. 375 kN/m
Undre valsens yttemperatur	~ 140°C uppvärmning med olja
Drift	2. valsen
samt alla mellanvalsarna	(1500 m/min)

Tillval: avkylning av banan, placerad mellan stativets bakre fötter.

1.1. Valsarnas dimensioneringsvärden

- Polymervalsar DuraStone/DuraGloss, 4 st
  - diam. med beläggning 785 x 6940 mm
  - valsbeklädnad DuraStone 93ShD, DuraGloss 92ShD, s = 25 mm
  - Ra 0,4 µm vid leverans
  - valsstommens diam. 735 mm
  - valsens vikt 19000 kg
- Kokillvalsar Äquitherm PS/W, 3 st.
  - diam. 760 x 6840 mm
  - hårdhet 550 HV +20 HV
  - Ra värde 0.2 µm vid leverans
  - valsens vikt 19500 kg
- Dragvals Äquitherm PS/W, 1 st
  - diam. 760 x 6940 mm
  - hårdhet 550 HV + 20 HV
  - Ra-värde 0.2 µm vid leverans

- valsens vikt 19500 kg

Zonstyrda valsar

- övre vals SymCD/HP, 1 st
  - diam. med beklädnad 870 x 7410 mm
  - valsbeklädnad DuraGloss 92 ShD s = 25 mm
  - Ra-värde 0.4 µm vid leverans
  - valsens vikt inkl. lagerhus 34800 kg
- undre vals SymCD/HP, 1 st
  - diam. med beklädnad 870 x 7410 mm
  - valsbeklädnad DuraGloss 92 ShD s = 25 mm
  - Ra-värde 0.4 µm vid leverans
  - valsens vikt inkl. lagerhus 34800 kg

Banföring till avrullningssidan

- riktvals, 2 st
  - dia 650 x 7130 mm
- kompositspänningsmätningsvals, 1 st
  - diam. 415 x 7130 mm
- breddsträckvals med konstant bågighet, 1 st
  - diam. 255 x 7290 mm

Banföring i valssystemet

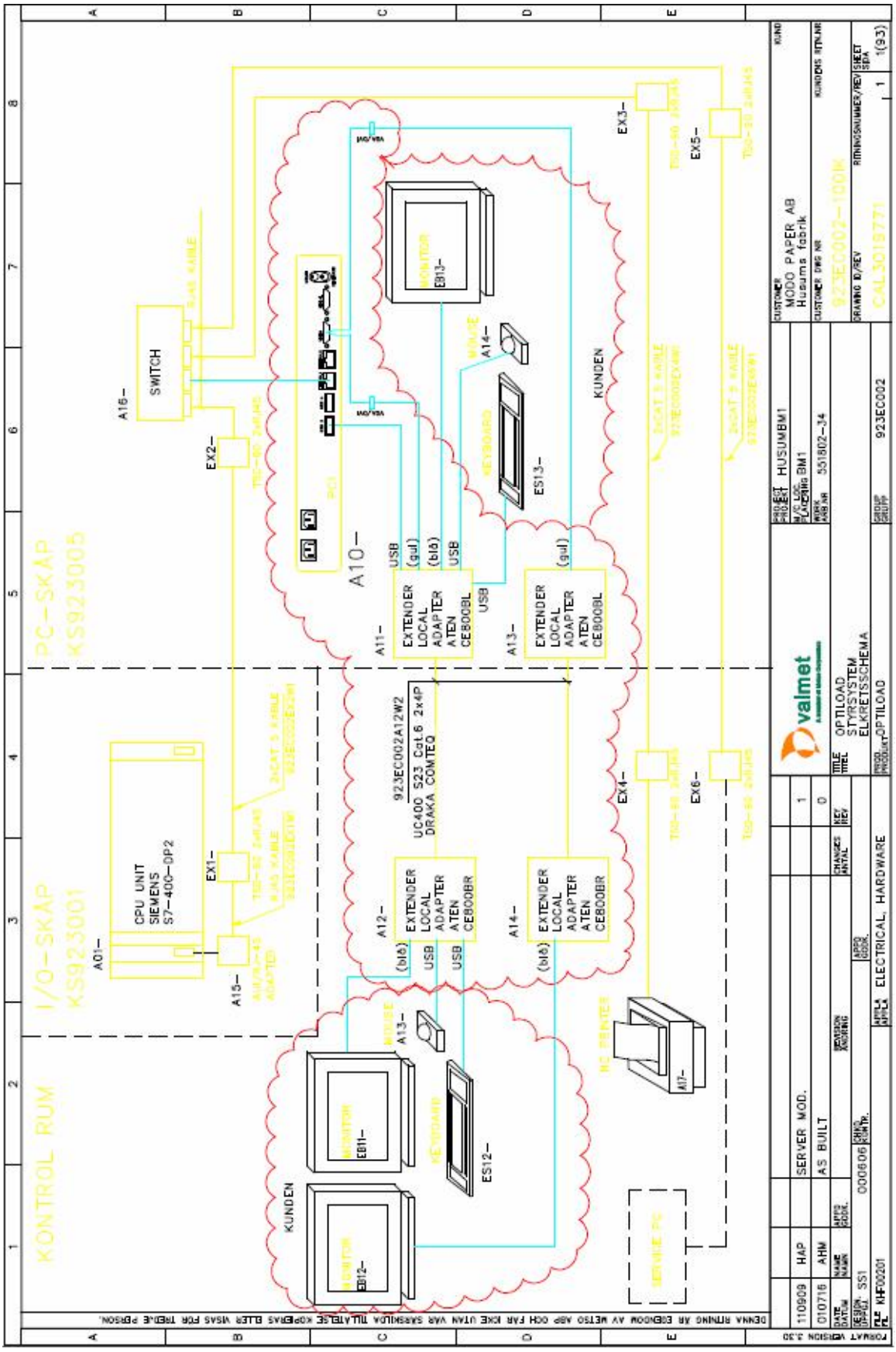
- Controspreadvals före varje nyp, 8 st
  - diam. 360 x 6870 mm
- breddsträckvals före andra nypet, 1 st
  - diam. 255 x 7290 mm

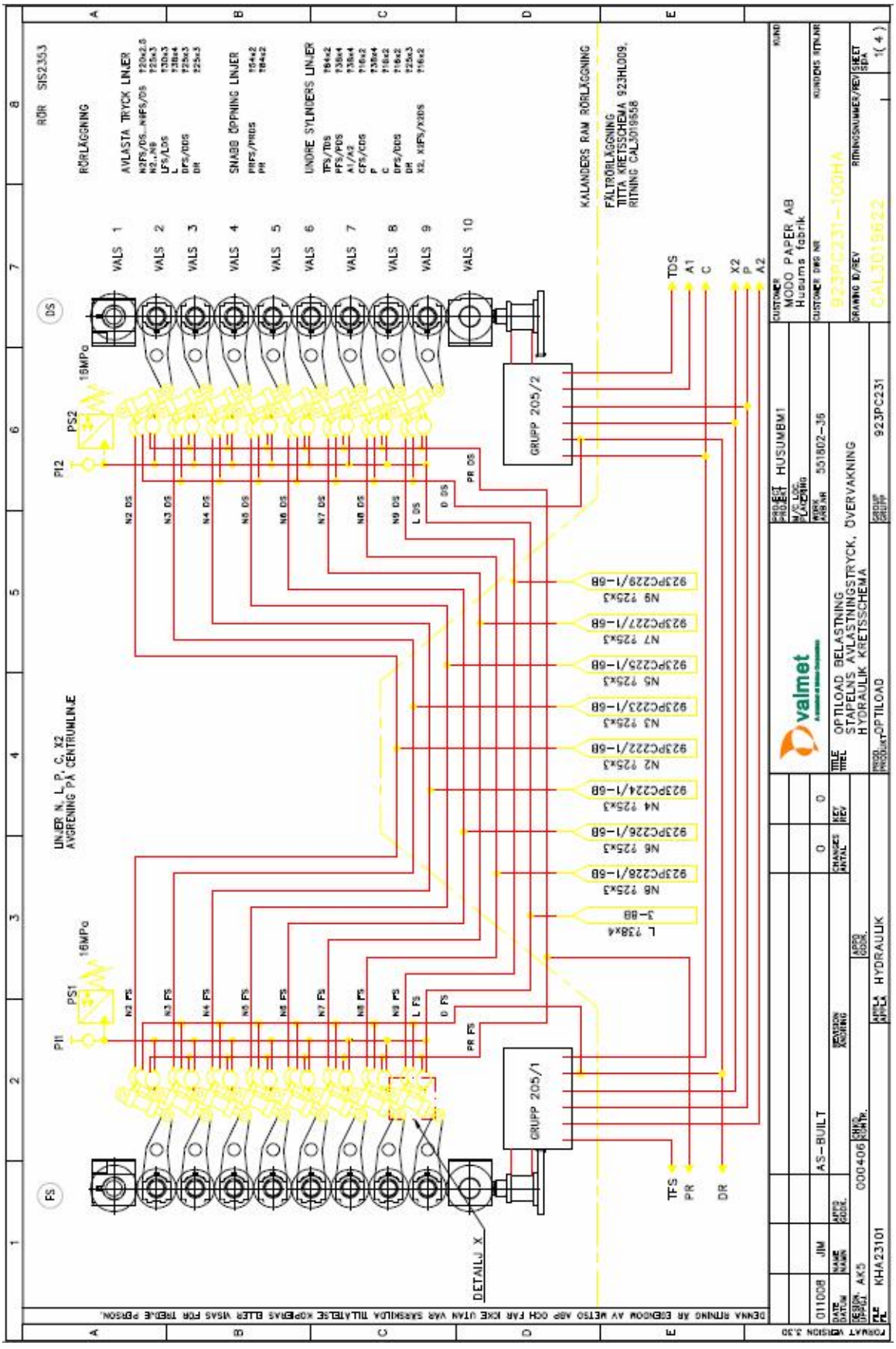
Banföring efter valssystemet

- riktvals, 4 st
  - diam. 650 x 7130 mm
- kompositspänningsmätningsvals, 1 st
  - diam. 415 x 7130 mm

Upprullningsmaskinens anpressvalsanordning

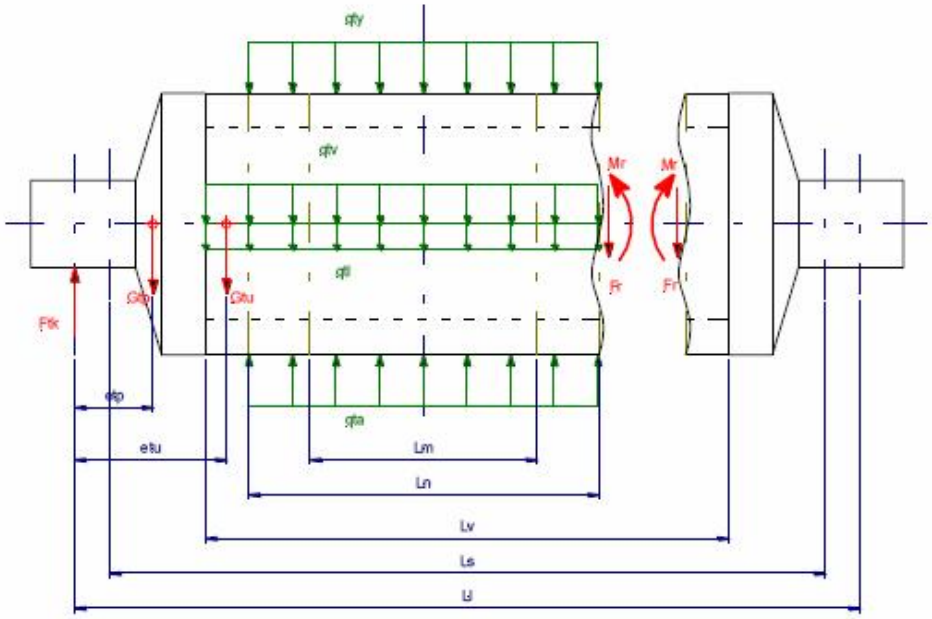
- sektionssträckvals, 1 st
  - diam. 198 x 6870 mm
- anpressvals, 1 st
  - diam. 800 x 7060 mm


KONTROL RUM
I/O-SKAP
KS923001
PC-SKAP
KS923005
CPU UNIT
SIEMENS
S7-400-DP2
A01-
A15-
EX1-
EX2-
SWITCH
A16-
A10-
A11-
A12-
A13-
A14-
EXTENDER
LOCAL
ADAPTER
ATEN
CE800BL
USB
(gul)
(blå)
923EC002A12W2
UC400 S23 Cat.6 2x4P
DRAKA COMTEQ
KUNDEN
MONITOR
EB11-
EB12-
EB13-
KEYBOARD
ES12-
ES13-
MOUSE
EX3-
EX4-
EX5-
EX6-
A17-
NC PRINTER
SERVER PC
valmet
OPTILOAD
STYRSYSTEM
ELKRETSSCHEMA
MODO PAPER AB
Husums fabrik
HUSUMBM1
BM1
551802-34
923EC002
923EC002
OPTILOAD
ELECTRICAL, HARDWARE
SERVER MOD.
AS BUILT
110909
010716
HAP
AHW
SS1
000606
KHF002201

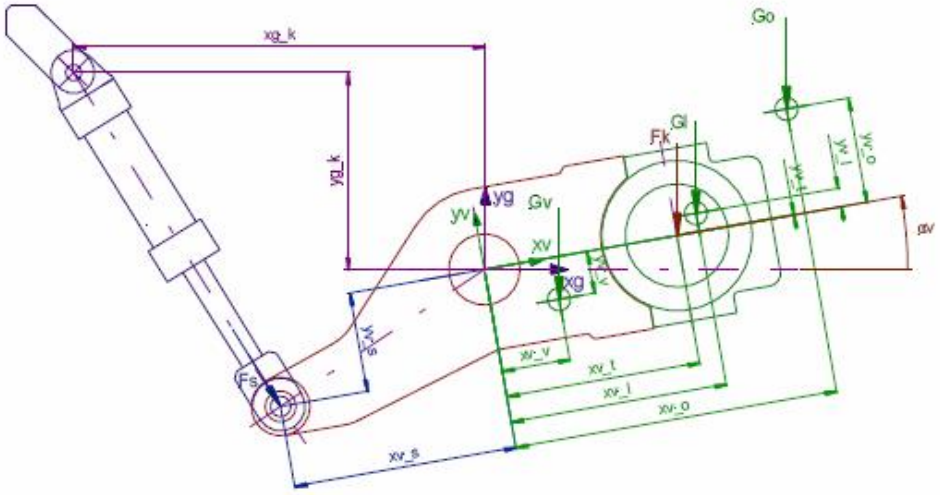

RÖR SIS2353
RÖRLÄGGNING
AVLASTA TRYCK LINJER
SNABB ÖPPNING LINJER
LINDRE SYLINDERS LINJER
VALS 1
VALS 2
VALS 3
VALS 4
VALS 5
VALS 6
VALS 7
VALS 8
VALS 9
VALS 10
KALANDERS RAM RÖRLÄGGNING
FÄLTRÖRLÄGGNING
TITTA KRETSSCHEMA 923HL009.
RITNING CAL3019558
LINJER N, L, P, C, X2
AVGRENING PÅ CENTRUMLINJE
GRUPP 205/2
GRUPP 205/1
DETALJ X
TDS
A1
C
X2
P
A2
TFS
PR
DR
PS1
PS2
P11
P12
16MPa
valmet
MODO PAPER AB
Husums fabrik
HUSUMBM1
OPTILOAD BELASTNING
STAPELNS AVLASTNINGSTRYCK, ÖVERVAKNING
HYDRAULIK KRETSSCHEMA
923PC231
923PC231-100HA
CAL3019622
OPTILOAD
HYDRAULIK
AS-BUILT
011008
AK5
KHA23101
000406
DENNA RITNING ÄR EGENDOM AV METSO ABP OCH FÅR EJ UTAN VÅR SKRIFTLIGA TILLÅTELSE KOPIERAS ELLER VISAS FÖR TREDJE PERSON.

3 KEVENNYSSUUREIDEN LASKENTA

3.1 TELA



Kuva 4 **Tela**



Kuva 5 **Telavarsi ja sylinteri**

4.1.4 Telakohtaiset suureet

Suureet **telalle k** :

$(E^*I)_{t,k}$ telan kokonaisjäykkyys pinnoitteen

$$(E \cdot I)_{t,k} = \left[\begin{array}{l} \left\{ \frac{\pi}{64} \cdot \left(D_{uv,k}^{\ 4} - D \right. \right. \\ \left. \left. \frac{\pi}{64} \cdot \left(D_{up,k}^{\ 4} - D \right. \right. \end{array} \right. \left. \begin{array}{l} \\ \left. \frac{\pi}{4} \cdot d_{r,k}^{\ 2} \cdot \frac{1}{8} \cdot D_{r,k}^{\ 2} \right) \right\} \cdot E_{tv,k} + \end{array} \right]$$

$I_{t,k}$ telapoikkipinnan jäykkyys telavaipan ja pinnoitteen

$$I_{t,k} = \left\{ \frac{\pi}{64} \cdot \left(D_{uv,k}^{\ 4} - D_{sv} \right. \right. \left. \left. \cdot D_{r,k}^{\ 2} \right) \right\} + \frac{\pi}{64} \cdot \left(D_{up,k}^{\ 4} - D_{uv,k}^{\ 4} \right)$$

$G_{tu,k,s}$ telavaipan nippipituuden ulkopuolisen osan paino yhdellä puolella

$$G_{tu,k,s} = \cdot \frac{1}{2} \left\{ \begin{array}{l} \left[\frac{\pi}{4} \cdot \left(D_{uv,k}^{\ 2} - D_{sv,k}^{\ 2} \right) - \left(N_{r,k} \cdot \frac{\pi}{4} \cdot d_{r,k}^{\ 2} \right) \right] \cdot \left(L_{v,k} - L_n \right) \cdot \rho_{v,k} + \\ \frac{\pi}{4} \cdot \left(D_{up,k}^{\ 2} - D_{uv,k}^{\ 2} \right) \cdot \left(L_{p,k} - L_n \right) \cdot \rho_{p,k} \end{array} \right\} \cdot g$$

Alaindeksit (toinen tai jäljempi):

k	tela k
s	kalanterin puoli HP / KP
$D_{uv,k}$	telavaipan ulkohalkaisija
$D_{sv,k}$	telavaipan sisähalkaisija
$L_{v,k}$	telavaipan pituus
$D_{r,k}$	periferiaporausten kehän halkaisija
$d_{r,k}$	periferiaporausten reiän halkaisija
$N_{r,k}$	periferiaporausten reikien lukumäärä
$E_{tv,k}$	telavaipan kimmokerroin
$\rho_{v,k}$	telavaipan tiheys
$D_{up,k}$	pinnoitteen ulkohalkaisija
$L_{p,k}$	pinnoitteen pituus
	Yleensä voidaan käyttää $L_{p,k} = L_{v,k}$
$\rho_{p,k}$	pinnoitteen tiheys
$E_{p,k}$	pinnoitteen kimmokerroin
g	maan vetovoiman kiihtyvyys 9.81 m/s^2

3.2 MERKINNÄT

3.2.1 Yleistiedot

nt telojen lukumäärä telastossa

Telanumerointi $i = 1 .. nt$

n_n telaston nippien lukumäärä

Nippien numerointi $1 .. n_n$

3.2.2 Mitat

L_l laakeriväli

L_n nippipituus

L_v telavaipan pituus

e_{tp} telapäädyn paino

e_{tu} telavaipan nippipituu isteen etäisyys laakerilinjasta

L_s telan tuentaväli taipuman

L_m taipuman mittauspituus taipu

3.2.3 Poikkipintasuureet

I_t telapoikkipinnan jäyhyysmomentti sisä tteen

3.2.4 Materiaali

E_t telavaipan ja pinnoitteen yhdistetty redusoitu ki smomentille I_t

3.2.5 Painot

G_t telan kokonaispaino

G_{tv} telan vaipan ja pinnoitteen paino

G_{tp} telapäädyn paino

G_{tu} telavaipan nippipituuden ulkopuolisen osan paino ***per puoli***

3.2.6 Kuormitukset

q_{yy} telaston yläpin viival

q_{aa} telaston al

q_{tt} telan vaipar

3.2.7 Voimasuureet

$F_{t,i}$ telan optimoit

$F_{ku,i}$ ulkopuolisten

$F_{tk,i}$ telan kokon

$F_{s,i}$ telan koko

```
// ---------------------------------------------------------------
// 1.5:     Gtu_ts = 0.5*( [(PI/4)*F1-(Nr*F2)]*(Lv-Ln)*Psv+(PI/4)*F3*(Lp-Ln)*Pp )*g
//          Gtu_ds = 0.5*( [(PI/4)*F1-(Nr*F2)]*(Lv-Ln)*Psv+(PI/4)*F3*(Lp-Ln)*Pp )*g
//          F1      = pow(Duv,2)-pow(Dsv,2)
//          F2      = (PI/4)*pow(Dr,2)
//          F3      = pow(Dup,2)-pow(Duv,2)
// ---------------------------------------
if (ol_prints > 0) printf("\n\nGtu:\n");
for(i=0; i< ol_mrcount; i++)
{
        // F1:
        k=Ftype("(p)-(p)");
        px1=2; px2=2;
        F1 = Formula(Duv[i],px1,Dsv[i],px2,dummy1,dummy1,k,printex);

        // F2:
        k=Ftype("/*(p)");
        tmptab1[0]=PI; tmptab1[1]=4.0;
        px1=2;
        F2 = Formula(tmptab1[0],tmptab1[1],Dr[i],px1,dummy1,dummy1,k,printex);

        // F3:
        k=Ftype("(p)-(p)");
        px1=2; px2=2;
        tmptab1[0]=Dup[i];          // test Dup, because in termoroll it is 0.0
        if (tmptab1[0]==0.0) tmptab1[0]=Duv[i];         // then use Duv <> 0.0
        F3 = Formula(tmptab1[0],px1,Duv[i],px2,dummy1,dummy1,k,printex);

        Gtu_ts[i] = 0.5*( (((PI/4.0)*F1-(Nr[i]*F2))*(Lv[i]-Ln))*Psv[i]+((PI/4.0)*F3*(Lp[i]-Ln)*Pp[i]) )*gee;
        Gtu_ts[i] = Gtu_ts[i]/1000000000.0; // Psv[i] and Pp[i] should be *1.0e-9
        if (ol_prints > 0) printf(" Gtu_ts r:%d=%6.3f\n", i, Gtu_ts[i]);
        Gtu_ds[i] = Gtu_ts[i];
        if (ol_prints > 0) printf(" Gtu_ds r:%d=%6.3f\n", i, Gtu_ds[i]);
}
```

```
/*********************************************************************/
/*                                                                   */
/*  Function for sigma calculation                                   */
/*                                                                   */
/********************************************************************* **/

float Sigma(                                /* Out: result  */
        float   *partab1,                   /* Inp: pointer to par1_table[0] */
        float   *partab2,                   /* Inp: pointer to par2_table[0] */
        float   *partab3,                   /* Inp: pointer to par3_table[0] */
        float   *partab4,                   /* Inp: pointer to par4_table[0] */
        float   *partab5,                   /* Inp: pointer to par5_table[0] */
        float   *partab6,                   /* Inp: pointer to par6_table[0] */
        char    *sform,                     /* Inp: calculation formula */
        int     *nbr,                       /* Inp: pointer to roll count */
        int     prnt)

{
float   s_result;
int i, s_idx, s_typ;

        s_result = 0;
        // roll count index
        s_idx = *nbr;
        if (s_idx > MAX_MRNBR_LEN) s_idx = MAX_MRNBR_LEN;

        // sigma formula index
        s_typ=Ftype(sform);

        if (prnt==1) printf("\n Sigma:\n");
        if (prnt==1) printf("  s_typ=%d\n",s_typ);
        if (prnt==1) printf("  rolls = 2 - %d\n",s_idx + 1);

        // calulate indexed sigma
        for (i=1; i <= s_idx; i++)
        {
                s_result += Formula(*partab1,*partab2,*partab3,*partab4,*partab5,*partab6,s_typ,prnt);
                partab1++;
                partab2++;
                partab3++;
                partab4++;
                partab5++;
                partab6++;
        }
        if (prnt==1) printf("  s_result=%6.3f\n",s_result);
        return s_result;

} // Sigma
```

```
/*************************************************************************/
/*                                                                       */
/*  Function for formula calculation                                     */
/*                                                                       */
/*************************************************************************/

float Formula(                          /* Out: result  */
        float   par1,                   /* Inp: pointer to par1_table[0] */
        float   par2,                   /* Inp: pointer to par2_table[0] */
        float   par3,                   /* Inp: pointer to par3_table[0] */
        float   par4,                   /* Inp: pointer to par4_table[0] */
        float   par5,                   /* Inp: pointer to par5_table[0] */
        float   par6,                   /* Inp: pointer to par6_table[0] */
        int     ftyp,                   /* Inp: calculation formula */
        int     prnt)                   /* Inp: print out */
{
float   f_result;

        f_result = 0;
        if (prnt==1) printf("\n Formula: type=%d\n", ftyp);

        // calulate indexed sigma
        switch (ftyp) {
        case 1:
                {       // multiply: par1*par2
                        f_result = par1*par2;
                        if (prnt==1)
                                printf("  ftyp=1: par1=%6.3f, par2=%6.3f\n",par1, par2);
                        if (prnt==1)
                                printf("  f_result=%6.3f\n",f_result);
                break;
                }
        case 2:
                {       // devide: par1/par2
                        f_result = par1/par2;
                        if (prnt==1)
                                printf("  ftyp=2: par1=%6.3f, par2=%6.3f\n",par1, par2);
                        if (prnt==1)
                                printf("  f_result=%6.3f\n",f_result);
                break;
                }
        case 3:
                {       // multiply and devide: (par1*par2)/par3
                        f_result = (par1*par2)/(par3);
                        if (prnt==1)
                                printf("  ftyp=3: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                        if (prnt==1)
                                printf(" f_result=%6.3f\n",f_result);
                break;
                }
        case 4:
                {       // devide and multiply : (par1/par2)*par3
                        f_result = ((par1/par2)*par3);
                        if (prnt==1)
                                printf("  ftyp=4: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                        if (prnt==1)
                                printf(" f_result=%6.3f\n",f_result);
                break;
                }
        case 5:
                {       // (devide and multiply) and plus: ((par1/par2)*par3)+par4

                        f_result +=((par1/par2)*par3) + par4;
                        if (prnt==1)
                                printf("  ftyp=5: par1=%6.3f, par2=%6.3f, par3=%6.3f, par3=%6.3f\n",par1,
                                par2, par3, par4);
                        if (prnt==1)
                                printf(" f_result=%6.3f\n",f_result);
                break;
                }
```

```
case 6:
        {       // (multiply and devide) + (multiply and devide): ((par1*par2)/par3)+(par4*par5)/par6)
                f_result = ((par1*par2)/par3) + ((par4*par5)/par6);
                if (prnt==1)
                        printf("  ftyp=6: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                if (prnt==1)
                        printf("          par4=%6.3f, par5=%6.3f, par6=%6.3f\n",par4, par5, par6);
                if (prnt==1)
                        printf("  f_result=%6.3f\n",f_result);
        break;
        }
case 7:
        {       // (multiply and devide) - (multiply and devide): ((par1*par2)/par3)-(par4*par5)/par6)
                f_result = ((par1*par2)/par3) - ((par4*par5)/par6);
                if (prnt==1)
                        printf("  ftyp=7: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                if (prnt==1)
                        printf("          par4=%6.3f, par5=%6.3f, par6=%6.3f\n",par4, par5, par6);
                if (prnt==1)
                        printf(" f_result=%6.3f\n",f_result);
        break;
        }
case 8:
        {       // (multiply and devide and multiply) : (((par1*par2)/par3)*par4)
                f_result = ((par1*par2)/par3)*par4;
                if (prnt==1)
                        printf("  ftyp=8: par1=%6.3f, par2=%6.3f, par3=%6.3f, par4=%6.3f\n",par1,
                        par2, par3, par4);
                if (prnt==1)
                        printf(" f_result=%6.3f\n",f_result);
        break;
        }
case 9:
        {       // (devide and multiply) + (devide and multiply): ((par1/par2)*par3)+(par4/par5)*par6)
                f_result = ((par1/par2)*par3) + ((par4/par5)*par6);
                if (prnt==1)
                        printf("  ftyp=9: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                if (prnt==1)
                        printf("          par4=%6.3f, par5=%6.3f, par6=%6.3f\n",par4, par5, par6);
                if (prnt==1)
                        printf(" f_result=%6.3f\n",f_result);
        break;
        }
case 10:
        {       // (devide and multiply) - (devide and multiply): ((par1/par2)*par3)-(par4/par5)*par6)
                f_result = ((par1/par2)*par3) - ((par4/par5)*par6);
                if (prnt==1)
                        printf("  ftyp=10: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                if (prnt==1)
                        printf("          par4=%6.3f, par5=%6.3f, par6=%6.3f\n",par4, par5, par6);
                if (prnt==1)
                        printf("  f_result=%6.3f\n",f_result);
        break;
        }
case 11:
        {       // (multiply and multiply and multiply): ((par1*par2)*par3)
                f_result = (par1*par2)*par3;
                if (prnt==1)
                        printf("  ftyp=11: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                if (prnt==1)
                        printf("  f_result=%6.3f\n",f_result);
        break;
        }
case 12:
        {       // (devide and multiply(power)): (par1/par2)*(pow(par3,par4))
                f_result = (par1/par2)*(pow(par3,par4));
                if (prnt==1)
                        printf("  ftyp=12: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                if (prnt==1)
                        printf("          par4=%6.3f\n", par4);
```

```
                        if (prnt==1)
                                printf("  f_result=%6.3f\n",f_result);
                break;
                }
        case 13:
                {       // (power-power): (pow(par1,par2)-pow(par3,par4))
                        f_result = pow(par1,par2)-pow(par3,par4);
                        if (prnt==1)
                                printf("  ftyp=13: par1=%6.3f, par2=%6.3f, par3=%6.3f\n",par1, par2, par3);
                        if (prnt==1)
                                printf("           par4=%6.3f\n",par4);
                        if (prnt==1)
                                printf("  f_result=%6.3f\n",f_result);
                break;
                }
        default:
                        f_result = 0;
                break;
        }

        return (f_result);

} // Formula
```