

PTP-PROTOCOL VOOR SERIËLE DATACOMMUNICATIE

Inleiding

Dit document beschrijft het PTP-protocol dat gebruikt kan worden voor seriële datacommunicatie tussen twee computersystemen. Het protocol is gebaseerd op gebruik van vaste communicatielijnen. De aangesloten computersystemen kunnen in principe gelijktijdig databerichten verzenden (master/master verbinding). Het protocol is bijzonder geschikt voor datacommunicatie tussen verkeersregelinstallaties. Het gebruik van het protocol voor het uitwisselen van zogenaamde koppelsignalen tussen verkeersregelinstallaties wordt dan ook nader beschreven.

Bericht layout (opbouw)

Er zijn 3 soorten berichten:

<SYN><SOH><dest><length><crc1><data><crc2> of <SYN><SOH><dest><0><crc1>
 <SYN><ACK><src><0><crc1>
 <SYN><NAK><src><0><crc1>
 (<SYN><ACK><src><length><crc1><data><crc2> - bericht bij master/slave toepassing)

De betekenis van de bovenstaande karakters is als volgt:

<SYN>	byte (0x16)	Elk bericht wordt voorafgegaan door een uniek start karakter. Tezamen met het volgende karakter (<SOH>, <ACK>, of <NAK>) levert dit een eenduidige 'start reeks'. Zie ook opmerking over het <SYN>-karakter.
<SOH>	byte (0x01)	Identificatie van een databericht.
<ACK>	byte (0x06)	Identificatie om aan te geven dat het databericht goed is ontvangen (bij een master/slave toepassing kan dit bericht ook data bevatten).
<NAK>	byte (0x15)	Identificatie om aan te geven dat een fout is gedetecteerd in de ontvangen data.
<dest>	byte	<SID> voor wie het databericht bestemd is.
<src>	byte	<SID> dat het databericht heeft verstuurd.
<length>	byte	Het aantal bytes in het veld <data>. De totale berichtlengte is dus <length> + 8 bytes. Maximaal kunnen per bericht 255 databytes worden verzonden.
<0>	byte	De waarde 0. Geeft aan dat in het bericht geen data aanwezig is. Ook t.b.v. vaste lengte berichtheader.
<crc1>	word	Dit is een 'cyclic redundancy check' (CRC) over de berichtheader, exclusief het <SYN> karakter. Dat is <TYPE>, <SID> en <length> (dus over 3 bytes).
<data>	bytereeks	De te versturen data. Het aantal bytes is vastgelegd in het veld <length>.
<crc2>	word	De CRC over alle bytes in het veld <data>.

<SYN> karakter

Hierboven is beschreven dat het <SYN> karakter uniek moet zijn. Het is echter mogelijk dat dit karakter ook voorkomt op een andere plaats in het bericht.

Om dit te ondervangen geldt de volgende procedure.

Als het <SYN> karakter in de normale datastroom voorkomt, dan wordt dit karakter vervangen door twee <SYN> karakters.

Noot: De extra ingevoegde <SYN> karakters tellen niet mee voor de berichtlengte <length> of de CRC-berekening <crc1> en <crc2>.

Berichtsoort

In de eerste twee karakters van het bericht ('startreeks') wordt de berichtsoort vastgelegd, namelijk <SYN><SOH>, <SYN><ACK> en <SYN><NAK>.

Indien tijdens het inlezen van de bytereeks een 'startreeks' (<SYN><SOH>, <SYN><ACK> of <SYN><NAK>) wordt gedetecteerd, dan wordt het inlezen geherstart (start van een nieuw bericht).

Side identifieer (SID: <dst> en <src>)

Aan een systeem/poort kan een <SID> worden toegekend. Dit <SID> wordt gebruikt om te controleren of het juiste systeem is aangesloten. Bij het versturen van een bericht wordt namelijk de <SID> van het ontvangende systeem meegezonden. Het ontvangende systeem geeft alleen een antwoord op een bericht, indien de <SID> juist is. Hierdoor kan het verkeerd aansluiten van de seriële poorten worden gecontroleerd. Indien aan verschillende systemen/poorten hetzelfde <SID> wordt toegekend, kan deze controle op aansluitfouten niet plaatsvinden.

Berichtlengte

In het veld <length> wordt het aantal bytes in het veld <data> vastgelegd.

Het veld <data> kan maximaal 255 bytes bevatten.

De totale berichtlengte is dus <length> + 8 bytes.

De extra ingevoegde <SYN> karakters tellen niet mee voor de berichtlengte.

Data

Het veld <data> bevat de te versturen data. Het aantal bytes in dit veld is vastgelegd in het veld <length>.

CRC berekening

Het PTP-protocol onderscheidt twee checksums, namelijk:

- een checksum <crc1> over de berichtheader (exclusief het SYN karakter), en
- een checksum <crc2> over alle bytes in het veld <data>.

De acties op het constateren van checksumfouten zijn beschreven onder procedures.

Noot: De extra ingevoegde <SYN> karakters tellen niet mee voor de CRC-berekening <crc1> en <crc2>.

Een CRC wordt volgens CCITT berekend met behulp van de onderstaande

C programmatuur.

```
/* CRC_UPDATE() */
/* ----- */
/* crc_update() berekent de CRC over een byte.
 * crc_update() wordt aangeroepen door crc_block().
 * CRC berekening volgens CCITT. Algoritme betrokken van ir. J. van Dillen.
 * crc_update() geeft als return-waarde de berekende CRC.
 */
```

```
static word crc_update (word crc, byte b)
{
    word i, x;

    for (i= 0; i < 8; i++) {
        x = b ^ crc;
        crc >>= 1;
        b >>= 1;
        if (x & 1) crc ^= 0x8408;
    }
    return crc;
}
```

```
/* CRC_BLOCK() */
/* ----- */
/* crc_block() berekent de CRC over een datablok.
 * crc_block() maakt gebruik van de functie crc_update().
 * crc_block() geeft als return-waarde de berekende CRC van het datablok.
 */
```

```
word crc_block(byte *data, word len)
{
    word crc= 0;

    while (len--> 0) crc= crc_update(crc, *data++);

    return crc;
}
```

Systeemtijden

Het PTP-protocol onderscheidt 3 systeemtijden, namelijk:

- een wachttijd
- een zendtijd
- een live-tijd

De wachttijd start op het moment dat de seriële verbinding niet meer aanwezig is (de verbinding is verbroken). De wachttijd is default ingesteld op 200 tienden seconde.

De zendtijd en live-tijd starten direct na het verzenden van een databericht (SOH-bericht). De zendtijd en live-tijd zijn default ingesteld op 40 tienden seconde. Het gebruik van deze systeemtijden wordt nader beschreven onder procedures.

Noot: De genoemde ingestelde waarden zijn defaultwaarden. De juiste waarden dienen op het gebruik van het protocol te worden afgestemd. De grootte van de in te stellen waarden is o.a. afhankelijk van de gekozen baudrate en de maximaal gebruikte berichtgrootte. Ook de wijze van het verzenden van de karakters (poll of interrupt basis) speelt ook nog een rol.

Status van de seriële verbinding

De status van de seriële verbinding wordt door een aparte variabele (type word) aangegeven. Indien de verbinding aanwezig is heeft deze variabele de status waar (ongelijk aan 0), anders niet waar (gelijk aan 0). De status van de verbinding wordt waar (bit 0 wordt 1) indien op een uitgaand databericht (SOH-bericht) tijdig een correct ACK-bericht wordt terugontvangen. De status van de verbinding wordt niet waar, indien op een uitgaand databericht (<SOH>-bericht) na het versturen van de herhalingsberichten geen ACK-bericht wordt terugontvangen.

Procedures

wachtprocedure

Op het moment dat de seriële verbinding niet meer aanwezig is (de verbinding is verbroken) start een 'wachttijd'. Gedurende het lopen van deze 'wachttijd' mag het systeem geen uitgaande berichten versturen. Na het verstrijken van de ingestelde waarde van de 'wachttijd' wordt getracht de verbinding op te bouwen, inkomende databerichten worden (weer) beantwoord. De seriële verbinding is aanwezig op het moment dat er op een verzonden databericht een correct ACK-bericht wordt terugontvangen. De status van de verbinding is dan waar (ongelijk aan 0).

zendprocedure databerichten

Een systeem behoeft in principe alleen een nieuw databericht (SOH-bericht) te verzenden, indien een wijziging in de informatie (data) is opgetreden. Direct na het verzenden van een databericht wordt een 'zend-datavlag' (type word) opgezet en een 'zendtijd' en een 'live-tijd' ge(her) start.

Het zendende systeem wacht gedurende de 'zendtijd' op een antwoord van het ontvangende systeem. Zolang er nog geen ACK- of NAK-bericht is terugontvangen,

mogen er gedurende het lopen van de 'zendtijd' geen nieuwe databerichten worden verzonden. Gedurende deze periode staat de 'zend-datavlag' op.

Bij het ontvangst van een ACK-bericht heeft het ontvangende systeem het data-bericht correct ontvangen, de 'zend-datavlag' wordt afgezet en kunnen er weer nieuwe databerichten worden verzonden.

Bij de ontvangst van een NAK-bericht of het bereiken van de instelde waarde van de 'zendtijd' wordt opnieuw hetzelfde databericht verzonden (maximaal drie maal hetzelfde bericht). Indien na de derde maal verzenden van hetzelfde databericht er binnen de ingestelde 'zendtijd' geen correct ACK-bericht is ontvangen, dan is de seriële verbinding verbroken (niet meer aanwezig). De verzendbuffer en 'zend-datavlag' worden gereset.

Let op! Gedurende het lopen van de 'wachtijd' worden er geen berichten verzonden.

ontvangstprocedure databerichten

Een systeem geeft uitsluitend een antwoord op een ontvangen databericht als de header (<SYN> t/m <crc1>) correct is ontvangen en de <src> code juist is. Indien ook <crc2> correct is of de waarde in <length> gelijk is aan 0, zal na ontvangst van het laatste byte een ACK-bericht worden teruggestuurd. Mocht <crc2> niet kloppen, dan stuurt het ontvangende systeem een NAK-bericht terug naar het zendende systeem. Indien een correct databericht is ontvangen dan wordt een 'ontvangst-datavlag' (type word) opgezet. Deze 'ontvangst-datavlag' blijft één systeemronde waar.

Let op! Gedurende het lopen van de 'wachtijd' worden er geen berichten verzonden.

lege databerichten/'live'-berichten

Indien gedurende het lopen van de 'live-tijd' door een systeem geen nieuwe databerichten zijn verstuurd, wordt bij het bereiken van de ingestelde waarde een leeg databericht verzonden ('live'-bericht). Vorm: <SYN><SOH><dest><0><crc1>. Dit 'live'-bericht wordt gebruikt om de status van de seriële verbinding te kunnen controleren. Indien de ingestelde waarde van de 'live-tijd' gelijk is aan de waarde 0, dan worden er geen live-berichten verzonden.

Let op ! Gedurende het lopen van de 'wachtijd' worden er geen berichten verzonden.

Berichtfouten

De fouten in de verzonden seriële berichten worden bitgewijs aangegeven in een aparte variabele (type word).

De volgende berichtfouten worden aangegeven:

- NAK-bericht ontvangen - BIT0
- zendtimeout - BIT1
- overschrijding maximum aantal herhalingsberichten - BIT2

De berichtfouten worden één systeemronde onthouden.

Kwaliteit van de seriële verbinding

De kwaliteit van de seriële verbinding kan in de regelapplicatie worden bijgehouden door gedurende het aanwezig zijn van de verbinding, de berichtfouten te tellen:

- aantal ontvangen NAK berichten
- aantal opgetreden zendtimeouts

PTP-PROTOCOL - APPLICATIE KOPPELSIGNALEN

Inleiding

Het PTP-protocol is met name geschikt voor het verzenden van zogenaamde koppelsignalen tussen verkeersregelinstanties. De procedures, die worden gehanteerd, zijn hieronder beschreven.

Koppelsignalen kunnen worden onderverdeeld in:

- inkomende koppelsignalen
- uitgaande koppelsignalen

Koppelsignalen zijn booleaanse variabelen die de waarde waar (TRUE) of niet waar (FALSE) kunnen aannemen.

Het maximaal aantal gebruikte koppelsignalen wordt vastgelegd in de macro-definities IKSMAX en UKSMAX.

Datavelden

De koppelsignalen worden weggeschreven naar de datavelden van een bericht. Het eerste dataveld van een bericht wordt gereserveerd voor het aangeven van de status van de seriële verbinding en het type van de applicatie. De status van de seriële verbinding wordt aangegeven in bit 0 t/m bit 2. Bit 0 wordt opgezet indien op een verzonden databericht een correct ACK-bericht wordt terugontvangen. Bit 1 wordt opgezet indien bit 0 waar is en een correct databericht wordt ontvangen en van dit ontvangen databericht in het eerste dataveld bit 0 waar is. Bit 2 wordt opgezet indien bit 1 waar is en een correct databericht wordt ontvangen en van dit ontvangen databericht in het eerste dataveld bit 0 en bit 1 waar zijn.

Bit 0 t/m 2 worden weer gereset indien de seriële verbinding is verbroken.

Het type van de applicatie wordt aangegeven in bit 3 t/m bit 7. Bij databerichten van het type koppelsignalen wordt bit 3 opgezet.

Vanaf het tweede dataveld kan de status van de koppelsignalen worden opgenomen. Koppelsignalen zijn booleaanse variabelen die de waarde waar (TRUE) of niet waar (FALSE) kunnen aannemen. Per databyte kunnen daarom 8 koppelsignalen worden opgenomen. Voor 16 koppelsignalen zijn dus 3 databytes nodig, 1 statusbyte en 2 bytes voor de koppelsignalen.

Inkomende koppelsignalen

Na ontvangst van een nieuw databericht wordt de nieuwe status van de inkomende koppelsignalen (mulv IKS[]) bepaald op basis van de inhoud van de datavelden.

Per systeemronde wordt maximaal één inkomend databericht verwerkt.

Gedurende de periode dat van de statusvariabele van de verbinding bit 0 en 1 nog niet beide waar zijn worden de inkomende koppelsignalen gereset (waarde 0).

Uitgaande koppelsignalen

De status van de uitgaande koppelsignalen (mulv UKS[]) wordt in de regelapplicatie bepaald. Voor het bepalen van het nieuwe uitgaande databericht wordt gebruik gemaakt van een hulpbuffer (byte HLP_US_DATA[]). In dit hulpbuffer worden de statuswijzigingen van de koppelsignalen weggeschreven. Hiertoe wordt de status van

de koppelsignalen in de zendbuffer (byte US_DATA[]) vergeleken met de huidige status van de koppelsignalen. Indien de waarden ongelijk zijn, wordt de nieuwe waarde naar de hulpbuffer weggeschreven.

Een nieuw databericht wordt verzonden indien:

- De zendtijd niet loopt ('zend-datavlag' is niet waar).
- De inhoud van de hulpbuffer ongelijk is aan de inhoud van de zendbuffer.
- Indien van de statusvariabele bit 1 en bit 2 nog niet beide waar zijn.
- Indien van het inkomende databericht bit 1 en bit 2 van de statusvariabele (eerste dataveld) nog niet beide waar is.

Voor het verzenden van een nieuw databericht wordt de inhoud van de hulpbuffer naar de zendbuffer gekopieerd.

Uitgebreide status (OKE-vlag)

De applicatie koppelsignalen kent een uitgebreide status melding.

BIT0 - goed databericht verzonden door SRC

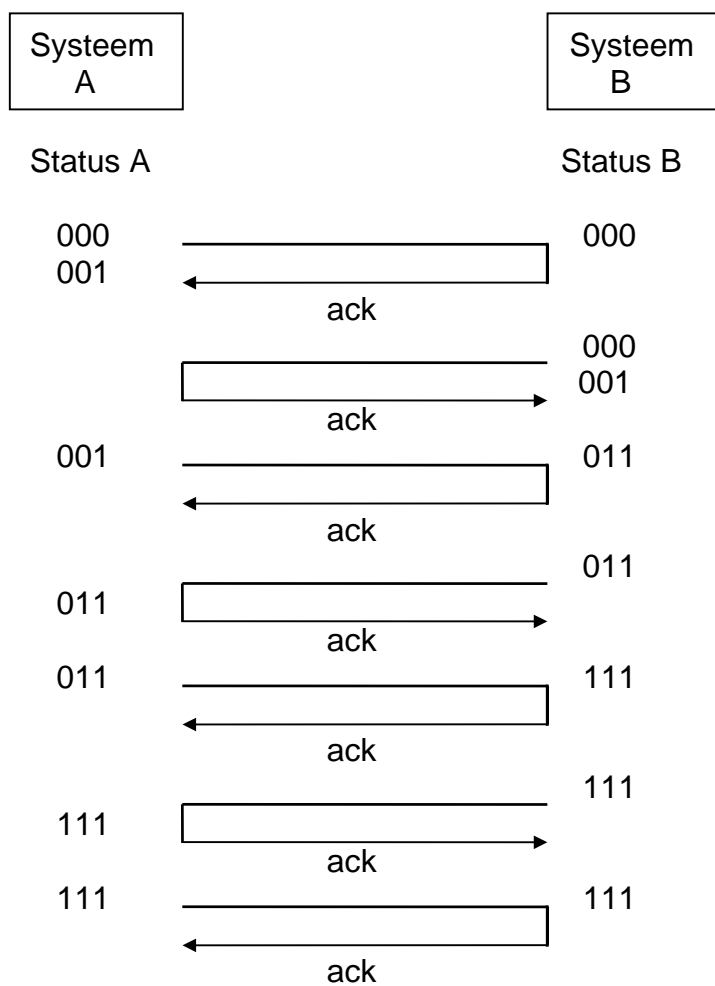
BIT1 - goed databericht verzonden door DEST (BIT0 is reeds waar)

BIT2- goed databericht ontvangen en BIT0 en BIT1 van SRC en DST zijn waar

Indien BIT2 waar is wordt de OKE-vlag waar.

De PTP-status en OKE-vlag worden niet waar indien de verbinding is verbroken.

Na het verbreken van de verbinding treedt de wachttijd in werking. Tijdens het lopen van de wachttijd worden inkomende berichten niet beantwoord, zodat van beide aangesloten systemen de verbinding wordt verbroken.

Opbouw van een goede PTP-verbinding (koppelsignalen)

PTP-PROTOCOL - INDELING VAN DE LAGEN

Inleiding

- driver laag
- enveloppe laag
- applicatie laag (b.v. gebruik koppelsignalen)

Driver laag

- functie voor initialiseren van de seriële poort
- Interrupt afhandeling voor inkomende en uitgaande karakters
- functie voor het initialiseren van de inkomende berichtenbuffer (lees- en schrijfpunter aan elkaar gelijk maken)
- functie die de aanwezigheid van karakters in de inkomende berichtenbuffer aangeeft
- functie voor het lezen van een karakter uit de inkomende berichtenbuffer
- functie voor het initialiseren van de uitgaande berichtenbuffer (lees- en schrijfpunter aan elkaar gelijk maken)
- functie voor het schrijven van een karakter naar de uitgaande berichtenbuffer
- (functie die de vrije ruimte in de uitgaande karakterbuffer aangeeft)
- XON/XOFF moet zijn uitgeschakeld

Files (Stevens) : serial_x.c, serial_x.h, serial8_15_x.h, rs232p_x.c

Files (Siemens): com.c, usercom.c, m pccomm.lib, cominter.h

Enveloppe laag

- functie voor het schrijven van een bericht naar de uitgaande berichtenbuffer
- functie voor het schrijven van een databericht naar de uitgaande berichtenbuffer
- functie voor het lezen van een bericht uit de inkomende berichtenbuffer
- functie voor het afhandelen van de procedures

Files: ptpdef.c, ptpccitt.c, ptpccitt.h, ptpvar.c ptpvar.h, ptpfunc.c, ptpcom.c, ptpcom.h

Applicatielaag Koppelsignalen

- vertalen van inkomende databerichten naar inkomende koppelsignalen
- resetten inkomende koppelsignalen indien geen goede verbinding
- vertalen van koppelsignalen naar uitgaande databerichten
- verzenden van uitgaande databerichten

Files: ptpkps.c, ptpkps.h, ptpksvar.c, ptpksvar.h, ptpksfun.c

Poort instellingen - PC104 (AMPRO)

Poort	Base	IRQ	Vector	PIC00	PIC01	bijzonderheden
1	0x3f8	4	12	0x20	0x21	
2	0x2f8	3	11	0x20	0x21	
3	0x3e8	7	15	0x20	0x21	
4	0x2e8	5	13	0x20	0x21	
5	0x220	2	10	0x20	0x21	
6	0x228	6	14	0x20	0x21	normaal: floppy disk

Opmerkingen

- De instellingen van de poorten 1 t/m 4 zijn gelijk aan de poortinstellingen van de interruptdriver van Siemens.
- De base-adressen en IRQ's dienen ook op de seriële kaart met behulp van jumpers te worden ingesteld.

Gebruik van de seriële poorten - PC104 (AMPRO)

Poort	Toepassing	Protocol	Baudrate	Bijzonderheden
1	Automaat	NH of PH-protocol	9600	geen xon/xoff
2	Terminal/Centrale	Terminal protocol	2400	xon/xoff (data)
3	Selectieve detectie	Vetag/Sics protocol	2400	geen xon/xoff
4	Seriële koppeling	PTP-protocol	2400	geen xon/xoff
5	Seriële koppeling	PTP-protocol	2400	geen xon/xoff
6	Seriële koppeling	PTP-protocol	2400	geen xon/xoff

Alle poorten

Baudrate: instelbaar
 databits: 8
 startbits: 1
 stopbits: 1
 parity: none (NH-protocol: odd)

Alternatieve instellingen voor poorten 5 en 6 - werkt echter (nog) niet

Poort	Base	IRQ	Vector	PIC00	PIC01	bijzonderheden
5	0x220	10	0x72	0xa0	0xa1	werkt niet
6	0x228	11	0x73	0xa0	0xa1	werkt niet

Overzicht globale variabelen koppelsignalen per seriële poort

```

#include "ptpvar.c"          /* structuur PTP-berichten          */
#include "ptpksvar.c"       /* structuur koppelsignalen        */

/* A-POORT */
/* ===== */
struct ptpstruct   PTPA;    /* definitie structuur PTP-berichten */
struct ptpksstruct PTPKSA; /* definitie structuur koppelsignalen */

mulv PTPKSA.IKS [IKSMAX]; /* inkomende koppelsignalen          */
mulv PTPKSA.UKS[UKSMAX]; /* uitgaande koppelsignalen          */

bool PTPA.ERROR      /* vlag verbindingfout                */
bool PTPKSA.OKE;     /* vlag goede verbinding              */

/* B-POORT */
/* ===== */
struct ptpstruct   PTPB;    /* definitie structuur PTP-berichten */
struct ptpksstruct PTPKSB; /* definitie structuur koppelsignalen */

mulv PTPKSB.IKS [IKSMAX]; /* inkomende koppelsignalen          */
mulv PTPKSB.UKS[UKSMAX]; /* uitgaande koppelsignalen          */

bool PTPB.ERROR      /* vlag verbindingfout                */
bool PTPKSB.OKE;     /* vlag goede verbinding              */

/* C-POORT */
/* ===== */
struct ptpstruct   PTPC;    /* definitie structuur PTP-berichten */
struct ptpksstruct PTPKSC; /* definitie structuur koppelsignalen */

mulv PTPKSC.IKS [IKSMAX]; /* inkomende koppelsignalen          */
mulv PTPKSC.UKS[UKSMAX]; /* uitgaande koppelsignalen          */

bool PTPC.ERROR      /* vlag verbindingfout                */
bool PTPKSC.OKE;     /* vlag goede verbinding              */

```

PTP-PROTOCOL - VOORBEELD GEBRUIK KOPPELSIGNALEN

Macrodefinities voor gebruik poort-A

Opgeven bij Turbo C++ compiler

Options/Compiler/Code generation/Defines: COMPORT_A=4;COMBAUD_A=2400

```
- COMPORT_A= 1      /* gebruik poort 4          */
- COMBAUD_A= 2400   /* baudrate: 2400 BAUD (default 1200) */
```

Projectfile - alle poorten

```
- ptpccitt.c
- ptpkps.c
- ptpfunc.c - afhandeling ptp-berichten
- ptpksfun.c - afhandeling koppelsignalen
```

*.SYS-file - alle poorten

```
#define CCOL_EXIT - t.b.v. afsluiten interrupt van de seriële poort
                    door de functie void CCOL_exit(void)
```

*.REG-file

```
#include "ptpvar.c"          /* definitie structuur ptp-berichten */
#include "ptpkpsvar.c"      /* definitie structuur koppelsignalen */

struct ptpstruct PTP19;    /* koppeling met kruispunt 19 */
struct ptpksstruct PTPKS19;

void control_parameters(void)
{
    PTP19.PORTNR= 3;        /* poort nummer          */
    PTP19.SRC = 1;         /* nummer van de source  */
    PTP19.DEST = 1;        /* nummer van destination */
    PTP19.TMSGW_max= 200;   /* wait time-out         */
    PTP19.TMSGGS_max= 10;   /* send time-out         */
    PTP19.TMSGGA_max= 10;   /* alive time-out        */
    PTP19.CMSGG_max= 3;     /* max. berichtenteller t.b.v. herhaling */

    PTPKS19.IKS_MAX= 16;    /* aantal inkomende koppelsignalen */
    PTPKS19.UKS_MAX= 16;    /* aantal uitgaande koppelsignalen */
}

/* declaratie communicatiefuncties voor PC */
/* ----- */
void communication_A(void);
void end_communication_A(void);
```

```
void system_application(void)
{
    if (SAPPLPROG) { /* start applicatieprogramma */
        ptp_init(&PTP19); /* initialisatie */
    }
    else {
        ptp_application_ks(&PTP19, &PTPKS19); /* afhandeling koppelsignalen */
        ptp_control(&PTP19); /* afhandeling ptp-berichten */
    }

    /* lezen koppelsignalen */
    /* ----- */
    IH[hiks0]= PTPKS19.IKS[0];
    IH[hiks1]= PTPKS19.IKS[1];

    /* schrijven koppelsignalen */
    /* ----- */
    PTPKS19.UKS[0]= G[fc02];
    PTPKS19.UKS[1]= G[fc08];
    PTPKS19.UKS[2]= D[d021];
    PTPKS19.UKS[3]= D[d081];

    /* foutmeldingen */
    /* ----- */
    IH[hptpoke19]= PTPKS19.OKE;
    IH[hptperr19]= PTP19.ERROR;

#ifdef CCOL_EXIT
    communication_A();
#endif

}

#ifdef CCOL_EXIT
void CCOL_exit(void)
{
    end_communication_A();
}
#endif
```