Contents

Lista över nödvändiga funktionsprimitiver

Följande funktioner har identifierats som nödvändiga. Tanken är att dessa, som rubriken anger, ska vara så primitiva som möjligt.

- Kontrollera drivning
  - Hastighet
  - Broms
  - Riktning
  - Inaktivera (?)
- Gafflar
  - Höja
  - Sänka
  - Inaktivera (?)
- Hydraulikfunktioner (exempel?)
- Skriva text på display
- Styra LED vid display
- Aktivera in-/utgångar
- . . . ?

Lista över nödvändiga signaler (realtidsvärden)

Det finns ett antal nödvändiga realtidsvärden, dynamiska värden som ändras över tid, som behöver skickas. Det finns även några värden som är statiska för sessionen.

För sessionen statiska värden

Dessa värden kan anses som statiska, och behöver därför endast skickas då de sätts, exempelvis vid uppstart.

- Förarprofil
- Truckkonfiguration i det fall ändring har skett (kontroll via checksumma)
- . . . ?

Dynamiska värden med frekvent förändring

Dessa värden uppdateras kontinuerligt medan trucken körs - en gång per programloop.

- Digitala ingångar på Spider-enheter (digital in)
- Digitala ingångar på ACT
- Knappstatus (exklusive num-pad, inklusive Belly Button)
- Nuvarande hastighet samt körriktning
- Tillåts körning?
- Gaffelhöjd
- Gaffelstatus
  - Lyft aktiv?
  - Sänk aktiv?

  - (Används av annan option? Finns funktion i kod, men kommer ej kunna vara pålitlig, i och med minimal delay på 60 ms)
- Tiller arm - status (styrarm?)

Frågor

- `SpiderSetupData[Address].SpiderFlag.optiontrig` - vad innebär denna flagga?

Händelseflöde för optioner

Kondenserad beskrivning över händelseförloppet för optionshanteringen och dess kommunikation mellan OCU och MCU.

Uppstart

Detta steg är inte på något vis färdigtänkt, och det är heller inte den stora delen av huvudproblemet.

Vid truckens uppstart kommer OCU och MCU kommunicera med varandra för att säkerställa att OCU har en korrekt och up to date datauppsättning. Om trucken har genomgått service och/eller mjukvaruuppdatering kan parametrar ha förändrats. Dessa måste därför synkroniseras över till OCU. För att spara bandbredd över den bandbreddsfattiga CAN-bussen skickas först en checksumma. Endast om denna inte är lika för båda enheter, begärs all data på nytt. Det är i detta läge helt acceptabelt med en längre uppstartstid.

Steg 1

MCU kommer kontinuerligt att läsa av tidigare identifierade som nödvändiga värden och skicka dessa till OCU. Detta sker en gång per programloop vilket är detsamma som en gång per 20 ms.

Steg 2

OCU väntar in nytt data från CAN-bussen. Eftersom CAN-bussen är interruptsstyrd kommer optioner beräknas kontinuerligt, medan ny data läses in. Detta eftersom det kan vara farligt att vänta in alla data och sedan beräkna optioner - skulle det vara trångt på bussen kan det tänkas dröja längre än 20 ms innan alla värden har skickats. Skulle OCU då vänta in alla värden uppstår följande: - Antalet uppdaterade värden måste skickas - Den maximala fördröjningen i systemet kan inte längre antas vara 60 ms

Steg 3

När en option har beräknats (dess villkor har kontrollerats) kan det uppstå ett behov av att anropa en funktion eller sätta en begränsning på MCU, eller annan enhet. För att inte CAN-bussen ska bli helt fylld med identiska begäran kontrolleras dessa innan de läggs till CAN-kön för att se att det inte redan existerar. Detta innebär att om en option träder i kraft vid tiden *T1*, och inga förändringar i någon variabel sker förrens i *T8* har endast en omgång av det option skulle genomföra skickats - vid *T1*.

Detta steg är alltså ett kontinuerligt steg. Det sker helt parallellt[1] med CAN-busshändelser.

[1] Så parallellt en enkelkärning CPU tillåter

Produkten som en option skapar (begränsning/funktionsanrop) paketeras i ett för OCU och MCU enigt format. Paketet innehåller information om

- Enhetsadress[2]
- Angivet system, som kan vara *drive*, *fork*, eller något annat
- Önskad funktion (funktionsanrop/begränsning)
- Argument

Steg 4

MCU kommer varje loopiteration att behandla inkomna funktionsanrop/begränsningar från OCU. Enligt den gemensamma protokolldefinitionen behandals alla begäran.

En funktion anropad från OCU behöver inte nödvändigtvis utföras på MCU. Ett anrop från OCU som ska utföras på densamme ska alltid skickas via MCU. OCU kan även skicka vanliga anrop till ACT, men dessa måste då alltså gå via MCU eftersom något annat med högre prioritet kan överskriva detta anrop.

Detta innebär också att även OCU implementerar detta protokoll, dock i en förenklad form där adress blir oväsentligt (vad vi vet). För att inte blanda ihop dynamiska värden med funktionsanrop kommer de att skickas med olika enhets- ID[3]. Detta kommer bli det första som tittas på när de tas emot, för att därefter avgöra hur meddelandet ska behandlas.

[2]SEU1/SEU2 - inte nödvändigtvis CAN-buss-ID, snarare 0 eller 1
[3]Det `id`-fält som finns i `tCanMsg`