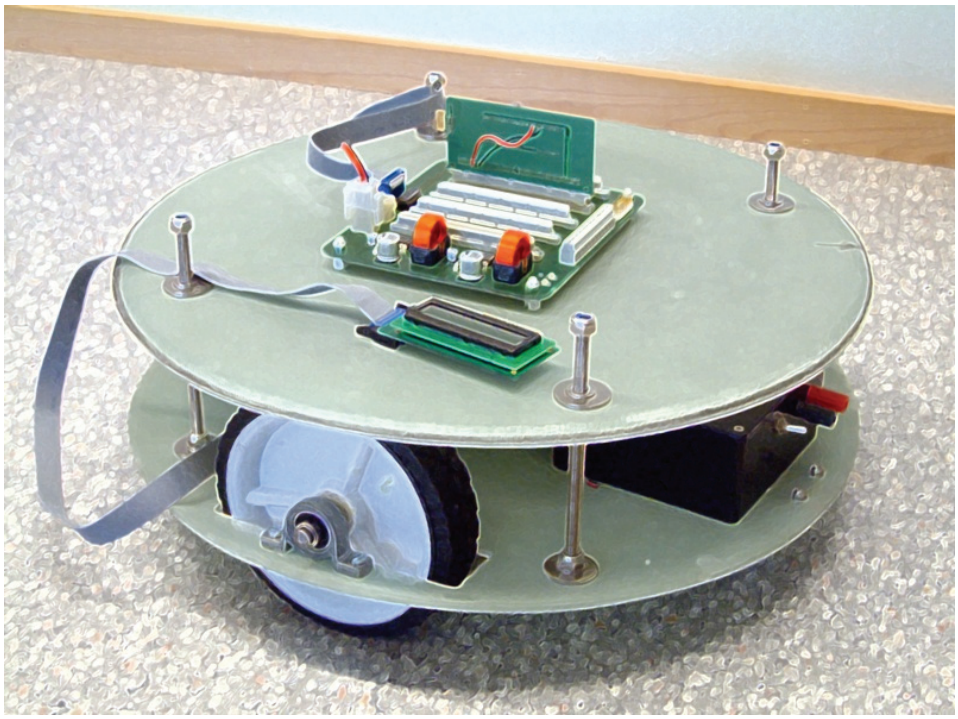
BRUKSANVISNING



2007-10-29

Bruksanvisning för roboten Saphira

Revisionshistoria

Datum	Beskrivning
2007-08-15	Första utgåva
2007-10-29	Uppdatering av firmware till version 1.1 med nya kommandon för att styra hastigheten.

Innehållsförteckning

BRUKSANVISNING

SAPHIRA

INLEDNING

Denna användarhandledning består av fyra delar. Den första delen är allmänna skötselinstruktioner för roboten. Del två beskriver kortfattat funktionen av motorstyrningskortet. Del tre beskriver kommunikationsprotokollet och del fyra visar ett exempel på hur mönsterkortet skall utformas för att passa bakplanet. För att använda sig av denna bruksanvisning förutsätts att läsaren har goda kunskaper i konstruktion med mikrodatorer, CAD med AltiumDesigner6 och C programmering.

Designunderlag finns att hämta på stud_dir under katalogen *\TN\E\076_Design av autonoma robotar\Saphira*. Dessa är till för att snabbt komma igång med design av hårdvara och mjukvara för styrning av roboten.

SKÖTSELINSTRUKTIONER FÖR ROBOTEN

Det viktigaste att sköta på rätt sätt är batteriet. Batteriet består av 4st ackumulatorer på vardera 6V och 1800mAh. Dessa seriekopplas för att ge utspänningen 24V. Laddningen sker genom att ansluta ett nätaggregat. Ställ in spänningen på 28V innan aggregatet ansluts till roboten. När det är anslutet justeras laddströmmen till 10 procent kapaciteten och i detta fall blir det 180mA. Batteriet är fullladdat när laddströmen sjunker under ca 10mA. Detta kan ta upp till 12h när batteriet är helt urladdat. Observera att det endast går att ladda roboten när strömbrytaren är i laddläget och roboten är strömlös.

Roboten får inte dras på golvet då motorerna inducerar en spänning och i värsta fall med omvänd polaritet. Risken är då mycket stor att elektroniken på motorkortet förstörs!

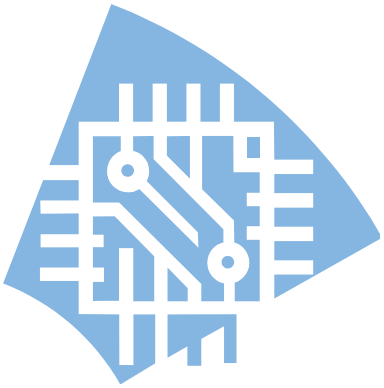
Innan körning skall kontrolleras att alla skruvar är åtdragna. Kontrollera även att inga sladdar kan fastna i hjulen då roboten är mycket stark kommer det att bli skador om en kabel fastnar i ett roterande hjul! Gäller även fingrar!

Placera alltid roboten på ett stöd så att hjulen inte rör marken under utvecklingen, så slipper den åka iväg och krascha ner från bordet. Vi har bara en robot!

Om säkringen går får den endast ersättas med en snabb säkring på 500mA. Säkringen skall gå sönder om roboten kör in i tillexempel en vägg!

Se dock alltid till att roboten inte kan köra emot fasta föremål som tillexempel en vägg.

I övrigt tänk först handla sedan. Även då du är trött, stressad och klockan är mycket!



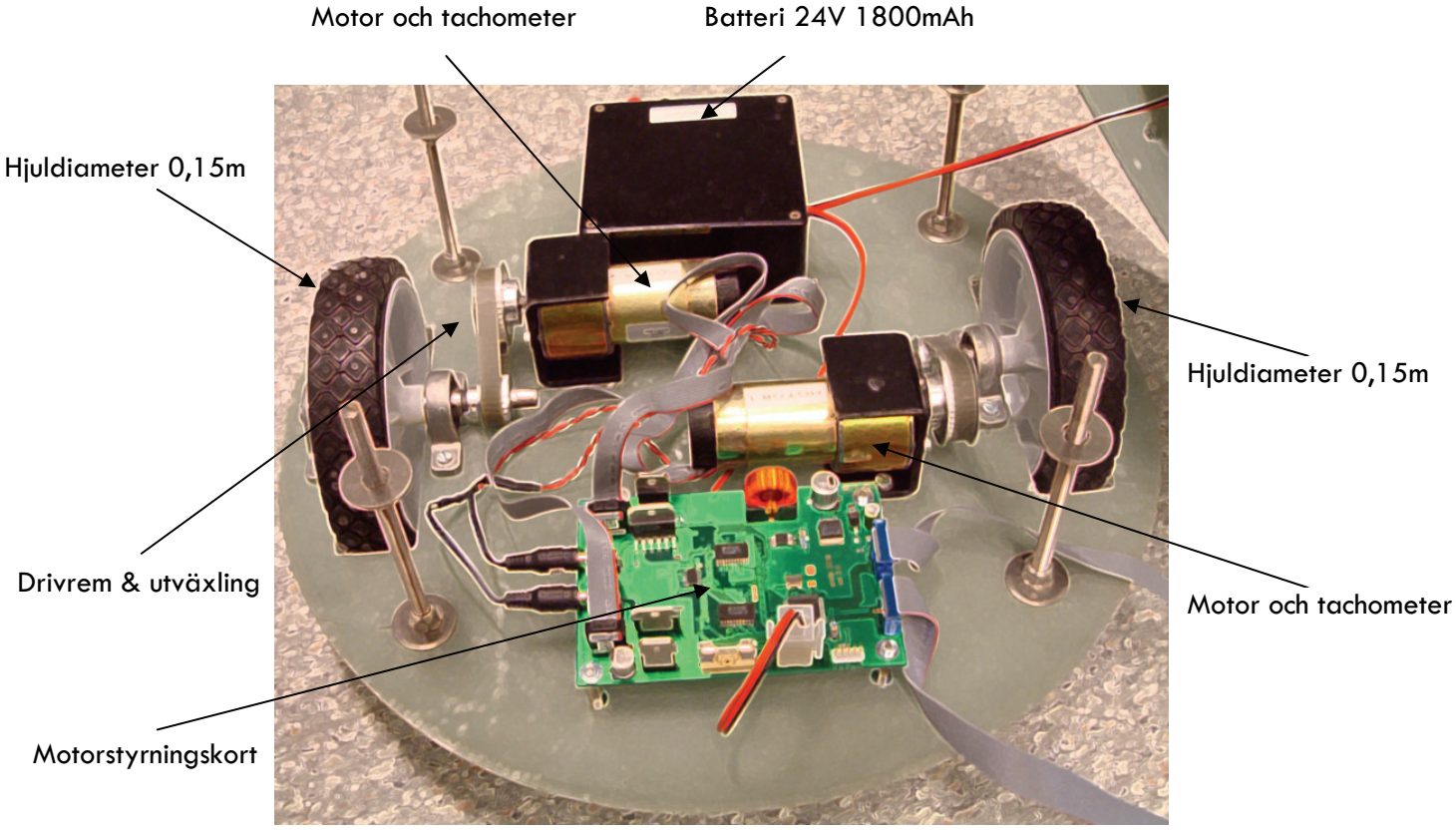
ROBOTENS UPPBYGGNAD

Inledning

Robotplattformen består av två stycken DC motorer med pulsigvare, motorstyrningskort, batteri och ett bakplan. Motorerna kontrolleras av motorstyrningskortet vars huvuddelar består av två kontrollkretsar (LM629 en för varje motor) samt en mikrokontroller (Atmega128) för programmering av kontrollkretsarna och kommunikation med kretskorten på bakplanet. Bakplanet är placerat ovanpå roboten och har plats för 5 stycken kretskort. Kretskorten kommunicerar med motorstyrningskortet via en RS232-länk och kan matas med 5 eller 3.3 volt.

Basplattan

På basplattan är de delar som tillhör drivpaketet monterade. Dessa är hjul, drivremmar med kuggremshjul, batteri samt drivelektroniken i form av motorstyrningskortet. Bilden nedan visar var dessa delar är placerade.

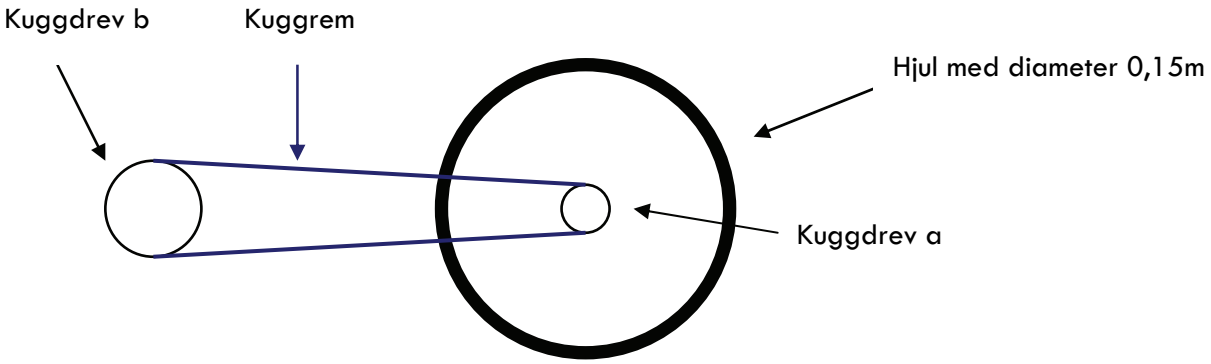


Infromationsdisplay

Till motorstyrningskortet finns en informationsdisplay kopplad. Den ger information om robotens status. Vid uppstart visas en text om att roboten måste aktiveras samt en rad med batterispänningen. Under körning visas vilken position som höger och vänster hjul befinner sig på.

Utväxling

Beräkningen nedan visar hur antalet pulser/m och maxhastighet beräknas. OBS att även om det är många decimaler med i beräkningen kan en viss kalibrering vara nödvändig!



Omkrets för driv hjul och kuggremshjul

Hjulomkretsen är π * 0,15 = 0,47234 m = c

Omkretsen för motorkuggdrev är 0,03820 m = a

Omkretsen för hjulaxelkuggdrev är 0,02387 m = b

Distans för varje varv motorn snurrar

D = (a/b)*c = 0,754140 m

Maxhastighet:

Enligt databladet till motorn är rotationshastigheten på utgående axel 91 varv/minut vid 24 V.

91/60 * 0,754140= 1,14 m/s

1,14 * 3.6 = 4,1 km/h

Antal pulser för given distans som skall skickas till motorstyrningskortet:

32768 pulser/varv på motoraxeln i LM629 kretsen ökar upplösningen med 4. Detta beror på hur signalerna från tachometern behandlas i LM629 kretsen. Se sidan 9 & 10 i databladet till LM628/LM629.

.

Systemets upplösning R = 32768 * 4 = 131072

L = 1m

Ger en upplösning på L/(D/R) = 1 / (0,754140 / 131072) = 173803.32 pulser/m = $2A6EB_{16}$

BAKPLANET

Inledning

Till bakplanet skall de egenkonstruerade kretskorten anslutas. Detta avsnitt beskriver bakplanets funktion.

Matningspänning

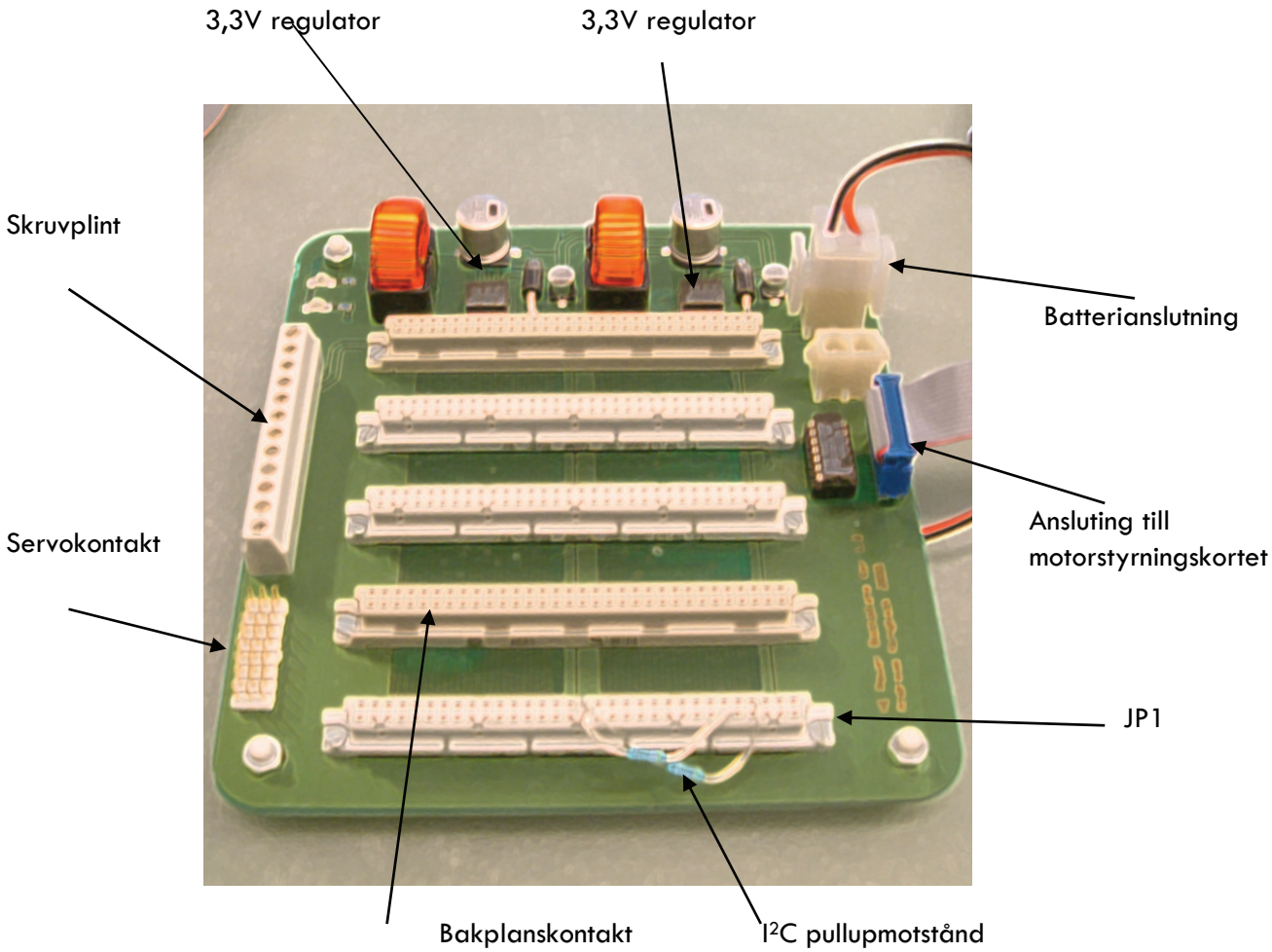
Bakplanet är direkt anslutet till batteriet på 24V. På kortet sitter sedan två stycken switchregulatorer som omvandlar 24V till 5,0V och 3,3V. Dessa två spänningar leds sedan ut till bakplanskontakterna. Strömmen får dock inte överskrida 5A. Om 24V behövs kan en sladd direkt anslutas till den extra batterikontakten som är placerad vid sidan av den som ansluter matningsspänningen till bakplanet.

Kommunikation

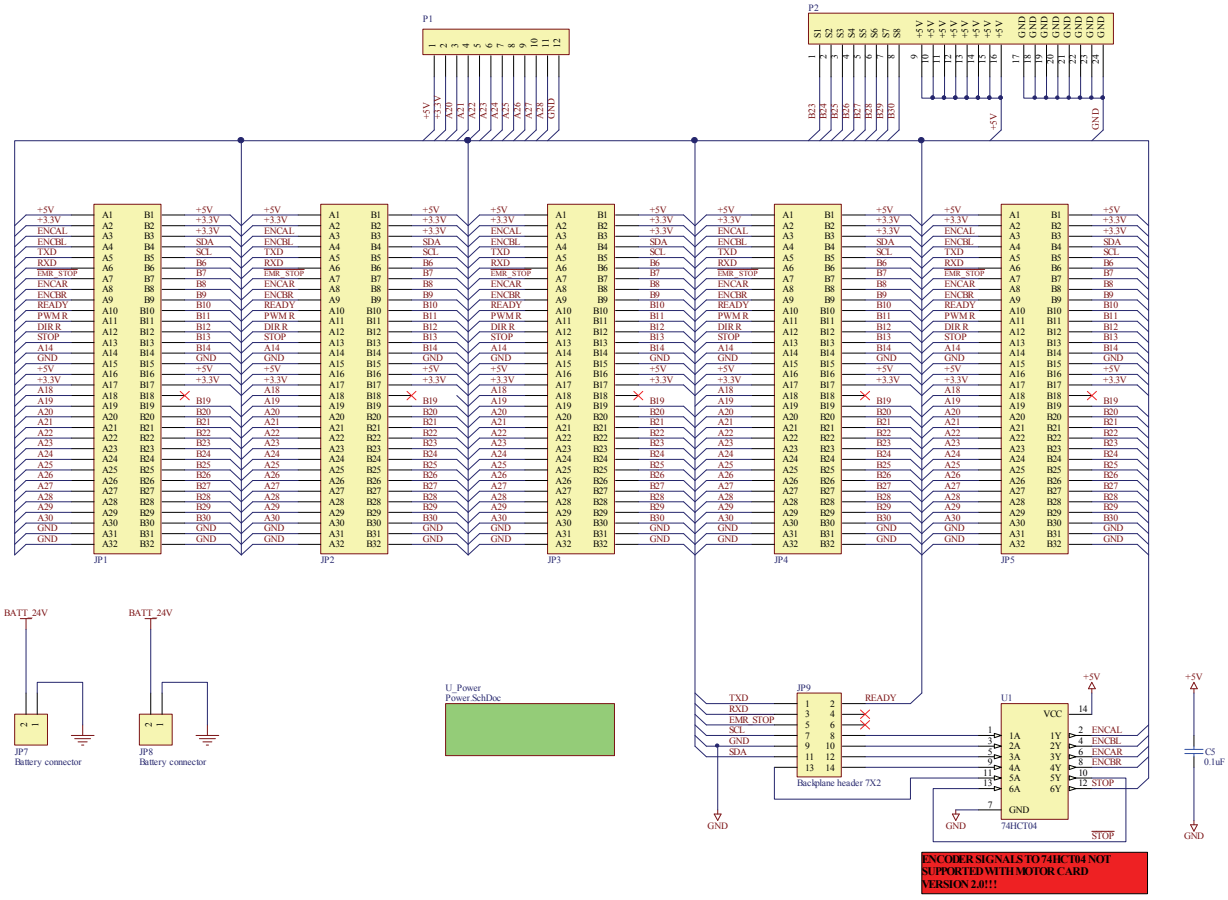
Från kortet går en flatkabel till motorstyrningskortet. Denna kabel används för kommunikationen via RS232. Det finns även i framtiden möjlighet till I²C kommunikation när mjukvaran har uppdaterats.

Utseende

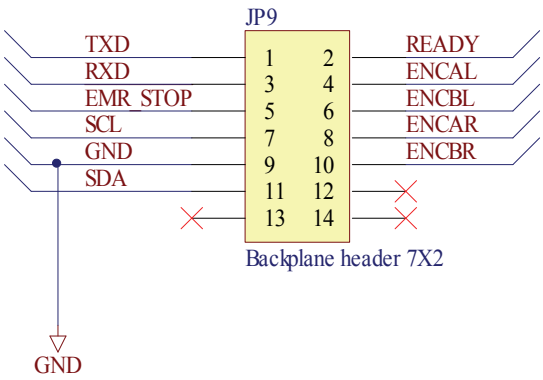
Bilden nedan visar bakplanet och vart de olika delarna är placerade.



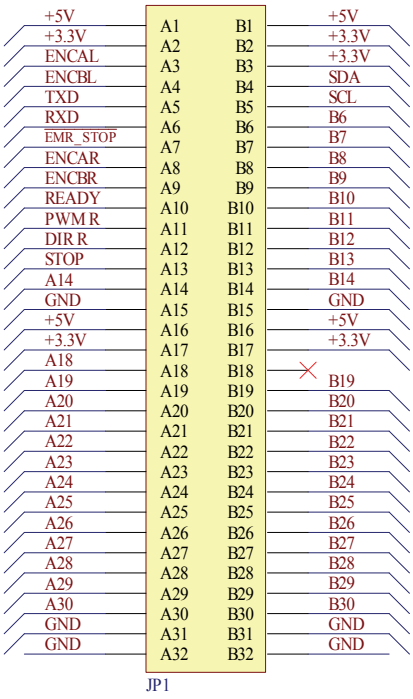
Schema för bakplanet



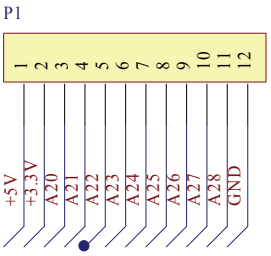
FIGUR 1: SCHEMA BAKPLAN



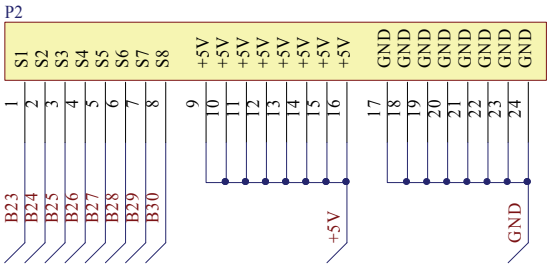
FIGUR 2: SCHEMA KONTAKT BAKPLAN / STYRKORT



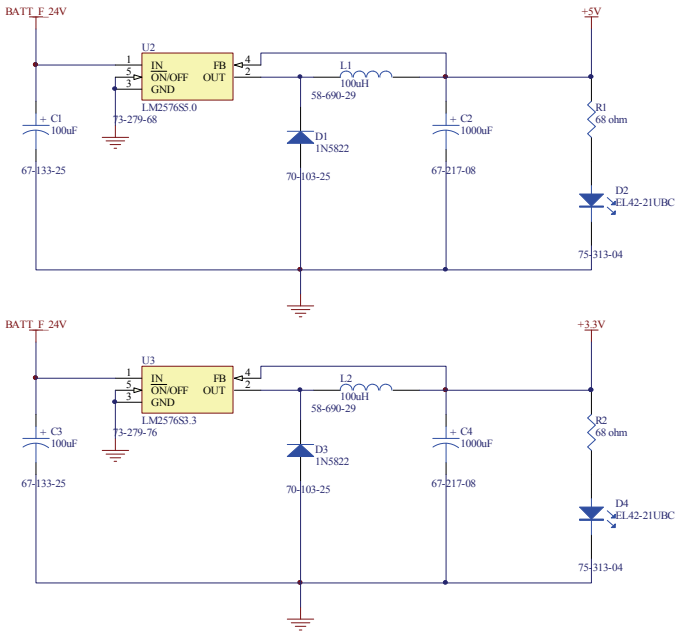
FIGUR 5: SCHEMA BAKPLANSKONTAKT



FIGUR 3: SCHEMA SKRUVPLINT



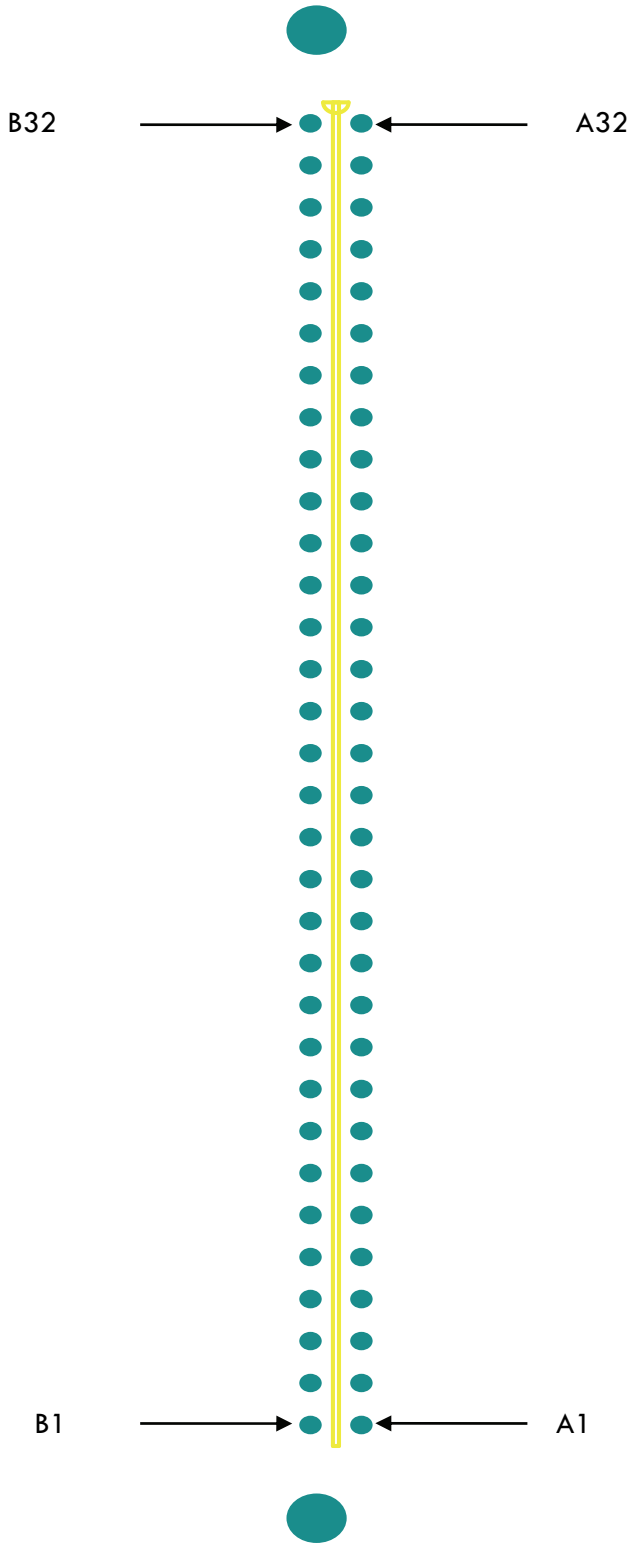
FIGUR 4: SCHEMA SERVOKONTAKT



FIGUR 7: SCHEMA SWITCHREGULATOR 5 & 3.3V

Mönsterkortssymbolen för bakplanskontakten

Bakplanskontakten sedd ovanifrån. OBS skillnaden mellan schemasymbolen!

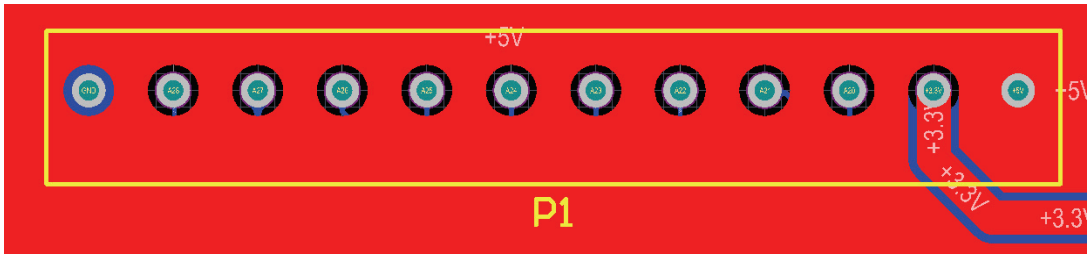


FIGUR 8:
MÖNSTERKORTSSYMBOL
BAKPLANSKONTAKT

Mönsterkortssymbolen för skruvplinten

Skruvplinten sedd ovanifrån:

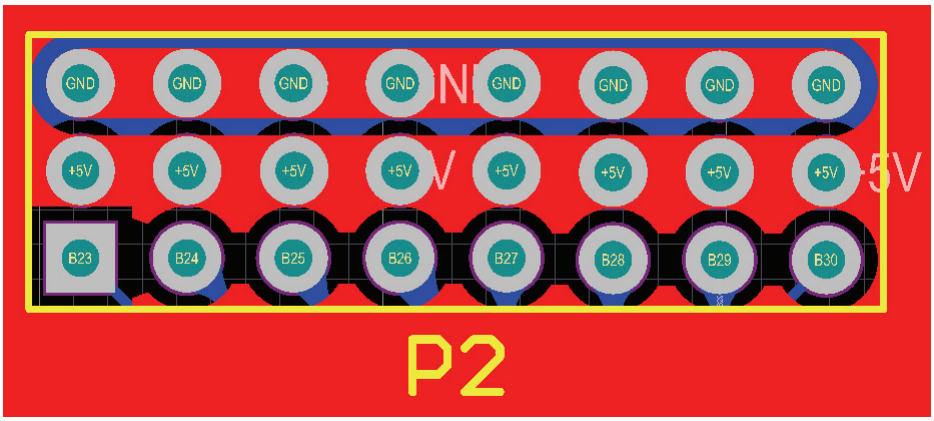
- Pinnumrering från vänster:
  - GND, A28,A27,A26,A25,A24,A23,A22,A21,A20,+3,3V & + 5,0V



Mönsterkortssymbolen för servokontakten

Kontakt för anslutning av servon sedd uppifrån

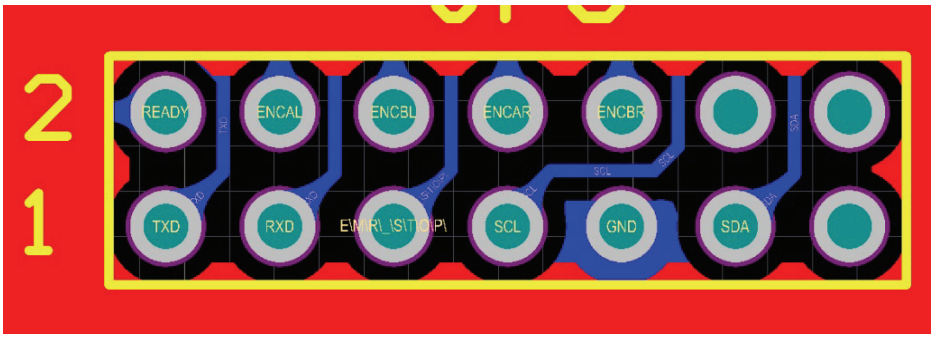
- Pinnumrering från vänster
  - Rad 1: GND
  - Rad 2: +5,0V
  - Rad 3: B23, B24, B25, B26, B27, B28, B29 & B30



Mönsterkortssymbolen för anslutning bakplan / styrkort

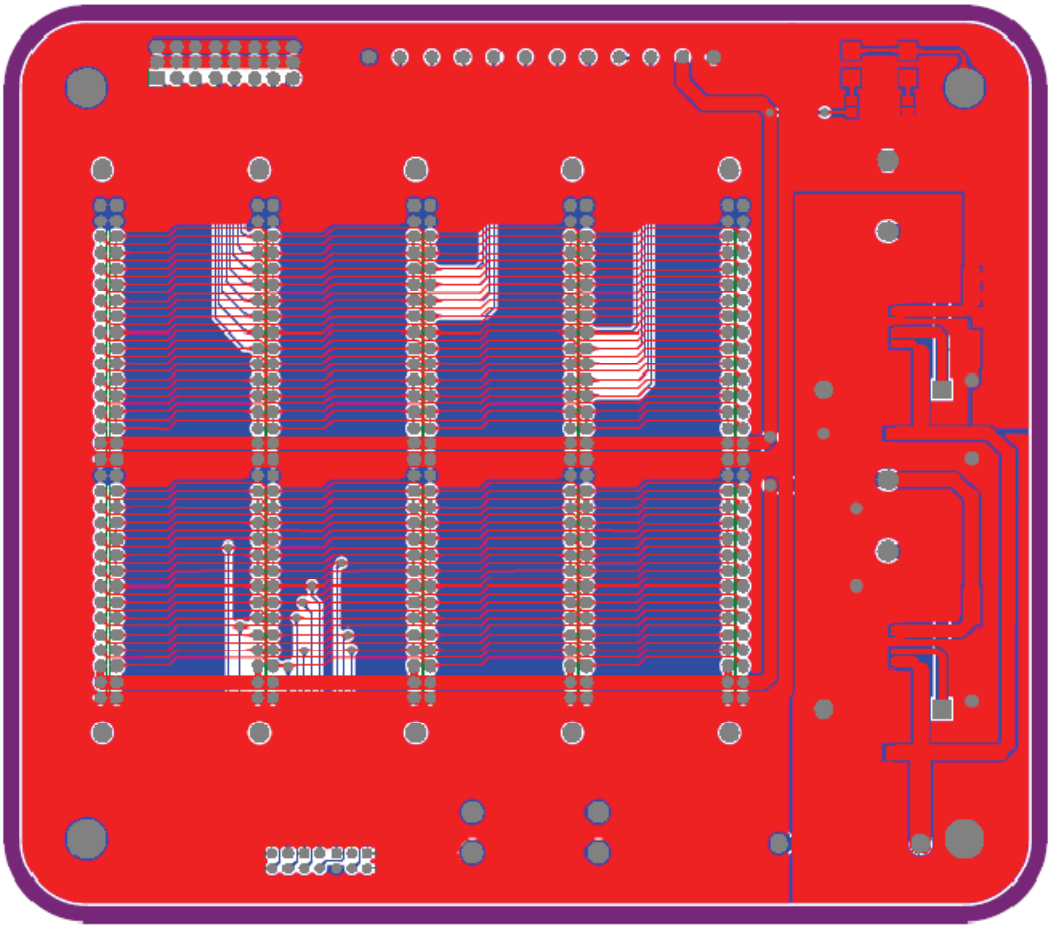
Kontakt för anslutning av kabel mellan bakplan och styrkort

- Pinnumrering
  - 1: TXD , 3: RXD , 9: GND (Övriga används inte)



Mönsterkortslayout för bakplanet

Nedan visas layouten för bakplanet



FIGUR 9: MÖNSTERKORTSLAYOUT FÖR BAKPLANET

KOMMUNIKATIONSPROTOKOLLET

Inledning

Kommunikationsprotokollet är till för att överföra kommandon från styrkortet till motorkortet via RS232. Protokollet är enligt KISS (keep it simple stupid) och är i skrivande stund inte testat i någon större utsträckning men har visat sig relativt stabilt om man följer reglerna nedan!

Initiering

Av säkerhetsskäl lägger sig motorstyrningskortet i spärrat läge när strömmen slås på och måste aktiveras. Motorstyrningskortet sänder då med ca 1sek mellanrum ut ett '?'. Styrkortet skall då svara med ett 'A'. Därefter är motorstyrningskortet aktiverat och kan ta emot styrkommandon.

Eko

För att veta när styrkortet tagit emot ett tecken skickas det i retur. Innan tecknet kommit i retur får inte nästa tecken skickas till styrkortet.

Checksumma

All data som skickas till roboten förutom aktiveringen skall följas av en 16-bitars checksumma enligt CCITT. Algoritmen kallas även för CRC (cyclic redundancy check). Checksumman beräknas med formeln $x^{16} + x^{12} + x^5 + 1$. Observera att det är ett 16 bitars tal och skickas efter data som msb + lsb. Implementering av funktionen återfinns i subrutinen crc16 nedan.

Kommandon som stöds i version 1.1

I versionen 1.1 av styrprogramvaran stöds kommandona nedan. Detta ger viss begränsning när det gäller hur roboten kan styras. T.ex. är det svårt att få roboten att köra i en cirkel eller svänga med radien > 0.

Kommando	Antal bytes	Kod	Beskrivning
CM_LEFT_LOAD_POS	4	20	Ladda position för höger motor
CM_RIGHT_LOAD_POS	4	21	Ladda position för vänster motor
CM_BOTH_LOAD_POS	4	22	Ladda samma position till båda motorerna
CM_LEFT_START_MOTION	0	23	Starta förflyttning för vänster motor
CM_RIGHT_START_MOTION	0	24	Starta förflyttning för höger motor
CM_BOTH_START_MOTION	0	25	Starta förflyttning för båda motorerna
CM_LEFT_SET_MAX_SPEED	1	26	Sätt max hastighet för vänster motor
CM_RIGHT_SET_MAX_SPEED	1	27	Sätt max hastighet för höger motor
CM_BOTH_SET_MAX_SPEED	1	28	Sätt samma max hastighet för båda motorerna

Efter att godkänt kommando och data mottagits av motorstyrkortet returneras koden CM_OK eller CM_FAILED.

Svar efter utförd förflyttning

När motorn har roterat till angiven position skickas följande svar:

TC[L/R][b3][b2][b1]b[0]

TC står för Traction Control

[L/R] antingten L=left eller R=right

[b3] msb av 32-bitars possition

[b0] lsb av 32-bitars possition

När höger motor har stannat vid angiven position (0 i detta fall) skickas följande:

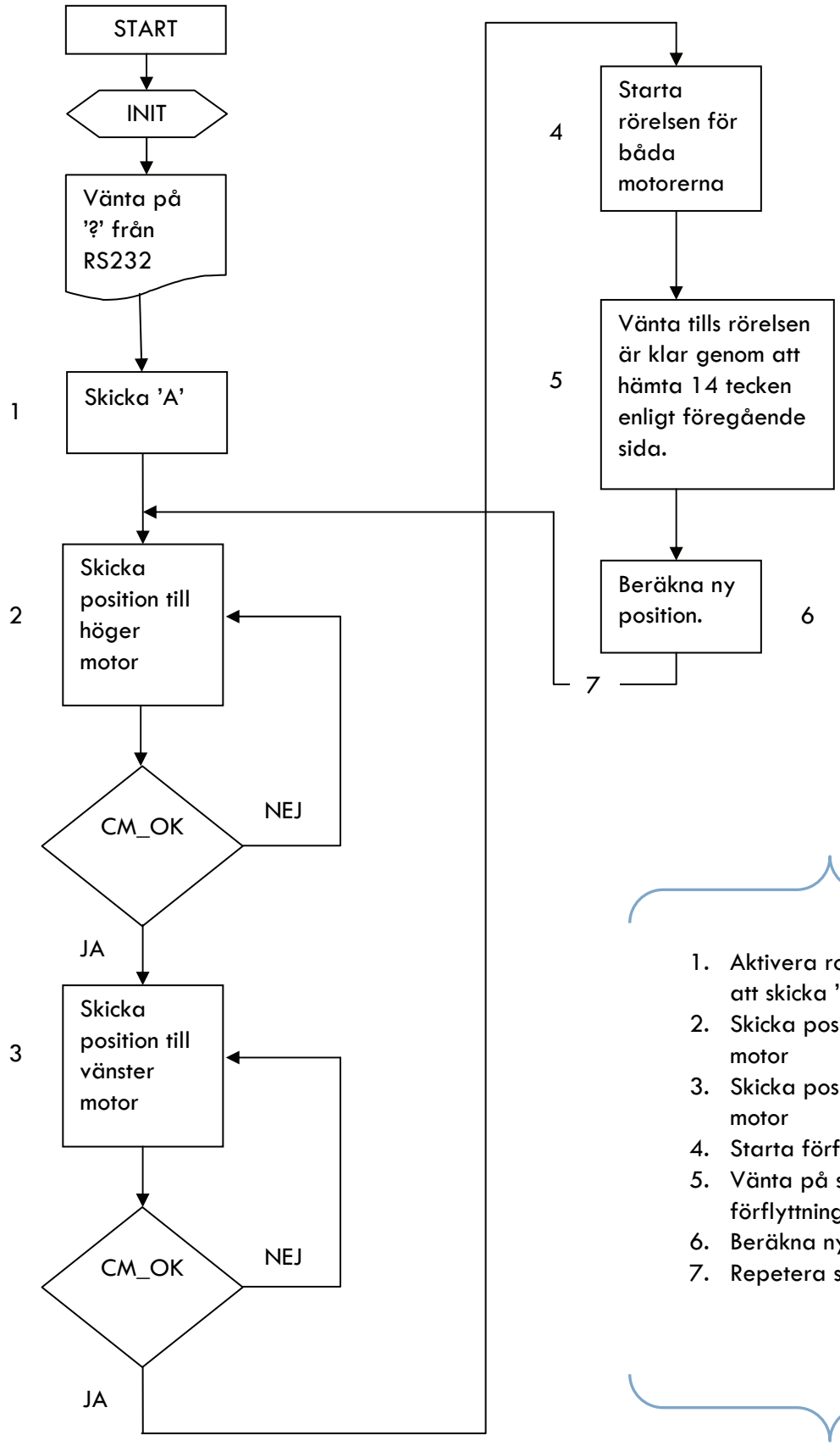
TCR'0''0''0''0' '0' = char 0

När vänster motor har stannat vid angiven position (0 i detta fall) skickas följande:

TCL'0''0''0''0' '0' = char 0

Positionen som returneras är begärda positionen +- ett positioneringsfel. Vilket beror av reglerparametrarna och vilka störningar som uppstår under förflyttningen.

Flödeschema



1. Aktivera roboten genom att skicka 'A'
2. Skicka position för höger motor
3. Skicka position för vänster motor
4. Starta förflyttningen
5. Vänta på svar att förflyttningen är utförd.
6. Beräkna ny position
7. Repetera steg 2 till 6

EXEMPEL PÅ EN ENKEL DESIGN AV ETT STYRKORT

Inledning

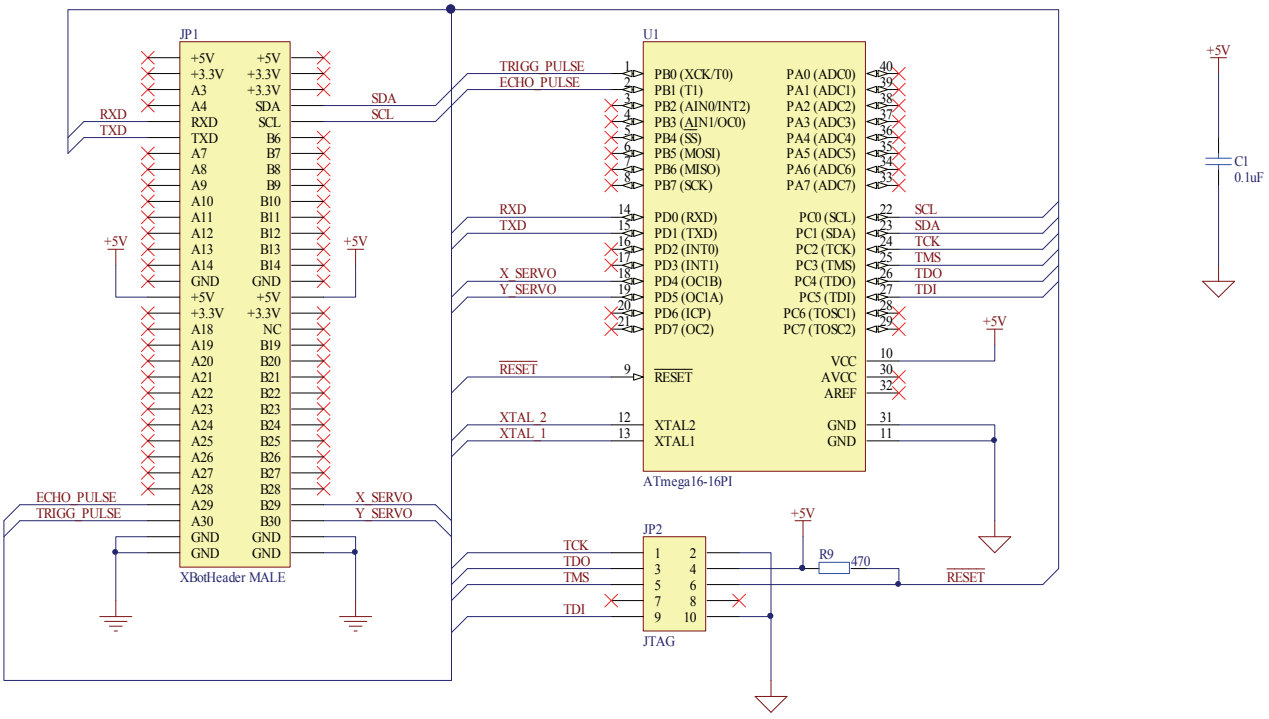
Detta kort skall ses som ett exempel på hur ett enkelt styrkort till roboten kan utformas utifrån de filer som finns tillgängliga på kurskatalogen under *\TN\E\076_Design av autonoma robotar\Saphira*hardware\saphira_controler

Funktion

Denna hårdvara är till för att göra det möjligt att kommunicera med styrkortet utifrån de regler som sätts upp i mjukvaran.

Styrkortets uppbyggnad

Konstruktionen utgår från en Atmega16 processor med JTAG interface. Utöver kommunikationen med motorstyrningskortet finns kopplingar till ultraljudssensorn, I²C-bussen och två servon. Processorn matas med 5V via bakplanet. Programmeringen sker via JTAG kontakten JP2 och kortet ansluts till bakplanet via kontakten JP1. Kommunikationen med motorstyrningskortet sker via anslutningarna RXD och TXD. I övrigt används den interna klockgeneratorn på 8MHz. Nedan visas det fullständiga schemat för styrkortet.



FIGUR 10: SCHEMA STYRKORT

Schemaritning

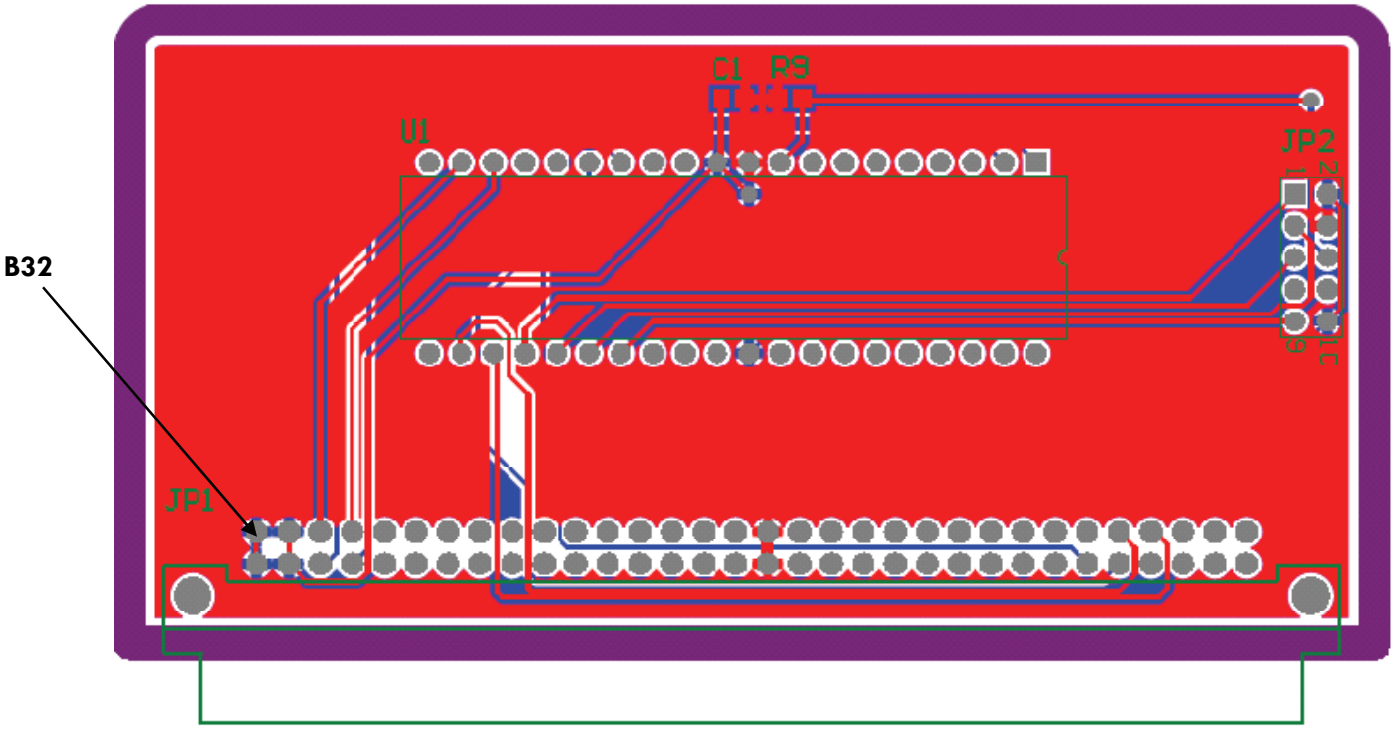
Till AltiumDesigner 6 finns ett bibliotek med specifika schemasymboler och en mall för schemat som bör användas vid ritning av schemat. Namnet på biblioteket är *saphira_controler.IntLib* och ligger i samma filkatalog som övriga filer. Se även manual för AltiumDesigner.

Mönsterkortet

Det viktiga vid designen av mönsterkortet är att kontakten JP1 kommer på rätt plats. Annars kommer inte pinnumreringen och avståndet till den nedre fräskanten att stämma. B32 (GND) ligger längst upp till vänster. Använd därför den färdiga mallen med placeringen av JP1 och fräsramen. Givetvis kan fräsramen ändras så att kortet blir större men tänk på att inte ändra läget på den nedre kanten i Y-led! I mallen finns även färdiga regler för ledarbredd och avstånd för polygonplanen till ledare och fräskant.

Använd med fördel ett jordplan och ett plan för matningsspänningen.

Figuren nedan visar det färdiga mönsterkortet.



Mönsterkortsritning

Till AltiumDesigner 6 finns ett bibliotek med specifika kretssymboler och en mall för kort som bör användas vid ritning av mönsterkortet. Namnet på biblioteket är saphira.intlib.
Namnet på biblioteket är *saphira_controler.IntLib* och ligger i samma filkatalog som övriga filer. Se även manual för AltiumDesigner.

MJUKVARA

Inledning

Detta program skall ses som ett exempel på hur roboten kan styras. För att förstå koden krävs god kännedom om C programmering och om Atmega16. För specifik information om Atmega16 hänvisas till databladet. Programfilerna återfinns på stud_dir under katalogen \TN\E\076_Design av autonoma robotar\Saphira\Dokumentation\Software\saphira_control\v0.1

Funktion

Detta programs funktion är att styra roboten så att den åker i fyrkanter med sidan 2m.

Uppbyggnad

Mjukvaran består av ett antal funktioner för kommunikation, beräkning av checksumma samt ett huvudprogram som sköter styrningen av roboten.

Filerna som bygger upp programmet enligt nedan:

Fil	Beskrivning
`main.c`	Huvudfunktioner för styrning av robot och beräkning av crc
`usart.c`	Funktioner för kommunikation via RS232
`usart.h`	Definitioner för kommunikation vi RS232
`control.h`	Definitioner för kommunikationsprotokollet

Kort berskrivning av de olika subrutinerna

- **main.c**
  - main()
    - Huvudprogram för styrning av roboten
  - void delay_1s(void)
    - Väntar 1 sekund
  - void crc16(unsigned char ser_data)
    - Beräknar crc16. OBS använder sig av globala variabeln crc som måste sättas till 0 innan beräkningen påbörjas.
  - void send_byte(char data)
    - Skickar en byte via RS232 till styrkortet
  - unsigned char send_pos(unsigned char lr, long int pos)
    - Skickar ny position till vald motor via RS232
  - long int turn(int r, float v)
    - Beräknar antalet pulser för att svänga v antal grader. OBS inte helt kalibrerad.

- **usart.c**
  - void USART_Init(double baud)
    - Initierar USART i Atmega16 med given baudhastighet. I detta fallet 9600baud.
  - int USART_Receive_time_out(void);void USART_Init(double baud);
    - Hämtar ett tecken från buffer I Atmega16. Hoppar ur efter ca 1 sek om inget tecken har kommit.
  - void USART_Transmit(unsigned char data);
    - Skickar ett tecken via RS232
  - void USART_Transmit_Str(const char *str);
    - Skickar en sträng via RS232
  - void USART_Flush(void);
    - Tömmer motagarbufferten i Atmega16

Övrigt

För övrig information se kommentarerna I koden.

EXEMPELKOD FÖR ATMEGA16

main.c

```
/*

    Shapira control demo (main.c) for Atmega16 for IAR

    Version 0.2         2007-10-29

    Copyright (C) Andreas Kingbäck  (andki@itn.liu.se)

   This program is free software; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by the
    Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

   This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with this program;
    if not, write to the Free Software Foundation, Inc.,
    59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

*/

#pragma language=extended

#ifndef ENABLE_BIT_DEFINITIONS
#define ENABLE_BIT_DEFINITIONS
// Enable the bit definitions in the iom128.h file
#endif

#include "control.h"
#include "usart.h"

static unsigned int crc;                              // global for crc calculation

void delay_1s( void )
{
   __delay_cycles(8000000); // Delay 1s
}


void crc16(unsigned char ser_data)
{
  // Update the CRC for transmitted and received data using
  // the CCITT 16bit algorithm (X^16 + X^12 + X^5 + 1).

  crc = (unsigned char)(crc >> 8) | (crc << 8);
  crc ^= ser_data;
  crc ^= (unsigned char)(crc & 0xff) >> 4;
  crc ^= (crc << 8) << 4;
  crc ^= ((crc & 0xff) << 4) << 1;
}


void send_byte(char data)
{
   int test = -1;
   while (test == -1) {
    USART_Transmit(data);
   test = USART_Receive_time_out();
   }
}
```

```
unsigned char send_pos(unsigned char lr, long int pos)
{
   unsigned char b,b1,b2,b3;                                  // type conversion

   USART_Flush();                                             // Flush rx buffer

  // Reset crc

  crc = 0;

  // Connvert to bytes

  b  = (pos & 0xFF000000) >> 24;
  b1 = (pos & 0x00FF0000) >> 16;
  b2 = (pos & 0x0000FF00) >> 8;
  b3 = (pos & 0x000000FF);

  // Calculate crc for data

  crc16(lr);
  crc16(b);
  crc16(b1);
  crc16(b2);
  crc16(b3);

 // Send command

   USART_Transmit(lr);
  USART_Receive_time_out();

  // Send data

  USART_Transmit(b);
  USART_Receive_time_out();

  USART_Transmit(b1);
  USART_Receive_time_out();
  USART_Transmit(b2);
  USART_Receive_time_out();
  USART_Transmit(b3);
  USART_Receive_time_out();

  // Send crc16

  USART_Transmit((crc & 0xFF00) >> 8);
  USART_Receive_time_out();
  USART_Transmit((crc & 0xFF));
  USART_Receive_time_out();

  // Get acc  (10101010=OK)

  if (USART_Receive_time_out() == CM_OK)
    return CM_OK;
  else
    return CM_FAILED;
}
```

```
unsigned char send_max_speed(unsigned char lr, unsigned char maxspeed)
{

   USART_Flush();                          // Flush rx buffer

  // Reset crc

  crc = 0;

  // Calculate crc for data

  crc16(lr);
  crc16(maxspeed);

 // Send command

   USART_Transmit(lr);
  USART_Receive_time_out();

  // Send data

  USART_Transmit(maxspeed);
  USART_Receive_time_out();

  // Send crc16

  USART_Transmit((crc & 0xFF00) >> 8);
  USART_Receive_time_out();
  USART_Transmit((crc & 0xFF));
  USART_Receive_time_out();

  // Get acc  (10101010=OK)

  if (USART_Receive_time_out() == CM_OK)
    return CM_OK;
  else
    return CM_FAILED;
}


long int turn(int r, float v)
{

   // Calculates no of puleses for the turn in v deg. (not calibrated!)
   // Radius = 0. Other radius than 0 is not implemented yet!

  float dist;
   long int p;
  dist = 0.00293705911111*v;
  p = dist / 5.753632903870394e-006;
  return p;
}



int main( void )
{
   long int pr;
   long int pl;
   unsigned char maxspeed;
   int i;
   char rx[14];

   USART_Init(9600);

   // Activate motor controler

   while (USART_Receive() != '?');                  // Wait for activation ready signal
   USART_Transmit('A');                             // Activate!
   delay_1s();                                      // Wait for controller to activate
   USART_Flush();                                   // Flush rx buffer

   __enable_interrupt();                            // Enable interrupts
```

```
   // This program makes the robot to go in a box 2 by 2 meters.

   pr=0;pl=0;

   // Set maxspeed 0x00 to 0x30

   maxspeed = 0x05;

   while (send_max_speed(CM_BOTH_SET_MAX_SPEED,maxspeed) == CM_FAILED);

   while (1==1) {
      pr+= 0x00054DD6;              // pulses for 2 meters
      pl+= 0x00054DD6;              // pulses for 2 meters

      // Forward 2m

      while (send_pos(CM_RIGHT_LOAD_POS,pr) == CM_FAILED);
      while (send_pos(CM_LEFT_LOAD_POS, pl) == CM_FAILED);

      USART_Transmit(CM_BOTH_START_MOTION);
      USART_Receive();

      for (i = 0 ; i < 14 ; i++)                          // Receive trajectory complete reply
               rx[i] = USART_Receive_time_out();

   // Turn 90 deg

      pr+=turn(0,90);                                     // Radius not implemented!
      pl-=turn(0,90);

      while (send_pos(CM_RIGHT_LOAD_POS,pr) == CM_FAILED);
      while (send_pos(CM_LEFT_LOAD_POS, pl) == CM_FAILED);

    USART_Transmit(CM_BOTH_START_MOTION);
      USART_Receive();

      for (i = 0 ; i < 14 ; i++)                          // Receive trajectory complete reply
               rx[i] = USART_Receive_time_out();

      // Set maxspeed for demo

      switch(maxspeed) {
         case 0x05:
                  maxspeed = 0x10;
                  break;
         case 0x10:
                  maxspeed = 0x20;
                  break;
         case 0x20:
                  maxspeed = 0x30;
                  break;
         case 0x30:
                  maxspeed = 0x05;
                  break;
      }

      while (send_max_speed(CM_BOTH_SET_MAX_SPEED,maxspeed) == CM_FAILED);

   }
}
```

usart.c

```
    USART functions (usart.c) for Atmega16 for WINAVR

    Version 0.1ß        2007-06-18

    Copyright (C) Andreas Kingbäck  (andki@itn.liu.se)

    This program is free software; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by the
    Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with this program;
    if not, write to the Free Software Foundation, Inc.,
    59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

*/

#pragma language=extended

#ifndef ENABLE_BIT_DEFINITIONS
#define ENABLE_BIT_DEFINITIONS
// Enable the bit definitions in the iom128.h file
#endif

#include <ioavr.h>
#include <stdio.h>
#include <intrinsics.h>
#include "usart.h"

void USART_Init(double baud)
{
   /* Set baud rate */
   double bbaud;
   bbaud = (8000000/(8*baud))-1;
   UBRRH = (unsigned char)((unsigned int)bbaud >> 8);
   UBRRL = (unsigned char)bbaud & 0xFF;
   UCSRB = (1<<RXEN)|(1<<TXEN);
   UCSRA |= 2;                                  // U2X
}

void USART_Transmit_Str(const char *str)
{
   /* Wait for empty transmit buffer */
   int i=0;
   while (str[i] != 0)
   {
      while ( !( UCSRA & (1<<UDRE)) );
      UDR = str[i];                             // Put data into buffer, sends the data
      i++;
   }
}

void USART_Transmit( unsigned char data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSRA & (1<<UDRE)) );
   /* Put data into buffer, sends the data */
   UDR = data;
}
```

```
int USART_Receive_time_out( void )
{
   int timeout = 10000;                  // time out value
   /* Wait for data to be received */
   int data;
   while (((UCSRA & 0x80) == 0) && (timeout > 0)) {
      __delay_cycles(8000);
      timeout --;
   }
   if ((UCSRA & 0x80) == 0)
      data = -1;                         // No data availible
   else
      data=UDR;
   return data;
}

unsigned char USART_Receive( void )
{
   /* Wait for data to be received */
   unsigned char data;
   while ((UCSRA & 0x80) == 0);
   data=UDR;
   /* Get and return received data from buffer */
   return data;
}

void USART_Flush( void )
{
   unsigned char dummy;
   while ( UCSRA & (1<<RXC) ) dummy = UDR;
}

unsigned char USART_ReadUCSRC( void )
{
   unsigned char ucsrc;
   /* Read UCSRC */
   ucsrc = UBRRH;
   ucsrc = UCSRC;
   return ucsrc;
}

INTERRUPT_USART_RXC()// Interrupt driven USART Receive
{
   unsigned char slask;
   __disable_interrupt();
   slask = UCSRA;               // Must read UCSRA with dummy read :-/
   slask = UDR;
   __enable_interrupt();
}
```

control.h

```
/*

    Parser functions (control.h) for Atmega16 and WINAVR

    Version 0.2         2007-10-29

    Copyright (C) Andreas Kingbäck 2005 (andki@itn.liu.se)

    This program is free software; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by the
    Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with this program;
    if not, write to the Free Software Foundation, Inc.,
    59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

*/


#ifndef _PARSER_H_
#define _PARSER_H_

#include <ioavr.h>

// Motor Control commands

#define CM_LEFT_LOAD_POS                20
#define CM_RIGHT_LOAD_POS               21
#define CM_BOTH_LOAD_POS                22
#define CM_LEFT_START_MOTION            23
#define CM_RIGHT_START_MOTION           24
#define CM_BOTH_START_MOTION            25
#define CM_LEFT_SET_MAX_SPEED           26
#define CM_RIGHT_SET_MAX_SPEED          27
#define CM_BOTH_SET_MAX_SPEED           28

// Return codes

#define CM_OK                           254
#define CM_FAILED                       255

// Status codes

#define ROBOT_DISABLED          0
#define ROBOT_ACTIVATED         1

// Status codes

#define true  1
#define false 0

#endif         /* _PARSER_H_ */
```

usart.h

```
    Usart functions (usart.h) for Atmega16 for IAR

    Version 0.1ß        2007-06-21

    Copyright (C) Andreas Kingbäck  (andki@itn.liu.se)

   This program is free software; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by the
    Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

   This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with this program;
    if not, write to the Free Software Foundation, Inc.,
    59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

*/

#ifndef ENABLE_BIT_DEFINITIONS
#define ENABLE_BIT_DEFINITIONS
// Enable the bit definitions in the iom128.h file
#endif

#ifndef _USART_H_
#define _USART_H_ 1

#include <ioavr.h>
#include <intrinsics.h>

#define true 1
#define false 0

unsigned char USART_Receive( void );
int USART_Receive_time_out( void );

#define INTERRUPT_USART_RXC()  _Pragma("vector=USART_RXC_vect")   __interrupt void
signal_func_USART1_RXC(void)
#define INTERRUPT_USART_DATA() _Pragma("vector=USART_UDRE_vect")  __interrupt void
signal_func_USART1_UDRE_vect(void)


void USART_Init(double baud);
void USART_Transmit( unsigned char data );
void USART_Transmit_Str(const char *str );
void USART_Flush( void );

#endif
/* _USARTUSB_H_ */
```