

# RCX Byte Code

Programmering och testning mellan klient och server mha en  
RCX-enhet genom användning av protokollet RCX Byte Code.

Kurs: Datorkommunikation och nät (DT2017-0200/DT2022-0222)

*Härmed försäkrar jag/vi att jag/vi utan att ha erhållit eller lämnat någon hjälp utfört detta arbete.*

Datum: 2015-10-22

Underskrift:

Özgun Mirtchev

Namn: Özgun Mirtchev  
Personnr: 920321-2379  
E-post: *ozzieee@gmail.com*  
Program: Dataingenjörsprogrammet

|                       |
|-----------------------|
| Lärarens anteckningar |
|-----------------------|

## Table of Contents

|   |   |
|---|---|
| Bakgrund .....                                  | 2 |
| Uppgift A: .....                                | 3 |
| 1.    Alive - operationskoder.....              | 3 |
| 2.    Transmitter range - operationskoder ..... | 4 |
| Uppgift B.....                                  | 5 |
| 1 .....   | 5 |
| 2.....  | 5 |
| 3.....  | 6 |
| 4.....  | 6 |

## Bakgrund

Syftet med denna laboration var att lära sig hur en klient och en server kommunicerar med varandra.

För att kommunicera med servern användes programmet `lego.hex` som laddades ned i en MCU som var kopplat till ett RS232 som i sin tur var kopplat till ett torn, som överförde data via IR till servern, som var på RCX-enheten. Enheten i fråga kan man se på en bild längst ned på denna sida. Klienten var även inställt på asynkron kommunikation.



*Figure 1 – En RCX-enhet kopplat med två motorer, tillsammans med ett IR-torn*

Koderna finns i [bilagor](#)-sektionen.

Resultat

## Uppgift A:

### 1. Alive - operationskoder

Del a)

| # | Header | Operation Code and Data             | Checksum                    |
|---|--------|-------------------------------------|-----------------------------|
| 1 | 55FF00 | 10EF ( $O\bar{O}$ , inga data)      | $C = 10 + EF (C + \bar{C})$ |
| 2 | 55FF00 | EF10 ( $S = \bar{O}, \bar{S} = O$ ) | $C = EF + 10 (C + \bar{C})$ |
| 3 | 55FF00 | 18E7 ( $0 + 8, \bar{O} + 8$ )       | 18E7                        |
| 4 | 55FF00 | E718 ( $S = \bar{O} + 8, 0 + 8$ )   | E718                        |

Del b)

Enligt bilaga för Operationskoder används koderna **10/18**.

$$O = 10;$$

$$\bar{O} = FF - 10 = EF$$

(ref Alive->Request, bilaga Operationskoder)

Del c)

Koderna som används för svar är **E7/EF**.

(ref Alive->Reply, se bilaga Operationskoder)

Del d)

Adressens hex-tal minus O. 0x55FF00 ->

$$\bar{O} = FF - 10 = 255 - 16_{dec} = 239_{dec} = 0xEF$$

(ref Labb-PM->sid 7)

## 2. Transmitter range - operationskoder

Del a)

Kommandonamn: **Set transmitter range**

Koder:

- Förfrågan: **31/39** (Request/Command)
- Svar: **c6/cc** (Reply)

Argument: **Short distance = 0; Long distance = 1.**

(ref Set transmitter range->(Request, Reply), se bilaga Operationskoder)

Del b)

Checksumman:  $C = O + D1 + \dots = 31 + 1(\text{långt avstånd}) = 32$

Begäran:

|  |                    |                                   |
|--|--------------------|-----------------------------------|
| <b>55FF00</b>                              | <b>31 CE 01 FE</b> | <b>32 EA (C, C<sup>-</sup> 1)</b> |
| (O)(O <sup>-</sup> 1)(D)(D <sup>-</sup> 1) |                    |                                   |

Svar:

|               |              |              |
|---------------|--------------|--------------|
| <b>55FF00</b> | <b>CE 31</b> | <b>CE 31</b> |
|---------------|--------------|--------------|

## Uppgift B

### 1

Sendfram() kompletterades med koden:

```
/* Övning B:1 - Plats för att skicka header och command (operationskod)
*/
/* Sänd header */
USART_SendByte(0x55);
USART_SendByte(0xFF);
USART_SendByte(0x00);

/* Sänd command och dess 1-komplement */
USART_SendByte(command);
USART_SendByte(0xFF - command);

checksum = command;

USART_SendByte(checksum);
USART_SendByte(0xFF - checksum);
```

USART\_SendByte() användes för att skicka header-koderna därefter skickades checksumman och dess 1-komplement.

### 2

GetBatteryPower() kompletterades med följande kod:

```
/* Övning B:2 - Plats för att skicka header, command (operationskod)
och checksum */
/*Sänd header */

USART_SendByte(0x55);
USART_SendByte(0xFF);
USART_SendByte(0x00);

/* Sänd command och dess 1-komplement */

USART_SendByte(command);
USART_SendByte(0xFF - command);

/* Sänd checksum och dess 1-komplement */

USART_SendByte(command);
USART_SendByte(0xFF - command);
```

I stort sett samma som i uppgift 1. Det enda som krävdes var att skicka header och checksumman.

3

Efter att kod hade lagts till för att införa ett nytt ljud så kunde man se resultatet på LCD-skärmen. Vid testkörning hördes ett nytt ljud jämfört med de förinstallerade alternativen. Koden finns i [Bilaga 1](#).

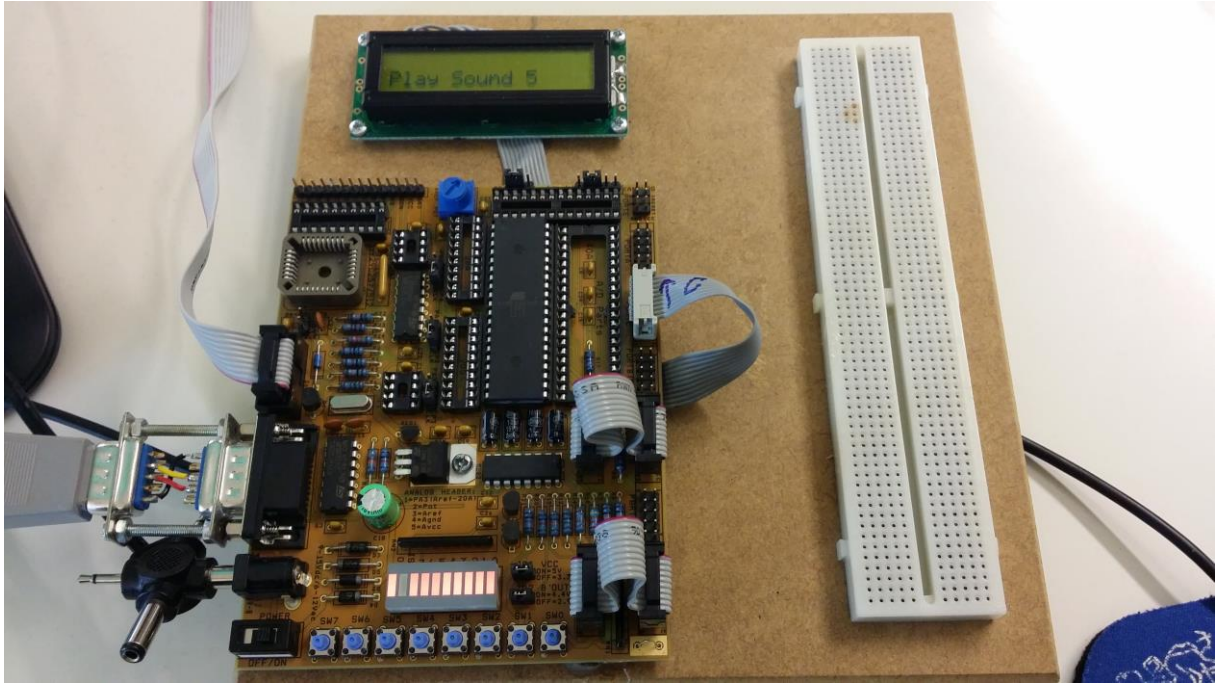


Figure 2 – Bild på testkörning av ny Sound-alternativ

4

Ännu ett alternativ lades till i menyn, denna gång en ny motor med snabbare takt än den förinstallerade. Koden finns i [bilaga 2](#). Dessvärre finns bara en bild på det andra alternativet för motorn. ”Motor B off”

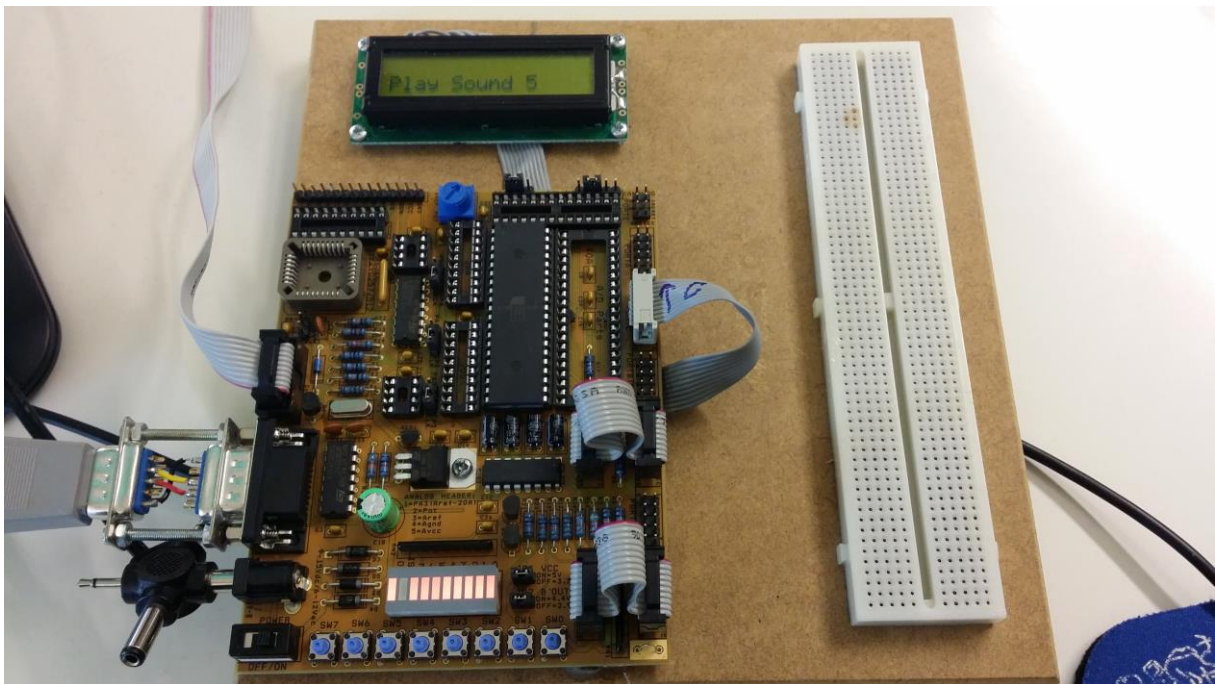


Figure 3 – Bild på testkörning av ny motor B – alternativ

## Bilaga/Bilagor

### 1.

Nytt ljud (playSound 3) lades till med identisk kod som Play Sound 2 har, enda skillnad är utbytt nummer från 2 till 5 :

/\* Övning B:3 - Plats för att lägga till ljud \*/

```
case 6:
    LCD_Clean(LINE2);
    LCD_StrOut("Play Sound 5");
    while(bit_is_set(PIND,7))
    {
        if(bit_is_clear(PIND,6))
        {
            b = (unsigned char*) "\x05"; // Ljudtyp nr 5
            SendFrame(0x51, b);          // PLAY SOUND
        }
    } break;
```



## 2.

Ny motor B kod: I stort sett samma kod som Motor A.

```
/* Övning B:4 - Plats för att lägga till motor */
case 7:
    LCD_Clean(LINE2);
    LCD_StrOut("Set motor B pow");
    while(bit_is_set(PIND,7))
    {
        if(bit_is_clear(PIND,6))
        {
            b = (unsigned char*) "\x02\x03\x07";
            // Motor B, Level 3, Power 7
            SendFrame(0x13, b);
            // SET MOTOR POWER
            LCD_Clean(LINE2);
            LCD_StrOut("Motor B is set");
            MenuDelays(4);
        }
    } break;

case 8:
    LCD_Clean(LINE2);
    LCD_StrOut("Motor B on");
    while(bit_is_set(PIND,7))
    {
        if(bit_is_clear(PIND,6))
        {
            b = (unsigned char*) "\x81"; // Modify/turn on
            // Motor B
            SendFrame(0x21, b); // SET MOTOR ON
            MenuDelays(4);
        }
    } break;

case 9:
    LCD_Clean(LINE2);
    LCD_StrOut("Motor B off");
    while(bit_is_set(PIND,7))
    {
        if(bit_is_clear(PIND,6))
        {
            b = (unsigned char*) "\x41"; // Modify/turn off
            // Motor B
            SendFrame(0x21, b); // SET MOTOR ON
            MenuDelays(4);
        }
    } break
```