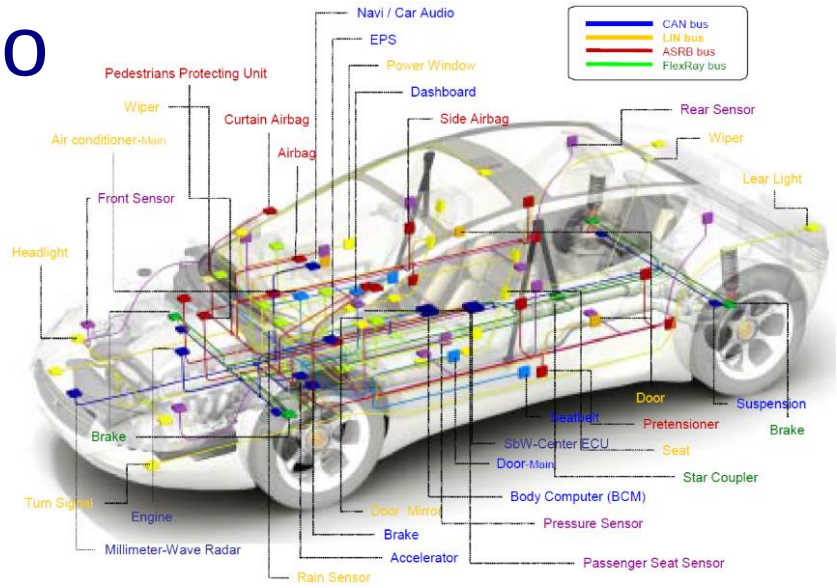
Vhodno izhodne naprave

Laboratorijska vaja 11 - LV 4 CANBUS

Laboratorijska vaja 11 - LV4

- 11.0: CANBUS osvežitev
- 11.1 Opis primera : Cybrotech CANBUS sistem
- 11.2: Krmiljenje Cybrotech IEX-2 modulov
- 11.3: CANBUS meritve
- 11.4: STM32H7 – osnovni IEX-2 modul

CANBUS vodilo



CANBUS (ISO-11898-2):

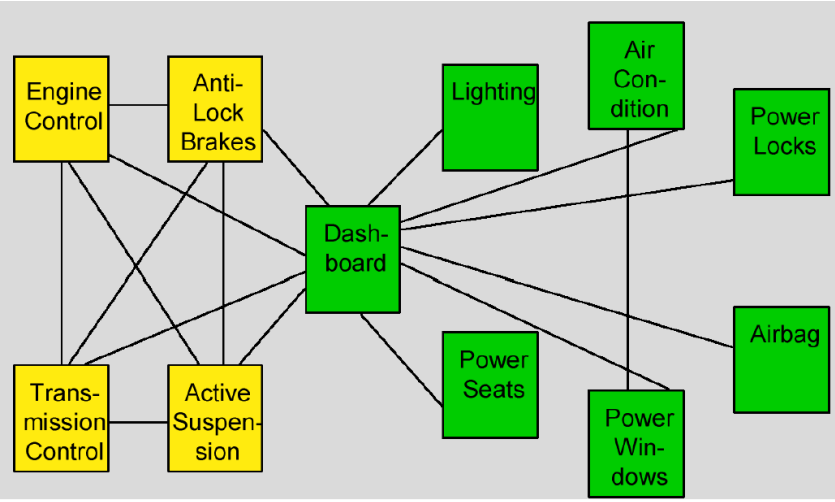
- Zgodovina
- Področja uporabe
  - Avtomobilska industrija
  - Industrijska avtomatika, pametne stavbe
- Pregled protokola, arbitraže, fizičnega nivoja
- Praktični primer: Pametna hiša, IEX-2 protokol

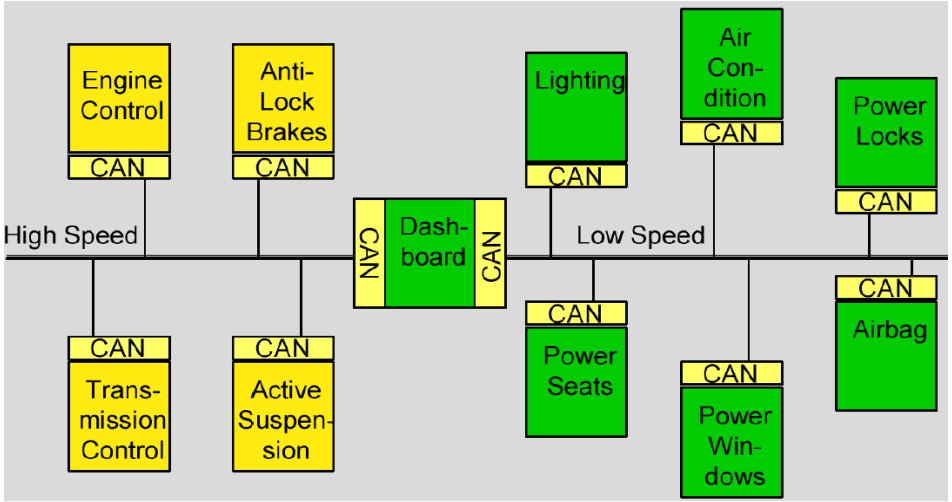
Lab. Vaja :

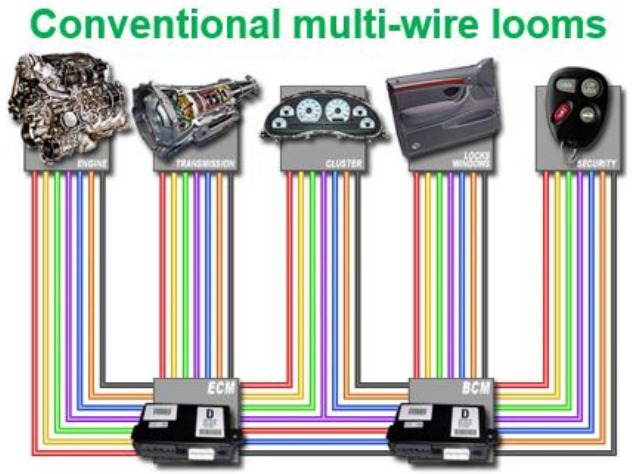
- Gradniki in shema testnega sistema
- Programiranje sistema
- Meritve signalov na povezavah

Zakaj vodilo ?

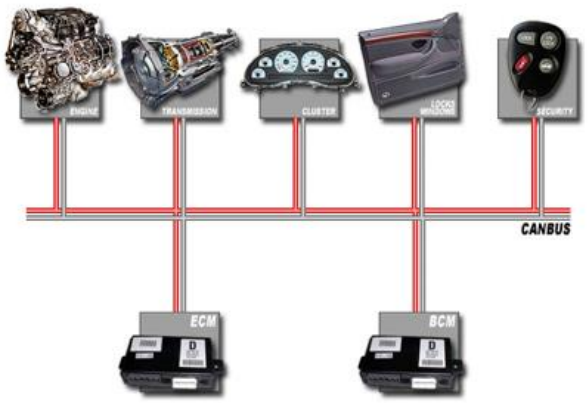
Primera povezav brez (levo) in z (desno) CANbus vodilom







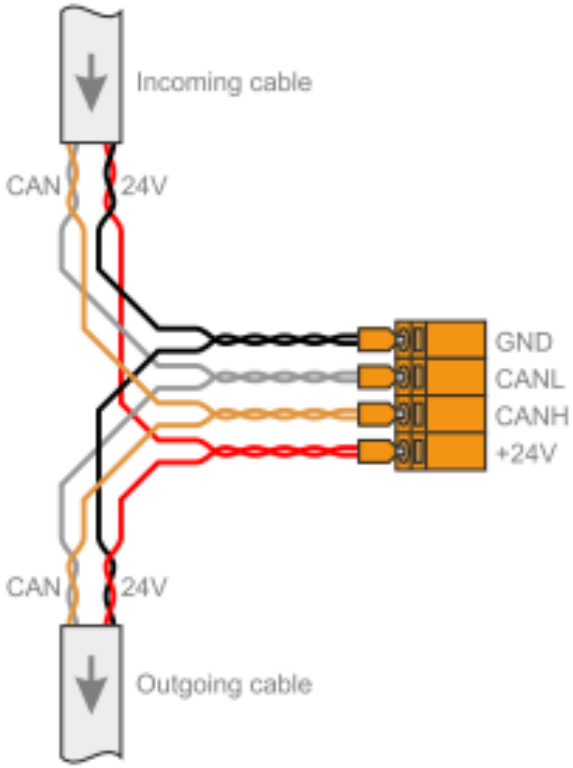
VS.



http://canbuskit.com/what.php

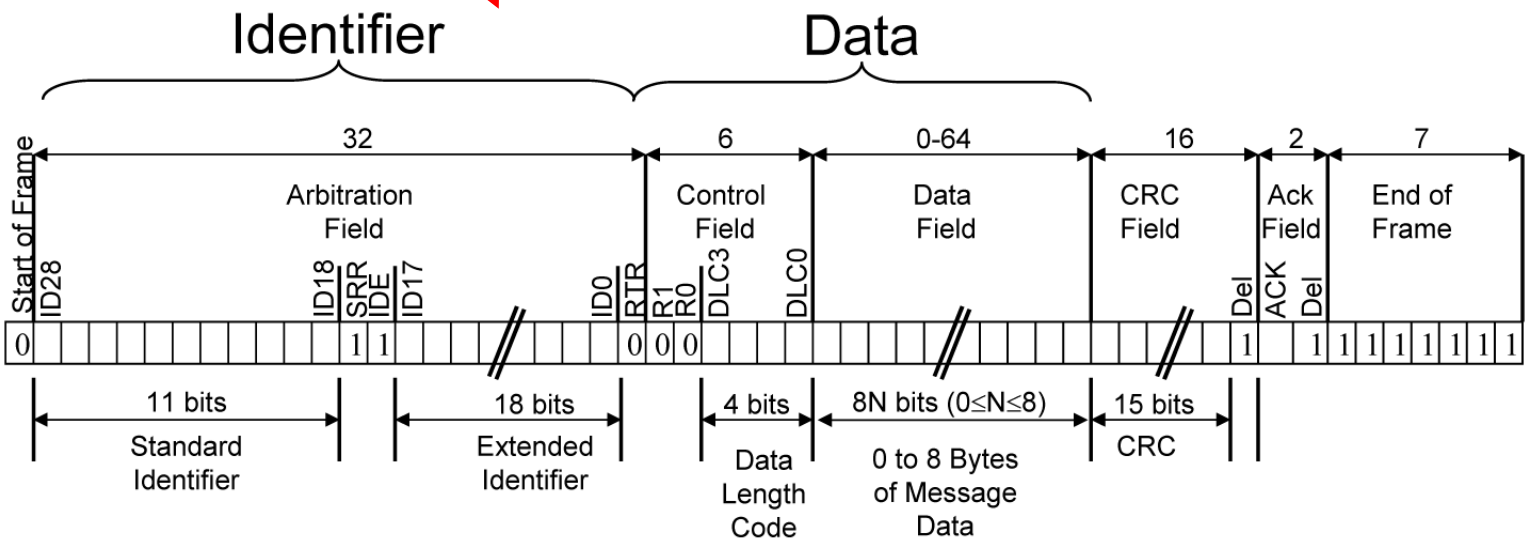
CANbus na kratko

- CANbus – **C**ontroller **A**rea **N**etwork bus
- CAN (Controller Area Network) je serijsko vodilo za komunikacijo med vgrajenimi mikrokrmilniki
- CAN bus na kratko :
  - serijsko vodilo
  - dve žici (CAN_H,CAN_L) + napajanje,
  - diferencialni prenos signala
    - odpornost na šum.
  - max 1Mbit/s, 40m,
  - sporočila do 8 bajtov (latenca)
- CAN-FD standard, ISO 11898-2:2016
  - 2Mbps, 5Mbps



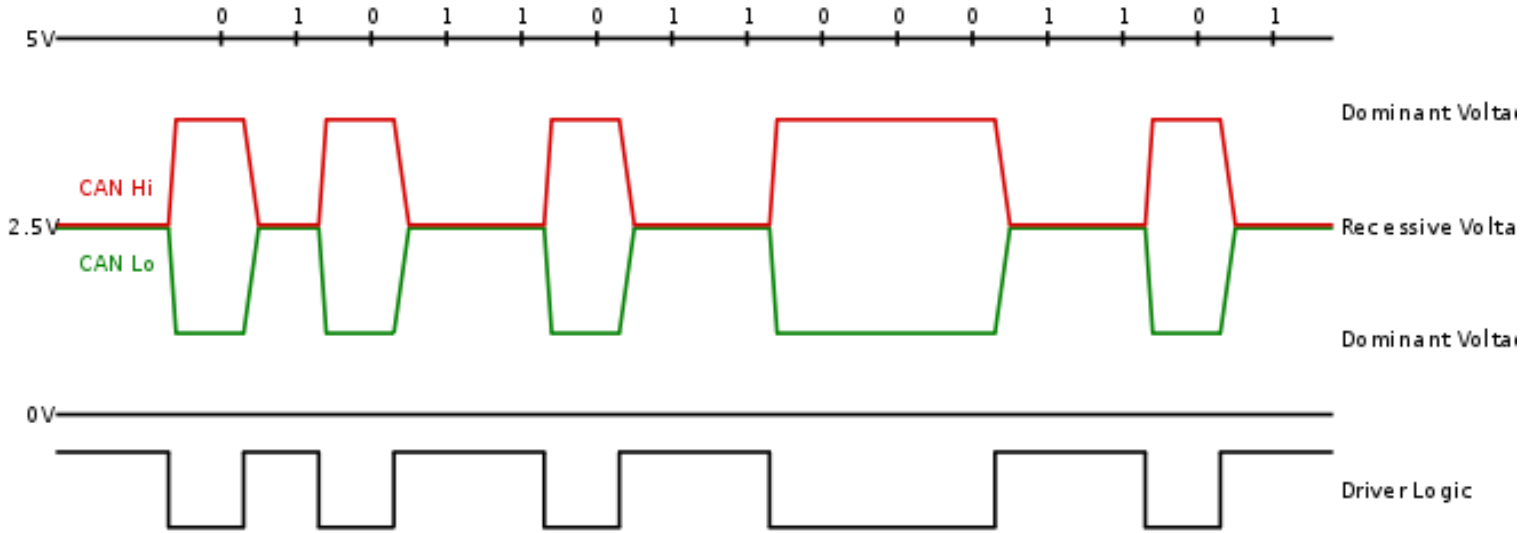
CANbus na kratko

- Prenos podatkov
  - Format okvirja
  - Protokol – sporočilno naravnan
  - Detekcija napake
    - Nivo Bitov (branje, „bit stuffing“)
    - Nivo sporočila (CRC, okvir, ACK napake)

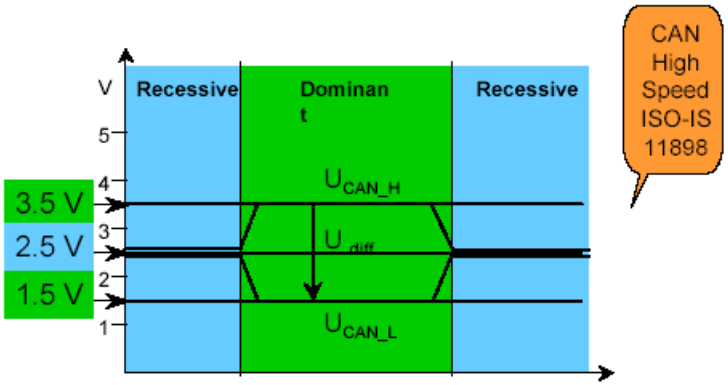


CANbus napetostni nivoji ISO-IS 11898

- Diferencialni prenos običajno na parici - Non-Return To Zero (NRZ) in bit-stuffing.
- Wired – AND povezava: vozlišče z logično 0 prevlada
  - 0 .. „dominant“, 1.. „recessive“)



CANbus napetostni nivoji ISO-IS 11898



•Recesivni bit „1“:

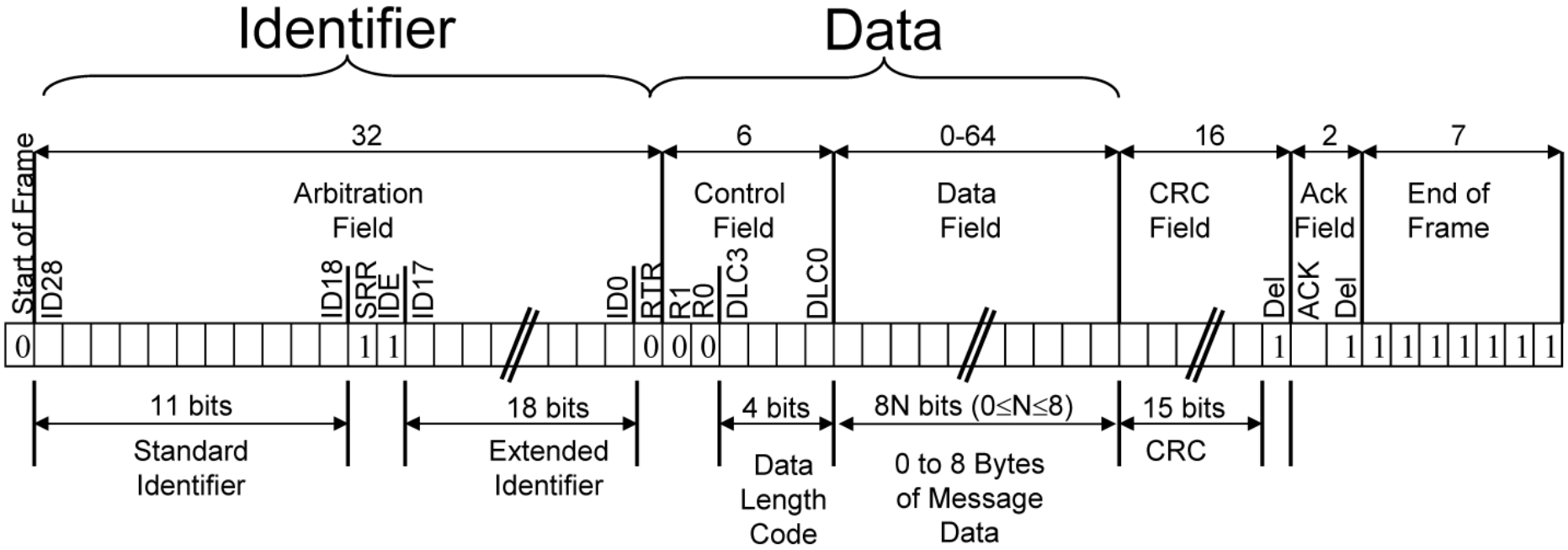
- obe liniji na približno 2.5V
- diferencialna napetost CAN_H in CAN_L ≈ 0 V

•Dominantni bit „0“:

- CAN_H na pribl. 3.5 V in CAN_L pribl. 1.5 V
- diferencialna napetost CAN_H in CAN_L ≈ 2 V

Format sporočila

- Vsako sporočilo ima ID, podatke in dodatke
- ID - 11 ali 29 bitov
- Data - do 8 bajtov
- Dodatki - start (SOF), CRC, ACK, end (EOF)



Format sporočila

CAN vs. RS-485: Why CAN Is on the Move

By Robert Gee, Executive Business Manager, Core Products Group, Maxim Integrated

- Recesivni bit „1“:
  - obe liniji na približno 2.5V
- Dominantni bit „0“:
  - CAN_H na pribl. 3.5 V in CAN_L pribl. 1.5 V

Field Name	Bit Length	Description
SOF	1	Start of frame
Identifier (green)	11/29; 12/32	Represents the message priority (11 or 29 bits for standard CAN and extended CAN; 12 or 32 bits for CAN-FD)
RTR (blue)	1	Remote transmission request
IDE	1	Identifier extension bit
r0	1	Reserved bit for future protocol expansion
DLC (yellow)	4/8/9	Code for number of data bytes (4-bit for standard CAN; 8 or 9 bits for CAN-FD)
Data Field (red)	0–64 (0–8 bytes); 0–512 (0–64 bytes)	Data to be transmitted (0–8 bytes for standard CAN; 0–64 bytes for CAN-FD)
CRC	15	Cyclic redundancy check
CRC Delimiter	1	Assigned recessive (1)
ACK slot	1	Dominant bit if error-free message; recessive to discard errant message
ACK Delimiter	1	Acknowledgement delimiter
EOF	7	End of frame

Table 1. CAN Message Data-Frame Format

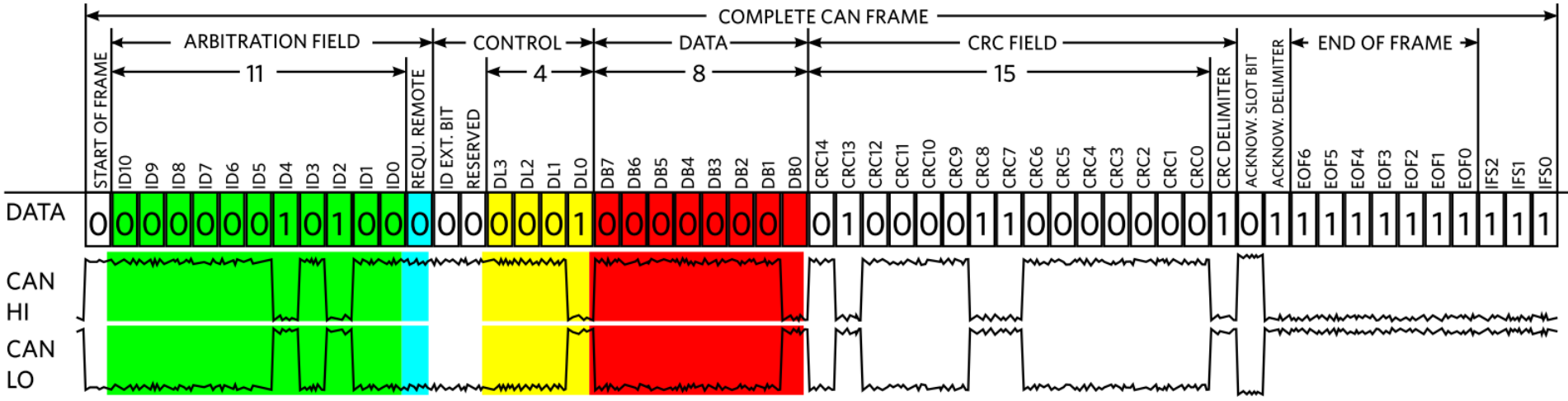


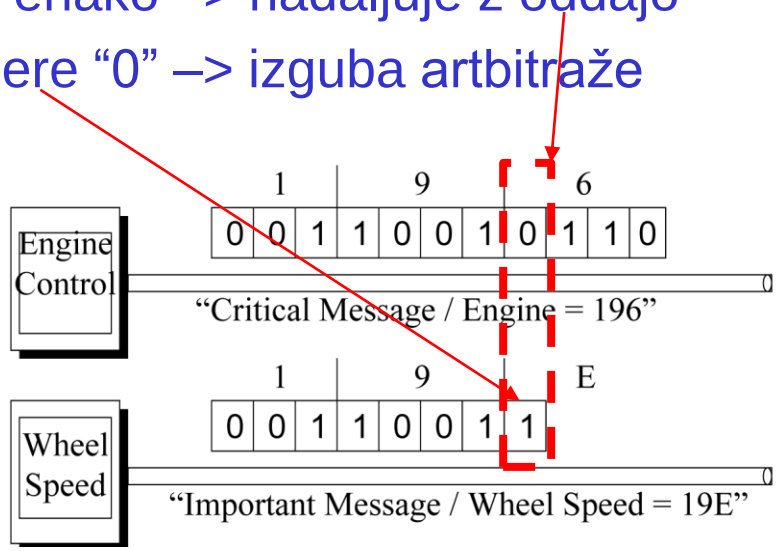
Figure 4. CAN Message Data-Frame Format

Arbitraža (Non-Destructive Arbitration)

- Pomembnos sporočila je določena z IDjem

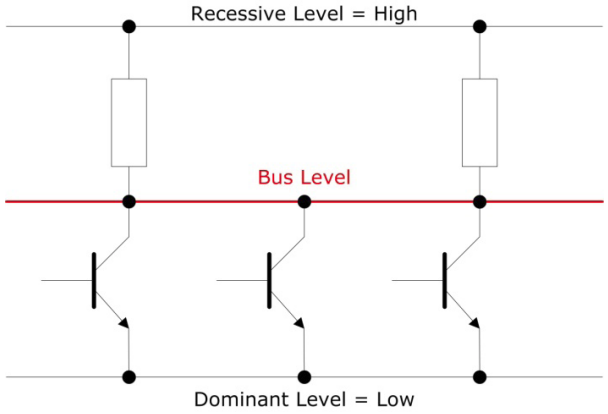
  Nižja vrednost = Višja pomembnost

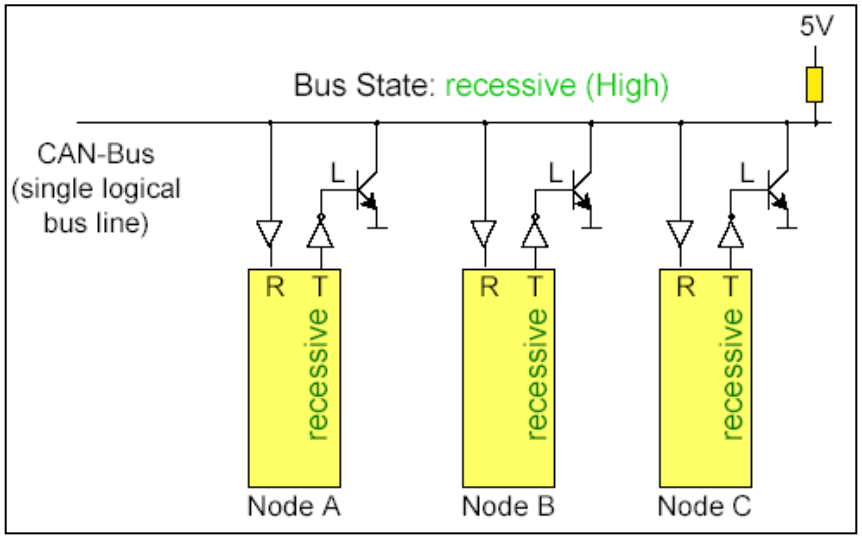
- Naprava odda in hkrati bere
  - “0” na vodilu prevlada “1” na vodilu
- Naprava:
  - odda in bere enako –> nadaljuje z oddajo
  - odda “1” in bere “0” –> izguba artbitraže

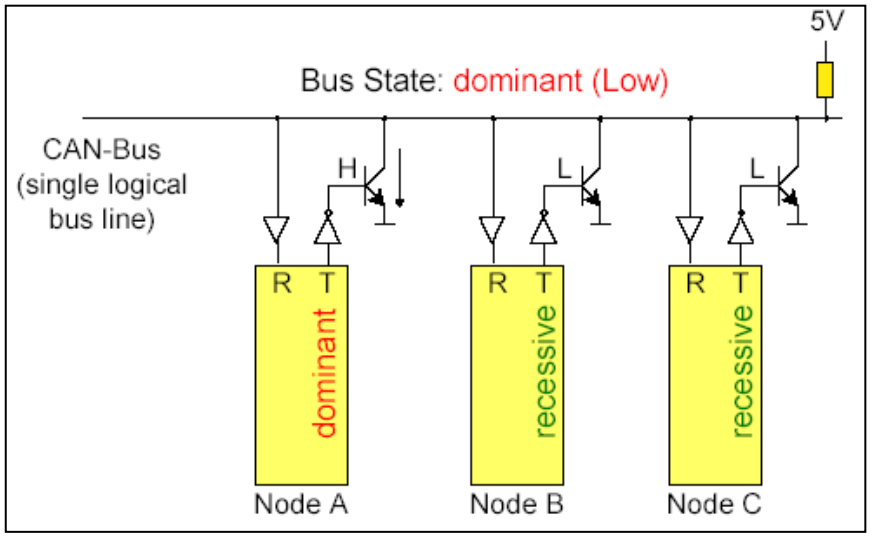


Wired AND (Arbitraža)

Stanje “0” (nizka napetost oz. dominantno stanje) na vodilu prevlada ostala stanja “1” (višja napetost oz. recesivno stanje) na vodilu.







Osciloskop: primer CANbus komunikacije

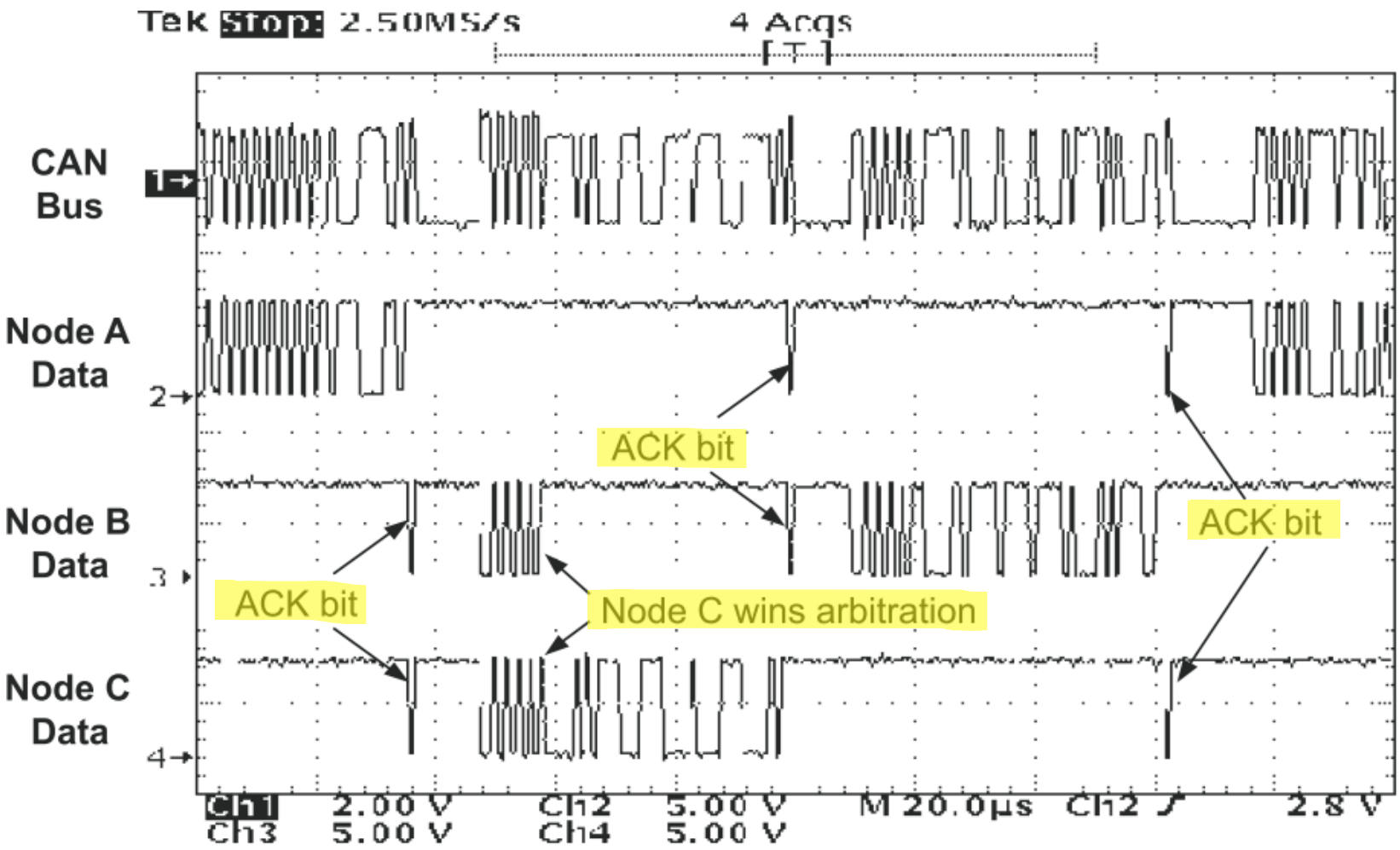
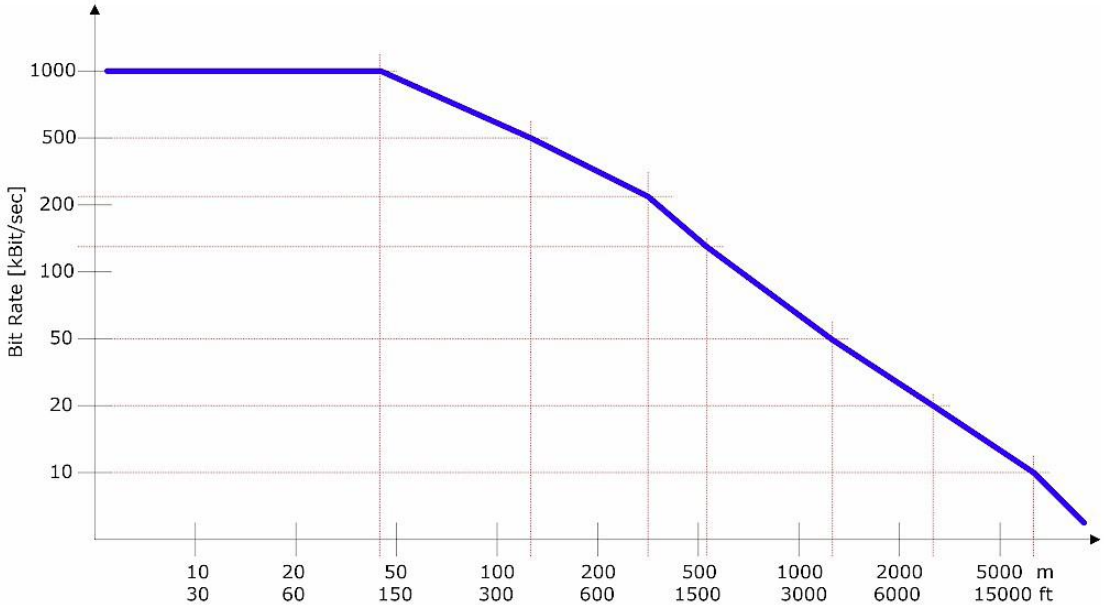


Figure 8. CAN Bus Traffic

Hitrost komunikacije

- Do 1 Mbit/sec.
- Standardne hitrosti: 1 MHz, 500 KHz and 125 KHz
- Max length: do 5000m, odvisno od:
  - hitrosti
  - lastnosti:
    - zaključitve, vrsta kabla, topologije, motenj, …



RS-485 vs CANBUS

Kako razrešiti ?

Podobno/enako:

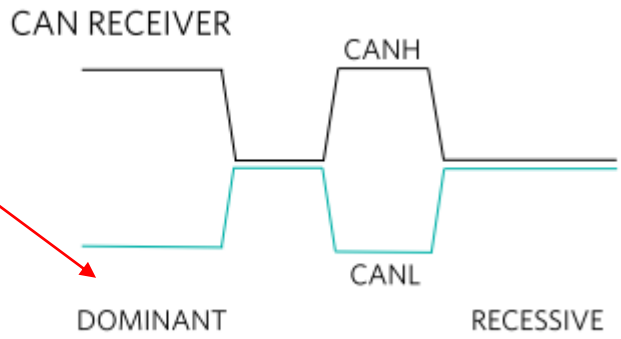
- Diferencialni prenos
- Multi-master
- Zaključitev 120Ω
- Različno

Prednosti CANBUS :

- Multi-master oddajanje
  - CANBUS arbitraža
  - RS485 –konflikt, poraba toka, segrevanje
- Dodatna preverjanja (nivo sporočila)
  - CRC, format sporočila
- Dodatna preverjanja(bitni nivo)
  - Spremljanje stanja linije (poslano/sprejeto)
  - Potrditev (Acknowledge)
  - Bit-stuff (6. bit)

Prednosti RS485 :

- Višja hitrost – do 35Mbit/s
  - Obe stanji sta aktivno vodeni
  - CANBUS (Wired AND) ima recesivno in dominantno stanje



Laboratorijska vaja 11 - LV4

- 11.0: CANBUS osvežitev
- 11.1 Opis primera : Cybrotech CANBUS sistem
- 11.2: Krmiljenje Cybrotech IEX-2 modulov
- 11.3: CANBUS meritve
- 11.4: STM32H7 – osnovni IEX-2 modul

Tinia – prijazen dom
TBS – „Tinia Building Server”

Kratek opis

TBS – „Tinia Building Server”:

Nadzor,upravljanje in vizualizacija delovanja prijaznega doma.

- majhen, varčen, tih (5W)
- povezuje zgradbo in pametno mesto
- informiranje, povratna inf.
  - •pametni telefoni, tablice
  - •splet, soc.omrežja
- programiranje s pravili,vtičniki
- povezava s soc.omrežji
  - •Twitter,FaceBook

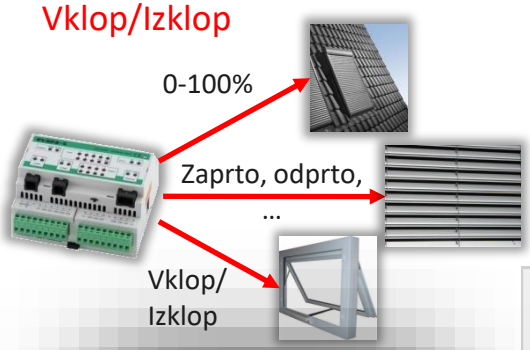
Rolete, Žaluzije, Okna

Pasivno ogrevanje/hlajenje…

- **Rolete**: med 0% - 100%
  (0% odprte, 100% zaprte)

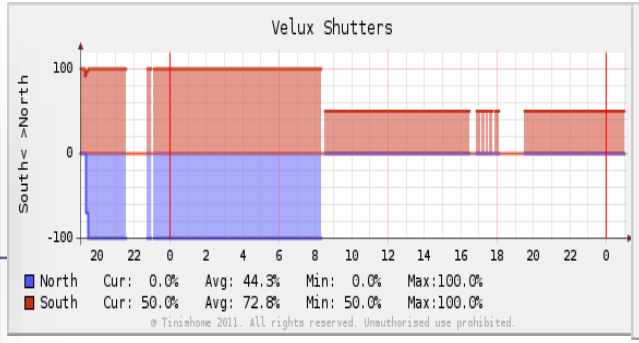
- **Žaluzije** imajo stanja :
  Zaprto(100%), Senčeno(75%), Odprto(50%), Solarno pasivno (25%), Dvignjeno(0%).

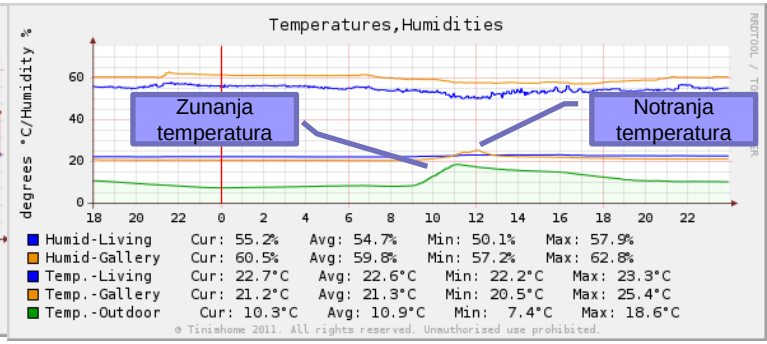
- **Motorizirana** okna:
  Vklop/Izklop



- **Strešna okna z roletami** :
  - severna, običajno:
    - **Odprta v toplem vremenu** za boljšo osvetlitev (poletje)
    - **Zaprta v hladnem vremenu** za ohranjanje toplote (zima)
  - južna, običajno:
    - **Odprta v hladnem, sončnem vremenu** za pasivno ogrevanje (zima, pomlad)
    - **Zaprta v vročem vremenu** proti pregrevanju (poletje)
- **Žaluzije:**
  - **Senčene ali zaprte ob izrazitem sončnem vremenu poleti**
  - **Odprte v "solarni" poziciji ob sončnih dnevih pozimi**
- **Motorizirana okna (s komarniki) :**
  - **Odprta v poletnih nočeh za pasivno ohlajanje**

Primer stanj rolet in temperatur v sončnem zimskem dnevu:



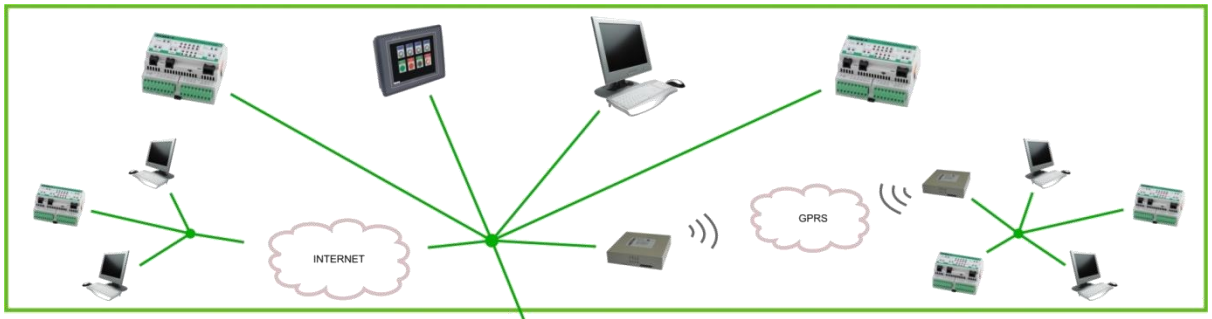


CANbus v praksi

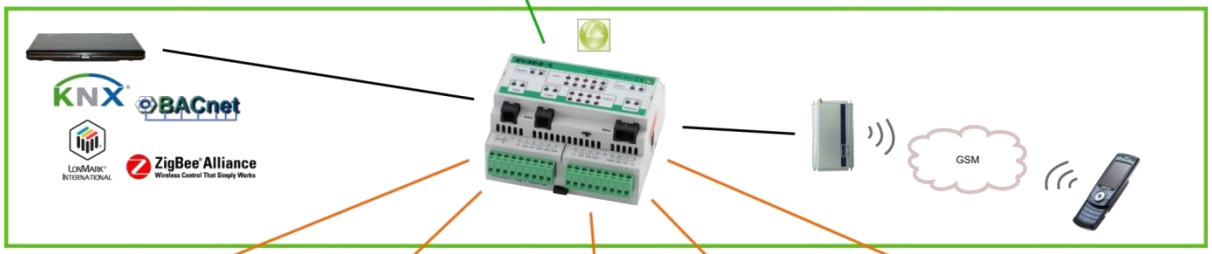
INTEGRA BM SYSTEM

Industrial & Building Automation

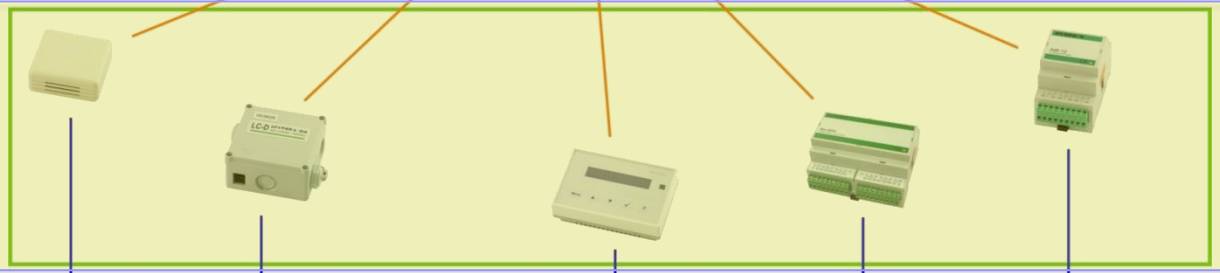
High level network (Ethernet, A-Bus, Modbus)



CyBro controller



Low level network (Canbus)



Dodatki (tipala, daljinci, …)

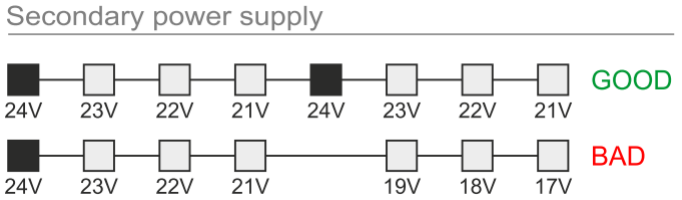
Bus length

Dolžina, hitrost in topologije

Regarding bus length, two points must be considered:

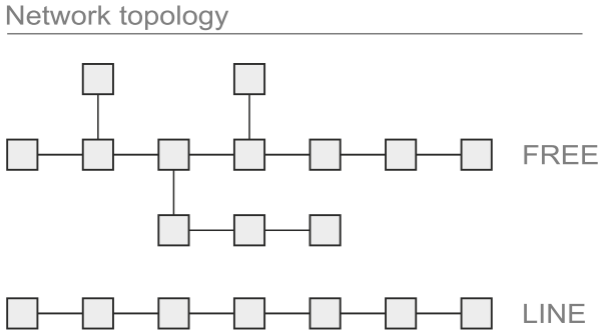
1. Voltage drop

Wire resistance cause voltage drop, which depends of cable length, wire diameter and power consumption. Cable must be selected to ensure each module have at least the minimum specified voltage.

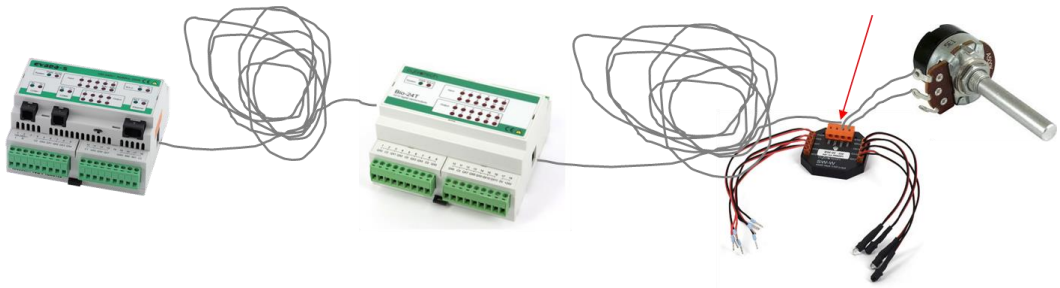


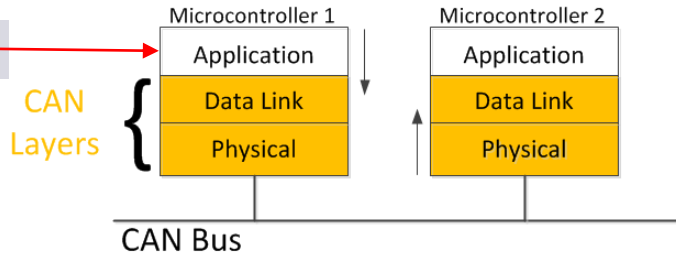
2. Signal delay

Communication speed is limited with propagation time and bus topology. With default 100kbps baudrate, 100m is safe without restrictions. For a longer distance, cable must be connected in a line (without trunks) and properly terminated.



Speed\Topology	FREE	LINE
100kbps	100m	300m
50kbps	200m	500m
20kbps	500m	1000m





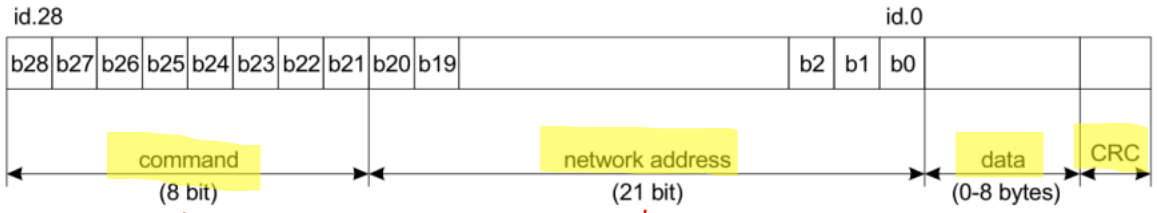
INTEGRA BM SYSTEM

IEX protocol (nadgradnja CANBUS)

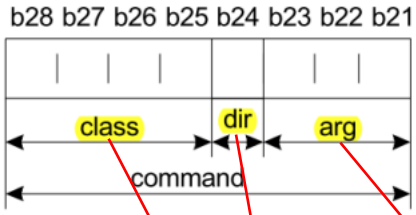
IEX PROTOCOL v2.8 POVZETEK

General

IEX-2 is based on CAN 2.0B. Message format is defined as follows:



Command summary



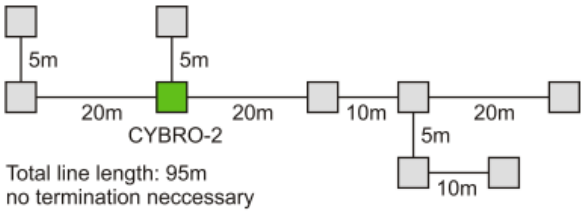
NAD – unikatni naslov IEX modula

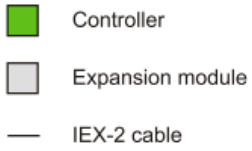
command	class	dir	command	arg	data bytes	description	PCAN view
	0000						
	0001						
	0010						
IX_DATA	0011	1		xxx	data(1..4)	binary inputs	070-07Exxxxxh
QX_DATA		0		xxx	data(1..4)	binary outputs	060-06Exxxxxh
	0100						
	0101						
	0110						
IW_DATA	0111	1		xxx	data(2..8)	analog inputs	0F0-0FExxxxxh
QW_DATA		0		xxx	data(2..8)	analog outputs	0E0-0EExxxxxh
BAUDSYNC	1111	1		111	-	autobaud sync msg	1FFFFFFFh

INTEGRA BM SYSTEM

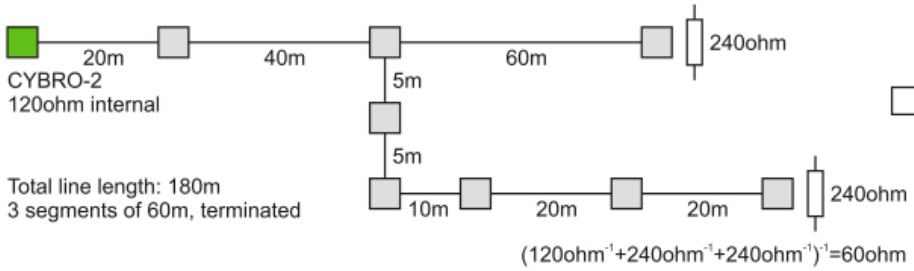
Cabling topology & Termination

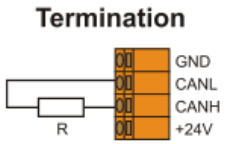
1) Total IEX-2 bus length <100m



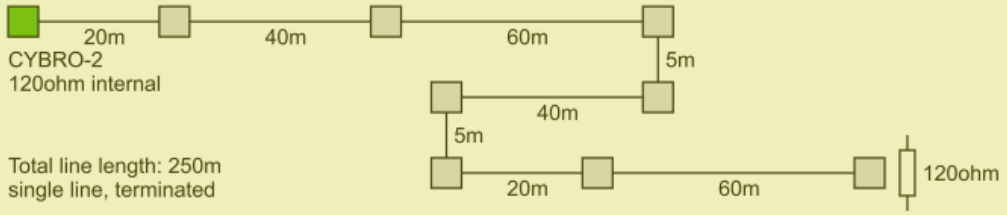


2) 100m< Total IEX-2 bus length <200m

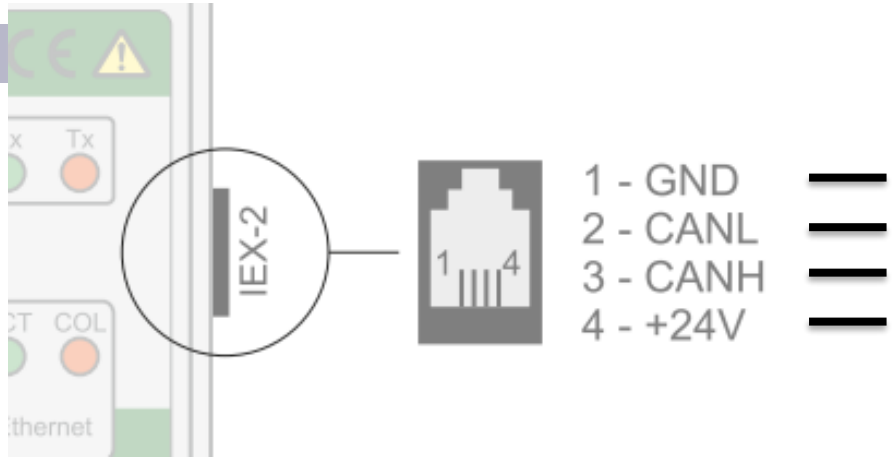




3) 200m < Total IEX-2 bus length <300m



CENTRALNI KRMILNIK CYBRO-2



Controller

Ethernet port

2 x RS-232 port

CAN interface

Digital and analog I/O

Communication and status LED signalization

Core

Modular

Block

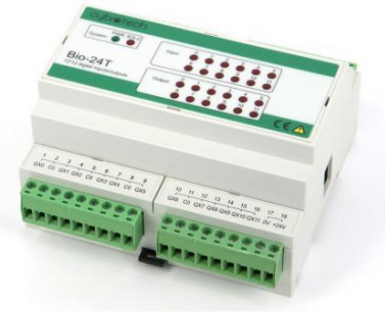
Retentive and permanent EEPROM memory

Removable connectors

230V AC or 24V DC

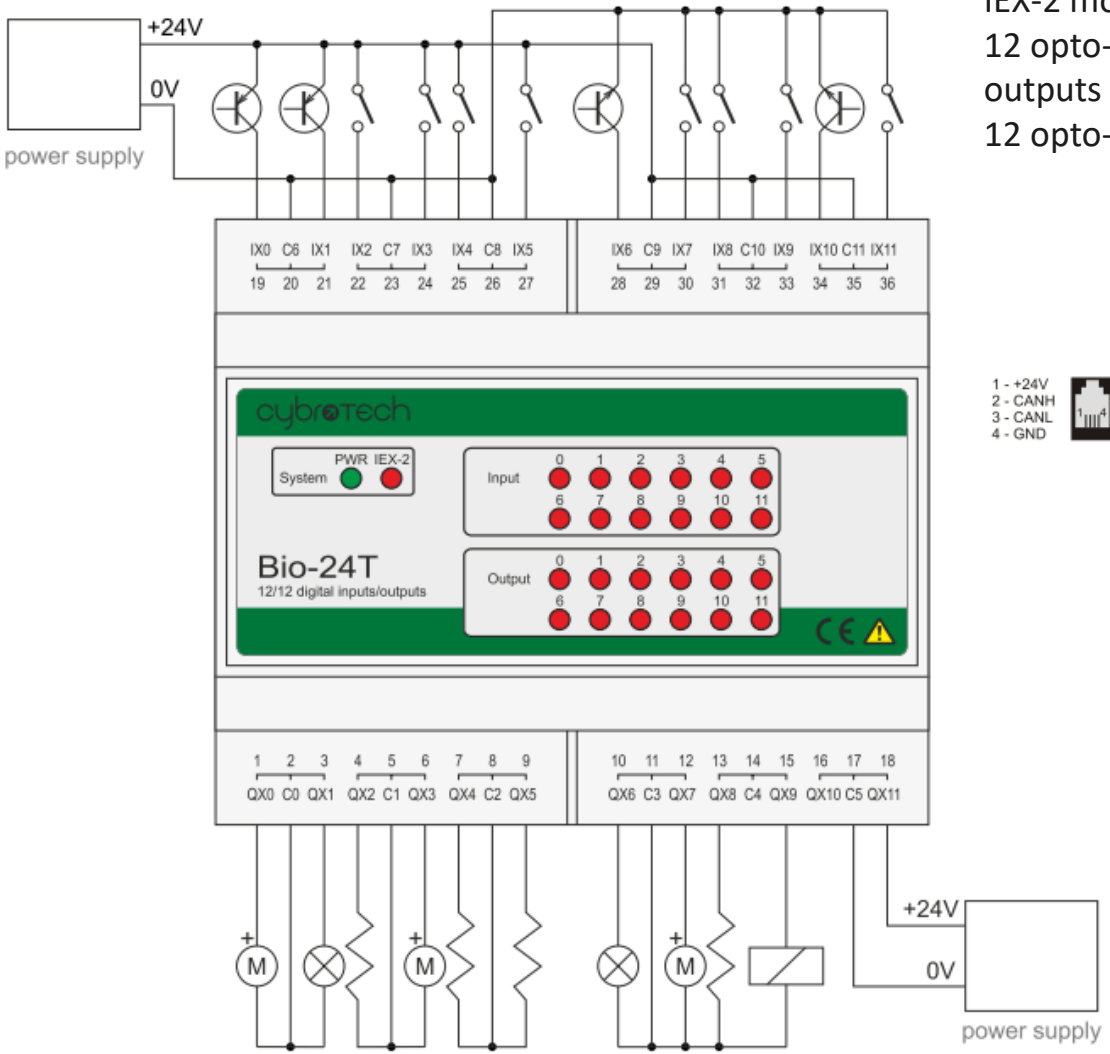
IEX MODULE Bio-24T

Bio-24T



IEX-2 module
12 opto-isolated PNP transistor outputs 1A
12 opto-coupler inputs 24V

Wiring diagram



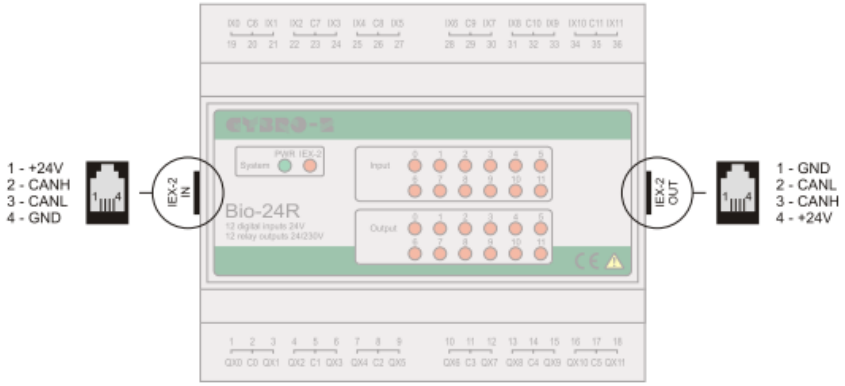
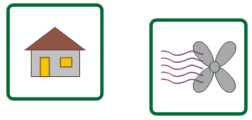


Figure 3: IEX-2 input and output ports.

IEX MODULE FC



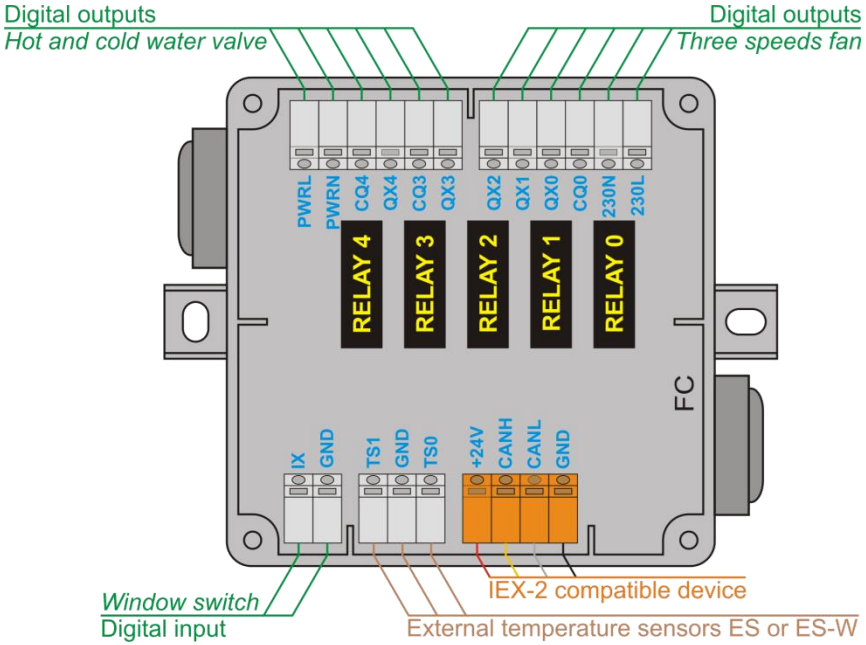
FC

fan coil module

SPECIFICATIONS:
1 x digital input
5 x relay output
2 x input temperature measurement
24V DC power supply
consumption: 110mA

MECHANIC:
field mountable

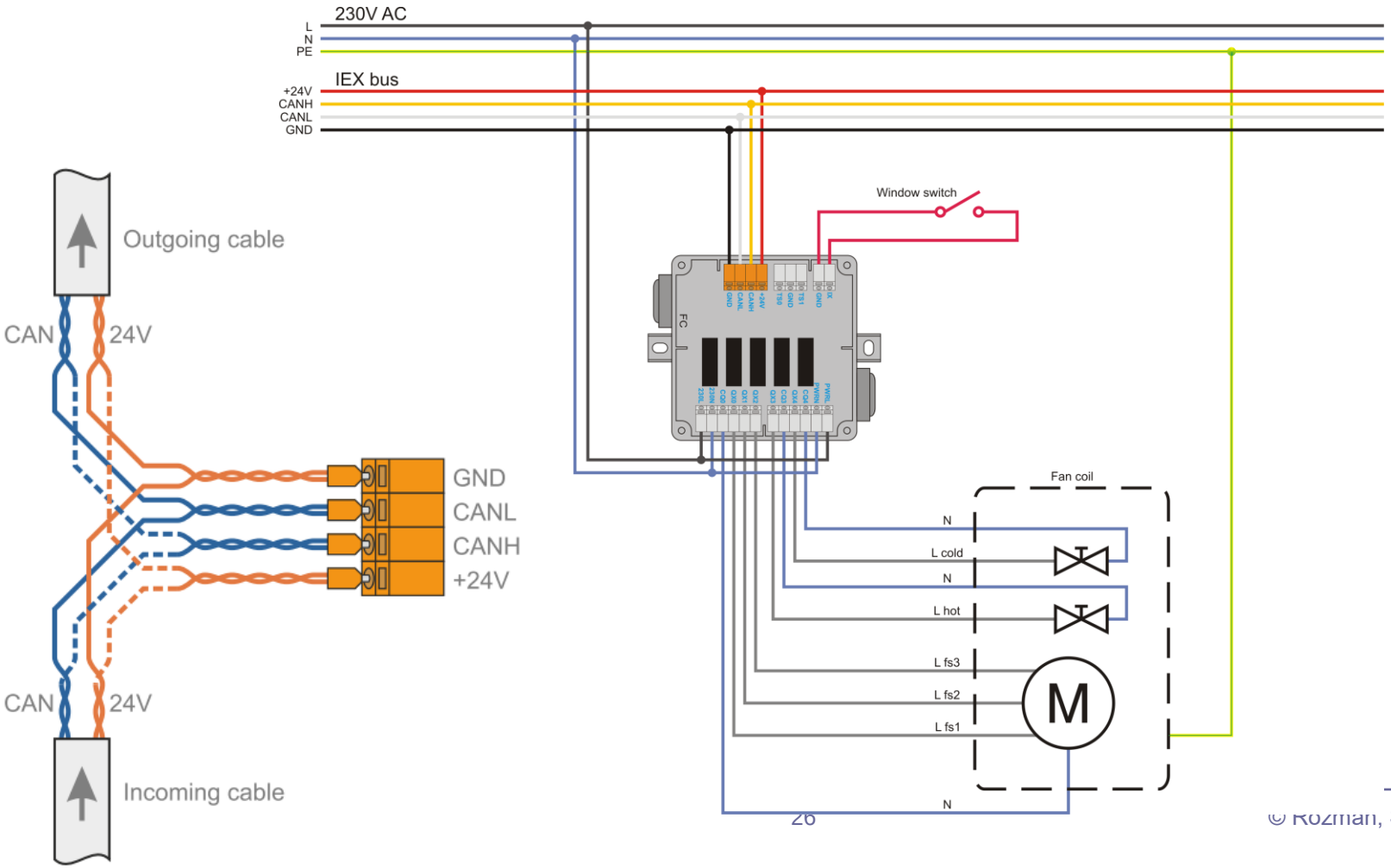
TYPE:
FC-FB



IEX MODULE FC

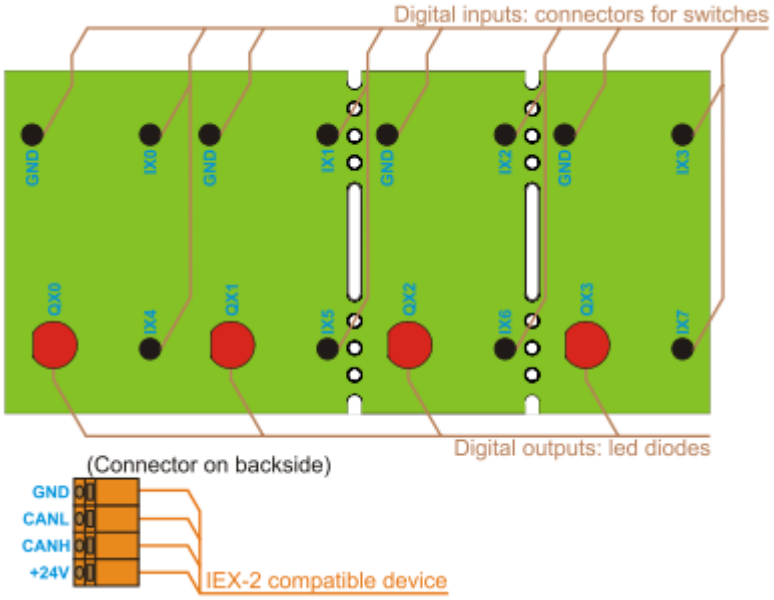
CONNECTING FAN COIL AND WINDOW SWITCH TO FC MODULE

FC



Primeri modulov

IEX MODULE SW-L



SW-L

IEX-2 module
4 switches
4 LED illuminations
Designed for Legrand, Bticino and TEM switches

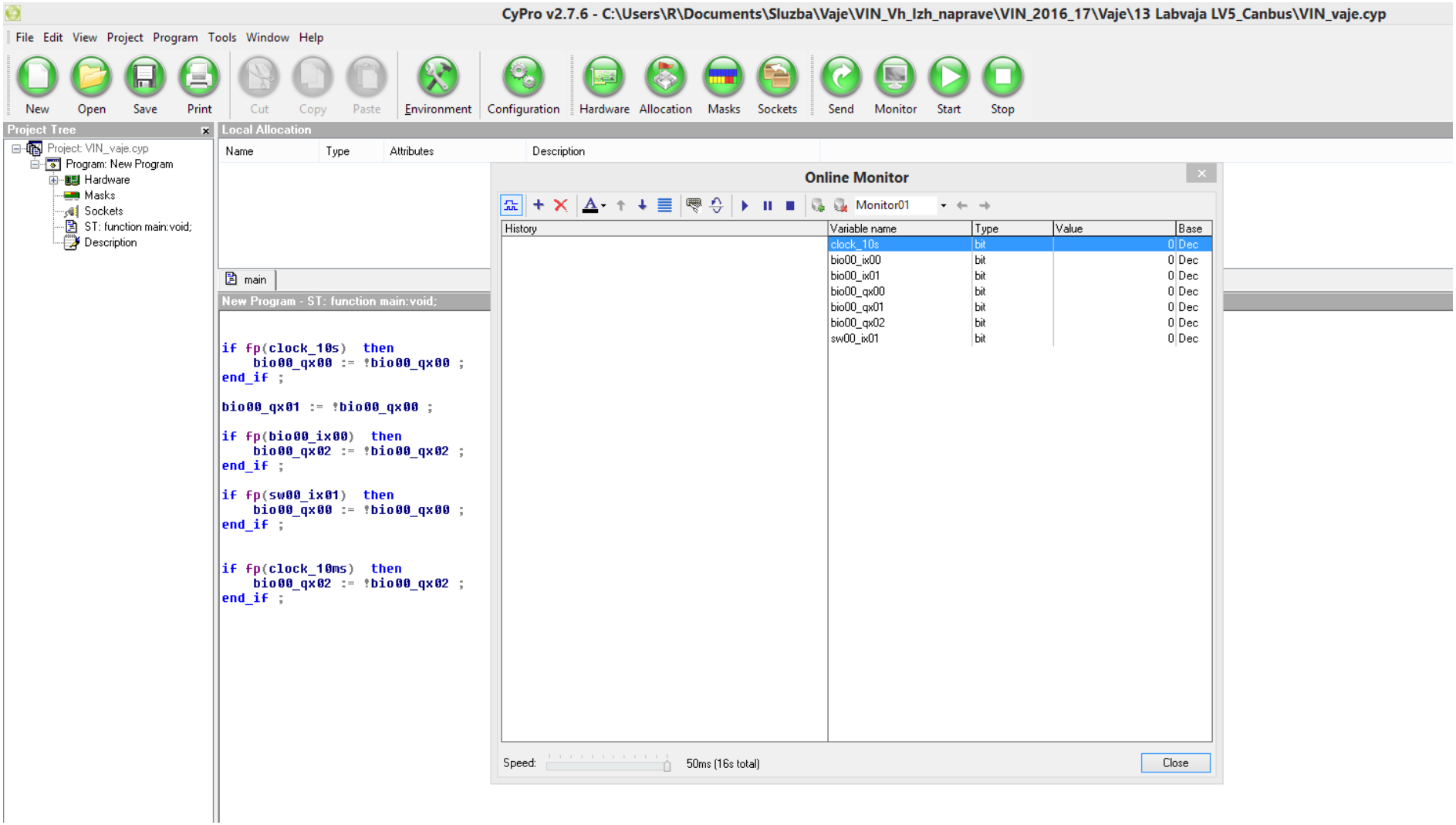
Technical specifications

IX (8 digital inputs)	for connecting 4 switches
Current	2.5mA/12V
QX (4 digital outputs)	
Led illumination	3mm red led-diodes
Power supply	24V DC (18..26V DC), over IEX-2 bus
Power consumption	40mA
Mounting	2 x switch: flush box (diameter 60mm, depth 55mm), in wall 3 x switch: flush box (size 95x58mm, depth 49mm), in wall 4 x switch: flush box (size 120x58mm, depth 49mm), in wall
Dimensions	89x44x38mm

CYPRO – SOFTWARE IDE

Primer IDE

CyPro



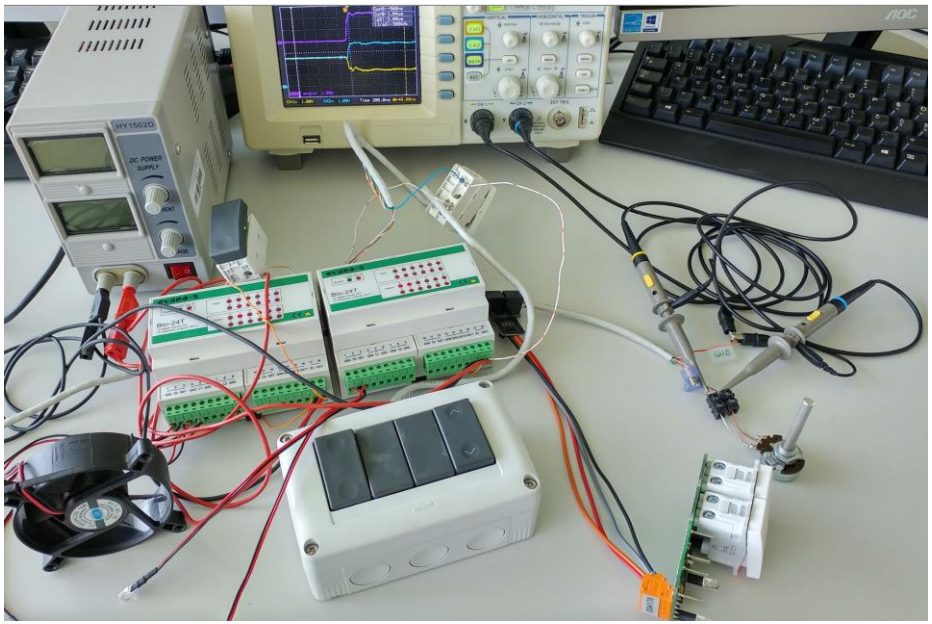
Laboratorijska vaja 11 - LV4

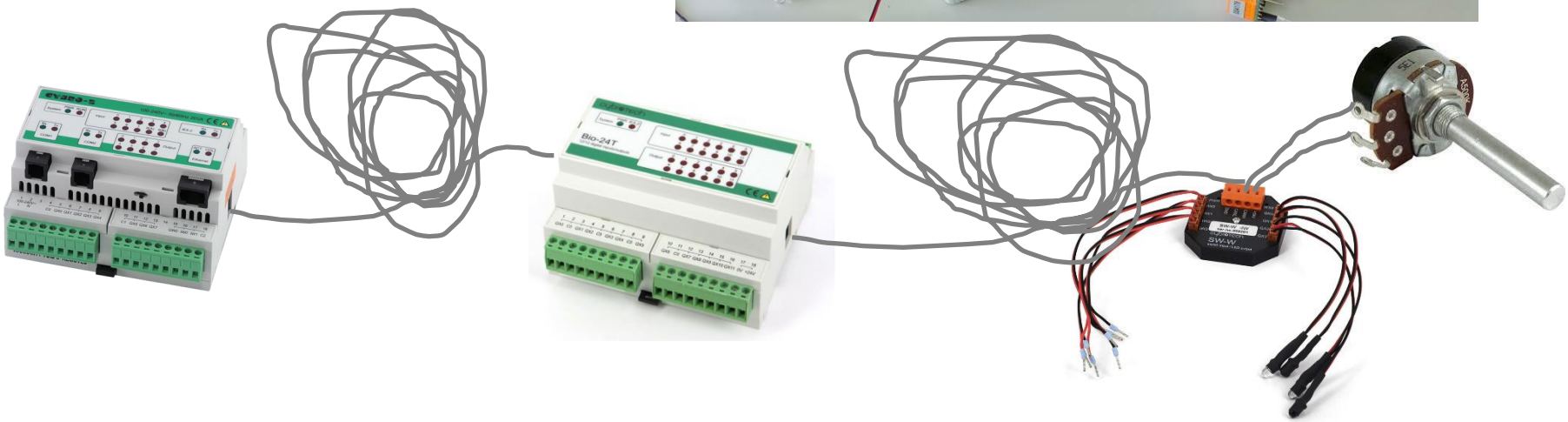
- 11.0: CANBUS osvežitev
- 11.1 Opis primera : Cybrotech CANBUS sistem
- 11.2: Krmiljenje Cybrotech IEX-2 modulov
- 11.3: CANBUS meritve
- 11.4: STM32H7 – osnovni IEX-2 modul

11.2: Krmiljenje Cybrotech IEX-2 modulov

Povežemo enostaven sistem :

- glavni krmilnik Cybro 2
- različni IEX moduli (V/I)

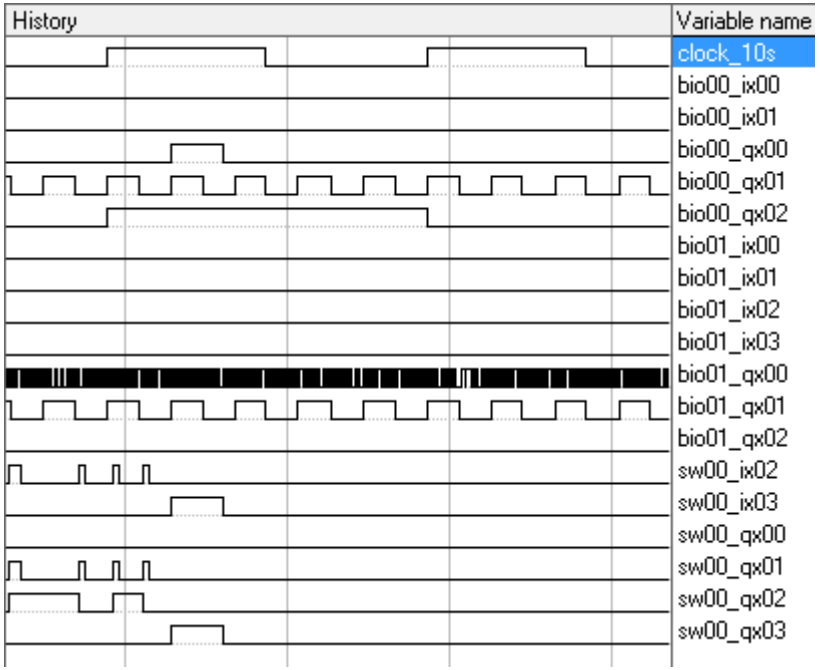




11.2: Krmiljenje Cybrotech IEX-2 modulov

Cypro IDE

Monitor



Program

```
// Periodic tasks
if fp(clock_10s)  then
    bio00_qx02 := !bio00_qx02 ;   // Red LED  every 10 secs
end_if ;


if fp(clock_1s)  then
    bio00_qx01 := !bio00_qx01 ;   // Red LED  every 1 sec
    bio01_qx01 := !bio01_qx01 ;   // Red LED  every 1 sec
end_if ;

if fp(clock_10ms)  then
    bio01_qx00 := !bio01_qx00 ;   // Red LED   every 10 msec
end_if ;

if fp(bio00_ix00)  then
    bio00_qx02 := !bio00_qx02 ;   // Red LED  on keypress
end_if ;


// SW Switch -> LED indicator & ventilator
sw00_qx03  := sw00_ix03;
bio00_qx00   := sw00_ix03;

sw00_qx01  := sw00_ix02;        // SW Key -> LED indicator

if fp(sw00_ix02)  then
    sw00_qx02 := !sw00_qx02 ;   // SW Key -> change LED indicator
end_if ;
```

11.2: Krmiljenje Cybrotech IEX-2 modulov

Cypro IDE – opisi modulov v .cym datotekah

BIO-24.cym

```
object THWModule
  Name = 'Bio-24'
  CardID = 11
  Description = 'Binary 12 inputs/12 outputs, 4 fast counters'
  Capabilities = []
  DisplayWidth = 0
  DisplayHeight = 0
  MaskMemorySize = 0
  VarPrefix = 'bio?_'
  IOAllocData =
…
 item
     Typ = vaOutBit
     EventPriority = epOnChange
     Vars =
     <
      item
        Name = 'qx*'
        Description = 'Relay output (0-open, 1-closed).'
        Offset = 0
      end
…
```

Program

```
// Periodic tasks
if fp(clock_10s)  then
    bio00_qx02 := !bio00_qx02 ;   // Red LED  every 10 secs
end_if ;

if fp(clock_1s)  then
    bio00_qx01 := !bio00_qx01 ;   // Red LED  every 1 sec
    bio01_qx01 := !bio01_qx01 ;   // Red LED  every 1 sec
end_if ;

if fp(clock_10ms)  then
    bio01_qx00 := !bio01_qx00 ;   // Red LED   every 10 msec
end_if ;

if fp(bio00_ix00)  then
    bio00_qx02 := !bio00_qx02 ;   // Red LED  on keypress
end_if ;


// SW Switch -> LED indicator & ventilator
sw00_qx03  := sw00_ix03;
bio00_qx00   := sw00_ix03;

sw00_qx01  := sw00_ix02;        // SW Key -> LED indicator

if fp(sw00_ix02)  then
    sw00_qx02 := !sw00_qx02 ;   // SW Key -> change LED indicator
end_if ;
```

Laboratorijska vaja 11 - LV4

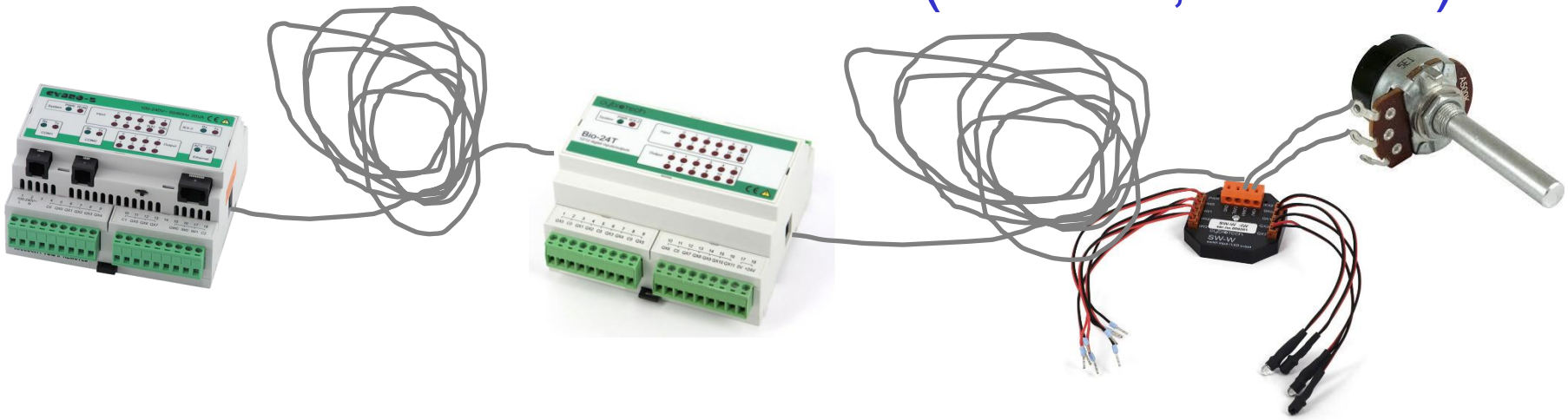
- 11.0: CANBUS osvežitev
- 11.1 Opis primera : Cybrotech CANBUS sistem
- 11.2: Krmiljenje Cybrotech IEX-2 modulov
- 11.3: CANBUS meritve
- 11.4: STM32H7 – osnovni IEX-2 modul

11.3: CANBUS meritve

Izmerite stanje na vodilu pri :

- Različnih zaključitvah na koncu vodila
  - Odprte sponke, 500ohm, zaključitev (107ohm)

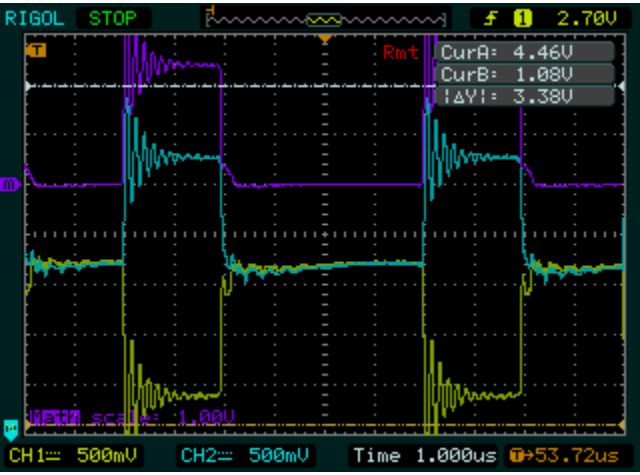
- Dveh različnih bitnih hitrostih (500kb/s, 100kb/s)



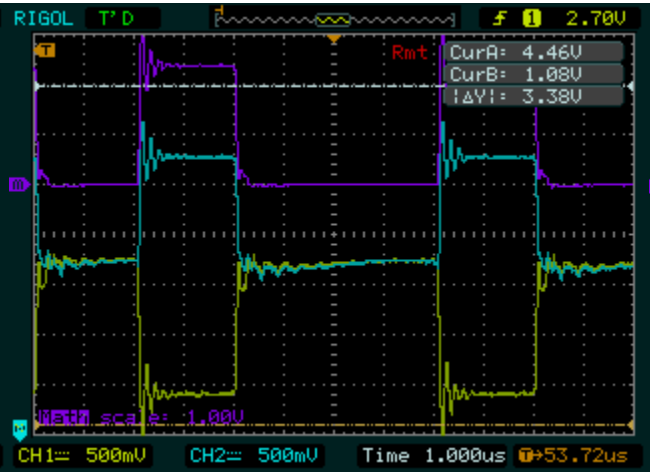
11.3: CANBUS meritve

500kb/s:

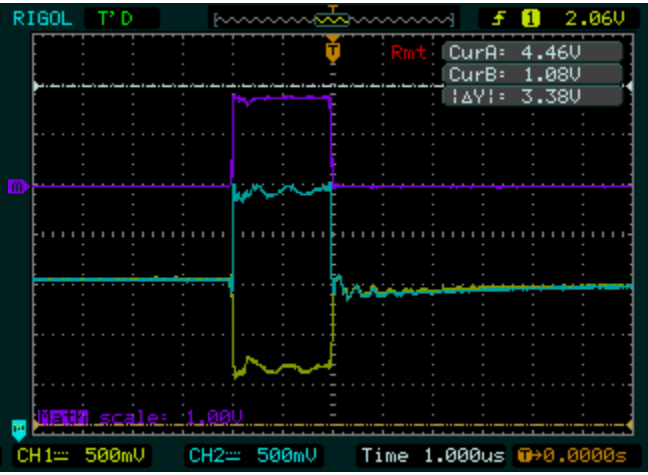
Odprte sponke



500ohm



107ohm

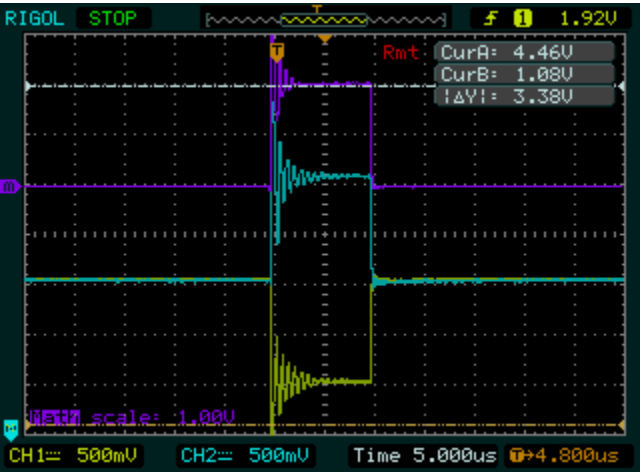


3 zavitki UTP kabla s spojniki – cca 40m…

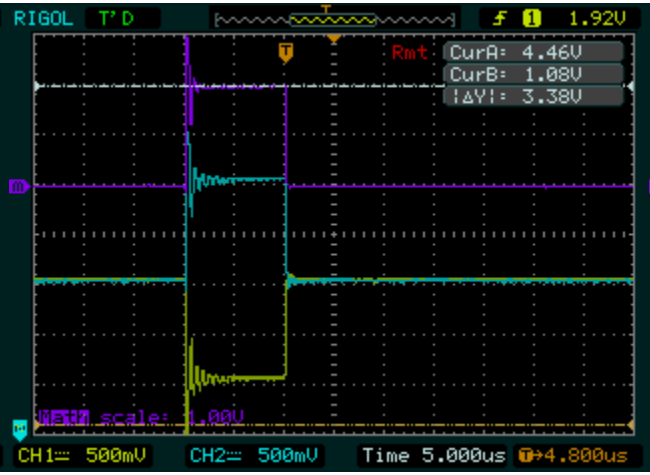
11.3: CANBUS meritve

100kb/s:

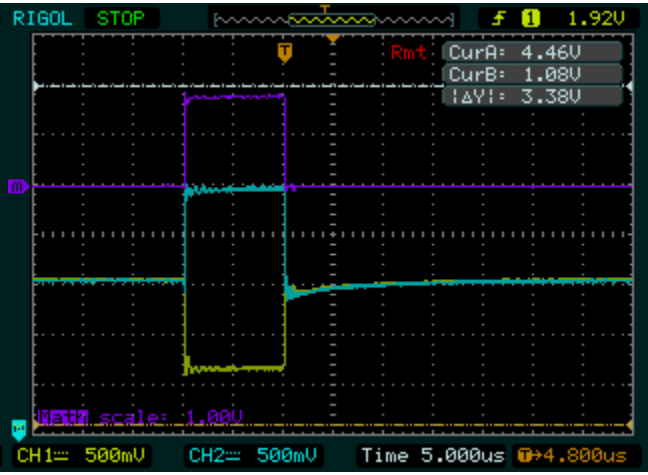
Odprte sponke



500ohm

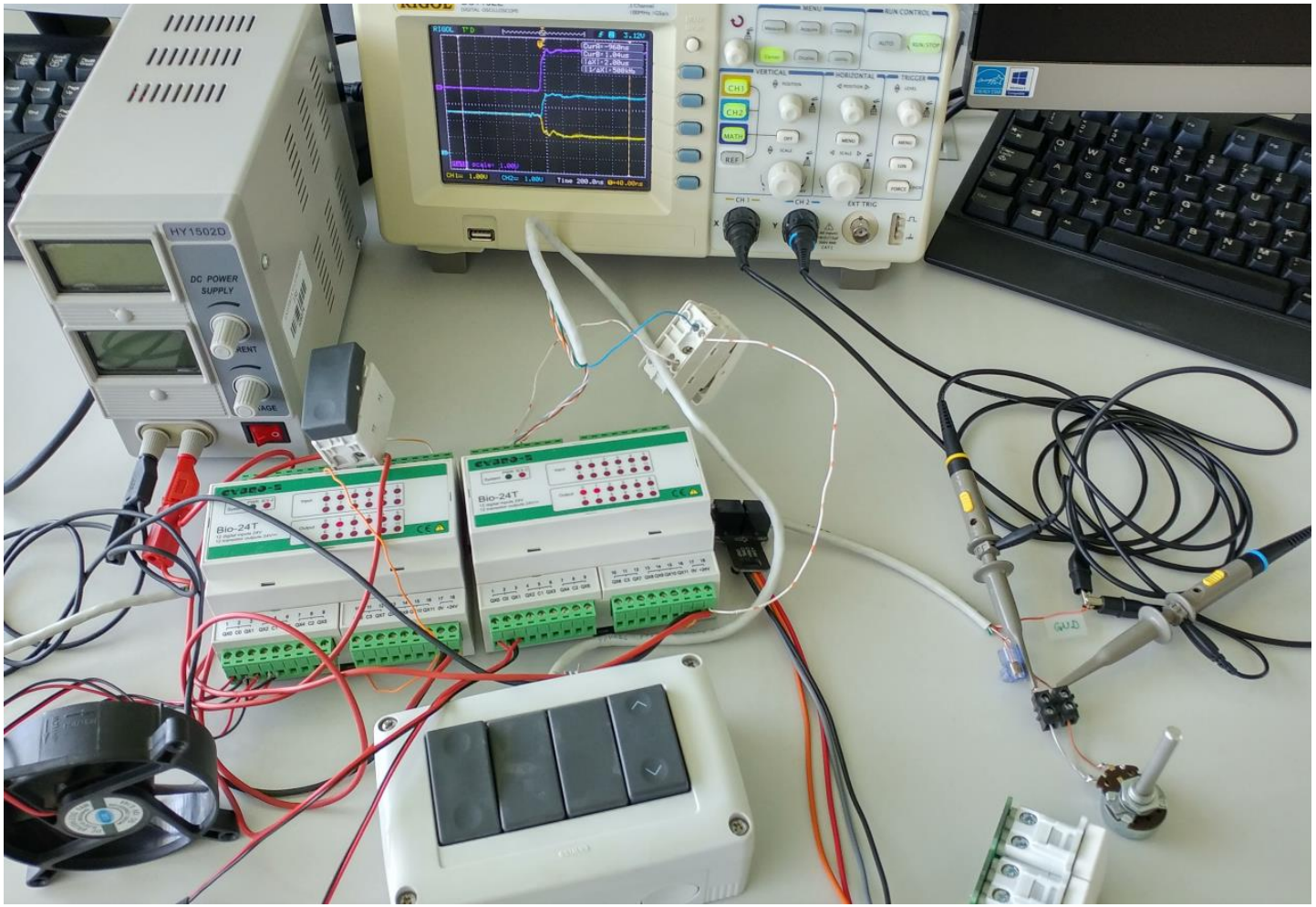


107ohm



3 zavitki UTP kabla s spojniki – cca 40m…

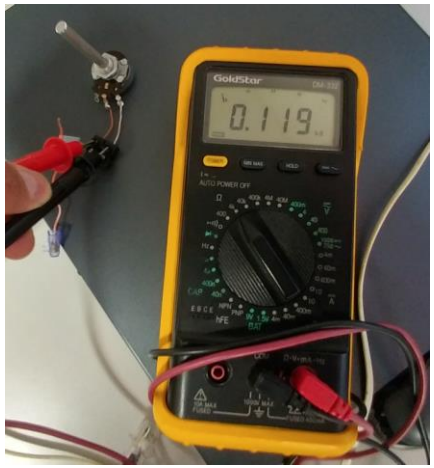
11.3: CANBUS meritve



11.3: CANBUS meritve

Nezaključena linija

Zaključena linija



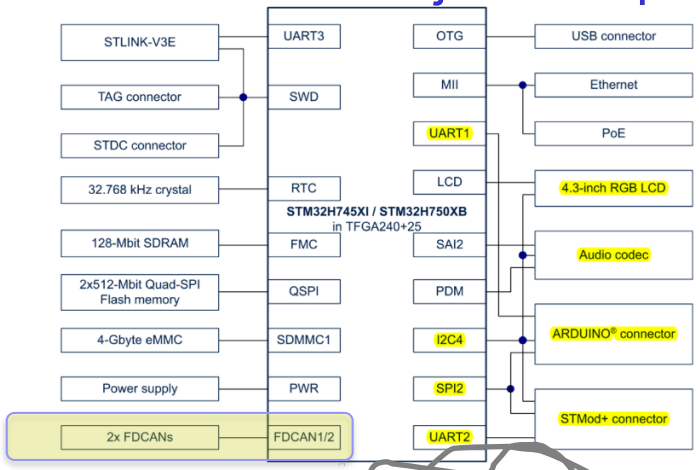
Laboratorijska vaja 11 - LV4

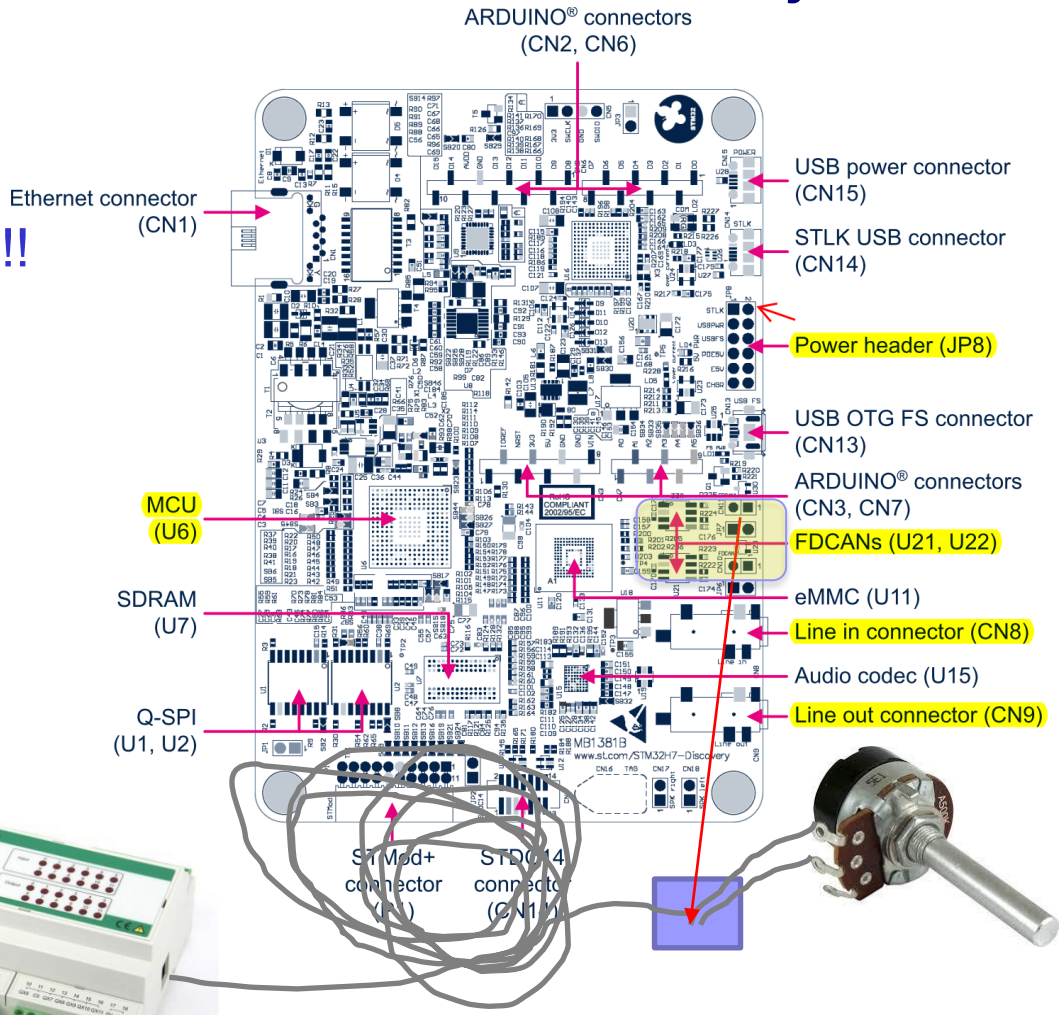
- 11.0: CANBUS osvežitev
- 11.1 Opis primera : Cybrotech CANBUS sistem
- 11.2: Krmiljenje Cybrotech IEX-2 modulov
- 11.3: CANBUS meritve
- 11.4: STM32H7 – osnovni IEX-2 modul

11.4: STM32H7 – osnovni IEX-2 modul - Ideja

Strojna oprema:

- STM32H750 Discovery in …
- CAN PHY vezje že na plošči !!!

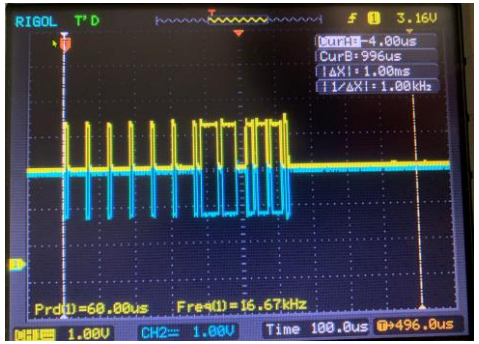


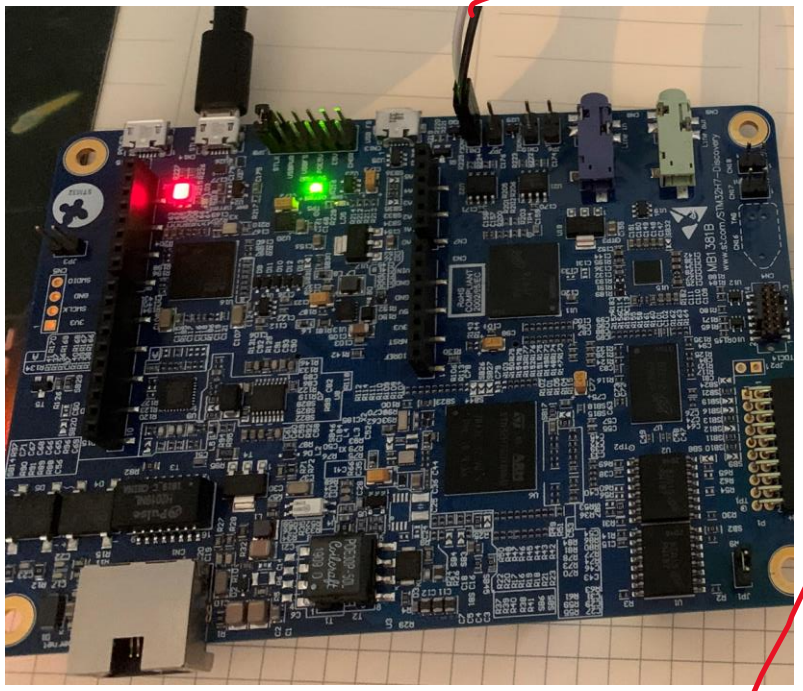


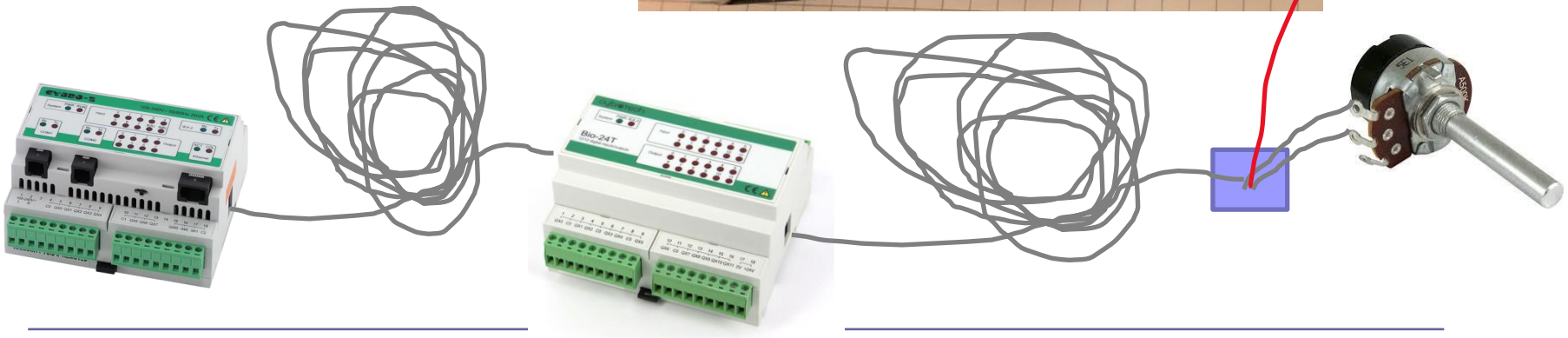
11.4: STM32H7 – osnovni IEX-2 modul

Strojna oprema:

- STM32H7 Discovery
  - že vsebuje CANBUS PHY vezje

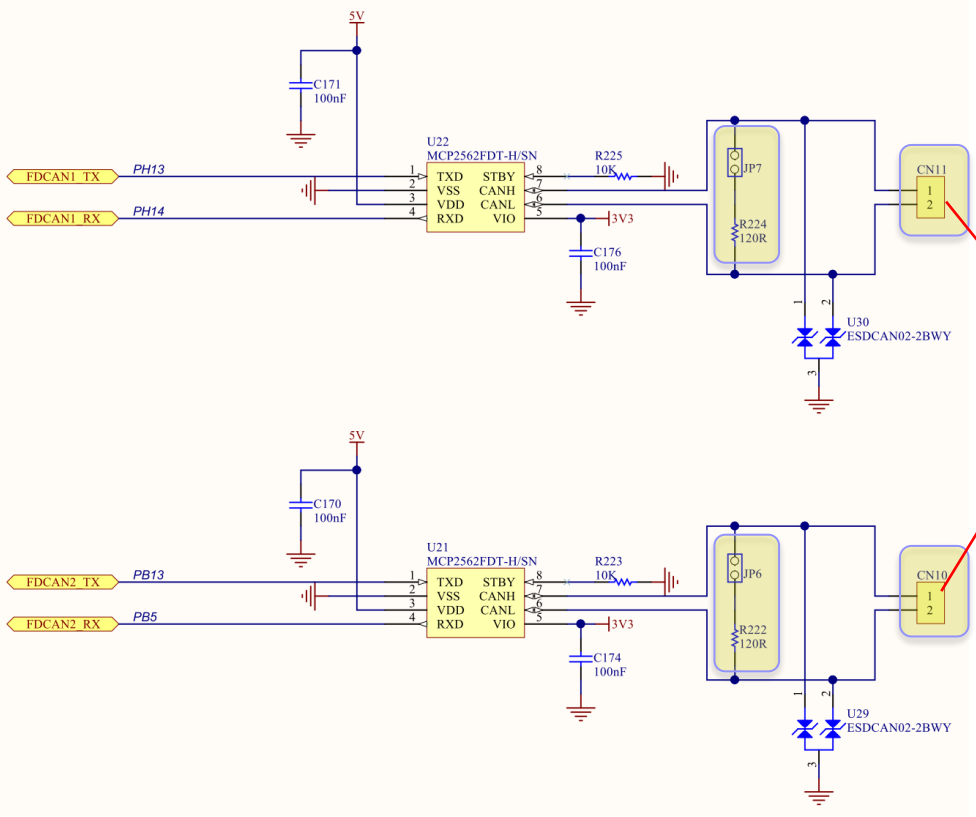


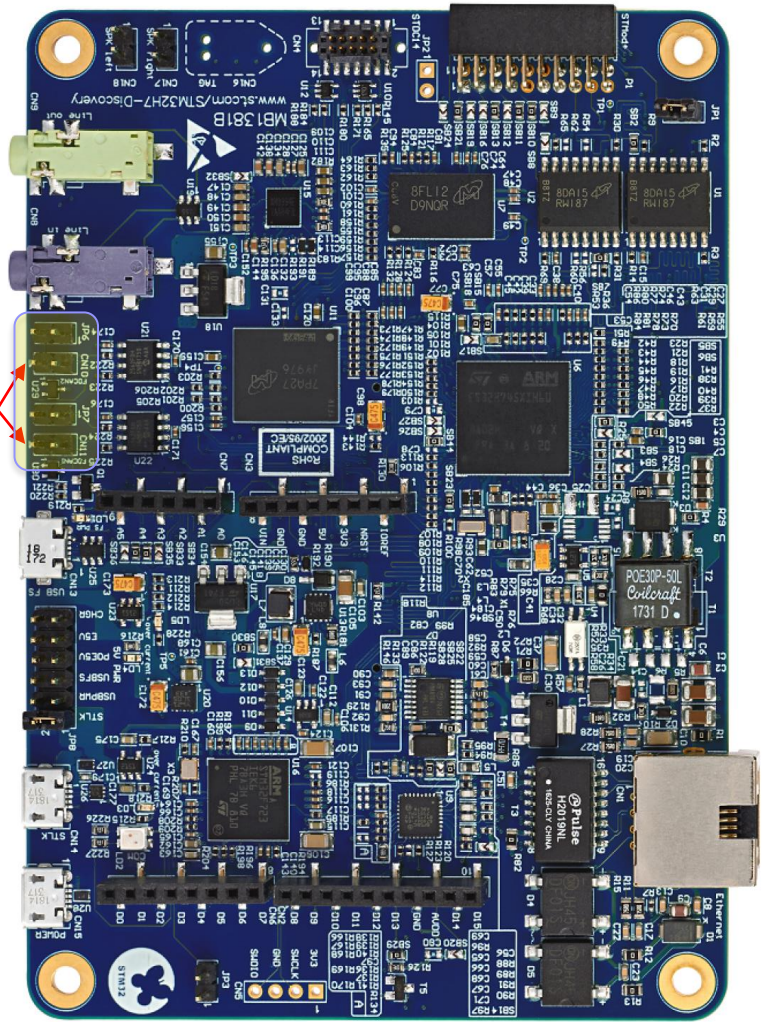




11.4: STM32H7 – osnovni IEX-2 modul

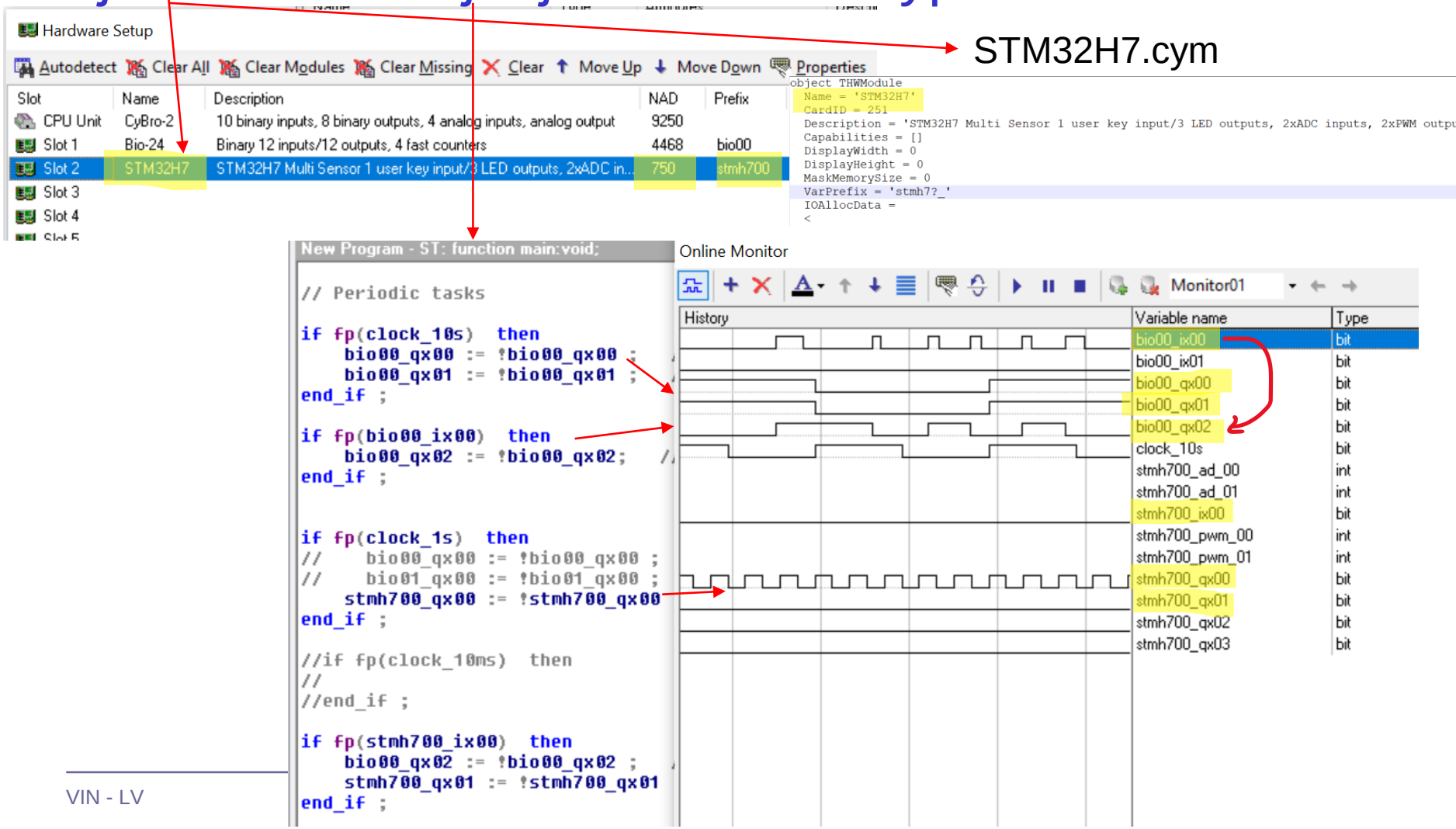
Shema :





11.4 : STM32 – osnovni IEX-2 modul

Vključitev in krmiljenje modula – Cypro IDE



11.4 : STM32 – osnovni IEX-2 modul

Cypro IDE – opisi modulov so v .cym datotekah

STM32H7.cym (definicija modula)

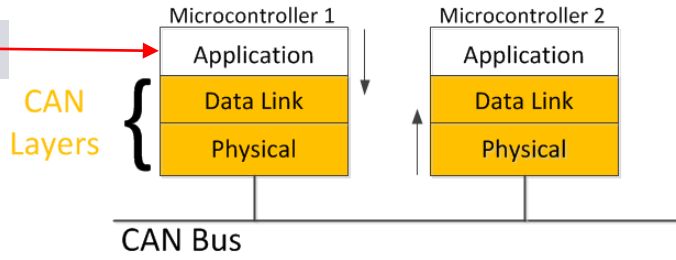
```
object THWModule
  Name = 'STM32H7'
  CardID = 251
  Description = 'STM32H7 Multi Sensor 1 user key input/4 LED outputs, 2xADC inputs, 2xPWM outputs'
  Capabilities = []
  DisplayWidth = 0
  DisplayHeight = 0
  MaskMemorySize = 0
  VarPrefix = 'stmh7?_'
  IOAllocData =
      item
        Typ = vaInBit
        EventPriority = epNone
        Vars =
        <
          item
            Name = 'ix*'
            Description = 'User (blue) key - button.'
            Offset = 0
          end
        >
item
  Typ = vaOutBit
  EventPriority = epOnChange
  Vars =
  <
    item
      Name = 'qx*'
      Description = 'LED output (0-off, 1-on).'
      Offset = 0
    end
    item
      Name = 'qx*'
      Description = 'LED output (0-off, 1-on).'
      Offset = 1
    end
    item
      Name = 'qx*'
      Description = 'LED output (0-off, 1-on).'
      Offset = 2
    end
    item
      Name = 'qx*'
      Description = 'LED output (0-off, 1-on).'
      Offset = 3
    end
  >
end
```

```
item
  Typ = vaOutWord
  EventPriority = epOnChange
  Vars =
  <
    item
      Name = 'pwm_*'
      Description = 'Level should be between 0-127.'
      Offset = 0
    end
    item
      Name = 'pwm_*'
      Description = 'Level should be between 0-127.'
      Offset = 1
    end
  >
end
```

```
item
  Typ = vaInWord
  EventPriority = epNone
  Vars =
  <
    item
      Name = 'ad_*'
      Description = 'ADC Channel.'
      Offset = 0
    end
    item
      Name = 'ad_*'
      Description = 'ADC Channel.'
      Offset = 1
    end
  >
end
```

PLC program - uporaba

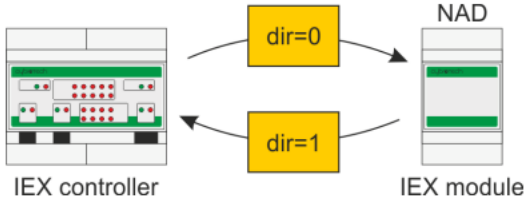
```
    fp(clock_1s)  then
       bio00_qx00 := !bio00_qx00 ;   // Red LED  every 1 sec
       bio01_qx00 := !bio01_qx00 ;   // Red LED  every 1 sec
     stmh700_qx00 := !stmh700_qx00 ;
end_if ;

//if fp(clock_10ms)  then
//
//end_if ;

if fp(stmh700_ix00)  then
    bio00_qx02 := !bio00_qx02 ;   // Red LED  on keypress
    stmh700_qx01 := !stmh700_qx01 ;
end_if ;
```



INTEGRA BM SYSTEM

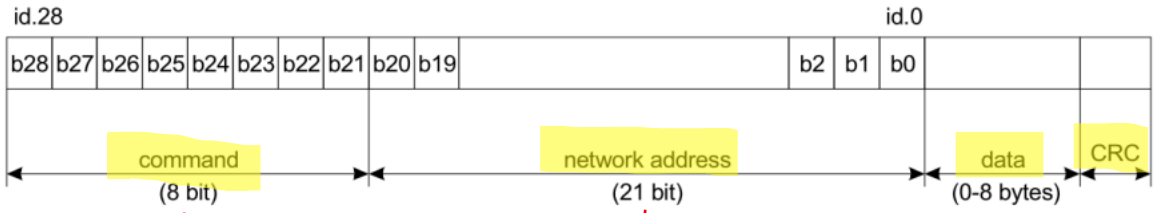
IEX protocol (nadgradnja CANBUS)

IEX PROTOCOL v2.8 POVZETEK

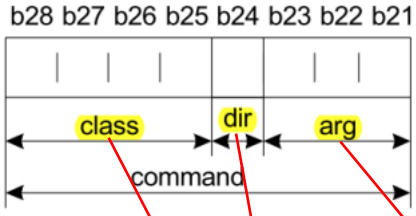


General

IEX-2 is based on CAN 2.0B. Message format is defined as follows:



Command summary



NAD – unikatni naslov IEX modula

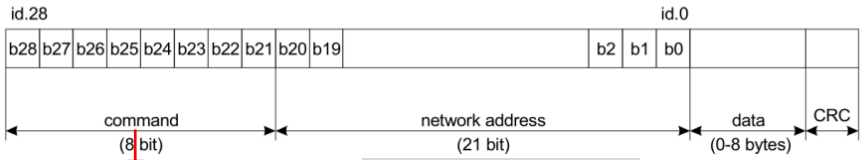
command	hex	binary
IX_DATA	38h + xxx	0011-1xxx
QX_DATA	30h + xxx	0011-0xxx
IW_DATA	78h + xxx	0111-1xxx
QW_DATA	70h + xxx	0111-0xxx

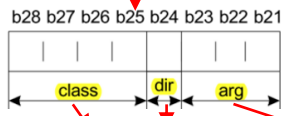
arg	IX / QX / IW / QW
111	28..31 B/W
110	24..27 B/W
101	20..23 B/W
100	16..19 B/W
011	12..15 B/W
010	8..11 B/W
001	4.. 7 B/W
000	0.. 3 B/W

command	class	dir	command	arg	data bytes	description	PCAN view
	0000						
	0001						
	0010						
IX_DATA	0011	1		xxx	data(1..4)	binary inputs	070-07Exxxxxh
QX_DATA		0		xxx	data(1..4)	binary outputs	060-06Exxxxxh
	0100						
	0101						
	0110						
IW_DATA	0111	1		xxx	data(2..8)	analog inputs	0F0-0FExxxxxh
QW_DATA		0		xxx	data(2..8)	analog outputs	0E0-0EExxxxxh
BAUDSYNC	1111	1		111	-	autobaud sync msg	1FFFFFFFh

General

IEX-2 is based on CAN 2.0B. Message format is defined as follows:





NAD – unikatni naslov IEX modula

command	class	dir	command	arg	data bytes	description	PCAN view
	0000						
	0001						
	0010						
IX_DATA	0011	1		xxx	data(1..4)	binary inputs	070-07Exxxxxh
QX_DATA		0		xxx	data(1..4)	binary outputs	060-06Exxxxxh
	0100						
	0101						
	0110						
IW_DATA	0111	1		xxx	data(2..8)	analog inputs	0F0-0FExxxxxh
QW_DATA		0		xxx	data(2..8)	analog outputs	0E0-0EExxxxxh
BAUDSYNC	1111	1		111	-	autobaud sync msg	1FFFFFFFh

IX_DATA : modul sporoči stanje dig. vhodov

QX_DATA: modul sprejme stanje dig. izhodov

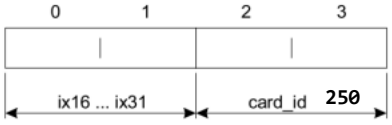
definicije:

```
#define CLASS_MASK 0xF0 /* mask for class - command */
#define DIR_MASK 0x08 /* mask for direction bit */
#define ARG_MASK 0x07 /* mask for argument */
#define IX_STATUS 0x04 /* Status Message Argument */


#define IX_DATA 0x38 /* binary inputs */
#define QX_DATA 0x30 /* binary outputs */
#define IW_DATA 0x78 /* analog inputs */
#define QW_DATA 0x70 /* analog outputs */

#define BAUDSYNC 0xF0 /* baud sync message (for autobaud sync of modules) */

#define IEX2_COMMAND_BIT_DATA   0x6000000
#define IEX2_DIRECTION_NODE2RC  0x1000000
#define IEX2_ARGUMENT_SYS_DATA16 0x800000
#define IEX2_ID_SEND_ONBUS_STATUS (IEX2_COMMAND_BIT_DATA |
IEX2_DIRECTION_NODE2RC | IEX2_ARGUMENT_SYS_DATA16)
//---------------------------------------------------------------------------
// S T A R T O F G L O B A L C O N F I G S E C T I O N

#define NAD_default (long)750 /*!< Defines Node V4 NAD for IEX bus protocol.
It could be NAD_v2+27. */
// These are IDs that are reported to IEX master for module identification
(read appropriate .cym files)
#define IEX2_CYM_ID 251 // 255 is max, select unique ID, also specified in
.cym file


unsigned long status_id = NAD_default +IEX2_ID_SEND_ONBUS_STATUS;
unsigned char status_data[4] = {0,0,0,IEX2_CYM_ID};
```

STATUS_ID

STATUS_ID is a special case of IX_DATA message. It contains data bits ix16-ix31 (2 bytes) and card_id (2 bytes):

IEX → PLC IX_DATA + 100 +



Module must send STATUS_ID every 500ms (+/-10ms). Module may send a range of input bits at any time (IX_DATA with no card_id bytes), but that is not considered as status message. STATUS_ID is used for module autodetection.

```
CurTimeStamp = HAL_GetTick();
if ((CurTimeStamp - lastTime1s) >= 500) {
        retval = HAL_FDCAN_AddMessageToTxFifoQ(&hfdcan1,
        &TxHeader, status_data);
        CanTxMsgCnt++;
        snprintf (SendBuffer,BUFSIZE,"\r\nCAN Message Nr. %d
        sent!\r\n", CanTxMsgCnt);
        HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1
        000);
        lastTime1s = HAL_GetTick();
}
```

11.4: STM32 – osnovni IEX-2 modul

Programska oprema – CubeIDE Projekt - izseki

main.c:

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{

// Check for received CANBUS messages
if(HAL_FDCAN_GetRxFifoFillLevel(&hfdcan1, FDCAN_RX_FIFO0) != 0)
{
        retval = HAL_FDCAN_GetRxMessage(&hfdcan1, FDCAN_RX_FIFO0, &RxHeader, RxData);
        CanRxMsgCnt++;
        CANBus_Parse_RX_Message (RxHeader.Identifier,RxHeader.DataLength, RxData);
}

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */

CurTimeStamp = HAL_GetTick();
if ((CurTimeStamp - lastTime1s) >= 500) {
        retval = HAL_FDCAN_AddMessageToTxFifoQ(&hfdcan1, &TxHeader, status_data);
        if (retval == HAL_OK) {
                CanTxMsgCnt++;
                snprintf (SendBuffer,BUFSIZE,"\r\nCAN Message Nr. %d sent!\r\n", CanTxMsgCnt);
                HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1000);
        }
        lastTime1s = HAL_GetTick();
}

}
/* USER CODE END 3 */
```

```
uint32_t CANBus_Parse_RX_Message (uint32_t ID,uint32_t
msg_size, unsigned char dptr [])
{
int iex_cmd;
long iex_NAD;
int iex_arg;
int iex_slot;
uint8_t bitmask, iex_dir, iex_class;
uint16_t ix_temp;

iex_cmd = ID >> 21;
iex_NAD = ID & 0x1fffff;
iex_arg = iex_cmd & ARG_MASK;
iex_dir = (iex_cmd & DIR_MASK) >> 3;
iex_class = (iex_cmd & CLASS_MASK) >> 4 ;

if ( iex_class != 15) {
        snprintf (SendBuffer,BUFSIZE,"CAN_RX:
        Class:%d|Dir:%d|Arg:%d|NAD:%d|Len:%d| 0x%02x %02x
        %02x %02x\r\n", iex_class, iex_dir, iex_arg,
        iex_NAD, msg_size,dptr [0],dptr [1],dptr [2],dptr
        [3]);
        HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuf
        fer),1000);
  }
}
```

```
#define NAD_default (long)750 #define IEX2_CYM_ID 251 //
255 is max, select unique ID, also specified in .cym file
unsigned long status_id = NAD_default
+IEX2_ID_SEND_ONBUS_STATUS;
unsigned char status_data[4] = {0,0,0,IEX2_CYM_ID};
// Prepare Tx Header
TxHeader.Identifier = status_id;
TxHeader.IdType = FDCAN_EXTENDED_ID;
TxHeader.TxFrameType = FDCAN_DATA_FRAME;
TxHeader.DataLength = FDCAN_DLC_BYTES_4;
TxHeader.ErrorStateIndicator = FDCAN_ESI_ACTIVE;
TxHeader.BitRateSwitch = FDCAN_BRS_OFF;
TxHeader.FDFormat = FDCAN_CLASSIC_CAN;
TxHeader.TxEventFifoControl = FDCAN_NO_TX_EVENTS;
TxHeader.MessageMarker = 0;
```

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_C_CAN_IEX_Module_Base

11.4: STM32 – osnovni IEX-2 modul – rešitev ?

Programska oprema – CubeIDE Projekt - izseki

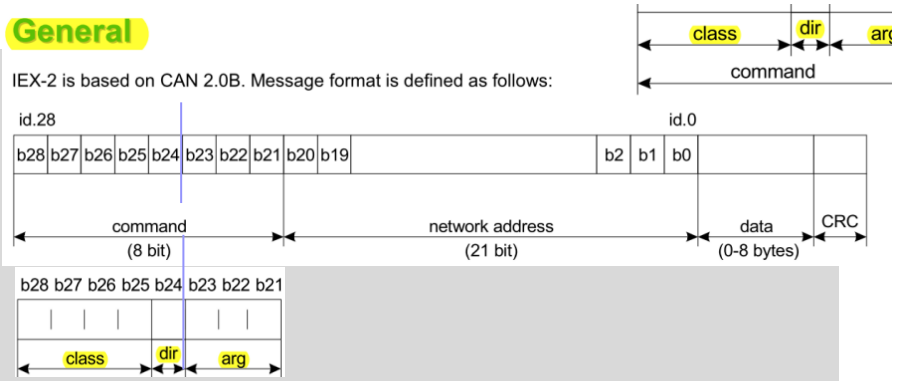
```
STM32H7 in IEX-2 protokol (minimalna implementacija)


modul mora :

• vsake pol sekunde poslati status sporočilo z vrsto modula, da ga Cybro zazna (že v kodi)
      CurTimeStamp = HAL_GetTick();
      if ((CurTimeStamp - lastTime1s) >= 500) {
            // Send two bytes (0xCC,0xCC) on FDCAN1 (as normal classic CAN message with ID
            0x555)
            retval = HAL_FDCAN_AddMessageToTxFifoQ(&hfdcan1, &TxHeader, status_data);
            if (retval == HAL_OK) {
                  CanTxMsgCnt++;
                  snprintf (SendBuffer,BUFSIZE,"\r\nCAN Message Nr. %d sent!\r\n",
                  CanTxMsgCnt);
                  HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1000);
            }
            lastTime1s = HAL_GetTick();
      }

• - sprejemati QX sporočila, ki določajo stanje LED diod kot izhodov
            dopolni program

• - pošiljati IX sporočila, če se spremeni stanje USER tipke
            dopolni program
```

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_C_CAN_IEX_Module_Base

11.4: STM32 – osnovni IEX-2 modul – rešitev ?

Programska oprema – CubeIDE Projekt - izseki

```
#define CLASS_MASK 0xF0 /* mask for class - command */
#define DIR_MASK 0x08 /* mask for direction bit */
#define ARG_MASK 0x07 /* mask for argument */
#define IX_STATUS 0x04 /* Status Message Argument */


#define IX_DATA 0x38 /* binary inputs */
#define QX_DATA 0x30 /* binary outputs */
#define IW_DATA 0x78 /* analog inputs */
#define QW_DATA 0x70 /* analog outputs */


#define BAUDSYNC 0xF0 /* baud sync message (for autobaud sync of modules) */


#define IEX2_COMMAND_BIT_DATA   0x6000000
#define IEX2_DIRECTION_NODE2RC  0x1000000
#define IEX2_ARGUMENT_SYS_DATA16 0x800000
#define IEX2_ID_SEND_ONBUS_STATUS (IEX2_COMMAND_BIT_DATA | IEX2_DIRECTION_NODE2RC | IEX2_ARGUMENT_SYS_DATA16)
```



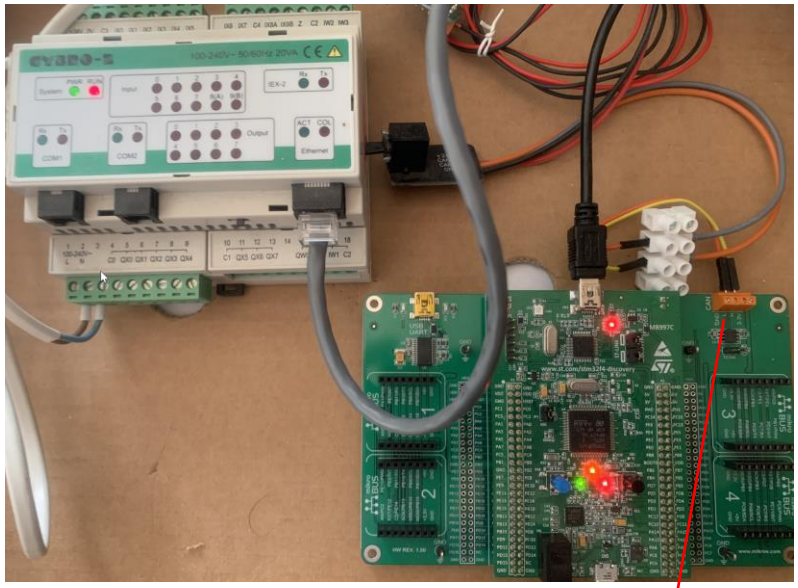
command	class	dir	command	arg	data bytes	description	PCAN view
	0000						
	0001						
	0010						
IX_DATA	0011	1		xxx	data(1..4)	binary inputs	070-07Exxxxxh
QX_DATA		0		xxx	data(1..4)	binary outputs	060-06Exxxxxh
	0100						
	0101						
	0110						
IW_DATA	0111	1		xxx	data(2..8)	analog inputs	0F0-0FExxxxxh
QW_DATA		0		xxx	data(2..8)	analog outputs	0E0-0EExxxxxh
BAUDSYNC	1111	1		111	-	autobaud sync msg	1FFFFFFFh

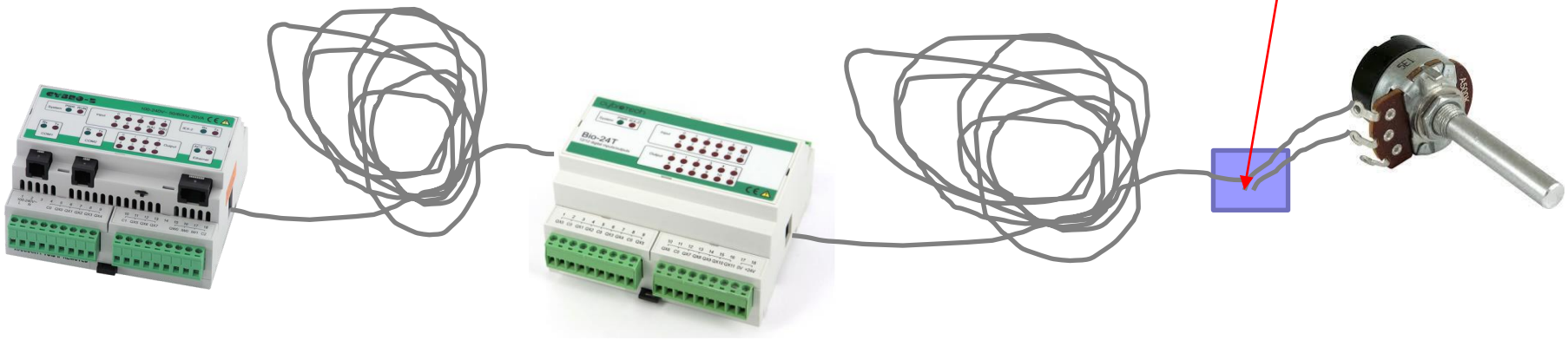
https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_C_CAN_IEX_Module_Base

11.4: STM32F4 – osnovni IEX-2 modul

Strojna oprema:

- STM32F4 Discovery in
- shield (Mikroelektronika)
  - vsebuje CANBUS PHY vezje
- ali zunanje CAN PHY vezje





DN2-LV4: LV4 CANBUS - Meritve in programiranje IEX modula

LV4-1 Meritev CANBUS signalov

Izmerite stanje signalov ob izbranem CANBUS sporočilu v primeru nezaključene in zaključene linije. Izračunajte bitno hitrost CANBUS komunikacije. Prikažite slike in opišite postopke.

LV4-2 Cypro IDE in IEX Modul

Na STM32H7 naložite program za osnovni IEX modul (pri tem spremenite njegov naslov NAD na unikatno številko). Na COM portu opazujte izpise prejetih/oddanih sporočil. Povežite ga s CANBUS vodilom in preverite v Cypro IDE, ali ga je zaznal. Dokumentirajte in opišite postopek.

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_C_CAN_IEX_Module_Base

***Neobvezno/dodatno:** lahko dopolnite osnovni program vsaj s sprejemom QX sporočil in oddajo IX sporočil, da se bo na modulu lahko uporabilo tudi tipke in LED diode. Program lahko razširite še na PWM izhoda in ADC vhoda.*