Protokół komunikacyjny sondy cyfrowej CS-26/RS-485 (lub RS-232)

Parametry transmisji : 9600, N, 8, 1
Sonda CS-26/RS-485 dołączona do interfejsu RS-485 pracuje poprawnie w trybie half-duplex. Oznacza to, że normalnie sonda pozostaje w trybie odbiorczym i odpowiada na zapytanie tylko wtedy gdy ramka transmisji jest odpowiednio skonstruowana (zawiera właściwy adres sondy i odpowiedni kod rozkazu).

UWAGA : W wykonaniu sondy dla RS-232, w polu DEVID - adres sondy przyjmuje się stałą wartość tj. adres=1).

FORMAT RAMKI ZAPYTANIA (12 BAJTÓW) :

Nazwa	Liczba Bajtów	Opis	Wartość
PREAMBULE	2	Preambuła	0xAA55
CRC-16	2	CRC-16 liczona od pola SIZE do końca ramki	0xXXXX[1]
SIZE	1	Informacja o liczbie bajtów ramki za polem SIZE	0x07
DESTINATION	1	Przeznaczenie ramki (do sondy paliwowej)	0x50
SOURCE	1	Źródło ramki (od rejestratora pojazdu)	0x43
VERSION	2	Wersja oprogramowania (oraz pole do wysyłania danych)	0xXXXX[1]
TYPE	1	Typ informacji w ramce (np. 0x01, 0x02 itd.)	0xXX
DEVID	2	Adres sondy (dla 0xFFFF tzw. adres „broadcast”)	0xXXXX[1]

FORMAT RAMKI ODPOWIEDZI (20 BAJTÓW) :

Nazwa	Liczba Bajtów	Opis	Wartość
PREAMBULE	2	Preambuła	0xAA55
CRC-16	2	CRC-16 liczona od pola SIZE do końca ramki	0xXXXX[1]
SIZE	1	Informacja o liczbie bajtów ramki za polem SIZE	0x0F
DESTINATION	1	Przeznaczenie ramki (do rejestratora pojazdu)	0x43
SOURCE	1	Źródło ramki (od sondy paliwowej)	0x50
VERSION	2	Wersja oprogramowania (jeśli pole TYPE = 0x01 lub dane jeśli pole TYPE ≠ 0x01)	0xXXXX[1]
TYPE	1	Typ informacji w ramce (np. 0x01, 0x02 itd.	0xXX
DEVID	2	Adres sondy (dla 0xFFFF tzw. adres „broadcast”)	0xXXXX[1]
LEVF	2	Poziom paliwa filtrowany (ze stałą czasową T)	0xXXXX[1]
UZAS	2	Napięcie zasilania sondy	0xXXXX[1]
LEV	2	Poziom paliwa chwilowy	0xXXXX[1]
RESERVE	2	Pole rezerwowe (np. temperatura paliwa)	0x0000

[1] **- kolejność bajtów w ramce dla liczb 16-bitowych wynosi : Byte_LO, Byte_HI**

Uwagi :

- VERSION = 0xXXXX : wersja oprogramowania sondy np. 0x03E8 oznacza wersję 1.000 (liczbę 1000 należy domyślnie podzielić przez 1000) jeżeli pole TYPE wynosi 0x01
- VERSION = 0xXXXX : pomocnicze pole do przesyłania danych do i od sondy np. 0x0002 oznacza nowy adres = 2 jeśli pole TYPE = 0x02 (stary adres znajduje się w polu DEVID, np. DEVID = 0x0001)
- TYPE : pole do wyróżnienia typu informacji (inaczej kod rozkazu) w ramce :
  0x01 - dla standardowej ramki zapytania i odpowiedzi
  dla adresowej ramki zapytania z adresem „broadcast” (65535) w polu adresowym DEVID
  0x02 - zmiana adresu sondy
  0x03 - korekcja minimum poziomu paliwa
  0x04 - kalibracja minimum poziomu paliwa
  0x05 - kalibracja maksimum poziomu paliwa
  0x06 - status kalibracji
  0x07 - kalibracja stałej czasowej filtra cyfrowego
  0x08 - korekcja zakresu pomiarowego
  0x09 - odczyt stałej czasowej filtra cyfrowego
  0x0A - zapis ustawień fabrycznych (np. zakresu pomiarowego w mm)

0x0B - powrót do ustawień fabrycznych

DEVID –adres sondy ; zakres od 1 do 65534

LEVF – poziom paliwa filtrowany : zakres znamionowy liczbowy od 100 (minimum poziomu paliwa) do 3800 (maksimum poziomu paliwa) , zakres dopuszczalny od 1 do 4095 – uwaga : można ustalić inną reprezentację poziomu np. w mm. Sygnał poziomu jest filtrowany cyfrowo – stała czasowa filtra T = 0...900s

UZAS – wartość napięcia zasilania sondy : zakres 600...3600, co oznacza 6,00...36,00V

LEV – poziom paliwa niefiltrowany : zakres znamionowy liczbowy od 100 (minimum poziomu paliwa) do 3800 (maksimum poziomu paliwa) , zakres dopuszczalny od 1 do 4095 – uwaga : można ustalić inną reprezentację poziomu np. w mm.

RESERVE – pole rezerwowe np. temperatura paliwa mierzona za pomocą cyfrowego termometru firmy DALLAS

1. Zależność między wartością w polu a rzeczywistą temperaturą T wg kodu U2 tj.
   RESERVE = T dla T≥0 (0...127)
   RESERVE = 256+T dla T<0 (128...255 dla T= -1...-128°C)
2. Zależność między wartością w polu a rzeczywistą temperaturą T wg funkcji :
   RESERVE = 100 + T
   Jeśli T = -10°C, to RESERVE = 90
   Jeśli T = +10°C, to RESERVE = 110

OBLICZANIE SUMY KONTROLNEJ CRC (tak jak w protokole modbus-rtu) REALIZOWANE JEST WEDŁUG NASTĘPUJĄCEGO ALGORYTMU:

1. Załadowanie FFFFh do 16-bitowego rejestru CRC.
2. Pobranie bajtu z bloku danych (zabezpieczana wiadomość) i wykonanie operacji EXOR z młodszym bajtem rejestru CRC (CRCLO). Umieszczenie rezultatu w rejestrze CRC.
3. Przesuniecie zawartości rejestru CRC w prawo o jeden bit połączone z wpisaniem 0 na najbardziej znaczący bit (MSB=0).
4. Sprawdzenie stanu najmłodszego bitu (LSB) w rejestrze CRC. Jeżeli jego stan równa się 0, to następuje powrót do kroku 3 (kolejne przesunięcie), jeżeli 1, to wykonywana jest operacja EXOR rejestru CRC ze stałą A00lh.
5. Powtórzenie kroków 3 i 4 osiem razy, co odpowiada przetworzeniu całego bajtu (ośmiu bitów).
6. Powtórzenie sekwencji 2, 3,4, 5 dla kolejnego bajtu wiadomości. Kontynuacja tego procesu aż do przetworzenia wszystkich bajtów wiadomości.
7. Zawartość CRC po wykonaniu wymienionych operacji jest poszukiwaną wartością CRC.

Przykładowy kod w języku Visual Basic :

- **dla 12 bajtowej ramki zapytania (obliczenia dla ostatnich 8 bajtów tej ramki)**

```
CRC_ZAP = 65535 ' wartość początkowa CRC
For i = 1 To 8                                  'przetwarzanie 8 bajtów ramki
   CRC_ZAP = CRC_ZAP Xor Ramka_zap(i)
   For j = 1 To 8                               'przetwarzanie 8 bitów 1-go bajtu
      If (CRC_ZAP And 1) = 1 Then
      CRC_ZAP = CRC_ZAP \ 2                     'przesunięcie o 1 bit w prawo
      CRC_ZAP = CRC_ZAP Xor 40961               'CRC_ZAP Xor &HA001
      Else
      CRC_ZAP = CRC_ZAP \ 2
      End If
   Next j
Next i
CRC_ZAP_Hi = CRC_ZAP \ 256                      'przes. w prawo o 8 bitów
CRC_ZAP_LO = CRC_ZAP And &HFF                   'maska z liczbą &HFF
```

- **dl 20 bajtowej ramki odpowiedzi (obliczenia dla ostatnich 16 bajtów tej ramki)**

```
For i = 1 To 16                                 'przetwarzanie n=16 bajtów ramki
   CRC_ODP = CRC_ODP Xor Ramka_odp(i)
   For j = 1 To 8                               'przetwarzanie 8 bitów 1-go bajtu
      If (CRC_ODP And 1) = 1 Then
      CRC_ODP = CRC_ODP \ 2                     'przesunięcie o 1 bit w prawo
      CRC_ODP = CRC_ODP Xor 40961               'CRC_ODP Xor &HA001
      Else
      CRC_ODP = CRC_ODP \ 2
      End If
   Next j
Next i
CRC_ODP_Hi = CRC_ODP \ 256                      'przesunięcie w prawo o 8 bitów
CRC_ODP_LO = CRC_ODP And &HFF                   'maska z liczbą &HFF
```

Przykładowy kod w języku C :

Funkcja do obliczenia CRC ma dwa argumenty:
*unsigned char *puchMsg;* wskaźnik do ramki (zapytania lub odpowiedzi), aby pobrać dane binarne
unsigned short usDataLen; liczba bajtów w ramce
Funkcja zwraca CRC typu unsigned short.

```
unsigned short CRC (puchMsg, usDataLen)
unsigned char *puchMsg; /* bufor do obliczenia CRC */
unsigned short usDataLen; /* liczba bajtów w buforze */
{
unsigned char uchCRChi = 0xFF; /* inicjalizacja starszego bajtu CRC */
unsigned char uchCRClo = 0xFF; /* inicjalizacja młodszego bajtu CRC */
while (usDataLen--)
{ uIndex = uchCRChi ^ *puchMsg++; /* obliczenie CRC */
uchCRChi = uchCRClo ^ crc_hi[uIndex];
uchCRClo = crc_lo[uIndex];
}
return(uchCRChi<< | uchCRClo);
}
```

Wszystkie możliwe wartości sumy CRC są umieszczone w dwóch tablicach. Pierwsza tablica zawiera starszy bajt wszystkich z 256 możliwych wartości 16-bitowego pola CRC, natomiast druga tablica młodszy bajt. Wyznaczenie sumy CRC odbywa się poprzez właściwe indeksowanie tych tablic.

```
//tablica starszego bajtu CRC
const unsigned char crc_hi[ ]={
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
};

//tablica młodszego bajtu CRC
const unsigned char crc_lo[ ]={
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
};
```

PRZYKŁADY RAMEK.

1. Ramka zapytania – standardowa - odczyt danych (przy adresie=1)

AA 55 6F 18 07 50 43 E8 03 **01** 01 00

UWAGA : W polu VERSION = 1000 (wersja programu **1000**/1000=1.000), w polu TYPE = 01 kod rozkazu

Ramka odpowiedzi :

AA 55 F5 89 0F 43 50 E8 03 01 01 00 **D8 0E 60 09 D8 0E** 00 00

UWAGA : poziom filtr.= 3800 bitów ; napięcie 24,00V ; poziom niefiltr.=3800 bitów; rezerwa=0

2. Ramka zapytania – korekcja minimum (przy adresie=1)

AA 55 CE D8 07 50 43 E8 03 **03** 01 00

UWAGA : W polu VERSION = 1000 (wersja programu **1000**/1000=1.000), w polu TYPE = 03 kod rozkazu

Ramka odpowiedzi :

AA 55 39 D0 0F 43 50 00 80 **03** 01 00 64 00 60 09 64 00 00 00

Uwaga : poziom filtr.= 100 bitów ; napięcie 24,00V ; poziom niefiltr.=100 bitów; rezerwa=0):
W polu „Version" dodatkowa dana z sondy– poziom liczbowy napięcia czujnika (=0mV→32768)

3. Ramka zapytania – korekcja zakres (np. na 400mm, przy adresie =1)

AA 55 C6 4F 07 84 18 **90 01 08** 01 00

UWAGA : W polu VERSION = 400 wartość nowego zakresu wysyłana do sondy, w polu TYPE = 08 kod rozkazu

Ramka odpowiedzi :

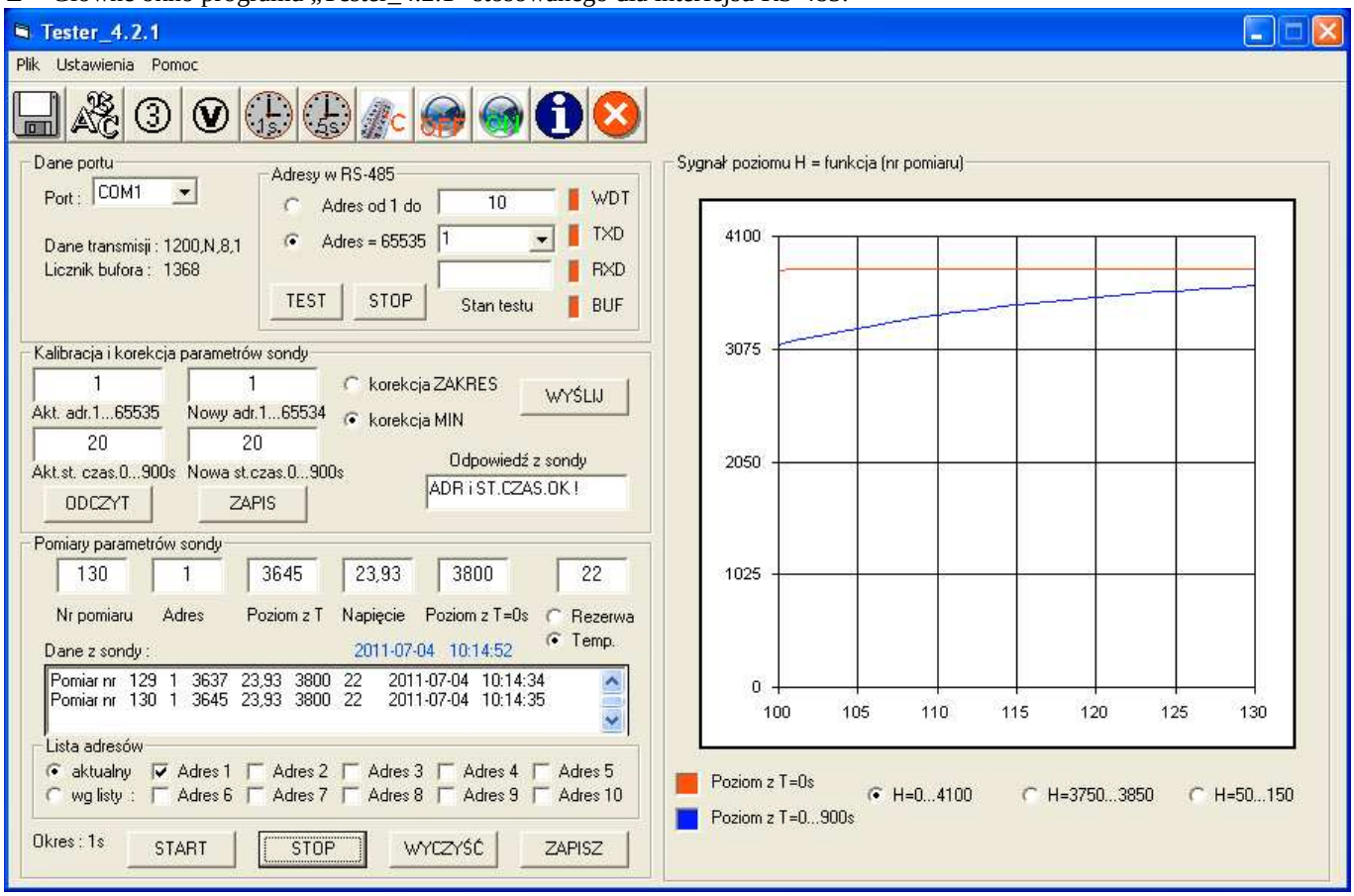
AA 55 22 18 0F 43 50 **90 01 08** 01 00 64 00 60 09 64 00 00 00

W polu „Version" dodatkowa dana z sondy tj. wartość nowego zakresu przyjęta przez sondę tj. 400mm)

PROGRAM TECHNOLOGICZNY DO KONFIGURACJI SONDY.

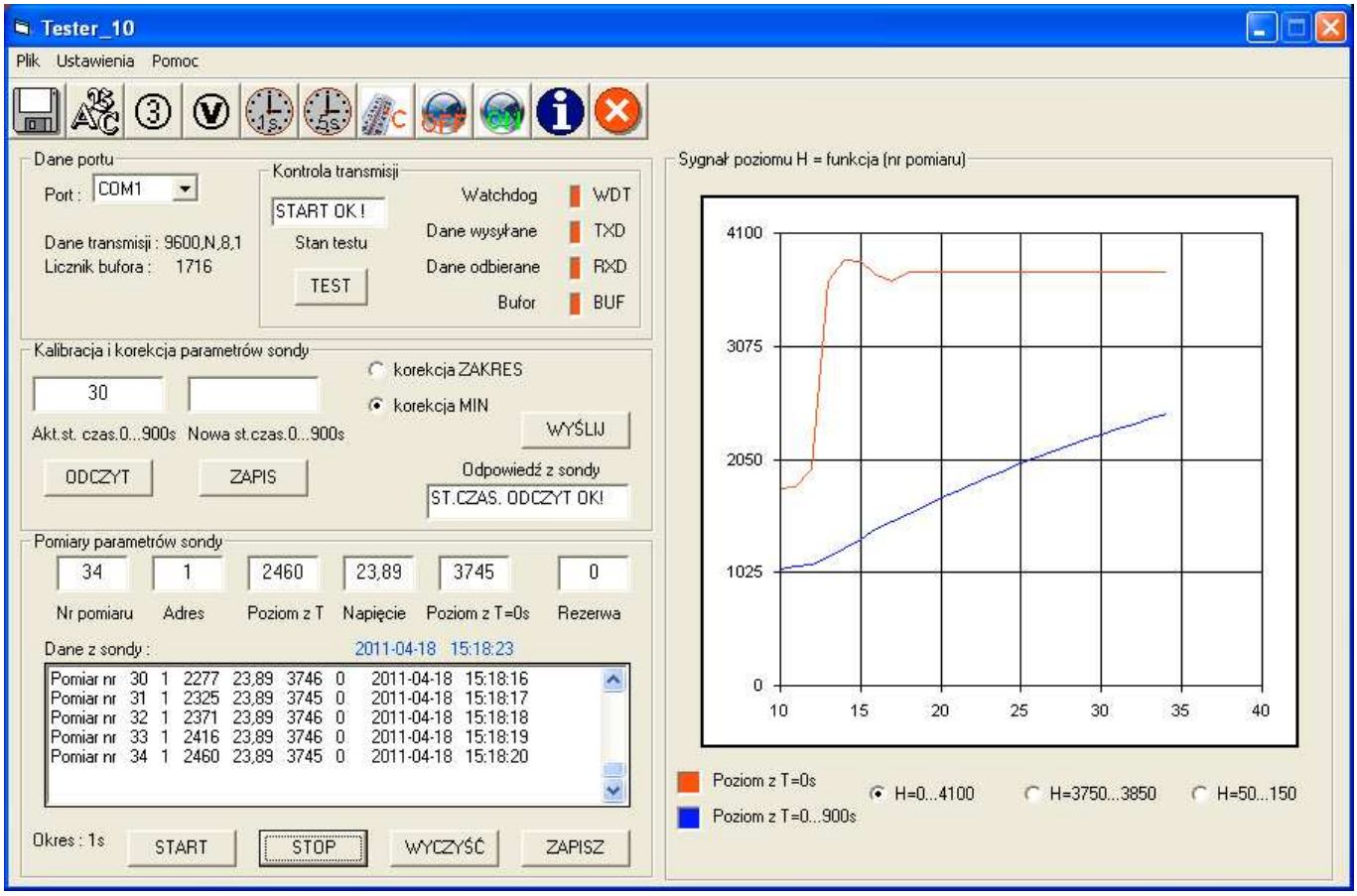
Do konfiguracji sondy służy program technologiczny „Tester_nr_wersji" (nr_wersji zależny od wymagań w zamówieniu). Program ten pozwala na konfigurację sondy w całym zakresie. Ponadto prezentuje przebieg sygnału poziomu dla wartości filtrowanej i niefiltrowanej.

- Główne okno programu „Tester_4.2.1" stosowanego dla interfejsu RS-485.



Uwaga : Program „Tester_4.2.1 pozwala na sprawdzanie pojedynczej sondy np. o adresie 1 (dopuszczalny zakres adresów : 1...65534) oraz maksymalnie 10 sond o różnych adresach (przy współpracy z programem adresy sond muszą być przyjęte w zakresie od 1 do 10).

- Główne okno programu „Tester_10” stosowanego dla interfejsu RS-232.



Uwaga : Program „Tester_10 pozwala na sprawdzanie sondy podobnym protokołem jak przy interfejsie RS-485, w którym przyjęto jednak w polu adresowym DEVID stałą wartość np. adres 1 (w ramce „Pomiary parametrów sondy, w polu „Adres”).