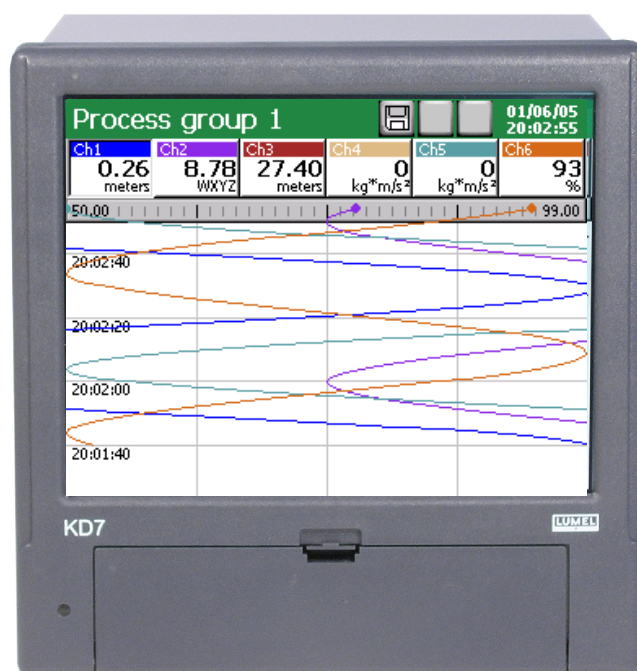


REJESTRATOR Z EKRADEM DOTYKOWYM TYPU KD7



Instrukcja obsługi protokołu transmisji „MODBUS”

SPIS TREŚCI

1. PRZEZNACZENIE.....	3
2. OPIS PROTOKOŁU MODBUS.....	3
2.1 RAMKA W TRYBIE ASCII	4
2.2 RAMKA W TRYBIE RTU	4
2.3 CHARAKTERYSTYKA PÓL RAMKI	4
2.4 WYZNACZENIE LRC	5
2.5 WYZNACZENIE CRC.....	5
2.6 FORMAT ZNAKU PRZY TRANSMISJI SZEREGOWEJ	5
2.7 PRZERWANIE TRANSAKCJI.....	5
3. OPIS FUNKCJI.....	6
3.1 ODCZYT N-REJESTRÓW (KOD 03).....	6
3.2 ZAPIS WARTOŚCI DO REJESTRU (KOD 06)	6
3.3 ZAPIS DO N-REJESTRÓW (KOD 16).....	7
3.4 RAPORT IDENTYFIKUJĄCY URZĄDZENIE (KOD 17) ŻĄDANIE:.....	7
4. KODY BŁĘDÓW	7
5. TABLICA REJESTRÓW	9
DODATEK A. OBLICZANIE SUMY KONTROLNEJ.....	17

1. PRZEZNACZENIE

Aby uzyskać wymianę informacji, przy wykorzystaniu łącza szeregowego, należy wybrać typ interfejsu i ustalić sposób interpretacji przesyłanych danych. Typ interfejsu definiuje jedynie parametry elektryczne transmisji i sposób łączenia urządzeń. Od interpretacji danych zależą takie cechy jak możliwość obsługi wielu urządzeń, sprawdzanie poprawności transmisji oraz zasady dostępu do urządzenia. Zadaniem protokołu jest określenie jaki typ danych jest interpretowany (dozwolony) i w jaki sposób są one interpretowane.

Na łączu szeregowym rejestratora KD7 został zaimplementowany asynchroniczny znakowy protokół komunikacyjny MODBUS. Konfiguracja parametrów łącza szeregowego RS 485 została opisana w instrukcji obsługi rejestratora KD7.

Zestawienie parametrów łącza szeregowego rejestratora KD7:

- adres rejestratora - 1..247
- prędkość transmisji - 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200 bit/s,
- tryby pracy - ASCII, RTU,
- jednostka informacyjna - ASCII: 8N1, 7N2, 7E1, 7O1; RTU: 8N2, 8N1, 8E1, 8O1,
- maksymalny czas odpowiedzi - 100 ms (w przypadku zapisu parametrów 400ms).

2. OPIS PROTOKOŁU MODBUS

Protokół MODBUS jest standardem przyjętym przez producentów sterowników przemysłowych dla asynchronicznej, znakowej wymiany informacji pomiędzy urządzeniami systemów pomiarowo kontrolnych. Posiada on takie cechy jak:

- prosta reguła dostępu do łącza oparta na zasadzie "master-slave",
- zabezpieczenie przesyłanych komunikatów przed błędami,
- potwierdzenie wykonywania rozkazów zdalnych i sygnalizacja błędów,
- skuteczne mechanizmy zabezpieczające przed zawieszeniem systemu,
- wykorzystanie asynchronicznej transmisji znakowej.

Kontrolery urządzeń pracujących w systemie **MODBUS** komunikują się ze sobą przy wykorzystaniu protokołu typu **master-slave**, w którym tylko jedno urządzenie może inicjalizować transakcje (jednostka nadrzędna-**master**), a pozostałe (jednostki podrzędne-**slave**) odpowiadają jedynie na zdalne zapytania jednostki nadrzędnej.

Transakcja składa się z polecenia wysyłanego z jednostki **master** do **slave** oraz z odpowiedzi przesyłanej w odwrotnym kierunku. Odpowiedź zawiera dane żądane przez **master** lub potwierdzenie realizacji jego polecenia.

Master może przysłać informację do pojedynczych odbiorców lub informację rozgłoszeniową (broadcast), przeznaczone dla wszystkich urządzeń podrzędnych w systemie (na polecenia rozgłoszeniowe master nie otrzymuje odpowiedzi).

Format przesyłanych informacji jest następujący:

- **master** → **slave** : adres odbiorcy, kod reprezentujący żądane polecenie, dane, słowo kontrolne zabezpieczające przesyłaną wiadomość,
- **slave** → **master** : adres nadawcy, potwierdzenie realizacji rozkazu, dane żądane przez master, słowo kontrolne zabezpieczające odpowiedź przed błędami.

Jeżeli urządzenie **slave** wykryje błąd przy odbiorze wiadomości, lub nie może wykonać polecenia, przygotowuje specjalny komunikat o wystąpieniu błędu i przesyła go jako odpowiedź do **mastera**.

Urządzenia pracujące w protokole **MODBUS** mogą być ustawione na komunikację przy użyciu jednego z dwóch trybów transmisji: **ASCII** lub **RTU**. Użytkownik wybiera żądany tryb wraz z parametrami portu szeregowego (prędkość transmisji, jednostka informacyjna), podczas konfiguracji każdego urządzenia.

W systemie **MODBUS** przesyłane wiadomości są zorganizowane w ramki o określonym początku i końcu. Pozwala to urządzeniu odbierającemu na odrzucenie ramek niekompletnych i sygnalizację związanych z tym błędów.

Ze względu na możliwość pracy w jednym z dwóch różnych trybów transmisji (**ASCII** lub **RTU**), definiuje się dwie ramki.

2.1 Ramka w trybie ASCII

W trybie ASCII każdy bajt wiadomości przesyłany jest w postaci dwóch znaków ASCII. Podstawową zaletą tego trybu jest to, iż pozwala on na długie odstępy między znakami (do 1s) bez powodowania błędów.

Format ramki przedstawiono poniżej:

Znacznik początku	Adres	Funkcja	Dane	Kontrola LRC	Znacznik końca
1 znak '\u0002'	2 znaki	2 znaki	n znaków	2 znaki	2 znaki CR LF

Znacznikiem początku jest znak dwukropka (":"- ASCII 3Ah), natomiast znacznikiem końca dwa znaki CR i LF. Część informacyjną ramki zabezpiecza się kodem LRC (Longitudinal Redundancy Check).

2.2 Ramka w trybie RTU

W trybie RTU wiadomości rozpoczynają i kończą się odstępem trwającym minimum 3.5 x (czas trwania pojedynczego znaku), w którym panuje cisza na łączu. Najprostszą implementacją wymienionego interwału czasowego jest wielokrotne odmierzenie czasu trwania znaku przy zadanej szybkości bodowej przyjętej na łączu.

Format ramki przedstawiono poniżej:

Znacznik początku	Adres	Funkcja	Dane	Kontrola CRC	Znacznik końca
T1-T2-T3-T4	8 bitów	8 bitów	n x 8 bitów	16 bitów	T1-T2-T3-T4

Znaczniki początku i końca zaznaczono symbolicznie jako odstęp równy czterem długościom znaku (jednostki informacyjnej). Słowo kontrolne jest 16 bitowe i powstaje jako rezultat obliczenia CRC (Cyclical Redundancy Check) na zawartości ramki.

2.3 Charakterystyka pól ramki

Pole adresowe

Pole adresowe w ramce zawiera dwa znaki (w trybie ASCII) lub osiem bitów (w trybie RTU). Zakres adresów jednostek slave wynosi 0 - 247. Master adresuje jednostki slave umieszczając jej adres na polu adresowym ramki. Kiedy jednostka slave wysłała odpowiedź, umieszcza swój własny adres na polu adresowym ramki, co pozwala masterowi sprawdzić, z którą jednostką realizowana jest transakcja.

Adres 0 jest wykorzystywany jako adres rozgłoszeniowy, rozpoznawany przez wszystkie jednostki slave podłączone do magistrali.

Pole funkcji

Pole funkcji zawiera dwa znaki w trybie ASCII lub 8-bitów w trybie RTU. Zakres kodów funkcji od 1 - 255.

Przy transmisji polecenia z jednostki master do slave, pole funkcji zawiera kod rozkazu, określający działanie, które ma podjąć jednostka slave na żądanie mastera.

Kiedy jednostka slave odpowiada masterowi, pole funkcji wykorzystuje do potwierdzenia wykonania polecenia lub sygnalizacji błędu, jeżeli z jakichś przyczyn nie może wykonać polecenia. Potwierdzenie pozytywne realizowane jest poprzez umieszczenie na polu funkcji kodu wykonanego rozkazu.

W przypadku stwierdzenia błędu, jednostka slave umieszcza na polu funkcji szczególną odpowiedź, którą stanowi kod funkcji z ustawionym na 1 najstarszym bitem.

Kod błędu umieszczany jest na polu danych ramki odpowiedzi

Pole danych

Pole danych tworzy zestaw dwucyfrowych liczb heksadecymalnych, o zakresie 00-FF. Liczby te przy transmisji w trybie ASCII reprezentowane są dwoma znakami, a przy transmisji w trybie RTU jednym. Pole danych ramki polecenia zawiera dodatkowe informacje potrzebne jednostce slave do wykonania rozkazu określonego kodem funkcji. Mogą to być adresy rejestrów, liczba bajtów w polu danych, dane itp.

W niektórych ramkach pole danych może posiadać zerową długość. Tak jest zawsze, gdy operacja określona kodem nie wymaga żadnych parametrów.

Pole kontrolne

W protokole MODBUS słowo kontrolne zabezpieczające część informacyjną zależy od zastosowanego trybu transmisji.

W trybie ASCII pole kontrolne składa się z dwóch znaków ASCII, które są rezultatem obliczenia Longitudinal Redundancy Check (LRC) na zawartości części informacyjnej ramki (bez znaczników początku i końca). Znaki LRC są dołączane do wiadomości jako ostatnie pole ramki, bezpośrednio przed znacznikiem końca (CR,LF).

W trybie RTU słowo kontrolne jest 16-bitowe i powstaje jako rezultat obliczenia Cyclical Redundancy Check (CRC) na zawartości ramki. Pole kontrolne zajmuje dwa bajty dołączane na końcu ramki. Jako pierwszy przesyłany jest mniej znaczący bajt, jako ostatni starszy bajt, który jest jednocześnie znakiem kończącym ramkę.

2.4 Wyznaczenie LRC

Obliczanie LRC polega na sumowaniu kolejnych 8-bitowych bajtów wiadomości, odrzuceniu przeniesień i na koniec wyznaczeniu uzupełnienia dwójkowego wyniku. Sumowanie obejmuje całą wiadomość za wyjątkiem znaczników początku i końca ramki. Wartość 8-bitowa sumy LRC jest umieszczana na końcu ramki w postaci dwóch znaków ASCII, najpierw znak zawierający starszą tetradę, a za nim znak zawierający młodszą tetradę LRC.

2.5 Wyznaczenie CRC

Obliczanie CRC realizowane jest według następującego algorytmu:

1. Załadowanie FFFFh do 16-bitowego rejestru CRC.
2. Pobranie bajtu z bloku danych i wykonanie operacji EXOR z młodszym bajtem rejestru CRC. Umieszczenie rezultatu w rejestrze CRC.
3. Przesunięcie zawartości rejestru CRC w prawo o jeden bit połączone z wpisaniem 0 na najbardziej znaczący bit (MSB=0).
4. Sprawdzenie stanu najmłodszego bitu (LSB) wysuniętego z rejestru CRC w poprzednim kroku. 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 A001h.
5. Powtórzenie kroków 3 i 4 osiem razy, co odpowiada przetworzeniu całego bajtu.
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.
8. Wartość CRC jest umieszczana na końcu ramki najpierw mniej znaczący bajt, a za nim bardziej znaczący bajt.

2.6 Format znaku przy transmisji szeregowej

W protokole MODBUS znaki są przesyłane od najmłodszego do najstarszego bitu.

Organizacja jednostki informacyjnej w trybie ASCII:

- 1 bit startu,
- 7 bitów pola danych,
- 1 bit kontroli parzystości (nieparzystości) lub brak bitu kontroli parzystości,
- 1 bit stopu przy kontroli parzystości lub 2 bity stopu przy braku kontroli parzystości

Organizacja jednostki informacyjnej w trybie RTU:

- 1 bit startu,
- 8 bitów pola danych,
- 1 bit kontroli parzystości (nieparzystości) lub brak bitu kontroli parzystości,
- 1 bit stopu przy kontroli parzystości lub 2 bity stopu przy braku kontroli parzystości.

2.7 Przerwanie transakcji

W jednostce master użytkownik ustawia ważny parametr jakim jest "maksymalny czas odpowiedzi na ramkę zapytania", po którego przekroczeniu transakcja jest przerywana. Czas ten dobiera się tak, aby każda jednostka slave pracująca w systemie (nawet ta najwolniejsza) zdążyła normalnie odpowiedzieć na ramkę zapytania. Przekroczenie tego czasu świadczy zatem o błędzie i tak jest traktowane przez jednostkę master.

Jeżeli jednostka slave wykryje błąd transmisji, nie wykonuje polecenia oraz nie wysyła żadnej odpowiedzi. Spowoduje to przekroczenie czasu oczekiwania na ramkę odpowiedzi i przerwanie transakcji.

3. OPIS FUNKCJI

W rejestratorze KD7 zaimplementowane zostały następujące funkcje protokołu:

kod	znaczenie
03	odczyt n-rejestrów
06	zapis pojedynczego rejestru
16	zapis n-rejestrów
17	identyfikacja urządzenia slave

3.1 Odczyt n-rejestrów (kod 03)

Żądanie:

Funkcja umożliwi odczyt wartości zawartych w rejestrach w zaadresowanym urządzeniu slave. **Rejestry są 16 lub 32-bitowymi jednostkami, które mogą zawierać wartości numeryczne związane ze zmiennymi procesowymi itp.**

Ramka żądania określa 16-bitowy adres początkowy rejestru oraz liczbę rejestrów do odczytania. Znaczenie zawartości rejestrów o danych adresach może być różne dla różnych typów urządzeń.

Funkcja nie jest dostępna w trybie rozgłoszeniowym.

Przykład. Odczyt 3 rejestrów zaczynając od rejestru o adresie 6Bh

adres	funkcja	adres rejestru Hi	adres rejestru Lo	liczba rejestrów Hi	liczba rejestrów Lo	suma kontrolna
11	03	00	6B	00	03	7E

LRC

Odpowiedź:

Dane rejestrów są pakowane począwszy od najmniejszego adresu: najpierw starszy bajt, potem młodszy bajt rejestru.

Przykład. Ramka odpowiedzi

adres	funkcja	liczba bajtów	wart. w rej.107 Hi	wart. w rej.107 Lo	wart. w rej.108 Hi	wart. w rej.108 Lo	wart. w rej.109 Hi	wart. w rej.109 Lo	suma kontrolna
11	03	06	02	2B	00	00	00	64	55

LRC

3.2 Zapis wartości do rejestru (kod 06)

Żądanie

Funkcja umożliwia modyfikację zawartości rejestru. Jest dostępna w trybie rozgłoszeniowym.

Przykład.

adres	funkcja	adres rejestru Hi	adres rejestru Lo	wartość Hi	wartość Lo	suma kontrolna
11	06	00	87	03	9E	C1

LRC

Odpowiedź:

Prawidłową odpowiedzią na żądanie zapisu wartości do rejestru jest retransmisja komunikatu po wykonaniu operacji.

Przykład.

adres	funkcja	adres rejestru Hi	adres rejestru Lo	wartość Hi	wartość Lo	suma kontrolna
11	06	00	87	03	9E	C1

LRC

3.3 Zapis do n-rejestrów (kod 16)

Żądanie:

Funkcja dostępna w trybie rozgłoszeniowym. Umożliwia modyfikacje zawartości rejestrów.

Przykład. Zapis dwóch rejestrów począwszy od rejestru o adresie 136

adres	funkcja	adres rej. Hi	adres rej. Lo	liczba rej. Hi	liczba rej. Lo	liczba bajtów	dane Hi	dane Lo	dane Hi	dane Lo	suma kontrolna
11	10	00	87	00	02	04	00	0A	01	02	45

 LRC

Odpowiedź:

Prawidłowa odpowiedź zawiera adres jednostki slave, kod funkcji, adres początkowy oraz liczbę zapisanych rejestrów.

Przykład.

adres	funkcja	adres rej. Hi	adres rej. Lo	liczba rej. Hi	liczba rej. Lo	suma kontrolna
11	10	00	87	00	02	56

 LRC

3.4 Raport identyfikujący urządzenie (kod 17)

Żądanie:

Funkcja pozwala użytkownikowi uzyskać informacje o typie urządzenia, statusie i zależnej od tego konfiguracji.

Przykład.

Adres	funkcja	suma kontrolna	
11	11	CD	EC

Odpowiedź:

Pole „identyfikator urządzenia” w ramce odpowiedzi oznacza unikalny identyfikator danej klasy urządzeń, natomiast pozostałe pola zawierają parametry zależne od typu urządzenia.

Przykład dla rejestratora KD7.

Adres slave	funkcja	liczba bajtów	identyfikator urządzenia	stan urządzenia	suma kontrolna	
11	11	02	A7	FF	46	8F

4. KODY BŁĘDÓW

Gdy urządzenie master wysła żądanie do urządzenia slave, to za wyjątkiem komunikatów w trybie rozgłoszeniowym, oczekuje prawidłowej odpowiedzi. Po wysłaniu żądania jednostki master może wystąpić jedno z czterech możliwych zdarzeń:

- ▣ Jeżeli jednostka slave odbiera żądanie bez błędu transmisji oraz może je wykonać prawidłowo, wówczas zwraca prawidłową odpowiedź.
- ▣ Jeżeli jednostka slave nie odbiera żądania, żadna odpowiedź nie jest zwracana. W programie urządzenia master zostaną spełnione warunki timeout dla żądania.
- ▣ Jeżeli jednostka slave odbiera żądanie, ale z błędami transmisji (błąd parzystości, sumy kontrolnej LRC lub CRC), żadna odpowiedź nie jest zwracana. W programie urządzenia master zostaną spełnione warunki timeout dla żądania.
- ▣ Jeżeli jednostka slave odbiera żądanie bez błędu transmisji, ale nie może go wykonać prawidłowo (np. jeżeli żądaniem jest odczyt nie istniejącego wyjścia bitowego lub rejestru), wówczas zwraca odpowiedź zawierającą kod błędu, informujący urządzenie master o przyczynie błędu.

Komunikat z błędną odpowiedzią zawiera dwa pola odróżniające go od prawidłowej odpowiedzi:

1. Pole kodu funkcji:

W prawidłowej odpowiedzi, jednostka slave retransmituje kod funkcji z komunikatu żądania na polu kodu funkcji odpowiedzi. Wszystkie kody funkcji mają najbardziej znaczący bit (MSB) równy 0 (wartości kodów są poniżej 80h). W błędnej odpowiedzi urządzenie slave ustawia bit MSB kodu funkcji na 1. To powoduje, że wartość kodu funkcji w błędnej odpowiedzi jest dokładnie o 80h większa niż byłaby w prawidłowej odpowiedzi.

Na podstawie kodu funkcji z ustawionym bitem MSB program urządzenia master może rozpoznać błędną odpowiedź i może sprawdzić na polu danych kod błędu.

2. Pole danych:

W prawidłowej odpowiedzi, urządzenie slave może zwrócić dane na polu danych (pewne informacje żądane przez jednostkę master). W błędnej odpowiedzi, urządzenie slave zwraca kod błędu na polu danych. Określa on warunki urządzenia slave, które spowodowały błąd.

Poniżej przedstawiono przykład żądania urządzenia master i błędną odpowiedź urządzenia slave. Dane są w postaci heksadecymalnej.

Przykład: żądanie

adres slave	funkcja	adres zmiennej Hi	adres zmiennej Lo	liczba zmiennych Hi	liczba zmiennych Lo	suma kontrolna
0A	01	04	A1	00	01	4F

 LRC

Przykład: błędna odpowiedź

adres slave	funkcja	kod błędu	suma kontrolna
0A	81	02	73

 LRC

W tym przykładzie urządzenie master adresuje żądanie do jednostki slave o numerze 10 (0Ah). Kod funkcji (01) służy do operacji odczytu stanu wyjścia bitowego. Ta ramka oznacza więc żądanie odczytu statusu jednego wyjścia bitowego o adresie 1245 (04A1h).

Jeżeli w urządzeniu slave nie ma wyjścia bitowego o podanym adresie, wówczas urządzenie zwróci błędną odpowiedź z kodem błędu nr 02. Oznacza on niedozwolony adres danych w urządzeniu slave.

W poniższej tabeli przedstawione są możliwe kody błędów i ich znaczenie

Kod	Znaczenie
01	niedozwolona funkcja
02	niedozwolony adres danych
03	niedozwolona wartość danej
04	uszkodzenie w przyłączonym urządzeniu
05	potwierdzenie
06	zajęty, komunikat usunięty
07	negatywne potwierdzenie
08	błąd parzystości pamięci

5. Tablica rejestrów

- Identyfikator rejestratora KD7 (wysyłany w odpowiedzi na funkcję identyfikacji) : 0xA7
- Typy rejestrów (kolumna „Typ”):
 - float – liczba zmiennoprzecinkowa (patrz opis poniżej),
 - sfloat – liczba zmiennoprzecinkowa (patrz opis poniżej),
- Tryby dostępu do rejestrów:
 - RO – tylko do odczytu,
- Reprezentacja liczb zmiennoprzecinkowych (float IEEE 754)

bajt 4 bajt 3 bajt 2 bajt 1
 SEEEEEE EMMMMMMM MMMMMMMM MMMMMMMM

S – bit znaku (Sign bit)

E – wykładnik (Exponent)

M – mantysa

Bajty rejestrów typu **float** przesyłane są w kolejności 4321

Bajty rejestrów typu **sfloat** przesyłane są w kolejności 2143

DANE PROCESOWE

Wartości pomiarów w kanałach 1-32 są umieszczone w rejestrach 2x16-bitowych o adresach 7000-7062 oraz 7100-7162. Rejestry adresowane 32-bitowo są dostępne pod adresami adresach 7500 – 7531 oraz 7600-7631.

Adres	Typ	Dostęp	Nazwa	Opis
7000	float	RO	MeasureCh1	Wartość pomiaru w kanale nr 1
7002			MeasureCh2	Wartość pomiaru w kanale nr 2
7004			MeasureCh3	Wartość pomiaru w kanale nr 3
...		
7062			MeasureCh32	Wartość pomiaru w kanale nr 32
Zdublowane rejestry 7000..7062, typ sfloat				
7100	sfloat	RO	MeasureCh1	Patrz opis rejestru 7000
7102			MeasureCh2	Patrz opis rejestru 7002
7104			MeasureCh3	Patrz opis rejestru 7004
...		
7162			MeasureCh32	Patrz opis rejestru 7062
Zdublowane rejestry 7000..7062, typ float, adresowane 32 bitowo				
7500	float	RO	MeasureCh1	Patrz opis rejestru 7000
7501			MeasureCh2	Patrz opis rejestru 7002
7502			MeasureCh3	Patrz opis rejestru 7004
...		
7531			MeasureCh32	Patrz opis rejestru 7062
Zdublowane rejestry 7000..7062, typ sfloat, adresowane 32 bitowo				
7600	sfloat	RO	MeasureCh1	Patrz opis rejestru 7000
7601			MeasureCh2	Patrz opis rejestru 7002
7602			MeasureCh3	Patrz opis rejestru 7004
...		
7631			MeasureCh32	Patrz opis rejestru 7062

DODATEK A. OBLICZANIE SUMY KONTROLNEJ

W dodatku tym przedstawiono przykłady funkcji w języku C, obliczające sumę kontrolną LRC dla trybu ASCII oraz CRC dla trybu RTU.

Funkcja do obliczenia LRC ma dwa argumenty:

*unsigned char *outMsg;*

Wskaźnik do bufora komunikacyjnego, zawierającego dane binarne, z których należy obliczyć LRC

unsigned short usDataLen; Liczba bajtów w buforze komunikacyjnym

Funkcja zwraca LRC typu *unsigned char*.

static unsigned char LRC(outMsg, usDataLen)

```
unsigned char *outMsg; /* bufor do obliczenia LRC */
unsigned short usDataLen; /* liczba bajtów w buforze */
{
    unsigned char uchLRC = 0; /* inicjalizacja LRC */
    while (usDataLen--)
        uchLRC += *outMsg++; /* dodaj bajt bufora bez przeniesienia */
    return ((unsigned char)(-(char uchLRC))); /* zwraca sumę w kodzie uzupełnienia do dwóch */
}
```

Poniżej przedstawiono przykład funkcji w języku C obliczającej sumę CRC. 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 poprzez indeksowanie tablic jest o wiele szybsze niż obliczenie nowej wartości CRC dla każdego znaku z bufora komunikacyjnego.

Uwaga: Poniższa funkcja przedstawia bajty sumy CRC starszy/młodszy, tak że wartość CRC zwracana przez funkcję może być bezpośrednio umieszczona w buforze komunikacyjnym.

Funkcja do obliczenia CRC ma dwa argumenty:

*unsigned char *puchMsg;* Wskaźnik do bufora komunikacyjnego, zawierającego dane binarne, z których należy obliczyć CRC

unsigned short usDataLen; Liczba bajtów w buforze komunikacyjnym

Funkcja zwraca CRC typu *unsigned short*.

```
unsigned short CRC16(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 młodszy bajtu CRC */
    while (usDataLen--)
    {
        uIndex = uchCRChi ^ *puchMsg++; /* obliczenie CRC */
        uchCRChi = uchCRClo ^ crc_hi[uIndex];
        uchCRClo = crc_lo[uIndex];
    }
    return(uchCRChi<<8 | uchCRClo);
}
```

//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 mlodszeo 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
};

```

KONIEC

Lubuskie Zakłady Aparatów Elektrycznych LUMEL S.A.
ul. Sulechowska 1,
65-022 Zielona Góra