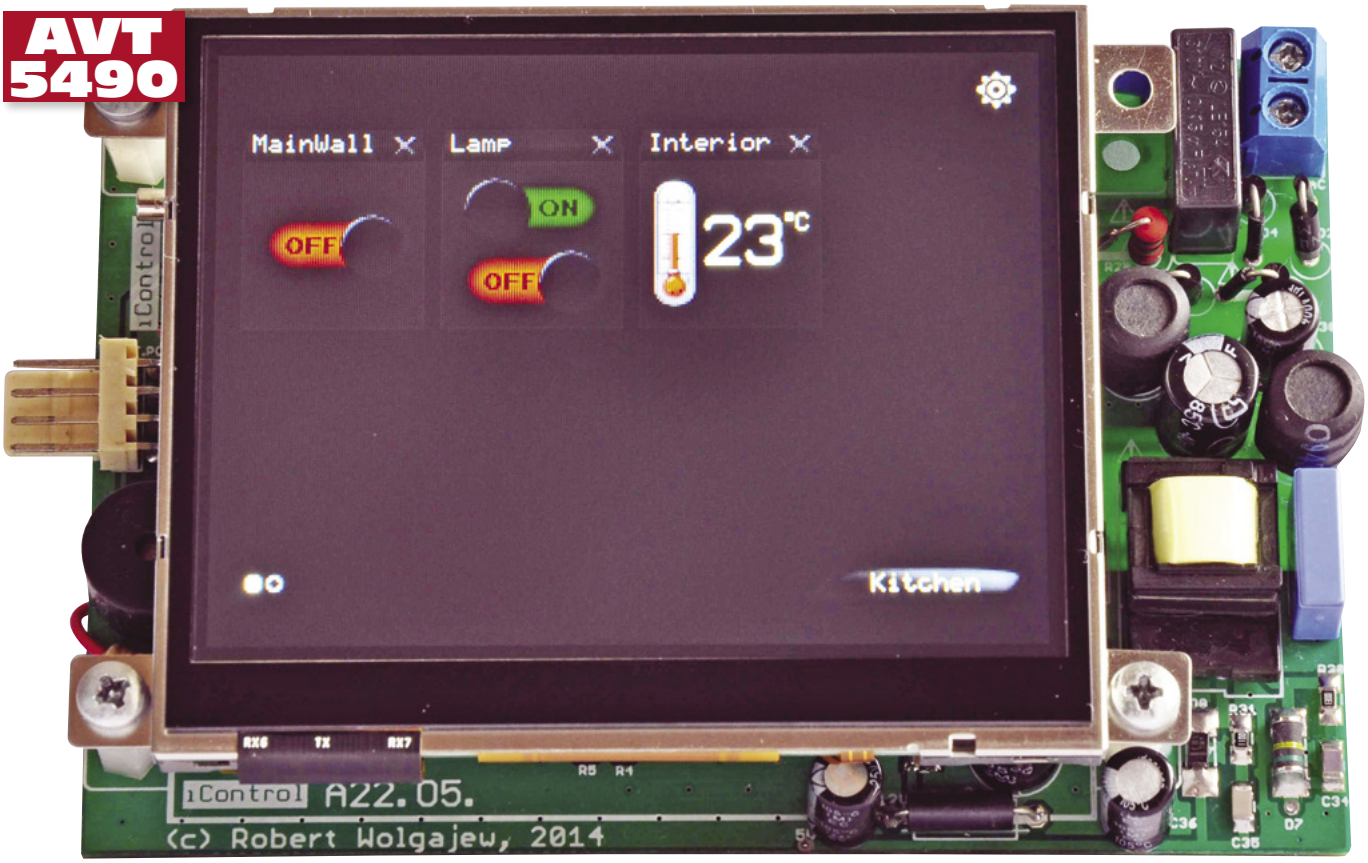
Podziękowania
Autor składa serdeczne podziękowania pani Beacie Stosik z firmy STOFL za dostarczenie wielu darmowych próbek układów CY8CPLC10, panu Sławomirowi Szweda z firmy Unisystem za dostarczenie darmowych modułów wyświetlaczy TFT oraz panu Dariuszowi Kowalczyk za cenne uwagi przekazywane w trakcie projektowania urządzenia.
Author would like to thank Mr Prem Sai V from Cypress Semiconductor customer support team for his huge and inestimable support and commitment while developing the device!

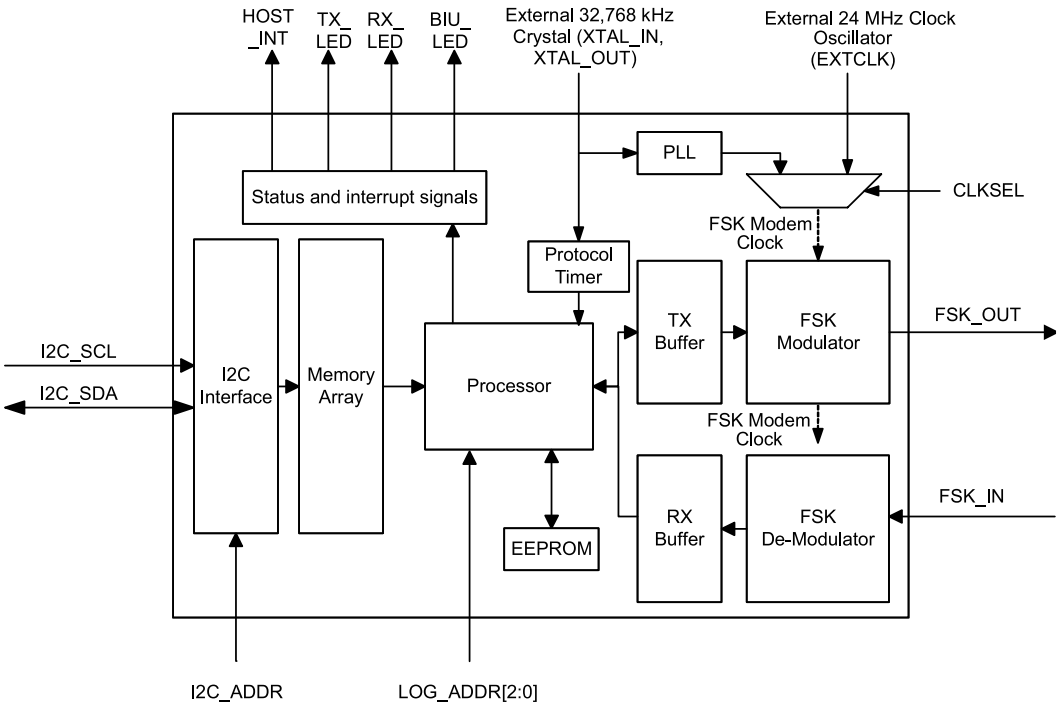


iControl
System automatyki domowej (1)

Film demonstrujący działanie systemu iControl: http://goo.gl/Q99yjy

W swojej praktyce konstruktora-programisty kilkukrotnie podejmowałem wyzwanie skonstruowania prostego systemu sterowania i kontroli typu „inteligentny dom", stosując w tym celu różne interfejsy w roli medium umożliwiającego współpracę specjalizowanych modułów. Jednym z przykładów takiego dość rozbudowanego i nowoczesnego projektu może być system intelli-Dom opublikowany na łamach Elektroniki Praktycznej 10…11/2011, który wykorzystywał specjalizowane i bardzo dobrze wyposażone moduły modemów ZigBee. System ten, mimo nie dawał jednak pełnej elastyczności w zakresie konfigurowania sieci modułów wykonawczych, gdyż korzystał z predefiniowanej funkcjonalności tychże modułów nazywanych tam „modułami pokojowymi", a poza tym używał transmisji bezprzewodowej o ograniczonym zasięgu, realizowanej w paśmie 2,4 GHz, opartej na standardzie IEEE 802.15.4. Czas więc na projekt zaawansowany, w pełni konfigurowalny i pozbawiony poprzednich ograniczeń a dodatkowo wyposażony w ultranowoczesny interfejs użytkownika.

***Rekomendacje:** projekt przyda się jako baza dla rozbudowanego systemu zarządzania inteligentnym budynkiem.*



Rysunek 1. Schemat blokowy układu CY8CPLC10

Pierwszym, naturalnym wyborem, z którym musiałem się zmierzyć, był oczywiście wybór rodzaju medium transmisyjnego. Tym razem i nie bez dłuższego namysłu, zdecydowałem się na wybór technologii znanej pod nazwą PLC (*Power Line Communication*). Jest to technologia transmisji danych oparta na przesyłaniu równolegle z napięciem zasilającym o częstotliwości 50 (lub 60) Hz sygnału o wiele wyższej częstotliwości (od kilku kiloherców do kilkudziesięciu megaherców) zawierającego dane.

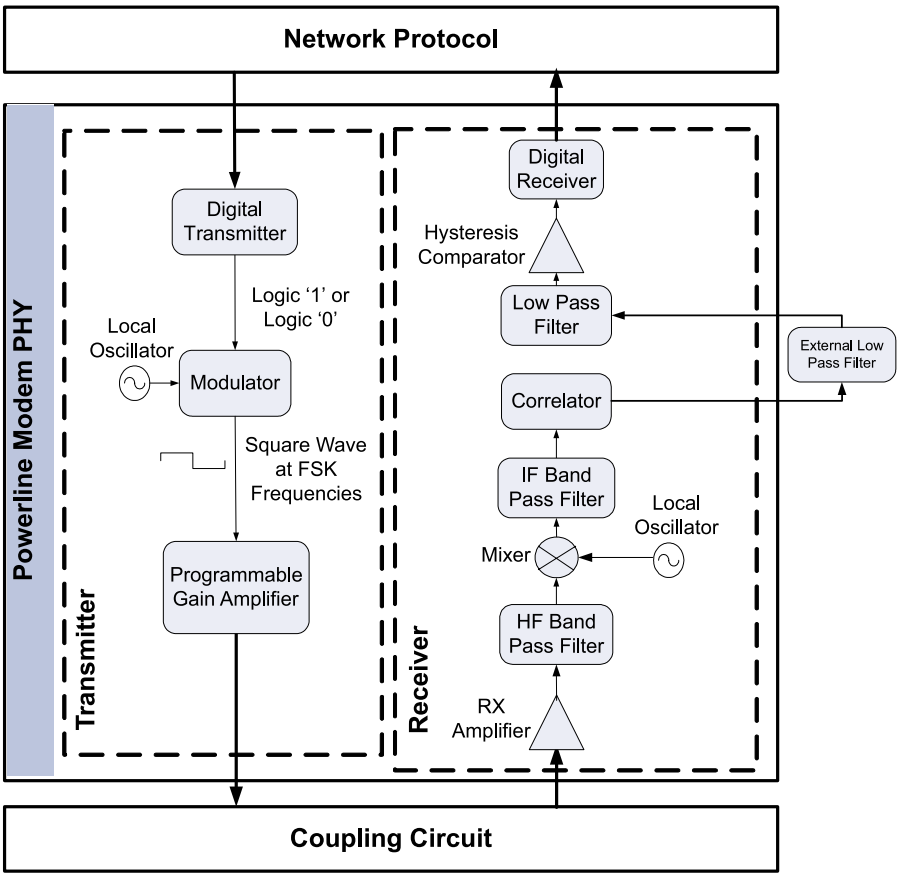
Pomysł wykorzystania medium, którym jest kabel sieci elektrycznej nie jest nowy i zrodził się już wiele lat temu, zaś jego początki związane były z potrzebą zapewnienia taniego sposobu dostępu do Internetu w miejscach, gdzie inne możliwości były niedostępne. Sieci PLC zostały podzielone na PLC wąskopasmowe (NPL – *Narrow Powerline Communication*) i szerokopasmowe (BPL – *Broadband Power Line*). Wąskopasmowe sieci PLC odniosły sukces dzięki wykorzystaniu w systemach telemetrii i inteligentnych sieciach elektroenergetycznych (tzw. Smart Grid). Współcześnie ich głównym zastosowaniem jest zdalny odczyt liczników, sterowanie ruchem ulicznym, oświetleniem itp. Szerokopasmowe sieci PLC wykorzystywano również do transmisji danych w zakresie dostępu do Internetu i w lokalnych sieciach domowych. Mimo pewnych wad tej technologii, jakimi może być zakłócone środowisko, niezbyt szybki transfer danych i problemy ze spełnieniem wymogów kompatybilności elektromagnetycznej, jest ona nadal rozwijana w wielu krajach, w tym także i w Polsce. Dużym przełomem w rozwoju szerokopasmowego PLC było ratyfikowanie w 2010 r. przez organizację IEEE standardu 1901. W najnowszych generacjach urządzeń opartych na tym standardzie wyeliminowano problemy z wolnym tempem transmisji danych, niewielkim zasięgiem oraz ograniczeniami związanymi z nieprzenikaniem sygnału pomiędzy fazami elektrycznymi w sieciach trójfazowych. Nowoczesne urządzenia wykorzystujące szerokopasmowe PLC mają prędkość fizyczną do 500 Mbit/s i zasięg aż do 500 metrów. Działają niezawodnie bez względu na stosowane fazy energii elektrycznej.

W takim razie pora na wybór rozwiązania układowego. Poszukiwania rozpocząłem od przejrzenia oferty producentów półprzewodników w zakresie gotowych układów scalonych pełniących rolę modemów PLC. Szybko okazało się, że nader często stosowanym układem pełniącym rolę interfejsu PLC jest TDA5051 firmy NXP (wcześniej Philips) będący zintegrowanym modemem PLC zapewniającym transmisję danych opartą o modulację ASK (*Amplitude Shift Keying*) z maksymalną prędkością na poziomie 1200 bitów na sekundę. Układ ten, mimo że dość chętnie stosowany (choć już „leciwy"), ma dwie podstawowe wady. Po pierwsze, zastosowana modulacja ASK jest stosunkowo podatna na różne zakłócenia występujące w sieci elektroenergetycznej (zwłaszcza zakłócenia komutacji i generowane przez odbiorniki o charakterze indukcyjnym), a po drugie, nie zaimplementowano w nim żadnego stosu komunikacyjnego zapewniającego chociażby wielodostęp do medium transmisyjnego pozostawiając to zadanie po stronie projektanta systemu. Mimo to, zacząłem projektowanie systemu oraz oprogramowywanie prostego stosu komunikacyjnego z wykorzystaniem kodowania typu Manchester jako remedium na zakłócenia sieciowe.

Problemem, z którym musiałem się zmierzyć, był wybór mechanizmu zapewniającego wielodostęp do medium transmisyjnego oraz fakt, że w rozwiązaniach wykorzystujących modulację ASK i układy z rodziny TDA5051 użyteczna transmisja danych jest przeprowadzana w krótkim oknie czasowym tuż przed i tuż po przejściu napięcia zasilającego przez zero w celu zminimalizowania wpływu zakłóceń na transmisję użytecznych danych (sposób stosowany na przykład w protokole X10 przeznaczonym do sterowania urządzeniami domowymi). Ta właściwość powoduje, że zmniejsza się użyteczna, wynikowa prędkość transmisji danych, co pogarsza właściwości użytkowe tak skonstruowanej sieci. Mimo że moje prace w tym kierunku były już dość zaawansowane, zacząłem poszukiwania lepszego rozwiązania układowego. I właśnie w tym czasie natknąłem się na doskonałe rozwiązanie firmy Cypress, którym jest układ scalony CY8CPLC10. Chip ten jest scalonym, zaawansowanym i kompletnym rozwiązaniem modemu PLC wykorzystującym modulację FSK (*Frequency Shift Keying*), mającym zaimplementowany stos komunikacyjny.

Układ CY8CPLC10 charakteryzuje się następującymi, wybranymi cechami użytkowymi:

- Zintegrowany interfejs fizyczny modemu PLC (*Physical Layer Interface*).
- Modulacja FSK zapewniająca prędkość transmisji na poziomie 2400 bitów na sekundę.
- Kompletny protokół komunikacyjny zoptymalizowany dla sieci PLC zapewniający realizację wszystkich warstw komunikacji (*Data Link, Transport i Network Layers*).
- Komunikacja dwukierunkowa typu half duplex z korekcją błędów (8 bitowe CRC) i sygnałem potwierdzenia (ACK).
- Wbudowany mechanizm wielodostępu do medium komunikacyjnego (CSMA - *Carrier Sense Multiple Access*).
- Zintegrowany interfejs sterujący I²C pracujący w trybie high-speed (400 kHz).
- Wsparcie dla sieci zmiennoprądowych AC 110…240 V i stałoprądowych DC 12…24 V.
- Wsparcie dla wielu trybów transmisji danych w sieci: master-slave, peer-to-peer i multimaster.
- 3 tryby adresowania urządzeń sieciowych: logiczny (8-bitów), rozszerzony logiczny (16-bitów) i fizyczny (64-bity).
- 2 tryby rozgłaszania adresów urządzeń: indywidualny i grupowy (możliwość przyporządkowania urządzeń do jednej z 256 grup i sterowania grupowego).
- Szereg kilkudziesięciu rejestrów kontrolnych/sterujących dających pełną kontrolę nad transmisją danych.
- Wbudowany detektor BIU (*Band-In-Use*) zgodny ze standardem CENELEC EN 50065-1.



Rysunek 2. Schemat funkcjonalny wbudowanego w układ CY8CPLC10 modemu PLC

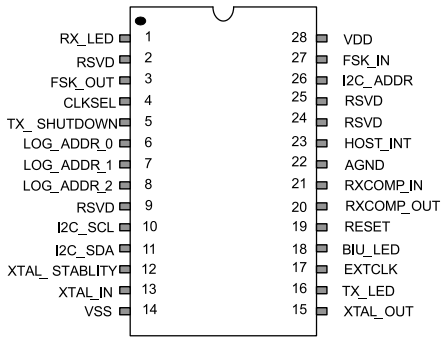
- Zgodność ze standardem CENELEC EN50065-1:2001 i FCC part 15 (dla Ameryki Północnej).

Już z pobieżnej analizy długiej listy możliwości widać, że układ idealnie wpisuje się w potrzeby bieżącego projektu zwalniając nas jednocześnie z konieczności rozwiązania wielu problemów technicznych wynikających z konstrukcji stosu komunikacyjnego i mechanizmów dostępu do medium transmisyjnego.

Na **rysunku 1** pokazano diagram obrazujący schemat blokowy układu CY8CPLC10, natomiast na **rysunku 2** diagram obrazujący schemat funkcjonalny wbudowanego modemu PLC.

Warto na chwilę zatrzymać się nad schematem funkcjonalnym modemu PLC, aby poznać sposób nadawania i odbioru pakietów danych. W konstrukcji nadajnika modemu PLC wykorzystano modulator, który „zasilany" kolejnymi bitami przesyłanych danych generuje sygnał wyjściowy o dwóch możliwych częstotliwościach, zależnych od wartości aktualnie przetwarzanego bitu danych. Dla bitu równego „0" jest to częstotliwość 133,3 kHz, zaś dla bitu o wartości „1" – 131,8 kHz (lub 130,4 kHz – w zależności od ustawień konfiguracyjnych). Wyjście modulatora jest połączone ze wzmacniaczem sygnału o regulowanym programowo wzmocnieniu, a następnie wyprowadzone na zewnątrz układu (pin FSK_OUT) dostarczając w ten sposób sygnał o modulacji FSK.

W części odbiorczej mamy do czynienia z dużo bardziej rozbudowanym układem. Zmodulowany, sieciowy sygnał wejściowy (pin FSK_IN) jest wprowadzany do wewnętrznego filtru pasmowoprzepustowego, który zawęża pasmo sygnału użytecznego do 125 kHz…140 kHz, po czym ten sygnał „zasila" demodulator zbudowany z prostego miksera, lokalnego generatora częstotliwości wzorcowej, filtra częstotliwości pośredniej i korelatora. Sygnał wyjściowy korelatora, który zawiera już zdemodulowany przebieg wyjściowy odpowiadający przesyłanym danym z komponentami o wyższych częstotliwościach, wchodzi na wejście zewnętrznego filtra dolnoprzepustowego (piny RXCOMP_IN i RXCOMP_OUT) o częstotliwości odcięcia równej 7,5 kHz, a następnie na wejście kolejnego, wewnętrznego filtra dolnoprzepustowego, który tłumi wszystkie wyższe składowe częstotliwości generowane w procesie korelacji. Sygnał z wyjścia tego filtra przechodzi jeszcze przez układ komparatora z histerezą, który służy eliminowania potencjalnych zakłóceń i korygowania opóźnienia wnoszonego przez korelator. Wyjście z komparatora „zasila" rejestr przesuwny, który zamienia strumień szeregowy na wyjściowe pakiety danych poddawane następnie analizie przez mechanizmy warstwy sieciowej stosu komunikacyjnego. Rozmieszczenie wyprowadzeń układu CY8CPLC10 pokazano na **rysunku 3** zaś ich opis funkcjonalny umieszczono w **tabeli 1**.



Rysunek 3. Rozmieszczenie wyprowadzeń układu CY8CPLC10

Można zauważyć jak wiele komponentów warstwy fizycznej jest zaangażowanych w proces dekodowania sygnału PLC, który w dalszej kolejności trafia do części logicznej, odpowiedzialnej za właściwą interpretację danych, ich integralność oraz odpowiednią synchronizację.

Warstwa sieciowa korzysta z ciekawej konstrukcji ramki pakietu danych, której to wygląd pokazano na **rysunku 4**. Można zauważyć, że struktura ramki składa się z nagłówka o zmiennej długości (6 do 20 bajtów) zależnej od typu przesyłanych adresów urządzenia źródłowego (***SA Type***) i urządzenia docelowego (***DA Type***), dołączonych danych o długości od 0 do 31 bajtów oraz sumy kontrolnej CRC8. Pierwszy bajt nagłówka (0x00) zawiera następujące informacje, determinujące logiczny podział całego pakietu danych:

- Pole ***SA Type*** (bit 7) określa rodzaj przesyłanego adresu urządzenia źródłowego (wysyłającego pakiet danych) według następującej specyfikacji: 0 → adres logiczny (8-bitów), 1 → adres fizyczny (unikalne 64-bity zdefiniowane przez producenta układu).
- Pole ***DA Type*** (bity 6..5) określa rodzaj przesyłanego adresu urządzenia docelowego (będącego adresatem przesyłanego pakietu danych) według następującej specyfikacji: 0 →adres logiczny (8-bitów), 1 → adres grupowy (8-bitów), 2 → adres fizyczny (unikalne 64-bity zdefiniowane przez producenta układu).
- Pole ***Service Type*** (bit 4) określa konieczność (jeśli ustawione) potwierdzania wysyłanych pakietów danych determinując generowanie sygnału ACK przez strony transmisji.
- Pole ***Response*** (bit 1) determinuje czy przesyłany pakiet jest odpowiedzią na żądanie przesłania danych (jeśli ustawione) czy też „zwykłą" transmisją danych.

Znaczenie kolejnych pól danych pozostałych bajtów pakietu przedstawia się następująco:

- Bajt(y) ***Destination Address*** (0x01) określa(ją) adres urządzenia docelowego. Ponieważ dopuszczalne są 3 sposoby adresowania różniące się długością (8 lub 64 bity), offset kolejnych bajów w zakresie nagłówka pakietu danych może podlegać zmianie.
- Bajt(y) ***Source Address*** (0x02) określa(ją) adres urządzenia źródłowego. Ponieważ dopuszczalne są 3 sposoby adresowania różniące się długością (8 lub 64 bity), offset kolejnych bajów w zakresie nagłówka pakietu danych może podlegać zmianie.
- Bajt ***Command*** (0x03) określa rodzaj przesyłanego rozkazu w ramach predefiniowanych rozkazów stosu komunikacyjnego, które to mogę powodować podjęcie różnych akcji automatycznych lub też powodować zmiany konfiguracji urządzenia docelowego, jak też w ramach zdefiniowanych przez użytkownika własnych rozkazów sterujących właściwych dla implementowanej aplikacji.

Tabela 1. Opis wyprowadzeń układu CY8CPLC10

Nr	Nazwa	Opis
1	RX_LED	Opcjonalna dioda LED sygnalizująca odbiór danych PLC
2		
3	FSK_OUT	Analogowe wyjście sygnału PLC z modulacją FSK
4	CLKSEL	Wybór źródła sygnału taktującego układ: 0→zewnętrzny sygnał zegarowy o częstotliwości 24 MHz doprowadzony do wejścia EXTCLK, 1→rezonator kwarcowy 32768 Hz podłączony do pinów XTAL_IN i XTAL_OUT
5	TX_SHUTDOWN	Wyjście służące do blokowania zewnętrznego układu transmisji (wzmacniacza) podczas odbioru danych: 0→gdy modem transmituje dane, 1→gdy modem nie transmituje danych.
6	LOG_ADDR_0	Bit LA0 adresu logicznego urządzenia, w przypadku, gdy adres ten nie został ustawiony programowo przez kontroler-host (wejście zanegowane).
7	LOG_ADDR_1	Bit LA1 j.w.
8	LOG_ADDR_2	Bit LA2 j.w.
9		
10	I2C_SCL	Sygnał zegarowy magistrali I²C
11	I2C_SDA	Sygnał danych magistrali I²C
12	XTAL_STABILITY	Stabilizacja oscylatora zegara taktującego. Między to wyprowadzenie a VSS należy podłączyć kondensator 100 nF
13	XTAL_IN	Wejście do podłączenia rezonatora kwarcowego 32768 Hz
14	VSS	Masa zasilania
15	XTAL_OUT	Wyjście do podłączenia rezonatora kwarcowego 32768 Hz
16	TX_LED	Opcjonalna dioda LED sygnalizująca transmisję danych PLC
17	EXTCLK	Opcjonalny, zewnętrzny sygnał zegarowy o częstotliwości 24 MHz
18	BIU_LED	Opcjonalna dioda LED sygnalizująca zajętość magistrali PLC
19	RESET	Zerowanie układu
20	RXCOMP_OUT	Wyjście analogowe do podłączenia zewnętrznego filtra dolnoprzepustowego (RC) 7.5 kHz
21	RXCOMP_IN	Wejście analogowe do podłączenia zewnętrznego filtra dolnoprzepustowego (RC) 7.5 kHz
22	AGND	Masa zasilania części analogowej. Między to wyprowadzenie a VSS należy podłączyć kondensator 1 μF
23	HOST_INT	Wyjście zgłoszenia przerwania do kontrolera-hosta. Rodzaj zdarzeń generujących przerwanie i polaryzacja tego wyjścia podlega konfiguracji użytkownika.
24		
25		
26	I2C_ADDR	Wejście wyboru 7-bitowego adresu układu na magistrali I²C: 0→adres 0x7A, 1→adres 0x01
27	FSK_IN	Analogowe wejście sygnału PLC z modulacją FSK
28	VDD	Napięcie zasilania (5 V)

Byte Offset	Bit Offset							
	7	6	5	4	3	2	1	0
0x00	SA Type	DA Type		Service Type			Response	
0x01	Destination Address (8-bit Logical, 16-bit Extended Logical or 64-bit Physical)							
0x02	Source Address (8-bit Logical, 16-bit Extended Logical or 64-bit Physical)							
0x03	Command							
0x04				Payload Length				
0x05	Seq Num				Powerline Packet Header CRC			
0x06	Payload (0 to 31 Bytes)							
	Powerline Transceiver Packet CRC							

Rysunek 4. Budowa ramki pakietu danych układu CY8CPLC10

- Pole ***Payload Length*** (bity 4…0 bajtu 0x04) określa długość (w bajtach) pakietu dołączonych danych.
- Pole ***Seq Num*** (bity 7…4 bajtu 0x05) jest używane przez stos komunikacyjny w wypadku retransmisji pakietu danych, na skutek błędów transmisji. Pole to podlega inkrementacji za każdym razem, gdy przesyłany jest zupełnie nowy pakiet danych, natomiast pozostaje bez zmian w przypadku retransmisji pakietu bieżącego. Dopuszczalna liczba retransmisji podlega konfiguracji ze strony użytkownika. Dzięki takiej konstrukcji stosu komunikacyjnego unika się duplikacji odbieranych danych po stronie kontrolera-hosta w przypadkach ich wielokrotnych retransmisji.
- Pole ***Powerline Packet Header CRC*** (bity 3…0 bajtu 0x05) zawiera sumę kontrolną nagłówka pakietu danych (wykorzystywaną przez mechanizmy stosu komunikacyjnego).
- Bajty ***Payload*** to użyteczne dane przesyłanego pakietu danych (dane użytkownika).
- Bajt ***Powerline Transceiver Packet CRC*** zawiera sumę kontrolną CRC8 całego pakietu danych, w tym nagłówka i danych użytecznych (wykorzystywaną przez mechanizmy stosu komunikacyjnego).

Firma Cypress zastosowała bardzo przemyślaną strukturę pakietu danych, co zaowocowało dużą elastycznością funkcjonalną układu zastosowanego w tym projekcie, otwierając nowe możliwości stosu komunikacyjnego. Kilka słów uwagi należy w tym miejscu poświęcić sposobom adresowania urządzeń w tak skonstruowanej sieci PLC.

Każdy układ CY8CPLC10 ma stały, nadany przez producenta w procesie produkcji, unikalny, 64-bitowy adres fizyczny (wzorem adresu MAC), który jednoznacznie identyfikuje go w sieci. Aby jednak zmniejszyć ruch w sieci spowodowany wysyłaniem dłuższych pakietów danych, które zawierają 64-bitowe adresy urządzenia źródłowego i docelowego, możliwe jest nadanie każdemu z modemów unikalnego, 8-bitowego adresu logicznego, co jednocześnie powoduje, że maksymalna liczba urządzeń, które mogą współpracować w tak skonfigurowanej sieci wyniesie wtedy 256. Jako opcjonalny rodzaj adresu, jeśli wymagana przez nas liczba urządzeń przekracza wartość 256, jest możliwe ustawienie rozszerzonego, 16-bitowego adresu logicznego, co powoduje zwiększenie liczby możliwych urządzeń do 65536. Jakby tego było mało, dostępne są 3 sposoby ustawienia 8-bitowego adresu logicznego:

- Przez kontrolera-hosta przy udziale magistrali I²C.
- Poprzez odpowiednie skonfigurowanie wyprowadzeń układu oznaczonych LOG_ADDR_2… LOG_ADDR_0 (jednak wtedy dostępnych jest wyłącznie 8 adresów).
- Zdalnie, za pomocą innego modemu PLC korzystającego z predefiniowanych rozkazów stosu PLC pod warunkiem, że **modem odbierający tego rodzaju rozkaz jest skonfigurowany w taki sposób by nie ignorował go!**

Oprócz możliwości adresowania indywidulanych urządzeń w implementacji stosu komunikacyjnego przewidziano możliwość pracy rozgłoszeniowej, to znaczy możliwość wysyłania pakietów danych do wielu urządzeń docelowych w jednym czasie (tzw. *multicast messages*). Jakby tego było mało, przewidziano dwa sposoby adresowania grupowego, które mogą być aktywne dla każdego urządzenia w tym samym czasie. Każde z urządzeń wyposażonych w układ CY8CPLC10 może być skonfigurowane w taki sposób by było członkiem jednej z 256 dostępnych grup urządzeń lub też może zostać przydzielone do 8 grup w tym samym czasie. Co oczywiste, z założenia przy wysyłaniu wiadomości grupowej wyłączone zostaje potwierdzanie wysyłanych pakietów danych bo w tym wypadku traci ono sens. Imponujące możliwości, nieprawdaż?

Rejestry konfiguracyjne CY8CPLC10

W tym miejscu dysponujemy już sporą dawką wiedzy na temat naszego modemu PLC, jednak by w pełni zrozumieć „drzemiące" w nim możliwości nie sposób nie przedstawić, choćby po krótce, najważniejszych rejestrów konfiguracyjnych, przy udziale których mamy dostęp do pełnej funkcjonalności układu.

Rejestr: INT_ENABLE (0x00)							
D7	D6	D5	D4	D3	D2	D1	D0
INT_CLEAR	INT_POLARITY	INT_UNABLE_TO_TX	INT_NO_ACK	INT_NO_RESP	INT_RX_PACKET_DROPPED	INT_DATA_AVAIBLE	INT_DATA_SENT

Znaczenie poszczególnych bitów rejestru **INT_ENABLE** przedstawia się następująco:

- Bit **INT_CLEAR** jest ustawiany przez układ CY8CPLC10 za każdym razem, gdy zajdzie zdarzenie, dla którego uruchomiono generowanie przerwań na wyprowadzeniu HOST_INT. Użytkownik powinien wyzerować ten bit po odczytaniu rejestru statusu INT_STATUS, który przechowuje flagi zdarzeń (spowoduje to wykasowanie flag tegoż rejestru za wyjątkiem flag: STATUS_RX_PACKET_DROPPED i STATUS_RX_DATA_AVAILABLE).
- Bit **INT_POLARITY** decyduje o polaryzacji sygnału na wyjściu HOST_INT: 0 → aktywny stan wysoki, 1 → aktywny stan niski.
- Bit **INT_UNABLE_TO_TX** uruchamia (gdy ustawiony) generowanie przerwań w przypadku, gdy modem PLC nie jest w stanie wysłać pakietu danych (upływa maksymalny czas w jakim modem PLC stara się wysłać pakiet danych sprawdzając ustawicznie zajętość magistrali).
- Bit **INT_NO_ACK** uruchamia (gdy ustawiony i wybrano potwierdzanie pakietów danych) generowanie przerwań w przypadku nieotrzymania potwierdzenia odbioru pakietu danych (od adresata wiadomości).
- Bit **INT_NO_RESP** uruchamia (gdy ustawiony) generowanie przerwań w przypadku, gdy modem nie otrzymał (w ustalonym czasie) odpowiedzi od adresata, od którego zażądał wysłania danych (wybrane, predefiniowane rozkazy stosu komunikacyjnego).
- Bit **INT_RX_PACKET_DROPPED** uruchamia (gdy ustawiony) generowanie przerwań w wypadku przepełnienia (nadpisania) bufora danych odbiorczych.
- Bit **INT_DATA_AVAIBLE** uruchamia (gdy ustawiony) generowanie przerwań w przypadku odebrania pakietu danych.
- Bit **INT_DATA_SENT** uruchamia (gdy ustawiony) generowanie przerwań w przypadku pomyślnego wysłania pakietu danych.

Rejestr: LOCAL_LA_LSB (0x01)							
D7	D6	D5	D4	D3	D2	D1	D0
LA7	LA6	LA5	LA4	LA3	LA2	LA1	LA0

Wartość rejestru **LOCAL_LA_LSB** określa 8-bitowy adres logiczny modemu PLC lub młodszy bajt 16-bitowego, rozszerzonego adresu logicznego (w przypadku korzystania z tego typu adresacji).

Rejestr: LOCAL_LA_MSB (0x02)							
D7	D6	D5	D4	D3	D2	D1	D0
LA7	LA6	LA5	LA4	LA3	LA2	LA1	LA0

Wartość rejestru **LOCAL_LA_MSB** określa starszy bajt 16-bitowego, rozszerzonego adresu logicznego (w przypadku korzystania z tego typu adresacji).

Rejestr: LOCAL_GROUP (0x03)							
D7	D6	D5	D4	D3	D2	D1	D0
GA7	GA6	GA5	GA4	GA3	GA2	GA1	GA0

Wartość rejestru **LOCAL_GROUP** określa 8-bitowy adres grupy, do której przyporządkowano modem PLC (w przypadku odbioru wiadomości typu multicast).

Rejestr: LOCAL_GROUP_HOT (0x04)							
D7	D6	D5	D4	D3	D2	D1	D0
GH7	GH6	GH5	GH4	GH3	GH2	GH1	GH0

Wartość każdego z bitów rejestru **LOCAL_GROUP_HOT** określa przynależność modemu PLC do dodatkowo zdefiniowanych grup (o numerach od 7 do 0, odpowiednio dla każdego z bitów rejestru). Na przykład, wpisanie do tego rejestru wartości 0b00010001 spowoduje, iż modem PLC należeć będzie do dodatkowych grup o numerach #5 i #1.

Rejestr: PLC_MODE (0x05)							
D7	D6	D5	D4	D3	D2	D1	D0
TX_ ENA-BLE	RX_ ENA-BLE	LOCK_ CON-FIG	DI-SAB-LE_ BIU	RX_ OVER-WRI-TE	SET_ EXT_ ADDRESS	PRO-MI-SCUO-US_ MASK	PRO-MI-SCUO-US_ CRC_ MASK

Rejestr **PLC_MODE** określa podstawowe właściwości sprzętowe modemu PLC. Poszczególne bity tegoż rejestru mają następujące znaczenie funkcjonalne:

- Bit **TX_ENABLE** uruchamia (gdy ustawiony) nadajnik modemu PLC.
- Bit **RX_ENABLE** uruchamia (gdy ustawiony) odbiornik modemu PLC.
- Bit **LOCK_CONFIG** uniemożliwia (gdy ustawiony) zdalną (przy udziale innego modemu PLC) zmianę konfiguracji modemu PLC
- Bit **DISABLE_BIU** wyłącza (gdy ustawiony) detektor zajętości magistrali PLC (mechanizm Band-In-Use).

Ustawienie bitu **RX_OVERWRITE** powoduje, iż nowe dane odebrane przez modem nadpisują dane ostatnio odebrane, nawet, jeśli nie zostały jeszcze odczytane.

Wartość bitu **SET_EXT_ADDRESS** określa rodzaj stosowanego adresu logicznego: 0 → 8-bitowy adres logiczny, 1 → 16-bitowy, rozszerzony adres logiczny.

Wartość bitu **PROMISCUOUS_MASK** określa sposób interpretowania danych odbieranych przez modem PLC: 0 → modem PLC ignoruje wszystkie dane, jeśli adres przeznaczenia nie jest zgodny z jego adresem (logicznym lub fizycznym), 1 → modem PLC odbiera wszystkie dane niezależnie od adresu ich przeznaczenia, jeśli tylko zaopatrzone zostały w poprawna sumę kontrolna CRC8.

Wartość bitu **PROMISCUOUS_CRC_MASK** określa mechanizmy kontroli poprawności przesyłanych danych: 0 → modem PLC ignoruje wszystkie dane, dla których przesłano niepoprawną wartość sumy kontrolnej CRC8, 1 → modem PLC odbiera wszystkie dane niezależnie od wartości sumy kontrolnej CRC8 (jeśli tylko adres przeznaczenia przesyłanych danych zgadza się z adresem modemu).

Rejestr: TX_MESSAGE_LENGTH (0x06)							
D7	D6	D5	D4	D3	D2	D1	D0
SEND_MESSAGE			PAYLOAD_LENGTH_MASK				

Rejestr **TX_MESSAGE_LENGTH** określa długość pakietu danych przeznaczonych do wysłania (użytecznych danych użytkownika) → bity D4…D0 oraz inicjuje, poprzez ustawienie bitu **SEND_MESSAGE**, transmisję danych. Należy mieć na uwadze, iż przed inicjacją transmisji danych należy ustawić wartości wszystkich rejestrów odpowiedzialnych za parametry tejże transmisji jak i wartości samych danych, tj. rejestry: TX_CONFIG, TX_DA, TX_COMMAND_ID i TX_DATA.

Rejestr: TX_CONFIG (0x07)							
D7	D6	D5	D4	D3	D2	D1	D0
TX_SA_TYPE	TX_DA_TYPE		TX_SERVI-CE_TYPE	TX_RETRY			

Rejestr **TX_CONFIG** określa podstawowe parametry nadajnika modemu PLC. Poszczególne bity tego rejestru mają następujące znaczenie:

- Bit **TX_SA_TYPE** określa rodzaj adresu urządzenia źródłowego (wysyłającego dane): 0 → adres logiczny, 1 → adres fizyczny.
- Bity **TX_DA_TYPE** określają rodzaj adresu urządzenia docelowego (urządzenia, dla którego przeznaczone są wysyłane dane): 0 → adres logiczny, 1 → adres grupowy, 2 → adres fizyczny.
- Bit **TX_SERVICE_TYPE** określa konieczność (jeśli ustawiony) potwierdzania wysyłanych pakietów danych determinując generowanie sygnału ACK przez strony transmisji.
- Bity **TX_RETRY** określają liczbę dopuszczalnych retransmisji pakietu danych w przypadku niepowodzenia tejże transmisji.

Rejestr: TX_DA (0x08)							
D7	D6	D5	D4	D3	D2	D1	D0
REMOTE_NODE_DESTINATION_ADDRESS							

Rejestr **TX_DA** określa adres urządzenia docelowego (jeśli wybrany wcześniej rodzaj adresu rządzenia docelowego to adres logiczny lub grupowy) lub też definiuje najstarszy bajt 64-bitowego, fizycznego adresu urządzenia docelowego. W tym drugim wypadku, kolejne rejestry o adresach 0x09… …0x0F definiują pozostałe bajty adresu fizycznego.

Rejestr: TX_COMMAND_ID (0x10)							
D7	D6	D5	D4	D3	D2	D1	D0
TX_COMMAND_ID							

Rejestr **TX_COMMAND_ID** determinuje rodzaj przesyłanego rozkazu do urządzenia docelowego. Predefiniowane rozkazy o numerach 0x01…0x0F powodują podjęcie przez modem docelowy różnych akcji automatycznych lub też generują zmiany jego konfiguracji, np. zdalną zmianę adresu logicznego modemu docelowego, żądanie danych, zdalny odczyt adresu fizycznego modemu docelowego itd i są ściśle określone w specyfikacji stosu komunikacyjnego. Rozkazy o numerach 0x30…0xFF nie są zdefiniowane przez stos komunikacyjny i przeznaczone są do swobodnego wykorzystania przez docelową aplikację użytkownika.

Rejestr: TX_DATA (0x11)							
D7	D6	D5	D4	D3	D2	D1	D0
TX_DATA							

Rejestr **TX_DATA** określa początek obszaru (od 0x11 do 0x2F) przeznaczonego na użyteczne dane przeznaczone do transmisji.

Rejestr: THRESHOLD_NOISE (0x30)							
D7	D6	D5	D4	D3	D2	D1	D0
	AUTO_BIU_ THRESHOLD				BIU_THRESHOLD_CON-STANT		

Rejestr **THRESHOLD_NOISE** określa parametry dla mechanizmu detekcji zajętości magistrali danych PLC (detektora BIU). Bity **BIU_THRESHOLD_CONSTANT** określają predefiniowany poziom sygnału (w dBμVrms), dla którego przyjmuje się, iż magistrala danych jest zajęta przez inne urządzenie PLC. Wartość domyślna (0x03) odpowiada ustawieniu 87 dBμVrms. Ustawienie bitu **AUTO_BIU_THRESHOLD** wymusza automatyczny dobór poziomu sygnału detektora BIU.

Rejestr: MODEM_CONFIG (0x31)							
D7	D6	D5	D4	D3	D2	D1	D0
	TX_DELAY			MODEM_ FSKBW_MASK		MODEM_BPS_ MASK	

Rejestr **MODEM_CONFIG** określa podstawowe parametry modemu PLC. Poszczególne bity rejestru mają następujące znaczenie funkcjonalne:

- Bity **TX_DELAY** określają opóźnienie, jakie wprowadzane jest przez modem przed każdą transmisja danych: 0 → 7 ms, 1 → 13 ms, 2 → 19 ms, 3 → 25 ms.
- Bit **MODEM_FSKBW_MASK** determinuje wartości częstotliwości modulacji odpowiadające poziomom logicznym „0” i „1”: 0 → 133,3 kHz dla „0” i 131,8 kHz dla „1”, 1 → 133,3 kHz dla „0” i 130,4 kHz dla „1”. Co oczywiste, ustawienie to musi być takie same dla wszystkich urządzeń PLC komunikujących się w obrębie projektowanej sieci.
- Bity **MODEM_BPS_MASK** określają prędkość transmisji modemu PLC: 0 → 600 bps, 1 → 1200 bps, 2 → 1800 bps, 3 → 2400 bps (ustawienie domyślne). Co oczywiste, ustawienie to powinno być takie same dla wszystkich urządzeń PLC w obrębie projektowanej sieci.

Rejestr: TX_GAIN (0x32)							
D7	D6	D5	D4	D3	D2	D1	D0
				TX_GAIN			

Rejestr **TX_GAIN** determinuje predefiniowane wzmocnienie sygnału dla wbudowanego w modem PLC nadajnika. Dostępne wartości z przedziału 0x00…0x0E odpowiadają ustawieniom 55 mVp-p…3.5 Vp-p (domyślna wartość 0x0B odpowiada ustawieniu 1.55 Vp-p). Zmiany ustawień tego rejestru niezbędne są w przypadku implementacji dodatkowego, zewnętrznego wzmacniacza sygnału nadajnika jak też dla spełnienia wymagań dotyczących poziomu sygnału transmisji PLC w zakresie odpowiednich norm.

Rejestr: RX_GAIN (0x33)							
D7	D6	D5	D4	D3	D2	D1	D0
					RX_GAIN		

Rejestr **RX_GAIN** determinuje predefiniowane wzmocnienie sygnału dla wbudowanego

w modem PLC odbiornika. Dostępne wartości z przedziału 0x00…0x07 odpowiadają ustawieniom 5 mVrms…125 µVrms (domyślna wartość 0x00 odpowiada ustawieniu 5 mVrms). Zmiany ustawień tego rejestru niezbędne są w przypadku implementacji dodatkowego, zewnętrznego wzmacniacza sygnału odbiornika.

Rejestr: RX_MESSAGE_INFO (0x40)							
D7	D6	D5	D4	D3	D2	D1	D0
NEW_RX_MSG	RX_DA_TYPE	RX_SA_TYPE	RX_MSG_LENGTH				

Rejestr **RX_MESSAGE_INFO** udostępnia informacje na temat właściwości odebranego pakietu danych. Poszczególne bity tegoż rejestru mają następujące znaczenie funkcjonalne:

- Bit **NEW_RX_MSG** jest flagą odebrania nowego pakietu danych. Bit ten powinien być wyzerowany przez aplikację użytkownika po odczytaniu przesłanych danych by umożliwić dalszy odbiór pakietów danych. Wyzerowanie tego bitu powoduje także skasowanie flag STATUS_VALUE_CHANGE, STATUS_RX_PACKET_DROPPED i STATUS_RX_DATA_AVAILABLE w rejestrze flag przerwań INT_STATUS_REGISTER.
- Bit **RX_DA_TYPE** przechowuje informację o rodzaju adresu urządzenia docelowego, jaki zdefiniowano dla odebranego pakietu danych: 0 → adres logiczny/fizyczny, 1 → adres grupowy.
- Bit **RX_SA_TYPE** przechowuje informację o rodzaju adresu urządzenia źródłowego, jaki zdefiniowano dla odebranego pakietu danych: 0 → adres logiczny, 1 → adres fizyczny.
- Bity **RX_MSG_LENGTH** przechowuje informację o długości danych użytecznych w odebranym pakiecie danych.

Rejestr: RX_SA (0x41)							
D7	D6	D5	D4	D3	D2	D1	D0
REMOTE_NODE_SOURCE_ADDRESS							

Rejestr **RX_SA** określa adres urządzenia źródłowego, które wysłało do naszego modemu PLC pakiet danych. W zależności od wartości bitu **RX_SA_TYPE** w rejestrze **RX_MESSAGE_INFO** adres ten stanowi wartość wyłącznie rejestru 0x41 dla rodzaju adresu typu logicznego, wartości rejestrów 0x41 i 0x42 (jako MSB) dla rozszerzonego adresu logicznego i wartości rejestrów 0x41…0x48 (jako LSB) dla adresu fizycznego.

Rejestr: RX_COMMAND_ID (0x49)							
D7	D6	D5	D4	D3	D2	D1	D0
RX_COMMAND_ID							

Rejestr **RX_COMMAND_ID** przechowuje informację o rodzaju przesłanego rozkazu w bieżącym pakiecie odebranych danych.

Rejestr: RX_DATA (0x4A)							
D7	D6	D5	D4	D3	D2	D1	D0
RX_DATA							

Listing 1. Funkcje obsługujące interfejs TWI

```
void TWI_Init(void)
{
    TWBR = 6;  //F_TWI=395kHz @ fosc=11059200Hz
}

void TWI_Start(void)
{
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTA);
    while (!(TWCR&(1<<TWINT)));
}

void TWI_Stop(void)
{
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
    while (TWCR&(1<<TWSTO));
}

void TWI_WriteByte(uint8_t Byte)
{
    TWDR = Byte;
    TWCR = (1<<TWINT)|(1<<TWEN);
    while ( !(TWCR&(1<<TWINT)));
}

uint8_t TWI_ReadByte(uint8_t ACK_NACK)
{
    TWCR = (1<<TWINT)|(ACK_NACK<<TWEA)|(1<<TWEN);
    while ( !(TWCR & (1<<TWINT)));
    return TWDR;
}
```

Listing 2. Ciała funkcji odpowiedzialnych za zapis/odczyt rejestrów układu CY8CPLC10

```
void writePLCregister(uint8_t registerNr, uint8_t registerValue)
{
    TWI_Start();
    TWI_WriteByte(PLC_WRITE_ADDR);
    TWI_WriteByte(registerNr);
    TWI_WriteByte(registerValue);
    TWI_Stop();
    _delay_us(BUS_FREE_TIME_BEFORE_STO_STA);
}

uint8_t readPLCregister(uint8_t registerNr)
{
    register uint8_t registerValue;
    TWI_Start();
    TWI_WriteByte(PLC_WRITE_ADDR);
    TWI_WriteByte(registerNr);
    _delay_us(50);
    TWI_Start();
    TWI_WriteByte(PLC_READ_ADDR);
    registerValue = TWI_ReadByte(NACK);
    TWI_Stop();
    _delay_us(BUS_FREE_TIME_BEFORE_STO_STA);
    return registerValue;
}

void writePLCregisters(uint8_t startRegNr,volatile  uint8_t *regValues,
uint8_t bytesToWrite)
{
    TWI_Start();
    TWI_WriteByte(PLC_WRITE_ADDR);
    TWI_WriteByte(startRegNr);
    while(bytesToWrite--) TWI_WriteByte(*(regValues++));
    TWI_Stop();
    _delay_us(BUS_FREE_TIME_BEFORE_STO_STA);
}

void readPLCregisters(uint8_t startRegNr, uint8_t *regValues, uint8_t
bytesToRead)
{
    TWI_Start();
    TWI_WriteByte(PLC_WRITE_ADDR);
    TWI_WriteByte(startRegNr);
    _delay_us(50);
    TWI_Start();
    TWI_WriteByte(PLC_READ_ADDR);
    while(bytesToRead--) *(regValues++) = TWI_ReadByte(bytesToRead? ACK:NACK);
    TWI_Stop();
    _delay_us(BUS_FREE_TIME_BEFORE_STO_STA);
}
```

Rejestr **RX_DATA** określa początek obszaru (od 0x4A do 0x68), który zawiera użyteczne dane przesłane w bieżącym pakiecie danych.

Rejestr: INT_STATUS (0x69)							
D7	D6	D5	D4	D3	D2	D1	D0
STATUS_VALUE_CHANGE		STATUS_BUSY	STATUS_TX_NO_ACK	STATUS_TX_NO_RESP	STATUS_RX_PACKET_DROPPED	STATUS_RX_DATA_AVAILABLE	STATUS_TX_DATA_SENT

Rejestr **INT_STATUS** jest rejestrem flag przerwań związanych z funkcjonowaniem modemu PLC. Aby skasować flagi w tym rejestrze (za wyjątkiem flag STATUS_RX_PACKET_DROPPED i STATUS_RX_DATA_AVAILABLE) należy wyzerować bit **INT_CLEAR** w rejestrze **INT_ENABLE**. Z kolei, aby skasować flagi STATUS_RX_PACKET_DROPPED, STATUS_RX_DATA_AVAILABLE i STATUS_VALUE_CHANGE należy wyzerować bit **NEW_RX_MSG** w rejestrze **RX_MESSAGE_INFO**.

Listing 3. Ciała funkcji odpowiedzialnych za ustawienie rodzaju adresów urządzenia źródłowego i docelowego modemu PLC oraz funkcji umożliwiającej ustawienie adresu urządzenia docelowego

```
//txSAtype: TX_SA_TYPE_LOGICAL, TX_SA_TYPE_PHYSICAL
//txDAtype: TX_DA_TYPE_LOGICAL, TX_DA_TYPE_GROUP, TX_DA_TYPE_PHYSICAL

void setPLCtxAddrType(uint8_t txSAtype, uint8_t txDAtype)
{
    register uint8_t configRegValue;
//Odczytanie bieżącej wartości rejestru TX_CONFIG i maskowanie bitów
//odpowiedzialnych za typy adresów SA i DA
    configRegValue = readPLCregister(TX_CONFIG_REG) & ~(TX_ADDR_MASK);
//Ustawienie nowych typów adresów będących argumentami funkcji - należy
//używać definicji typów z pliku nagłówkowego
    writePLCregister(TX_CONFIG_REG, configRegValue|txSAtype|txDAtype);
}

void setPLCtxDA(uint8_t txDAtype,volatile  uint8_t *txDA)
{
//W zależności od typu adresu DA zapisujemy odpowiednią
//ilość bajtów adresu DA
    switch(txDAtype)
    {
        case TX_DA_TYPE_LOGICAL:
        case TX_DA_TYPE_GROUP:
            writePLCregister(TX_DA_REG, *txDA);
            break;
        case TX_DA_TYPE_PHYSICAL:
            writePLCregisters(TX_DA_REG, txDA, 8);
            break;
    }
}
```

Listing 4. Ciała funkcji umożliwiających ustawienie adresu logicznego i adresu grupowego modemu PLC

```
void setPLCnodeLA(uint8_t logicalAddress)
{
    writePLCregister(LOGICAL_ADDR_LSB_REG, logicalAddress);
}

void setPLCnodeGA(uint8_t groupAddress)
{
    writePLCregister(GROUP_ADDR_REG, groupAddress);
}
```

Listing 5. Ciała funkcji umożliwiających ustalenie typów adresów urządzenia źródłowego i docelowego przesłanego pakietu danych PLC jak i samego adresu urządzenia źródłowego

```
//rxSAtype: RX_SA_TYPE_LOGICAL lub RX_SA_TYPE_PHYSICAL
//rxDAtype: RX_DA_TYPE_LOGICAL_PHYSICAL lub RX_DA_TYPE_GROUP

void getPLCrxAddrType(uint8_t *rxSAtype, uint8_t *rxDAtype)
{
    register uint8_t messageInfo;
//Odczytanie bieżącej wartości rejestru RX_MESSAGE_INFO_REG
//i ustalenie typu adresu DA i SA otrzymanej wiadomości
    messageInfo = readPLCregister(RX_MESSAGE_INFO_REG);
//Zwraca typ DA odebranej wiadomości: RX_DA_TYPE_LOGICAL_PHYSICAL
//lub RX_DA_TYPE_GROUP
    *rxDAtype =  messageInfo & 0b01000000;  //Typ zapisany w bicie 6
//Zwraca typ SA odebranej wiadomości: RX_SA_TYPE_LOGICAL lub
//RX_SA_TYPE_PHYSICAL
    *rxSAtype = messageInfo & 0b00100000;  //Typ zapisany w bicie 5
}

void getPLCrxSA(uint8_t *rxSA)
{
//Niezależnie od rodzaju SA otrzymanej wiadomości odczytujemy
//zawsze 8 bajtów. Jeśli adres urządzenia, które przesłało nam dane
//jest typu LA to istotny jest tylko pierwszy bajt odczytanego adresu,
//w przypadku PA, ważne jest całe 8 bajtów.
    readPLCregisters(RX_SA_REG, rxSA, 8);
}
```

Listing 6. Ciało funkcji umożliwiającej odczytanie przesłanego pakietu danych PLC

```
void readPLCrxPacket(uint8_t *rxCommand, uint8_t *rxData, uint8_t
*rxDataLength)
{
    register uint8_t infoRegister;
//Odczytanie bieżącej wartości rejestru RX_MESSAGE_INFO_REG i ustalenie
rozmiaru otrzymanej wiadomości
    *rxDataLength = readPLCregister(RX_MESSAGE_INFO_REG) & 0x1F;  //0...31
//Ustalenie rodzaju otrzymanej komendy
    *rxCommand = readPLCregister(RX_COMMAND_ID_REG);
//Wczytanie do tablicy rxData będącej argumentem funkcji wszystkich,
//otrzymanych danych
    readPLCregisters(RX_DATA_REG, rxData, *rxDataLength);
//Aby skasować flagi STATUS_VALUE_CHANGE, STATUS_RX_PACKET_DROPPED
//i STATUS_RX_DATA_AVAIBLE w rejestrze INTERRUPT_STATUS_REG musimy
//wyzerować flagę NEW_PACKET_RECEIVED w rejestrze RX_MESSAGE_INFO_REG
//(datasheet)
    infoRegister = readPLCregister(RX_MESSAGE_INFO_REG);
    writePLCregister(RX_MESSAGE_INFO_REG, infoRegister & ~NEW_PACKET_
RECEIVED);
}
```

Poszczególne bity rejestru **INT_STATUS** mają następujące znaczenie funkcjonalne:

- Bit **STATUS_VALUE_CHANGE** ustawiany jest za każdym razem, gdy odebrano nowe dane.
- Bit **STATUS_BUSY** ustawiany jest za każdym razem, gdy nie ma możliwości przesłania pakietu danych z uwagi na zajętość magistrali danych PLC (timeout 1,1…3 s w zależności od ustawień detektora BIU).
- Bit **STATUS_TX_NO_ACK** ustawiany jest za każdym razem, gdy w czasie 500 ms od przesłania pakietu danych nie odebrano potwierdzenia odebrania tegoż pakietu danych po stronie urządzenia docelowego (w przypadku konfiguracji sieci obejmującej potwierdzanie pakietów danych).
- Bit **STATUS_TX_NO_RESP** ustawiany jest w przypadku wybranych rozkazów stosu komunikacyjnego, jak na przykład rozkazów, które wymuszają przesłanie przez urządzenie docelowe danych (żądanie przesłania danych wysłane przez urządzenie źródłowe, np. rozkaz SEND_REMOTE_DATA) lub rozkazu zdalnej zmiany adresu logicznego urządzenia docelowego, a dane takie nie zostaną przez to urządzenie przesłane w czasie 1.5 s od przesłania rozkazu.
- Bit **STATUS_RX_PACKET_DROPPED** ustawiany jest w przypadku nadpisania bufora odbiorczego nowymi danymi, gdy dane poprzednie nie zostały jeszcze odczytane.
- Bit **STATUS_RX_DATA_AVAILABLE** ustawiany jest za każdym razem, gdy modem PLC odbierze nowy pakiet danych.
- Bit **STATUS_TX_DATA_SENT** ustawiany jest wyłącznie wtedy, gdy bieżąca transmisja danych zakończy się powodzeniem.

Rejestr: LOCAL_PA (0x6A)							
D7	D6	D5	D4	D3	D2	D1	D0
PHYSICAL_ADDRESS (MSB)							

Rejestr **LOCAL_PA** określa początek obszaru (od 0x6A do 0x71), który zawiera unikalny, 64-bitowy, fizyczny adres modemu PLC.

Funkcje sterujące pracą modemu CY8CPLC10

Przebrnęliśmy przez szczegółowy opis rejestrów konfiguracyjnych układu CY8CPLC10, dzięki czemu dysponujemy już niezbędną wiedzą by przejść do konfiguracji naszego systemu. Zanim jednak, opiszę szczegóły implementacyjne tego, konkretnego rozwiązania, warto pokazać, choćby po krótce, funkcje, za pomocą których jest możliwe sterowanie pracą modemu. Na początek opiszę podstawowe funkcje, dzięki którym jest możliwa komunikacja przy udziale interfejsu TWI mikrokontrolera, przy czym należy

zaznaczyć, iż są to najprostsze implementacje poszczególnych funkcjonalności niewyposażone np. w mechanizm obsługi błędów, który moim zdaniem, nie jest elementem niezbędnym w tak prostych i niewielkich systemach mikroprocesorowych.

Ciała funkcji odpowiedzialnych za obsługę interfejsu TWI pokazano na **listingu** 1.

Myślę, że nie wymagają one dodatkowego komentarza, gdyż ich nazwy są na tyle wymowne, iż nie pozostawiają wątpliwości, co do realizowanych przez nie funkcjonalności. Przejdźmy, zatem do podstawowych funkcji umożliwiających zapis i odczyt pojedynczych rejestrów modemu PLC, jak i całych grup rejestrów, które to pokazano na **listingu 2**. Oczywiście, można by było zastosować jeden rodzaj funkcji dla odczytu/zapisu zarówno pojedynczych rejestrów jak i całych ich grup, jednak takie rozwiązanie powoduje optymalizację kodu wynikowego zarówno pod względem szybkości wykonania jak i wielkości generowanego kodu.

Pora na przedstawienie kluczowych funkcji umożliwiających konfigurację podstawowych parametrów naszego modemu PLC. Na początek dwie funkcje, jedna przeznaczona do ustawienia typów adresów urządzenia źródłowego i docelowego pakietu danych PLC, zaś druga do ustawienia samego adresu urządzenia docelowego. Funkcje te przedstawiono na **listingu 3**.

Kolejne, dwie, bardzo krótkie funkcje narzędziowe, które pokazano na **listingu 4** służą do ustawienia adresu logicznego modemu PLC jak i adresu grupy, do której przyporządkowujemy nasze urządzenie.

Pora na funkcje użyteczne w przypadku odbioru pakietu danych PLC. Pierwsza z nich, przedstawiona na **listingu 5** służy do ustalenia typów adresów urządzenia źródłowego i docelowego, które zostały przesłane w odebranym pakiecie danych PLC, zaś druga z nich służy do odczytania adresu urządzenia źródłowego.

Do „kompletu" brakuje nam funkcji, która umożliwia odczytanie przesłanego pakietu danych, towarzyszącego im rozkazu oraz określenie długości tegoż pakietu danych. Ciało tej funkcji pokazano na **listingu 6**.

Dodatkowo, w przypadku odbioru pakietu danych PLC, przydatna może być funkcja, która pozwala nam na ustalenie zdarzenia, jakie spowodowało wyzwolenie przerwania odpowiedzialnego za obsługę pakietów danych. Ciało tej funkcji pokazano na **listingu 7**.

No i już zupełnie na koniec, funkcja, która pozwala na pełne skonfigurowanie modemu PLC. Ciało tej funkcji pokazano na **listingu 8**.

Dla porządku przedstawię również listing pliku nagłówkowego modemu PLC, który jak zwykle konstruuję w taki sposób, by bez potrzeby zaglądania do dokumentacji układu, zorientować się w dostępnej liście możliwości. Zawartość tego pliku nagłówkowego pokazano na **listingu 9**.

Listing 7. Ciało funkcji umożliwiającej ustalenie zdarzenia, jakie spowodowało wyzwolenie przerwania odpowiedzialnego za obsługę pakietów danych

```
uint8_t readPLCintRegister(void)
{
    register uint8_t intStatusReg, intEnableReg;
//Odczyt rejestru statusu przerwań PLC by ustalić rodzaj zdarzenia
//jakie zostało zgłoszone
    intStatusReg = readPLCregister(INTERRUPT_STATUS_REG);
//Po odczycie powyższego rejestru należy wyzerować bit INT_CLEAR
//w rejestrze INTERRUPT_ENABLE_REG (datasheet, page 7.)
    intEnableReg = readPLCregister(INTERRUPT_ENABLE_REG) & ~INT_CLEAR;
    writePLCregister(INTERRUPT_ENABLE_REG, intEnableReg);
    return intStatusReg;
}
```

Listing 8. Ciało funkcji umożliwiającej kompletną konfigurację modemu PLC

```
void initPLCdevice(uint8_t nodeLA)
{
//Uruchomienie i podstawowa konfiguracja modemu PLC
    writePLCregister(PLC_MODE_REG, TX_ENABLE|RX_ENABLE|RX_OVERRIDE|ENABLE_
BIU|CHECK_DA|VERIFY_PACKET_CRC8);
//Ustawienie poziomu sygnału dla mechanizmu CSMA (Carrier Sense Multimaster
Access) zapewniającego wielodostęp do medium transmisyjnego
    writePLCregister(THRESHOLD_NOISE_REG, BIU_TRESHOLD_87DBUV);
//Konfiguracja parametrów transmisji
    writePLCregister(MODEM_CONFIG_REG, MODEM_BPS_2400|MODEM_FSK_BAND_
DEV_3KHZ);
//Uruchomienie przerwań dla wybranych zdarzeń (aktywny poziom niski
//na wyprowadzeniu HOST_INT)
    writePLCregister(INTERRUPT_ENABLE_REG, INT_POLARITY_LOW|INT_UNABLE_TO_
TX|INT_TX_NO_ACK|INT_TX_NO_RESP|INT_RX_DATA_AVAILABLE|INT_TX_DATA_SENT);
//Ustawienie trybu potwierdzania pakietów danych oraz liczby prób
//transmisji = 5 (domyślne logiczne typy adresów SA i DA)
    writePLCregister(TX_CONFIG_REG, TX_SERVICE_ACKNOWLEDGED|0x05);
//Ustawienie wzmocnienia dla modułu nadajnika PLC
    writePLCregister(TX_GAIN_REG, TX_GAIN_LEVEL_1550MV);
//Ustawienie czułości dla modułu odbiornika PLC
    writePLCregister(RX_GAIN_REG, RX_GAIN_LEVEL_250UV);
    setPLCnodeLA(nodeLA);  //Ustawienie numeru LA modemu PLC
//Ustawienie adresu Grupowego modemu PLC
    setPLCnodeGA(MASTER_GROUP_ADDR);
//Konfiguracja i aktywacja przerwania INT0 odpowiedzialnego za
//obsługę odbioru wiadomości PLC
    HOST_INT_PORT |= (1<<HOST_INT_NR);  //Podciągnięcie INT0 do VCC
    EICRA |= (1<<ISC01);  //Zbocze opadające na INT0 wyzwala przerwanie
    EIMSK |= (1<<INT0);  //Odblokowanie przerwania INT0
}
```

Założenia projektowe, funkcjonalność systemu

Na tą chwilę dysponujemy już kompletną wiedzą, która daje nam możliwość obsługi modemu PLC w związku, z czym pora przejść do konkretnej implementacji systemu iControl. Podstawowe założenia systemu iControl przedstawiają się następująco:

W ramach jednej sieci systemu iControl (jednego mieszkania lub domu) zakłada się współistnienie maksymalnie 64 modułów wykonawczych (typu Slave) o adresach logicznych z zakresu 0x00…0x3F oraz maksymalnie 16 modułów sterujących (typu Master), wyposażonych w interfejs użytkownika z wyświetlaczem TFT, o adresach logicznych z zakresu 0xF0÷0xFF.

Adresy logiczne modułów wykonawczych nadawane są automatycznie przez moduły sterujące podczas konfigurowania sieci, zaś adresy logiczne modułów sterujących nadawane są przez użytkownika przy pomocy interfejsu użytkownika GUI.

Każdy moduł sterujący może zapamiętać i zaadresować maksymalnie 64 moduły wykonawcze (staje się wtedy tzw. „rodzicem" modułu wykonawczego).

Wszystkie moduły wykonawcze zapamiętane przez dany moduł sterujący mogą zostać połączone w maksymalnie 8 grup, dowolnie podczas konfiguracji sieci, reprezentujących pomieszczenia, nad którymi moduł ten ma kontrolę (np. pokoje).

Każdy moduł wykonawczy, którego dany moduł sterujący jest „rodzicem" może być współdzielony innemu modułowi sterującemu, dzięki czemu w jednym czasie kilka modułów sterujących może mieć kontrolę nad jednym i tym samym modułem wykonawczym (daje to możliwość sterowania np. oświetleniem w kilku pokojach z jednego modułu sterującego umieszczonego w innym miejscu), przy czym w takim wypadku, moduły sterujące wymieniają się na bieżąco parametrami nadzorowanego modułu sterującego by stan tego modułu był znany każdemu z nich.

W ramach graficznego interfejsu użytkownika modułu sterującego, każdy moduł wykonawczy identyfikowany jest przez unikalną nazwę nadaną przez użytkownika systemu. Podobnie, każda z 8 możliwych grup (pomieszczeń), w które mogą być zebrane moduły wykonawcze podlega konfiguracji (aktywacji/dezaktywacji, nadaniu nazwy) w ramach wspomnianego interfejsu użytkownika.

Przewidziano 5 rodzajów modułów wykonawczych, automatycznie rozpoznawanych przez moduły sterujące, co determinuje wygląd tego modułu w ramach interfejsu użytkownika oraz sposób sterowania:

Listing 9. Plik nagłówkowy modemu PLC

```
#define I2C_ADDR_PIN_FLOATING 1
#define HOST_INT_PORT PORTD
#define HOST_INT_PIN PIND
#define HOST_INT_NR PD2
#define BUS_FREE_TIME_BEFORE_STO_STA 550

#if I2C_ADDR_PIN_FLOATING
    #define PLC_READ_ADDR 0x03
    #define PLC_WRITE_ADDR 0x02
#else
    #define PLC_READ_ADDR 0xF5
    #define PLC_WRITE_ADDR 0xF4
#endif

// Rejestry nadajnika modemu PLC
#define INTERRUPT_ENABLE_REG 0x00
    #define INT_CLEAR (1<<7)
    #define INT_POLARITY_HIGH (0<<6)
    #define INT_POLARITY_LOW (1<<6)
    #define INT_UNABLE_TO_TX (1<<5)
    #define INT_TX_NO_ACK (1<<4)
    #define INT_TX_NO_RESP (1<<3)
    #define INT_RX_PACKET_DROPPED (1<<2)
    #define INT_RX_DATA_AVAILABLE (1<<1)
    #define INT_TX_DATA_SENT (1<<0)

#define LOGICAL_ADDR_LSB_REG 0x01
#define LOGICAL_ADDR_MSB_REG 0x02
#define GROUP_ADDR_REG 0x03
#define LOCAL_GROUP_HOT_REG 0x04

#define PLC_MODE_REG 0x05
    #define TX_ENABLE (1<<7)
    #define RX_ENABLE (1<<6)
    #define LOCK_CONFIGURATION (1<<5)
    #define DISABLE_BIU (1<<4)
    #define ENABLE_BIU (0<<4)
    #define RX_OVERRIDE (1<<3)
    #define SET_EXT_ADDRESS (1<<2)
    #define ACCEPT_ALL_MSGS (1<<1)
    #define CHECK_DA (0<<1)
    #define DISCARD_PACKET_CRC8 (1<<0)
    #define VERIFY_PACKET_CRC8 (0<<0)

#define TX_MESSAGE_LENGTH_REG 0x06
    #define SEND_MESSAGE (1<<7)
    #define PAYLOAD_LENGTH_MASK 0x1F

#define TX_CONFIG_REG 0x07
    #define TX_ADDR_MASK 0b11100000
    #define TX_SA_TYPE_LOGICAL (0<<7)
    #define TX_SA_TYPE_PHYSICAL (1<<7)
    #define TX_DA_TYPE_LOGICAL (0<<5)
    #define TX_DA_TYPE_GROUP (1<<5)
    #define TX_DA_TYPE_PHYSICAL (2<<5)
    #define TX_SERVICE_ACKNOWLEDGED (1<<4)
    #define TX_SERVICE_TYPE_NO_ACKNOWLEDGED (0<<4)

#define TX_DA_REG 0x08
#define TX_COMMAND_ID_REG 0x10
#define TX_DATA_REG 0x11

//Rejestry konfiguracyjne modemu PLC
#define THRESHOLD_NOISE_REG 0x30
    #define AUTO_BIU_DISABLED (0<<6)
    #define AUTO_BIU_ENABLED (1<<6)
    #define BIU_TRESHOLD_70DBUV 0x00
    #define BIU_TRESHOLD_75DBUV 0x01
    #define BIU_TRESHOLD_80DBUV 0x02
    #define BIU_TRESHOLD_87DBUV 0x03
    #define BIU_TRESHOLD_90DBUV 0x04
    #define BIU_TRESHOLD_93DBUV 0x05
    #define BIU_TRESHOLD_96DBUV 0x06
    #define BIU_TRESHOLD_99DBUV 0x07

#define MODEM_CONFIG_REG 0x31
    #define TX_DELAY_7MS (0<<5)
    #define TX_DELAY_13MS (1<<5)
    #define TX_DELAY_19MS (2<<5)
    #define TX_DELAY_25MS (3<<5)
    #define MODEM_FSK_BAND_DEV_3KHZ (1<<3)  //133.3:130.4
kHz
    #define MODEM_FSK_BAND_DEV_1_5KHZ (0<<3)
//133.3:131.8 kHz
    #define MODEM_BPS_600 0x00
    #define MODEM_BPS_1200 0x01
    #define MODEM_BPS_1800 0x02
    #define MODEM_BPS_2400 0x03

#define TX_GAIN_REG 0x32
    #define TX_GAIN_LEVEL_55MV 0x00
    #define TX_GAIN_LEVEL_75MV 0x01
    #define TX_GAIN_LEVEL_100MV 0x02
    #define TX_GAIN_LEVEL_125MV 0x03
    #define TX_GAIN_LEVEL_180MV 0x04
    #define TX_GAIN_LEVEL_250MV 0x05
    #define TX_GAIN_LEVEL_360MV 0x06
    #define TX_GAIN_LEVEL_480MV 0x07
    #define TX_GAIN_LEVEL_660MV 0x08
    #define TX_GAIN_LEVEL_900MV 0x09
    #define TX_GAIN_LEVEL_1250MV 0x0A
    #define TX_GAIN_LEVEL_1550MV 0x0B  //Default
    #define TX_GAIN_LEVEL_2250MV 0x0C
    #define TX_GAIN_LEVEL_3000MV 0x0D
    #define TX_GAIN_LEVEL_3500MV 0x0E

#define RX_GAIN_REG 0x33
    #define RX_GAIN_LEVEL_5000UV 0x00  //Default
    #define RX_GAIN_LEVEL_3500UV 0x01
    #define RX_GAIN_LEVEL_2500UV 0x02
    #define RX_GAIN_LEVEL_1250UV 0x03
    #define RX_GAIN_LEVEL_600UV 0x04
    #define RX_GAIN_LEVEL_350UV 0x05
    #define RX_GAIN_LEVEL_250UV 0x06
    #define RX_GAIN_LEVEL_125UV 0x07

//Rejestry odbiornika modemu PLC
#define RX_MESSAGE_INFO_REG 0x40
    #define NEW_PACKET_RECEIVED (1<<7)
    #define NO_PACKET_RECEIVED (0<<7)
    #define RX_DA_TYPE_LOGICAL_PHYSICAL (0<<6)
    #define RX_DA_TYPE_GROUP (1<<6)
    #define RX_SA_TYPE_LOGICAL (0<<5)
    #define RX_SA_TYPE_PHYSICAL (1<<5)
    #define RX_MESSAGE_LENGTH_BIT_MASK 0b11111

#define RX_SA_REG 0x41
#define RX_COMMAND_ID_REG 0x49
#define RX_DATA_REG 0x4A

#define INTERRUPT_STATUS_REG 0x69
    #define STATUS_VALUE_CHANGE (1<<7)
    #define STATUS_BUSY (1<<5)  //BIU timeout
    #define STATUS_TX_NO_ACK (1<<4)
    #define STATUS_TX_NO_RESP (1<<3)
    #define STATUS_RX_PACKET_DROPPED (1<<2)
    #define STATUS_RX_DATA_AVAIBLE (1<<1)
    #define STATUS_TX_DATA_SENT (1<<0)
```

wyłącznik 1-biegunowy, wyłącznik 2-biegunowy, ściemniacz, sensor temperatury, sterownik oświetlenia RGB LED.

System iControl sygnalizuje dołączenie nowych, jeszcze nieskonfigurowanych, modułów wykonawczych w sieci (zapamiętuje do 16 nowych modułów wykonawczych) jak również wystąpienie błędów transmisji (sterowania modułem wykonawczym).

System iControl umożliwia usuwanie modułów wykonawczych z sieci, a co za tym idzie – rekonfigurowanie sieci.

Zakłada się następującą kolejność czynności w ramach konfiguracji sieci: podłączone i skonfigurowane (nadany adres logiczny) wszystkie, planowane moduły sterujące a w każdym z nich zdefiniowana przynajmniej jedna z grup (pomieszczeń). Następnie, dołączamy i konfigurujemy każdy, kolejny moduł sterujący, przy czym dla uproszczenia procedury konfiguracji z punktu widzenia użytkownika, zakłada się, iż kolejny moduł wykonawczy zostanie dołączony do sieci dopiero po zakończeniu konfigurowania modułu ostatnio dołączonego (pozwala to łatwiej zorientować się, który moduł fizycznie jest aktualnie konfigurowany).

Znając logiczną strukturę systemu iControl pora na przedstawienie konstrukcji pakietów danych (przesyłanych rozkazów i towarzyszących im danych) w ramach konfiguracji sieci jak i jej „normalnego" funkcjonowania. Opiszę je i zaprezentuję schematy i projekty płytek drukowanych w drugiej części artykułu.

Robert Wołgajew, EP