multiTID – wielofunkcyjny, samochodowy komputer pokładowy (1)

Kilkakrotnie podejmowałem wyzwanie zaprojektowania komputera pokładowego i to zarówno takiego, który korzysta z oryginalnego, wbudowanego w deskę rozdzielczą wyświetlacza pokładowego, jak i własnego LCD, OLED czy TFT. Korzystając z doświadczenia, które zdobyłem projektując wspomniane urządzenia, postanowiłem zbudować komputer pokładowy będący syntezą wcześniej zastosowanych rozwiązań, lecz wyróżniający się największą i niespotykaną w innych rozwiązaniach funkcjonalnością. Miał to być komputer, który „tchnie" nowe życie w starsze pojazdy. MultiTID, czyli urządzenie, o którym mowa, ma możliwość współpracy z oryginalnymi radioodbiornikami montowanymi w pojazdach spod znaku Opla, a więc daje możliwość zastąpienia wbudowanego (dość archaicznego), trójfunkcyjnego wyświetlacz pokładowego nazywanego skrótem TID (Triple Info Display) przez niezwykle efektowny, duży i szerokoekranowy kolorowy wyświetlacz TFT o przekątnej 4.2". Co więcej, ma możliwość pomiaru spalania i rejestracji przeróżnych statystyk w pojazdach wyposażonych w instalację zasilania gazem LPG i to niezależnie dla obu rodzajów paliwa.

Rekomendacje: **prezentowany komputer pokładowy może zostać zamontowany w pojeździe dowolnej marki. W pojeździe innej marki po prostu nie będzie korzystał z funkcjonalności wyświetlania danych za pomocą wyświetlacza radioodbiornika.**

Oto lista funkcji, w które jest wyposażony komputer pokładowy multiTID::

- Pełna funkcjonalność typowego, samochodowego komputera pokładowego.
- Możliwość emulacji wyświetlacza TID montowanego w pojazdach marki Opel (8- i 10-znakowego), a więc możliwość obsługi komunikatów wysyłanych przez oryginalne radioodbiorniki.
- „Obsługa" dwóch rodzajów paliwa (Pb/LPG),
- Czytelny, intuicyjny, graficzny interfejs użytkownika wzorowany na najnowszych rozwiązaniach z segmentu Premium.
- Możliwość regulacji jasności podświetlenia wyświetlacza graficznego zgodnie z ustawieniem jasności podświetlenia zegarów pojazdu.
- Łatwość instalacji uzyskana poprzez zastosowanie oryginalnego złącza połączeniowego wyświetlacza TID (oraz złącz dodatkowych dla funkcji spalania).

Ponadto, postawiłem dość wysokie wymagania dotyczące funkcjonalności komputera pokładowego, którego zdecydowałem się wyposażyć w następujące funkcje:

- Pokazywanie temperatury na zewnątrz pojazdu (z zastosowaniem czujnika montowanego w pojazdach marki Opel) oraz ostrzeżenia o śliskiej nawierzchni (dla temperatury zewnętrznej niższej, niż 5°C).
- Pokazywanie chwilowej prędkości pojazdu (w km/godz.).
- Pokazywanie średniej prędkości pojazdu na przejechanym odcinku drogi (w km/godz.),
- Pokazywanie maksymalnej prędkości pojazdu na przejechanym odcinku drogi (w km/godz.),
- Pokazywanie chwilowego zużycia paliwa (w l/godz. dla prędkości ≤5 km/godz. oraz l/100 km dla pozostałych prędkości) dla obu rodzajów paliwa.
- Pokazywanie średniego zużycia paliwa (w l/100 km) dla obu rodzajów paliwa.
- Pokazywanie paliwa pozostającego w baku pojazdu(w l, jak i graficznie – bargraf) dla obu rodzajów paliwa.
- Pokazywanie przewidywanego zasięgu pojazdu na paliwie pozostającym w baku pojazdu (w km) dla obu rodzajów paliwa.
- Pokazywanie przejechanego dystansu od ostatniego kasowania (w km).
- Pokazywanie aktualnego czasu i daty (z zastosowaniem mechanizmu podtrzymania zasilania).
- Pokazywanie statystyk spalania na każde przejechane 10 km/1 km (dla ostatnio przejechanych 150 i 15 km) dla obu rodzajów paliwa.
- Automatyczna detekcja bieżącego rodzaju paliwa.

Jak widać, cel jest dość ambitny, jednak doświadczenia zdobyte przy konstruowaniu poprzednich urządzeń upraszczają znacząco proces jego implementacji. Zanim jednak przejdziemy do opisu samego urządzenia warto przybliżyć nieco temat pokładowego wyświetlacza TID, gdyż możliwość jego emulacji jest jedną z głównych zalet i zarazem unikalnych cech urządzenia multiTID, bo umożliwia współpracę sterownika z oryginalnymi radioodbiornikami dedykowanymi do pojazdów spod znaku Opla. Co więcej, sterownik nasz obsługuje wszystkie rodzaje radioodbiorników montowanych przez tego producenta, które to wyposażono w magistralę sterującą z sygnałami SDA/SCL/MRQ i może emulować wyświetlacze TID z 8-znakową i 10-znakową organizacją pola tekstowego, a więc wyświetlacze pokładowe montowane w takich modelach jak Astra F, Corsa B, Combo, Astra G, Corsa C, Meriva, Movano, Omega B, Vectra B, Tigra. Wyjątkiem są nowsze pojazdy wyposażone w wyświetlacz pokładowy obsługujący magistralę CAN. Ta cecha naszego sterownika czyni go wyjątkowo atrakcyjnym, z punktu widzenia użytkownika pojazdu tej marki i co warto podkreślić, nie znajduje odpowiednika ani w konstrukcjach amatorskich, ani też na rynku komercyjnym.

Zanim przejdziemy do szczegółów implementacyjnych warto przyjrzeć się rozwiązaniu zastosowanemu przez Opla, gdyż jest to przykład dobrze wykonanej pracy inżynierskiej i dogłębnie przemyślanej konstrukcji. Zgodnie z tym, o czym wspominano wcześniej, firma Opel od wielu lat stosuje w swoich pojazdach zewnętrzne, zintegrowane z deską rozdzielczą, podświetlane wyświetlacze LCD, których funkcje zależne są od modelu auta, jego wyposażenia, roku produkcji jak i modelu samego wyświetlacza. Ogólnie rzecz ujmując można wyróżnić kilka typów wyświetlaczy, których poglądowe porównanie przedstawiono w **tabeli 1**.

Poza wymienionymi, podstawowymi różnicami, w zależności od roku produkcji i wyposażenia pojazdu, stosowano wyświetlacze o różnej organizacji i rozdzielczości. Starsze miały organizację 1×8 znaków (tylko wielkie litery

Tabela 1. Porównanie funkcjonalności wyświetlaczy pokładowych stosowanych przez firmę Opel

Nazwa	Funkcje	Typ magistrali sterującej
DID	Data, godzina, obsługa radioodbiornika	Zmodyfikowana I²C
TID	Funkcje DID + temperatura zewnętrzna	
MID	Funkcje TID + obsługa komputera pokładowego	
NAVI	Funkcje MID + obsługa nawigacji satelitarnej	CAN

DODATKOWE MATERIAŁY NA FTP:

ftp://ep.com.pl

USER: 66532, PASS: 8nnjjeaa

W ofercie AVT*

AVT-5562

Podstawowe informacje:

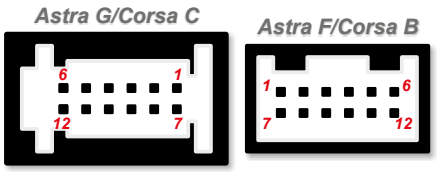
- Napięcie zasilania: 8...15 V DC.
- Maksymalny prąd obciążenia (z napięcia +12 V): 10 mA.
- Prąd podtrzymania zegara RTC (z napięcia BATT): 1 mA.
- Maksymalny prąd podświetlenia (z napięcia ILL+): 75 mA.
- Dokładność pomiaru temperatury: 1°C.
- Zakres pomiarowy temperatury zewnętrznej: –30...35°C.
- Zakres pomiarowy prędkości pojazdu: 0...255 km/godz.
- Zakres pomiarowy chwilowego zużycia paliwa: 0...99,9 l/100 km.
- Zakres pomiarowy średniego zużycia paliwa: 0...25,5 l/100 km.
- Zakres pomiarowy paliwa dostępnego w baku: 0...99,9 l.
- Zakres pomiarowy przejechanej odległości: 0...9999 km.
- Zakres pomiarowy dystansu do przejechania na dostępnym paliwie: 0...999 km.
- Zakresy regulacji parametrów konfiguracyjnych:
  - Stałe wtryskiwaczy: 1...999 ml/min.
  - Stała przetwornika drogi: 1...99 impulsów/obrót.
  - Obwód opony: 50...255 cm.
  - Liczba cylindrów: 2...8.
  - Pojemności baków: 25...99 l.
- Przesunięcie belek informacyjnych: 0÷9 pikseli.

Projekty pokrewne na FTP:

(wymienione artykuły są w całości dostępne na FTP)

AVT-5545	Komputer samochodowy Mee MK II (2.0) (EP 7-8-9/2016)
AVT-5495	Uniwersalny komputer samochodowy Mee (EP 3/2015)
AVT-3095	Komputer samochodowy (EdW 4-5/2014)
AVT-5405	TripCo – komputer samochodowy (EP 7/2013)
AVT-5395	TIDex – komputer dla samochodów z silnikiem Diesla (EP 5/2013)
AVT-5397	Komputer pokładowy z funkcją tempomatu (EP 5/2013)
AVT-1664	Transceiver CAN (EP 2/2012)
AVT-5280	Urządzenie diagnostyczne do sieci CAN (EP 3/2011)
AVT-5271	VAGlogger – Przyrząd diagnostyczny dla samochodów z grupy VW – Audi (EP 1/2011)
AVT-5160	Climatic – sterownik klimatyzacji samochodowej (EP 12/2008)
AVT-286	„Komputerek" pokładowy do samochodu (EP 5-6/1996)

*** Uwaga:**
Zestawy AVT mogą występować w następujących wersjach:
AVT xxxx UK to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+ płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). ***http://sklep.avt.pl***



Rysunek 1. Rozmieszczenie wyprowadzeń dwóch złącz wyświetlacza TID stosowanych w najbardziej popularnych modelach Opla

i cyfry, wyświetlacz alfanumeryczny), które były dostępne dla radioodbiornika plus dodatkowe piktogramy obrazujące tryb pracy radia. Nowsze pracują w organizacji 1×10 znaków (plus piktogramy, pełne ASCII, wyświetlacz mozaikowy 5×7 pikseli) w dolnym wierszu, zaś górny wiersz zarezerwowano dla wbudowanego systemu mikroprocesorowego (zegar, termometr). Dodatkowo, przewidziano możliwość synchronizacji wbudowanego weń zegara czasu rzeczywistego sygnałem RDS radioodbiornika. Niestety, od roku 2005/2006 firma Opel zdecydowała się (wzorem innych producentów w branży motoryzacyjnej) na zastosowanie magistrali CAN we wszystkich nowych pojazdach osobowych, co pociągnęło potrzebę implementacji tego interfejsu także w wyświetlaczu LCD. Niemniej jednak, w zdecydowanej większości popularnych modeli pojazdów tej marki znajdziemy wyświetlacze, które z powodzeniem mogą być zastąpione przez opisywane urządzenie. Oczywiście, w zależności od typu pojazdu i rodzaju zastosowanego wyświetlacza, stosowano różne złącza. Na **rysunku 1** zamieszczono rozkład wyprowadzeń dwóch złącz wyświetlacza TID stosowanych w najbardziej popularnych modelach Opla, tj. Astra F/G i Corsa B/C. W **tabeli 2** umieszczono opis wyprowadzeń tych złącz. Paleta stosowanych złącz jest znacznie szersza, w związku z czym zainteresowanych czytelników odsyłam na bardzo ciekawą stronę poświęconą tematyce car-audio, znajdującą się pod adresem ***https://goo.gl/NaAfwS***.

Wyświetlacz jest systemem mikroprocesorowym wyposażonym w funkcje dodatkowe (oprócz możliwości wyświetlania danych z magistrali), takie jak: wbudowany kalendarz, zegar czy też termometr. Kolejnym ciekawym zagadnieniem jest rodzaj magistrali sterującej.

Tabela 2. Opis wyprowadzeń dwóch złącz wyświetlacza TID stosowanych w najbardziej popularnych modelach Opla

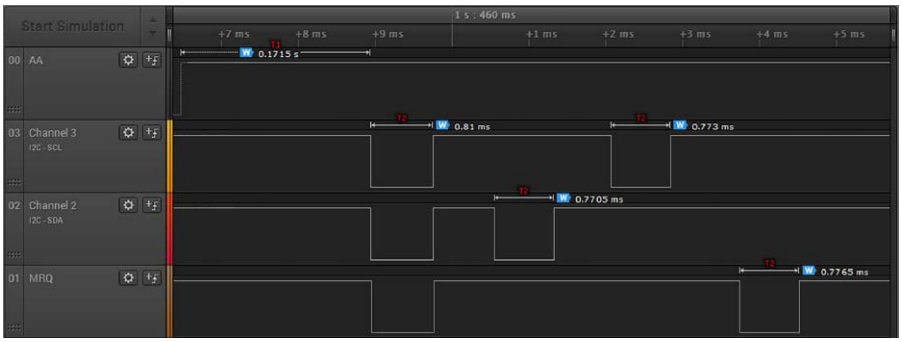
Astra G/Corsa C	
Nr pinu	**Funkcja/Opis**
1	Zasilanie: 12V po przekręceniu kluczyka stacyjki
2	AA – 12V po włączeniu radioodbiornika
3	Zasilanie: 12V z akumulatora
4	Regulacja jasności podświetlenia wyświetlacza (sygnał PWM)
5	NTC – wyprowadzenie do podłączenia czujnika temperatury zewnętrznej
6	Masa
7	NTC – wyprowadzenie do podłączenia czujnika temperatury zewnętrznej
8	Wyprowadzenie testowe – do sprzętu diagnostycznego
9	WEG – sygnał prędkości pojazdu (korekta wskazań termometru)
10	SCL - sygnał sterujący magistrali – Serial Clock
11	SDA - sygnał sterujący magistrali – Serial Data
12	MRQ - sygnał sterujący magistrali – Master Request
Astra F/Corsa B/Tigra	
Nr pinu	**Funkcja/Opis**
1	Zasilanie: 12V z akumulatora
2	NTC – wyprowadzenie do podłączenia czujnika temperatury zewnętrznej
3	Masa
4	NTC – wyprowadzenie do podłączenia czujnika temperatury zewnętrznej
5	Zasilanie: 12V po przekręceniu kluczyka stacyjki
6	Oświetlenie
7	Regulacja jasności podświetlenia wyświetlacza (sygnał PWM)
8	AA – 12V po włączeniu radioodbiornika
9	SCL - sygnał sterujący magistrali – Serial Clock
10	MRQ - sygnał sterujący magistrali – Master Request
11	SDA - sygnał sterujący magistrali – Serial Data
12	WEG – sygnał prędkości pojazdu (korekta wskazań termometru)

Jest ona bardzo zbliżona funkcjonalnością do dobrze znanej magistrali I²C, lecz poszerzona o dodatkowy sygnał sterujący – MRQ (Master Request). Na tę magistralę składają się 4 linie oznaczonych: AA, MRQ, SDA i SCL, przy czym wyłącznie trzy ostatnie realizują rzeczywistą transmisję danych, zaś linia AA sygnalizuje zdarzenie załączenia/wyłączenia radioodbiornika przygotowując tym samym wbudowany wyświetlacz do odbierania danych. Po wyłączeniu radioodbiornika napięcie na tej linii wynosi 0 V, zaś po jego załączeniu 12 V. Linia MRQ jest używana przez układ master (radioodbiornik) do sprawdzenia obecności układu slave (wyświetlacza TID) na magistrali danych oraz do rozróżnienia rodzaju przesyłanych danych, o czym później. Zanim jednak radioodbiornik rozpocznie jakąkolwiek transmisję danych sprawdza, tuż po włączeniu jego zasilania, stan wszystkich linii transmisyjnych wykonując tzw. *Power On Test*, dla którego przebiegi sygnałów sterujących pokazano na **rysunku 2**, a którego celem jest sprawdzenie ciągłości wszystkich linii danych jak i wykluczenie potencjalnych zwarć.

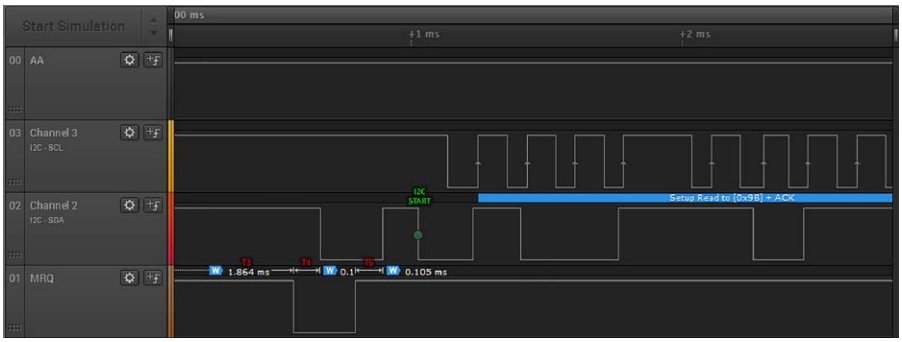
Tuż po włączeniu zasilania radioodbiornika, któremu to towarzyszy zmiana poziomu napięcia występującego na linii AA z 0 V na 12 V, następuje po czasie T1 (100…500 ms) ściągnięcie wszystkich linii danych (MRQ, SDA i SCL) do masy na czas T2 (500…1000 µs), następnie zwolnienie ich (także na czas T2) oraz naprzemienne ściągnięcie do masy kolejnych linii SDA, SCL i na końcu MRQ. Taka sekwencja zdarzeń pozwala, co napisano wcześniej, na zbadanie stanu wszystkich linii sterujących i zakończenie transmisji w przypadku wykrycia potencjalnych problemów. Jeśli powyższa procedura zakończy się powodzeniem (brak zwarć i nieciągłości linii danych), master przechodzi do właściwej transmisji danych, dla której to przebiegi sygnałów sterujących pokazano na **rysunku 3.**

Po czasie T3 (1…2 ms) od ostatniego zwolnienia linii MRQ będącego jednocześnie końcowym elementem procedury *Power On Test*, układ Master ponownie zeruje linię MRQ, tym razem jednak z zupełnie innego powodu. Po pierwsze, sygnalizuje w ten sposób zamiar transmisji danych, a po drugie, sprawdza obecność slave na magistrali danych oraz jego gotowość na odbieranie pakietu danych, gdyż działanie takie podejmowane jest każdorazowo na początku każdej transmisji z radioodbiornika do wyświetlacza TID. Po wymuszeniu poziomu niskiego na linii MRQ, slave musi w czasie T4 (100 µs…15 ms) zewrzeć linię SDA do masy sygnalizując w ten sposób swoją obecność, jak i gotowość do odbierania danych. Jeśli tego nie zrobi, master powinien zakończyć w tym miejscu bieżącą transmisję danych. Dalej, wyzerowaniu linii SDA przez slave, master zwalnia linię MRQ (zostaje ona ustawiona), co powinno skutkować zwolnieniem linii danych SDA przez slave po czasie T5 (100…200 µs). Na tym etapie rozpoczyna się właściwa transmisja danych zgodna ze specyfikacją standardu I²C, z jednym, drobnym, acz istotnym wyjątkiem, o którym mowa będzie w dalszej części artykułu.

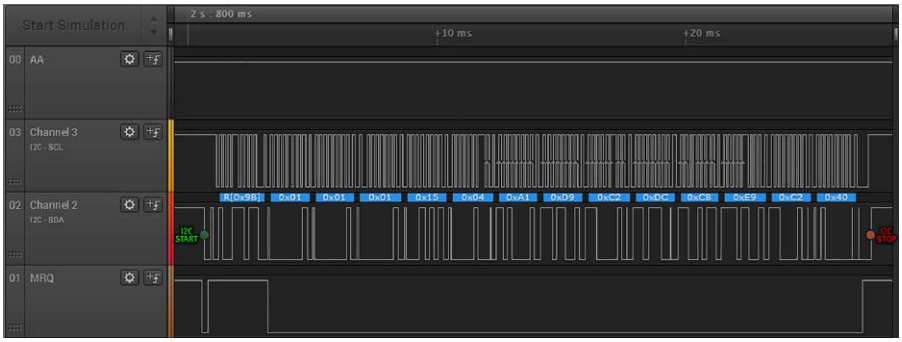
Wygląd kompletnej ramki danych pokazano na **rysunku 4**. Transmisja rozpoczyna się od wygenerowania sekwencji *Start* przez master'a, po czym jako pierwszy bajt transmitowany jest adres slave o wartości 0x4D (0x9B w notacji 8-bitowej z ustawionym bitem R/W). Właśnie w tym miejscu jest widoczna subtelna różnica pomiędzy standardowym interfejsem zgodnym ze standardem I²C, a rozwiązaniem zastosowanym przez inżynierów Opla. Skoro transmitowany bajt adresu ma ustawiony najmniej znaczący bit będący jednocześnie bitem R/W (odczyt/zapis) w standardzie I²C powinno to oznaczać, że master (radioodbiornik) zgłasza chęć odczytania danych ze slave (wyświetlacza TID) i kolejne dane, które zostaną przesłane będą danymi przesyłanymi ze slave do master, a jest zgoła inaczej, ponieważ master kontynuuje w tym miejscu transmisję danych do slave, tak jakby najstarszy bit przesłanego wcześniej adresu był wyzerowany. Z czego wynika ta drobna, acz znacząca różnica? Odpowiedź jest prosta. Z chęci maksymalnego zabezpieczenia się przed możliwością wystąpienia zaburzeń transmisji i będących tego następstwem, niepoprawnych danych, o co, jak łatwo się domyślić, nietrudno w tak niesprzyjającym środowisku, jak instalacja samochodowa. Otóż, zgodnie z pomysłem inżynierów Opla, każdy bajt danych składa się z 7 bitów właściwych danych oraz bitu kontroli parzystości umieszczonego na pozycji najmniej znaczącej. I niestety, dotyczy to wszystkich bajtów danych, w tym adresu urządzenia, co powoduje, że dla adresu 0x4D (w notacji 7-bitowej) otrzymujemy bit kontroli parzystości o wartości „1", czyli ustawiony bit R/W. Różnica jest drobna, lecz niesie za sobą poważne konsekwencje implementacyjne. Otóż, w przypadku takiej konstrukcji ramki danych, do obsługi transmisji danych nie możemy użyć wygodnego, sprzętowego interfejsu TWI (odpowiednik I²C)



Rysunek 2. Przebiegi sygnałów sterujących magistrali danych w trakcie sekwencji Power On Test



Rysunek 3. Przebiegi sygnałów sterujących magistrali danych po wykonaniu sekwencji Power On Test



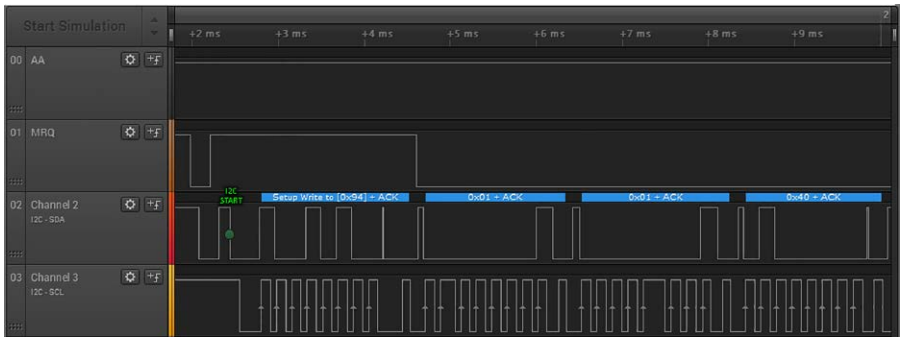
Rysunek 4. Kompletna ramka danych

REKLAMA

mikrokontrolera ATmega644PA, którego zastosowanie znacznie uprościłoby mechanizmy emulujące wyświetlacz TID. Zwyczajnie, i co oczywiste, sprzęg TWI nie przewiduje sytuacji, w której to master wysyła żądanie odczytu do slave (wysyła jego adres z ustawionym bitem R/W), po czym najzwyczajniej w świecie kontynuuje transmisję danych w kierunku do slave. Takiego scenariusza transmisji w module TWI (i stosownych statusów jego stanu w rejestrze TWSR mikrokontrolera) po prostu nie przewidziano, gdyż jest on zgodny ze standardem I²C. Czy to stanowi jakiś problem? W zasadzie nie, lecz z pewnością skomplikuje mechanizm obsługi interfejsu Opla, gdyż do obsługi transmisji I²C będziemy musieli zaprzęgnąć inne zasoby sprzętowe mikrokontrolera, w tym przypadku przerwania zewnętrzne oraz zaimplementować niezbędne procedury programowe, co na pewno przyczyni się do większego obciążenia rdzenia.

W tym miejscu należy wspomnieć o jeszcze jednej funkcji linii MRQ, którą można zaobserwować analizując przebiegi kompletnej ramki danych z rys. 4. Otóż, linia danych MRQ jest wyzerowana w trakcie przesyłania przez radioodbiornik wszystkich bajtów danych za wyjątkiem bajta adresu, co w prosty sposób odróżnia ten bajt od pozostałych bajtów danych. Takie rozwiązanie stanowi zapewne kolejny element poprawy integralności przesyłanej ramki danych przyczyniający się do zwiększenia marginesu bezpieczeństwa, bo identyfikacja bajta adresu mogłaby być wykonana przecież w prostszy sposób, choćby przez fakt, że jest to zawsze pierwszy, przesyłany bajt danych. Niemniej jednak, rozwiązanie Opla z pewnością należy uznać za bardzo przemyślane. Transmisję właściwych danych, jak to zwykle ma miejsce w standardzie I²C, kończy sygnał *stop* wygenerowany przez master.

Jak do tej pory nie powiedziałem nic na temat samych danych, które to radioodbiornik wysyła do wyświetlacza pokładowego TID. Dane te to 13 bajtów, z których pierwsze trzy bajty odpowiadają za wyświetlanie piktogramów pokazywanych na ekranie wyświetlacza TID, zaś kolejne 10 bajtów to kody znaków ASCII przeznaczonych do wyświetlenia w polu tekstowym, przy czym



Rysunek 5. Wygląd ramki danych wyświetlacza TID w wypadku nieobsługiwanego adresu I²C

Tabela 3. Znaczenie poszczególnych bitów trzech pierwszych bajtów danych wyświetlacza TID

	Bajt 1 (status radia)	Bajt 2 (status magnetofonu)	Bajt 3 (status CD)
Bit 7	przecinek	„CD-In”	0
Bit 6	„RDS”	„Dolby C”	Symbol „Track”
Bit 5	„TP”	„Dolby B”	„RDM”
Bit 4	Symbol „Stereo”	„Cr”	„PGM”
Bit 3	0	„CPS”	„DISC”
Bit 2	„AS”	0	0
Bit 1	Nawias kwadratowy dla „TP”	0	0
Bit 0	Bit parzystości	Bit parzystości	Bit parzystości

obsługiwanych jest kilka znaków specjalnych o kodach poniżej wartości 32 (spacji). W **tabeli 3** opisano znaczenie poszczególnych bitów trzech pierwszych bajtów danych odpowiedzialnych za wyświetlanie specjalnych piktogramów na ekranie wyświetlacza TID. W **tabeli 4** umieszczono kody ASCII i opis znaków specjalnych wyświetlanych przez oryginalny wyświetlacz TID a obsługiwane przez sterownik multiTID. Wcześniejszy opis dotyczy wyświetlaczy o organizacji 10-znakowej. Dla starszych wyświetlaczy, o organizacji 8-znakowej, jest przesyłanych wyłącznie 10 bajtów danych, z czego 2 pierwsze bajty odpowiadają za wyświetlanie piktogramów pokazywanych na ekranie wyświetlacza TID (tylko status radia i magnetofonu), zaś kolejne 8 bajtów to kody znaków ASCII przeznaczonych do wyświetlenia w polu tekstowym wyświetlacza (tylko wielkie litery i cyfry, bez znaków specjalnych). Inny jest też adres samego wyświetlacza, który przyjmuje wtedy wartość 0x4A. Już tylko dla porządku dodam, że każdy bajt danych przesłany przez master do slave jest potwierdzany przez układ podrzędny (TID) poprzez wygenerowanie sygnału ACK (wyzerowanie linii SDA) w dziewiątym takcie zegara magistrali (SCL) i to niezależnie czy przesyłane dane dotyczą naszego urządzenia (przesłano wcześniej zgodny adres I²C), czy też nie, co również wymyka się standardom wyznaczonym przez specyfikację interfejsu I²C. Sytuację taką przedstawiono na **rysunku 5**, który to prezentuje przebiegi na magistrali danych zarejestrowane podczas współpracy radioodbiornika CAR300

Tabela 4. Kody ASCII znaków specjalnych wyświetlanych przez oryginalny, 10-znakowy wyświetlacz TID

Kod ASCII	Znak
1	Pomniejszona cyfra 1
2	Pomniejszona cyfra 2
3	Pomniejszona cyfra 3
4	Pomniejszona cyfra 4
5	Pomniejszona cyfra 5
6	Pomniejszona cyfra 6
8	Pomniejszona, wielka litera A
9	Pomniejszona, wielka litera F
10	Pomniejszona, wielka litera U
11	Pomniejszona, wielka litera M
12	Pomniejszona, wielka litera L
13	Pomniejszona, wielka litera P
14	Symbol “nutki”

Listing 1. Treść pliku nagłówkowego emulatora wyświetlacza TID

```
//Zmienne globalne modułu TIDemulator
extern volatile uint8_t TIDdataReady;  //Flaga gotowości danych
extern volatile uint8_t TIDdata[14];  //Dane (element 0 to adres I2C)

//Prototypy funkcji
void initTIDemulator(uint8_t TID_type);
//Definicje portów TID'a
#define I2C_SDA_DDR DDRC
#define I2C_SDA_PIN PINC
#define I2C_SDA_NR PC1  //ISR: PCINT17
#define I2C_SCL_PIN PINB
#define I2C_SCL_NR PB2  //ISR: INT2
#define MRQ_PIN PIND
#define MRQ_NR PD3  //ISR: INT1
#define AA_PIN PIND
#define AA_NR PD2  //ISR: INT0
//Adres wyświetlacza TID
#define TID_TYPE_8DIGITS 0x4A
#define TID_TYPE_10DIGITS 0x4D
#define TID_TIMESTAMP 0x47
//Stany pracy interfejsu TID
#define RADIO_IS_OFF 0x00
#define WAITING_FOR_FIRST_MRQ 0x01
#define FIRST_MRQ_STARTED 0x02
#define WAITING_FOR_SECOND_MRQ 0x03
#define SECOND_MRQ_STARTED 0x04
#define WAITING_FOR_MRQ_BEFORE_TRANSMISSION 0x05
#define START_DETECTED 0x06
//Definicje czasów dla Preskalera Timera1 = 256 i fosc = 12.288MHz [w taktach timera]
#define TIME_TOLERANCE 14  //300us
#define T2_MIN (24 - TIME_TOLERANCE)  //500us - TIME_TOLERANCE
#define T2_MAX (48 + TIME_TOLERANCE)  //1000us + TIME_TOLERANCE
//Średnie wartości pozostałych czasów [us]
#define T4_MID 120  //120us
#define T5_MID 120  //120us
//Makra dla portów
#define SDA_SET I2C_SDA_DDR &= ~(1<<I2C_SDA_NR) //Ustawiamy, jako wejście podciągnięte przez zewnętrzny rezystor
#define SDA_RESET I2C_SDA_DDR |= (1<<I2C_SDA_NR)  //Ustawiamy, jako wyjście z domyślnym stanem „0”
#define SDA_READ ((I2C_SDA_PIN & (1<<I2C_SDA_NR))>>I2C_SDA_NR)  //Wartość: 0x00 lub 0x01
#define SDA_IS_RESET (!(I2C_SDA_PIN & (1<<I2C_SDA_NR)))
#define SDA_IS_SET (I2C_SDA_PIN & (1<<I2C_SDA_NR))
#define SCL_IS_RESET (!(I2C_SCL_PIN & (1<<I2C_SCL_NR)))
#define SCL_IS_SET (I2C_SCL_PIN & (1<<I2C_SCL_NR))
#define MRQ_IS_RESET (!(MRQ_PIN & (1<<MRQ_NR)))
#define MRQ_IS_SET (MRQ_PIN & (1<<MRQ_NR))
#define AA_IS_SET (AA_PIN & (1<<AA_NR))
//Makra dla przerwań
#define SDA_INTR_INIT_ON_BOTH_EDGES PCMSK2 = (1<<PCINT17)  //Zmiana stanu na PCINT17 (sygnał SDA) generuje przerwanie
#define SDA_INTR_ENABLE PCICR |= (1<<PCIE2)  //Zezwolenie na przerwanie PCINT2 (PCINT23..16)
#define SDA_INTR_DISABLE PCICR &= ~(1<<PCIE2)  //Zablokowanie przerwania PCINT2 (PCINT23..16)
#define SDA_INTR_CLEAR_FLAG PCIFR |= (1<<PCIF2)  //Skasowanie flagi przerwania PCINT2 (PCINT23..16)
#define SDA_INTR_NAME PCINT2_vect  //Nazwa wektora przerwania PCINT2 (SDA)
#define SCL_INTR_INIT_ON_RISING_EDGE EICRA |= (1<<ISC21)|(1<<ISC20)  //Rosnące zbocze sygnału na INT2 (sygnał SCL) generuje przerwanie
#define SCL_INTR_ENABLE EIMSK |= (1<<INT2)  //Zezwolenie na przerwanie INT2
#define SCL_INTR_DISABLE EIMSK &= ~(1<<INT2)  //Zablokowanie przerwania INT2
#define SCL_INTR_CLEAR_FLAG EIFR |= (1<<INTF2)  //Skasowanie flagi przerwania INT2
#define SCL_INTR_NAME INT2_vect  //Nazwa wektora przerwania INT2 (SCL)
#define MRQ_INTR_INIT_ON_BOTH_EDGES EICRA |= (1<<ISC10)  //Dowolne zbocze sygnału na INT1 (sygnał MRQ) generuje przerwanie
#define MRQ_INTR_ENABLE EIMSK |= (1<<INT1)  //Zezwolenie na przerwanie INT1
#define MRQ_INTR_DISABLE EIMSK &= ~(1<<INT1)  //Zablokowanie przerwania INT1
#define MRQ_INTR_CLEAR_FLAG EIFR |= (1<<INTF1)  //Skasowanie flagi przerwania INT1
#define MRQ_INTR_NAME INT1_vect  //Nazwa wektora przerwania INT1 (MRQ)
#define AA_INTR_INIT_ON_BOTH_EDGES EICRA |= (1<<ISC00)  //Dowolne zbocze sygnału na INT0 (sygnał AA) generuje przerwanie
#define AA_INTR_ENABLE EIMSK |= (1<<INT0)  //Zezwolenie na przerwanie INT0
#define AA_INTR_DISABLE EIMSK &= ~(1<<INT0)  //Zablokowanie przerwania INT0
#define AA_INTR_CLEAR_FLAG EIFR |= (1<<INTF0)  //Skasowanie flagi przerwania INT0
#define AA_INTR_NAME INT0_vect  //Nazwa wektora przerwania INT0 (AA)
```

z nowym, 10-znakowym wyświetlaczem pokładowym TID.

Uważny Czytelnik zastanowi się z pewnością, w jakim celu pokładowy wyświetlacz TID miałby potwierdzać każdy bajt danych, które nie są do niego adresowane, to znaczy nie zostały „okraszone” stosownym adresem układu podrzędnego (w naszym przypadku 0x4D). To kolejny przykład przemyślanej konstrukcji Opla. Otóż wiele fabrycznych radioodbiorników może pracować tak w starszych, jak i nowszych modelach pojazdów tego producenta, które to mogą być wyposażone w różne rodzaje wyświetlaczy TID. Aby umożliwić współpracę radioodbiornika z nowymi i starszymi typami wyświetlaczy, które to przecież charakteryzują się inną organizacją ekranu i odrębnym adresem w przestrzeni I²C, wprowadzono zasadę, iż radioodbiornik wysyła pakiety danych dla starego i nowego typu wyświetlaczy, co oczywiste, stosownie je modyfikując. W związku z tym, każdy przesyłany komunikat transmitowany jest w dwóch „wersjach”, dla starego i nowego typu wyświetlacza, stąd różne adresy i różna liczba towarzyszących im danych na jednej i tej samej magistrali. Nie tłumaczy to oczywiście faktu potwierdzania (sygnałem ACK) przez wyświetlacz TID nie swojego adresu I²C, jak i danych nie dla niego przeznaczonych, lecz myślę, że wynika to z potrzeby ciągłego „informowania” (przy użyciu sygnału ACK) radioodbiornika o nasłuchu magistrali I²C przez układ podrzędny. Jak widać, zastosowana przez Opla magistrala, na pozór bardzo podobna do rozwiązania Philips'a (I²C), ma wiele drobnych, acz znaczących różnic, przez które trudno byłoby ją „obsłużyć” korzystając ze sprzętowego sprzęgu TWI.

Na koniec prawdziwa „wisienka na torcie”. Jak wiadomo, oryginalny wyświetlacz TID wyposażono w zegar czasu rzeczywistego, którego wskazania są wyświetlane w pierwszej linii ekranu (obok

REKLAMA

temperatury zewnętrznej). W związku z tym producent modułu wyświetlacza przewidział możliwość synchronizacji wskazań tego zegara sygnałem czasu nadawanym przez stacje radiowe, a zawartym w treści wybranych komunikatów RDS. W jaki sposób wyświetlacz TID synchronizuje swój zegar RTC sygnałem czasu radioodbiornika? Wprowadzono dodatkowy adres I²C wyświetlacza TID, dla którego przesyłane dane są interpretowane przez ten moduł jako dane bieżącego czasu sygnału RDS. Adres ten to 0x47 (0x8F przy ustawionym bicie R/W), natomiast odpowiednia ramka danych składa się z 7 bajtów o następującej organizacji: X-M-G-D-X-X-X, gdzie: M to minuty, G to godziny, D to dzień miesiąca. Bajty X nie zostały przeze mnie jednoznacznie zidentyfikowane, choć z całą pewnością ich obecność nie jest przypadkowa. Jeśli bajty te przechowują bieżący miesiąc i rok, jak się zapewne domyślacie, czas oczekiwania na potwierdzenie tej tezy byłby dość długi. Wygląd ramki danych znacznika czasu wyświetlacza TID zarejestrowanej 17/08/2016 o godzinie 17:49 przedstawiono na **rysunku 6**. Jak poprzednio, bit najmniej znaczący jest bitem kontroli parzystości, w związku z czym cały bajt należy przesunąć w prawo o jedno miejsce).

To tyle, jeśli chodzi o opis modułu wyświetlacza pokładowego. Przejdźmy zatem do szczegółów implementacyjnych. Jako, że nasze urządzenie multiTID realizuje szereg dość skomplikowanych funkcjonalności, które wymagają ścisłych zależności czasowych i zaangażowania wielu peryferiów mikrokontrolera, obsługa emulacji wyświetlacza TID (czyli obsługa radioodbiornika) została zrealizowana z wykorzystaniem 4 zewnętrznych przerwań systemowych. Taka mnogość wykorzystywanych przerwań wynika z faktu obsługi aż 4 sygnałów sterujących (AA, MRQ, SCL, SDA) przy braku możliwości realizacji sprzętowej obsługi magistrali I²C. W **tabeli 5** pokazano zestawienie wykorzystywanych, zewnętrznych

Listing 2. Funkcja ISR odpowiedzialna za obsługę sygnału AA interfejsu danych wyświetlacza TID

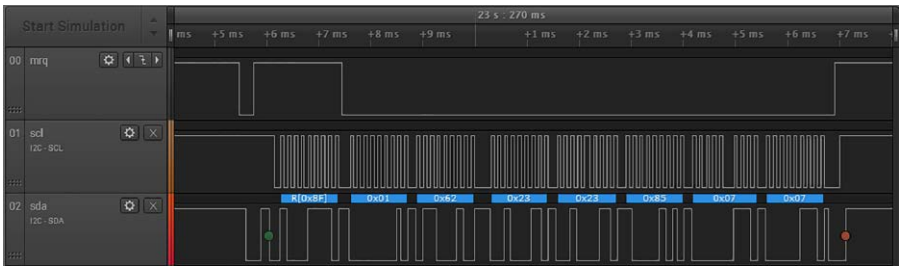
```
//Przerwanie odpowiedzialne za obsługę sygnału AA - wyzwalane po każdym zboczu sygnału
na AA
ISR(AA_INTR_NAME)
{
    if(AA_IS_SET)  //Włączenie radioodbiornika
    {
        Status = WAITING_FOR_FIRST_MRQ;
        MRQ_INTR_INIT_ON_BOTH_EDGES;  //Konfiguracja i włączenie przerwania dla MRQ
        MRQ_INTR_CLEAR_FLAG;
        MRQ_INTR_ENABLE;
    }
    else  //Wyłączenie radioodbiornika
    {
        Status = RADIO_IS_OFF;  //Startowy stan algorytmu
        SDA_SET;  //Zwolnienie SDA, gdyby było ściągane przez Slave'a
        MRQ_INTR_DISABLE;
        SDA_INTR_DISABLE;
        SCL_INTR_DISABLE;
    }
}
```

Listing 3. Funkcja ISR odpowiedzialna za obsługę sygnału MRQ interfejsu danych wyświetlacza TID

```
//Przerwanie odpowiedzialne za obsługę sygnału MRQ - wyzwalane po każdym zboczu sygnału na MRQ
ISR(MRQ_INTR_NAME)
{
    register uint16_t pulseLength;
    static uint16_t lastTimer1;
    switch(Status)
    {
        case WAITING_FOR_FIRST_MRQ:
            if(SDA_IS_RESET && SCL_IS_RESET && MRQ_IS_RESET) Status = FIRST_MRQ_STARTED; lastTimer1 = TCNT1;
            else Status = WAITING_FOR_FIRST_MRQ;
            break;
        case FIRST_MRQ_STARTED:
            pulseLength = TCNT1 - lastTimer1;
            if(SDA_IS_SET && SCL_IS_SET && MRQ_IS_SET && pulseLength>T2_MIN && pulseLength<T2_MAX) Status = WAITING_FOR_SECOND_MRQ;
            else Status = WAITING_FOR_FIRST_MRQ;
            break;
        case WAITING_FOR_SECOND_MRQ:
            if(SDA_IS_SET && SCL_IS_SET && MRQ_IS_RESET) Status = SECOND_MRQ_STARTED; lastTimer1 = TCNT1;
            else Status = WAITING_FOR_FIRST_MRQ;
            break;
        case SECOND_MRQ_STARTED:
            pulseLength = TCNT1 - lastTimer1;
            if(SDA_IS_SET && SCL_IS_SET && MRQ_IS_SET && pulseLength>T2_MIN && pulseLength<T2_MAX) Status = WAITING_FOR_MRQ_BEFORE_TRANS-
MISSION;
            else Status = WAITING_FOR_FIRST_MRQ;
            break;
        case WAITING_FOR_MRQ_BEFORE_TRANSMISSION:
            if(SDA_IS_SET && SCL_IS_SET && MRQ_IS_RESET)
            {
                //Sekwencja Slave'a: ściągnięcie SDA do „0" po czasie T4
                _delay_us(T4_MID); SDA_RESET;
                //Czekamy, aż Master zwolni MRQ (MRQ = 1)
                while(MRQ_IS_RESET);
                //Sekwencja Slave'a: zwolnienie SDA po czasie T5
                _delay_us(T5_MID); SDA_SET;
                //Czekamy na sekwencję START
                while(SDA_IS_SET);
                //Czekamy, aż Master ściągnie SCL do „0",
                //czyli przygotuje się do wysłania bitu
                while(SCL_IS_SET);
                Status = START_DETECTED;
                //Konfiguracja i włączenie przerwania SCL dla
                //odbioru danych (poszczególnych bitów)
                SCL_INTR_INIT_ON_RISING_EDGE;
                SCL_INTR_CLEAR_FLAG;
                SCL_INTR_ENABLE;
                //Konfiguracja i włączenie przerwania SDA dla
                //wykrycia późniejszej sekwencji STOP
                SDA_INTR_INIT_ON_RISING_EDGE;
                SDA_INTR_CLEAR_FLAG;
                SDA_INTR_ENABLE;
                //Wyczyszczenie flagi MRQ
                MRQ_INTR_CLEAR_FLAG;
            }
            else Status = WAITING_FOR_FIRST_MRQ;
            break;
    }
}
```

Tabela 5. Lista zewnętrznych przerwań systemowych wraz z opisem ich konfiguracji i funkcjonalności niezbędnych dla obsługi magistrali radioodbiornika Opla

Nazwa przerwania	Sposób konfiguracji	Funkcjonalność
INT0	Wyzwalane przy każdym zboczu sygnału na wejściu AA	Wykorzystywane do obsługi sygnału AA w zakresie detekcji włączenia/wyłączenia radioodbiornika
INT1	Wyzwalane przy każdym zboczu sygnału na wejściu MRQ	Wykorzystywane do obsługi sygnału MRQ w zakresie kontroli procesu odbioru ramki danych
INT2	Wyzwalane przy narastającym zboczu sygnału na wejściu SCL	Wykorzystywane do obsługi sygnału SCL w zakresie odczytu bitów danych
PCINT2	Wyzwalane zmianą stanu na wejściu SDA	Wykorzystywane do obsługi sygnału SDA w zakresie wykrywania sekwencji Stop



Rysunek 6. Wygląd ramki danych znacznika czasu wyświetlacza TID

Listing 4. Funkcja ISR odpowiedzialna za obsługę sygnału SDA interfejsu danych wyświetlacza TID

```
//Przerwanie odpowiedzialne za obsługę sygnału SDA - wyzwalane po każdej zmianie stanu
na SDA
ISR(SDA_INTR_NAME)
{
    if(SCL_IS_SET)
    {
        if(SDA_IS_SET)  //Odebrano sekwencję STOP
        {
        //Wyłączamy przerwania SDA i oczekujemy na nową sekwencję MRQ
        //(przed pakietem danych)
            Status = WAITING_FOR_MRQ_BEFORE_TRANSMISSION;
            SDA_INTR_DISABLE;
            //Zerujemy zmienną, bo sygnał STOP mógłby wystąpić przed końcem ramki
            byteIndex = 0;
        }
    }
}
```

Listing 5. Funkcja ISR odpowiedzialna za obsługę sygnału SCL interfejsu danych wyświetlacza TID

```
//Przerwanie odpowiedzialne za obsługę sygnału SCL - wyzwalane przy rosnącym zboczu sygnału
na SCL
ISR(SCL_INTR_NAME)
{
    register uint8_t Bytes = (TIDtype == TID_TYPE_8DIGITS)? 11:14;
    static uint8_t bitIndex, readValue, TID[14];
    readValue = (readValue<<1)|SDA_READ;
    if(++bitIndex == 8)  //Odczytano 8 bitów bieżącego bajta danych
    {
        TIDisr[byteIndex++] = readValue;  //Element nr 0 zawiera adres I2C TID'a
        bitIndex = readValue = 0;
        //Generujemy ACK po każdych, odebranych 8 bitach danych
        while(SCL_IS_SET);  //Czekamy, aż Master ustawi SCL = 0
        SDA_RESET;  //sygnał ACK generowany przez Slave'a
        while(SCL_IS_RESET);  //Czekamy, aż Master ustawi SCL, tj. odbierze ACK
        while(SCL_IS_SET);  //Czekamy, aż Master wyzeruje SCL, czyli przejdzie do następ-
nego bitu
        SDA_SET;  //Zwolnienie SDA przez Slave'a
        SCL_INTR_CLEAR_FLAG;
        SDA_INTR_CLEAR_FLAG;
        //Sprawdzamy, czy mamy komplet bajtów by oczekiwać na sekwencję
        //STOP (niezależnie od rodzaju TID'a)
        if((TID[0]>>1 == TID_TYPE_8DIGITS && byteIndex == 11)
            ||(TID[0]>>1 == TID_TYPE_10DIGITS && byteIndex == 14)
            ||(TID[0]>>1 == TID_TIMESTAMP && byteIndex == 8))
        {
            if((TID[0]>>1) == TIDtype)  //Sprawdzamy, czy to obsługiwany TID
            {
                for(uint8_t i=0; i<Bytes; ++i) TIDdata[i] = TID[i];
                TIDdataReady = 1;
            }
            byteIndex = 0;
            //Wyłączamy przerwanie SCL i oczekujemy na sekwencję STOP
            SCL_INTR_DISABLE;
        }
    }
}
```

przerwań systemowych, ich konfigurację oraz realizowaną funkcjonalność.

W tym momencie posiadamy już niezbędną wiedzę w zakresie sposobu komunikacji radioodbiornika z oryginalnym wyświetlaczem pokładowym TID, w związku z czym pora na szczegóły implementacyjne mechanizmów pozwalających na emulację tego rodzaju wyświetlacza w zakresie interfejsu naszego komputera pokładowego. Zanim jednak przedstawię ciała funkcji zewnętrznych przerwań systemowych realizujących obsługę poszczególnych sygnałów magistrali danych muszę przedstawić plik nagłówkowy, który ułatwia napisanie wspomnianych funkcji a zarazem czyni je bardziej czytelnymi. Treść pliku nagłówkowego, o którym mowa pokazano na **listingu 1**. Napisano go w sposób dość „uniwersalny", aby łatwo można było zmieniać wiele z predefiniowanych nazw i stałych, bez potrzeby edycji funkcji, które to korzystają z jego zawartości. Pora na przedstawienie funkcji obsługi zewnętrznych przerwań systemowych, które łącznie, odpowiedzialne są za obsługę interfejsu danych firmy Opel. Te funkcje pokazano na **listingach** od **2** do **5**. Do kompletu brakuje funkcji inicjalizacyjnej, która ustawia parametry emulowanego wyświetlacza TID oraz aktywuje niezbędne, startowe funkcje ISR – pokazano ją na **listingu 6**.

Przejdźmy zatem do krótkiej analizy przedstawionych mechanizmów programowych. Można zauważyć, że funkcja obsługująca sygnał MRQ sprawdza zależności czasowe podczas sekwencji *Power On Test* i dopiero po ich pozytywnej weryfikacji pozwala na dalszą analizę sygnałów sterujących wykonywaną przez funkcje ISR, odpowiednio dla SDA i SCL. Oczywiście, sam start procesu obsługi magistrali wyświetlacza TID jest możliwy wyłącznie po załączeniu radioodbiornika, czemu towarzyszy wywołanie przerwania dla sygnału AA.

ROBERT WOŁGAJEW, EP

Listing 6. Funkcja inicjalizacji modułu emulatora wyświetlacza TID

```
inline void initTIDemulator(uint8_t TID_
type)
{
    AA_INTR_INIT_ON_BOTH_EDGES;
    AA_INTR_CLEAR_FLAG;
    AA_INTR_ENABLE;
    TIDtype = TID_type;
}
```