Jak oswoić i skłonić do pracy transceiver ISM CC1100, część 4

W artykule znajdują się wskazówki jak wykorzystać i oprogramować układ radiowego transceivera firmy Texas Instruments. Poza tym będzie też garść praktycznych porad dla tych z czytelników, którzy po raz pierwszy stykają się z opisywanym układem. Nabyte wcześniej wiadomości uzupełniamy zestawem procedur wykorzystywanych do prowadzenia transmisji, tym samym możemy przystąpić do realizacji własnych projektów.

Wyjście GDO0 – semafor transmisji i odbioru danych

Zanim przejdziemy do omawiania procedur transmisji i odbioru, poświęćmy chwilę wyprowadzeniom GDO0 i GDO2. Te uniwersalne wyprowadzenia układu CC1100 są dla tego tak ważne, ponieważ ich funkcja może być programowana poprzez rejestr IOCFG0 dla GDO0 i IOCFG2 dla GDO2 (to drugie wyprowadzenie nie jest używane przy opisie, skupię się więc na GDO0). Funkcje, jakie może pełnić to wyprowadzenie zestawiono w tabeli 33 „GDOx signal selection" dokumentacji technicznej. Jak widać jest to pokaźna lista: od sygnalizacji stanu układu CC1100, sygnalizacji zapełnienia buforów nadawczego i odbiorczego, po zaprogramowanie tego wyjścia jako monitora jakości odbieranego sygnału, czy wyjścia sygnału wewnętrznego zegara.

Jednak najbardziej interesująca jest opcja, która staje się aktywna po zapisaniu do rejestru IOCFG0 wartości 0x06. Od tego momentu wyjście będzie sygnalizowało fakt transmisji i odebrania danych, a dokładniej:

– w czasie transmisji wyjście GDO0 będzie miało po-

List. 11. Procedura *Rf_Send_Packet* i jej przykładowe wywołanie

```
#include  <CC1100_ini.h>

void Rf_Send_Packet(unsigned char *txBuffer, unsigned char size);

void main(void)
{
 unsigned char wysylany_tekst_tab[] ={"Transmisja CC1100"};
 unsigned char tabela_pomocnicza[64], ile_bajtow;

 RfWriteRfSettings(&rfSettings);//inicjacja rejestrów
 //po inicjacji przełączenie układu w tryb odbioru
 SpiWriteCommand(CCxxx0_SIDLE);
 SpiWriteCommand(CCxxx0_SRX);

 //główna nieskończona pętla programu
 while(1)
 {
 //oczekiwanie na naciśnięcie przycisku zwierającego port do masy
  if (SW_KL ==0)
  {
   ile_bajtow =sizeof( wysylany_tekst_tab); //ustalania rozmiaru transmisji
   //przepisywanie wysyłanego tekstu do tabeli pomocniczej
   for (x=0; x< ile_bajtow; x++)
   {
    tabela_pomocnicza[x+1] = wysylany_tekst_tab[x];
   }
   //wpisywanie na pierwszą pozycję bajtu długości transmisji
    tabela_pomocnicza[0] = ile_bajtow;
   //wywołanie procedury transmisji z parametrami: wskaźnikiem do miejsca gdzie znajdują się
   //dane do wysłania, bajtem ilości wysyłanych danych + 1 bajt długości transmisji
   Rf_Send_Packet(&tabela_pomocnicza[0], ile_bajtow+1);
   //po zakończeniu transmisji układ może automatycznie wrócić do trybu odbioru
   SpiWriteCommand(CCxxx0_SIDLE);
   SpiWriteCommand(CCxxx0_SRX);
  }
 }
}

//transmisja pakietu danych
//---------------------------------
void Rf_Send_Packet(unsigned char *txBuffer, unsigned char size)
{
 #define POUT_10dBm  0xC0 //przykładowa deklaracja etykiety dla wartości mocy wyjściowej 10dBm
 unsigned char moc_wyjsciowa;

 moc_wyjsciowa = POUT_10dBm;

 SpiWriteCommand(CCxxx0_SIDLE);
 SpiWriteBurstReg(CCxxx0_TXFIFO, txBuffer, size);
 SpiWriteBurstReg(CCxxx0_PATABLE, &moc_wyjsciowa, 1);
 SpiWriteCommand(CCxxx0_STX);
 //oczekiwanie na stan wysoki na wyjściu GDO -> sync transmitted
 while (GDO_RF_PIN !=1);
 LED_TRANS_PORT =0;
 //oczekiwanie na stan niski wyjścia GDO -> end of packet
 while (GDO_RF_PIN ==1);
 LED_TRANS_PORT =1;
}
```

ziom wysoki, gdy rozpocznie się wysyłanie sygnału synchronizacji i z powrotem przyjmie poziom niski, gdy zostanie wysłana cała zawartość bufora nadawczego wraz z dodatkowymi bajtami pomocniczymi tworzącymi tzw. pakiet,

– w czasie odbioru wyjście GDO0 przyjmie poziom wysoki, gdy zostanie odebrany sygnał będący sygnałem synchronizacji i powróci do poziomu niskiego po odebraniu całego pakietu transmisji.

Dzięki temu do śledzenia bieżącego stanu transmisji nadawczej i odbiorczej wystarczy jedynie obserwacja wyjścia GDO0. W większości sytuacji jest to najwygodniejsze rozwiązanie zwalniające z konieczności ciągłego odczytu rejestrów statusu, czy badania stanu zapełnienia buforów transmisji i odbioru. Taką funkcję wyprowadzenia GDO0 przyjmuje program „Smart RF Studio" jako domyślną. Jeżeli więc w czasie inicjalizacji rejestrów do rejestru IOCFG0 została wpisana inna wartość, należy ją zmienić na 0x06. W przeciwnym wypadku opisane dalej procedury transmisji i odbioru nie zadziałają.

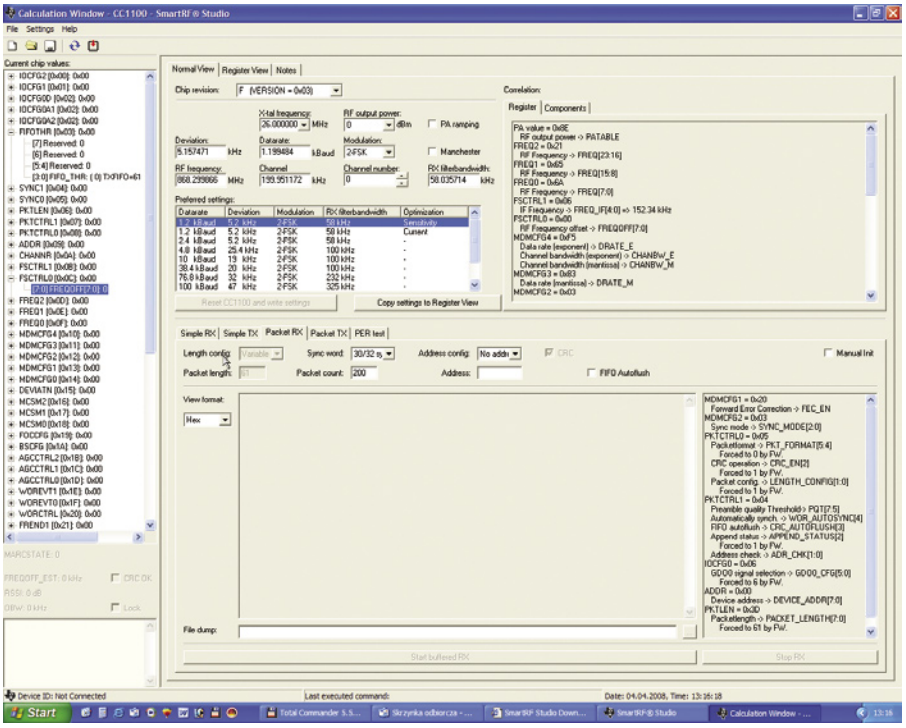
List. 12. Procedura *Rf_Rec_Packet* realizująca odbiór danych

```
#include  <CC1100_ini.h>

unsigned char Rf_Rec_Packet(unsigned char *rxBuffer);

void main(void)
{
 unsigned char odebrana_transmisja_tab[64], ile_bajtow;

 RfWriteRfSettings(&rfSettings);//inicjacja rejestrów
 //po inicjacji przełączenie układu w tryb odbioru
 SpiWriteCommand(CCxxx0_SIDLE);
 SpiWriteCommand(CCxxx0_SRX);
 //główna nieskończona pętla programu
 while(1)
 {
  if (GDO_RF_PIN ==1)
  {
   //początek odbioru danych torem radiowym
   ile_bajtow  =Rf_Rec_Packet(&odebrana_transmisja_tab[0]);
  }
 }
}

//odczyt danych z modułu radiowego
//--------------------------------
unsigned char ModRxStart( unsigned char *rxBuffer)
//we:  *rxBuffer wskaźnik do buforu gdzie będzie zapisana odebrana transmisja
//wy: ilość odebranych danych, 0- błędna transmisja
{
 unsigned char status[2], packet_length;
 unsigned char x;

 LED_REC_PORT =0;
 // oczekiwanie na stan niski wyjścia DGO0 sygnalizujący zakończenie odbioru
 while (GDO_RF_PIN);
 //odczyt rejestru statusu
 SpiReadBurstReg(CCxxx0_RXBYTES, &packed_length, 1);
 packet_length =packet_length & 0x7F;
 if (packet_length !=0)//odczytano co najmniej 1 bajt
 {
  //odczyt pierwszego przesłanego bajtu zawierającego ilość przesłanych danych
  //w celu usunięcia go z buforu odbiorczego
  packet_length =SpiReadReg(CCxxx0_RXFIFO);
  if (packet_length >0 && packet_length <=64)
  {
   //przepisanie danych z buforu odbiorczego CC1100 do buforu w pamięci mikrokontrolera
   SpiReadBurstReg(CCxxx0_RXFIFO, rxBuffer, packet_length);
   //odczyt 2 bajtów statusu status[0]=RSSI, status[1] =LQI
   SpiReadBurstReg(CCxxx0_RXFIFO, status, 2);
   if ((status[LQI] & 0x80) ==0) packet_length =0;//błąd sumy CRC przesłanego pakietu
  }
  else  packet_length =0;
 }
 if (packet_length ==0)
 {
  // w przypadku błędu czyszczenie buforu odbiorczego rozkazem SFRX
  SpiWriteCommand(CCxxx0_SFRX);
 }
 SpiWriteCommand(CCxxx0_SIDLE);
 SpiWriteCommand(CCxxx0_SRX);
 LED_REC_PORT =1;

 //procedura zwraca długość odebranej transmisji, jeśli packet_length=0 transmisja była błędna
return  packet_length;
}
```

Procedura transmisji

Cała procedura transmisji sprowadza się do kilku elementarnych kroków:

1. Układ CC1100 należy wprowadzić w tryb bezczynności komendą SIDLE.
2. Do bufora transmisji należy przesłać dane, które mają być wysłane z tym, że:
   a) na pierwszym miejscu w buforze powinien się znaleźć bajt będący liczbą binarną określającą ilość wysyłanych danych (nie licząc samego bajtu ilości),
   b) z uwagi na to, że bufor transmisji ma rozmiar 64 bajtów, jednorazowo można przesłać 63 bajty danych +1 bajt długości transmisji na początku. Przesyłanie większych partii danych w tym trybie pracy musi być podzielone na kilka oddzielnych transmisji.
3. Do rejestru PATABLE należy wpisać wartość określającą poziom mocy wyjściowej, z jaką będzie wysyłana bieżąca transmisja. Jeżeli poziom mocy wyjściowej zawsze będzie taki sam, można ten punkt pominąć i poprzestać na jednorazowej inicjalizacji rejestru PATABLE na początku programu.
4. Wysłać komendę STX, która inicjuje automatyczne wysyłanie zawartości bufora nadawczego.
5. Obserwować wyprowadzenie GDO0. Gdy wyprowadzenie przyjmie poziom wysoki, a następnie niski, będzie to oznaczać, że cały pakiet został wy-



Rys. 5.

słany i procedura transmisji może się zakończyć.

Na **list. 11** pokazano przykładowe wywołanie procedury transmisji *Rf_Send_Packet* i samą procedurę. Oprócz portów obsługujących magistralę SPI i wyprowadzenie GDO0, w przykładzie użyto jeszcze portu wejściowego, do którego jest dołączony zewnętrzny przycisk „SW_KL", jak również portu wyjściowego obsługujący diodę LED sygnalizującą transmisję „LED_TRANS_PORT". Po naciśnięciu przycisku zwierającego port do masy wywoływana jest procedura transmisji przykładowego napisu znajdującego się w tablicy `wysylany_tekst_tab[]`.

Procedura odbioru

Procedura odbioru nie jest bardziej skomplikowana od procedury transmisji. Zakładamy, że układ jest przestawiony w tryb odbioru i oczekuje na transmisję od partnera:

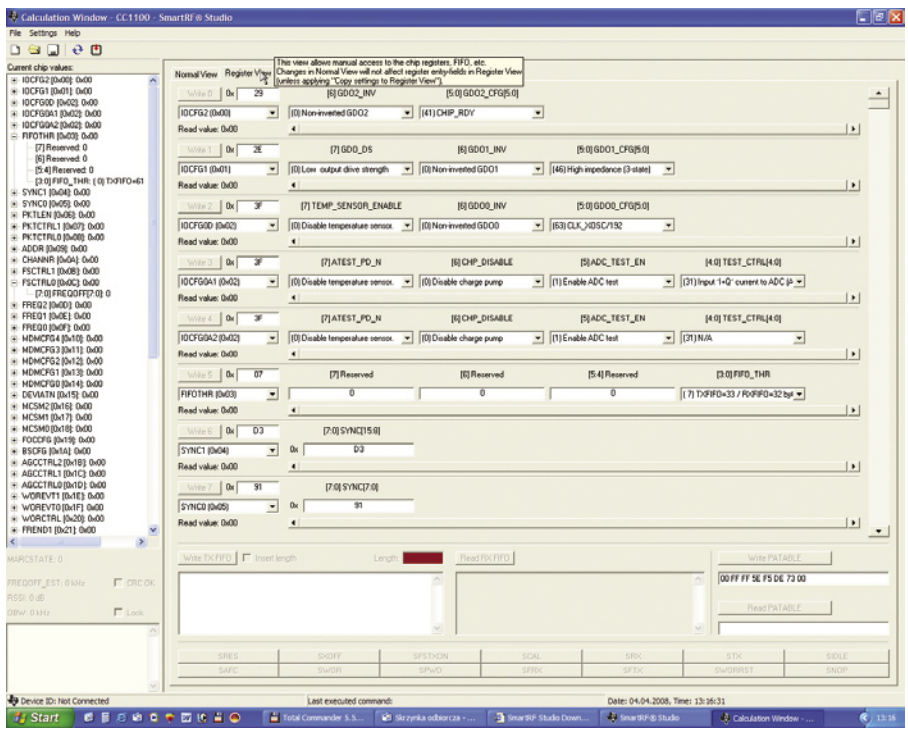
1. Stan wysoki na wyjściu GDO0 oznacza początek odbioru transmisji i wywołanie procedury odbioru.
2. Oczekiwanie na zakończenie odbioru jest sygnalizowane stanem niskim wyjścia GDO0
3. Przeprowadzenie operacji kontrolnych dla stwierdzenia czy odebrana transmisja jest prawidłowa:
   a) sprawdzenie czy odebrano co najmniej 1 bajt danych
   b) sprawdzenie czy w rejestrze statusu jest ustawiona flaga sygnalizująca prawidłową sumę kontrolną w odebranej transmisji
4. W przypadku, gdy operacje kontrolne zakończą się sukcesem można odczytać odebrane dane z bufora odbiorczego układu CC1100 do pamięci mikrokontrolera.

Na **list. 12** pokazano obsługę odbioru transmisji przez procedurę *Rf_Rec_Packet*. Wprowadzona została obsługa dodatkowego portu wyjściowego LED_REC_PORT, który steruje diodę LED sygnalizującą fakt odbioru transmisji.

Uwagi końcowe

Przedstawiony sposób sterowania układem CC1100 nadaje się do transmisji nie dłuższych niż 64 bajty. Do kontroli odbioru i wysyłania danych używane jest wyprowadzenie GDO0. Są to chyba najprostsze rozwiązania, ale w większości wypadków zupełnie wystarczające. Możliwe są oczywiście znacznie bardziej zaawansowane techniki, np. jednorazowe przesyłanie bloków danych wielokrotnie dłuższych od 64 bajtów, kodowanie transmisji, czy usypianie układu dla zmniejszenia średniego poboru prądu. Oznacza to najczęściej wpisanie do pewnych rejestrów innych wartości i zmianę procedur obsługujących transmisję i odbiór. Szczegółowych informacji należy szukać w danych technicznych układu i w przykładach zamieszczonych na stronach producenta.

Ryszard Szymaniak, EP
ryszard.szymaniak@ep.com.pl



Rys. 6.