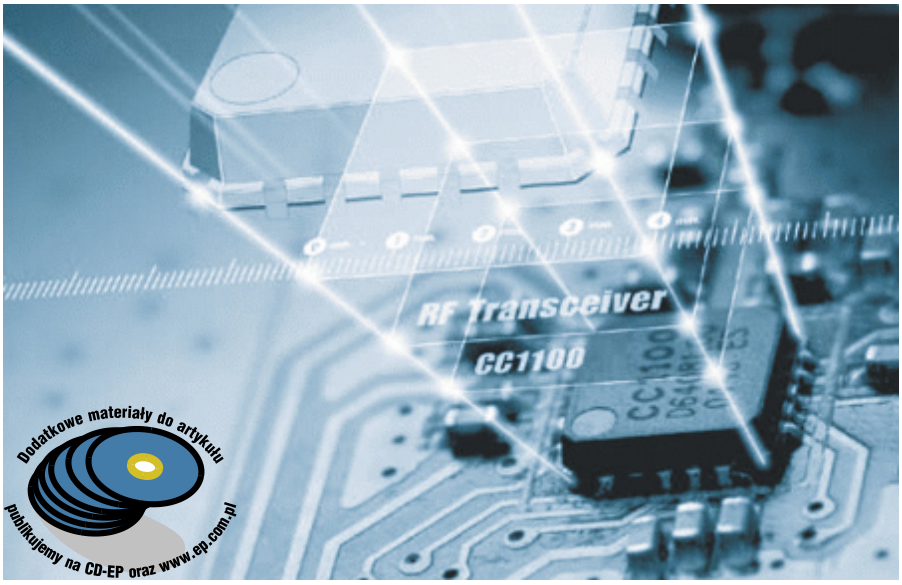
Jak oswoić i skłonić do pracy transceiver ISM CC1100, część 3

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. Wiadomości nabyte w pierwszej części uzupełniamy zestawem procedur wykorzystywanych do prowadzenia transmisji, tym samym możemy przystąpić do realizacji własnych projektów.



Ujarzmianie SmartRF Studio

Po czynnościach wstępnych nadszedł czas zapisania do wewnętrznych rejestrów układu odpowiednich wartości. Trzeba przyznać, że ustalanie wartości rejestrów samodzielnie jest bardzo żmudną pracą, podczas której łatwo o potencjalne błędy. Na szczęście konstruktorzy CC1100 zlitowali się nad użytkownikami i opracowali sympatyczny program, który wykonuje większość pracy automatycznie i rzadko się myli. W pożytecznych dodatkach zostało podane źródło, w którym można pobrać program, i jeżeli czytający ten tekst jeszcze tego nie zrobili, powinni jak najszybciej pobrać i zainstalować plik programu. Po jego uruchomieniu ukazuje się kilka zakładek z nazwami elementów, których oprogramowanie jest wspierane. Nas interesuje zakładka *SmartRF04*, a na niej pozycja *Calculation Window CC1100*. Po jej kliknięciu otwiera się plansza z wieloma okienkami parametrów. Plansza jest rozbudowana, ponieważ program jest przystosowany do współpracy w trybie on–line z fabryczną płytką bazową przeznaczoną do testowania i eksperymentowania z ustawieniami różnych układów radiowych. Nasze wymagania w stosunku do programu są dużo skromniejsze i ograniczą się do wy-

List. 7. Inicjalizacja rejestrów – plik nagłówkowy

```
„CC1100_ini.h”
//deklaracje nazw rejestrów wygenerowane opcją  “*Adress definitions”
#define CCxxx0_IOCFG2    0x00        // GDO2 output pin configuration
#define CCxxx0_IOCFG1    0x01        // GDO1 output pin configuration
#define CCxxx0_IOCFG0    0x02        // GDO0 output pin configuration
#define CCxxx0_FIFOTHR   0x03        // RX FIFO and TX FIFO thresholds
#define CCxxx0_SYNC1     0x04        // Sync word, high byte
// ...
// w tym miejscu powinny znaleźć się deklaracje pozostałych rjestrów
// ...

//deklaracja struktury RF_SETTINGS wygenerowana opcją “RF settings struct typedef”
typedef struct S_RF_SETTINGS{
  unsigned char FSCTRL1; // Frequency synthesizer control.
  unsigned char FSCTRL0; // Frequency synthesizer control.
  unsigned char FREQ2;   // Frequency control word, high byte.
// ...
// w tym miejscu powinna się znaleźć pozostała część deklaracji struktury
// ...
  unsigned char PKTCTRL0;  // Packet automation control.
  unsigned char ADDR;      // Device address.
  unsigned char PKTLEN;    // Packet length.
} RF_SETTINGS;

//inicjacja struktury wygenerowana opcją “RF settings”
RF_SETTINGS rfSettings = {
  0x08,   // FSCTRL1   Frequency synthesizer control.
  0x00,   // FSCTRL0   Frequency synthesizer control.
  0x20,   // FREQ2     Frequency control word, high byte.
  0x28,   // FREQ1     Frequency control word, middle byte.
// ...
// w tym miejscu powinna się znaleźć pozostała część pliku inicjacji
// ...
  0x05,   // PKTCTRL0  Packet automation control.
  0x00,   // ADDR      Device address.
  0xFF    // PKTLEN    Packet length.
};
```

List. 8. Procedura *RfWriteRfSettings* umieszczona w programie głównym, która inicjuje wewnętrzne rejestry układu CC1100

```
//programowanie rejestrów CC1100
//----------------------------------------
void RfWriteRfSettings(RF_SETTINGS *pRfSettings)
{
    // Write register settings
    SpiWriteReg(CCxxx0_FSCTRL1,  pRfSettings->FSCTRL1);
    SpiWriteReg(CCxxx0_FSCTRL0,  pRfSettings->FSCTRL0);
    SpiWriteReg(CCxxx0_FREQ2,    pRfSettings->FREQ2);
    SpiWriteReg(CCxxx0_FREQ1,    pRfSettings->FREQ1);
    SpiWriteReg(CCxxx0_FREQ0,    pRfSettings->FREQ0);
    SpiWriteReg(CCxxx0_MDMCFG4,  pRfSettings->MDMCFG4);
    SpiWriteReg(CCxxx0_MDMCFG3,  pRfSettings->MDMCFG3);
    SpiWriteReg(CCxxx0_MDMCFG2,  pRfSettings->MDMCFG2);
    SpiWriteReg(CCxxx0_MDMCFG1,  pRfSettings->MDMCFG1);
    SpiWriteReg(CCxxx0_MDMCFG0,  pRfSettings->MDMCFG0);
    SpiWriteReg(CCxxx0_CHANNR,   pRfSettings->CHANNR);
    SpiWriteReg(CCxxx0_DEVIATN,  pRfSettings->DEVIATN);
    SpiWriteReg(CCxxx0_FREND1,   pRfSettings->FREND1);
    SpiWriteReg(CCxxx0_FREND0,   pRfSettings->FREND0);
    SpiWriteReg(CCxxx0_MCSM0 ,   pRfSettings->MCSM0 );
    SpiWriteReg(CCxxx0_FOCCFG,   pRfSettings->FOCCFG);
    SpiWriteReg(CCxxx0_BSCFG,    pRfSettings->BSCFG);
    SpiWriteReg(CCxxx0_AGCCTRL2, pRfSettings->AGCCTRL2);
    SpiWriteReg(CCxxx0_AGCCTRL1, pRfSettings->AGCCTRL1);
    SpiWriteReg(CCxxx0_AGCCTRL0, pRfSettings->AGCCTRL0);
    SpiWriteReg(CCxxx0_FSCAL3,   pRfSettings->FSCAL3);
    SpiWriteReg(CCxxx0_FSCAL2,   pRfSettings->FSCAL2);
    SpiWriteReg(CCxxx0_FSCAL1,   pRfSettings->FSCAL1);
    SpiWriteReg(CCxxx0_FSCAL0,   pRfSettings->FSCAL0);
    SpiWriteReg(CCxxx0_FSTEST,   pRfSettings->FSTEST);
    SpiWriteReg(CCxxx0_TEST2,    pRfSettings->TEST2);
    SpiWriteReg(CCxxx0_TEST1,    pRfSettings->TEST1);
    SpiWriteReg(CCxxx0_TEST0,    pRfSettings->TEST0);
    SpiWriteReg(CCxxx0_IOCFG2,   pRfSettings->IOCFG2);
    SpiWriteReg(CCxxx0_IOCFG0,   pRfSettings->IOCFG0);
    SpiWriteReg(CCxxx0_PKTCTRL1, pRfSettings->PKTCTRL1);
    SpiWriteReg(CCxxx0_PKTCTRL0, pRfSettings->PKTCTRL0);
    SpiWriteReg(CCxxx0_ADDR,     pRfSettings->ADDR);
    SpiWriteReg(CCxxx0_PKTLEN,   pRfSettings->PKTLEN);
}
```

generowania gotowych plików, które będzie można wkleić do oprogramowania mikrokontrolera współpracującego z CC1100.

Na szczęście po uruchomieniu program proponuje wiele ustawień wstępnych zupełnie wystarczających do pierwszych eksperymentów i niewiele trzeba samemu zmieniać. Jeżeli częstotliwość kwarcu zastosowanego z naszym CC1100 jest taka jak w okienku „X–tal frequency", pozostawiamy to bez zmian. W przeciwnym wypadku należy wpisać dokładnie wartość częstotliwości kwarcu. Następnie w okienku „RF frequency" należy wpisać wartość podstawowej częstotliwości kanału, na którym ma pracować nasz transceiver, np. 433.0 i nie należy się przejmować jeżeli program trochę zmieni podaną wartość ponieważ będzie to wynikiem jej zoptymalizowania. Można także wybrać z listy szybkość transmisji radiowej lub wpisać ją samodzielnie w okienku „Datarate:". I na początek to wszystko. Uwzględniając podane przez nas parametry program samodzielnie wyliczy wartości, które trzeba wpisać do wszystkich rejestrów. Teraz należy tylko te wyliczone wartości wydobyć z programu. W tym celu z menu „File" wybieramy opcję „Export CC1100 code", co powinno spowodować wyświetlenie kolejnej planszy, na której początkowo jest niewiele informacji. Jeżeli jednak dwukrotnie klikniemy na napis „*Adress definitions", po prawej stronie powinien pojawić się plik tekstowy z opisem najważniejszych opcji jakie zostały ustawione i przypisaniami nazw mnemonicznych wszystkich rejestrów. Po naciśnięciu „RF settings struct typedef" pojawi się plik tekstowy z definicją struktury zawierającą wszystkie rejestry, które trzeba zaprogramować, zaś po wybraniu „RF settings" zostanie wyświetlony plik z wartościami rejestrów, które należy wpisać do struktury. Każdy z plików można zapisać na dysku jako plik tekstowy po naciśnięciu przycisku „Write to file", podaniu ścieżki dostępu i nazwy pod jaką ma być zapisany.

Inicjalizacja rejestrów

Wygenerowane przez program „SmartRF Studio" pliki inicjalizacji rejestrów w różny sposób można użyć w oprogramowaniu swojego mikrokontrolera. Po małej adaptacji można utworzyć tablicę wartości, które mają być wpisane do kolejnych rejestrów przy pomocy procedury *SpiWriteReg* i będzie to najłatwiejsze rozwiązanie jeśli np. korzystamy z interpretera BASIC–a dla AVR. Jeśli jednak używamy kompilatora języka C, najprościej będzie posłużyć się strukturą, tym bardziej, że wygenerowane pliki wprost nadają się dla takiego rozwiązania. Sposób postępowania może wtedy wyglądać następująco:

1. Tworzymy plik nagłówkowy o nazwie np. *CC1100_ini.h*, któ-

List. 9. Wywołanie procedury inicjującej *RfWriteRfSettings* z głównej pętli

```
#include  <CC1100_ini.h>

void main(void)
{
 RfWriteRfSettings(&rfSettings);
 //po zakończeniu inicjacji układ powinien był przełączony w tryb odbioru
 SpiWriteCommand(CCxxx0_SIDLE);
 SpiWriteCommand(CCxxx0_SRX);

 //dalsze linie programu
}
```

Tab. 1. Zależność pomiędzy wartościami wpisanymi do rejestru a mocą wyjściową

Moc wyjściowa [dBm]	433 MHz		868 MHz	
	Wartość rejestru	Prąd typ. [mA]	Wartość rejestru	Prąd typ. [mA]
–30	0x04	11,5	0x03	11,9
–20	0x17	12,0	0x0D	12,4
–15	0x1C	12,7	0x1C	13,0
–10	0x26	14,0	0x34	14,5
–5	0x57	13,7	0x57	14,1
0	0x60	15,5	0x8E	16,9
5	0x85	19,0	0x85	20,0
7	0xC8	24,2	0xCC	25,8
10	0xC0	28,9	0xC3	30,7

List. 10. inicjalizacja rejestru PATABLE

```
#define POUT_5dBm  0x85 //przykładowa deklaracja etykiety wartości dla mocy wyjściowej 5dBm
unsigned char moc_wyjsciowa;

  moc_wyjsciowa = POUT_5dBm;
 SpiWriteBurstReg(CCxxx0_PATABLE, &moc_wyjsciowa, 1);//zapis do rejestru PATABLE w trybie "burst"
```

ry potem dyrektywą #include! włączymy do głównego programu.

2. Do pliku wklejamy deklaracje nazw wszystkich rejestrów wygenerowane przy pomocy opcji „*Adress definitions" programu „SmartRF Studio".
3. W pliku nagłówkowym dodajemy deklarację struktury RF_SETTINGS wygenerowaną przy pomocy opcji „RF settings struct typedef". Zależnie od wymagań stosowanego kompilatora być może trzeba będzie ręcznie zmienić formę deklaracji, np. deklarację typu zmiennych z BYTE na unsigned char itp. Notabene można zmusić „SmartRF Studio" by wygenerował plik deklaracji dokładnie w takiej formie, w jakiej sobie tego życzymy. Należy jednak poeksperymentować z ustawieniami na planszy „Code Export".
4. Na koniec w pliku nagłówkowym dopisujemy plik tekstowy wygenerowany opcją „RF settings", który jest inicjacją struktury wartościami.
5. W głównym programie tworzymy procedurę, np. o nazwie *RfWriteRfSettings*, która w pętli lub kolejnymi rozkazami pobierze wartości ze struktury RF_SETTINGS i zapisze je do właściwych rejestrów.

Przykładowy kod realizujący wymienione powyżej funkcje jest przedstawiony na **list. 7** i **8**.

Wywołanie procedury inicjującej *RfWriteRfSettings* z głównej pętli programu będzie wyglądało tak, jak na **list. 9**.

W ostatniej linii kodu z list. 9 procedura inicjalizacji układu CC1100 jest niemal ukończona. Niemal, bo pozostało jeszcze jednorazowe ustawienie poziomu mocy wyjściowej, która będzie dostarczana do anteny w trakcie nadawania. Chodzi w tym przypadku o wpisanie 1–bajtowej wartości do rejestru PATABLE. **Tab. 1** skopiowana z dokumentacji technicznej CC1100 pokazuje zależność pomiędzy wartościami wpisanymi do rejestru a mocą wyjściową:

W najprostszym przypadku inicjalizacja rejestru PATABLE może wyglądać jak na **list. 10.**

Teraz wszystkie niezbędne rejestry są już zainicjowane i możemy przystąpić do omawiania procedur transmisji i odbioru.

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