

Pierwsze kroki z modulem CC1100EM

Najważniejsze informacje o możliwościach i sposobach sterowania modulem radiowego transceiwera



1. Zastosowania
2. Budowa
 - 2.1. Układ elektryczny
 - 2.2. Najważniejsze parametry
 - 2.3. Wymiary mechaniczne
 - 2.4. Rozkład wyprowadzeń
 - 2.5. Oznaczenia
3. Współpraca modułu z innymi układami
 - 3.1. Połączenie z układami zasilanymi tym samym napięciem zasilania
 - 3.2. Połączenie z układami zasilanymi napięciem o wyższym poziomie
4. Magistrala SPI
 - 4.1. Wyprowadzenia magistrali i format transmisji
5. Rejestry układu CC1100
 - 5.1. Tryby dostępu do rejestrów
6. Przykłady kodów procedur sterujących modulem CC1100EM
 - 6.1. Procedury programowej transmisji magistralą SPI
 - a) SpiReadReg
 - b) SpiWriteReg
 - c) SpiReadBurstReg
 - d) SpiWriteBurstReg
 - e) SpiWriteCommand
 - 6.2. Inicjacja i ustawianie parametrów pracy CC1100
 - a) Zerowanie CC1100
 - b) Inicjacja rejestrów
 - 6.3. Obsługa transmisji i odbioru torem radiowym
 - a) Procedura transmisji
 - b) Procedura odbioru
7. Wykorzystanie programu SmartRF Studio do automatycznej generacji ustawień rejestrów CC1100
 - 7.1. Wybór układu radiowego
 - 7.2. Wybór parametrów toru radiowego
 - 7.3. Generacja plików ustawień rejestrów CC1100

Użyte skróty:

ISM -Industrial Scientific and Medical (układ do zastosowań przemysłowych, naukowych i medycznych)

SRD -Short Range Device (układ do łączności w bliskim zasięgu)

Krajowa Tablica Przeznaczeń Częstotliwości -akt prawny przydzielający do wykorzystania poszczególne częstotliwości radiowe

BER -bit error rate (stopa błędów transmisji, procent błędnie przesłanych bitów w transmisji).

dBm -jednostka mocy odniesiona do 1mW. I tak 10dBm =10mW a -10dBm =0,1mW.

1. Zastosowania

Moduł CC1100EM należy do grupy urządzeń SRD i służy do dwustronnej transmisji danych cyfrowych torem radiowym. Jest to urządzenie typu ISM czyli przeznaczone do zastosowań medycznych, przemysłowych i w urządzeniach powszechnego użytku. Może zostać użyty do budowy układów zdalnego włączania i wyłączania, przesyłania danych z czujników pomiarowych, dla układów ochrony i zabezpieczenia do sterowania robotami, modelami, zabawkami itp.

2. Budowa

2.1. Układ elektryczny

Moduł CC1100EM jest zbudowany w oparciu o układ CC1100 firmy Texas Instruments (która wchłonęła pierwotnego producenta czyli firmę CHIPCON). Zależnie od pasma częstotliwości wartości niektórych elementów modułu lub schemat elektryczny układów wejściowych mogą być różne w różnych typach modułów.

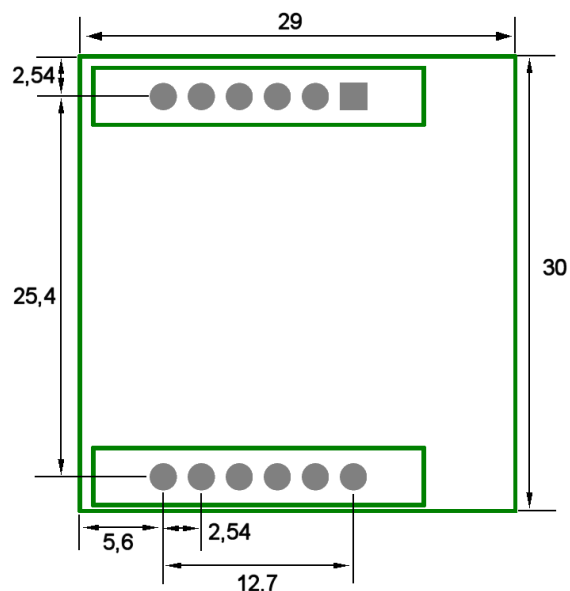
2.2. Najważniejsze parametry

- zależnie od opcji wykonania praca w pasmach od 300MHz do 928MHz. Częstotliwość pracy ustawiana jest programowo. Możliwości techniczne pozwalają pracować układowi na pasmach częstotliwości używanych zarówno w Europie jak i w USA i Japonii. W Polsce dostęp do określonych częstotliwości reguluje Krajowa Tablica Przeznaczeń Częstotliwości.
- szybkość transmisji danych ustawiana programowo w zakresie od 1,2 kilobita do 500 kilobitów.
- duża czułość odbiornika, nawet -110dBm przy szybkości transmisji 1,2 kilobita dla BER nie przekraczającej 1%.
- moc wyjściowa nadajnika do 10dBm ustawiana programowo w szerokim zakresie.
- dwa oddzielne 64 bajtowe bufory nadawczo-odbiorcze. Dzięki znacznemu zautomatyzowaniu procesu transmisji i odbioru danych moduł może współpracować nawet z bardzo wolnymi mikrokontrolerami lub w systemach gdzie mikrokontroler jest silnie obciążony innymi zadaniami.
- wewnętrzny układ automatycznej korekcji błędów pozwalający eliminować lub częściowo naprawiać błędy transmisji.
- wybierany programowo format transmisji radiowej: FSK, OOK, ASK.
- wybierany programowo numer kanału radiowego z możliwością ustawienia do 255 kanałów.
- możliwość wprowadzenia układu w tryb czuwania gdy średni pobierany prąd jest zredukowany do wartości mniejszej od 1μA.
- magistrala SPI używana zarówno do programowania aktualnych parametrów pracy układu jak i do transmisji i odbioru danych.
- zasilanie pojedynczym napięciem stałym w zakresie od 1,8V do 3,3V.

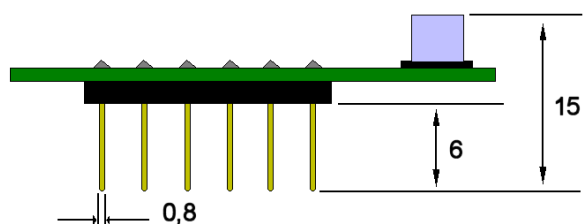
2.3. Wymiary mechaniczne

Moduł CC1100EM zbudowany jest w formie niewielkiej dwustronnej płytki drukowanej z dwoma rzędami styków do połączenia z zewnętrznymi współpracującymi układami. Elementy zamontowane są na górnej stronie płytki. Na rysunkach 1 i 2 podane są wymiary modułu w milimetrach.

Rys. 1



Rys. 2



Istnieje możliwość zamówienia wykonania bez wlutowanych styków złączy.

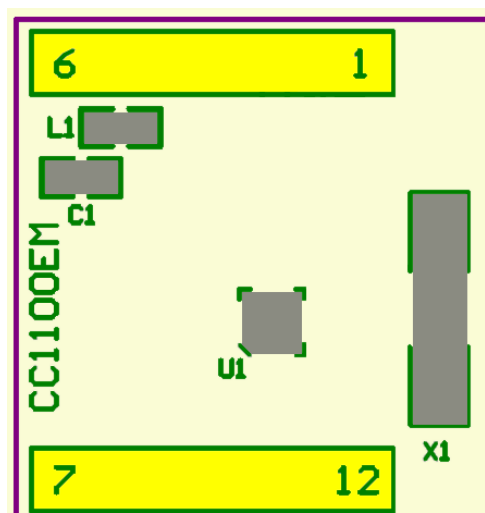
2.4. Rozkład wyprowadzeń

Na rysunku 3 pokazano rozłożenie niektórych elementów na płytce i numerację złączy. W Tabeli 1 opisano funkcje poszczególnych wyprowadzeń złączy.

Tabela 1

numer	funkcja
1	GND
2	antena
3	GND
4	VDD (1,8 3,3V)
5	n.c.
6	GND
7	CSn (interfejs SPI wejście chip select)
8	SI (interfejs SPI wejście serial input)
9	SCLK (interfejs SPI wejście clock input)
10	SO (interfejs SPI wyjście serial output)
11	GDO0 (wyjście wielofunkcyjne)
12	GDO2 (wyjście wielofunkcyjne)

Rys. 3



2.5. Oznaczenia

Na każdym module jest umieszczona jego nazwa, symbol producenta oraz zaznaczenie pasma częstotliwości w których pracuje.

3. Współpraca modułu z innymi układami

Moduł CC1100EM może współpracować z układami zewnętrznymi np. mikrokontrolerem zasilanym napięciem o takim samym lub wyższym poziomie.

3.1. Połączenie z układami zasilanymi tym samym napięciem zasilania

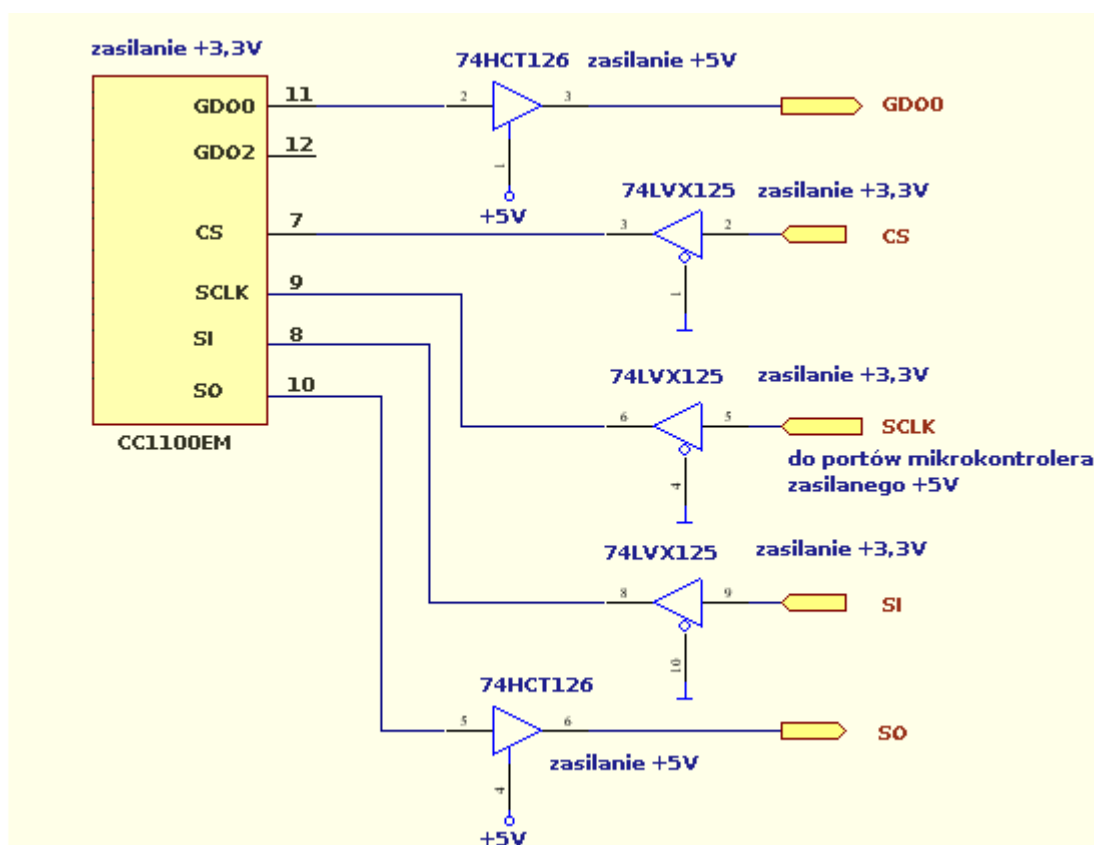
W przypadku współpracy z mikrokontrolerem zasilanym napięciem o takim samym poziomie jak poziom zasilania modułu wyprowadzenia obydwu układów mogą być połączone bezpośrednio:

- wyprowadzenia magistrali SPI modułu, które są wyjściami powinny być połączone z portami mikrokontrolera ustawionymi jako wejściowe
- wyprowadzenia magistrali SPI modułu, które są wejściami powinny być połączone z portami mikrokontrolera ustawionymi jako wyjściowe
- wyjścia uniwersalne GDO0 i GDO2 modułu powinny być połączone z portami mikrokontrolera ustawionymi jako wejściowe

3.2. Połączenie z układami zasilanymi napięciem o wyższym poziomie

Moduł CC1100EM może współpracować z mikrokontrolerem zasilanym napięciem o wyższym poziomie np. +5V. W takim przypadku pomiędzy wyprowadzeniami modułu a portami mikrokontrolera należy zastosować układy dopasowania poziomów. Przykład układu dopasowania poziomu z wykorzystaniem bramek HCT126 i LVX125 (lub podobnych) pokazany jest na rysunku 4.

Rys. 4



4. Magistrala SPI

Do komunikacji pomiędzy modułem CC1100EM a układami zewnętrznymi np. mikrokontrolerem sterującym wykorzystywana jest 4-przewodowa magistrala SPI. Magistrala służy zarówno do transmisji rozkazów sterujących pracą modułu jak i do przesyłania danych: transmitowanych i odbieranych torem radiowym.

4.1. Wyprowadzenia magistrali i format transmisji

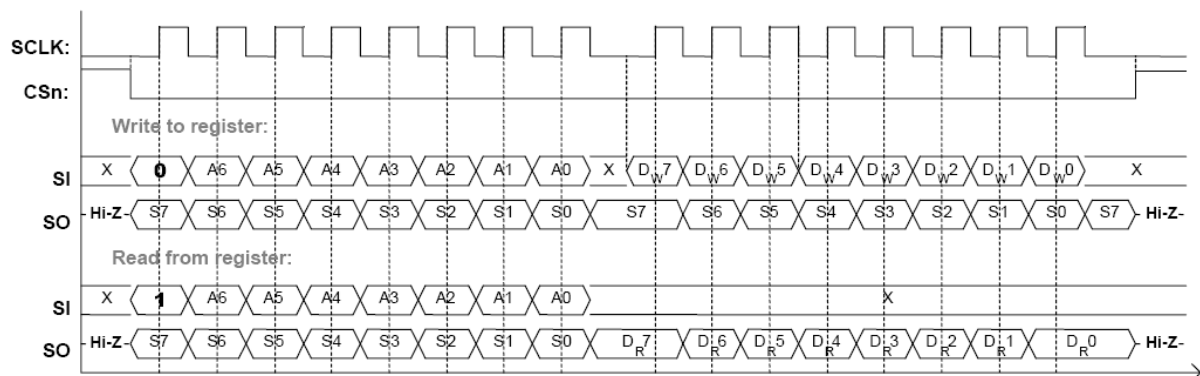
W skład magistrali wchodzi 4 jednokierunkowe linie których funkcje zestawiono w Tabeli 2.

Tabela 2

CS	wejście	podanie na wejście stanu niskiego uaktywnia magistralę SPI
SCLK	wejście	wejście zegara taktującego przesyłanie poszczególnych bitów na liniach danych
SI	wejście	wejście danych wysyłanych do modułu CC1100EM
SO	wyjście	wyjście danych odczytywanych z modułu CC1100EM

Magistrala SPI jest magistralą szeregową. Oznacza to, że dane do zapisu i odczytu przesyłane są kolejno bit po bicie. Zapis i odczyt poszczególnych bitów synchronizowany jest narastającym zboczem impulsów zegarowych. Rysunek 5 pokazuje przebiegi impulsów na poszczególnych liniach w czasie zapisu i odczytu pojedynczego bajtu danych.

Rys. 5



Możliwość wymiany danych z modułem CC1100EM poprzez magistralę SPI następuje z chwilą ustawienia linii CSn w stanie niskim.

W trakcie zapisu każde zbocze narastające zegara SCLK powoduje zapis kolejnego bitu podanego na wejście SI modułu. W trakcie odczytu każde zbocze narastające zegara SCLK powoduje odczyt kolejnego bitu z wyjścia SO modułu CC1100EM.

W trakcie zapisu jako pierwszy powinien być wysłany bit o wartości "0" a po nim 7 bitów adresu rejestru (w kolejności od najstarszego bitu A6 do najmłodszego A0). Następnie wysyłanych jest 8 bitów bajtu danych także z najstarszym bitem D7 jako pierwszym.

Podczas odczytu zawartości rejestru jako pierwszy powinien być wysłany bit o wartości "1" a po nim 7 bitów adresu rejestru (w kolejności od najstarszego bitu A6 do najmłodszego A0). Następnie z wyjścia SO można odczytać 8 bitów bajtu danych z bitem D7 odczytywanym jako pierwszy. Po zakończeniu transmisji należy zamknąć magistralę SPI podając stan wysoki na wyprowadzenie Csn.

5. Rejestry układu CC1100

Układ scalony CC1100 zamontowany na płycie modułu jest programowanym nadajnikiem - odbiornikiem radiowym. W każdej chwili można zmienić tryb jego pracy wpisując nowe wartości do odpowiednich wewnętrznych rejestrów CC1100. Także transmisja i odbiór danych torem radiowym traktowane są jako zapis i odczyt z rejestrów danych.

Dokładny opis rejestrów układu CC1100 dostępny jest w dokumentacji technicznej. Dokument można pobrać z sekcji Pliki strony firmy ARIES www.ars.info.pl lub ze strony producenta układu firmy Texas instruments www.ti.com.

5.1. Tryby dostępu do rejestrów

Większość rejestrów układu CC1100 można zarówno odczytywać jak i zapisywać. Sekwencje sygnałów podczas zapisu i odczytu do rejestru pokazuje rysunek 5.

Stan najstarszego 7 bitu adresu który jest wysyłany jako pierwszy decyduje o tym czy transmisja jest zapisem czy odczytem z rejestru. Podczas odczytu bit 7 jest ustawiony a skasowany podczas zapisu.

Adres rejestru do którego chcemy mieć dostęp ustawiany jest bitami A5-A0. Przyporządkowanie adresów rejestrów wraz z opisem ich wpływu na sposób działania układu CC1100 można znaleźć w tabeli 37 "SPI Address Space" w danych technicznych CC1100.

Zapis i odczyt rejestrów można zrealizować dwoma sposobami.

a/ Pierwszy tryb nazwany w dokumentacji "single byte" polega na podaniu adresu rejestru do którego chcemy mieć dostęp a następnie na zapisie lub odczycie zawartości rejestru.

b/ Drugi tryb zapisu/odczytu danych jest trybem grupowym i nosi nazwę "burst".

Polega na hurtowym zapisie lub odczycie rejestrów. Po podaniu adresu pierwszego rejestru z grupy następuje cykliczny zapis lub odczyt kolejnych rejestrów bez konieczności podawania ich adresów.

O tym czy zapis/odczyt jest typu "single byte" czy "burst" decyduje 6 bit adresu. Jeżeli bit jest wyzerowany mamy do czynienia z dostępem typu "single byte". Jeżeli bit A6 jest ustawiony chodzi o dostęp grupowy czyli "burst".

W zależności od typu rejestru możliwy jest dostęp w obydwu trybach albo wybór trybu oznacza dostęp do różnych rejestrów mających ten sam adres.

Rejestry o adresach od 00h do 2Eh (litera "h" oznacza zapis adresu w kodzie haxadecymalnym) można zapisywać i odczytywać w obydwu trybach.

Dla rejestrów o adresach od 30h do 3Dh wybór trybu oznacza dostęp do rejestrów statusu lub komend. Różnica pomiędzy obydwoma typami rejestrów polega na tym, że rejestry statusu można jedynie odczytywać.

Rejestry komend mają wpływ na zasadnicze funkcje transeiwera np. przełączają go w tryb odbioru lub nadawania. Do rejestrów komend w trybie zapisu nie ma potrzeby przesyłania żadnych danych, wystarczy podanie ich adresu by uaktywnić funkcję za którą odpowiadają.

6. Przykłady kodów procedur sterujących modulem CC1100EM

W tej sekcji zamieszczone zostały przykłady procedur sterujących pracą modułu CC1100EM. Procedury umożliwiają inicjalizację modułu, ustawianie jego parametrów a także wysłanie i odbiór danych torem radiowym. Wszystkie przykłady kodu napisane zostały w języku C dla mikrokontrolera z rodziny AVR. Wybrany typ mikrokontrolera narzucił konwencję składni jeśli chodzi o fragmenty związane z bezpośrednim dostępem do portów. Po niewielkich zmianach kod można łatwo adaptować dla innych typów mikrokontrolerów.

W przykładach pomijane są np. deklarację nazw dla poszczególnych portów. Zawsze będą one takie same:

CS_RF_PORT	wyjscie CS do CC1100
SI_RF_PORT	wyjscie SI do CC1100
SCLK_RF_PORT	wyjscie SCLK do CC1100
SO_RF_PIN	wejście SO z CC1100
GDO_RF_PIN	wejście GDO z CC1100

Pojawiające się w niektórych listingach rozkazy assemblerowe #asm("nop") można pominąć lub zastąpić krótkimi pauzami o czasie trwania np. 1µs.

6.1.Procedury programowej transmisji magistralą SPI

W przypadku gdy nie ma możliwości wykorzystania sprzętowych mechanizmów transmisji magistralą SPI można wykorzystać pokazane na poniższych listingach programowe procedury transmisji.

Procedury umożliwiają zapis i odczyt z rejestrów w 5 trybach:

SpiReadReg	-odczyt z rejestru w trybie "single byte"
SpiWriteReg	-zapis do rejestru w trybie "single byte"
SpiReadBurstReg	-odczyt rejestrów w trybie "burst"
SpiWriteBurstReg	-zapis rejestrów w trybie "burst"
SpiWriteCommand	-przesłanie do rejestru komendy

a) SpiReadReg

Pokazana na Listingu 1 procedura służy do odczytu z pojedynczego rejestru. Procedura wywoływana jest z parametrem "adres" w którym przekazywany jest numer odczytywanego rejestru w trybie "single byte", po zakończeniu procedura zwraca wartość odczytaną z rejestru. W bajcie adresu bit 7 jest ustawiany a bit 6 zerowany.

Listing 1

```
//odczyt zawartości pojedynczego rejestru
//-----
unsigned char SpiReadReg(unsigned char adres)
//we: adres -adres rejestru
//wy: bajt odczytany z rejestru
{
    unsigned char x, maska, wartosc;

    SCLK_RF_PORT =0;
```

```

CS_RF_PORT =0;
while (SO_RF_PIN !=0); //oczekiwanie na stan niski na wyjściu SO
adres =adres & 0b00111111; //wyzerowany b.7 i b.6
adres =adres | 0b10000000; //b.7=1, b.6=0 -odczyt w trybie "single byte"
maska =0b10000000;
//wysłanie adresu rejestru
for (x=0; x<8; x++)
{
    if ((adres &maska) ==0) SI_RF_PORT =0;
    else SI_RF_PORT =1;
    #asm("nop");
    SCLK_RF_PORT =1;
    #asm("nop");
    SCLK_RF_PORT =0;
    maska =maska >>1;
}
wartosc =0;
maska =0b10000000;
//odczyt zawartości rejestru
for (x=0; x<8; x++)
{
    SCLK_RF_PORT =1;
    #asm("nop");
    if (SO_RF_PIN ==1) wartosc =wartosc |maska;
    #asm("nop");
    SCLK_RF_PORT =0;
    maska =maska >>1;
}
CS_RF_PORT =1;
return wartosc;
}

```

b) SpiWriteReg

Pokazana na Listingu 2 procedura służy do zapisu do pojedynczego rejestru. Procedura wywoływana jest z parametrami "adres" w którym przekazywany jest numer zapisywanego rejestru w trybie "single byte" i wartość będącym bajtem danej do zapisu w rejestrze. W bajcie adresu bit 7 i bit 6 są zerowane.

Listing 2

```

//zapis wartości do pojedynczego rejestru
//-----
void SpiWriteReg(unsigned char adres, unsigned char wartosc)
//we:   adres -adres rejestru, wartosc -wpisywana do rejestru wartość
{
    unsigned char x, maska;

    SCLK_RF_PORT =0;
    CS_RF_PORT =0;
    while (SO_RF_PIN !=0); //oczekiwanie na stan niski na wyjściu SO
    adres =adres & 0b00111111; //wyzerowany b.7 i b.6
    adres =adres | 0b00000000; //b.7 i b.6 =0 zapis do rejestru w trybie "single byte"
    maska =0b10000000;
    //wysłanie adresu rejestru
    for (x=0; x<8; x++)
    {
        if ((adres &maska) ==0) SI_RF_PORT =0;
        else SI_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =0;
        maska =maska >>1;
    }
    //wysłanie kolejnych bitów bajtu zapisywanego do rejestru
    maska =0b10000000;
    for (x=0; x<8; x++)
    {
        if ((wartosc &maska) ==0) SI_RF_PORT =0;
        else SI_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =0;
        maska =maska >>1;
    }
    CS_RF_PORT =1;
}

```

```
}
```

c) SpiReadBurstReg

Pokazana na Listingu 3 procedura służy do odczytu z grupy rejestrów. Procedura wywoływana jest z parametrem "adres" w którym przekazywany jest numer pierwszego z odczytywanych rejestrów w trybie "burst", wskaźnikiem pbufor do buforu w którym zostaną zapisane bajty odczytane z rejestrów, parametrem ile w którym przekazywana jest ilość rejestrów do odczytu (łącznie z pierwszym). W bajcie adresu bity 7 i bit 6 są ustawiane.

Listing 3

```
//odczyt grupowy zawartości rejestrów
//-----
void SpiReadBurstReg(unsigned char adres, unsigned char *pbufor, unsigned char ile)
//we:   adres -adres pierwszego odczytywanego rejestru,
//      *pbufor -wskaźnik do buforu gdzie mają się znaleźć odczytane z rejestrów dane
//      ile -ilość rejestrów do odczytu
{
    unsigned char x, y, maska, wartosc;

    SCLK_RF_PORT =0;
    CS_RF_PORT =0;
    while (SO_RF_PIN !=0); //oczekiwanie na stan niski na wyjściu SO
    adres =adres & 0b00111111; //wyzerowany b.7 i b.6
    adres =adres | 0b11000000; //b.7 i b.6 =1 odczyt w trybie "burst"
    maska =0b10000000;
    //wysłanie adresu pierwszego rejestru
    for (x=0; x<8; x++)
    {
        if ((adres &maska) ==0) SI_RF_PORT =0;
        else SI_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =0;
        maska =maska >>1;
    }
    //odczyt kolejnych rejestrów
    for (y=0; y<ile; y++)
    {
        wartosc =0;
        maska =0b10000000;
        for (x=0; x<8; x++)
        {
            SCLK_RF_PORT =1;
            #asm("nop");
            if (SO_RF_PIN ==1) wartosc =wartosc |maska;
            #asm("nop");
            SCLK_RF_PORT =0;
            maska =maska >>1;
        }
        *pbufor =wartosc;
        pbufor++;
    }
    CS_RF_PORT =1;
}
```

d) SpiWriteBurstReg

Pokazana na Listingu 4 procedura służy do zapisu do grupy rejestrów. Procedura wywoływana jest z parametrem "adres" w którym przekazywany jest numer pierwszego z zapisywanych rejestrów w trybie "burst", wskaźnikiem pbufor do buforu w którym znajdują się bajty do zapisu do rejestrów, parametrem ile w którym przekazywana jest ilość rejestrów do zapisu (łącznie z pierwszym). W bajcie adresu bit 7 jest wyzerowany a bit 6 jest ustawiony.

Listing 4

```
//grupowy zapis do kolejnych rejestrów
//-----
void SpiWriteBurstReg(unsigned char adres, unsigned char *pbufor, unsigned char ile)
//we:   adres -adres pierwszego zapisywanego rejestru,
//      *pbufor -wskaźnik do buforu z wartościami które mają być wpisywane do rejestrów
//      ile -ilość rejestrów do zapisu
{
```

```

unsigned char x, y, maska, wartosc;

SCLK_RF_PORT =0;
CS_RF_PORT =0;
while (SO_RF_PIN !=0); //oczekiwanie na stan niski na wyjściu SO
adres =adres & 0b00111111; //wyzerowany b.7 i b.6
adres =adres | 0b01000000; //b.7=0, b.6=1 zapis w trybie "burst"
maska =0b10000000;
//wysłanie adresu pierwszego rejestru
for (x=0; x<8; x++)
{
    if ((adres &maska) ==0) SI_RF_PORT =0;
    else SI_RF_PORT =1;
    #asm("nop");
    SCLK_RF_PORT =1;
    #asm("nop");
    SCLK_RF_PORT =0;
    maska =maska >>1;
}
//zapis do kolejnych rejestrów
for (y=0; y<ile; y++)
{
    maska =0b10000000;
    wartosc =*pbufor;
    for (x=0; x<8; x++)
    {
        if ((wartosc &maska) ==0) SI_RF_PORT =0;
        else SI_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =1;
        #asm("nop");
        SCLK_RF_PORT =0;
        maska =maska >>1;
    }
    pbufor++;
}
CS_RF_PORT =1;
}

```

e) SpiWriteCommand

Pokazana na Listingu 5 procedura służy do wysłania pojedynczej komendy. Procedura wywoływana jest z parametrami "adres" w którym przekazywany jest numer rejestru komendy. W bajcie adresu bit 7 i bit 6 są zerowane.

Listing 5

```

//wysłanie pojedynczego rozkazu
//-----
void SpiWriteCommand(unsigned char adres)
{
//we: adres rejestru rozkazów
unsigned char x, maska;

SCLK_RF_PORT =0;
CS_RF_PORT =0;
while (SO_RF_PIN !=0); //oczekiwanie na stan niski na wyjściu SO
adres =adres & 0b00111111; //wyzerowany b.7 i b.6
adres =adres | 0b00000000; //b.7 i b.6 =0 zapis pojedynczego rozkazu
maska =0b10000000;
//wysłanie adresu rejestru
for (x=0; x<8; x++)
{
    if ((adres &maska) ==0) SI_RF_PORT =0;
    else SI_RF_PORT =1;
    #asm("nop");
    SCLK_RF_PORT =1;
    #asm("nop");
    SCLK_RF_PORT =0;
    maska =maska >>1;
}
CS_RF_PORT =1;
}

```

6.2. Inicjacja i ustawianie parametrów pracy CC1100

Po włączeniu zasilania zamontowany na płycie modułu układ CC1100 każdorazowo powinien zostać programowo

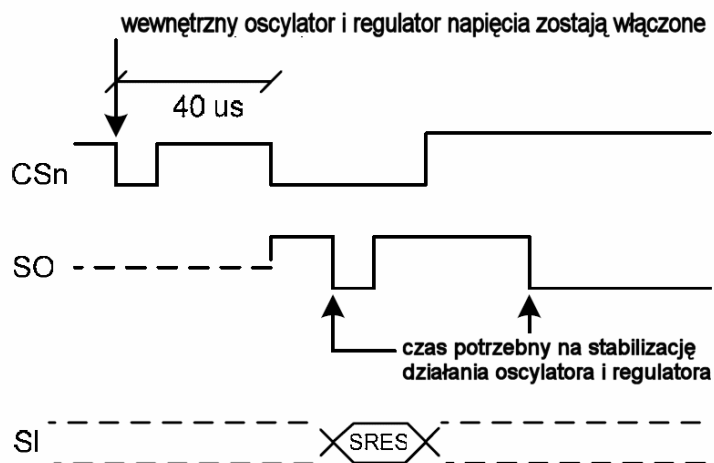
wyzerowany a do jego rejestrów powinny zostać wpisane odpowiednie wartości. Dane do zapisu do rejestrów najłatwiej wygenerować automatycznie korzystając z programu *SmartRF Studio*. Sposób korzystania z programu zostanie opisany w punkcie 7.

Do wyzerowania układu CC1100 i zapisu do jego rejestrów można wykorzystać procedury magistrali SPI opisane w punkcie 6.1. Procedury programowej transmisji magistralą SPI.

a) Zerowanie CC1100

Układ CC1100 posiada co prawda wewnętrzne mechanizmy zerujące ale jeśli czas narastania napięcia zasilającego jest zbyt powolny i nie ma pewności czy wewnętrzny oscylator osiągnął właściwą stabilność należy przeprowadzić zalecaną procedurę. Składa się na nią wygenerowanie na linii CSn odpowiednich impulsów oraz obserwacja linii SO i wysłanie do układu komendy zerującej SRES. Całą sekwencję procedury zerującej pokazuje rysunek 6:

Rys. 6



Procedura zerowania może wyglądać podobnie jak na poniższym listingu

Listing 6

```
//sekwencja zerowania transceiwera po włączeniu zasilania
//-----
void Rf_Power_Up_Reset(void)
{
#define CCxxx0_SRES    0x30    //kod komendy SRES
unsigned char maska, x;

    SCLK_RF_PORT =1;
    SI_RF_PORT =0;
    CS_RF_PORT =0;
    #asm("nop");
    #asm("nop");
    CS_RF_PORT =1;
    SCLK_RF_PORT =0;
    delay_us(40); //pauza 40ms
    CS_RF_PORT =0;
    while (SO_RF_PIN !=0); //oczekiwanie na stan niski linii SO

    //wysłanie komendy CCxxx0_SRES
    SpiWriteCommand(CCxxx0_SRES);

    while (SO_RF_PIN ==0); //oczekiwanie na stan wysoki linii SO
    while (SO_RF_PIN !=0); //oczekiwanie na ponowny stan niski linii SO
    CS_RF_PORT =1;
}
}
```

Linia "delay_us(40);" oznacza, że następuje odwołanie do podprogramu pauzy o długości 40µs. Taką pauzę może też generować pętla typu for(;;) o ilości powtórzeń zależnej od szybkości działania mikrokontrolera.

b) Inicjacja rejestrów

Wygenerowane przez program "SmartRF Studio" pliki inicjacji rejestrów można użyć w oprogramowaniu mikrokontrolera sterującego pracą modułu CC1100EM i inicjującego jego rejestry. 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 dla AVR. Jeśli jednak używamy kompilatora języka C najprościej będzie posłużyć się strukturą tym bardziej, że wygenerowane pliki wprost się nadają dla takiego rozwiązania. Sposób postępowania może wtedy wyglądać następująco:

1. Tworzymy plik nagłówkowy o nazwie np. "CC1100_ini.h" który 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.
4. 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 może wyglądać następująco:

Plik nagłówkowy.

Listing 7

```
"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 rejestrów
//
//
//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.
};
```

Procedura "RfWriteRfSettings" umieszczona w programie głównym, która inicjuje wewnętrzne rejestry układu CC1100.

Listing 8

```
//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_IOCFCG2, pRfSettings->IOCFCG2);
    SpiWriteReg(CCxxx0_IOCFCG0, pRfSettings->IOCFCG0);
    SpiWriteReg(CCxxx0_PKTCTRL1, pRfSettings->PKTCTRL1);
    SpiWriteReg(CCxxx0_PKTCTRL0, pRfSettings->PKTCTRL0);
    SpiWriteReg(CCxxx0_ADDR, pRfSettings->ADDR);
    SpiWriteReg(CCxxx0_PKTLEN, pRfSettings->PKTLEN);
}

```

Wywołanie procedury inicjującej "RfWriteRfSettings" z głównej pętli programu. Na Listingu 9 pokazana została główna pętla programu, wywołanie procedury inicjującej i wysłanie do CC1100 komendy przełączającej układ w tryb odbioru danych radiowych.

Listing 9

```

#include <CC1100_ini.h>

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

    //dalsze linie programu
}

```

Po procedurze inicjacji RfWriteRfSettings można jeszcze jednorazowo ustawić poziom mocy wyjściowej która będzie dostarczana do anteny w trakcie nadawania. Polega to na wpisaniu 1 bajtowej wartości do rejestru PATABLE. W tabeli 3 pokazano zależność pomiędzy wartościami wpisanymi do rejestru a mocą wyjściową:

Tabela 3

	433MHz		868MHz	
Moc wyjściowa [dBm]	Wartość wpisana do rejestru PATABLE	Pobór prądu typ. [mA]	Wartość wpisana do rejestru PATABLE	Pobór prądu 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

W najprostszym przypadku inicjacja rejestru PATABLE może wyglądać następująco:

Listing 10

```
#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"
```

W tym momencie wszystkie niezbędne rejestry są już zainicjowane i moduł jest gotowy do transmisji i odbioru drogą radiową.

6.3. Obsługa transmisji i odbioru torem radiowym

Transmisja i odbiór danych torem radiowym są bardzo ułatwione dzięki specjalnym mechanizmom układu CC1100. Są to dwa wewnętrzne 64 bajtowe bufor nadawczo odbiorcze i wyjścia sygnalizacyjne GDO0 i GDO2. Najprostszym sposobem obsługi transmisji jest zaprogramowanie np. wyjścia GDO0 do sygnalizacji początku i zakończenia procesu nadawania lub odbioru kompletnego pakietu danych. Opcja ta stanie się aktywna po zapisie do rejestru IOCFG0 wartości 0x06. Można to zrobić w trakcie inicjacji rejestrów opisanej w poprzednich punktach. Od momentu inicjacji wyjście będzie sygnalizowało fakt transmisji i odebrania danych a dokładniej:

- w czasie transmisji wyjście GDO0 będzie miało poziom wysoki gdy rozpocznie się wysyłanie sygnału synchronizacji i z powrotem przyjmie poziom niski gdy wysłana zostanie 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.

a) Procedura transmisji

Cała procedura transmisji sprowadza się do kilku elementarnych kroków:

1. Układ CC1100 należy wprowadzić w tryb bezczynności komendą SIDLE.
 2. Do buforu 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/ ponieważ 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 inicjacji 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 wysłany i procedura transmisji może się zakończyć.
- Listing 11 pokazuje 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 dołączony jest zewnętrzny przycisk "SW_KL" i portu wyjściowego obsługujący diodę LED sygnalizującą świeceniem fakt transmisji "LED_TRANS_PORT". Po naciśnięciu przycisku zwierającego port do masy wywoływana jest procedura transmisji przykładowego napisu znajdującego się w tablicy wysylany_tekst_tab[[]].

Listing 11

```
#include <CC1100_ini.h>

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

```

void main(void)
{
unsigned char wyslany_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( wyslany_tekst_tab); //ustalania rozmiaru transmisji
            //przepisywanie wysłanego tekstu do tabeli pomocniczej
            for (x=0; x< ile_bajtow; x++)
            {
                tabela_pomocnicza[x+1] = wyslany_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 wysłanych 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;
}

```

b) 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 zasygnalizowane 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 buforu odbiorczego układu CC1100 do pamięci mikrokontrolera.

Listing 12 pokazuje obsługę odbioru transmisji przez procedurę Rf_Rec_Packet. Wprowadzona została obsługa dodatkowego portu wyjściowego LED_REC_PORT sterującego diodę LED sygnalizującą fakt odbioru transmisji.

Listing 12

```
#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;
        #define LQI      1

        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;
    }
}

```

7. Wykorzystanie programu SmartRF Studio do automatycznej generacji ustawień rejestrów CC1100

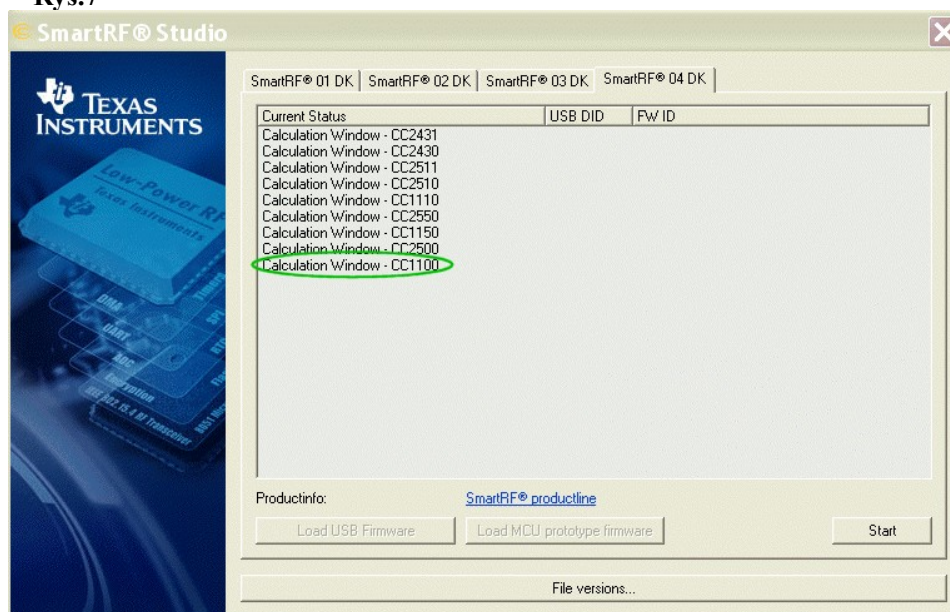
Program *SmartRF Studio* jest bezpłatnym oprogramowaniem narzędziowym rozpowszechnianym przez firmę CHOPCON wchodzącą obecnie w skład koncernu Texas Instruments. Program służy do tworzenia plików konfiguracyjnych rejestrów sterujących różnego typu układów radiowych produkowanych przez firmę. Wersję instalacyjną programu dla systemu Windows można pobrać ze strony firmy Texas Instruments <http://focus.ti.com/docs/toolsw/folders/print/smartrfm-studio.html> lub ze strony firmy ARIES <http://ars.info.pl/download/dema.html>.

7.1. Wybór układu radiowego

Po instalacji i uruchomieniu programu *SmartRF Studio*, korzystając z jego zakładek należy wybrać typ układu radiowego dla którego generowany będzie plik ustawień rejestrów.

O ile w parametrach technicznych modułu CC1100EM nie jest to podane inaczej należy wybrać opcję *CC1100* zaznaczoną na Rysunku 7 kolorem zielonym. Po zaznaczeniu kursorem opcji należy kliknąć przycisk *Start*.

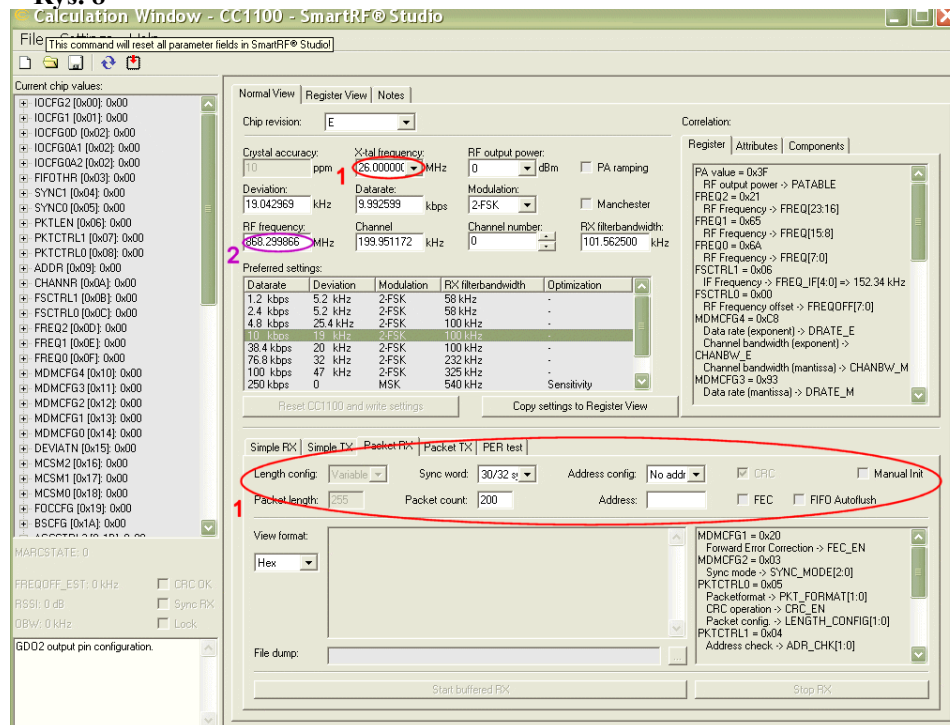
Rys. 7



7.2. Wybór parametrów toru radiowego

Zakładka *Calculation Window* służy do zmiany parametrów toru radiowego. Po otwarciu zakładki wszystkie parametry mają wartości domyślne.

Rys. 8



Użytkownik może zmieniać dowolne parametry z następującymi zastrzeżeniami dla modułu CC1100EM:

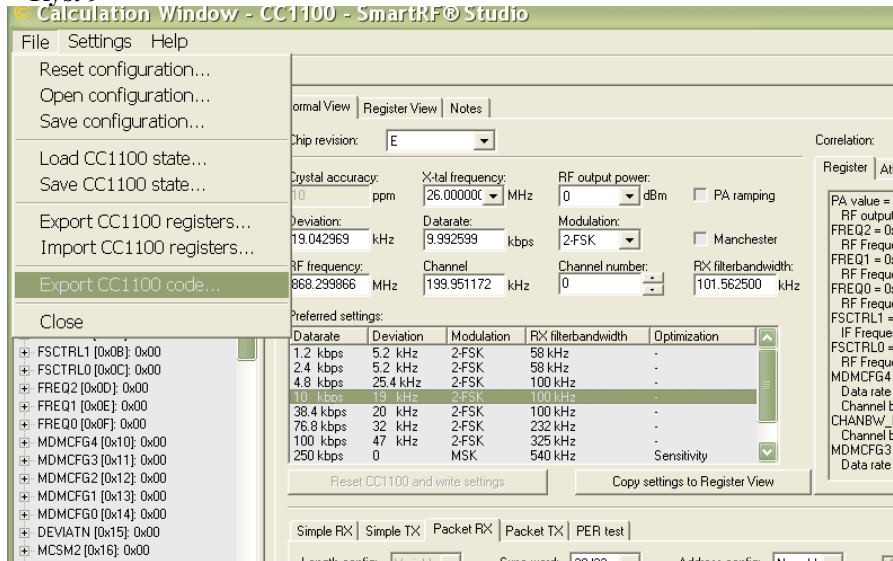
a/ Parametry zaznaczone na Rysunku 8 kolorem czerwonym i cyfrą 1 nie powinny być zmieniane. Dotyczy to zwłaszcza częstotliwości kwarcu (o ile nie zostało to inaczej zaznaczone w dokumentacji technicznej)

modułu). Także korzystając z procedur opisanych w punkcie 6.3 niniejszego tekstu nie należy zmieniać ustawień w drugim regionie zaznaczonym kolorem czerwonym i cyfrą 1.

b/ Zależnie od opcji wykonania zaznaczonej na płycie modułu należy wybrać odpowiednią częstotliwość pracy z pasma 433MHz lub 868MHz -region zaznaczony kolorem fioletowym i cyfrą 2.

Po wprowadzeniu wszystkich pożądanych ustawień należy wybrać z menu *Settings* opcję *Export CC1100 code*.

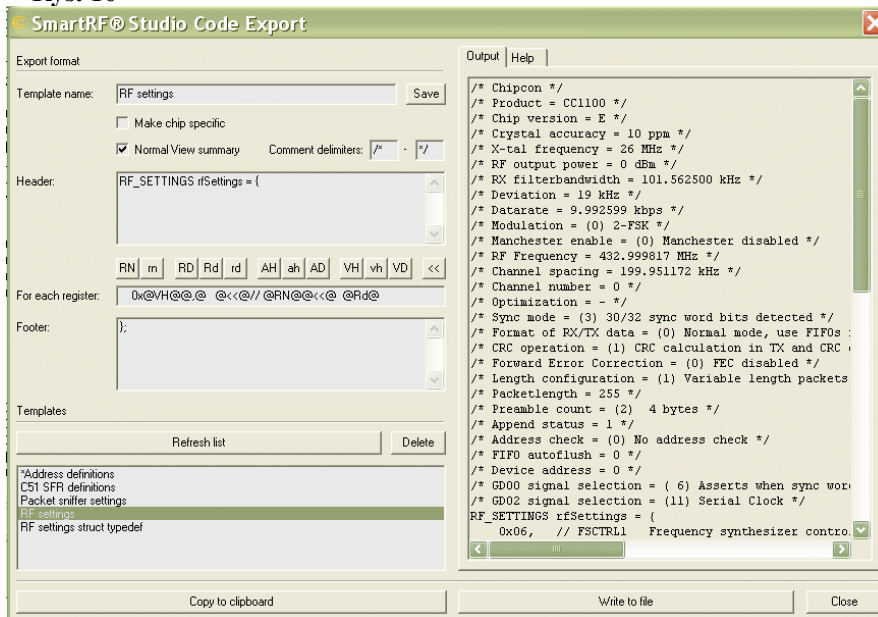
Rys. 9



7.3. Generacja plików ustawień rejestrów CC1100

Zakładka *Code Export* służy do generacji tekstowych plików ustawień rejestrów CC1100. Na Rysunku 10 pokazany został wygląd zakładki.

Rys. 10



W zależności od wybranych wcześniej ustawień zakładka pozwala automatycznie generować pliki wykorzystywane do inicjacji wewnętrznych rejestrów układu CC1100. W szczególności są to następujące pliki: *"*Adress definitions"*, *"RF settings struct typedef"*, *"RF settings"*. Opis użycia plików w programie użytkownika i przykładowe procedury inicjacji rejestrów zamieszczone są w punkcie 6.2. *Inicjacja i ustawianie parametrów pracy CC1100* podpunkt *b/ Inicjacja rejestrów*.

Po podwójnym kliknięciu nazwy pliku w oknie z prawej strony zakładki pojawi się jego zawartość. Klikając na klawisz *Copy to clipboard* lub *Write to file* zawartość okienka można zapisać do schowka lub do pliku tekstowego na dysku i potem wykorzystać w programie użytkownika.

opracowanie firma ARIES

www.ars.info.pl

biuro@ars.info.pl

1.2008

© [Wszelkie prawa zastrzeżone dla firmy ARIES](#)

Firma ARIES Ryszard Szymaniak www.ars.info.pl