Cyfrowe rejestratory dźwięku ISD151xx (4)

Układy z rodziny ISD151xx to zaawansowane, cyfrowe układy służące do rejestracji dźwięku. Mogą być elastycznie konfigurowane, a wbudowane kompresory/dekompresory pozwalają na zapisanie nawet 30 minut komunikatów słownych w najbardziej pojemnym układzie ISD15132. Producent nie zapomniał też o dźwięku lepszej jakości wyposażając układy w interfejs I²S i możliwość operowania na 16-bitowych nieskompresowanych danych audio.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 16732, pass: 630v2nfb
- poprzednie części artykułu

Narzędzia firmowe

Jak widać z dotychczasowego opisu, układ jest dość skomplikowany. Jego skonfigurowanie, a potem sterowanie jego działaniem można powierzyć sterownikowi mikroprocesorowemu, ale zaprojektowanie i oprogramowanie takiego sterownika będzie wymagało sporo pracy. Nie zawsze jest to możliwe, a często wręcz nieopłacalne. Zapewne z tych powodów producent przygotował firmowe narzędzia pozwalające na łatwe i szybkie sterowanie wszystkimi funkcjami układu, łącznie z pełną obsługą Voice Prompt i Voice Macro.

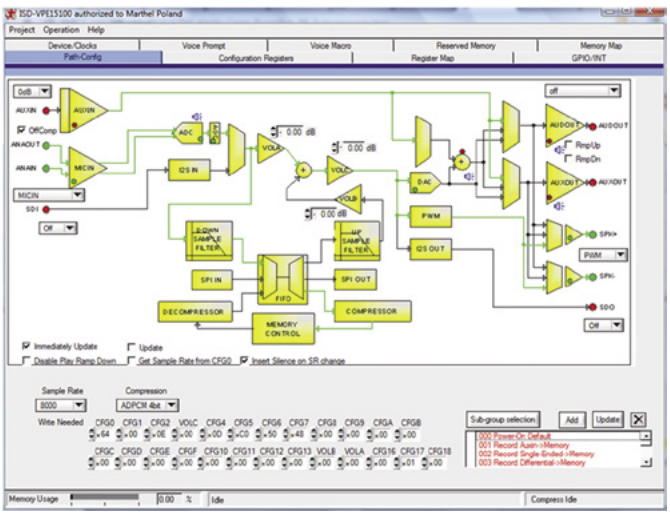
Głównym elementem przeznaczonym do sterowania układem ISD151xx jest program ISD-VPE15100. Jest to aplikacja typu IDE z możliwością grupowania efektów pracy w projekt zapisywany na dysku komputera. Praca z programem pozwala na utworzenie mapy pamięci, którą można będzie potem wgrać do układu, skonfigurowaniu ścieżki sygnałowej z ustawieniem poziomu tłumienia w cyfrowych regulatorach, zaprogramowanie układu AGC itp. Ponadto umożliwia wykonanie kompresji pliku wav do każdego obsługiwanego przez układ standardu i każdej długości słowa i zapisanie go jako Voice Prompt. Przetestowany projekt jest kompilowany do pliku binarnego, który potem można wgrać do pamięci układu. Aplikacja również to umożliwia, ale do tego celu niezbędny jest firmowy interfejs USB/SPI.

Po uruchomieniu programu można utworzyć nowy projekt, lub wczytać wcześniej utworzony. Interfejs użytkownika jest zbudowany z kilku ekranów wybieranych myszką przez zakładki umieszczone u góry interfejsu:
- Ekran konfiguracji ścieżki sygnałowej – Path Config.
- Ekran wyboru układu z rodziny ISD151xx, oraz konfiguracji zegara systemowego – Device/Clocks.
- Ekran zapisywania i odczytywania zwartości rejestrów konfiguracyjnych – Configuration Registers.
- Ekran Mapy wszystkich rejestrów – Register Map.
- Ekran Mapy pamięci flash – Memory Map.
- Ekran konfiguracji linii GPIO.
- Ekran zawartości komunikatów Voice Prompt.
- Ekran trworzenia makr Voice Makro.

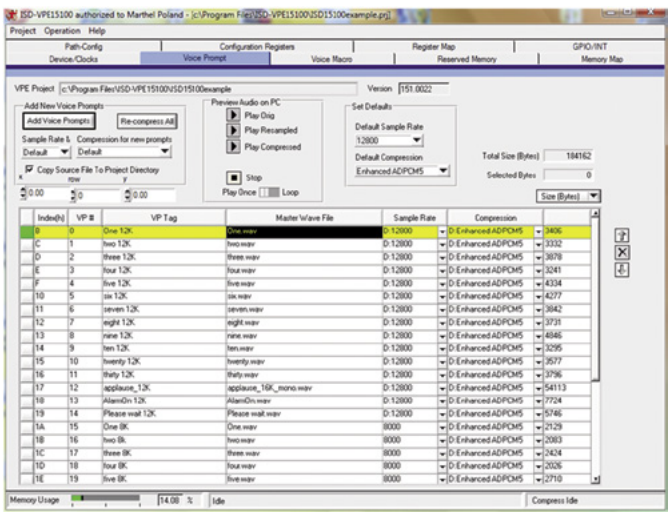
Bardzo przydatne są ekrany konfigurowania ścieżki sygnałowej, wyboru układu i konfigurowania zegara oraz tworzenia Voice macro i Voice Prompt.

W czasie poznawania budowy i działania układu warto najpierw zapoznać się z ekranem path config (**rysunek 19**). Można tu w bardzo prosty sposób konfigurować ścieżkę sygnału od źródła sygnału wejściowego (w naszym przypadku wejścia mikrofonowego) poprzez układy przetwornika ADC i kompresor AGC, do kompresora i zapisywania do pamięci Flash. Ścieżkę konfiguruje się przez klikanie myszką na ikony bloków funkcjonalnych. Dla konstruktorów, którzy chcieliby zaprojektować własny sterownik, dużą pomocą są wyświetlane na dole ekranu wartości rejestrów konfiguracji. Wystarczy przestudiować ich opis, aby docenić łatwość konfiguracji przy pomocy Config Path. Dodatkowo, ustawia się częstotliwość próbkowania przetwornika ADC i algorytm kompresji kompresora.

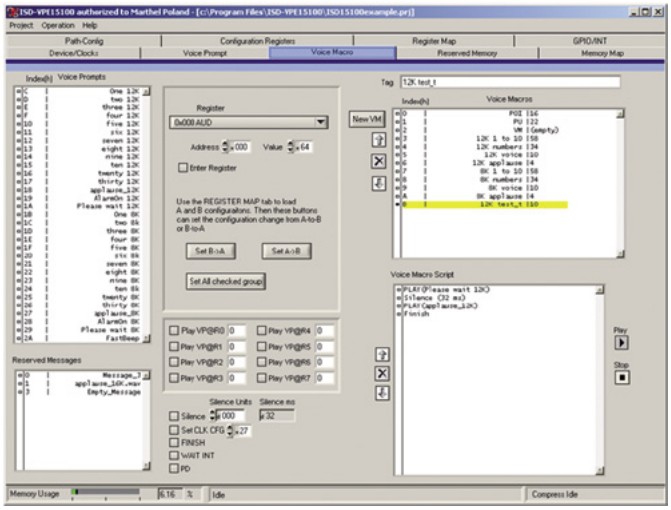
Ekran tworzenia Voice Prompt pokazano na **rysunku 20**. Nowe makra dodaje się po kliknięciu na belkę *Add Voice Prompt*. Aplikacja



Rysunek 19. Ekran Path-Config



Rysunek 20. Ekran Voice Propmpt



Rysunek 21. Ekran tworzenia Voice Macro

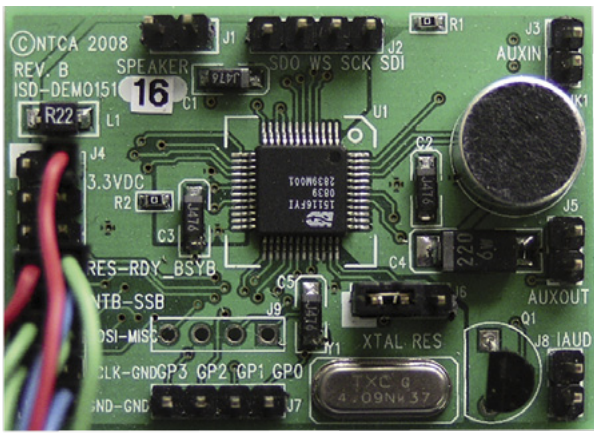
Rysunek 22. Interfejs ISD-ES-MINI-USB

wczytuje z dysku komputera wcześniej zapisany plik w formacie WAV z krótkimi komunikatami głosowymi i wykonuje kompresję za pomocą algorytmu wybranego w oknie *Default Compression* i konwersję częstotliwości próbkowania ustawioną w oknie *Default Sample Rate*. Każdy plik może mieć inny algorytm i różne częstotliwości próbkowania. Wczytane pliki można przesuwać na liście (zmieniać indeks). Belka na samym dole okna informuje o zajętość pamięci wybranego typu układu.

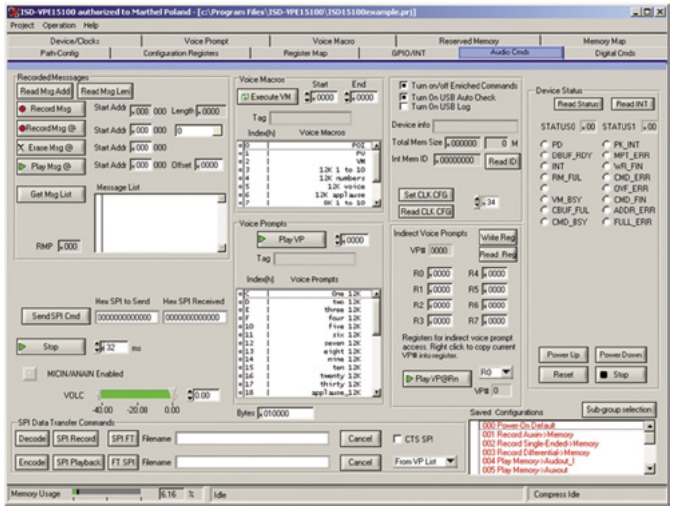
Ekran tworzenia Voice Macro pokazano na **rysunku 21**. Nowe makro jest tworzone po kliknięciu na przycisk *NewVM*. Tworzenie nowego Voice Macro jest bardzo proste. Jeżeli chcemy dodać komunikat Voice Propmpt, to wybieramy go z listy w oknie *Voice Prompts*.

Po skonfigurowaniu układu, wczytaniu Voice Propmpt, utworzeniu niezbędnych makr w oknie Memory Map tworzy się plik do zaprogramowania pamięci Flash – przycisk *Create Programing File*. Obok tego przycisku jest nieaktywny przycisk *Burn Device*, przeznaczony do zaprogramowania układu. Przycisk staje się aktywny po wykryciu połączenia układu z aplikacją ISD-VPE15100.

Współczesne aplikacje uruchamiane na komputerach PC w większości przypadków komunikują się z urządzeniami zewnętrznymi przez port USB, a układy rodziny ISD151xx wykorzystują do komunikacji z Hostem interfejs SPI. Żeby była możliwa komunikacja pomiędzy komputerem a układem niezbędny jest konwerter USB/SPI. Taka przejściówka nazywana ISD-ES-MINI-USB jest oferowana przez firmę Nuvoton (**fotografia 22**). Razem z ISD-VPE15100 dostarczany jest driver USB. Ponieważ dzięki firmie Marthel, dystrybutorowi produktów Nuvotona, dysponowałem interfejsem ISD-ES-MINI-USB oraz płytką ewaluacyjną ISD-DEMO15100 z układem ISD15116 (**rysunek 23**), to przeprowadzenie testów działania tego zestawu z programem ISD-VPE15100 nie stanowiło problemu.

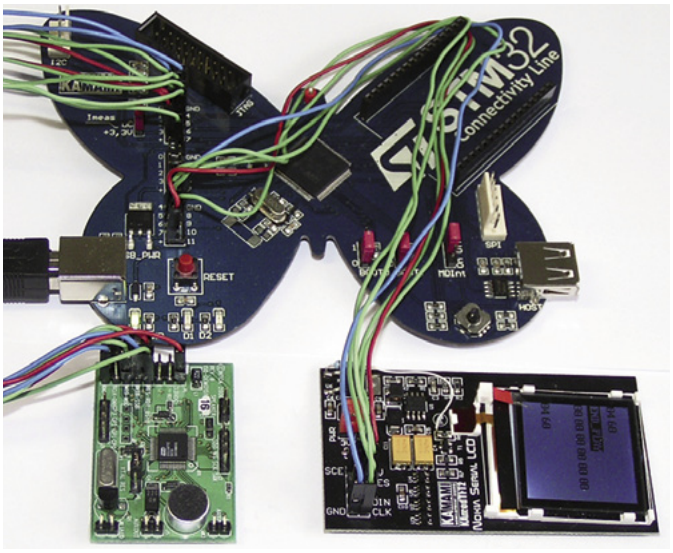


Rysunek 23. Moduł ISD-DEMO15100



Rysunek 24. Ekran Audio Cmds.

Po podłączeniu modułu demo poprzez ISD-ES-MINI-USB do komputera uzyskujemy możliwość zapisania pamięci plikiem wynikowym, ale też możliwość testowania wgranych Voice Prompts i makr bezpośrednio z układu. Po uzyskaniu połączenia z modułem w programie pojawia się dodatkowe okno nazwane *Audio Cmds* (**rysunek 24**). Ponieważ moduł nie ma wbudowanego wzmacniacza mocy, skonfigurowałem ścieżkę sygnału, tak by układ sterował głośnikiem przez wyjście PWM. Klikając na *Play VP* i *Execute VM* sprawdziłem poprawność działania wgranych komunikatów. W tym oknie można też testować nagrywanie własnych komunikatów – okno *Recorded Messages*. Przez nagrywaniem trzeba przeprogramować ścieżkę sygnału tak, by sygnał wejściowy pochodził z wbudowanego mikrofonu elektretowego i był nagrywany z żądaną kompresją (okno *Path Config*). Nagrany komunikat można odtworzyć klikając na *Play MSG*, ale przedtem trzeba ponownie przeprogramować ścieżkę sygnału z nagrywania na odtwarzanie.



Rysunek 25. Sterowanie układem ISD15116 za pomocą mikrokontrolera

Listing 1. Procedura zapisu i odczytu danych przez SPI
```
u8 WriteSpiISD( u8 data){ //zapis danej na SPI ISD
u8 i, status;
  status=0;
  SetBitISD(CLK_ISD);
  for (i=0;i<8;i++) {
    ClrBitISD(CLK_ISD);
    if ((data&0x80)==0) ClrBitISD(DIN_ISD);
    else SetBitISD(DIN_ISD);
    SetBitISD(CLK_ISD);
    data<<=1;
    status<<=1;
    if(ReadBit(DOUT_ISD)==1) status|=0x01;
  }
  SetBitISD(CLK_ISD);
  while(ReadBit(RDY_ISD)==0);//czekaj na zdjecie
zajetosci
  return(status);
}
```

Listing 2. Wykonanie komendy PowerUp
```
u16 ISDPowerUpCmd(){
u8 stat;
  ClrBitISD(SSB_ISD);
  stat=WriteSpiISD(0x10);//komenda POWER UP
  SetBitISD(SSB_ISD);
  while(1){
    stat=ISDGetStat();
    if((stat&0x44)==0x40)//czekaj na DBUF_RDY=1 i VM_
BSY=0
    break;
  }
  ISDGetInt(); //kasuj bit INT jezeli PU aktywne
  return(ISDGetStat());
}
```

Listing 3. Odczytanie rejestrów statusowych
```
u16 ISDGetStat(){
u16 stat;
u8 statl, stath;
  ClrBitISD(SSB_ISD);
  statl=WriteSpiISD(0x40);//odczytaj status
  stath=WriteSpiISD(0xff);//odczytaj status INT
  SetBitISD(SSB_ISD);
  stat=stath;
  stat<<=8;
  stat|=statl;// stat=(ststh:statl)
  return(stat);
}
```

Listing 4. Zapisanie konfiguracji układu
```
const u8 PWM_play_path[]={
0x64,0,0x44,0,0,0,0
xf0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0};//odtwarzanie
z wyjściem PWM
const u8 Mic_rec_path[]={
0x64,0,0x02,0,0x05,0x40,0x50,
0x48,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0};//nagranie
własnego komunikatu przez mikrofon
//wyslij konfiguracje
//reg- adres poczatkowy, ld - ilość bajtow, buf - bufor
z konfiguracją do zapisania
u16 ISDSendConfig(u8 reg, u8 ld, const u8 *buf){
u8 i;
  ClrBitISD(SSB_ISD);
  WriteSpiISD(0xb8);//kod Write Configuration reg
  WriteSpiISD(reg);//początkowy adres rejestru
  for(i=0;i<ld;i++)
  WriteSpiISD(buf[i]);
  SetBitISD(SSB_ISD);
  return(ISDGetStat());
}
```

Sterowanie przez sterownik mikroprocesorowy

Zachęcony pozytywnymi próbami ze sterowaniem przez firmową aplikację postanowiłem spróbować sterować układem przez sterownik mikroprocesorowy. Przestudiowałem dokładnie listę komend SPI i podłączyłem firmowy moduł ISD-DEMO15100 do płytki ewaluacyjnej STM32 Butterfly (**rysunek 25**). Do płytki został dołączony wyświetlacz przydatny w trakcie testowania procedur sterujących.

Obsługa interfejsu SPI została zaimplementowana programowo. Na **listingu 1** przedstawiono procedurę zapisania 8-bitowej danej za pomocą interfejsu SPI. Procedura jednocześnie odczytuje i zwraca bajt statusowy lub danych z układu. Testy działania polegały głównie na odtwarzaniu

zapisanych wcześniej Voice Prompt oraz na zapisaniu i odtwarzaniu własnych komunikatów. Żeby to zrealizować, trzeba umieć ustawiać ścieżkę sygnałową oraz wykonywać szereg komend SPI. Niezbędne jest też odczytywanie i interpretowanie bajta statusu i statusu przerwań. Po włączeniu zasilania układu trzeba wykonać komendę programowego włączenia zasilania *Power UP* (**listing 2**).

Po wysłaniu komendy w pętli jest odczytywany status układu. Zakończenie procesu programowego załączania napięcia jest sygnalizowane ustawieniem się bitu *DBUF_RDY* i wyzerowaniem *VM_BSY*. Na koniec żeby skasować wskaźnik przerwania odczytywany jest rejestr statusowy przerwań. Funkcja zwraca 16-bitową wartość, gdzie 8 starszych bitów to rejestr statusu przerwania, a 8 młodszych to rejestr statusu układu. Rejestry statusowe są odczytywane przez funkcję ISDGetStat() (**listing 3**).

W konfigurowaniu ścieżki sygnałowej wykorzystałem dane generowane przez ekran *Config Path*. Bajty konfiguracji dla odtwarzania i nagrywania własnych komunikatów zostały umieszczone w osobnych tablicach. Te tablice z procedurą wpisywania konfiguracji zostały pokazane na **listingu 4**.

Procedura nagrywania własnego komunikatu jest bardziej skomplikowana (**listing 5**). Po skasowaniu flag przerwania układ jest konfigurowany do nagrywania przez mikrofon. Potem jest wysyłana komenda startu nagrywania od 24-bitowego adresu zawartego w argumencie *adrs*. W zwróconym przez układ bajcie statusu sprawdzamy, czy nie wystąpiły błędy przepełnienia pamięci (*FULL_ERR*) lub niewłaściwego adresu (*ADDR_ERR*). Jeżeli tak, to jest wyświetlana informacja na ekranie wyświetlacza i procedura kończy działanie. Jeżeli nie ma błędów, to w pętli jest sprawdzane, czy w trakcie nagrywania nie wystąpiło przepełnienie pamięci, lub nie został naciśniety przycisk stopu nagrywania. Po naciśnięciu klawisza wysyłana jest komenda STOP. Po zakończeniu nagrywania jest wysyłana komenda odczytująca z układu adres początkowy i ilość zapisanych sektorów. Te informacje są zapisywane w buforze. Wskaźnik do bufora jest przekazywany w argumencie procedury.

Ten sposób nagrywania nie może zastąpić wgrywanych Voice Prompt, bo nie znamy dokładnego adresu końca nagrania. Po nagraniu znamy adres początku i ilość zapisanych sektorów. Nagrany materiał jest odczytywany przez procedurę *PlayMessageAddr* (**listing 6**).

Na początku jest konfigurowana ścieżka sygnału, tak by odtwarzane dane były kierowane do modulatora PWM. Potem jest inicjowana komenda Play Message At Address. Adres początkowy nagrania jest zawarty w argumencie addr. Ponieważ komenda dopuszcza rozpoczęcie odtwarzania z przesunięciem w stosunku do początku nagrania, to drugim argumentem jest offset.

Odtwarzanie trwa do naciśnięcia przycisku STOP. Wtedy jest wykonywana komenda END i na ekranie wysiedlany komunikat STOP PLAY. Kiedy odtwarzanie dobiegnie końca, to w rejestrze statusowym przerwania jest ustawiany bit CMD_FIN. Program testuje ten bit i jeżeli zostanie ustawiony, to procedura kończy działanie, a na ekranie jest wyświetlany komunikat END PLAY.

Jeżeli w układzie są zapisane przez aplikację ISD-VPE15100 komunikaty Voice Prompt, to w prosty sposób można je odtwarzać komedami SPI (**listing 8**).

Listing 5. Procedura nagrywania własnego komunikatu
```
u16 RecMessageAdd(u32 adrs, u8 *buf){
u16 stat;
  ISDGetInt();//kasuj flagi przewan
  LCDPutStr(„START_REC”, 0,0, LARGE, WHITE,BLACK);
  ISDSendConfig(0,28,Mic_rec_path);//konfiguracja do
nagrywania przez mikrofon
  stat=ISDRecMsgAdd(adrs);//inicjalizacja nagrywania
z adresem
  if((stat&(1<<8))==(1<<8))//FULL_ERR {
    LCDPutStr(„FULL_ERR_P”, 80,0, LARGE, WHITE,BLACK);
    return(stat);}
    if((stat&(1<<9))==(1<<9))//ADDR_ERR {
      LCDPutStr(„ADDR_ERR”, 80,50, LARGE, WHITE,BLACK);
      return(stat);}
    while(1){
    if(ReadBit(KEY_STOP)==0){
    ISDStop();
    LCDPutStr(„STOP_REC”, 80,0, LARGE, WHITE,BLACK);
    break;}
  stat=ISDGetStat();
  if((stat&(1<<8))==(1<<8))//FULL_ERR {
    LCDPutStr(„FULL_ERR_K”, 80,0, LARGE, WHITE,BLACK);
    break;}
  }
  ISDReadRecAddr(buf);
  return(ISDGetStat());
}
```

Listing 6. Procedura odtwarzania nagranego komunikatu
```
u16 PlayMessageAddr(u32 addr, u16 offset){
u16 stat;
  ISDSendConfig(0,28,PWM_play_path);//konfiguracja
odtwarzania przez PWM
  ISDGetInt();//kasuj flagi przerwania
  stat=ISDGetStat();
  if(stat!=0x40) //DBUF_RDY=1
  return(ISDGetStat());//nie mozna wykonać komendy
  ClrBitISD(SSB_ISD);
  WriteSpiISD(0x3c);//kod Play message at address
  WriteSpiISD(addr>>16);//A[23:16]
  WriteSpiISD(addr>>8);//A[15:8]
  WriteSpiISD(addr);//A[7:0]
  WriteSpiISD(offset>>8);//OFF[15:8]
  WriteSpiISD(offset);//OFF[7:0]
  SetBitISD(SSB_ISD);
  stat=ISDGetStat();
  if((stat&(1<<9))==(1<<9))//ADDR_ERR {
    LCDPutStr(„ADDR_ERR”, 80,50, LARGE, WHITE,BLACK);
    return(stat);}
    while(1){
    if(ReadBit(KEY_STOP)==0){
      ISDStop();
      LCDPutStr(„STOP_PLAY”, 80,0, LARGE, WHITE,BLACK);
      break;}
    stat=ISDGetStat();
    DispHex(20,0,stat>>8);
    DispHex(20,20,stat);
    if((stat&1<<10)==(1<<10)){//CMD_FIN
    LCDPutStr(„END_PLAY”, 80,0, LARGE, WHITE,BLACK);
    break;}
  }
  return(ISDGetStat()); //powrót z bajtami statusu
i statusu przerwania
}
```

Listing 7. Odczytanie adresu i liczby zapisanych sektorów po nagraniu
```
//odczytaj adres i l.sektorow w czasie i po nagrywaniu
//buf[0]addh, buf[1]addm, buf[2]addl, buf[3]lsech, buf[4]
lsecl
u16 ISDReadRecAddr(u8 *buf){
u8 i;
  ClrBitISD(SSB_ISD);
  WriteSpiISD(0x42);//kod RD_MSG_ADD
  for(i=0;i<5;i++) buf[i]=WriteSpiISD(0xff);
  SetBitISD(SSB_ISD);
  return(ISDGetStat());
}
```

Podsumowanie

Układy rodziny ISD151xx to zaawansowane cyfrowe układy do rejestracji dźwięku. Mogą być bardzo elastycznie konfigurowane, a wbudowane kompresory/dekompresory pozwalają na zapisanie nawet 30 minut komunikatów słownych w najbardziej pojemnym układzie ISD15132. Producent nie zapomniał też o dźwięku lepszej jakości, wyposażając układy w interfejs I²S i możliwość operowania na 16-bitowych, nieskompresowanych danych audio.

Firmowe narzędzia pozwalają na bardzo efektywne wykorzystanie pamięci przez stosowanie mechanizmów Voice Prompt i Voice Macro. Nagranie odpowiedniej ilości Voice prompt i grupowanie ich w makra pozwala na zbudowanie prostych cyfrowych syntetyzatorów komunikatów dźwiękowych. Wielka zaleta tych układów jest stosunkowo prosta aplikacja przy olbrzymich możliwościach.

Tomasz Jabłoński, EP

Listing 8. Odtworzenie Voice Prompt
```
//odtworz Voice Prompt
u16 ISDPlayVoice(u16 idx){
u8 stat;

  stat=ISDGetStat();
  if(stat!=0x40) //DBUF_RDY=1
  return(ISDGetStat());//nie mozna wykonać komendy
  ClrBitISD(SSB_ISD);
  WriteSpiISD(0xa6);//kod Play Voice Prompt Index
  WriteSpiISD(idx>>8);//zapisz index voice prompt
  WriteSpiISD(idx);
  SetBitISD(SSB_ISD);
  return(ISDGetStat());
}
```