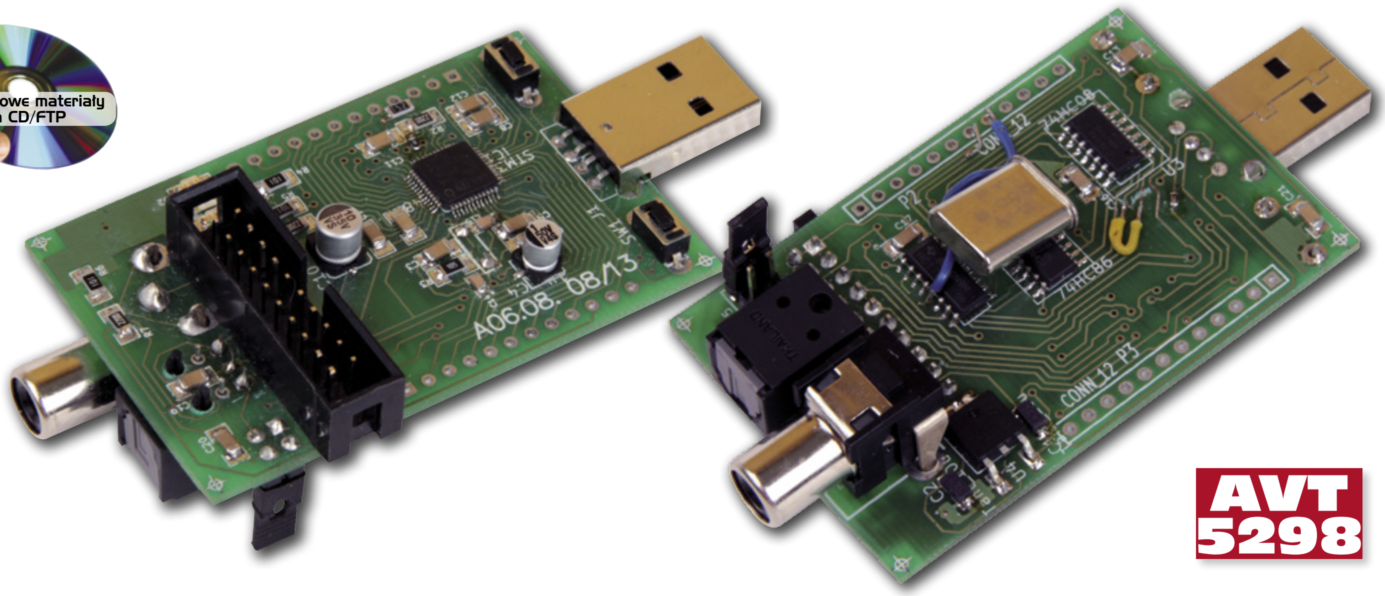


AVT 5298

Konwerter USB na S/PDIF

Komputerowe przetworniki C/A dla sygnałów audio, wbudowane w płyty główne komputerów osobistych, to zwykle sprzęt z tzw. średniej półki. Bardziej wymagający amatorzy muzyki z komputera powinni wyposażyć się w lepszy sprzęt np. zestaw z zewnętrznym wzmacniaczem audio oraz wejściem cyfrowym S/PDIF. Nasz interfejs umożliwia wykonanie połączenia pomiędzy komputerem PC a takim zestawem muzycznym.

Rekomendacje: *Konwerter przyda się, gdy komputer PC jest używany jako magazyn do przechowywania utworów muzycznych, chociażby w popularnym formacie MP3.*

AVT-5298 w ofercie AVT:
AVT-5298A – płytka drukowana

Podstawowe informacje:
- Konwersja sygnału USB na S/PDIF
- Wyjście optyczne (Toslink) oraz RCA (kabel koncentryczny 75 Ω)
- Zasilanie z portu USB
- Częstotliwość próbkowania 44,1 kHz i 48 kHz
- Prosta konstrukcja
- Współpraca z popularnymi programami odtwarzaczy multimedialnych

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 16732, pass: 630v2nfb
- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w Wykazie elementów kolorem czerwonym

Projekty pokrewne na CD/FTP:
(wymienione artykuły są w całości dostępne na CD)

AVT-5148	Stereofoniczny kodek z interfejsem S/PDIF (EP 9/2008)
AVT-574	Przełącznik optyczny SPDIF (EP 5/2004)
AVT-566	Procesor audio z wejściem S/PDIF (EP 3-4/2004)
AVT-514R	Regenerator optyczny S/PDIF (EP 6/2003)
AVT-514CO	Konwerter S/PDIF Coaxial → Optical (EP 6/2003)
AVT-514OC	Konwerter S/PDIF Optical → Coaxial (EP 6/2003)

Bardziej wymagający amatorzy muzyki z komputera powinni wyposażyć się np. w zestaw z zewnętrznym wzmacniaczem audio oraz wejściem cyfrowym S/PDIF. Jednak aby doprowadzić sygnał z komputera, trzeba użyć odpowiedniego nadajnika sygnału S/PDIF. Można taki przetwornik kupić w sklepie, ale odbiera to przyjemność z tworzenia własnych konstrukcji i możliwość poeksperymentowania z wykonanym układem. Dlatego zdecydowałem się na wykonanie własnego urządzenia.

Standard S/PDIF

S/PDIF to specyfikacja standardu transmisji cyfrowego sygnału audio opracowana przez firmy Sony i Philips. Intencją konstruktorów było uproszczenie innego standardu AES/EBU w celu udostępnienia zalet transmisji cyfrowej większemu gronu konsumentów. Oryginalny AES/EBU ma więcej możliwości, ale też jest bardziej skomplikowany układowo. Teraz jego implementacja nie jest już problemem, ale dwie dekady temu koszt tego rozwiązania był istotnym czynnikiem hamującym rozpowszechnianie wyposażonego w nie sprzętu wśród użytkowników nieprofesjonalnych.

Dokumenty opisujące interfejs S/PDIF zostały opracowane przez organizację IEC (60958-3). Niestety, za pełną specyfikację trzeba słono zapłacić. Ja poradziłem sobie, czerpiąc informacje z innych źródeł, w szczególności w sieci Internet można znaleźć kilka publikacji pozwalających dość dokładnie zapoznać się z tym interfejsem. Standard definiuje warstwę fizyczną, sposób kodowania poszczególnych bitów oraz format struktur porządkujących przesyłane dane. W warstwie fizycznej są dostępne dwie możliwości: sygnał optyczny przesyłany światłowodem typu Toslink lub sygnał elektryczny przesyłany kablem koncentrycznym o impedancji 75 Ω zakończonym wtykami RCA. Dane transmitowane są szeregowo bez wydzielonego kanału dla sygnału zegarowego, dlatego też zdecydowano się na kodowanie BMC, które umożliwia odtworzenie sygnału zegarowego z kanału danych, kosztem podwojenia częstotliwości pierwotnego sygnału wejściowego. Zasady przy tworzeniu takiego sygnału są następujące:

– sygnał zawsze zmienia polaryzację na granicy bitów,
– transmisja jedynki oznacza dodatkowo konieczność zmiany polaryzacji w środku okresu przeznaczonego na transmisję jednego bitu,
– transmisja zera oznacza, że przez cały okres przeznaczony na transmisje bitu polaryzacja się nie zmienia.

Opisane wyżej zasady obowiązujące podczas transmisji sygnału danych zilustrowano na **rysunku 1**. Transmitowane słowo danych składa się z 32 bitów. Ich znaczenie opisano w **tabeli 1**.

Taka struktura bitów, jak opisana w tabeli 1, tworzy ramkę dla każdego z kanałów. Dodatkowo, wprowadzone jest pojęcie bloku, na który składają się 192 ramki. 4-bitowy początek każdej struktury zaczyna się od sekwencji niebędącej kodem BMC. Dzięki temu odbiornik jest w stanie rozpoznać początek bloku, ramki i subramki. Dodatkowo, początkowe sekwencje mają ułatwić procedurę synchronizacji zegara.

Format subramki przewiduje, co najwyżej 24-bitową rozdzielczość sygnału o wartościach znormalizowanych do przedziału [–1; 1). Jeśli korzystamy z mniejszej rozdzielczości, to mniej znaczące pozycje należy uzupełnić 0. Bity *Channel Status Information* z każdej subramki tworzą 192-bitowy ciąg w obrębie pojedynczego bloku pozwalający zakodować takie informacje, jak liczba kanałów czy częstotliwość próbkowania, przy czym większość ciągu jest nieustandaryzowana, a praktyka pokazuje, że współczesne odbiorniki doskonale radzą sobie bez informacji udostępnianych w tym kanale.

Interfejs USB

Jako interfejs dla komputera PC zastosowano USB. Jako bazę wykorzystano klasę „USB Audio class". Jej dokumentacja jest dostępna za darmo w Internecie na stronie komitetu standaryzacyjnego. Zdefiniowano w niej uniwersalną strukturę urządzenia audio. Topologię takiego urządzenia, jak i właściwości poszczególnych bloków, opisują tzw. deskryptory. Dla opisywanego konwertera niezbędne są następujące deskryptory:

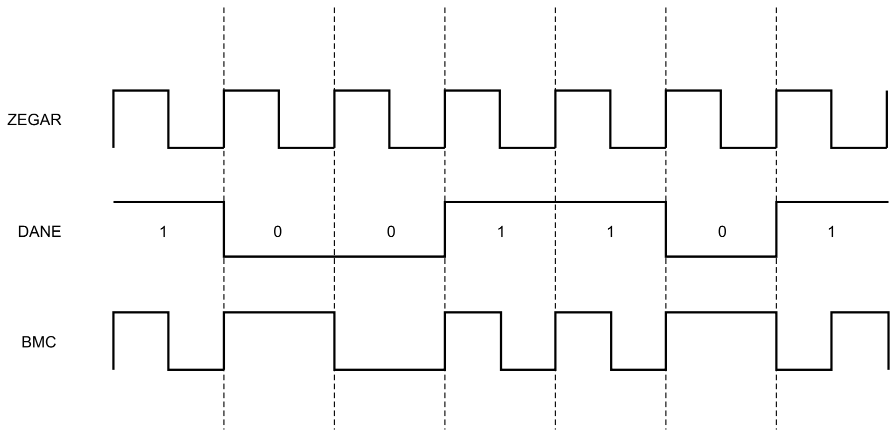
– USB Standard Device Descriptor
– Interface 0 – Audio Control
  – Audio Control Header
  – Input Terminal USB Streaming
  – Feature Unit – Mute
  – Output Terminal: SPDIF
– Interface 1 – Audio Streaming: Alternate Setting 0
– Interface 1 – Audio Streaming: Alternate Setting 1
  – General – PCM
  – Type I Format – freq 44100 Hz, 48000 Hz
  – Endpoint
– Interface 1 – Audio Streaming: Alternate Setting 2
  – General – AC3
  – Type III Format – freq 48000 Hz
  – Endpoint

Jak można zauważyć, konwerter zgłasza dwie częstotliwości próbkowania tj. 44,1 i 48 kHz dla sygnału PCM oraz wsparcie dla sygnałów kodowanych standardem AC3 i wysyłanych z prędkością 48 kHz (dane tego formatu będą transmitowane bezpośrednio przez łącze S/PDIF).

Do transmisji danych USB wymagających gwarancji na dostępność ustalonego pasma, standard USB definiuje specjalny tryb nazywany izochronicznym. W tym trybie kontroler USB cyklicznie odpytuje dane urządzenie lub cyklicznie (okres definiowany polem *bInterval* w deskryptorze EP) wysyła dane do danego urządzenia, zgodnie z wymaganiami opisanymi za pomocą deskryptorów. Taki tryb nie zapewnia retransmisji w przypadku błędów, co jest dopuszczalne dla strumienia audio.

Budowa

Schemat ideowy konwertera przedstawiono na **rysunku 2**.



Rysunek 1. Sposób kodowania bitów

Wykaz elementów

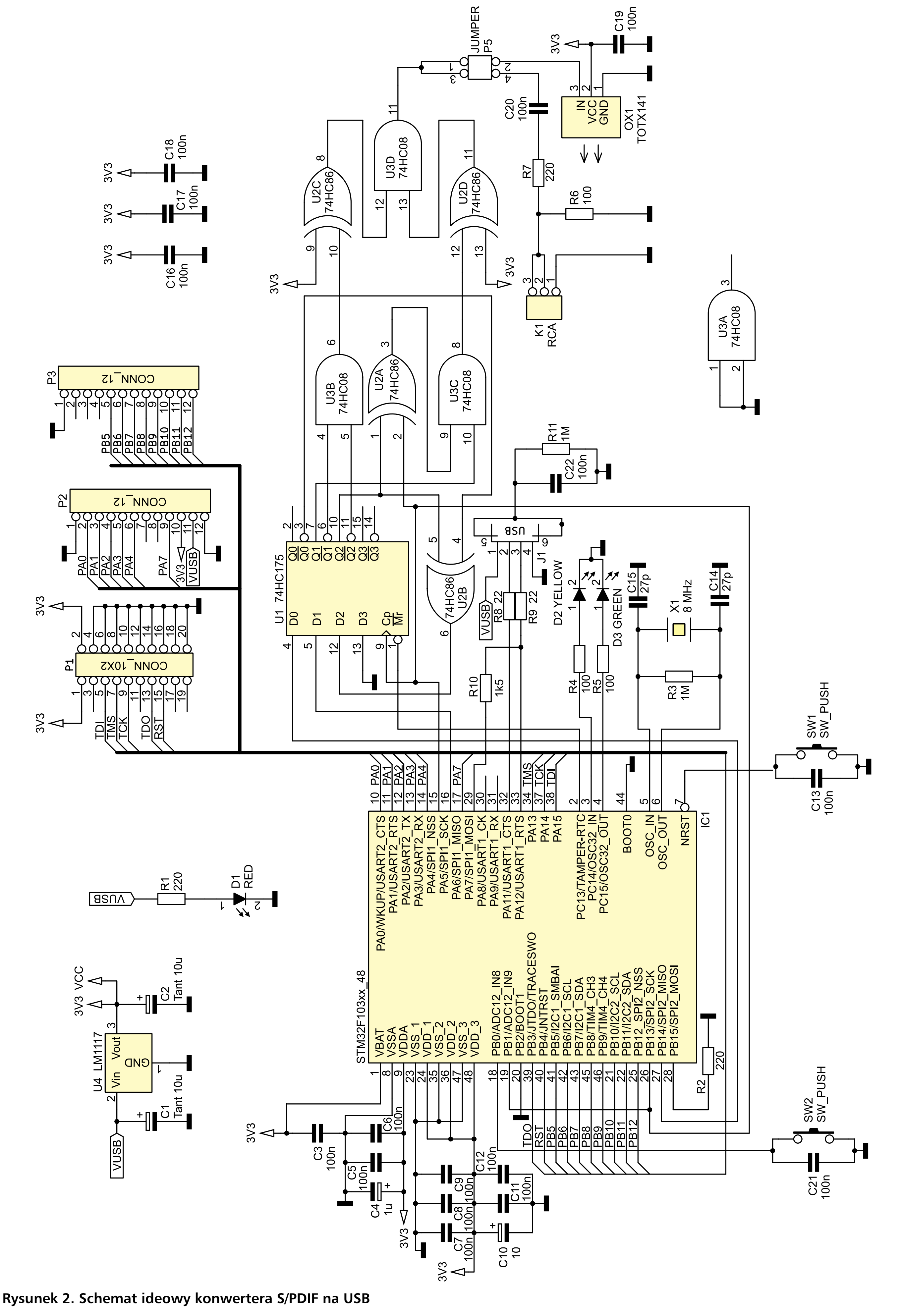
Rezystory:
R1, R2, R7: 220 Ω
R3, R11: 1 MΩ
R4...R6: 100 Ω
R8, R9: 22 Ω
R10: 1,5 kΩ
Kondensatory:
C1, C2: 10 μF (tantalowy „A")
C3, C5...C9, C11...C13, C16...C22: 100 nF
C4: 1 μF
C10: 10 μF
C14, C15: 27 pF
Półprzewodniki:
IC1: STM32F103C8T6
D1: dioda LED (czerwona)
D2: dioda LED (żółta)
D3: dioda LED (zielona)
U1: 74HC175
U2: 74HC86
U3: 74HC08
U4: LM1117
Inne:
J1: gniazdo USB A
K1: gniazdo RCA
OX1: nadajnik optyczny TOTX141
P1: Gniazdo 2×10
P2, P3: linijka 12-pinowa (PB12)
P5: jumper 2×2
SW1, SW2: mikroprzełącznik
X1: 8 MHz

Wykaz elementów programatora

C1: 1 μF
C2: 100 nF
J1: DB25
P1: PB10×2
Q1: DTC 114ES
R1...R3: 100 Ω
R4...R9: 51 Ω
U1: 74HC244

Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym

Tabela 1. Znaczenie bitów w słowie danych S/PDIF		
Nr bitu	**Nazwa pola**	**Opis**
0..3	Preambule	Wzorzec preambuły zależny od pozycji w bloku, nie jest kodem BMC
4..7	Aux	Miejsce na informacje pomocnicze lub dodatkowe 4 bity na próbkę audio
8..27	Data	Próbka audio
28	Valid	Informacja o poprawności próbki. Urządzenie może wysyłać uszkodzoną próbkę, aby nie stracić synchronizacji
29	Subcode	Bit składający się na 192-bitowy ciąg takich bitów zebranych z całego bloku. To dodatkowy kanał do przesyłania informacji opisujących strumień audio
30	Channel Status	Informacja kontrolno-statusowa. Podobnie jak Subcode, jest to fragment 192-bitowego ciągu w obrębie bloku
31	Parity	Bit parzystości, przy czym bity 0...3 są wyłączone z kalkulacji

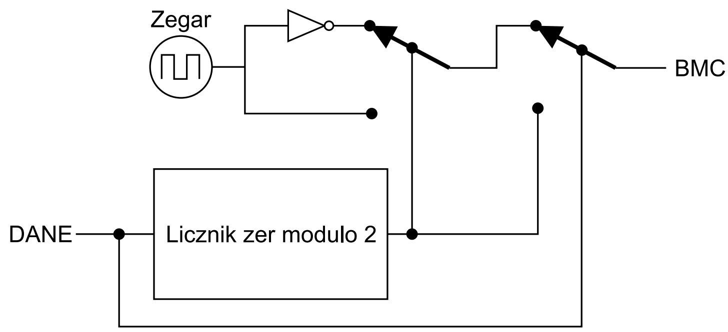


Rysunek 2. Schemat ideowy konwertera S/PDIF na USB

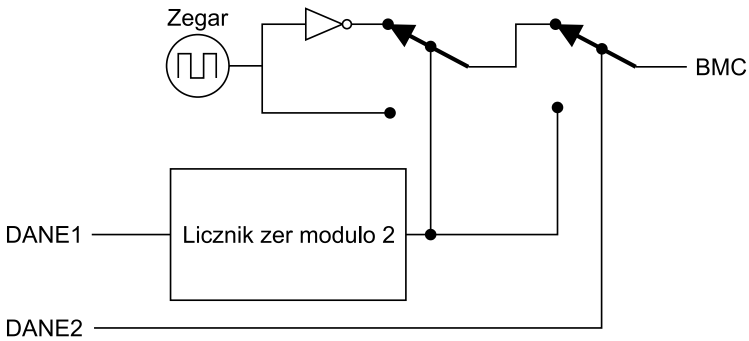
Interfejs S/PDIF jest na tyle nieskomplikowany, że można próbować realizacji na drodze programowej. Przed przystąpieniem do wyboru wariantu należałoby obliczyć dostępny budżet cykli zegara na konstrukcję pojedynczej ramki. Mikrokontroler STM32F103 może być taktowany zegarem 72 MHz. Najwyższa, dostępna częstotliwość próbkowania to 48 kHz. Czyli na przygotowanie każdej próbki zawierającej dane dla dwóch kanałów jest 1500 cykli zegara. Każda próbka to z kolei 64 bity, a przy uwzględnieniu kodu BMC 128 bitów do przygotowania i transmisji. Na przygotowanie każdego bitu jest 11 cykli. Jeśli mielibyśmy przy tym programowo sterować którymś z pinów, to od razu widać, że jest to zadanie niewykonalne.

Problem braku cykli można spróbować rozwiązać z użyciem sprzętowego, wbudowanego w STM32F103 bloku SPI. Dodatkowo, dane dla modułu SPI można przesyłać z użyciem DMA, dzięki czemu program prawie nie musi zajmować się transmisją. W tym miejscu trzeba jednak zauważyć, że wartość bitu n+1 zależy od bitu n, więc potencjalne obliczenia w przeliczeniu na pojedynczy bit mogą być bardzo pracochłonne. Można by co prawda wymyślić jakiś bardziej wyszukany algorytm (np. stablicować kody dla np. długości 8 bitów), jednak poza pracą związaną z przeliczaniem należy jeszcze obliczać bit parzystości oraz obsługiwać strumienie i sekwencje sterujące z USB. Dodatkowo, jak się później okaże, generowanie sygnału zegarowego nie będzie zadaniem trywialnym i będzie miało znaczny udział w wykorzystaniu mocy procesora. Wobec powyższego zdecydowałem się na dodatkowe rozwiązanie sprzętowe, które pozwoliłoby odciążyć CPU. Zaprojektowałem układ złożony z kilku bramek zawartych w układach U1...U3. Nie jest to typowy układ kombinacyjny. Decydując się na klasyczne rozwiązanie do realizacji BMC, musiałbym nieco rozbudować układ i dodatkowo potrzebowałbym zegara o dwa razy wyższej częstotliwości niż bitowa prędkość strumienia audio. Wykorzystałem spostrzeżenie, że do zbudowania „1" w kodzie BMC można by bezpośrednio użyć sygnału zegarowego. Układ musi jeszcze zliczać modulo 2 liczbę „0" i rezultat tego zliczania wykorzystywać do generowania bitu parzystości, a także do ewentualnego odwracania polaryzacji zegara. Na **rysunku 3** zilustrowano zasadę działania zaprojektowanego układu.

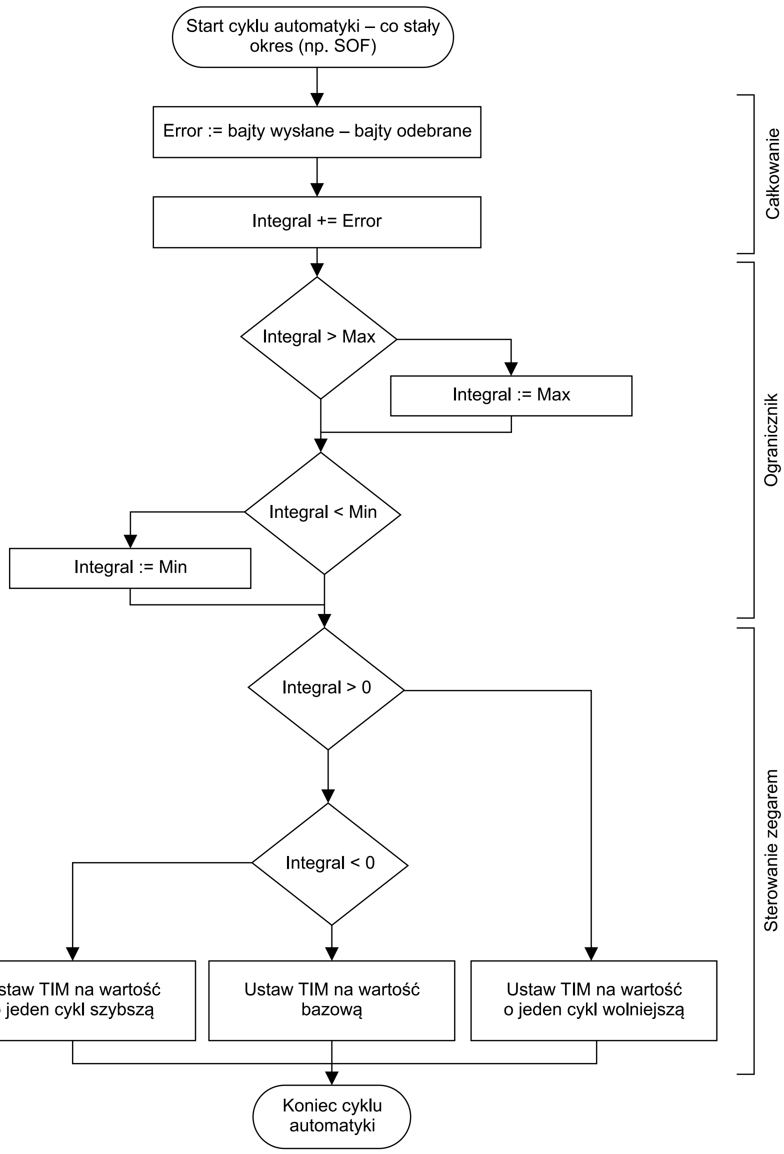
Takie podejście nie jest pozbawione wad. Sygnał zegarowy powinien mieć wypełnienie 50%. W takim układzie występuje również ryzyko powstania pewnych zakłóceń np. szpilek przy zmianach stanu spowodowanych różnym czasem propagacji, jednak praktyka pokazała, że to niestandardowe podejście działa. Jedynym problemem, który pozostaje do rozwiązania, jest generowanie początku każdej subramki, który nie jest ko-



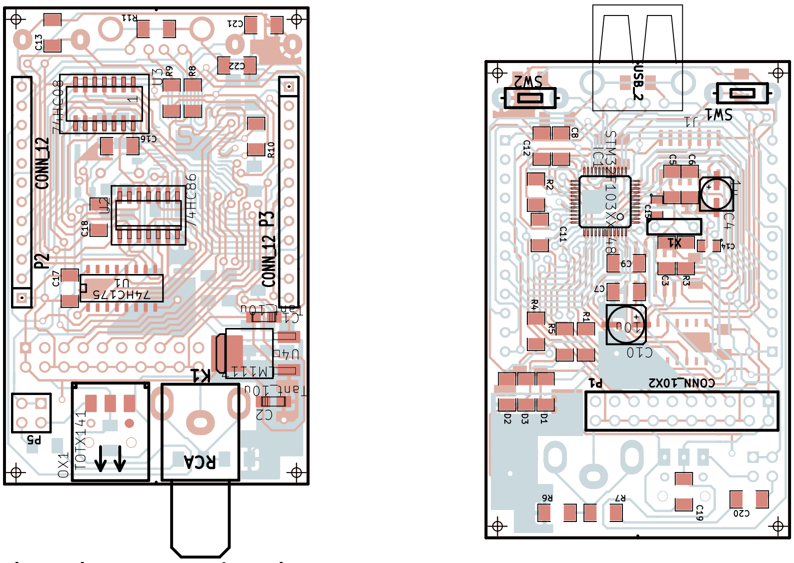
Rysunek 3. Zasada działania układu kombinacyjnego



Rysunek 4. Klucz wyjściowy



Rysunek 5. Algorytm działania sterownika „I"



Rysunek 6. Schemat montażowy konwertera S/PDIF na USB

dem BMC. Rozwiązaniem jest wyprowadzenie z układu dwóch wejść pozwalających na odseparowanie sterowania licznikiem od sterowania końcowym układem wyboru między zegarem a wyjściem tego licznika. Dzięki temu mikrokontroler może dość dowolnie kształtować sygnał wyjściowy, a jednocześnie podając te same dane na oba wejścia, zapewnić generowanie kodu BMC.

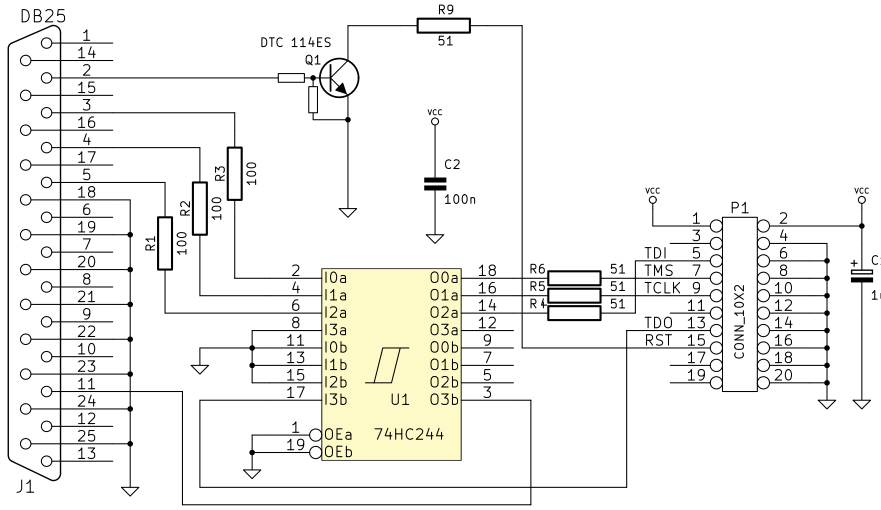
Na schemacie ideowym licznik tworzą bramka U2B i przerzutniki U1-0 i U1-2. Bramka U2A służy do odwracania polaryzacji zegara, a pozostałe bramki są realizacją klucza wyjściowego pokazanego na **rysunku 4**.

Zgodnie z założeniami konwerter miał wspierać dwie częstotliwości próbkowania: 44,1 kHz oraz 48 kHz. Częstotliwość sygnału, który musi być doprowadzony do nadajnika, aby układ mógł spełnić te założenia, musi wynosić, odpowiednio: 2,8224 MHz i 3,072 MHz. Mając sygnał zegarowy o częstotliwości 72 MHz, nie jest łatwo wygenerować sygnały o takich częstotliwościach, jeśli jednak dopuścimy fluktuację fazy, to uda się to zrobić. Współczesny sprzęt Hi-Fi najczęściej ma pewien bufor, który pozwoli zniwelować niedoskonałości związane z jitterem. Dodatkowo, standard S/PDIF również dopuszcza pewne wahania fazy.

Częstotliwość 3,072 MHz można rozpisać na 36 okresów po 23 cykle i 28 okresów po 24 cykle. Zaprogramowanie takiej procedury pracy zegara wymagać będzie więcej pracy na poziomie oprogramowania, ale w jej wyniku dostaniemy dokładnie 1500 cykli w ciągu 1/48 ms. W taki sposób można wygenerować sygnał o potrzebnej częstotliwości. Znacznie trudniej jest w wypadku 2,8224 MHz. Tu nie da się rozpisać 1/44,1 ms na sumę okresów o długościach będących wielokrotnością cyklów sygnału zegarowego CPU. Widać stąd, że program trzeba będzie wyposażyć w mechanizm korekty częstotliwości, najlepiej w oparciu o zewnętrzny wzorzec. W długim okresie takim wzorcem mogą być dane odbierane poprzez USB. Generalnie obowiązuje zasada, że liczba danych odebranych musi równać się liczbie danych wysłanych przez interfejs S/PDIF. Różnica badana w jakichś stałych odstępach czasu może stanowić bazę do wyliczenia poprawki. Jak okazało się w praktyce, taka poprawka jest konieczna nie tylko dla częstotliwości próbkowania 44,1 kHz, ale również dla

48 kHz. Kwarc pomimo swojej doskonałej stabilności i dokładności może różnić się od tego zainstalowanego w komputerze i z czasem może okazać się, że taka skumulowana różnica powoduje sytuację, w której nadajnik nie ma danych do nadania lub odbiornik USB nie ma gdzie przechować próbek. Najprostszym podejściem w takim wypadku jest albo wysłanie przez S/PDIF dodatkowej próbki zerowej, albo pominięcie pojedynczej próbki. Nie jest to duży problem przy transmisji PCM, gdyż takie przekłamanie próbki jest prawie niezauważalne dla słuchacza, o ile nie powtarza się zbyt często. Gorzej, jeśli przesyłamy dane skompresowane, niebędące bezpośrednio próbkami sygnału audio. Wtedy wysłanie dodatkowej próbki albo jej pominięcie da słyszalną przerwę. Dla takich sytuacji stworzyłem proste sterowanie, które w świecie automatyki zostałoby nazwane sterownikiem typu „I". Algorytm dla takiego sterowania przedstawia **rysunek 5**.

Wiadomo, że sterowniki typu „I" zmniejszają błąd do zera. Jednak ich czas ustalania jest dość długi, co może powodować pewne zakłócenia w odbiorze tuż po uruchomieniu transferu danych z USB lub po zmianie częstotliwości próbkowania (analogia do skoku jednostkowego w automatyce). Prostym rozwiązaniem w tym wypadku jest ograniczenie na wartość skumulowanego błędu, które widać na rysunku. Dodatkowo, deskryptor EP (Endpoint) ma pole informujące host o czasie stabilizacji zegara (wLockDelay). W ten sposób rozwiązanie tego problemu można częściowo przerzucić na sterownik działający po stronie hosta. Zerowa wartość sygnału błędu oznacza programowanie częstotliwości zegara na wartość najbliżej pasującą do żądanego sygnału. W przypadku, gdy błąd jest dodatni, częstotliwość jest programowana na wartość nieco większą, a dla ujemnych na nieco mniejszą. Dzięki temu układ automatycznie dostosowuje się do prędkości odbieranych z USB danych.



Rysunek 7. Schemat programatora kompatybilnego z popularnym Wigglerem

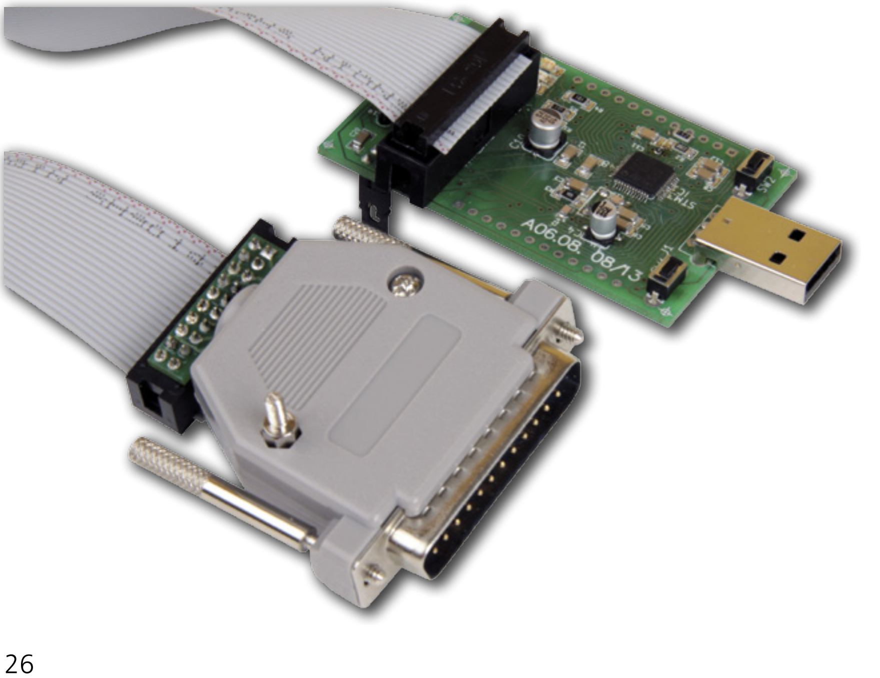
Montaż i uruchomienie

Schemat montażowy konwertera pokazano na **rysunku 6**. Projekt płytki drukowanej wykonano z użyciem darmowego programu KiCAD. Prawie wszystkie użyte komponenty są przeznaczone do montażu SMT. Pewną trudność może sprawić przylutowanie mikrokontrolera, a to ze względu na gęsty raster wyprowadzeń.

Złącza P2 i P3 są opcjonalne, doprowadzone do nich nieużywane linie procesora można wykorzystać do realizacji innych funkcji.

Konwerter jest przystosowany do łączenia światłowodem oraz kablem koncentrycznym. Wyboru medium transmisji dokonuje się za pomocą zworki. Nie jest konieczny montaż obu rodzajów wyjść, można ograniczyć się do jednego i na stałe wlutować odpowiednie połączenie zamiast zwory.

Po montażu należy zaprogramować mikrokontroler. Ja posłużyłem się w tym celu własnoręcznie wykonanym programatorem kompatybilnym z popularnym Wigglerem. Jego schemat pokazano na **rysunku 7**. Do sterowania programatorem zastosowano program OpenOCD (*Open On-Chip Debugger*). W celu zaprogramowania układu wystarczy wejść do katalogu *sw/sound_dev* z kodem źródłowym, bibliotekami oraz innymi plikami pomocniczymi i wydać plecenie *make flash*. Jest wymagane, aby mieć zainstalowane odpowiednie pakiety oprogramowania, a w szczególności kompilator GCC ze wsparciem dla architektury ARM Cortex-M3 oraz OpenOCD. Używałem GCC zmodyfikowanego przez firmę CodeSourcery. Wersja „*Sourcery G++ Lite Edition*" jest do pobrania za darmo ze strony tej firmy.



Po zaprogramowaniu układ jest gotowy do pracy. Przycisk SW1 umożliwia restart procesora. Dodatkowy przycisk SW2 służy do przełączania trybów między PCM a AC3/DTS passthrougth. Układ był testowany w systemach Linux Debian i Windows 7. Pod kontrolą obu był uruchomiony program odtwarzacza multimedialnego, który pozwala użyć przesyłania danych AC3 bez ich dekodowania na poziomie komputera PC. Niestety, w praktyce użycie tego trybu nie oznaczało przełączenia konfiguracji opisanej deskryptorami, a co za tym idzie, dane nadawane przez S/PDIF nie były oznaczone jako *non-PCM*. Ciekawe jest, że mimo to niektóre amplitunery mogły rozpoznać dane AC3 i przełączyć się w odpowiedni tryb. Mimo wszystko, na wszelki wypadek jednak dodałem możliwość ręcznego przełączania trybu przyciskiem SW2.

SW2 ma też inną funkcję. Przytrzymanie go na dłużej niż przez 2 s (po tym czasie wszystkie trzy diody LED powinny świecić jasnym światłem) zamienia ze sobą kanały stereo, a kolejne wciśnięcie na dłużej niż 2 s powoduje wprowadzenie urządzenia w tryb miksowania próbek z obu kanałów. Diody LED, poza sygnalizacją upływu czasu 2 s, sygnalizują również przesyłanie danych przez interfejs USB. I tak: dioda zielona oznacza tryb PCM, przy czym jej jasne świecenie sygnalizuje odbiór danych z USB. Dioda pomarańczowa oznacza tryb AC3.

Marcin Orłowski