**Listing 1. Zawartość pliku nagłówkowego do obsługi układu TDA7418**

**typedef** struct

**{**

volatile uint8\_t Input**;** //0...3 -> PD/SE4, SE1...SE3

volatile uint8\_t frontVolume**;** //0...79 -> -79dB...0dB

volatile uint8\_t rearVolume**;** //0...79 -> -79dB...0dB

volatile uint8\_t Bass**;** //0...30 -> -15dB...+15dB

volatile uint8\_t Middle**;** //0...30 -> -15dB...+15dB

volatile uint8\_t Treble**;** //0...30 -> -15dB...+15dB

**}** TDA7418config**;**

extern TDA7418config TDA7418**;** //Zmienna przechowująca interesujące nas parametry TDA7418

#define TDA7418\_WR\_ADDR 0x88

#define TDA7418\_RD\_ADDR 0x89

//Option

#define TEST\_MODE\_OFF (0<<7)

#define TEST\_MODE\_ON (1<<7)

#define AUTO\_ZERO\_REMAIN\_OFF (0<<6)

#define AUTO\_ZERO\_REMAIN\_ON (1<<6)

#define AUTO\_INCR\_MODE\_OFF (0<<5)

#define AUTO\_INCR\_MODE\_ON (1<<5)

//Registers and options

#define REG\_SOURCE\_SELECTOR 0x00

#define SOURCE\_SE4 0x00

#define SOURCE\_SE1 0x01

#define SOURCE\_SE2 0x02

#define SOURCE\_SE3 0x03

#define SINGLE\_ENDED (0<<7)

#define DIFFERENTIAL\_STEREO (1<<7)

/\*\*/

#define REG\_LOUDNESS 0x01

#define CENTER\_FREQ\_FLAT (0<<4)

#define CENTER\_FREQ\_400HZ (1<<4)

#define CENTER\_FREQ\_800HZ (2<<4)

#define CENTER\_FREQ\_2400HZ (3<<4)

#define LOW\_BOST (0<<6)

#define LOW\_HIGH\_BOST (1<<6)

#define LOUDNESS\_SOFT\_STEP\_ON (0<<7)

#define LOUDNESS\_SOFT\_STEP\_OFF (1<<7)

/\*\*/

#define REG\_VOLUME 0x02

#define VOLUME\_SOFT\_STEP\_ON (0<<7)

#define VOLUME\_SOFT\_STEP\_OFF (1<<7)

/\*\*/

#define REG\_TREBLE 0x03

#define TREBLE\_CENTER\_FREQ\_10KHZ (4<<5)

#define TREBLE\_CENTER\_FREQ\_12\_5KHZ (5<<5)

#define TREBLE\_CENTER\_FREQ\_15KHZ (6<<5)

#define TREBLE\_CENTER\_FREQ\_17\_5KHZ (7<<5)

/\*\*/

#define REG\_MIDDLE 0x04

#define MID\_Q\_FACT\_0\_5 (0<<5)

#define MID\_Q\_FACT\_0\_75 (1<<5)

#define MID\_Q\_FACT\_1 (2<<5)

#define MID\_Q\_FACT\_1\_25 (3<<5)

#define MID\_SOFT\_STEP\_ON (0<<7)

#define MID\_SOFT\_STEP\_OFF (1<<7)

/\*\*/

#define REG\_BASS 0x05

#define BASS\_Q\_FACT\_1 (0<<5)

#define BASS\_Q\_FACT\_1\_25 (1<<5)

#define BASS\_Q\_FACT\_1\_5 (2<<5)

#define BASS\_Q\_FACT\_2 (3<<5)

#define BASS\_SOFT\_STEP\_ON (0<<7)

#define BASS\_SOFT\_STEP\_OFF (1<<7)

/\*\*/

#define REG\_MID\_BAS\_CF 0x06

#define MID\_CENTER\_FREQ\_500HZ (0<<0)

#define MID\_CENTER\_FREQ\_1KHZ (1<<0)

#define MID\_CENTER\_FREQ\_1\_5KHZ (2<<0)

#define MID\_CENTER\_FREQ\_2\_5KHZ (3<<0)

#define BASS\_CENTER\_FREQ\_60HZ (0<<2)

#define BASS\_CENTER\_FREQ\_80HZ (1<<2)

#define BASS\_CENTER\_FREQ\_100HZ (2<<2)

#define BASS\_CENTER\_FREQ\_200HZ (3<<2)

#define BASS\_DC\_MODE\_OFF (0<<4)

#define BASS\_DC\_MODE\_ON (1<<4)

#define SMOOTHING\_FILTER\_OFF (0<<5)

#define SMOOTHING\_FILTER\_ON (1<<5)

/\*\*/

#define REG\_SPK\_ATT\_FL 0x07

#define REG\_SPK\_ATT\_RL 0x08

#define REG\_SPK\_ATT\_RR 0x09

#define REG\_SPK\_ATT\_FR 0x0A

#define REG\_SUBB\_ATT 0x0B

#define SPEAKER\_MUTED 0x60

/\*\*/

#define REG\_SOFT\_MUTE 0x0C

#define SOFT\_MUTE\_ON (0<<0)

#define SOFT\_MUTE\_OFF (1<<0)

#define SOFT\_MUTE\_TIME\_48MS (0<<1)

#define SOFT\_MUTE\_TIME\_96MS (1<<1)

#define SOFT\_MUTE\_TIME\_123MS (2<<1)

#define SOFT\_STEP\_TIME\_0\_16MS (0<<3)

#define SOFT\_STEP\_TIME\_0\_32MS (1<<3)

#define SOFT\_STEP\_TIME\_0\_64MS (2<<3)

#define SOFT\_STEP\_TIME\_1\_28MS (3<<3)

#define SOFT\_STEP\_TIME\_2\_56MS (4<<3)

#define SOFT\_STEP\_TIME\_5\_12MS (5<<3)

#define SOFT\_STEP\_TIME\_10\_24MS (6<<3)

#define SOFT\_STEP\_TIME\_20\_48MS (7<<3)

#define AUTO\_ZERO\_OFF (0<<6)

#define AUTO\_ZERO\_ON (1<<6)

/\*\*/

#define REG\_TEST 0x0D

**Listing 2. Funkcja inicjalizacji układu TDA7418**

void TDA7418init**(**void**)**

**{**

//Konfigurujemy układ TDA7418 wpisując wartości do kolejnych rejestrów w trybie autoinkrementacji

TWIstart**();**

TWIwriteByte**(**TDA7418\_WR\_ADDR**);**

TWIwriteByte**(**REG\_SOURCE\_SELECTOR**|**AUTO\_INCR\_MODE\_ON**);**

//Input Gain zależny od wejścia, Single Ended Stereo

**switch(**TDA7418**.**Input**)**

**{**

**case** INPUT\_FM**:** TWIwriteByte**(**TDA7418**.**Input**|**6**<<**3**);** **break;** //6 dB

**case** INPUT\_BT**:** TWIwriteByte**(**TDA7418**.**Input**|**8**<<**3**);** **break;** //8 dB

**default:** TWIwriteByte**(**TDA7418**.**Input**);** //0 dB

**}**

TWIwriteByte**(**CENTER\_FREQ\_FLAT**|**LOW\_BOST**|**LOUDNESS\_SOFT\_STEP\_ON**);** //Loudness: Attenuation = 0dB, Flat, Low Boost, Soft Step=On

TWIwriteByte**(**0x00**);** //Master Volume: Attenuation = 0dB, Soft Step=On

TWIwriteByte**((**TDA7418**.**Treble**>**15**?** 46**-**TDA7418**.**Treble**:**TDA7418**.**Treble**)|**TREBLE\_CENTER\_FREQ\_12\_5KHZ**);** //Treble

TWIwriteByte**((**TDA7418**.**Middle**>**15**?** 46**-**TDA7418**.**Middle**:**TDA7418**.**Middle**)|**MID\_Q\_FACT\_1**);** //Middle

TWIwriteByte**((**TDA7418**.**Bass**>**15**?** 46**-**TDA7418**.**Bass**:**TDA7418**.**Bass**)|**BASS\_Q\_FACT\_1**);** //Bass

TWIwriteByte**(**BASS\_CENTER\_FREQ\_80HZ**|**MID\_CENTER\_FREQ\_1\_5KHZ**|**SMOOTHING\_FILTER\_ON**);**

TWIwriteByte**(**95**-**TDA7418**.**frontVolume**);** //Front Left Attenuation, Soft Step=On

TWIwriteByte**(**95**-**TDA7418**.**rearVolume**);** //Rear Left Attenuation, Soft Step=On

TWIwriteByte**(**95**-**TDA7418**.**rearVolume**);** //Rear Right Attenuation, Soft Step=On

TWIwriteByte**(**95**-**TDA7418**.**frontVolume**);** //Front Right Attenuation, Soft Step=On

TWIwriteByte**(**SPEAKER\_MUTED**);** //Subwoofer Attenuation = Mute, Soft Step=On

TWIwriteByte**(**SOFT\_MUTE\_OFF**|**SOFT\_MUTE\_TIME\_48MS**|**SOFT\_STEP\_TIME\_1\_28MS**|**AUTO\_ZERO\_ON**);**

TWIwriteByte**(**0x00**);** //Audio Processor Testing Mode=Off

TWIstop**();**

**}**

**Listing 3. Funkcja odpowiedzialna za wybór aktywnego wejścia selektora wejściowego układu TDA7418**

void TDA7418setInput**(**void**)**

**{**

TWIstart**();**

TWIwriteByte**(**TDA7418\_WR\_ADDR**);**

TWIwriteByte**(**REG\_SOURCE\_SELECTOR**|**AUTO\_ZERO\_REMAIN\_OFF**);**

//Input Gain zależny od wejścia, Single Ended Stereo

**switch(**TDA7418**.**Input**)**

**{**

**case** INPUT\_FM**:** TWIwriteByte**(**TDA7418**.**Input**|**6**<<**3**);** **break;** //6 dB

**case** INPUT\_BT**:** TWIwriteByte**(**TDA7418**.**Input**|**8**<<**3**);** **break;** //8 dB

**default:** TWIwriteByte**(**TDA7418**.**Input**);** //0 dB

**}**

TWIstop**();}**

**Listing 4. Funkcja odpowiedzialna za regulację tonów układu TDA7418**

void TDA7418setTone**(**uint8\_t toneReg**)**

**{**

TWIstart**();**

TWIwriteByte**(**TDA7418\_WR\_ADDR**);**

TWIwriteByte**(**toneReg**|**AUTO\_ZERO\_REMAIN\_ON**);**

**switch(**toneReg**)**

**{**

**case** REG\_TREBLE**:** TWIwriteByte**((**TDA7418**.**Treble**>**15**?** 46**-**TDA7418**.**Treble**:**TDA7418**.**Treble**)|**TREBLE\_CENTER\_FREQ\_12\_5KHZ**);** **break;**

**case** REG\_MIDDLE**:** TWIwriteByte**((**TDA7418**.**Middle**>**15**?** 46**-**TDA7418**.**Middle**:**TDA7418**.**Middle**)|**MID\_Q\_FACT\_1**);** **break;**

**case** REG\_BASS**:** TWIwriteByte**((**TDA7418**.**Bass**>**15**?** 46**-**TDA7418**.**Bass**:**TDA7418**.**Bass**)|**BASS\_Q\_FACT\_1**);** **break;**

**}**

TWIstop**();**

**}**

**Listing 5. Funkcja odpowiedzialna za regulację głośności układu TDA7418**

void TDA7418setVolume**(**void**)**

**{**

TWIstart**();**

TWIwriteByte**(**TDA7418\_WR\_ADDR**);**

TWIwriteByte**(**REG\_SPK\_ATT\_FL**|**AUTO\_INCR\_MODE\_ON**|**AUTO\_ZERO\_REMAIN\_ON**);**

TWIwriteByte**(**95**-**TDA7418**.**frontVolume**);** //Front Left Attenuation, Soft Step=On

TWIwriteByte**(**95**-**TDA7418**.**rearVolume**);** //Rear Left Attenuation, Soft Step=On

TWIwriteByte**(**95**-**TDA7418**.**rearVolume**);** //Rear Right Attenuation, Soft Step=On

TWIwriteByte**(**95**-**TDA7418**.**frontVolume**);** //Front Right Attenuation, Soft Step=On

TWIstop**();**

**}**

**Listing 6. Plik nagłówkowy do obsługi modułu Bluetooth w zakresie profilu A2DP**

//Prędkość interfejsu

#define USART\_BAUD 115200

#define \_\_UBRR (F\_CPU/16/USART\_BAUD-1)

//Maksymalna liczba bajtów w odpowiedzi

#define MAX\_BYTES 32

//Definicje statusów interfejsu BT

#define BT\_DISCONNECTED 0x00

#define BT\_CONNECTED 0x01

#define BT\_MUSIC\_PAUSE 0x02

#define BT\_MUSIC\_PLAY 0x03

//Znaczniki ASCII

#define ASCII\_LF 10

#define ASCII\_CR 13

//Zmienne modułu

extern volatile uint8\_t BTstatus**;** //Flaga statusu interfejsu BT

**Listing 7. Lista stałych związanych z obsługą modułu Bluetooth**

//Definicje obsługiwanych rozkazów sterujących i odpowiedzi modułu Flaircomm BTM-501

const char cmd\_ENTER\_PAIRING\_MODE**[]** PROGMEM **=** "AT#CA"**;**

const char cmd\_CANCEL\_PAIRING\_MODE**[]** PROGMEM **=** "AT#CB"**;**

const char cmd\_PLAY\_PAUSE\_MUSIC**[]** PROGMEM **=** "AT#MA"**;**

const char cmd\_STOP\_MUSIC**[]** PROGMEM **=** "AT#MC"**;**

const char cmd\_FORWARD\_MUSIC**[]** PROGMEM **=** "AT#MD"**;**

const char cmd\_BACKWARD\_MUSIC**[]** PROGMEM **=** "AT#ME"**;**

const char cmd\_CHANGE\_DEVICE\_NAME**[]** PROGMEM **=** "AT#MM"**;**

const char cmd\_POWER\_ON\_MODULE**[]** PROGMEM **=** "AT#MW"**;**

const char cmd\_GET\_PLAY\_STATUS**[]** PROGMEM **=** "AT#QD"**;**

const char cmd\_GET\_ELEMENT\_ATTRIBUTES**[]** PROGMEM **=** "AT#QE"**;**

const char cmd\_ACCEPT\_PAIRING\_REQUEST**[]** PROGMEM **=** "AT#QJ"**;**

const char cmd\_REJECT\_PAIRING\_REQUEST**[]** PROGMEM **=** "AT#QK"**;**

const char cmd\_ALWAYS\_DISCOVERABLE**[]** PROGMEM **=** "AT#QP"**;**

const char ind\_ENTER\_PAIRING\_MODE**[]** PROGMEM **=** "II"**;**

const char ind\_EXIT\_PAIRING\_MODE**[]** PROGMEM **=** "IJ"**;**

const char ind\_INIT\_COMPLETE**[]** PROGMEM **=** "IS"**;**

const char ind\_AV\_STOP**[]** PROGMEM **=** "MA"**;**

const char ind\_AV\_PLAY**[]** PROGMEM **=** "MB"**;**

const char ind\_ELEMENT\_ATTRIBUTES**[]** PROGMEM **=** "MH"**;**

const char ind\_PLAY\_STATUS**[]** PROGMEM **=** "MJ"**;**

const char ind\_REPORT\_DEVICE\_NAME**[]** PROGMEM **=** "MM"**;**

const char ind\_A2DP\_CONNECTED**[]** PROGMEM **=** "MK"**;**

const char ind\_A2DP\_DISCONNECTED**[]** PROGMEM **=** "MY"**;**

const char ind\_A2DP\_LINK\_LOSS**[]** PROGMEM **=** "PP"**;**

const char ind\_PAIRING\_REQUEST**[]** PROGMEM **=** "QA"**;**

const char ind\_PAIRING\_SUCCESSFUL**[]** PROGMEM **=** "QB"**;**

const char ind\_PAIRING\_FAILRUE**[]** PROGMEM **=** "QC"**;**

**Listing 8. Funkcje obsługi USART mikrokontrolera**

void USARTinit**(**void**)**

**{**

//Ustawienie prędkości 115200 bps

UBRR0H **=** **(**uint8\_t**)**\_\_UBRR**>>**8**;** UBRR0L **=** **(**uint8\_t**)**\_\_UBRR**;**

//Załączenie nadajnika i odbiornika oraz uruchomienie przerwania od RX

UCSR0B **|=** **(**1**<<**RXEN0**)|(**1**<<**TXEN0**)|(**1**<<**RXCIE0**);**

//Ustawienie formatu ramki: 8bitów danych, 1 bit stopu

UCSR0C **=** **(**3**<<**UCSZ00**);**

**}**

inline void USARTsendByte**(**uint8\_t Byte**)**

**{**

//Czekamy, aż bufor nadawczy będzie pusty

**while** **(!(**UCSR0A **&** **(**1**<<**UDRE0**)));**

UDR0 **=** Byte**;**

**}**

void USARTsendString\_P**(**const char **\***String**)**

**{**

register char Byte**;**

//Wysyłamy łańcuch z pamięci FLASH dopóki nie napotkamy 0

**while** **((**Byte **=** pgm\_read\_byte**(**String**++)))** USARTsendByte**(**Byte**);**

**}**

**Listing 9. Funkcja przeznaczona do wysłania rozkazu z parametrem do modułu Bluetooth**

void BTsendCommand**(**const char **\***Command**,** const char **\***Parameters**)**

**{**

USARTsendString\_P**(**Command**);**

USARTsendString\_P**(**Parameters**);**

USARTsendString\_P**(**PSTR**(**"\r\n"**));**

**}**

**Listing 10. Funkcja do konfiguracji modułu Bluetooth**

void BTconfig**(**void**)**

**{**

BTsendCommand**(**cmd\_POWER\_ON\_MODULE**,** **NULL);**

BTsendCommand**(**cmd\_ENTER\_PAIRING\_MODE**,** **NULL);**

//Zmieniamy domyślną nazwę BT

BTsendCommand**(**cmd\_CHANGE\_DEVICE\_NAME**,** PSTR**(**"eagleAudio SAR"**));**

**}**

**Listing 11. Funkcja obsługi przerwania od odebrania znaku USART**

/\* Funkcja obsługuje interfejs USART0, czyli odbiór danych wysyłanych

przez modem. Ramka: Odpowiedź<CR><LF> \*/

ISR**(**USART0\_RX\_vect**)**

**{**

static char String**[**MAX\_BYTES**];** //Tablica przechowująca odpowiedź modemu

static uint8\_t Idx**;** //Indeks bieżącego elementu

char recByte **=** UDR0**;** //Odczytanie odebranego bajta danych

**switch(**recByte**)**

**{**

**case** ASCII\_CR**:**

**break;**

**case** ASCII\_LF**:**

//Odebraliśmy kompletną odpowiedź modemu, więc ją analizujemy

String**[**Idx**]** **=** '\0'**;** //Na końcu C-stringa umieszczamy NULL

**if((**strcmp\_P**(**String**,** ind\_HFP\_DISCONNECTED**)==**0**)** **||** **(**strcmp\_P**(**String**,** ind\_A2DP\_DISCONNECTED**)==**0**))** BTstatus **=** BT\_DISCONNECTED**;**

**else** **if((**strcmp\_P**(**String**,** ind\_HFP\_CONNECTED**)==**0**)** **||** **(**strcmp\_P**(**String**,** ind\_A2DP\_CONNECTED**)==**0**))** BTstatus **=** BT\_CONNECTED**;**

**else** **if(**strcmp\_P**(**String**,** ind\_AV\_STOP**)** **==** 0**)** BTstatus **=** BT\_MUSIC\_PAUSE**;**

**else** **if(**strcmp\_P**(**String**,** ind\_AV\_PLAY**)** **==** 0**)** BTstatus **=** BT\_MUSIC\_PLAY**;**

Idx **=** 0**;**

**break;**

**default:**

String**[**Idx**++]** **=** recByte**;**

**}**

**}**