High Speed USB 2.0 Development Board

Instrukcja użytkownika.
wersja 0.1
Autor: Łukasz Krzak

Spis treści.

1. Opis układu.

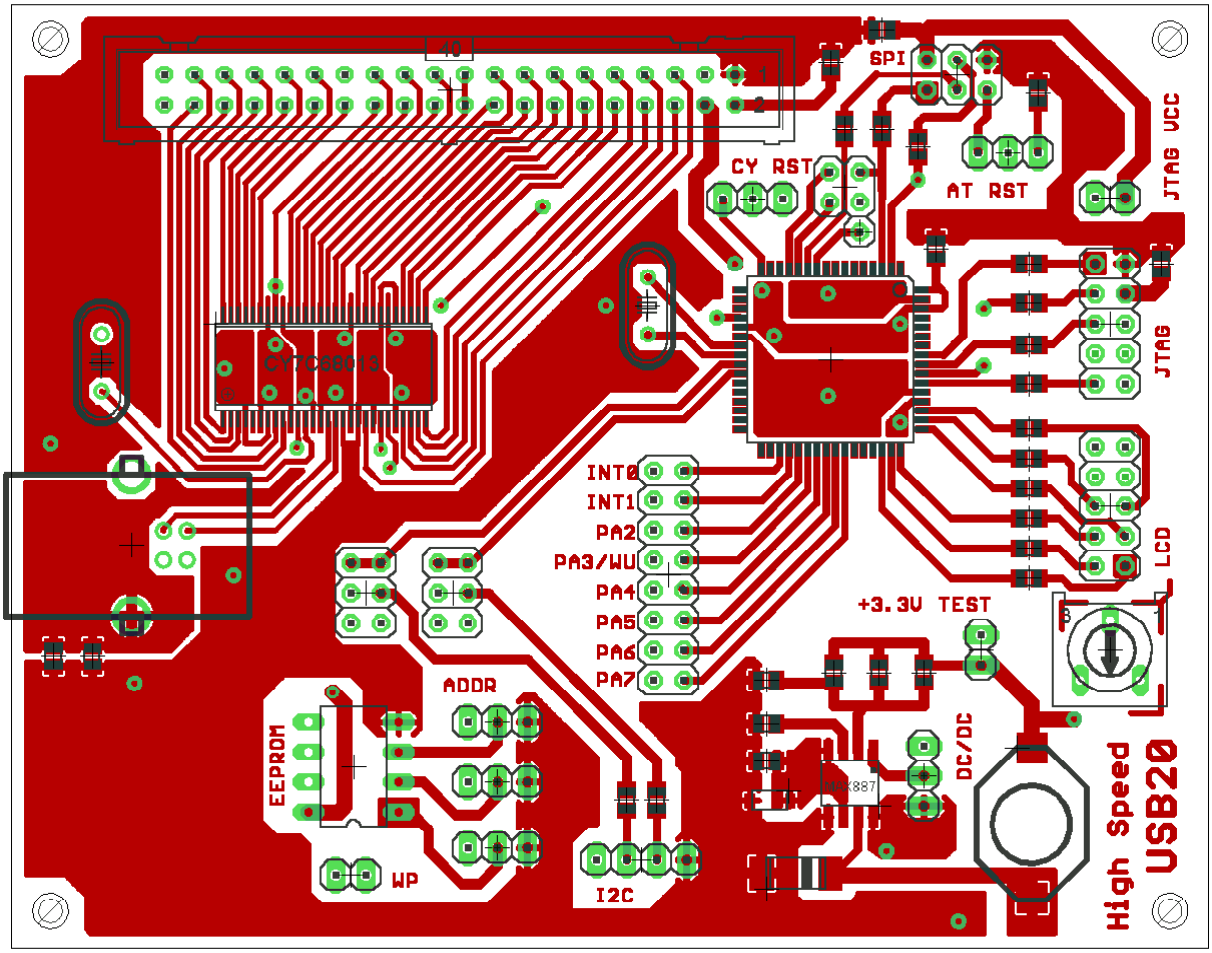
Moduł wykorzystuje interfejs USB 2.0 do komunikacji z hostem i przesyłania danych do/z zewnętrznych układów dołączanych do gniazda IDE. Za obsługę USB odpowiada kontroler z rodziny FX2: CY7c68013 firmy Cypress Semiconductor. Dodatkowy kontroler z rodziny AVR: ATMega128L firmy Atmel działa jako bootloader oraz monitoruje i steruje pracą kontrolera USB.

Moduł posiada 2 linie zasilające: +5V pobierane wprost z gniazda USB oraz +3.3V otrzymywane z przetwornicy DC/DC opartej o układ MAX887 firmy Maxim.

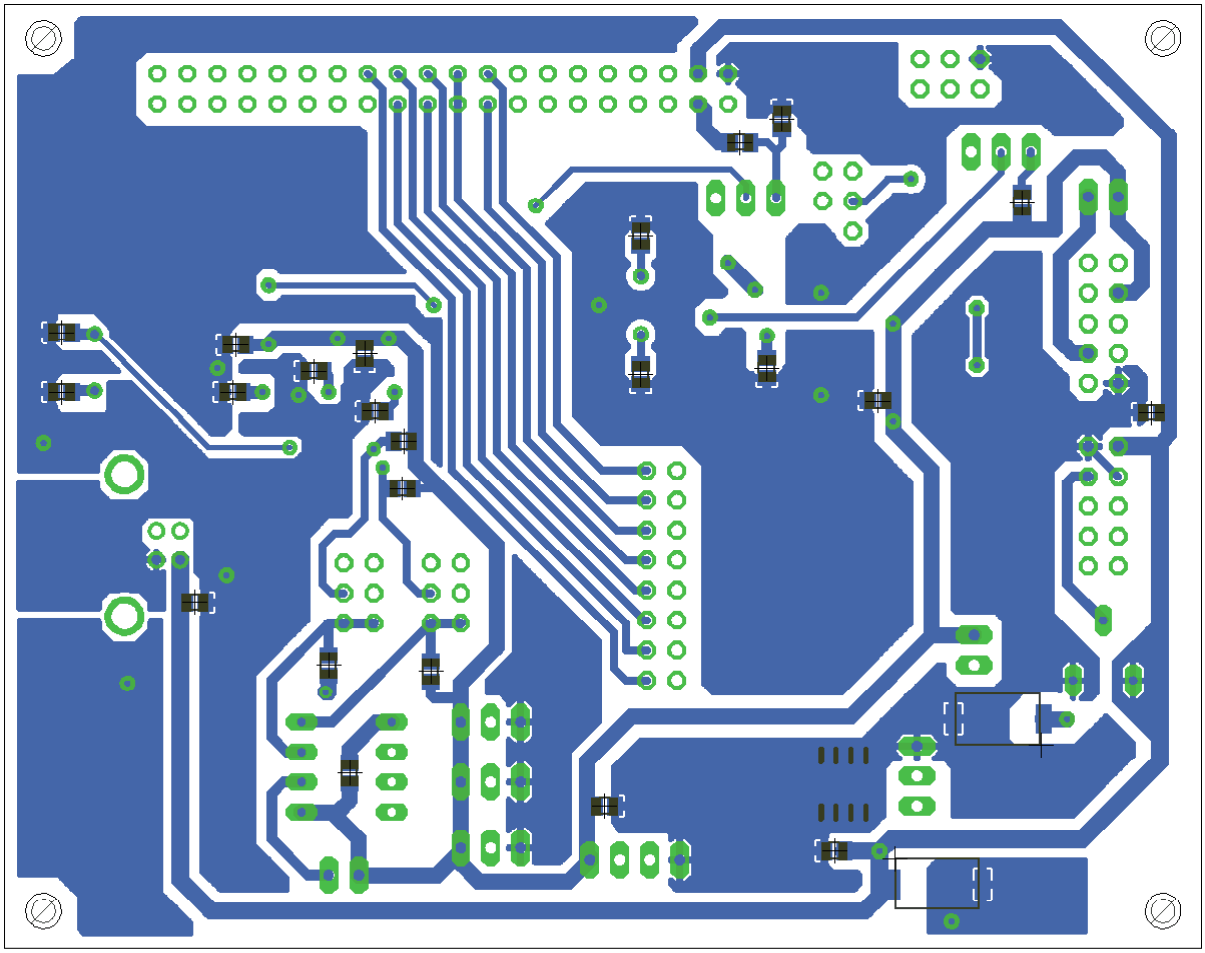
Oba kontrolery mogą być połączone magistralą I^2C wraz z opcjonalną pamięcią EEPROM. Kontrolery można również łączyć pin po pinie poprzez 8bitowy port równoległy.

1.1 Widok płytki.

Poniżej przedstawiono dwa widoki płytki drukowanej modułu, wierzch (rys. 1.1) oraz spód (rys 1.2).



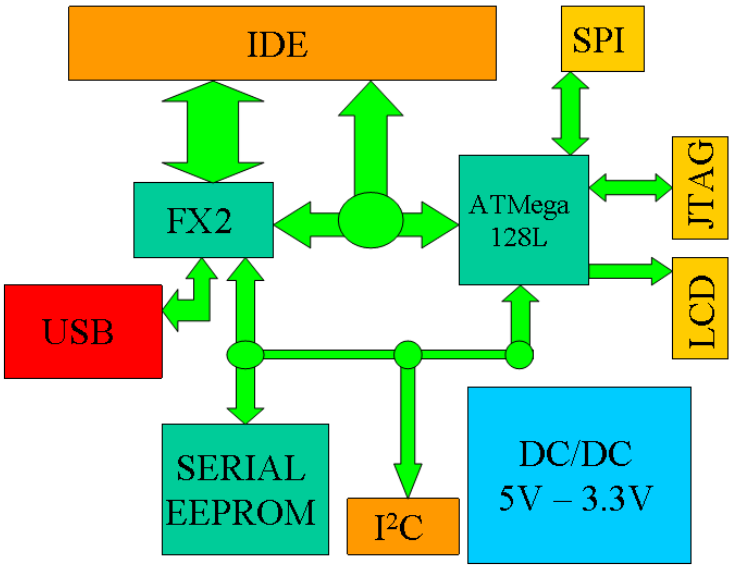
Rys. 1.1 – widok płytki drukowanej modułu (wierzch)



Rys. 1.2 – widok płytki drukowanej modułu (spód)

1.2. Diagram połączeń i przepływu informacji.

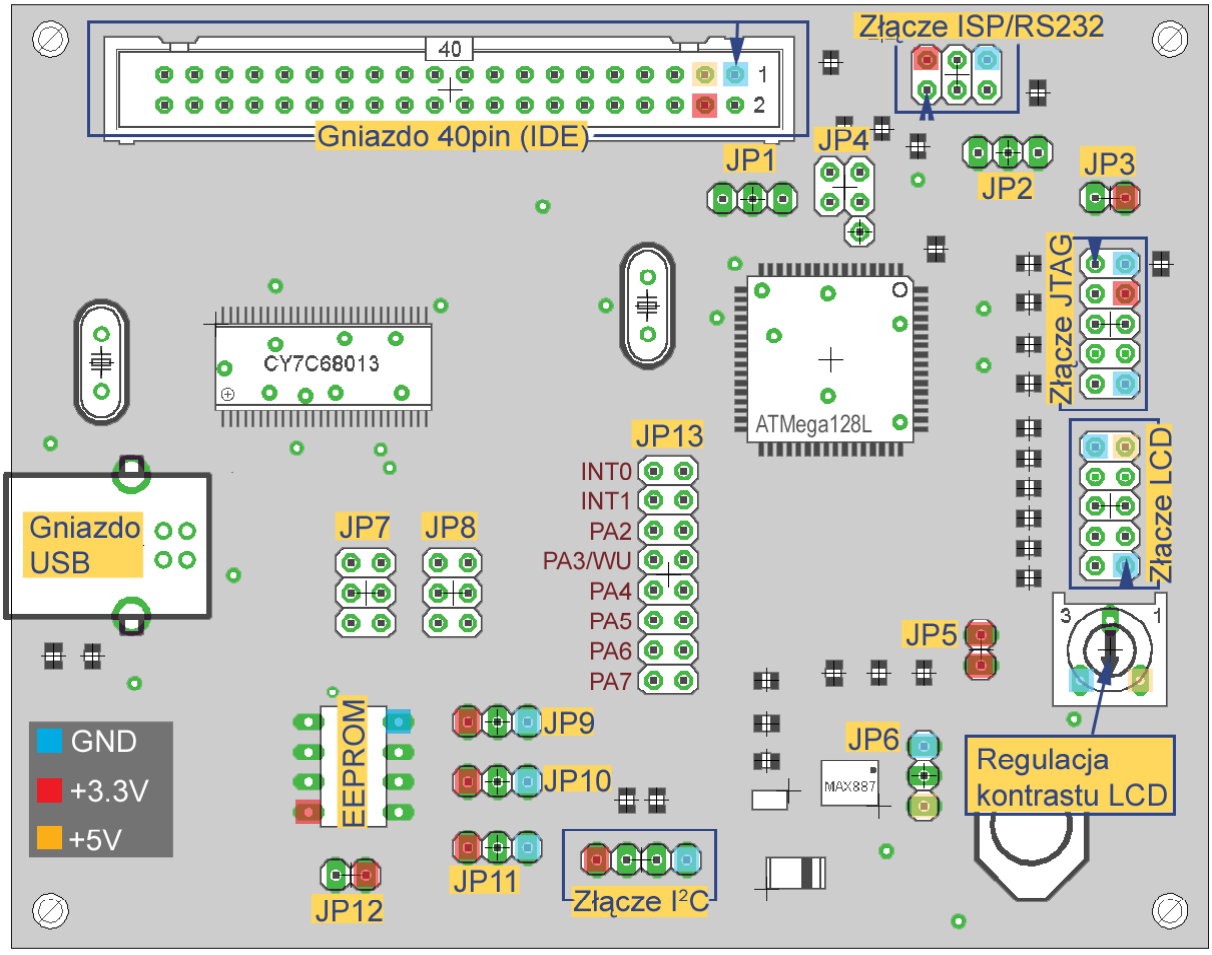
Poniżej przedstawiono schematycznie diagram połączeń przepływu informacji:



Rys. 1.3 – diagram połączeń i przepływu informacji.

2. Konfiguracja układu.

Moduł posiada kilka możliwości konfiguracyjnych ustalanych zworami. Poniżej znajduje się widok modułu z zaznaczonymi wyprowadzeniami, portami i miejscami na zwory. Trzema kolorami zaznaczono także charakterystyczne punkty wyprowadzeń, tj. masa (GND) oraz 2 napięcia zasilania (+3.3V i +5V). Strzałkami oznaczono pierwszy pin każdego portu



2.1 Opis wyprowadzeń.

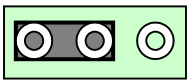
Złącze ISP/RS232 – port ISP do szeregowego programowania kontrolera ATMega128L oraz wyprowadzenie USART0 wykorzystywane do komunikacji poprzez interfejs RS232.

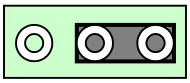
Złącze JTAG – standardowe 10pinowe złącze JTAG.
Złącze LCD – przystosowane do pracy w trybie 4bit i transmisji jednokierunkowej złącze wyświetlacza LCD.

Złącze I^2C – 4pinowe złącze zawierające oprócz sygnałów SDA i SCK również +3.3V i GND.

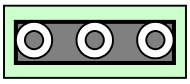
2.2 Ustawienia zworek.

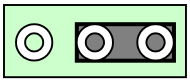
JP1 – wybór źródła resetu dla kontrolera USB CY7c68013.

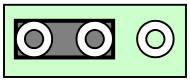
 - reset sterowany poprzez kontroler ATMega128.

 - reset standardową siecią RC.

JP2 – wybór źródła resetu dla kontrolera ATMega128.

 - reset sterowany poprzez programator ISP.

 - reset standardową siecią RC.

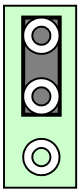
 - zewnętrzny reset ze złącza ISP/RS232

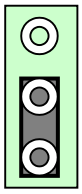
JP3 – wybór źródła zasilania dla JTAG. Powinno być rozwarte.

JP4 – wybór pinów wyprowadzonych na złącze ISP/RS232. Dla standardowej pracy wszystkie powinny być rozwarte. Odpowiednie ustawienie JP4 umożliwia m.in. opcjonalne wyprowadzenie pinów używanych w transmisji synchronicznej.

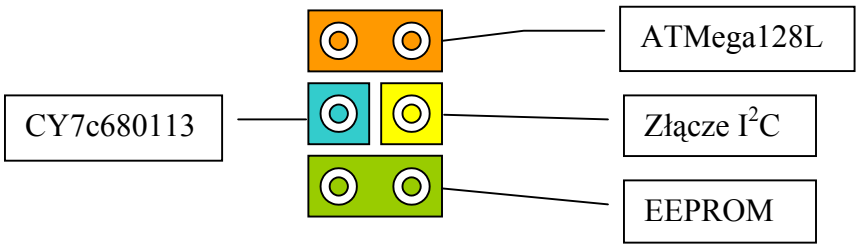
JP5 – załączenie zasilania +3.3V z przetwornicy do reszty układu. Aby układ pracował prawidłowo w tym miejscu musi być zwora. Złącze można wykorzystać do pomiaru prądu pobieranego z przetwornicy DC/DC.

JP6 – włączenie/ wyłączenie przetwornicy DC/DC.

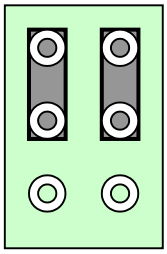
 - przetwornica wyłączona
układy niezasilane

 - przetwornica włączona

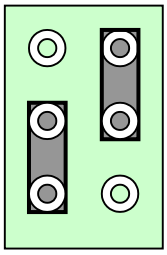
JP7 i **JP8** – służą do ustawienia połączeń na magistrali I^2C. Muszą być ustawione jednakowo!



Przykładowe ustawienia zworek JP7 i JP8:



- Na wspólnej magistrali ATMega128L, CY7c68013 oraz złącze I^2C.



- CY7c68013 połączony z pamięcią EEPROM.
  ATMega128L połączona ze złączem I^2C.

JP9 , JP10 , JP11 – wybór adresu na magistrali I^2C dla pamięci EEPROM. Zobacz dokumentację EEPROMu.

JP12 – ustawienie „write protect" dla pamięci EEPROM. Zobacz dokumentację EEPROMu.

JP13 – połączenie portu A kontrolera CY7c68013 i portu C kontrolera ATMega128L. Zaleca się połączenie tych portów poprzez rezystory (minimum 1kΩ).
UWAGA!! Nieodpowiednie skonfigurowanie portów obu kontrolerów może spowodować ich uszkodzenie, jeśli zostaną połączone.

3. Programowanie kontrolera ATMega128L.

Obecny w układzie kontroler ATMega128L może być programowany poprzez łącze szeregowe ISP lub interfejs JTAG, przy pomocy jednego z wielu dostępnych programów, przeznaczonych do współpracy z kontrolerami AVR. Kod źródłowy programu został napisany w darmowym edytorze SynTextEditor zaś skompilowany również darmową wersją kompilatora WinAVR

3.1 System uC/OS-II i struktura programu.

Program sterujący pracą kontrolera AVR wykorzystuje system czasu rzeczywistego uC/OS-II (wersja 2.6) firmy Micrium. Ten prosty system operacyjny zapewnia kontekstowe przełączanie zadań uruchamianych na procesorze, oraz oferuje wiele mechanizmów wspomagających projektowanie wydajnych aplikacji czasu rzeczywistego.

Program wykorzystuje przerwanie od timer'a nr 0 jako przerwanie zegara systemowego o częstotliwości 50Hz. Uruchomione zadania obsługują wyświetlacz LCD, komunikację poprzez RS232 (USART0) zgodnie z interfejsem VT100 (VT102), oraz monitorowanie magistrali I^2C. Głównym wątkiem odpowiadającym za komunikację jest funkcja **IOTask** z pliku **„tasks.c"**. Dodatkowo w tle działa systemowy wątek statystyczny, który co sekundę liczy aktualne zużycie procesora i zgodnie z wybraną opcją wyświetla wynik na wyświetlaczu LCD i/lub terminalu RS232.

Struktura plików projektu i ich opis przedstawiono poniżej:

main.c	Główny plik projektu (target)
makefile	Skrypt kompilacji projektu
includes.h	Plik grupujący wszystkie wykorzystywane nagłówki. Dołączany w każdym module projektu.
os_cfg.h	Plik konfiguracyjny systemu uC/OS-II
avr_isr.h	Zbiór makr użytecznych przy obsłudze przerwań w procesorze AVR
os_cpu.h	Pliki portu systemu uC/OS-II dla procesorów AVR
os_cpu_a.asm	
os_cpu_c.c	
OS_CORE.C (.H)	Pliki systemu uC/OS-II
OS_FLAG.C (.H)	
OS_MBOX.C (.H)	
OS_MEM.C (.H)	
OS_MUTEX.C (.H)	
OS_Q.C (.H)	
OS_SEM.C (.H)	
OS_TASK.C (.H)	
OS_TIME.C (.H)	
uCOS_II.C (.H)	
LCD.c (.h)	Moduł obsługi wyświetlacza LCD
USART.c (.h)	Moduł obsługi USARTu
VT100.c (.h)	Moduł obsługi interfejsu VT100 (VT102)
I2C.c (.h)	Moduł obsługi magistrali I^2C
tasks.c (.h)	Plik zawierający procedury zadań uruchamianych w systemie
Buffers.c (.h)	Moduł obsługi buforów wykorzystywanych w obsłudze RS232 i I^2C

Poniżej przedstawiono opis modułów obsługujących urządzenia we/wy i zewnętrzne interfejsy:

3.2. Moduł wyświetlacza LCD.

Moduł zawiera funkcje odpowiedzialne za inicjalizację i wyświetlanie znaków na dołączonym wyświetlaczu LCD. Aktualnie obsługują wyświetlacz 2x16 znaków w trybie transmisji jednokierunkowej 4bit. Wszystkie sygnały powinny znajdować się w obrębie jednego portu.

Pliki :

LCD.c
LCD.h

Funkcje i procedury:

```
void LCD_WriteByte(int8_t LCDByte,
                   int8_t IsData
                   );
void LCD_WriteText(char text[]);
```

```
void LCD_Goto(int8_t column,
              int8_t line
             );
void LCD_ClrLine(int8_t line);
void LCD_Init(void);
void LCD_BarGraphInit(void);
void LCD_BarGraph(int8_t line,
                  int8_t value
                 );
```

Opis funkcji:

LCD_Init – inicjalizuje wyświetlacz 16x2, w modzie czterobitowym. Konfiguruje kierunek odpowiednich pinów zdefiniowanych w sekcji #define w pliku LCD.h. Musi zostać wywołana przed korzystaniem z innych funkcji modułu.

LCD_ClrLine – czyści zadaną linię wyświetlacza (numeracja linii od zera).

LCD_Goto – przesuwa kursor do zadanej pozycji na wyświetlaczu (numeracja linii i kolumn od zera).

LCD_WriteByte – wysyła bajt na wyświetlacz w dwóch cyklach 4-bitowych (najpierw starsza połowa). Parametr IsData wskazuje typ wysyłanej informacji (0 – instrukcje, 1 - dane).

LCD_WriteText – wysyła zakończony zerem łańcuch znaków na wyświetlacz.

LCD_BarGraphInit – inicjalizuje wyświetlanie paska stanu. Definiuje potrzebne znaki w pamięci CGRam wyświetlacza. Musi zostać wywołana przed funkcją LCD_BarGraph.

LCD_BarGraph – wyświetla pasek stanu w zadanej linii wyświetlacza (numeracja od zera).

Korzystanie z modułu LCD:

1. Dołącz plik LCD.h w sekcji `#include`

2. W sekcji `#define` przypisz odpowiednie piny do odpowiednich sygnałów sterujących wyświetlaczem (E, RS, DB4..7), zdefiniuj port wyświetlacza (LCD_PORT) oraz jego rejestr kierunku (LCD_DDR).

3. Zdefiniuj czas trwania pętli opóźniającej (czekającej na zakończeni instrukcji wyświetlacza) w zależności od zegara taktującego kontroler:
dla fosc = 8MHz : LOOP_TIME = 0.5
dla fosc = 4MHz : LOOP_TIME = 1
dla fosc = 2MHz : LOOP_TIME = 2 ... itd

4. Wywołaj funkcję LCD_Init przed korzystaniem z innych funkcji modułu.

3.3. Moduł obsługi USART.

Moduł odpowiada za komunikację poprzez uniwersalny szeregowy nadajnik/odbiornik – USART (***U**niversal **S**ynchronous/**A**synchronous **R**eceiver/**T**ransmitter*).

Pliki:

USART.c
USART.h

Funkcje i procedury:

```
void USART_Init(int8_t  USARTNo,
                            int8_t  USARTMode,
                            int16_t BaudRatePresc,
                            int8_t  CharacterSize,
                            int8_t  PSOptions,
                            int8_t  IEOptions,
                            int8_t  OtherOptions
                            );
int8_t USART_GetChar(int8_t  USARTNo);
void USART_PutChar(int8_t USARTNo,
                   char Character
                   );
void USART_PutStr(int8_t USARTNo,
                  const char String[],
                  int8_t Location
                  );
int8_t USART_DataReceived(int8_t USARTNo);
void SystemLog(const char *String);
```

Opis funkcji:

USART_Init – inicjalizuje zadany USART w zadanym trybie. Funkcja musi być wywołana przed korzystaniem z innych funkcji tego modułu.
Parametry:

int8_t USARTNo	[i] - numer USARTu
int8_t USARTMode	[i] - wybor trybu Async/Sync
int16_t BaudRatePresc	[i] - preskaler zegara ustalajacy predkosc transmisji
int8_t CharSize	[i] - ilosc bitow na znak (5-9)
int8_t PSOptions	[i] - opcje parity i stop bit
int8_t IEOptions	[i] - opcje interrupt enable
int8_t OtherOptions	[i] - dodatkowe opcje

USART_DataReceived – funkcja zwraca 1 jeżeli w buforze zadanego USART-u znajduje się nieodebrana dana oraz 0 w przeciwnym przypadku.

USART_GetChar – zwraca daną odczytaną z bufora zadanego USART-u. Jeżeli bufor jest pusty funkcja czeka na jego zapełnienie

USART_PutChar – wysyła znak poprzez zadany USART.

USART_PutStr – wysyła przez zadany USART ciąg znaków zakończony zerem. Parametr Location określa w którym obszarze pamięci znajduje się wysyłany ciąg (MEM_RAM, MEM_EEPROM, MEM_FLASH)

SystemLog – wysyła zadany log systemowy znajdujący się w pamięci Flash na USART0

Korzystanie z modułu USART:

1. Dołącz plik USART.h w sekcji `#include`

2. W sekcji `#define` zdefiniuj czy moduł ma być częścią systemu uC/OS-II (UCOS_PROJECT) – spowoduje to uruchomienie odpowiednich opcji kompilacji

3. Wywołaj funkcję USART_Init przed korzystaniem z innych funkcji modułu.