1-Wire Emulator & Skaner (1)

Emulator układów 1-Wire będzie bardzo przydatny np. podczas uruchamiania elektronicznych termometrów, ponieważ umożliwi zastąpienie czujnika i ustawienie wymaganej temperatury w kilka sekund. Funkcja skanera będzie przydatna przy pisaniu oprogramowania hosta ponieważ wyniki są prezentowane w czytelniejszy sposób niż na rejestratorach logicznych (np. SALEAE) i umożliwiają łatwe tworzenie dokumentacji. Skaner może także wyzwalać oscyloskop różnymi zdarzeniami pojawiającymi się na magistrali.

Urządzenie umożliwia emulowanie układów z interfejsem 1-Wire, także tych trudno dostępnych i kosztownych, a nawet takich, które nie istnieją. Opcje synchronizacji mogą przydać się przy analizowaniu przebiegów przez proste oscyloskopy lub rejestratory logiczne, ponieważ synchronizacja może być wywołana sygnałem *reset*, *presence* lub wybranym rozkazem, czy też zgodnością adresu albo danej. W przypadku bardziej zaawansowanych oscyloskopów funkcje takie również mają istotne znaczenie, ponieważ przyrządy z analizą protokołu 1-Wire są rzadkością.

Emulując popularny czujnik temperatury typu DS18B20 możemy łatwo ustawić temperaturę 100°C, a za chwilę –30°C. Możemy również ustawić wartości spoza zakresu układu DS18B20, np. 300°C. Przy uruchamianiu oprogramowania z układami *only add* (np. OTP EPROM) zaoszczędzimy na układach, które mogą być kosztowne.

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-----

Podstawowe parametry:

- funkcja emulatora i skanera 1-Wire,
- obsługa prędkości standardowej i overdrive,
- akceptacja napięcia magistrali do 15 V,
- wbudowany enkoder i wyświetlacz LCD 20×4,
- komunikacja przez USB z programem terminala VT100.

Projekty pokrewne na www.media.avt.pl:

AVT-5567	1-Wire za pomocą UART (EP 1/2019)
AVT-5649	HUB 1-Wire z izolacją galwaniczną (EP 9/2018)
AVT-1949	Emulator DS18B20 (EP 4/2017)
AVT-1948	Interfejs termopary K z 1-Wire (EP 3/2017)
AVT-1787	Konwerter USB/1-Wire (EP 8/2013)

* Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!

Podstawową wersją zestawu jest wersja [B] nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji [B] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.

Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:

- wersja [C] – zmontowany, uruchomiony i przetestowany zestaw [B] (elementy wlutowane w płytkę PCB)
- wersja [A] – płytka drukowana bez elementów i dokumentacji

Kity w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:

- wersja [A+] – płytka drukowana [A] + zaprogramowany układ [UK] i dokumentacja
- wersja [UK] – zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! http://sklep.avt.pl. W przypadku braku dostępności na http://sklep.avt.pl, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.

Funkcje emulatora

- ustawienie dowolnego identyfikatora układu,
- automatyczne obliczanie CRC identyfikatora,
- emulacja prędkości standardowej i overdrive,
- akceptacja napięć na magistrali do 15 V (emulacja pamięci EPROM),
- wbudowana emulacja 6 układów:
  a. numerów seryjnych DS1990, DS2400, DS2401, DS2411 wszystkie komendy (*ReadRom*, *SkipRom*, *MatchRom*),
  b. termometru DS18B20 łącznie z EEPROM i alarmami, *Search Alarm*,
  c. EEPROM DS2431 łącznie z ustawianiem zabezpieczenia przed zapisem (do resetu mikrokontrolera),
- zgłaszanie przerwań/alarmu przez slave,
- emulowane EPROM, EEPROM „resetują" się do wartości fabrycznych po restarcie emulatora, zapewniając im „wieczny żywot".

Funkcje skanera

- pułapki na rozkazie i/lub rodzinie układów,
- synchronizacja oscyloskopu: *reset*, *presence*, rozkaz, rodzina układów,
- wyświetlanie danych na terminalu VT100 w kolorze z prędkością 500 kb/s (FT220),
- wyróżnianie kolorem *resetu*, *presence*, komendy, kodu rodziny, danych z mastera i slave, poprawnej lub złej crc,
- opis tekstowy komendy i kodu rodziny,
- akceptacja napięć na magistrali do 15 V (emulacja pamięci EPROM).

Czym jest tryb overdrive?

Standardowa prędkość transmisji wynosi: 15,4 kbps (max), a overdrive: 125 kbps (max). Zwiększona prędkość osiągnięto przez skrócenie czasu sygnału reset oraz nadawania i odbierania bitu. Dodatkowo przepustowość wzrosła dzięki rozkazowi RESUME (jeden bajt), który zastępuje MATCHROM (dziewięć bajtów) w sytuacji gdy slave był już wcześniej zaadresowany. Przejście do trybu overdrive następuje po wydaniu rozkazu: OverdriveSKIPROM lub OverdriveMATCHROM. Wyjście z tego trybu następuje po wykonaniu resetu w standardowej prędkości (reset w prędkości overdrive nie zmienia ustawienia).

Wykrywanie sygnału RESET i generowanie sygnału PRESENCE

Na początek, kilka ważnych zagadnień dotyczących realizacji kluczowych funkcji programu. Najprostsza procedura rozpoznawania sygnału *reset* jest realizowana w obsłudze przerwania od opadającego zbocza sygnału i przebiega w następujący sposób:

- czekaj, aż magistrala powróci do stanu wysokiego,
- jeśli czas >460 µs generuj *presence*,
- jeśli >60 µs to odebrano zero,
- w przeciwnym wypadku odebrano jeden.

Podstawową wadą procedury jest to, że przerwanie „wisi" przez czas generowania zera przez host. W tym czasie program główny jest zawieszony. Ponadto, gdy magistrala zostanie zwarta program zostanie zatrzymany, chyba że zadziała watchdog.

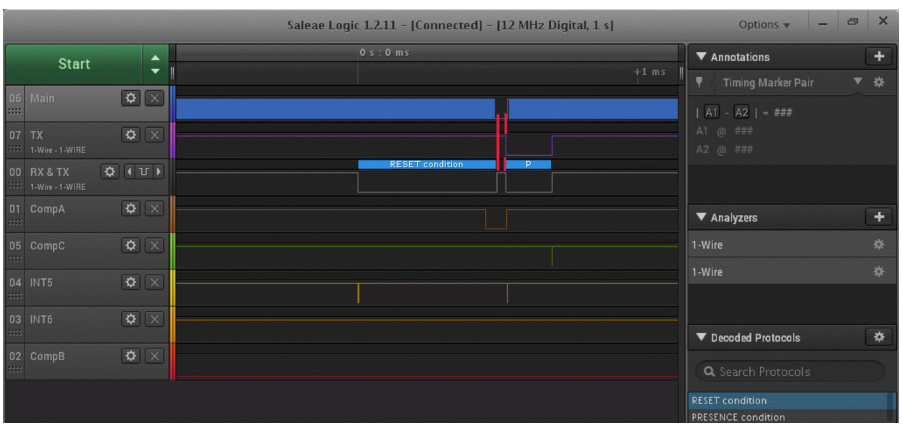
Host, po sygnale *reset*, oczekuje na sygnał *presence*. Pomiędzy *reset* a *presence* magistrala powinna przejść w stan wysoki na 30 µs, a przy opisanym wcześniej algorytmie tak nie jest i inne układy slave mogą źle zinterpretować sygnał *reset* lub wygenerować *presence* zbyt późno.

Listing 1. Procedura pomiaru czasu trwania sygnału reset

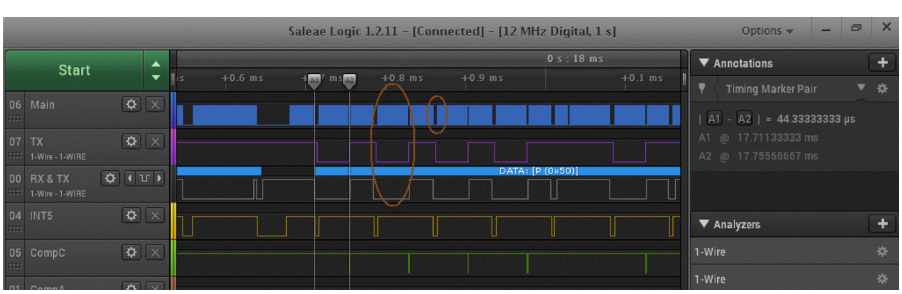
```
INTERRUPT( TIMER3_COMPA_vect )
{
long static srednia=0;
word tim;

    tim = TCNT_OW;
    // Jeśli timer przepełniono a magistrala jest w stanie niskim
    if ( !WIRE1_RD ) {
        // to mamy reset (przepełnienie jest na 440us a nie 460)
        // Czekamy, aż master zwolni magistralę 1-wire
        while( !WIRE1_RD ) {
            // Jeśli zbyt długo to wyjdź
            if ( tim > microSecondsOW( TIM_1W_RST*1.5 ) ) return;
            INIT_ZMIENNE;           // Wyłączenie overdrive
            WIRE1_RESET;            // Początek sygnału ACK
            START_TIMER_OW;         // Uruchamiamy odliczanie
            TIFR_OW |= _BV(OCFC_OW);// Kasuj flagę przerwania
            // Odblokowanie przerwań od porównania z Comp C (240us)
            TIMSK_OW |= _BV(OCIEC_OW);

            // Wyliczanie średniej czasu sygnału reset
            // Jeśli nie skończono jeszcze pomiarów
            if ( RstTigMeasurement ) {
                if ( tim > microSecondsOW( 400 ) &&
                tim < microSecondsOW( 500 ) ) {
                    srednia += tim;
                }
                if ( !--RstTigMeasurement ) {
                    srednia /= DEFRSTMEASURMENT;
                    // Do timera nie można wpisać czasu odczytanego z CNTx
                    srednia -= microSecondsOW( 5 );
                    // ponieważ od chwili wygenerowania IRQ do odczytania
                    // wartości timera minął jakiś czas i CNT uległo zmianie
                    // Czas ten zależny jest od liczny rejestrów odłożonych
                    // na stosie
                    // oraz mógłbyć zakłócony przez inne przerwania
                    TimRstTrigger = srednia;
                    OCRA_OW = TimRstTrigger;        // Nowy czas przerwania
                }
            }
        }
    // Timer przepełniony a magistrala w wysokim, zatrzymaj więc go
    else
        {
            STOP_TIMER_OW;
        }
}
```



Rysunek 1. Przebiegi obrazujące działanie programu w czasie generowania sygnałów



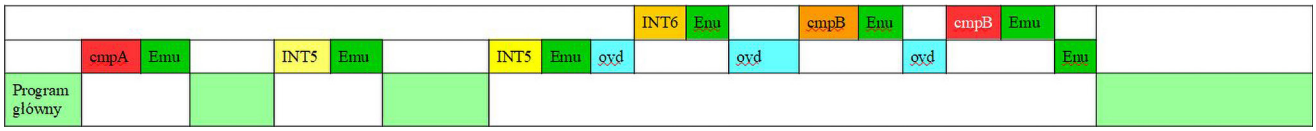
Rysunek 2. Przebiegi obrazujące działanie programu podczas generowania bitu wartości zero

Problem zawieszenia programu głównego na czas pomiaru sygnałów *reset* i *presense* mogą rozwiązać timery, ale jaki czas ustawić jako *reset*? Jeśli ustawimy 480 µs, a host będzie generował krótszy sygnał, to nigdy nie wykryjemy sygnału *reset*. Jeśli ustawimy zbyt krótki czas to procedura przerwania będzie niepotrzebnie blokowała program główny (w przerwaniu musimy czekać na narastające zbocze sygnału). Rozwiązania są dwa:

1. Czas sygnału *reset* będzie analizowany w przerwaniu od narastającego zbocza sygnału. Aby nie kolidowało to z obsługą przerwania od obsługi bitu można te przerwania wyłączyć do czasu wygenerowania *presence*.
2. Czas trwania sygnału *reset* będzie mierzony. Przykład programowy pokazano na **listingu 1**.

Problem „zawieszania" programu głównego na czas rozpoznawania sygnału *reset*, generowania *presence* i *zera* można rozwiązać timerami. Na **rysunku 1** widać, że program główny (niebieski przebieg) pracuje, w czasie generowania sygnałów. Widać wyraźną przerwę oczekiwania na koniec resetu, następnie opóźnienie 30 µs oraz start timera generującego sygnał *presence*. Podczas generowania bitu o wartości zero nie trzeba czekać na koniec sygnału, więc program główny jest zatrzymywany na krócej (**rysunek 2**). Widoczne są tylko krótkie przerwy w działaniu programu głównego (4,5 µs) po opadającym zboczu sygnału na magistrali.

W trybie overdrive odczyt bitu powinien nastąpić 1 µs po sygnale strobującym odczyt, a okno bitu trwa 7,6 µs. Biorąc pod uwagę, że przy 16 MHz, wejście w IRQ to około 0,6 µs (dla układów z pamięcią flash >128 kB trochę dłużej, ponieważ procesor odkłada 3 a nie 2 bajty adresu powrotu) a operacje na stosie takie jak *push Rx, eor r1,r1* oraz *push* rejestru stanu i RAMPZ dla procesorów z pamięcią ponad 64 kB dają łącznie 2,3 µs, nie ma szansy na czas wystawić bitu zero. Jakim

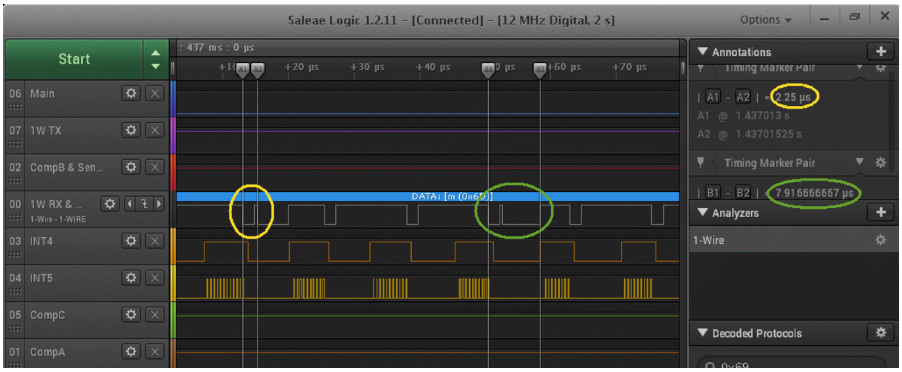


Rysunek 3. Schemat występowania przerwań

więc sposobem rozwiązano ten problem? Przejście w overdrive powoduje:

1. Zezwolenie na IRQ tylko od INT4 (overdrive) i Timera3 (wykrywającego *reset* i generującego *presence*). Pozostałe przerwania (Timer0, USART, impulsator) są zablokowane.
2. „Wieczna pętla" z której wyjście nastąpi po zmianie trybu z overdrive na standardową prędkość (*reset* w standardowej prędkości).
3. Po wyjściu z „wiecznej pętli" następuje wyłączenie przerwań od INT4 (overdrive) oraz przywrócenie wcześniej zawieszonych (INT5, Timer0, USART, impulsator).

Dodatkowo przerwanie INT dla overdrive zadeklarowano z flagą NAKED. Wcześniejsze wyzerowanie rejestru R1 (po wejściu w IRQ od prędkości standardowej) daje gwarancję, że ma on wartość zero i nie trzeba tego robić. Dzięki temu wejście w przerwanie trwa niecałe 0,6 µs.



Rysunek 4. Przebiegi obrazujące działanie programu przy taktowaniu 16 MHz i bez przerzutnika D

Niestety zanim program przetestuje jaki bit wystawić na magistralę mija około 2 µs przy taktowaniu 22 MHz, przy 14,7 MHz jest to 3,5 µs więc za późno aby wystawić bit. Jak rozwiązano ten problem? Wiedząc jakiej wartości będzie wysyłany bit podczas kolejnego slotu odczytu ustawiam odpowiednio flagę FL_SEND_ZERO. W kolejnym przerwaniu pierwszą czynnością jest sprawdzenie tej flagi i w razie potrzeby ustawienie magistrali w stan niski. Dzieje się to 1,4 µs po opadającym zboczu sygnału strobującego odczyt wiec jest jeszcze zapas 0,6 µs – prawie dwa rozkazy. Obsługa przerwania trwa niecałe 3 µs przy 22 MHz (6 µs przy 14,7) więc jest zapas czasu na wyjście z przerwania i oczekiwanie na następne.

Wykaz elementów:

Rezystory: (SMD1206)
R1, R2: 27 Ω
R5: 1 MΩ
R20…R24, R28, R29, R44…R47: 1 kΩ
R27: 4,7 kΩ
R38: 470 Ω
R41: 2,2 kΩ
P9: 10 kΩ PT10LV potencjometr montażowy

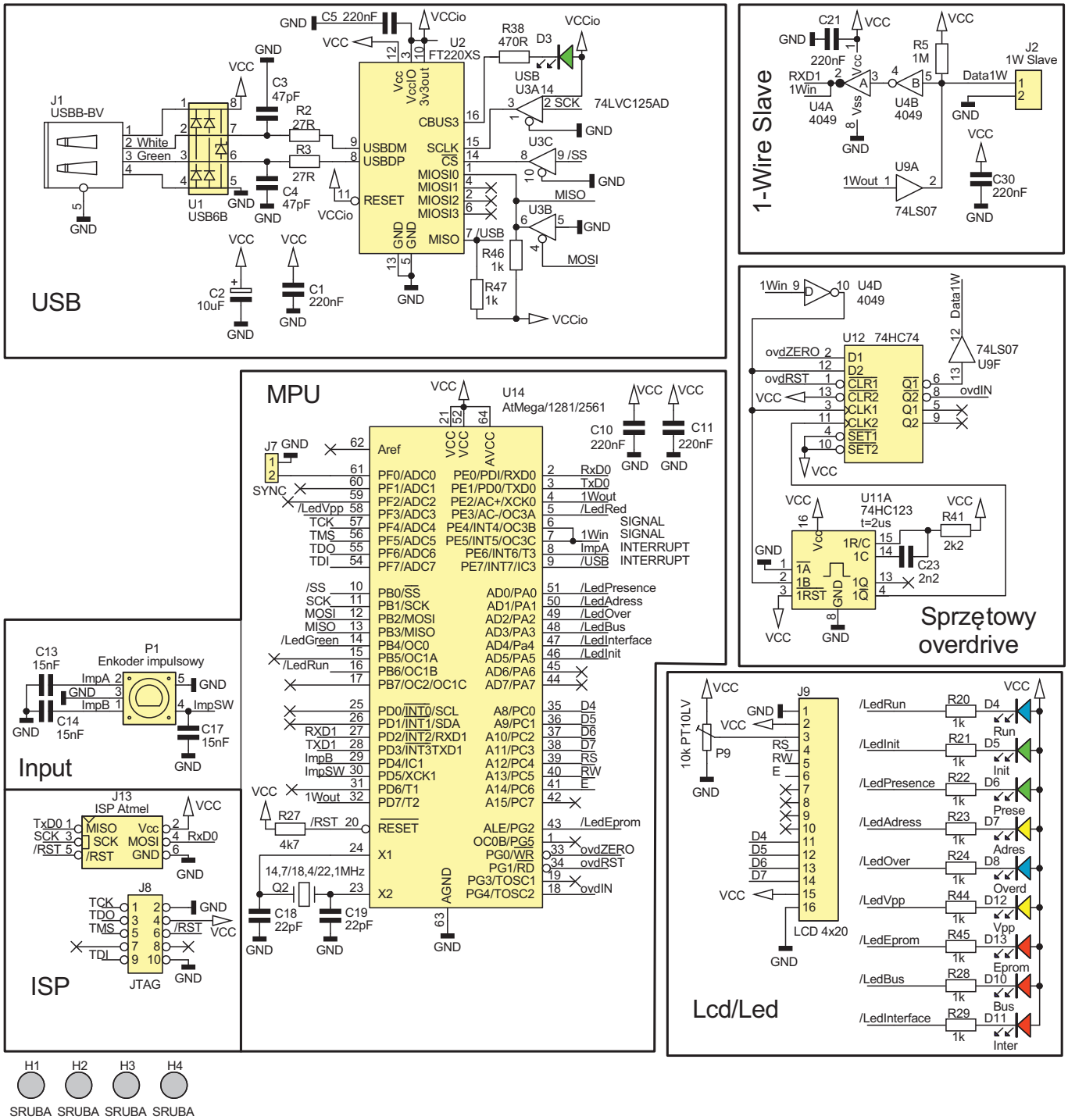
Kondensatory: (SMD1206 jeśli nie napisano inaczej)
C1, C5, C10, C11, C21, C30: 220 nF
C2: 10 µF/16 V elektrolityczny
C3, C4: 47 pF
C13, C14, C17: 15 nF
C18, C19: 22 pF
C23: 2,2 nF

Półprzewodniki:
D3 (USB): dioda LED zielona SMD1206
D4 (Run): dioda LED niebieska SMD1206
D5 (Init): dioda LED zielona SMD1206
D6 (Prese): dioda LED zielona SMD1206
D7 (Adres): dioda LED żółta SMD1206
D8 (Overd): dioda LED niebieska SMD1206
D10 (Bus): dioda LED czerwona SMD1206
D11 (Inter): dioda LED czerwona SMD1206
D12 (Vpp): dioda LED żółta SMD1206
D13 (Eprom): dioda LED czerwona SMD1206
U1: USB6B zabezpieczenie ESD
U2: FT220XS SSOP-16
U3: 74LVC125AD SO-14
U4: 4049 SO-16
U9: 74LS07 SO-14
U11: 74HC123 SO-16
U12: 74HC74 SO-14
U14: ATmega/1281/2561 PQFP64

Inne:
F1: bezpiecznik polimerowy SN010-60
ZW1, ZW2, ZW3: zwora 0 Ω SMD1206
Q1: rezonator kwarcowy SMD 18,4321 MHz
P1: enkoder impulsowy, impulsator
J1: gniazado katowe USB
J2: złącze typu ARK2
J9: LCD 4×20 + złącze goldpin
J7: złącze typu HU02
J8: złącze JTAG typu IDC10
J13: złącze ISP Atmel typu IDC6

Tabela 1. Opis znaczenia diod led

Oznaczenie LED		Funkcja	Uwagi
Nazwa	Symbol		
Power	D2	Sygnalizuje napięcie zasilania	Brak świecenia może oznaczać zwarcie zasilania magistrali 1-Wire
USB	D3	Praca interfejsu USB	
Run	D4	Pulsuje w trakcie pracy	Przerywa pulsowanie na czas zawieszenia IRQ
Init	D5	Zaświeca po poprawnym teście interfejsu i magistrali	Przygasa po wykryciu IRQ zgłaszanego przez emulator lub innego svale
Adr (Adress)	D6	Zaświeca po odebraniu komendy SkipRom, MatchRom ze zgodnym adresem, SkipRomOvd, MatchRomOvd, resume. Gaśnie po odebraniu resetu	
Prese (Presence)	D7	Zaświeca w czasie generowania tego sygnału	W trybie skanera sygnalizuje wykrycie sygnału presence
Overd (Overdrive)	D8	Zaświeca po przejściu w tryb overdrive, gaśnie po wykryciu resetu o standardowym czasie trwania (reset z prędkością overdrive nie zmienia jej stanu)	
Bus (Bus error)	D10	Zaświeca gdy stan niski na magistrali utrzymuje się zbyt długo (ponad 1 ms)	
Inter (Interface error)	D11	Sygnalizuje uszkodzenie interfejsu (układy 4049 i 74HC07).	Dalsza praca urządzenia zostaje wstrzymana i przeprowadzany jest cykliczny test interfejsu, gdy będzie poprawny urządzenie podejmie dalszą pracę



Rysunek 5. Schemat elektryczny urządzenia

Procedura wygląda poprawnie gdy trwa transfer danych ale co z pierwszym bitem? Czy milcząco założono, że będzie on równy jeden? Nie, po przejściu w tryb wysyłania danych do hosta jest odpowiednio ustawiana flaga FL_SEND_ZERO na podstawie pierwszego bitu bajtu do wysłania. Schemat występowania przerwań pokazuje **rysunek 3**. W przerwaniach wykonuje się kod emulacji oznaczony ciemnozielonym kolorem. Kolor czerwony to przerwanie od komparatora A (wykrycie *reset*), pomarańczowy komparator B (reset overdrive). Oprogramowanie przerwania comB stwierdza czy impuls resetu trwał 70 µs czy 440 µs. Jeśli 440 µs postępuje tak jak w przypadku przerwania od komparatora A (czerwone tło z białą czcionką). Podczas procedur emulacji w przerwaniu INT5, po zdekodowaniu rozkazu *skip* lub *match overdrive*, uruchamiany jest tryb overdrive (zezwolenie na INT4 i CompareB, blokada INT5). Tryb ten jest oznaczony kolorem seledynowym. Przerwanie INT6 lub COMPB może przerwać pętlę w INT5. Jeśli był to *reset* overdrive lub operacja na bicie (zapis, odczyt), program powraca do overdrive, jeśli stwierdzono standardowy sygnał reset następuje powrót do programu głównego. Jest to dość skomplikowane ale działa.

Pewną wadą takiego rozwiązania jest fakt, że program główny nie jest wykonywany

podczas przejścia do trybu overdrive, ale jest to emulator i program główny ma niewiele do zrobienia. Ponadto overdrive obsługuje niewiele układów i jest on włączany na krótko.

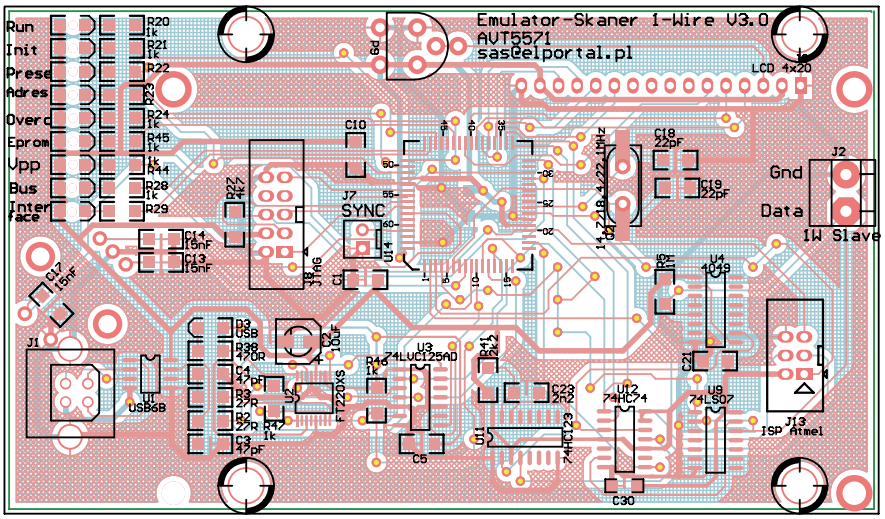
Jak się później okazało, po optymalizacji wszystko realizuje jedno przerwanie z wstawką asemblerową. Dlaczego więc opisałem sposób z dwoma przerwaniami? Ponieważ wymagania stawiane szybkości pracy mikrokontrolera są mniejsze i możliwa jest implementacja emulacji na niższych, częstotliwościach. Kolejne przyspieszenie uzyskuje się dzięki wykorzystaniu przerzutnika D. Na jego wejście podawany jest stan bitu do wystawienia w następnej transmisji zamiast ustawiania flagi FL_SEND_ZERO. Opadające zbocze sygnału strobującego odczyt przepisuje stan wejścia D przerzutnika na jego wyjście, które za pośrednictwem bramki z otwartym kolektorem wystawia bit na magistralę. Odstępstwem emulowanego układu od protokołu jest próbkowanie stanu bitu w trybie overdrive po 3 µs a nie 1 µs od opadającego zbocza sygnału ale biorąc pod uwagę, że czas stanu niskiego bitu 0 wynosi 7,5 µs nie stanowi to problemu. Jeśli jednak zajdzie konieczność zachowania pełnej zgodności należy wykorzystać sprzętowy układ odczytujący stan bitu składający się z multiwibratora 74HC123 i przerzutnika D. Na **rysunku 4** pokazano analizę działania programu przy taktowaniu 16 MHz i bez przerzutnika D. Impuls kończący się ponad 2 µs po opadającym zboczu odczytu. Dla programowych hostów nie stanowi to problemu, natomiast układy DS248x odczytują bit dużo wcześniej i wtedy pojawiają się błędy. Oczywiście przy wykorzystaniu przerzutnika D impulsu nie ma, tak jak i przy zwiększeniu częstotliwości taktującej do 22 MHz.

Budowa i działanie

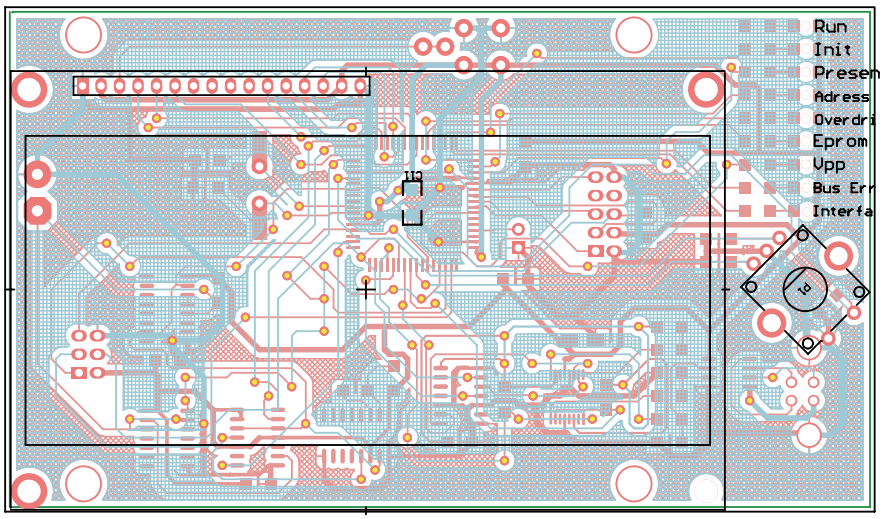
Schemat urządzenia pokazano na **rysunku 5**. Zasilanie jest doprowadzone z portu USB. Konwerter USB-SPI akceptuje na wejściach napięcia do 3,3 V dlatego zastosowano konwerter poziomów na układzie 74LVC125. Komunikację z użytkownikiem zapewnia enkoder impulsowy i wyświetlacz LCD 4×20 znaków. Elementy te zamontowano od spodniej strony płytki dzięki czemu łatwiej urządzenie zamknąć w obudowie. Kontrast wyświetlacza można regulować potencjometrem. Elementy U11 i U12 wspomagają sprzętowo obsługę trybu overdrive. Diody led informują o stanie urządzenia (**tabela 1**).

Montaż i uruchomienie

Schemat płytki drukowanej wraz z rozmieszczeniem elementów pokazano na **rysunku 6 i rysunku 7**. Montaż jest typowy i nie wymaga



Rysunek 6. Schemat płytki PCB strona TOP



Rysunek 7. Schemat płytki PCB strona BOTTOM

omawiania. Aby ułatwić sobie pracę enkoder impulsowy i wyświetlacz należy zamontować jako ostatnie elementy (od spodniej strony płytki). Wyświetlacz z płytką warto zamontować za pomocą wtyku i gniazda goldpin, co ułatwi ewentualne naprawy. Urządzenie może pracować bez tych elementów, wtedy całe sterowanie odbywa się za pośrednictwem programu terminala. Ustawienie bitów konfiguracyjnych to: EXT=0xFD, HIGH=0x19, LOW=0xEF (**rysunek 8**).

Obsługa

Po ekranie startowym z informacją o trybie pracy, statusie (CPU, VID, PID) pojawi się menu główne z treścią podobną jak na **rysunku 9**. Przyciskiem enkodera zmieniamy cyklicznie typ emulowanego układu a obracając enkoder zmieniamy np emulowaną temperaturę. Znaczenie liter po CFG jest następujące:

P – (mała/wielka litera) emulowanie zasilania pasożytniczego lub lokalnego DS18B20. Ustawianie programem terminala.

I – (mała/wielka litera) generowanie lub nie alarmu/IRQ przez slave. Ustawianie programem terminala.

A – (mała/wielka litera) alarm temperatury. Ustawiany automatycznie podczas emulacji termometru na podstawie rejestrów tH i tL.

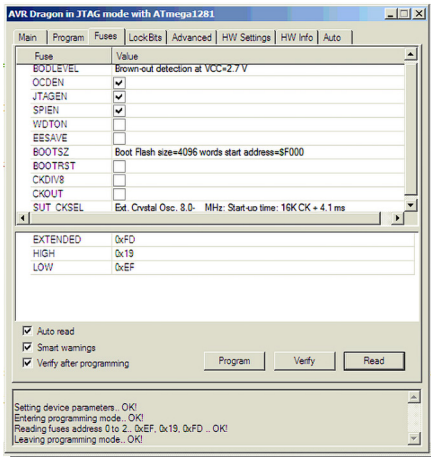
Sxyy – synchronizacja oscyloskopu, gdzie **x** oznacza:

r – sygnałem *reset*,

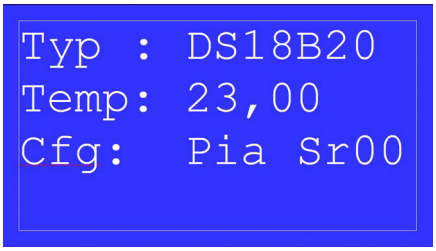
p – sygnałem *presence*,

c – komendą,

f – rodziną układu.



Rysunek 8. Ustawienie bitów konfiguracyjnych



Rysunek 9. Widok okna menu głównego

yy – oznacza nr komendy lub rodziny (zależy od parametru **x**) układu, z którą synchronizujemy oscyloskop. Opcje ustawianie programem terminala.

Podczas emulacji/skanowania overdrive reakcje wyświetlacza nie są zbyt szybkie, jest to spowodowane dużym obciążeniem mikrokontrolera podczas operacji symulacji i szybkim transferem danych po USB w trybie skanera.

Sterowanie urządzeniem jest możliwe całkowicie poprzez program typu terminal, wykaz komend sterujących zawiera **tabela 2**.

Uwagi końcowe

Mikrokontroler jest nieznacznie „przetaktowany" (18,4321 MHz zamiast 16 MHz) ale nie wpływa to na pogorszenie stabilności pracy układu.

Przebadałem reakcję układów slave na przedłużony reset. Te zasilane pasożytniczo, zachowują się poprawnie, po powrocie zasilania, po *resecie* zaczynają pracować wystawiając *presence*. Wewnętrzna pojemność jest rozładowana w czasie sygnału *reset*, a gdy wróci zasilanie (dlatego wymagane jest po resecie przetrzymanie magistrali w stanie wysokim) w układach tych następuje wewnętrzny *reset*. Układy zasilane zewnętrznie (np termometr DS18B20, nr seryjny DS2411), także reagują prawidłowo. Układ DS2482, również zachowuje się poprawnie (nie wysyła danych w „kosmos", ale w przeciwieństwie do DS2480 nie informuje o tym fakcie w statusie.

Znane mi oprogramowanie master i slave działa błędnie w takiej sytuacji. Master wysyła dane choć magistrala po zakończeniu resetu przez mastera jest jeszcze w stanie niskim i slave zakłóca transmisję innych układów. Proszę wziąć to pod uwagę w czasie pisania oprogramowania. W przypadku układów slave nigdy nie wiadomo, czy jakiś slave nie zgłosi przerwania.

Dobrą praktyką w operacjach 1-Wire jest reset magistrali po zakończeniu komunikacji z urządzeniem.

Tabela 2. Wykaz komend sterujących

Komendy bez parametru (naciśnięcie wybranego klawisza)	
Komenda	**Funkcja**
?	wyświetlenie spisu komend
SPACJA	start/stop przechwytywania
TABULATOR	emulacja naciśnięcia przycisku enkodera
SHIFT + '+'	zwiększenie temperatury emulowanej przez DS18B20
SHIFT + '–'	zmniejszenie temperatury
Komendy bez parametru (wydanie komendy zatrzymuje przechwytywanie)	
Komenda	Funkcja
:v	wersja programu
:n	nazwa programu
:R	restart
Komendy z parametrem:	
Komenda	**Funkcja**
:Ex	echo on/off gdzie x=1 on, x=0 off
:txx	ustawienie temperatury emulowanego DS18B20, gdzie xx temperatura w postaci dziesiętnej
:e	typ emulowanego układu, zakres 01...FF. Na poziomie komend podstawowych (ReadRom, SkipRom, SearchRom) można emulować dowolny układ, tak komendy specyficzne dla danego układu są emulowane tylko dla numerów seryjnych, DS18B20 i DS2431
:ix	alarm/IRQ on/off. Włącza lub wyłącza generowanie alarmu przez slave po wygenerowaniu sygnału reset przez host
:cxx	filtrowanie ramek dla wybranej komendy (0 – wyłącza filtrowanie)
:fxx	filtrowanie ramek dla wybranej rodziny (28 – DS18B20, 01 – nr seryjny), 0 – wyłącza filtrowanie. Gdy aktywne są oba filtry wyświetlane będą tylko ramki ze zgodnym kodem rodziny i komedą.
:px	włączenie/wyłączenie zasilania pasożytniczego dla DS18B20
:sx	włączenie synchronizacji oscyloskopu gdzie: x=1 – tylne zbocze sygnału reset, x=2 – tylne zbocze sygnału presence, x=3 – zgodna komenda ustawiana rozkazem „:c", x=4 – zgodny kod rodziny ustawiany rozkazem „:f". Synchronizacja oscyloskopu funkcjonuje tylko gdy przechwytywanie jest włączone.

Baza emulowanych układów jest rozbudowywana. Aktualnie trwają prace nad: DS1820, DS1821, DS1920, DS1961, DS1973, DS2430, DS2432, DS2433. W sprawie nowych wersji oprogramowania proszę o kontakt.

SaS
sas@elportal.pl