Nie bój się VFD - *post scriptum*

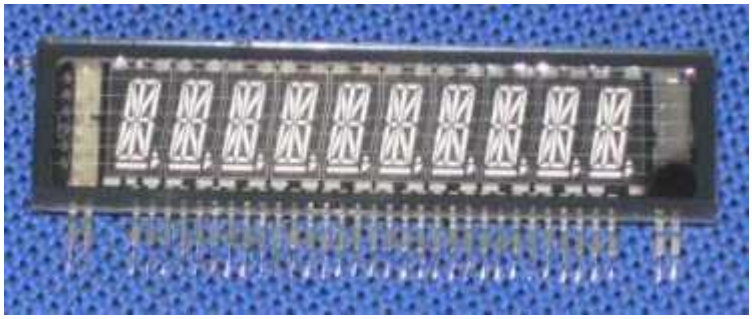
Elektronika dla Wszystkich, 5/2005

"*Budując kolejny super zegar, termometr cyfrowy lub inny dowolnego rodzaju wynalazek(...)*"

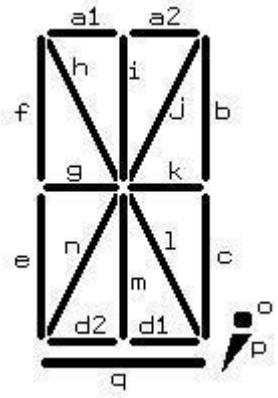
Tak, tak, to już znamy. Więc może inaczej: planując użycie lampy VFD we własnej konstrukcji musimy zaprojektować układ sterowania siatek i anod wyświetlacza. Jakie rozwiązanie układowe będzie najodpowiedniejsze? Jakie mamy możliwości?

Zanim omówimy poszczególne przykładowe aplikacje, zapoznajmy się z dwoma lampami VFD, które będą nam dzielnie towarzyszyć do końca tego artykułu.

Przedstawiam zatem: wyświetlacz **10-LY-01** firmy Futaba (**fotografia 1**)

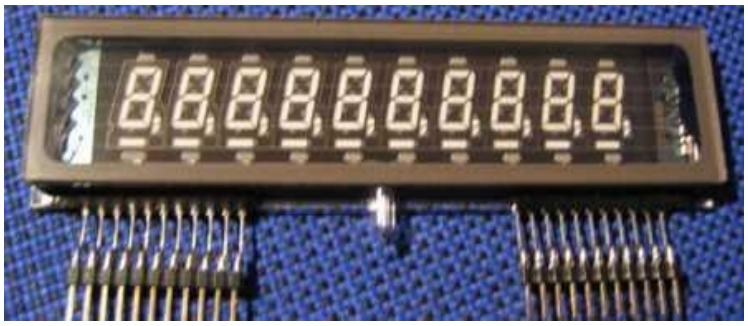


Jest to niewielka lampa, posiadająca dziesięć dziewiętnastosegmentowych pól odczytowych o układzie anod jak na **rysunku 1**

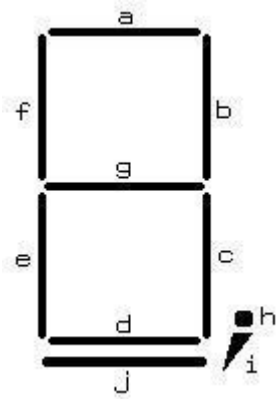


Doświadczalnie ustalone napięcie zasilania żarnika to około 3.6V (dla żarzenia DC)

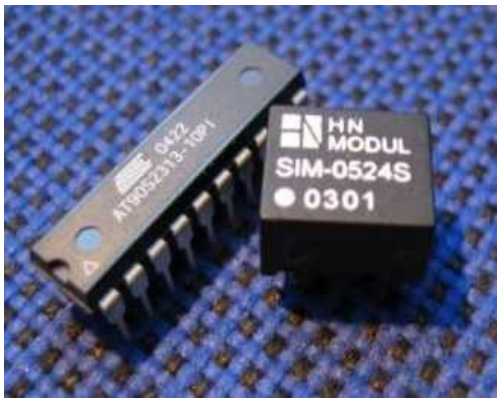
Druga lampa, która także weźmie udział w naszych eksperymentach to wyświetlacz **FG1010RB1** firmy Itron.
Jest to dość prosty wyświetlacz o dziesięciu polach odczytowych, a można go spotkać w cyfrowych wagach sklepowych lub kasach fiskalnych. Lampkę prezentuje **fotografia 2**



a układ jej segmentów **rysunek 2**



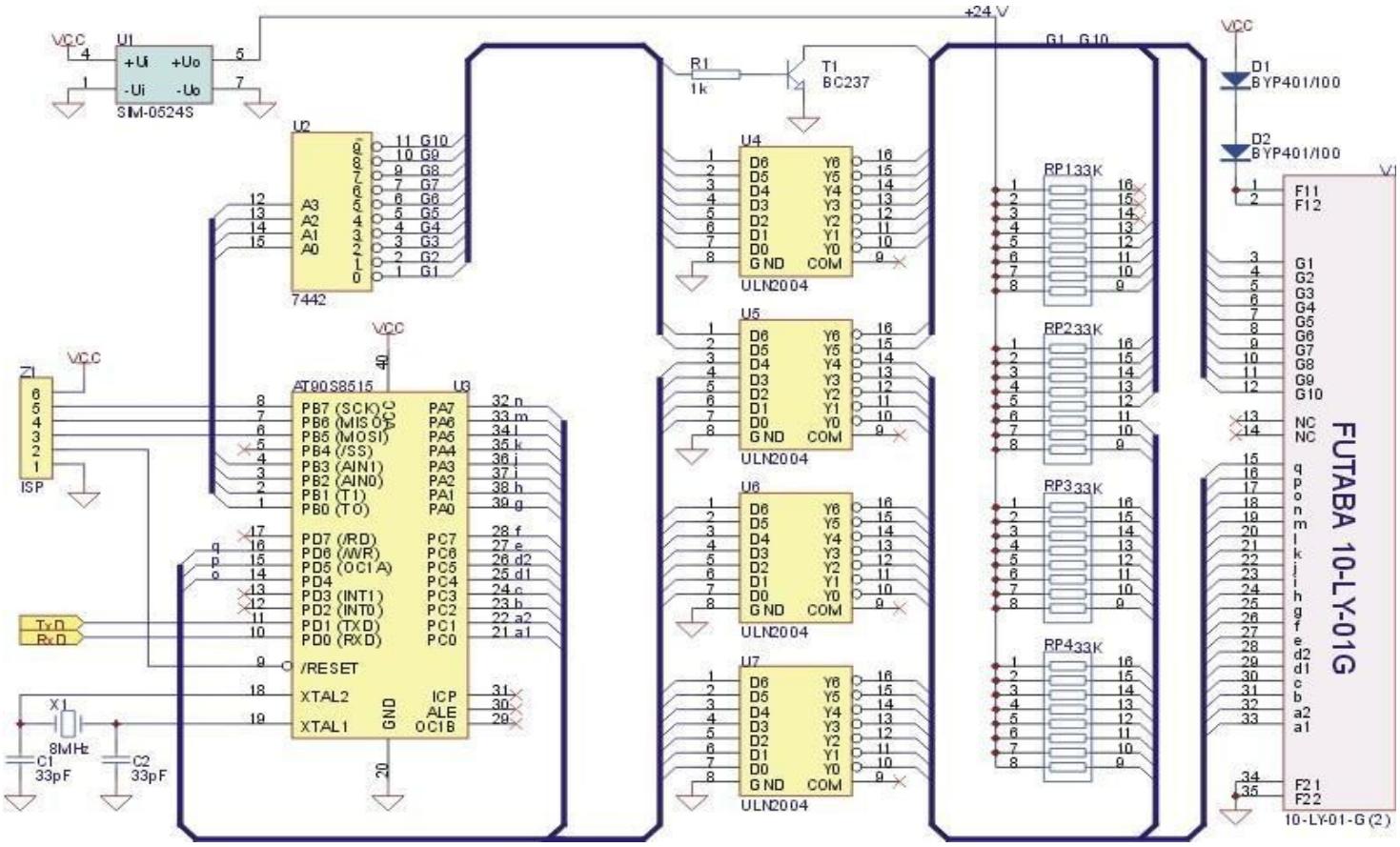
Do poprawnej pracy lampie tej wystarczy napięcie żarzenia o wartości około 4V DC.
W naszych układach zastosujemy żarzenie prądem stałym, odpowiedni spadek napięcia na końcówkach żarnika uzyskamy przez szeregowe włączenie zwykłych diod krzemowych.
Oba wyświetlacze pracują poprawnie przy napięciu anodowym 24V (co też ustalono doświadczalnie), więc do zasilania siatek i segmentów wykorzystamy małą przetwornicę DC/DC typu SIM-0524S, konwertującą napięcie 5V na 24V przy wydajności prądowej strony wtórnej 50 mA. Przetworniczkę podziwiamy na **fotografii 3** (tu w towarzystwie procesora AT90S2313, aby lepiej zaprezentować ogrom jej rozmiarów)



a potem przechodzimy do głównego tematu.

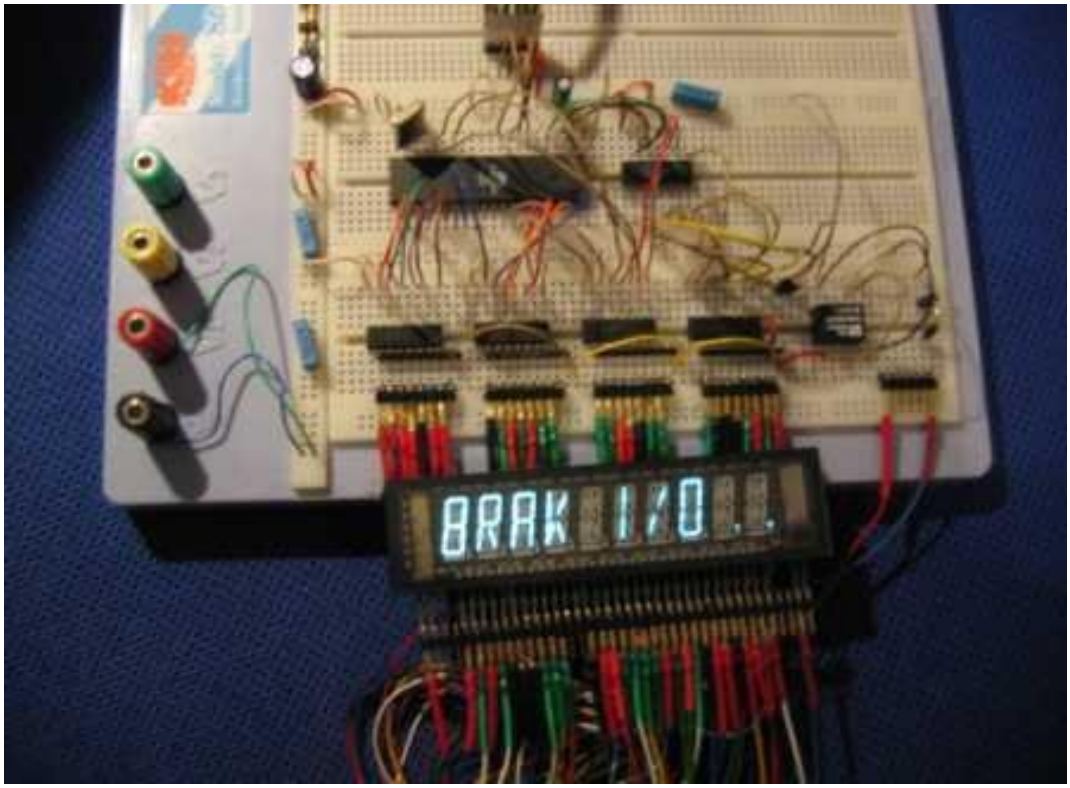
<u>Odsłona pierwsza - sterowanie proste i skuteczne</u> , ale...

Popatrzmy na schemat ideowy na **rysunku 3**.



Układ sterowania jest bardzo podobny do układu przedstawionego w artykule "Nie bój się VFD...", tylko zamiast tranzystorów npn zostały wykorzystane układy ULN2004 (siedem tranzystorów npn w układzie Darlingtona).
Ponieważ lampa 10-LY-01 wymaga 29 linii sterujących (10 siatek + 19 segmentów), konieczny był jeden dodatkowy tranzystor, na schemacie oznaczony jako T1.
Zasada działania, tak dla przypomnienia:
do wybierania siatek wyświetlacza używamy dekodera 4/10 typu 7442 (U2), którego aktywne wyjście określa odpowiednia kombinacja stanów logicznych bitów 0..3 portu PB. Niski stan logiczny wybranego wyjścia dekodera otwiera odpowiedni tranzystor, co powoduje, że siatka przyjmuje potencjał +24V. Pozostałe, nieaktywne wyjścia są w stanie wysokim, co sprowadza resztę siatek praktycznie do potencjału masy.
Segmenty wyświetlacza sterujemy bezpośrednio z portów PA,PC i trzech bitów (4,5,6) portu PD.

Uruchamiamy zatem oprogramowanie sterujące (plik **demo1.c**) i na wyświetlaczu (**fotografia 4**) ukaże się wymowny napis:

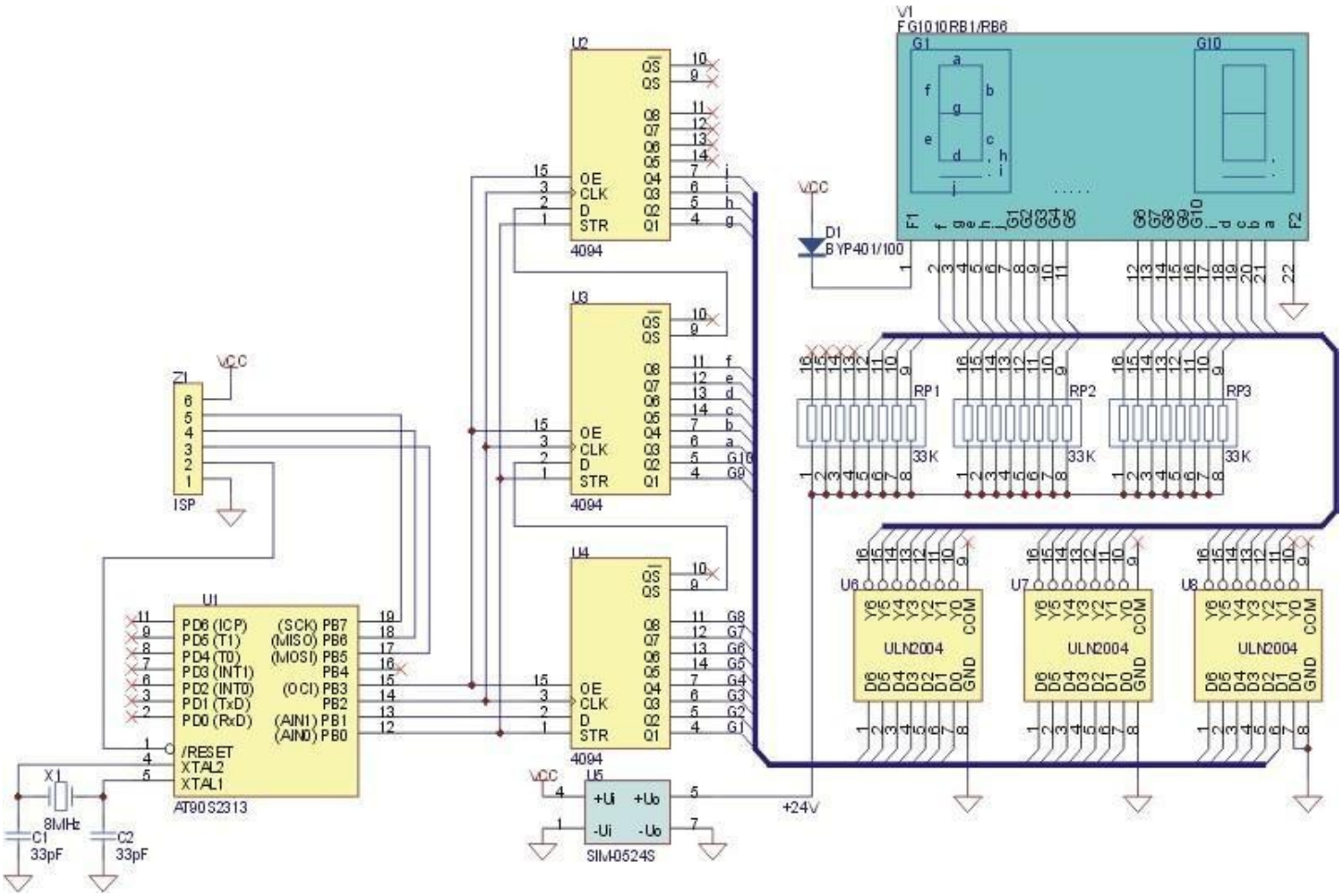


Niestety program ma rację.
Zaproponowany układ sterowania, pomimo że bardzo prosty w realizacji i oprogramowaniu zagospodarował nam praktycznie wszystkie dostępne linie I/O procesora. Kiedy więc taka aplikacja ma sens?
Na przykład, gdy budujemy urządzonko mające pełnić rolę inteligentnego panela odczytowego, dlatego też na **rysunku 3** zostały narysowane linie do transmisji szeregowej.
Ale gdy do procesora zamierzamy podłączyć jeszcze inne układy peryferyjne należy zastanowić się nad bardziej oszczędnym wykorzystaniem jego portów. Dlatego też...

Odsłona druga - zmieniamy podejście z równoległego na szeregowe

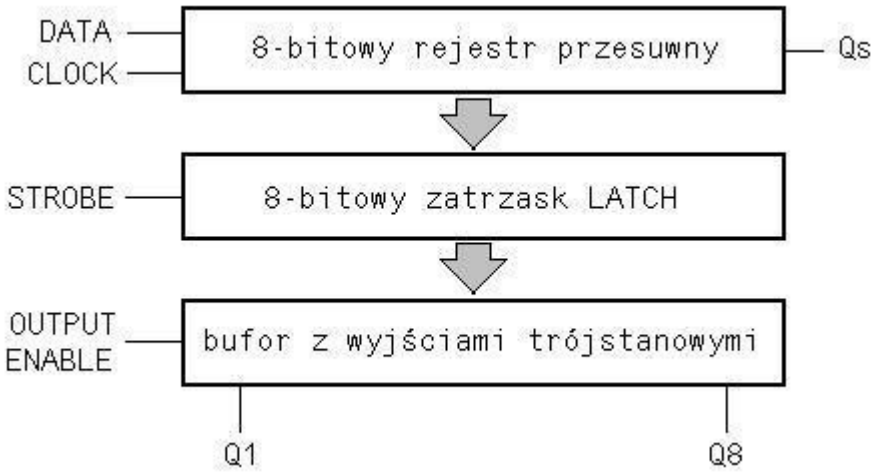
...i aby zbytnio nie inspirowała nas duża liczba portów I/O procesora AT90S8515, zastąpimy go mniejszym, na przykład AT90S2313. A niejako przy okazji sprawdzimy jak działa wymieniona na wstępie druga lampa - Itron FG1010RB1.

Spójrzmy na **rysunek 4**



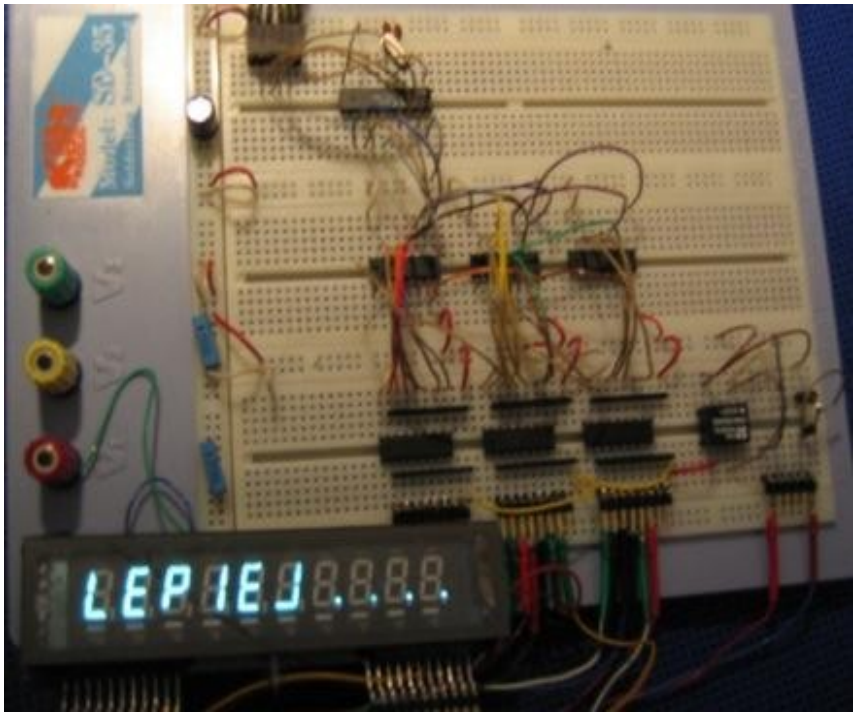
Najciekawszym fragmentem schematu są trzy układy **CD4094** (U2,U3,U4) - ośmiobitowe rejestry przesuwne z wyjściem równoległym. Dokumentację tych kostek można znaleźć na stronach firmy Texas Instruments (**http://focus.ti.com/lit/ds/symlink/cd4094b.pdf**), więc w tym miejscu zasada działania w wielkim skrócie.

Schemat blokowy układu CD4094 przedstawia **rysunek 5**.



Stan logiczny podany na wejście DATA jest wpisywany do rejestru szeregowego wraz z narastającym zboczem sygnału zegarowego CLOCK. Jeżeli na wejście STROBE podamy wysoki stan logiczny, zawartość rejestru będzie natychmiast widoczna na wyjściach Q1...Q8 układu, opadające zbocze sygnału STROBE zapisze ostatnio dostępną zawartość rejestru szeregowego w ośmiobitowym rejestrze LATCH. Wyjścia układu będą odzwierciedlać stan rejestru LATCH tak długo, jak na wejściu OUTPUT ENABLE będzie podany wysoki stan logiczny. W przypadku zera logicznego, wyjścia Q1...Q8 przechodzą w stan wysokiej impedancji (High-Z). Ważną dla nas cechą układu CD4094 jest możliwość łączenia go w kaskady - wyjście **Qs** n-tego układu łączymy po prostu z wejściem **DATA** następnego układu w kaskadzie, a wejścia STROBE, OUTPUT ENABLE i CLOCK sterujemy wspólnymi sygnałami - co możemy zobaczyć właśnie na **rysunku 4**.

Ponieważ lampa Itron FG1010RB1 wymaga tylko dwudziestu linii sterujących, cztery ostatnie wyjścia układu U2 (ostatni w kaskadzie) pozostają niewykorzystane. Do wyjść naszego 20-bitowego rejestru podłączamy "drivery" (znowu ULN2004 w ilości trzech sztuk) a do nich lampę.
Po uruchomieniu oprogramowania sterującego (plik **demo2.c**) wyświetlacz (**fotografia 5**) pokaże nam że jest...



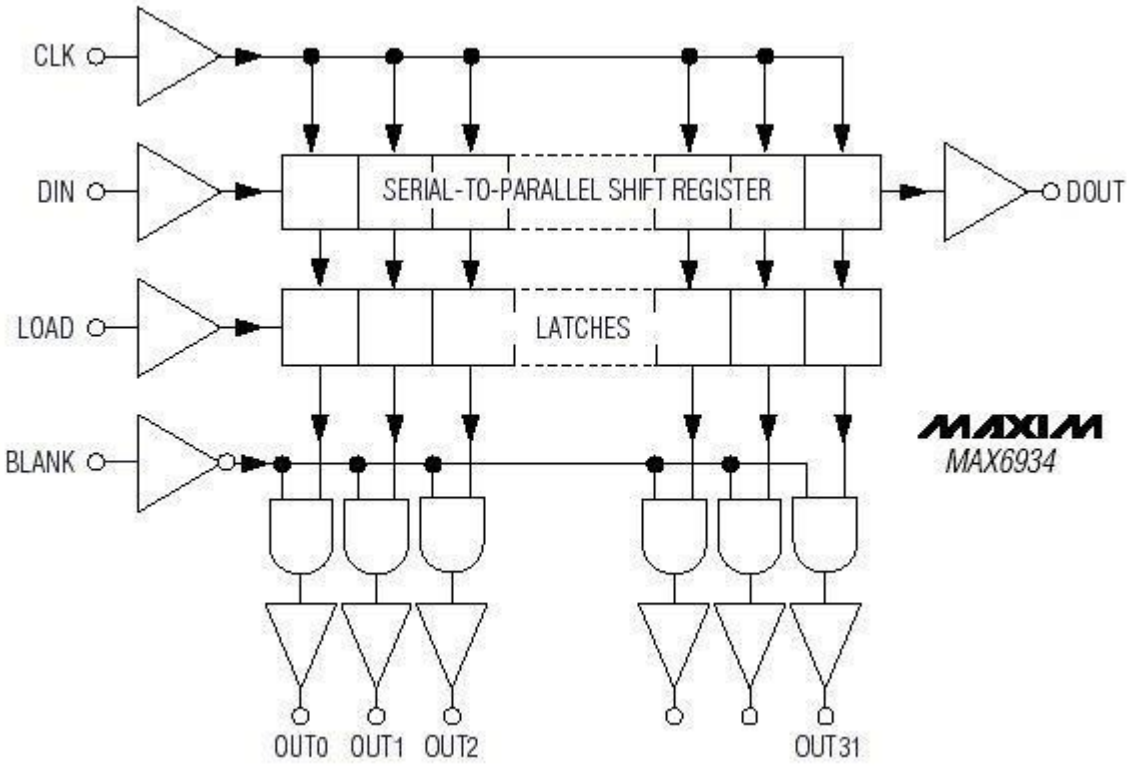
No i słusznie. Zróbmy więc w tym miejscu mały bilans:

zalety - do sterowania display-a wykorzystaliśmy tylko cztery linie I/O kontrolera, dzięki możliwości kaskadowego łączenia układów CD4094 możemy w ten sposób zaprojektować sterowanie nawet bardzo skomplikowaną (o dużej ilości wyprowadzeń) lampą VFD;
wady - sześć układów scalonych, sześć podstawek, 96 otworków w laminacie (o rezystorach polaryzujących nie wspominając)...

Jeżeli uznamy, że to jednak zbyt skomplikowane to...

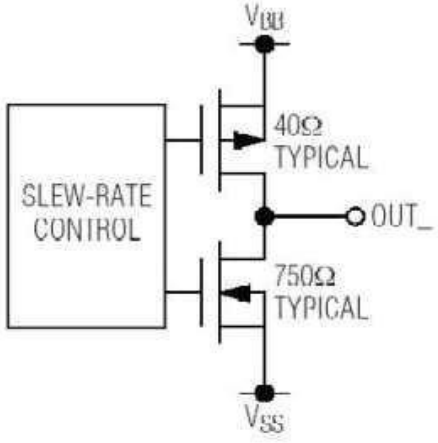
Odsłona trzecia - sterowanie MAXIM-alne

...w tym momencie przychodzi nam z pomocą ogólnie lubiana firma MAXIM, która (**jeszcze!!**) w miarę
chętnie wysyła zainteresowanym osobom próbki swoich układów scalonych.
Zajrzyjmy na stronkę MAXIM-a, a konkretnie tu: **http://pdfserv.maxim-ic.com/en/ds/MAX6922-MAX6934.pdf**
I to jest to czego nam potrzeba!
Wszelkie niezbędne detale są zawarte w dokumentacji układu, wiec w tym miejscu opis sterownika w telegraficznym wręcz skrócie.
Popatrzmy na wnętrze układu (**rysunek 6** zaczerpnięty z dokumentacji):

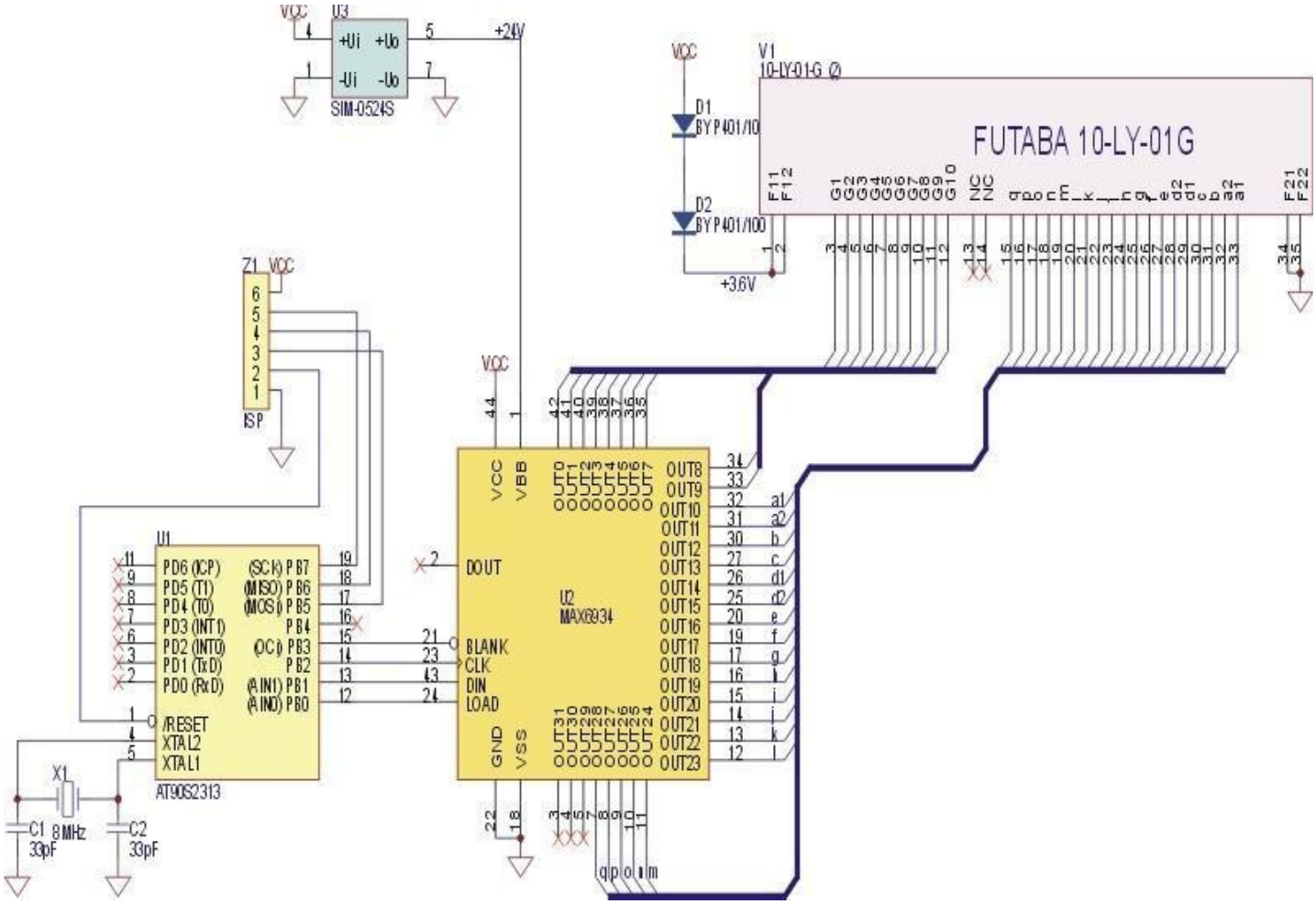


MAX6934 to sterownik lampy VFD zawierający w środku 32-bitowy rejestr przesuwny,
do którego dane z wejscia DIN ładowane są szeregowo w takt sygnału zegarowego CLK.
Wyjścia rejestru podłączone są do 32-bitowego zatrzasku typu latch sterowanego linią LOAD.
Jeżeli na wejście LOAD podamy wysoki stan logiczny, zatrzask jest "przezroczysty" dla danych z rejestru szeregowego.
Opadające zbocze sygnału LOAD zapamiętuje ostatnią wartość rejestru.
Rejestr równoległy połączony jest z "driverami" poprzez logikę sterowaną sygnałem /BLANK, która umożliwia programowe wygaszenie
wyświetlacza bez modyfikacji zawartości rejestru lub kontrolę jego jasności sygnałem PWM.
Generalnie: jeżeli wejście /BLANK jest w niskim stanie logicznym - wyjścia OUT0...OUT31 sterownika są aktywne i odzwierciedlają stan zatrzasku wyjściowego.

Każde z wyjść (OUT0...OUT31) układu MAX6934 wyposażone jest w "końcówkę mocy" zbudowaną w oparciu o dwa tranzystory CMOS (**rysunek 7** - także z dokumentacji), umożliwiającą bezpośrednie podłączenie do wyprowadzenia lampy VFD.



Warto pamiętać o tym, że układ MAX6934 posiada możliwość łączenia w kaskady (na identycznej zasadzie jak układy CD4094), dane dla następnego w łańcuchu układu pobieramy z wyjścia DOUT poprzedniego. Tyle o samym MAX6934. Spójrzmy teraz na schemat ideowy na **rysunku 8**.



Witamy ponownie lampę Futaba 10-LY-01 tym razem we wręcz minimalistycznym układzie sterującym.
Kostka MAX6934 kontrolowana jest przez cztery linie I/O procesora AT90S2313, lampę 10-LY-01 podłączamy bezpośrednio do układu MAX6934.
Wyjścia **OUT0...OUT9** układu połączone są z wyprowadzeniami siatek **G1...G10**, wyjścia **OUT10...OUT28** z wyprowadzeniami segmentów **a1...q**.

I to wszystko!

Cała sztuka polega na odpowiednim oprogramowaniu tego układu (plik **demo3.c**), dalej więc trochę "odczarujemy" sposób programowania tej kostki.

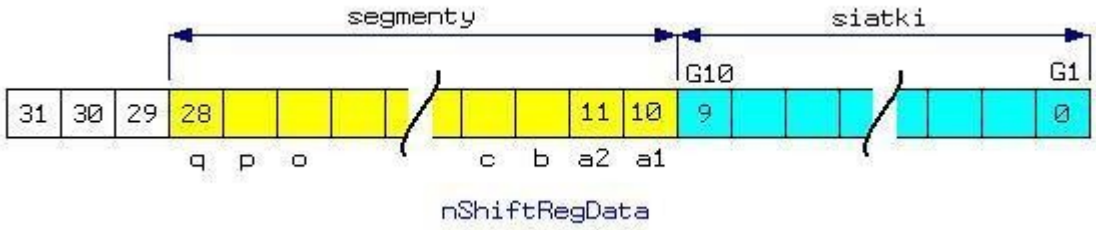
Ponieważ stosujemy sterowanie multipleksowane (w jednej chwili czasowej aktywna jest tylko jedna siatka), zdefiniujmy sobie funkcję, która jako pierwszy parametr dostanie numer pola odczytowego (siatki), a jako drugi parametr podany zostanie wzorzec bitowy określający kształt znaku do wyświetlenia. Na przykład tak:

```
void vfd_update_reg (unsigned char nGridNum, unsigned long ulData );
```

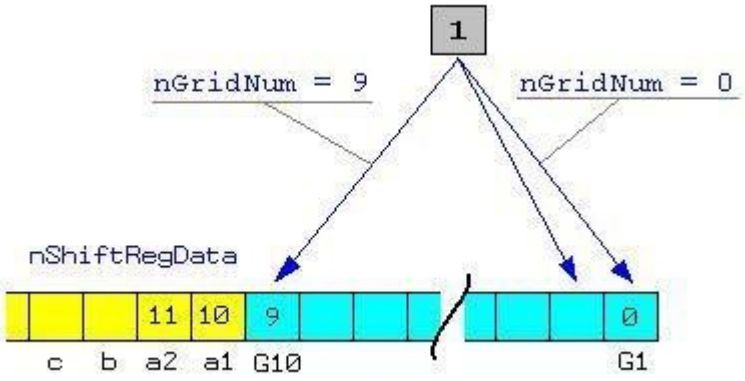
Parametr **nGridNum** przyjmuje wartość 0..9 (lampa ma 10 pól odczytowych), a z parametru **ulData** interesować nas będzie tylko pierwsze 19 bitów tej wartosci (bo tyle segmentów ma nasza lampa).
Teraz definiujemy zmienną tymczasową, z której odpowiednio spreparowane dane zostaną przesłane do rejestru układu MAX6934.

```
unsigned long nShiftRegData;
```

Jak pamiętamy, lampa ta wymaga 29 bitów danych, dlatego zmienna tymczasowa **nRegShiftData** jest typu `unsigned long` (32 bity) a jej oczekiwaną zawartość pokazuje **rysunek 9**.



Najpierw zajmiemy się siatkami lampy. Musimy wartość **nGridNum** przekształcić na odpowiednią kombinację bitów 0...9 zmiennej **nShifRegData** dlatego wykonamy sobie "*programowy dekoder 4/10*" według **rysunku 10**.



pisząc jedną linię kodu:

```
nShiftRegData = 1 << nGridNum;
```

Tym sposobem siatki mamy z głowy, teraz bity odpowiedzialne za segmenty.
Ponieważ pierwszych dziesięć bitów zmiennej **nShifRegData** jest już zajętych, zawartość **ulData** przesuwamy bitowo w lewo o wartość **VFD_GRIDS_NUM** (stała określająca ilość siatek lampy),
a następnie składamy w całość operacją sumy logicznej:

```
nShiftRegData |=  ulData << VFD_GRIDS_NUM;
```

Po tej operacji zmienna tymczasowa zawiera gotowe dane dla rejestru MAX6934. Pozostała tylko ich wysyłka.
Kolejne bity zmiennej **nShiftRegData** (począwszy od najstarszego /**31**/) wyślemy do rejestru w jednej pętli, generując odpowiednie stany logiczne na wejściu **DIN** rejestru i sygnał przebiegu zegarowego na wejściu **CLK**:

```
signed char nBitNo;
nBitNo = VFD_REG_SIZE - 1;    // zaczynamy od 31 bitu
while ( nBitNo >= 0 ) {
      // linia DIN w stan L
      VFD_REG_DIN_L;
      // jezeli odpowiedni bit zmiennej tymczasowej nShiftRegData jest 1
      // to przestawiamy ja na H
      if ( nShiftRegData & ((unsigned long)1 << nBitNo) ) {
            VFD_REG_DIN_H;
      }
      // majac ustalona wartosc linii DIN, generujemy sygnal zegarowy
      // (jeden takt) aby stan z linii DIN wpisal sie do rejestru przesuwnego
      NOP;
      VFD_REG_CLK_H;
      NOP;
      VFD_REG_CLK_L;
      NOP;
      // kolejne bity 30,29,28...
      nBitNo--;
} // while
```

Po wykonaniu powyższej pętelki, rejestr będzie zawierał dokładnie to co ustaliliśmy w zmiennej **nShiftRegData**, pozostało tylko "zatwierdzić" wpisane dane impulsem na wejściu **LOAD**:

```
NOP;
VFD_REG_LOAD_H;
NOP;
VFD_REG_LOAD_L;
NOP;
```

No a gdzie to multipleksowanie?

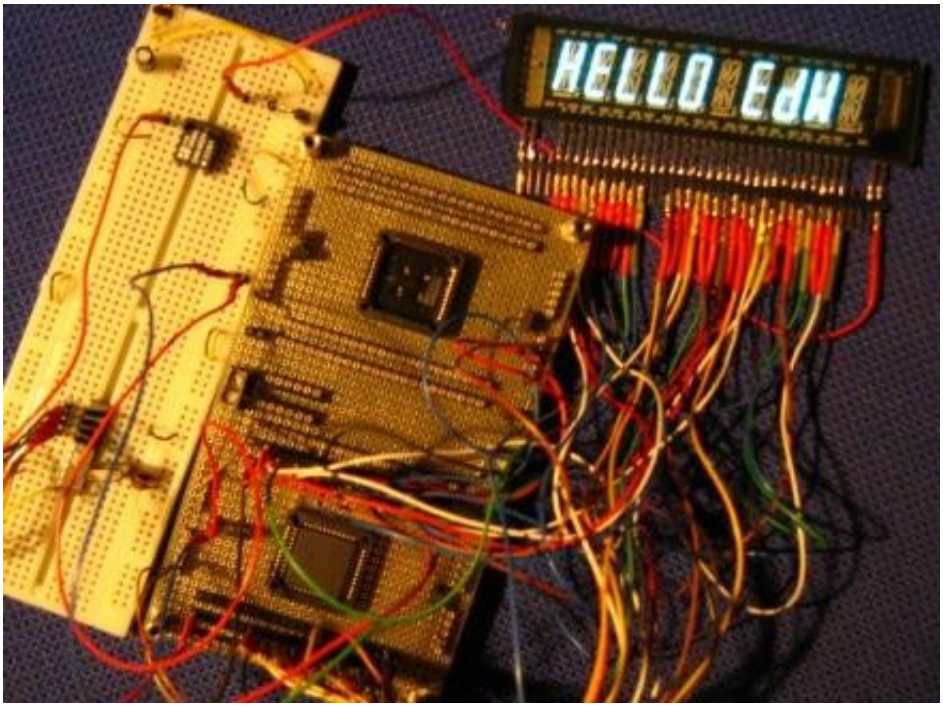
Napiszmy więc funkcję, która wywoływana z poziomu handlera przerwania zegarowego, będzie dbała o podawanie poprawnych wartości do funkcji **vfd_update_reg()**, na przykład tak:

```
void vfd_update_display ( void ) {

	// tu wysyłamy "paczkę" danych do rejestru MAX6934
	vfd_update_reg ( uchVfdCurrentGrid, ulVfdDataBuffer [ uchVfdCurrentGrid ] );

	// tu zwiększamy licznik siatek
	uchVfdCurrentGrid++;

	// a kiedy dojdziemy do ostatniej, zerujemy go, aby zacząć od początku...
	if ( uchVfdCurrentGrid == VFD_GRIDS_NUM ) {
		uchVfdCurrentGrid = 0;
	}
}
```

Powyższa funkcja korzysta z dwóch zmiennych globalnych: **uchVfdCurrentGrid** oraz **ulVfdDataBuffer**, zdefiniowanych następująco:

```
unsigned long	ulVfdDataBuffer [ VFD_GRIDS_NUM ];
unsigned char	uchVfdCurrentGrid = 0;
```

Zawartość tabelki **ulVfdDataBuffer[]** jest kontrolowana przez inne, bardziej wysokopoziomowe funkcje, zapewniające konwersję znaków ASCII na wzorce bitowe, kasowanie display-a, itp. - po szczegóły odsyłam do kodu źródłowego **demo3.c**. A jak już jesteśmy przy pliku **demo3.c** - uruchommy wreszcie nasz programik aby podziwiać standardowy w takich przypadkach, wyświetlany przez MAXIM-alnie sterowaną lampę VFD napis...



Natasza Biecek
natasza.biecek@gmail.com