Cyfrowa archeologia - mikrokontroler 80C42

Prolog (gadu-gadu)

PG (7-05-2008 14:58)

„(...) to bardzo interesujący przykład. Dla starszych Czytelników prawdopodobnie kwestia sentymentalna poruszająca serce, a dla najmłodszych z apewne muzeum. Wiec nie ma odpowiedzi jednoznacznej. Z punktu widzenia EdW to interesująca ciekawostka - pokazanie jak to się robiło dawniej, jakie były możliwości i problemy."

Inspiracja (Forum Elportalu)

Forum *elportal.pl* stało się dla mnie ostatnio doskonałym źródłem inspiracji do własnych poszukiwań. Często pojawiają się tam dyskusje, udział w których owocuje wieloma pomysłami i w jakiś sposób zmusza do aktywności w dziedzinie elektroniki. Poniższy tekst inspirowany był wątkiem z Forum założonym przez Adama Kulpińskiego (Kulpina) o tytule „Recycling, czyli wykorzystanie elementów z odzysku... „.

Pierwotnie sprawa dotyczyła pamięci EPROM, potem temat zszedł na stare mikrokontrolery i ogólnie na wykorzystanie zabytkowych, przestarzałych już komputerów. I właśnie te stare mikrokontrolery szczególnie nie dawały mi spokoju. Zabytkowe płyty główne PC, które aktualnie coraz częściej lądują na złomowisku, są doskonałym źródłem różnego rodzaju elementów - począwszy od drobnicy typu złącza, dławiki, kwarce itp., a skończywszy na pamięciach czy innych specjalizowanych układach. Niektóre z nich są tak mocno związane z budową płyty, że ich wykorzystanie poza dedykowanym układem jest praktycznie niemożliwe. Inne, pomimo że w elektronice płyty mają swoją ściśle określoną rolę, dają się „oswoić" na płytce stykowej i dostarczyć wielu emocji na zasadzie: zadziała mi czy nie? Przykładem takich właśnie układów są kontrolery interfejsu klawiatury. W starych płytach głównych (XT/AT/386) są to układy scalone, które można znaleźć w okolicach klawiaturowego złącza DIN. Jeżeli delikatnie pozbędziemy się okrywającej kostkę nalepki, jest bardzo duże prawdopodobieństwo, że okaże się ona się mikrokontrolerem jednoukładowym 8041 lub 8042 firmy Intel lub innej, która miała licencję na produkcję tego typu procesora (w tym tekście występuje układ firmy NEC). Mikrokontroler ma już zaszyte firmowe oprogramowanie (właśnie do obsługi klawiatury) i zaraz po wyjęciu z płyty głównej jest średnio użyteczny. Ale przecież możemy zmusić ten układ, aby wykonywał nasze, a nie wbudowane oprogramowanie. Kwestia tylko - jak to zrobić i ile to będzie wymagało wysiłku. Oczywistym jest, że aby taka zabawa zakończyła się sukcesem, musimy mieć dostęp do dokumentacji kontrolera i narzędzi programistycznych, czyli jakiegokolwiek assemblera akceptującego mnemoniki rodziny 8041/42. Z narzędziami problemu wielkiego nie miałam, ponieważ już dawno temu zlokalizowałam w Sieci bardzo sprytny translator assemblera SB-Assembler. Rozumie on mnemoniki bardzo wielu procesorów i kontrolerów, także 8041/42. Z dokumentacją był znacznie większy problem. Znalazłam przy pomocy Google bardzo interesującą stronkę: „Devster Specialties", na której pokazano przykłady układów z wykorzystaniem kontrolera 8042. Dodatkowo dowiedziałam się, że dalszych wiadomości o tych kostkach należy szukać w Google także pod hasłami UPI-41 oraz UPI-42. UPI to od angielskich słów „Universal Peripheral Interface" - ponieważ tak był nazywany i taką rolę pełnił w płytach PC.

Więc teraz, krok po kroku - przedstawię jak próbowałam nawiązać choćby najcieńszą nić porozumienia ze świeżo wyciągniętym ze starej płyty '386 kontrolerem 80C42.

Nalepka AMI (a pod nią...)

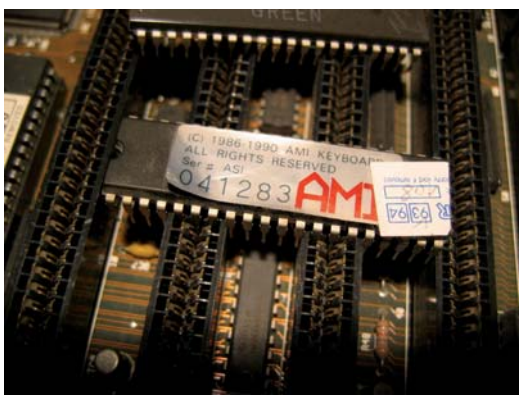
Kontroler, jak wspominałam wcześniej, siedział sobie cicho pośród innych elementów płyty głównej, zakryty dla niepoznaki firmową naklejką American Megatrends Inc. (**fotografia 1**). Nalepki mają to do siebie, że zawsze ciekawi co jest pod spodem. Po oderwaniu owej zobaczyłam coś, co wreszcie wyglądało znajomo: układ 80C42 firmy NEC (**fotografia 2**).

Niestety, dwie pozostałe płyty, które też pozwoliłam sobie „zbadać" na podobnej zasadzie, miały jako UPI nie mikrokontrolery z rodziny 8041/42, ale specjalizowane kostki. No trudno, dobrze że jest choć jeden do zabaw i testów, a zaraz okaże się co on jest wart.

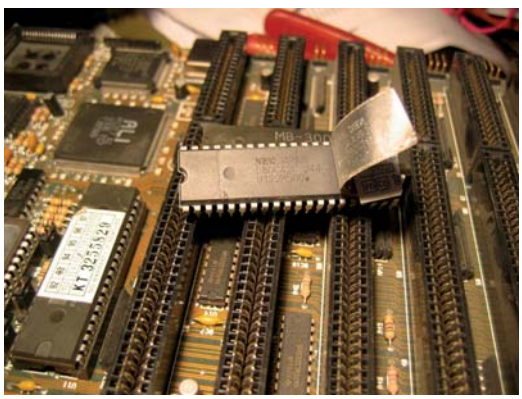
Klonowanie (przecież ktoś już to zrobił)

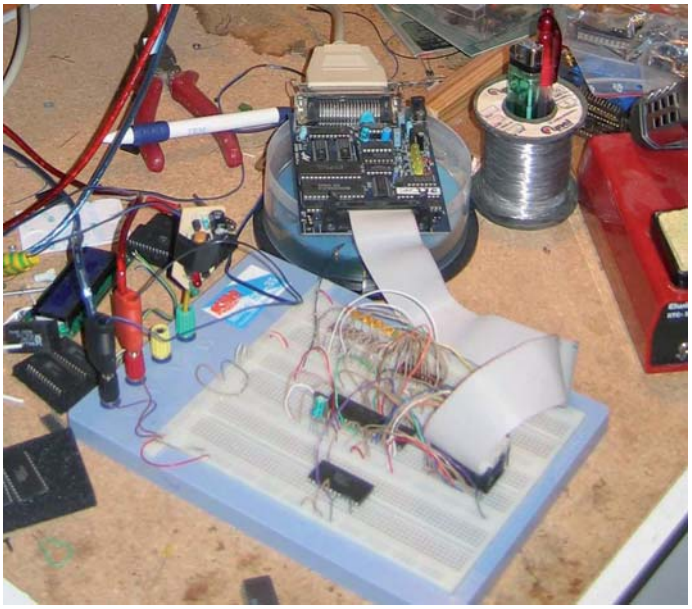
Na stronce „Devster Specialties" znajduje się podstrona „8042 and 8041 Microcontrollers", która okazuje się cenna z kilku powodów. Po pierwsze, jest tam link do dokumentacji kostki, po drugie trzy przykładowe programy i co ważne - zdjęcia działających, próbnych układów. To niejako dowód rzeczowy, że wykorzystanie odzyskanych z płyt PC kontrolerów jest możliwe i że dalsze prace nie będą stratą czasu. Na początek postanowiłam zbudować prosty układzik do sterowania diodami LED, identyczny z zaprezentowanym w punkcie 6 wspomnianej strony. Pierwotnie planowałam rozpocząć pracę z emulatorem EPROM, jest to po prostu wygodniejsze od ciągłego przeprogramowywania pamięci, ale ten pomysł zarzuciłam. Powstało ryzyko, że emulator „nie dogada się" z testowym układem i

Fot. 1

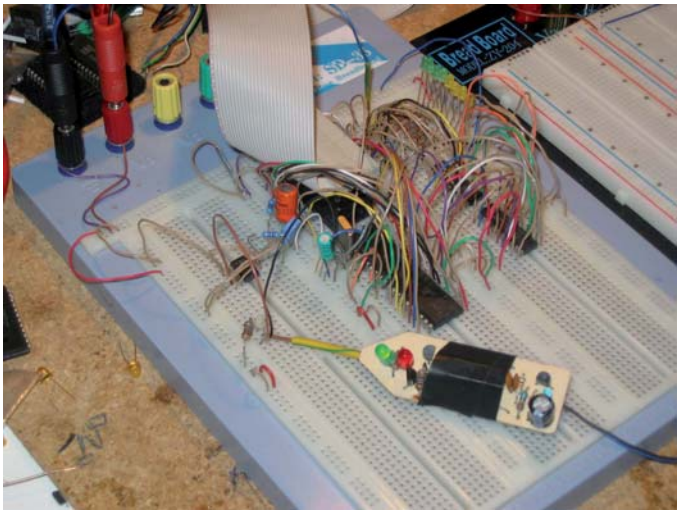


Fot. 2





Fot. 3



Fot. 4

całość nie zadziała. Testowy programik został więc zapisany do pamięci EEPROM typu 28C64, a ta została podłączona do reszty układu. No i pierwszy sukces, który nie ukrywam dodał mi skrzydeł - całość zadziałała! Pora późna już była, więc tego dnia został tylko nagrany pierwszy, mały filmik oraz powstało kilka zdjęć - ten materiał można obejrzeć w serwisie YouTube. Skoro układ zadziałał tak jak oczekiwałam, podłączyłam do kontrolera emulator pamięci EPROM i załadowałam do jego pamięci powstały z kompilacji plik binarny. No i ponownie sukces - układ uparcie pracował dalej. Aby wyjaśnić dlaczego to sukces, pozwolę sobie na małą dygresję.

Jak wiadomo, korzystanie z emulatora pamięci EPROM w przypadku układów mikroprocesorowych z zewnętrzną pamięcią programu niesamowicie poprawia komfort pracy. Jedno polecenie z konsoli tekstowej Windows i już mamy nowy wsad w pamięci emulatora, a przy okazji automatycznie zresetowany mikrokontroler. Z drugiej strony, zawsze istnieje ryzyko, że emulator akurat w tym szczególnym układzie nie będzie w stanie poprawnie emulować pamięci stałej, stąd też ten etap pośredni z prawdziwą kością pamięci EEPROM. Zwyczajnie, chciałam sobie oszczędzić ewentualnego zniechęcenia. I to podejście także polecam - jeżeli powielamy jakiś układ, co do którego mamy wątpliwości czy nie bazuje na jakichś trikach z sygnałami sterującymi pamięcią stałą (/OE, /CS, itp.) - kopię lepiej uruchomić używając rzeczywistej kostki EPROM lub EEPROM. Dopiero jak układ testowy poprawnie zadziała, spróbować na emulatorze. I właśnie na **fotografii 3** widzimy układ testowy współpracujący z emulatorem pamięci EPROM.

Inwencja twórcza (własne eksperymenty)

Następny krok, to mała reorganizacja układu na płytce stykowej - **fotografia 4**.

Pierwszy układ, który zbudowałam, zajmował zbyt wiele miejsca, a jak wiadomo - apetyt rośnie w miarę jedzenia i potrzebowałam więcej przestrzeni na jego dalszą rozbudowę. Przy okazji przebudowy zamieniłam sposób sterowania LED oraz dodałam elementy pomocne przy dalszych eksperymentach z kontrolerem, czyli generator sygnału testowego zbudowany na bazie popularnego układu NE555. Generator ten posłużył do eksperymentów ze sterowanymi zewnętrznie skokami warunkowymi, przerwaniami oraz do taktowania wbudowanego w kontroler licznika/timera.

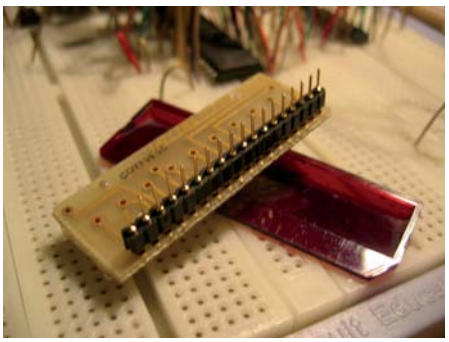
Tu chyba powinnam jedną sprawę wyjaśnić. Dostępna dokumentacja UPI-42 nie zawiera dokładnego opisu wymienionych wcześniej funkcji i bloków kontrolera. I tu właśnie widać uroki wykorzystania takich staroświeckich kostek, do których dokumentacja jest bardzo skąpa, często niekompletna i pozbawiona detali krytycznych dla procesu pisania oprogramowania. Testy z licznikiem oraz przerwaniami w 80C42 wykonałam bazując także na dokumentacji (o niebo lepszej!) kontrolera 8048, traktując sprawę następująco: skoro model ‘48 niejako wywodzi się z ‘42 to przynajmniej część podstawowych bloków procesora powinna działać podobnie. Oczywiście, trzeba to sprawdzić eksperymentalnie, ale mając na uwadze ryzyko błędnego wnioskowania wynikające z niewiedzy o innych zjawiskach, które nie są udokumentowane, a za naszymi plecami wpływają na pracę kontrolera.

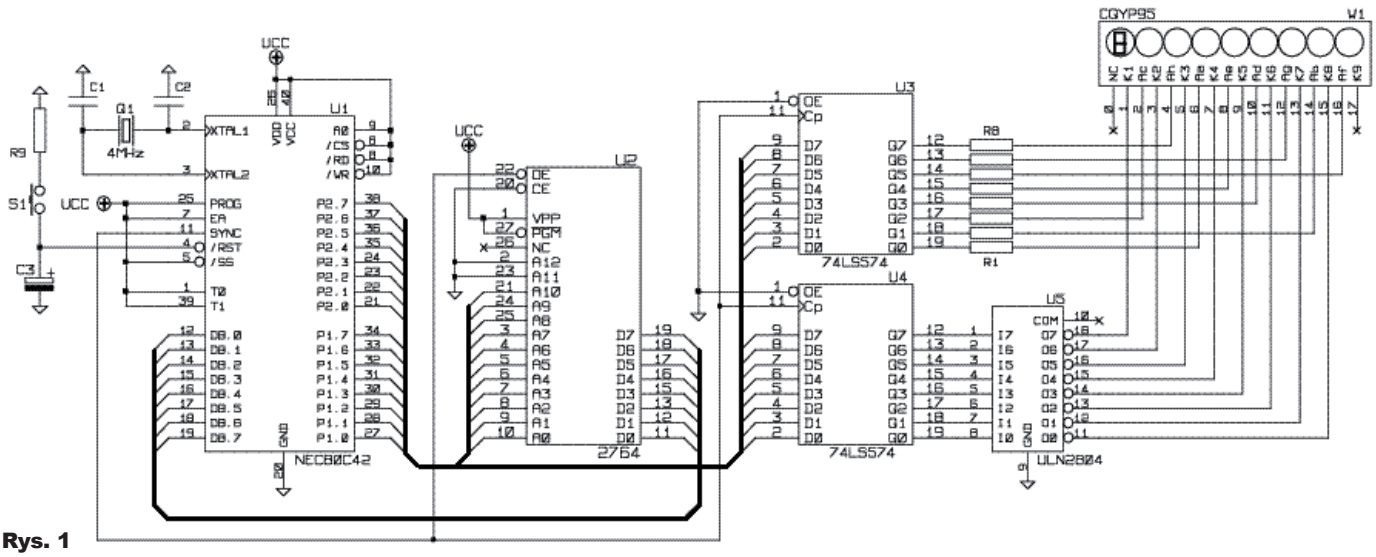
Po napisaniu kilku programów testowych i wykonaniu dokumentacji filmowej ich działania (ponownie odsyłam do YouTube), postanowiłam wykorzystać zdobyte doświadczenia do budowy czegoś bardziej skomplikowanego. Do tej pory kontroler komunikował się ze mną na migi (dosłownie: migając diodkami LED), co pomimo niewątpliwego uroku miało wiele ograniczeń. W końcu ile informacji możemy przekazać na zasadzie coś świeci/nie świeci?. Dlatego też ponownie przebudowałam swoje poletko doświadczalne i w ten sposób powstał układ do multipleksowanego sterowania wyświetlaczem siedmiosegmentowym.

Nowy, nigdy jeszcze nie używany wyświetlacz przedstawia **fotografia 5**, na **fotografii 6** ma już dolutowane złącze igłowe aby ładnie dał się połączyć z płytką stykową. Ten wyświetlacz to polski produkt o nazwie CQYP95 - dziewięć pól odczytowych, każde po siedem segmentów plus kropka dziesiętna, diody LED w układzie wspólna katoda. Element ten (podobnie jak kontroler 80C42) pochodzi z „poprzedniej

Fot. 5

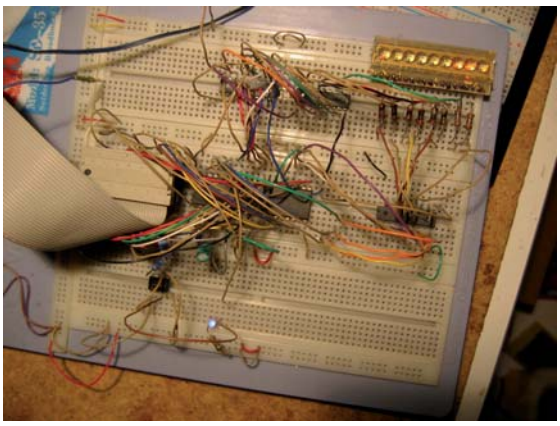
Fot. 6





Rys. 1

Fot. 7



epoki" i dlatego chyba dobrze pasuje do tego tekstu. Układ sterowania wyświetlaczem zrealizowany na płytce stykowej przestawia **fotografia 7**, a jego schemat ideowy - **rysunek 1**.

Szczegółową analizę zarówno schematu jak i powstałego oprogramowania pominę, chcę natomiast zwrócić uwagę na kilka „uroków" kontrolera 8042, które zepsuły mi humor podczas pisania programu do obsługi CQYP95.

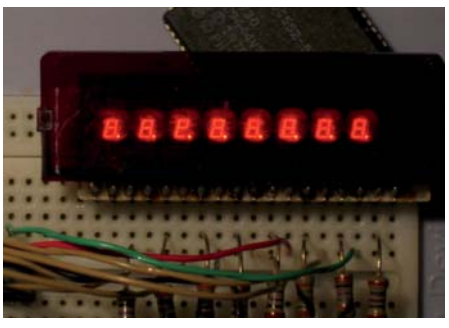
Po pierwsze - brak dostępnego dla programisty stosu. Stos, owszem jest - ale obsługiwany niejawnie przez rozkazy call/ret oraz przez wywołanie przerwania i z niego powrót rozkazem retr. O rozkazach, znanych choćby z '51, typu push, pop należy szybciutko zapomnieć. A jak wiadomo bardzo często potrzebujemy swoistej ochrony stanu rejestrów roboczych, szczególnie gdy program korzysta z przerwań. One wykonują się w tle i jeżeli obsługa przerwania zmodyfikuje nam zawartość rejestrów to program główny pójdzie w przysłowiowe maliny w dowolnie wybranym przez siebie momencie. A stos jest idealnym rozwiązaniem do chwilowego przechowywania nawet sporej ilości danych...o ile w ogóle jest. Na szczęście 8042 (podobnie jak 8048) posiada alternatywny bank ośmiu rejestrów roboczych, wyboru banku dokonuje się rozkazami sel RB0 oraz sel RB1. To jest niesamowite ułatwienie, ponieważ możemy napisać program tak, że obsługa przerwań korzysta z drugiego banku, pierwszy (podstawowy) zostanie na potrzeby programu głównego. Dodatkowo, powrót z przerwania rozkazem retr automatycznie ustawia jako aktywny ten bank, który był w użyciu w momencie zgłoszenia przerwania i skoku do procedury jego obsługi. Na naszej głowie pozostaje więc tylko ochrona akumulatora, a to jest dość łatwe - wymaga jednak poświęcenia jednego rejestru roboczego lub komórki pamięci wewnętrznej, do wyboru.

Druga, dość niemiła cecha 8042 to sposób dostępu do stałych zapisanych w pamięci programu czyli rozkaz movp A,@A. Rozkaz ten pobiera do akumulatora zawartość komórki pamięci programu o adresie wskazanym poprzednią (z chwili wykonania) zawartością tego rejestru. Akumulator jest ośmiobitowy, łatwo więc zgadnąć, że możemy rozkazem movp operować w zakresie adresów 00h...0FFh, a to nie jest zbyt dużo i trzeba czasem dobrze pogłówkować, jak w pamięci umieścić stałe i tablice, tak aby program mógł w ogóle z nich skorzystać.

Trzecia ciekawostka - brak rozkazu odejmowania (czyli sub w jakiejkolwiek postaci). No nie ma i już! Oczywiście, możemy odejmować dodając wartość ujemną i łatwo to zrealizować przy pomocy raptem trzech rozkazów. I może tak jest nawet bardziej interesująco, pytanie tylko - kiedy stracimy cierpliwość...

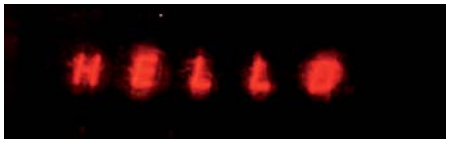
Po tej dawce narzekań - wracamy do multipleksowanego wyświetlacza CQYP95. Oprogramowanie uruchamiałam etapami, najpierw wybór katod, potem dobór optymalnej częstotliwości multipleksowania, wreszcie sekwencyjne wyświetlanie trzech napisów na wyświetlaczu. **Fotografia 8** przedstawia jeden z pierwszych etapów pracy: test wyświetlacza - czyli zapalenie wszystkich segmentów na wszystkich polach odczytowych.

Tu, z przykrością stwierdzam, display spłatał mi psikusa - jeden z segmentów trzeciej od lewej strony cyfry (segment C) - nie zaświecił. Przez dłuższą chwilę łudziłam się, że może to jakiś błąd w programie, ale niestety nie. Wystawienie samych jedynek na



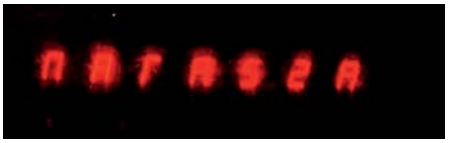
Fot. 8

Fot. 9a



9b

9c



port obsługujący anody wyświetlacza musiało wysterować wszystkie segmenty, skoro w całości świeciła poprzednia i następna cyfra - ta niekompletna okazała się jednak feralna. Szkoda trochę, ponieważ wyświetlacz wygląda naprawdę niezwykle, a cyferki pomimo że małe, są dość dobrze widoczne. Historię o CQYP95 zakończę trzema niewielkimi zdjęciami (**fotografie 9a, 9b, 9c**), na których widać statyczne napisy generowane przez testowy program.

Epilog (pomalutku kończymy)

No, jak ktoś dotarł do tego miejsca z lekturą - naprawdę gratuluję wytrwałości! Artykuł ten powinnam zakończyć jakimś spektakularnym, powalającym na kolana projektem z wykorzystaniem mikrokontrolera 80C42...ale tego nie zrobię. Szczerze mówiąc, brakuje mi pomysłu na coś rzeczywiście ciekawego, a kolejny zegarek czy mikroprocesorowe lampki choinkowe? To chyba byłoby nudne, więc na koniec tylko kilka zdań podsumowania.
Powyższy tekst to wyraźny sygnał, że przy odrobinie wysiłku stare, pozyskane ze złomu komputerowego mikrokontrolery można sensownie zagospodarować. Pamiętajmy, że mikrokontroler mamy praktycznie za darmo (płyta PC przecież i tak idzie do kosza), procesor wprawdzie nie poraża mocą obliczeniową, ale za to jest dość wdzięczny do programowania (pomimo, że lista rozkazów powoduje pewien niedosyt). Do tych, którzy zdecydują się pójść tą właśnie drogą - kilka słów, niestety w formie kubełka zimnej wody na głowę. Ponieważ we wszystkich zaprezentowanych układach procesor pracuje z zewnętrzną pamięcią programu, musimy mieć możliwość programowania pamięci EPROM/EEPROM lub posiadać emulator tychże pamięci, szczególnie typu 2764/2864. Bez tego po prostu nic nie zrobimy! Jeżeli chodzi o programator - ja do pracy korzystam z LabTool-48, ale to jest sprzęt profesjonalny, wiec niejako poza konkursem. Polecam za to zainteresować się programatorem Willem, który wspomniane wcześniej kostki potrafi programować, a można go z powodzeniem wykonać w warunkach domowych. Co do emulatorów - proszę nie posądzać mnie o kryptoreklamę, ale z tanich i skutecznych emulatorów EPROM ja osobiście polecam zestaw AVT270 (emulacja układów 2716...27512). Nie jest to droga rzecz, a posiadanie takiego emulatora otwiera zupełnie nowe możliwości. Ważne jest też to, że nie wymaga on specjalnego oprogramowania, ładowanie danych wykonuje się znanym z DOS poleceniem copy wywołanym z konsoli tekstowej Windows. Niektórzy mawiają: „jeżeli czegoś nie ma w Google, to znaczy że nie istnieje”. A właśnie na kilku stronach internetowych wskazanych przez Google znalazłam wzmiankę, że Intel opublikował (w latach siedemdziesiątych) podręcznik użytkownika do tej rodziny kontrolerów. Tytuł pozycji brzmi: „MCS-48 and UPI-41 Assembly Language Manual”. Niestety, publikacja ta jest niedostępna w formie elektronicznej, ale może przypadkiem znajduje się w zasobach uczelnianej biblioteki - polecam to sprawdzić. I to chyba wszystko...
Kończąc, ewentualnym naśladowcom życzę przede wszystkim cierpliwości i wiary w swoje siły. I nie mówię, że będzie łatwo - ale za to na pewno będzie ciekawie...

Natasza Biecek
bienata@wp.pl