Mgr inż. Tomasz ŻABIŃSKI
Dr inż. Ryszard LENIOWSKI
Politechnika Rzeszowska

PROGRAMOWO - SPRZĘTOWY STEROWNIK RUCHU Z UKŁADAMI FIRMY PMD

Streszczenie: Przedstawiono system sterowania ruchu oparty na specjalizowanym kontrolerze ruchu MC1401A firmy PMD. Zaproponowano wielowątkową strukturę programu nadrzędnego pozwalającego na sterowanie w układzie otwartym oraz realizację ustalonych trajektorii dla etapu pozycjonowania i nadążania. Zaprezentowany system nadaje się do sterowania typowych robotów oraz prostych obrabiarek CNC.

Motion Control System with PMD chipset

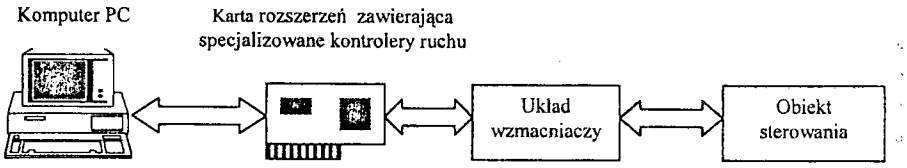
Abstract: Motion Control System for typical industrial robots and simple CNC machines has been presented in this paper. This system uses Precision Motion Controller MC1401A produced by Performance Motion Devices, Inc. Hardware configuration and the skeleton of software of the system have been demonstrated. The multi-threaded software of the system has been written in the language C for Windows 95.

1. WSTĘP

Pojawienie się na współczesnym rynku specjalizowanych kontrolerów ruchu, wywołało istotne zmiany w procesie projektowania serwomechanizmów dla typowych robotów przemysłowych i prostych obrabiarek CNC. Standardowe podejście do projektowania systemów sterowania opartych na wspomnianych układach polega na wykorzystaniu komputera PC jako jednostki MASTER, dla której kontroler ruchu sprawuje funkcję SLAVE realizując sterowanie najniższego poziomu. Wyżej wspomniana struktura systemu sterowania sprawia, że proces jego projektowania staje się zdecydowanie prostszy i szybszy niż w przypadku zastosowania tradycyjnych układów niskiej skali integracji.

Jednym z pierwszych specjalizowanych kontrolerów ruchu, powszechnie dostępnych na rynku w ostatnich latach, był układ LM628/629 firmy National Semiconductor [1]. Układ ten zapewniał standardowe funkcje sterowania ruchu i umożliwiał współpracę z jednym silnikiem DC. Aktualnie na rynku dostępna jest szeroka oferta specjalizowanych kontrolerów ruchu, między innymi firmy Performance Motion Devices (PMD), obejmująca układy serii MC1401 przeznaczone do współpracy z silnikami DC, MC1231 przeznaczone do współpracy z silnikami BLM, MC1241/MC1451 przeznaczone do obsługi silników krokowych oraz nowe układy serii MC2100, MC2300, MC2400, w których wprowadzono udoskonalenia sprzętowe przy zachowaniu zgodności programowej z seriami MC1xxx. Na podkreślenie zasługuje fakt, iż większość mikrokontrolerów firmy PMD produkowana jest w wersjach, które pozwalają na sterowanie przy pomocy jednego układu systemami wyposażonymi w 1, 2 bądź 4 silniki.

System sterowania ruchu oparty na specjalizowanych kontrolerach ruchu, w typowej konfiguracji, składa się z komputera PC zawierającego kartę rozszerzeń z wbudowanym mikrokontrolerem, układu wzmacniaczy oraz obiektu sterowania (rys. 1).

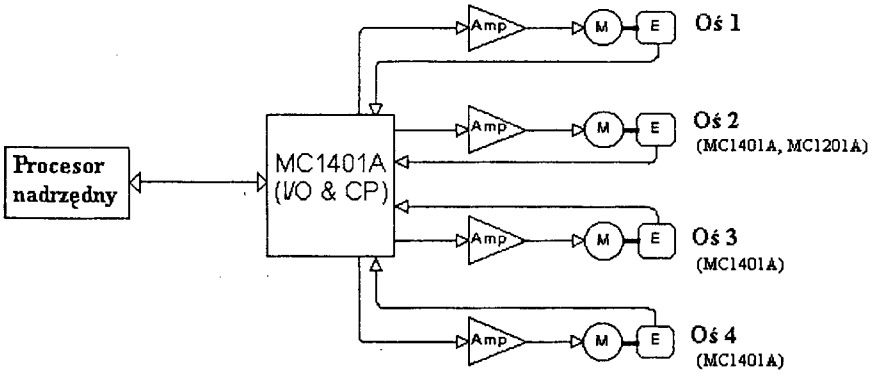


Rys. 1. Typowa konfiguracja systemu sterowania ruchu opartego na specjalizowanych kontrolerach ruchu

Celem badań prowadzonych przez autorów jest opracowanie uniwersalnego, wielowymiarowego systemu sterowania opartego na układzie MC1401A przeznaczonego dla typowych robotów przemysłowych (6 osi) i obrabiarek CNC. Artykuł prezentuje wyniki badań dotyczących opracowania warstwy komunikacyjnej pomiędzy sterownikiem osi a komputerem nadrzędnym, sterowania w układzie otwartym oraz realizacji ustalonych trajektorii dla etapu pozycjonowania i nadążania. Systemem operacyjnym wybranym dla komputera nadrzędnego jest Windows 95, gdyż zapowiadany system czasu rzeczywistego Windows CE nie jest jeszcze ustabilizowany. Wybór systemu Windows 95 podyktowany był jego dostępnością, powszechnością, niską ceną oraz wbudowanym graficznym interfejsem użytkownika. Windows 95 nie jest jednak systemem „czasu rzeczywistego", tak więc zastosowanie go do celów sterowania wymagało rozwiązania problemu precyzyjnego odmierzania czasu.

2. CHARAKTERYSTYKA UKŁADU MC1401A

MC1401A jest specjalizowanym kontrolerem ruchu przeznaczonym do budowy systemów sterowania ruchu opartych na silnikach DC. Typowa konfiguracja [2] w jakiej pracuje MC1401A przedstawiona jest na rys.2.

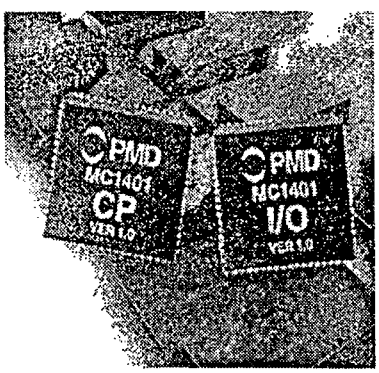


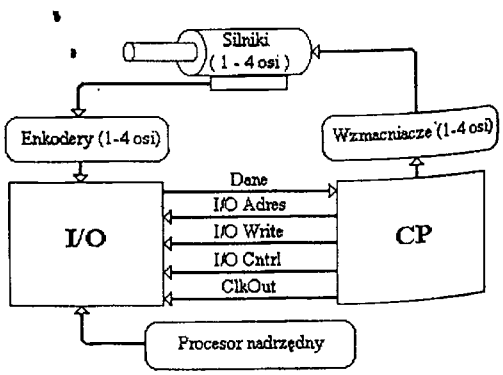
Rys. 2. Typowa konfiguracja systemu sterowania z układem MC1401A

Cechy układu MC1401A:

- dostępny w konfiguracji obsługującej 1, 2 bądź 4 osie;
- 32-bitowe rejestry prędkości, przyśpieszenia i szarpnięcia (ang. jerk);
- wejścia dla czujników krańcowych (po dwa na każdą oś);
- dostępne trajektorie: krzywa-S, trapezoidalna, prędkościowa;
- programowalny czas cyklu (od 100 μs);
- możliwość pracy w trybie elektronicznych kół zębatych;
- regulator PID z członem realizującym prędkościowe sterowanie antycypacyjne (ang. velocity feedforward);
- wyjścia PWM bądź DAC;
- szeregowe albo równoległe wyjście DAC;
- wejście dla sygnałów enkoderów, czujników krańcowych oraz pozycji bazowej (zakres pracy 1 MHz);
- automatyczne wyłączenie silnika w przypadku wystąpienia błędu podczas ruchu;
- łatwy w użyciu protokół komunikacji z procesorem nadrzędnym (możliwość sprawdzania sumy kontrolnej);
- możliwość programowania przerwań dla procesora nadrzędnego;
- możliwość zmiany większości parametrów ruchu oraz nastaw regulatorów „w locie";
- mechanizm synchronizacji osi;
- 96 instrukcji programujących;

Fizycznie MC1401A składa się z dwóch układów wykonanych w technologii CMOS, zasilanych napięciem 5 V i umieszczonych w dwóch obudowach posiadających po 68 nóżek (rys. 3). Pierwszy z układów nazywany układem We/Wy (ang. I/O chip) jest odpowiedzialny za komunikację z procesorem nadrzędnym oraz zawiera wejścia dla danych pochodzących z enkoderów. Drugi układ nazywany procesorem komend (ang. Command Procesor chip - CP) stanowi główny moduł obliczeniowy odpowiedzialny za wykonywanie komend zlecanych przez procesor nadrzędny, obliczanie trajektorii, wyjść regulatorów PID oraz wyjść PWM bądź DAC. Odpowiednią szybkość pracy MC1401A zapewnia procesor sygnałowy wchodzący w skład jednostki obliczeniowej.





Rys.3. Układ MC1401A

3. PROGRAMOWY INTERFEJS KOMUNIKACYJNY

Programowanie ruchu za pomocą MC1401A odbywa się z poziomu procesora nadrzędnego, który wysyła do układu sekwencje komend (numery komend) oraz danych ustalających między innymi tryb pracy układu (typ trajektorii), wartości prędkości, przyspieszeń i pozycji docelowych. Po otrzymaniu wymaganego, dla danego typu trajektorii, zestawu danych MC1401A samodzielnie realizuje zadaną trajektorię i po jej zakończeniu informuje procesor nadrzędny o zakończeniu zadania. Komunikacja pomiędzy układem a procesorem nadrzędnym odbywa się poprzez 8-bitowy port komunikacyjny, który służy do przesyłania numerów komend oraz do odczytu bądź zapisu danych.

Przykładowa sekwencja komend pozwalająca na zrealizowanie trajektorii trapezoidalnej :

```
SET_1                          // wybór osi
SET_PRFL_TRAP                  // ustalenie typu trajektorii
SET_POS        0x2000          // załadowanie wartości pozycji docelowej
SET_VEL        0x10000         // załadowanie wartości prędkości
SET_ACC        0x10000         // załadowanie wartości przyspieszenia
UPDATE                         // rozpoczęcie ruchu
```

Z poziomu systemu nadrzędnego układ MC1401A widziany jest jako zestaw trzech rejestrów. Odpowiadają temu porty o adresach: 340h – DATA, 341h – COMM, 344 – BUSY. Programowy interfejs komunikacyjny wykorzystywany podczas programowania MC1401A składa się z 5 podstawowych funkcji, które pozwalają na łatwe realizowanie wszystkich komend obsługiwanych przez układ. Funkcje te wyposażone są dodatkowo w mechanizm sprawdzania sumy kontrolnej dla przesyłanych danych, a w przypadku pojawienia się błędów informują program sterujący o wystąpieniu sytuacji awaryjnej.

Częściowy kod funkcji wchodzącej w skład programowego interfejsu komunikacyjnego, obsługującej komendy wymagające przesłania do układu 16 bitów danych przedstawiono poniżej.

```
void C1Write(UCHAR komenda, USHORT var)
{
    char i=0;
 do
 {
  ReadBusy();                              // makro sprawdzające bit BUSY
  outportb(COMM,komenda);
  ReadBusy();
  outportb(DATA,(UCHAR)((var & 0xff00)>>8));
  outportb(DATA,(UCHAR)(var & 0x00ff));
  ReadBusy();
  i++;
                                           // CheckSum – makro sprawdzające sumę kontrolną
 } while (CheckSum()!=(USHORT)(var + komenda) && i!=IloscPowtorzen);

                                           // reakcja na blad sumy kontrolnej
  if (i==IloscPowtorzen)
    {

    }
}
```

Przykładowa sekwencja komend pozwalająca na zrealizowanie trajektorii trapezoidalnej przy pomocy programowego interfejsu komunikacyjnego jest następująca:

```
C1Read(SET_1);                  // wybór osi
C0(SET_PRFL_TRAP);              // ustalenie typu trajektorii
C2Write(SET_POS,0x20000);       // załadowanie wartości pozycji docelowej
C2Write(SET_VEL,0x10000);       // załadowanie wartości prędkości
C2Write(SET_ACC,0x10000);       // załadowanie wartości przyspieszenia
C0(UPDATE);                     // rozpoczęcie ruchu
```

Jak widać zestaw instrukcji realizujących trajektorię trapezoidalną jest bardzo prosty i czytelny, ogranicza się do podania rodzaju ruchu i jego trzech głównych parametrów.

4. PROGRAMOWA WARSTWA NADRZĘDNA STEROWNIKA RUCHU

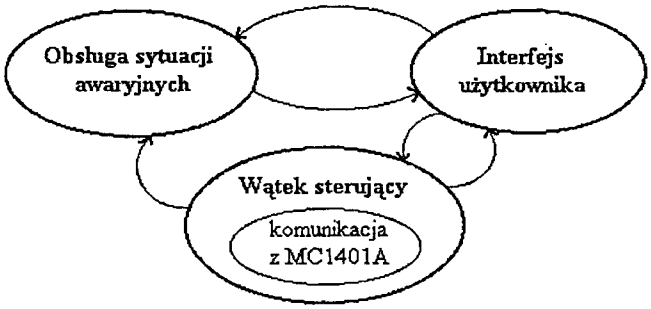
Program stanowiący warstwę nadrzędną dla układu MC1401A został napisany w języku C dla systemu Windows 95. Jego struktura oparta jest na architekturze wielowątkowej, która pozwala na zaimplementowanie trzech głównych modułów systemu czyli: interfejsu użytkownika (okno główne aplikacji), wątku sterującego oraz modułu obsługi sytuacji awaryjnych (rys.4).

Okno główne aplikacji spełnia funkcję interfejsu użytkownika pozwalając mu na wybór funkcji testującej, jej parametrów i przeprowadzenie eksperymentu.

Wątek sterujący zostaje uruchomiony przez użytkownika z poziomu okna głównego aplikacji. Ze względu na fakt, iż eksperyment wymaga często precyzyjnego odmierzania czasu wątek sterujący posiada najwyższy priorytet jaki udostępnia system Windows 95 [3, 4] a więc: REALTIME_PRIORITY_CLASS oraz THREAD_PRIORITY_TIME_CRITICAL.

Okno obsługi sytuacji awaryjnych zabezpiecza urządzenie fizyczne przed uszkodzeniem oraz informuje użytkownika o ewentualnych błędach systemu. Aktualnie obsługiwane są następujące sytuacje awaryjne:

- błąd sumy kontrolnej dla komendy – użytkownik jest informowany przy pomocy okna dialogowego o nazwie komendy dla której pojawił się błąd oraz o numerze osi przy, której programowaniu wystąpił;
- błąd testu komunikacji z kartą, który wykonywany jest w momencie uruchomienia programu;
- błąd resetu układu, który użytkownik może wykonać z poziomu okna głównego aplikacji;
- błąd przekroczenia zakresu przestrzeni roboczej sygnalizowany przez czujniki hallotronowe umieszczone na urządzeniu – obsługa omawianej sytuacji awaryjnej jest aktywna w czasie realizowania trajektorii, w przypadku jej wystąpienia w danej osi wyłączany jest silnik oraz zakończone zostają wszystkie aktualnie aktywne wątki sterujące.



Rys. 4. Wielowątkowa architektura programu nadrzędnego

Aby rozwiązać problem precyzyjnego odliczania czasu, opracowano i przetestowano trzy zegary:

- oparty na zegarze Windows 95 (aktualny czas pobierany jest przy pomocy funkcji timeGetTime() udostępniającej zegar o dokładności 1 ms);
- oparty na zegarze układu MC1401A (czas odczytywany jest z układu przy pomocy komendy GET_TIME, dokładność zegara 0.1 ms);
- mieszany, w którym występuje połączenie wcześniejszych dwóch typów zegarów (główny czas cyklu jest wyznaczony przez zegar Windows95 zaś dokładny czas pobierany jest z układu).

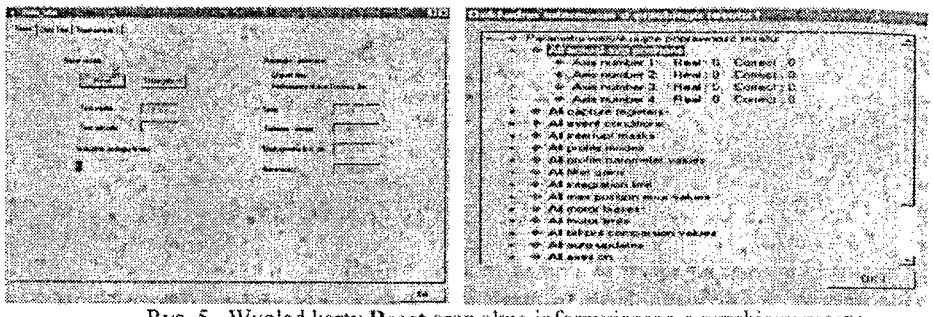
Na podstawie przeprowadzonych testów uznano, iż najbardziej wiarygodnym rozwiązaniem jest zegar oparty na układzie MC1401A ze względu na jego większą dokładność oraz niezależność od systemu Windows 95. Jednak sama struktura tego rozwiązania nie pozwala na uzyskanie czasu cyklu mniejszego od 1.5 ms, ze względu na konieczność komunikacji z układem. Zegar mieszany pozwala na osiągnięcie minimalnego czasu cyklu wynoszącego 1.1 ms ale kosztem uzależnienia głównej pętli zegara od systemu Windows 95. Zegar oparty tylko i wyłącznie na systemie Windows 95 jest rozwiązaniem, które może służyć do akwizycji danych jednak nie jest zalecane do wyznaczania czasu cyklu sterowania.

Interfejs użytkownika stanowi arkusz właściwości składający się z trzech kart o nazwach: **Reset, Chirp Test** oraz **Trajektoria.**

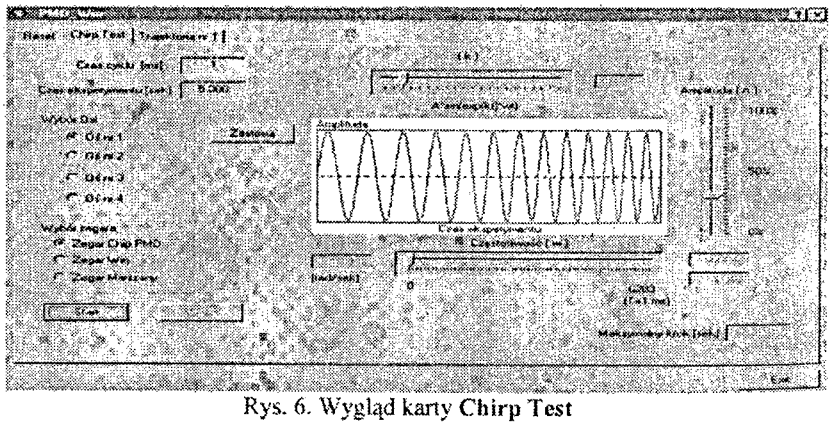
Karta **Reset** pozwala użytkownikowi na przeprowadzenie resetu układu i sprawdzenie jego poprawności. Jeżeli reset nie przebiegnie poprawnie przycisk *Szczegóły* udostępnia dokładne informacje dotyczące błędów. Karta udostępnia również informacje, pobierane z układu, dotyczące serii oraz wersji układu. Wygląd karty **Reset** oraz okna informującego o szczegółach resetu przedstawia rys. 5.

Karta **Chirp Test** pozwala na przeprowadzenie sterowania w układzie otwartym. Sygnałem sterującym jest sygnał typu chirp o przebiegu opisanym wzorem: $A\sin(e^{kt}\cdot\omega t)$. Obiekt sterowania stanowi wzmacniacz oraz silnik wraz z suportem. Użytkownik ma możliwość wyboru osi, dla której przeprowadzony zostanie test, ustalenia parametrów sygnału testującego oraz parametrów eksperymentu takich jak: czas trwania eksperymentu oraz czas cyklu generatora sygnału sterującego. Możliwy jest również wybór jednego z zegarów służących do odliczania czasu cyklu i omówionych w punkcie 4 artykułu. Rezultatem przeprowadzonego eksperymentu jest zestaw danych pomiarowych zgromadzonych w plikach

zawierających wartości położenia osi silnika oraz prędkości obliczonej poprzez numeryczne różniczkowanie sygnału położenia. Użytkownik ma również możliwość wykreślenia przebiegów uzyskanych podczas eksperymentu przy pomocy modułu graficznej reprezentacji danych sprzężonego z kartą **Chirp Test.** Wygląd karty **Chirp test** przedstawia rys. 6.



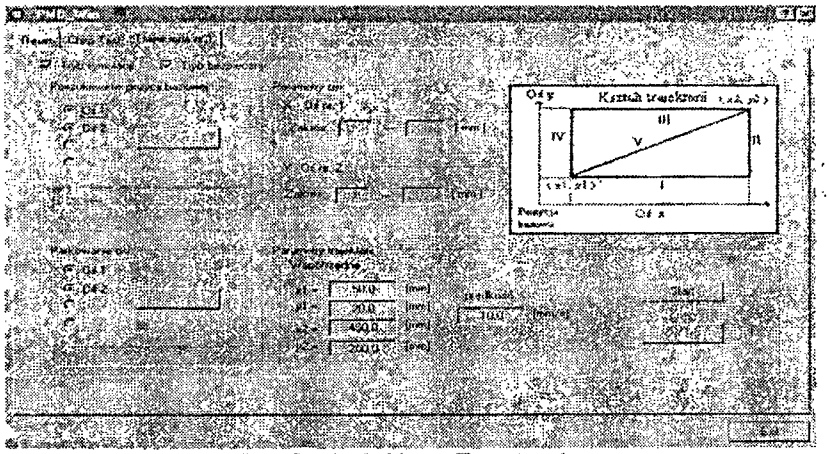
Rys. 5. Wygląd karty **Reset** oraz okna informującego o przebiegu resetu



Rys. 6. Wygląd karty **Chirp Test**

Karta **Trajektoria** pozwala na zrealizowanie trajektorii złożonej z pięciu prostych, wykonywanej fizycznie przez osie nr 1 i 2 (rys.7). Karta pozwala na modyfikację współrzędnych wierzchołków opisujących trajektorię oraz prędkości z jaką ma się odbywać ruch. Dodatkowe funkcje jakie udostępnia **Trajektoria** to możliwość pracy w *Trybie symulacji*, który pozwala na projektowanie trajektorii teoretycznej (generowanej przez układ) bez potrzeby rzeczywistego uruchamiania osi. Dzięki temu trybowi możliwe jest uniknięcie niebezpiecznych błędów, które mogą pojawić się w przypadku testowania trajektorii bezpośrednio na urządzeniu fizycznym. Drugim trybem w jakim może pracować program to tzw. *Tryb bezpieczny* w którym aktywne są zabezpieczenia wyłączające silniki w przypadku gdy suport przekroczy granicę przestrzeni roboczej. Karta wyposażona jest również w funkcję *Poszukiwania pozycji bazowej*, która umożliwia dokładne ustalenie pozycji odniesienia w przestrzeni roboczej. Funkcja ta realizuje kilkufazową procedurę poszukiwania czujników hallotronowych wyznaczających położenie punktu odniesienia. Ostatnią funkcją jaką udostępnia omawiana karta jest funkcja *Parkowania osi* wybranych przez użytkownika. Opcja ta pozwala na ustawienie suportów w bezpiecznym położeniu w 1/3 zakresu przestrzeni roboczej dla danej osi. Podczas wykonywania trajektorii prowadzony jest zapis danych,

pozwalający na analizę trajektorii projektowanej, teoretycznej oraz rzeczywistej realizowanej przez urządzenie. Dane te dostępne są po wykonaniu eksperymentu w zbiorze plików zapisanych w katalogu bieżącym programu. Użytkownik ma również (tak jak w przypadku karty **Chirp test**) możliwość wykreślenia trajektorii uzyskanych podczas eksperymentu. Wygląd karty **Trajektoria** przedstawia rys. 7.

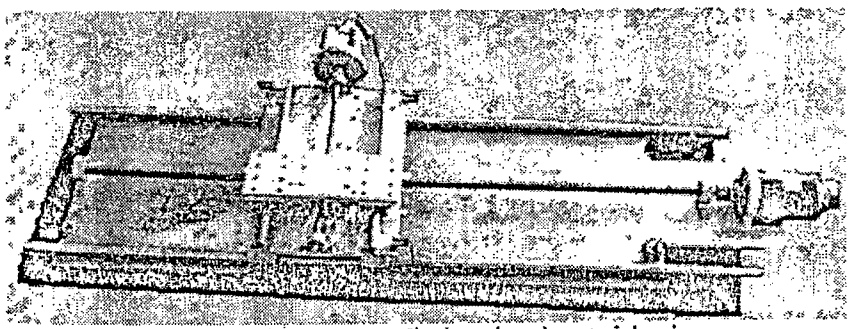


Rys. 7. Wygląd karty **Trajektoria**

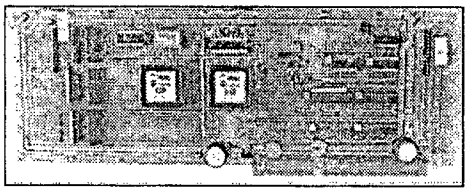
5. TESTY LABORATORYJNE

Prezentowane stanowisko laboratoryjne składa się z komputera typu PC, karty rozszerzeń zawierającej układ MC1401A firmy PMD (rys. 9), zestawu zasilaczy i wzmacniaczy, układu dwóch silników DC napędzających 2 suporty w układzie 2T (rys. 8) oraz z oprogramowania dla komputera nadrzędnego napisanego w języku C dla środowiska Windows 95.

Wszystkie testy, których przykładowe wyniki podane są poniżej przeprowadzone były dla dwóch komputerów PC: 486 100MHz, 16MB RAM, dysk IDE oraz Pentium 200MHz, 98MB RAM, dysk SCSI. Podstawową zaletą wykorzystania szybszego procesora jest bardziej stabilny czas cyklu zegarów omówionych w punkcie 4 i jego mniejsza czułość na zmiany obciążenia systemu wywoływane np. przez mysz. Jednak dolne granice czasu cyklu badanych zegarów nie uległy zmianie w wyniku zastosowania szybszego procesora, gdyż wynikają one głównie z procesu komunikacji z układem.



Rys. 8. Układ suportów realizujących zadane trajektorie



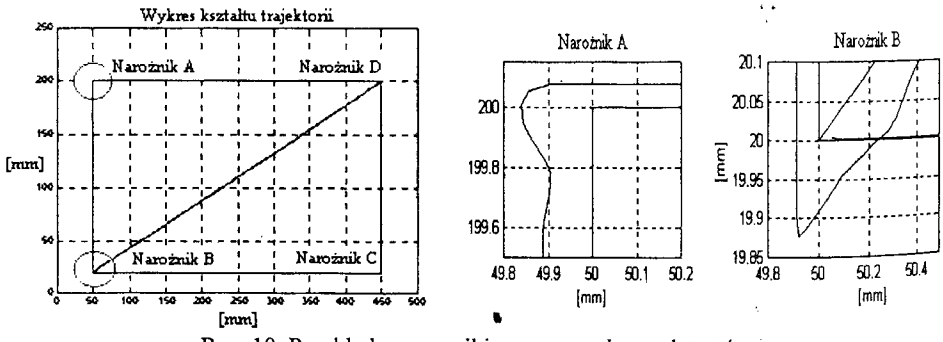
Rys. 9. Prototypowa karta rozszerzeń zawierająca układ MC1401A.

Przykładowe wyniki testów przeprowadzonych przy pomocy karty **Trajektoria** dla parametrów zawartych w tabeli 1, przedstawia rys.10 (komputer: P200MHz, 98MB RAM, dysk: SCSI):

Tabela 1. Parametry trajektorii

$x_1 = 50.0$ mm	$x_2 = 450.0$ mm
$y_1 = 20.0$ mm	$y_2 = 200.0$ mm
Prędkość: 10.0 mm/s	czas cyklu zapisu danych: 5 ms

Współrzędne (x_1, y_1) oraz (x_2, y_2) opisują współrzędne narożników, odpowiednio: B oraz D (rys. 10). Kolejność realizacji fragmentów trajektorii wyznaczają narożniki: B, C, D, A, B, D. Kolor czerwony wykresu reprezentuje trajektorię zaprojektowaną zaś niebieski trajektorię rzeczywistą realizowaną przez układ suportów.



Rys. 10. Przykładowe wyniki przeprowadzonych testów

Aby określić maksymalne błędy realizacji trajektorii przeprowadzono ponad 50 testów na podstawie których uzyskano wyniki przedstawione w tabeli 2. Eksperymenty wykonane zostały dla parametrów zawartych w tabeli 1 z wyjątkiem prędkości, której wartość była modyfikowana w celu zbadania jej wpływu na jakość odtwarzania trajektorii.

Jak widać na podstawie tabeli 2, zastosowanie szybszego komputera powoduje: wzrost odporności systemu sterowania na zmiany obciążenia systemu operacyjnego, większą powtarzalność realizacji trajektorii (zmniejszenie zakresu błędów prezentowanych w kolumnie 4 tabeli 2). Zastosowanie komputera o lepszych parametrach nie powoduje jednak jakościowych zmian w dokładności odtwarzania trajektorii. Na podstawie danych zawartych w tabeli 2, widać również, że jakość realizacji trajektorii zależy w dużym stopniu od przyjętej prędkości ruchu.

Tabela 2. Maksymalne błędy realizacji trajektorii

Parametry komputera	Prędkość realizacji trajektorii [mm/sek]	Dodatkowe obciążenie systemu przez użytkownika (mysz)	Zakres maksymalnych błędów realizacji trajektorii* [mm]
486**	10.0	Brak	0.14 – 0.17
486	10.0	Występuje	0.20 – 0.40
486	30.0	Brak	0.42 – 0.50
486	30.0	Występuje	0.60 – 0.85
P200	10.0	Brak	0.16 – 0.17
P200	10.0	Występuje	0.14 – 0.19
P200	30.0	Brak	0.46 – 0.47
P200	30.0	Występuje	0.46 – 0.50

6. PODSUMOWANIE

W pracy scharakteryzowano specjalizowany kontroler ruchu MC1401A firmy PMD oraz system sterowania ruchu oparty na wspomnianym układzie. Przedstawiono oprogramowanie warstwy komunikacyjnej pomiędzy układem a komputerem nadrzędnym. Zaprezentowano programowe narzędzia pozwalające na sterowanie obiektem laboratoryjnym w układzie otwartym oraz realizację ustalonych trajektorii dla etapu pozycjonowania i nadążania. Zaproponowano wielowątkową strukturę programu nadrzędnego oraz przedstawiono wyniki przeprowadzonych eksperymentów. Podkreślić należy jednak, że obecna wersja systemu sterowania jest uważana za prototypową. System wymaga między innymi opracowania algorytmów projektowania trajektorii, które uniezależnią jakość realizacji trajektorii od założonej prędkości ruchu oraz uściślenia związków pomiędzy poszczególnymi wątkami programu nadrzędnego.

LITERATURA

[1] Witold Brzozowski: *Programowanie specjalizowanych kontrolerów ruchu firmy National Semiconductor*; Praca dyplomowa, Politechnika Rzeszowska 1998.

[2] PMD, *Advanced Multi-Axis Motion Control Chipset*, February, 1996.

[3] F. Houlette, W.S. Holderby, D.R. Black, S. Dillon, C.J. Lewis, J.W. Nelsen, D. Shack, D. Smith: *Windows 95 od wewnątrz. Przewodnik programisty*; Oficyna wydawnicza LT&P, Warszawa 1997.

[4] MSDN Library, Microsoft, April 1999.

* Wartości błędów są przeliczone na podstawie danych pochodzących z enkodera umieszczonego po stronie silnika.

** Dokładne dane o konfiguracji komputerów patrz wstęp do punktu 5 artykułu.