PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

Ośrodek Automatyzacji
Procesów Produkcji
Zespół Sprzętu Cyfrowego

074 A

Główny wykonawca

Wykonawcy dr inż. Bohdan Kontrymowicz
mgr inż. Dariusz Okrasa
mgr inż. Andrzej Zasucha

Konsultant

Nr zlecenia RP-61

System wizyjny dla robotów IRp.

Zadanie nr 1.2.1

Projekt sprzętu i oprogramowania systemu wizyjnego 2-D i badanie algorytmów przetwarzania wizji.

Zleceniodawca CPBR 7.1 "Roboty przemysłowe"

Pracę rozpoczęto dnia 1987.10.01 **zakończono dnia** 1988.04.30

Kierownik Zespołu — dr inż. B.Kontrymowicz

Z-ca Dyrektora d/s Automatyki — dr inż. T.Gałązka

Kierownik Ośrodka — mgr inż. A.Aderek

Praca zawiera:

stron	26
rysunków	
fotografii	
tabel	
tablic	
załączników	10
Nr rejestr.	6035

Rozdzielnik - ilość egz:

Egz. 1	BOINTE
Egz. 2	OAP
Egz. 3	OAR
Egz. 4	IAP PW
Egz. 5	
Egz. 6	

Analiza deskryptorowa

Systemy wizyjne
Analiza sceny
Roboty przemysłowe
Rozpoznawanie obrazów
Struktura systemów wizyjnych

Analiza dokumentacyjna

W sprawozdaniu przedstawiono projekt sprzętu na system wizyjny 2-D dla robotów przemysłowych IRp. Zaprojektowano oprogramowanie systemu na podstawie wybranych wcześniej algorytmów rozpoznawania obrazów i określania ich orientacji. Przeprowadzono badania algorytmów wstępnej filtracji obrazu.

Tytuły poprzednich sprawozdań

Zadanie 1.1 Prace studialne.
Nr rejestr. 5836
Zadanie 1.2 Opracowanie założeń i wybór algorytmów.
Nr rejestr. 5913

UKD 338.45:62/69].002.1.2 Roboty przemysłowe

PIAP-252/83-6000

Spis treści

1.)KONSTRUKCJA SYSTEMU WIZYJNEGO 2-D DLA ROBOTA PRZEMYSLOWEGO.

System wizyjny 2-D dla robota przemysłowego z kamerą matrycową 256x256 punktów składa się z trzech pakietów o wymiarach podwójnej eurokarty, które komunikować się będą poprzez część rezydentną magistrali Inteldigit-Proway. System wykorzystuje jeden standardowy pakiet mikroprocesorowego systemu automatyki kompleksowej Inteldigit-Proway: pakiet MM16 z procesorem 16-bitowym 8086, który będzie pełnił w systemie rolę procesora nadrzędnego. Pakiet MM16 zawiera 16Kx8 bitów pamięci danych RAM oraz 32Kx8 bitów pamięci programu EPROM. Wystarcza to na potrzeby programów wykorzystywanych przez system, tak że nie jest konieczny dodatkowy pakiet pamięci.

Pozostałe pakiety systemu to: pakiet binaryzacji PB-21 oraz pakiet preprocesora PP-21.

1) PAKIET BINARYZACJI.

Funkcje pakietu:

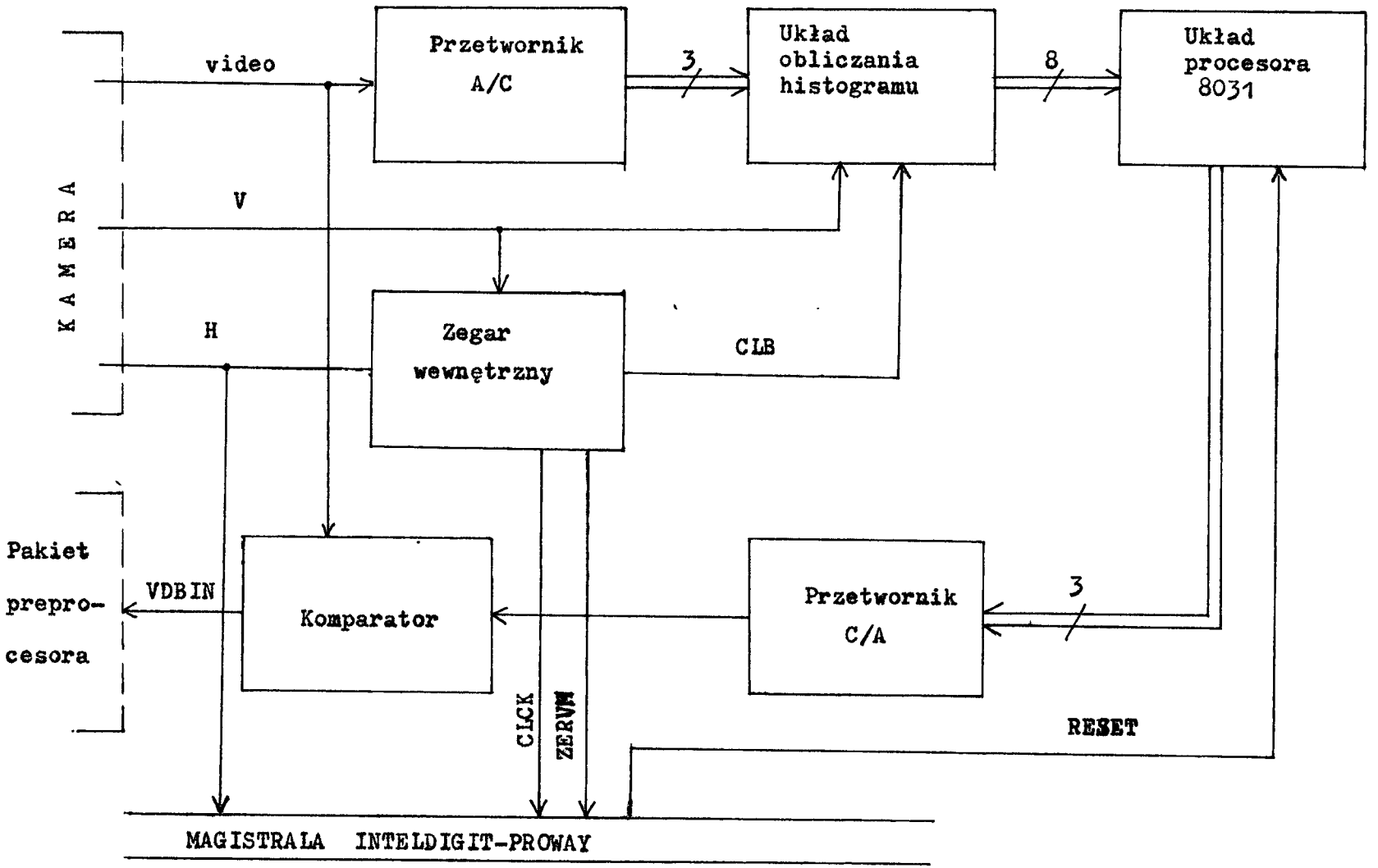
- obliczanie histogramu w 8 poziomach szarości - dopuszcza się uzasadnione przybliżanie wyników,
- binaryzacja obrazu na podstawie obliczonego histogramu.

Schemat blokowy pakietu binaryzacji przedstawiony jest na rysunku 1.

Opis poszczególnych bloków pakietu binaryzacji.

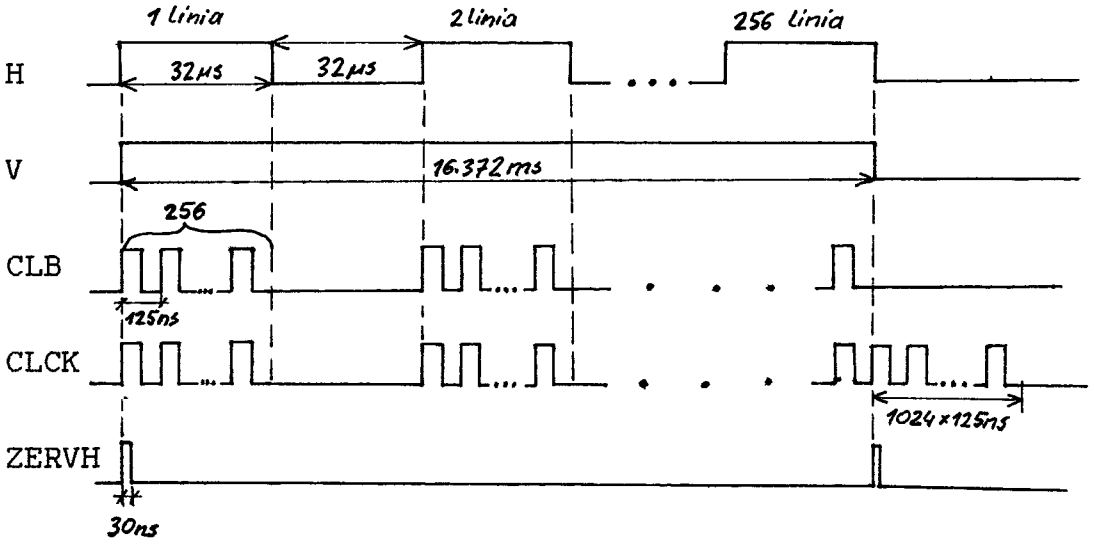
1.1)Zegar wewnętrzny.

Główną część układu stanowi generator fali prostokątnej zbudowany na rezonatorze kwarcowym 8 MHz (układ Q2). Generator jest wyzwalany sygnałem synchronizacji linii H z kamery



Rys. 1. Schemat blokowy pakietu binaryzacji.

na czas 32μs dla każdej linii obrazu. Pozwala to na wygenerowanie 256 taktów zegarowych o czasie trwania 125 ns. Licznik U23 zezwala na rozpoczęcie zliczania przez układ liczników U24 i U25 (tworzących licznik modulo 1024) od końca 256 linii obrazu. Dodatkowe 1024 takty zegara konieczne są do wysunięcia wszystkich punktów obrazu z rejestrów przesuwających linii opóźniających pakietu preprocesora (punkt 2.1). Sygnał ZERVH zeruje liczniki na początku każdego obrazu.



Rys.2. Postać sygnałów wykorzystywanych i generowanych w układzie zegara wewnętrznego.

1.2)Układ obliczania histogramu.

Zadaniem układu jest zliczenie wszystkich punktów o takim samym poziomie szarości. Układ składa się z demultipleksera '155 (U22), 8 liczników '193 (U13-U20) oraz 3 liczników Intel 8253 (U10÷U12).

Na wejścia adresowe demultipleksera 8-wyjściowego U22 podawane są 3-bitowe wartości odpowiadające poziomowi szarości danego punktu. Na wejścia strobujące podawany jest sygnał CLK. Każde z 8 wyjść demultipleksera odpowiada za punkty o tym samym poziomie szarości. Wyjścia te są podane na wejścia zliczające 8 liczników 93 liczących modulo 4. Jednak rzeczywiste zliczanie punktów obrazu o tym samym poziomie szarości odbywa się za pomocą układów licznikowych 8253 (U10÷U12). Układy te zawierają 3 liczniki 16-bitowe, co pozwala na zliczenie do

64K punktów dla każdego z 8 poziomów szarości. Liczniki U13-U20 zostały wprowadzone w celu zredukowania częstotliwości impulsów podawanych na liczniki 8253, które pracują poprawnie do częstotliwości 2 MHz. W rezultacie punkty są zliczane z dokładnością do 4. Dla normalnych obrazów - dla 64K punktów - nie wprowadza to istotnych zniekształceń wyników. Liczniki 8253 pracują w trybie 4; programowane są przez procesor 8031, który również odczytuje ich zawartości po każdym obrazie w czasie niskiego poziomu sygnału synchronizacji ramki V. Koniec zliczania (koniec obrazu) jest sygnalizowany procesorowi poprzez przerzutnik U21, którego wyjście nieodwracające podane jest na wejście przerwań procesora INTØ. Narastające zbocze sygnału V (początek obrazu) zeruje liczniki '93, a poziom wysoki tego sygnału odblokowuje zliczanie liczników 8253.

1.3) Procesor 8031.

Pakiet binaryzacji zawiera procesor jednoukładowy 8031 z pamięcią programu EPROM (U2) 2 lub 4Kx8 bitów i pamięcią danych RAM (U3) 2 lub 4Kx8 bitów taktowany własnym zegarem. Linie adresowe pamięci sterowane są poprzez bufor 8282 (U4).

Zadaniem mikrokontrolera 8031 jest zaprogramowanie układów licznikowych 8253 oraz odczytanie ich zawartości po zakończeniu zliczania punktów danego obrazu, a następnie określenie wartości progu komparacji na podstawie rozkładu jasności obrazu. Procedura obliczająca próg komparacji inicjowana jest przerwaniem INTØ. Wartość progu komparacji podawana jest do rejestru typu "latch" '75 (U6), który utrzymuje jego wartość do obliczenia nowego poziomu komparacji dla następnego obrazu.

1.4) Przetwornik C/A.

Pakiet wykorzystuje 10-bitowy przetwornik C/A produkcji radzieckiej K572PA1A (U7) o czasie przetwarzania 500ns, który jest w pełni zgodny z przetwornikiem AD7520 firmy Analog Devices. Układ jest wykonany w technologii CMOS, ale jest kompatybilny z układami TTL. Wartość progu komparacji z prze-

rzutnika U6 steruje 3 najstarsze bity wejść przetwornika, tak że jego zakres napięć wyjściowych jest podzielony na 8 poziomów. Wyjścia cyfrowe przetwornika podawane są do wzmacniacza operacyjnego AD741K. Dopasowanie poziomu napięcia wyjściowego wzmacniacza operacyjnego odpowiadającemu danemu progowi komparacji do zakresu napięcia Ø÷1V sygnału video dokonywane jest poprzez rezystor R=Ø÷5ØØ dołączony do napięcia V_{REF}.

1.5)Komparator.

Wykorzystywany jest szybki komparator napięciowy ULY771ØNA (U9). Ma on dwa wejścia różnicowe i pojedyncze wyjście, którego poziomy napięcia dostosowane są do wymagań układów cyfrowych. Wejście nieodwracające sterowane jest napięciem odpowiadającym danemu progowi komparacji, natomiast wejście odwracające sygnałem video z wtórnika emiterowego zbudowanego na tranzystorze T1. Zbinaryzowany obraz podawany jest kablem koncentrycznym poprzez złącze BNC 5Ø/N1 na pakiet preprocesora.

1.6)Przetwornik A/C.

Zastosowano 6-bitowy przetwornik A/C produkcji radzieckiej K11Ø7PW1 (U32) pracujący metodą bezpośredniego przetwarzania równoległego ("flash"). Jego czas przetwarzania wynosi 40 ns, co jest w zupełności wystarczające dla sygnału video o częstotliwości 8 MHz. Układ dopasowujący przetwarza poziom sygnału video do poziomu wymaganego przez przetwornik.

2)PAKIET PREPROCESORA.

Funkcje pakietu:

- filtracja zbinaryzowanego sygnału wizyjnego za pomocą maksymalnie trzech różnych filtracji,
- wydzielenie z obrazu konturu detalu,
- zapisanie konturu do pamięci RAM,
- obliczanie pola detalu,

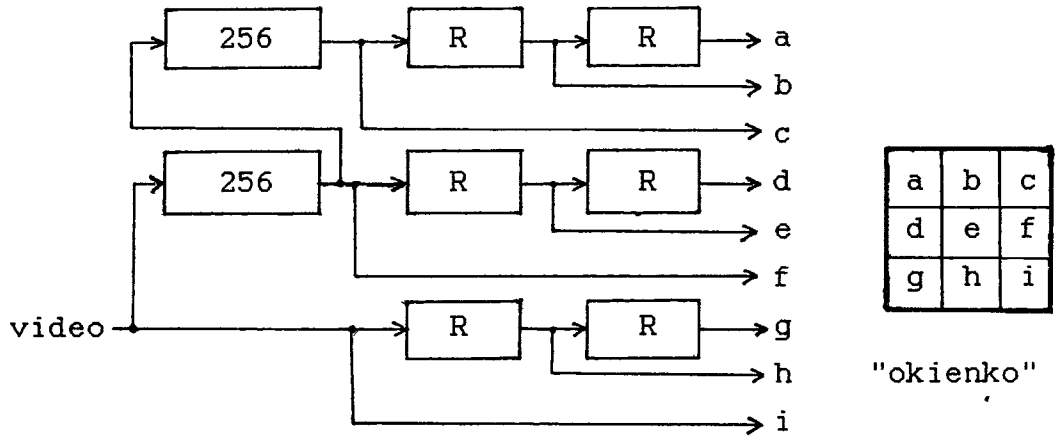
- obliczanie obwodu detalu,
- obliczanie współrzędnych S1 i S2 potrzebnych do obliczenia położenia środka ciężkości,
- zapewnienie dostępu do obliczonych danych poprzez bufory magistrali rezydentnej.

Schemat blokowy pakietu preprocesora przedstawiony jest na rysunku 3.

Opis poszczególnych bloków pakietu preprocesora.

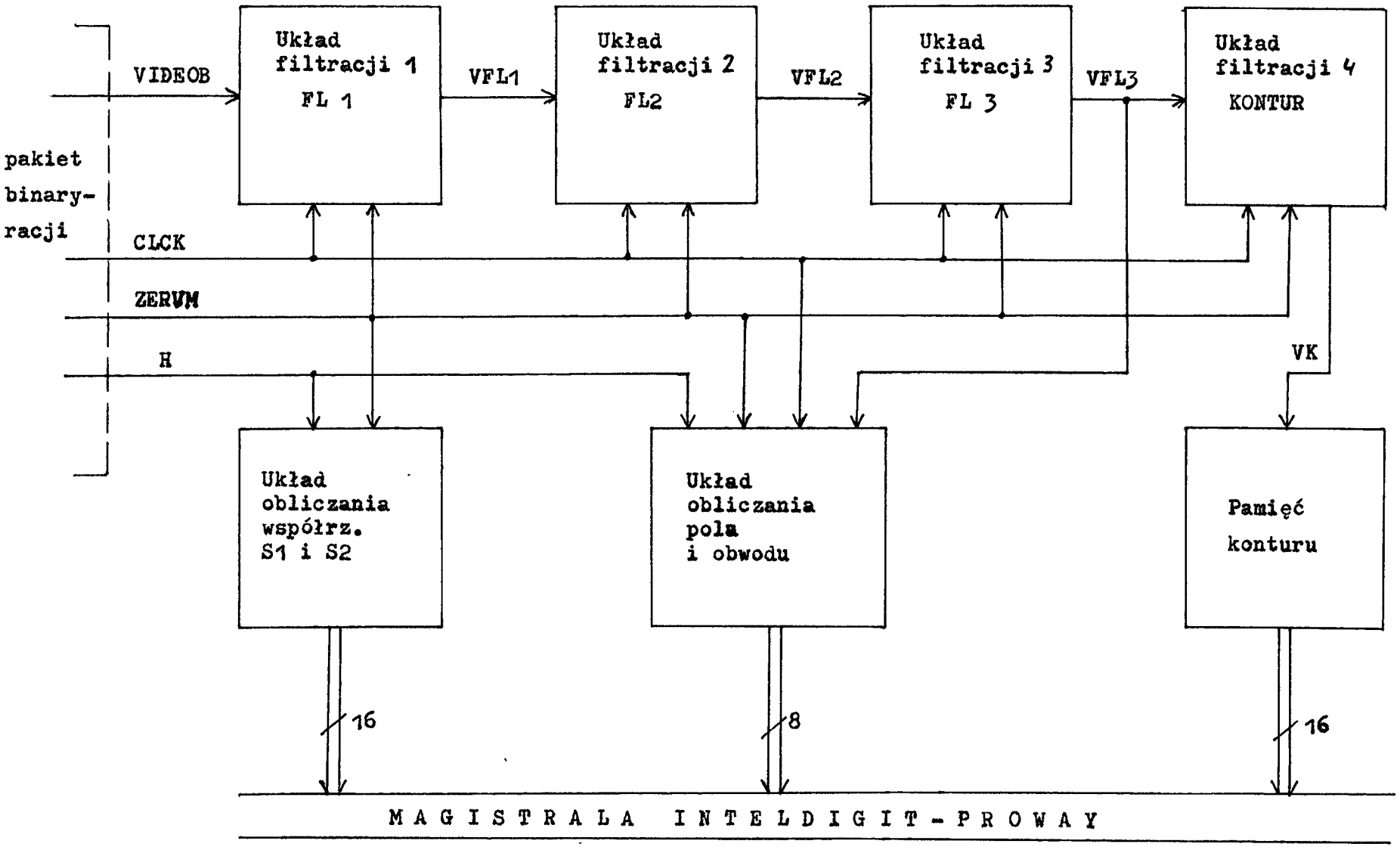
2.1)Układy filtracji obrazu 1÷4.

Zbinaryzowany obraz z pakietu binaryzacji poddawany jest czterem kolejno po sobie następującym filtracjom, z których czwarta wydziela kontur detalu. Filtracje te operują na okienku 3x3. Wymagana do przygotowania odpowiedniej sekwencji punktów linia opóźniająca zgodna z rysunkiem 4 realizowana jest przez układ dwóch rejestrów przesuwających 256-bitowych typu Intel 4731 (U39,U40) oraz przerzutnika 174 (U33).



Rys.4)Linia opóźniająca.

Taka sekwencja punktów obrazu steruje linie adresowe pamięci PROM - KP556077 (U38) o pojemności 1Kx8 bitów i czasie dostępu 60 ns. Pamięć jest zaprogramowana w ten sposób, że



Rys. 3. Schemat blokowy pakietu preprocesora.

każdy bajt zawiera zmodyfikowane wartości środkowego punktu okienka odpowiadające 8 różnym filtracjom. Wybór odpowiedniej filtracji dokonywany jest poprzez odpowiednie ustawienie mikroprzełączników sterujących wejścia adresowe multipleksera '151 (U41). W ten sposób istnieje możliwość wyboru dla każdego z czterech układów filtracji jednej z ośmiu filtracji; w przypadku konieczności zastosowania innego rodzaju filtracji można tego dokonać poprzez wymianę pamięci PROM. Sygnały odpowiadające punktom a,d,h,c,f,i okienka są odpowiednio zbramkowane w celu wyzerowania tych punktów przy obliczaniu nowych wartości dla punktów bocznych linii obrazu. Odpowiednie sygnały generuje układ złożony z licznika '393, inwertera U49, dwóch bramek NOR (U42) oraz dwóch bramek NAND (U44).

2.2)Układ obliczania pola i obwodu sylwetki.

Układ oblicza obwód sylwetki sumując punkty konturu oraz pole poprzez zliczanie wszystkich należących do niej punktów. Służą do tego liczniki '393 (U7) oraz 8253 (U27). Licznik U7 - liczący modulo 4 - użyty został w celu zredukowania częstotliwości impulsów podawanych na wejścia zliczające licznika 8253 do 2 MHz. Wprowadza to przybliżenie w obliczaniu obwodu i pola z dokładnością do 4 punktów. Licznik 393 (U6) wraz z bramkami U47, 2xU48, 2xU49 generuje sygnał LICZP, który blokuje zliczanie punktów sylwetki na okres trzech pierwszych linii obrazu (wtedy w rejestrach układów filtracji znajduje się informacja przypadkowa) oraz sygnał LICZK, który zezwala na zliczanie punktów konturu od chwili pojawienia się danych obrazowych na wyjściach rejestrów układu filtracji wydzielającym kontur (po czterech pierwszych liniach obrazu). Licznik 8253 ma wyjścia trójstanowe, tak że procesor nadrzędny odczytuje dane o obwodzie i polu sylwetki poprzez magistralę rezydentną bez dodatkowych buforów.

2.3)Układ obliczania współrzędnych S1 i S2.

Procesor nadrzędny oblicza współrzędne środka ciężkości ze wzorów:

X = S1/p - współrzędna x środka ciężkości,

Y = S2/p - współrzędna y środka ciężkości,

gdzie p jest polem obiektu, natomiast parametry S1 i S2 obliczane są zgodnie ze wzorami:

$$S1 = \sum_{i=1}^{N}\sum_{j=1}^{M} i \cdot h_{ij}$$

$$S2 = \sum_{i=1}^{N}\sum_{j=1}^{M} j \cdot h_{ij}$$

$$; \quad h_{ij} = \begin{cases} 1 - \text{gdy punkt należy do obszaru obiektu,} \\ \emptyset - \text{w przeciwnym przypadku.} \end{cases}$$

$$; \quad N = M = 256$$

Wartości wyrażeń S1 i S2 mogą przekroczyć 8M dla obrazu 256x256 punktów. Konieczne są zatem liczniki 24-bitowe dla obu współrzędnych. Współrzędna S1 jest obliczana przez zespół liczników '393 (U4,U9,U10) oraz jednostki arytmetyczno-logiczne '181 (U29,U30); wartość współrzędnej S2 obliczają odpowiednio układy U5,U31,U32,U11,U12. Licznik U4 określa współrzędną danego punktu w linii obrazu, a licznik U5 jego numer linii. Sygnał LICZS sygnalizuje czy dany punkt obrazu należy do obszaru obiektu. Jego wysoki poziom zezwala na dodanie współrzędnej aktualnego punktu w linii do zawartości jednostek arytmetyczno-logicznych U29 i U30 oraz numeru linii dla tego punktu do U31 i U32. Chwilowe wartości 8 najmłodszych bitów współrzędnych S1 i S2 przechowywane są w rejestrach '174 (U34,U35,U36). Liczniki U9 i U10 przechowują 16 najstarszych bitów współrzędnej S1 natomiast,dla współrzędnej S2 zadanie to spełniają liczniki U11,U12. Wartości S1 i S2 są odczytywane przez procesor nadrzędny poprzez bufory trójstanowe '245 (U13,U14,U15,U16,U17,U18). Liczniki i zawartości przerzutników są zerowane na początku każdego obrazu sygnałem ZERVH.

2.4)Pamięć konturu.

Pamięć konturu zbudowana jest na dwóch układach pamięci RAM 6116 o pojemności 2Kx8 bitów. Pierwsza z nich (U1) prze-

chowuje współrzędne x punktów konturu, druga zaś (U2) odpowiadające im współrzędne y. Danymi dla pamięci są zawartości omawianych już liczników U4 i U5; adresy generowane są przez liczniki U34 i U8, które zliczają kolejno punkty konturu. Dostępem do pamięci steruje sygnał CSRLC.

Procesor 8086 odczytuje dane z pamięci poprzez bufory '245 (U23,U24) wystawiając odpowiednie sygnały na magistralę rezydentną.

2.)OPROGRAMOWANIE.

Oprogramowanie systemu wizyjnego 2-D dla robota przemysłowego sprowadza się do napisania odpowiednich procedur dla mikrokontrolera 8031 oraz dla procesora 8086, który pełni w systemie rolę procesora nadrzędnego.

Procesor 8031 ma za zadanie inicjalizację układów pakietu binaryzacji oraz określenie wartości progu komparacji na podstawie histogramu w celu zbinaryzowania obrazu. Inicjalizację procesor wykonuje po sygnale RESET z przełącznika na płycie czołowej pakietu preprocesora lub od procesora 8086. Natomiast obliczanie progu komparacji rozpoczyna sygnał końca obrazu podawany na wejście przerwań procesora.

Procesor 8086 inicjalizuje układy pakietu preprocesora po włączeniu zasilania lub po sygnale RESET z płyty czołowej. Po sygnale końca obrazu podawanym na wejście przerwań procesor odczytuje dane obliczone sprzętowo przez układy preprocesora oraz oblicza parametry konieczne do określenia położenia detalu i jego orientacji zgodnie z procedurami przedstawionymi w dalszej części sprawozdania.

W trybie uczenia system wizyjny wykorzystuje monitor pakietu MM-16.

W trybie pracy automatycznej konieczna jest wymiana informacji pomiędzy systemem wizyjnym a układem sterowania robota. Komunikacja ta odbywa się łączem szeregowym. Układy transmisji szeregowej obsługiwane są przez układ wizyjny i układ sterowania robota za pomocą przerwań, przy czym dla układu wizyjnego przerwanie od układu sterowania robota ma najwyższy priorytet. Układ sterowania robota wysyła do układu wizyjnego kody instrukcji o postaci przedstawionej w "Założeniach na oprogramowanie współpracy układu sterowania robotów IRp z inteligentnymi układami sensorycznymi". W systemie wizyjnym każda instrukcja obsługiwana jest oddzielną procedurą wywoływaną po identyfikacji rodzaju instrukcji.

Przy takim zorganizowaniu pracy systemu wizyjnego w trybie pracy automatycznej nie jest konieczne korzystanie z systemów operacyjnych.

Sytuacje alarmowe będą sygnalizowane lampkami sygnalizacyjnymi na płycie czołowej pakietu preprocesora.

Przetwarzanie odczytanego przez kamerę obrazu podzielone jest na trzy etapy:

- wstępne przetwarzanie obrazu,
- rozpoznawanie obrazu,
- określanie położenia i orientacji obiektu.

2.1) W etapie pierwszym dokonywane jest eliminowanie zakłóceń występujących w obrazie przy minimalnym zniekształceniu zawartej w nim informacji. Ze względu na wymagania czasu rzeczywistego filtracje zrealizowano sprzętowo w sposób opisany w punkcie 1.2.1 niniejszego sprawozdania zgodnie z metodami przedstawionymi w sprawozdaniu do zadania 1.2 zlecenia RP-61. Filtracje dobierane są z wykorzystaniem programu SOKNO (załącznik).

2.2) Celem etapu drugiego jest zaliczenie obiektu do jednej z klas lub stwierdzenie, że obiekt jest nieznany.
Uzyskany po etapie wstępnego przetwarznia obrazu kontur detalu zostaje zakodowany w postaci wektora parametrów lub pewnych funkcji opisujących jego kształt. W tym rozwiązaniu przyjęto parametryczną metodę kodowania konturu, ze względu na fakt, że niektóre parametry można obliczyć w prosty sposób sprzętowo, co znacznie przyspiesza proces rozpoznawania. Parametry dobrane zostały tak, aby nie były zależne od pozycji i orientacji obiektu w polu widzenia kamery.

Wybrane parametry to:

p1 - pole obiektu
p2 - minimalny moment bezwładności
p3 - maksymalny moment bezwładności
p4 - obwód
p5 - minimalna odległość środka ciężkości od obwodu
p6 - maksymalna odległość środka ciężkości od obwodu
p7 - średnia odległość środka ciężkości od obwodu

Wzory pozwalające obliczyć powyższe parametry zostały przedstawione w założeniach na system wizyjny 2-D dla robota przemysłowego (zadanie 1.2). Pole obiektu oraz obwód liczone są sprzętowo. W ten sam sposób liczone są współrzędne konieczne do wyznaczania położenia środka ciężkości. Pozostałe parametry - p2,p3,p5,p6,p7 - liczone są programowo w ten sposób, że wykorzystywane są tylko dane o konturze, nie jest natomiast konieczna pamięć obrazu RAM o pojemności 64KB. Przedstawiona w dalszej części metoda określania orientacji obiektu również bazuje na danych konturu. Zawartość pamięci konturu przepisywana jest po każdym obrazie do pamięci pakietu MM16 w celu wyeliminowania ewentualnych konfliktów z dostępem do niej.

Przyjęta metoda rozpoznawania obrazu dzieli się na dwie fazy: uczenia i pracy automatycznej.
W fazie uczenia obliczane są wzorcowe wartości wybranych parametrów jako średnia arytmetyczna 10-ciu kolejnych obrazów dla każdego stabilnego położenia (klasy) detali.
W fazie pracy automatycznej obliczane są parametry dla obiektu w polu widzenia kamery, a następnie porównywane z tablicą parametrów wzorcowych.

Na rys.5 przedstawiony jest schemat blokowy programu KLASYFIKACJA dla fazy uczenia.

Założenia do programu:
- w polu widzenia kamery znajduje się pojedynczy obiekt,
- kamera znajduje się nad obiektem na wysokości ostrego widzenia,

Informacje wejściowe do programu:
- pole obiektu p1,
- obwód p4,
- kontur obiektu,
- współrzędne S1,S2.

Informacje wyjściowe:

- tablica wzorcowych parametrów dla danej klasy o długości 15 bajtów w postaci:

ilość bajtów - 1 2 2 2 2 2 2 2

kl	p1	p2	p3	p4	p5	p6	p7

gdzie: kl - numer klasy.

Używane w programie oznaczenia:

- X_i, Y_i - współrzędne i-tego punktu konturu,
- flaga=1 mówi, że punkty pomiędzy dwoma punktami konturu należą do sylwetki,
  flaga=0 w przeciwnym przypadku,
- IX,IY,IXC,IYC,IXCYC odpowiadają chwilowym wartościom parametrów $I_x, I_y, I_{xc}, I_{yc}, I_{xcyc}$.

W trakcie ruchu ramienia robota na podstawie różnicy pomiędzy środkiem ciężkości obiektu a środkiem obrazu kamery wyznaczane są wartości korekcji dla układu sterowania robota. Wartości te są przesyłane do obszaru pamięci wspólnej w formie przesyłki zgodnej z protokołem wymiany informacji pomiędzy układem sterowania robota a układami sensorycznymi. Dane te są wykorzystywane przez instrukcję pozycjonowania swobodnego. Przesyłka o wartościach zerowych oraz osiągnięcie przez kamerę odległości ostrego widzenia detalu rozpoczyna rozpoznawanie, a następnie określanie orientacji obiektu.

Procedura rozpoznawania (klasyfikacji) wykorzystuje metodę najbliższego sąsiada. Polega ona na znalezieniu takiego i, dla którego prawdziwa jest nierówność:

$$\|X - X_i\| \leqslant \|X - X_k\| \quad , \quad k = 1,2,\ldots,N$$

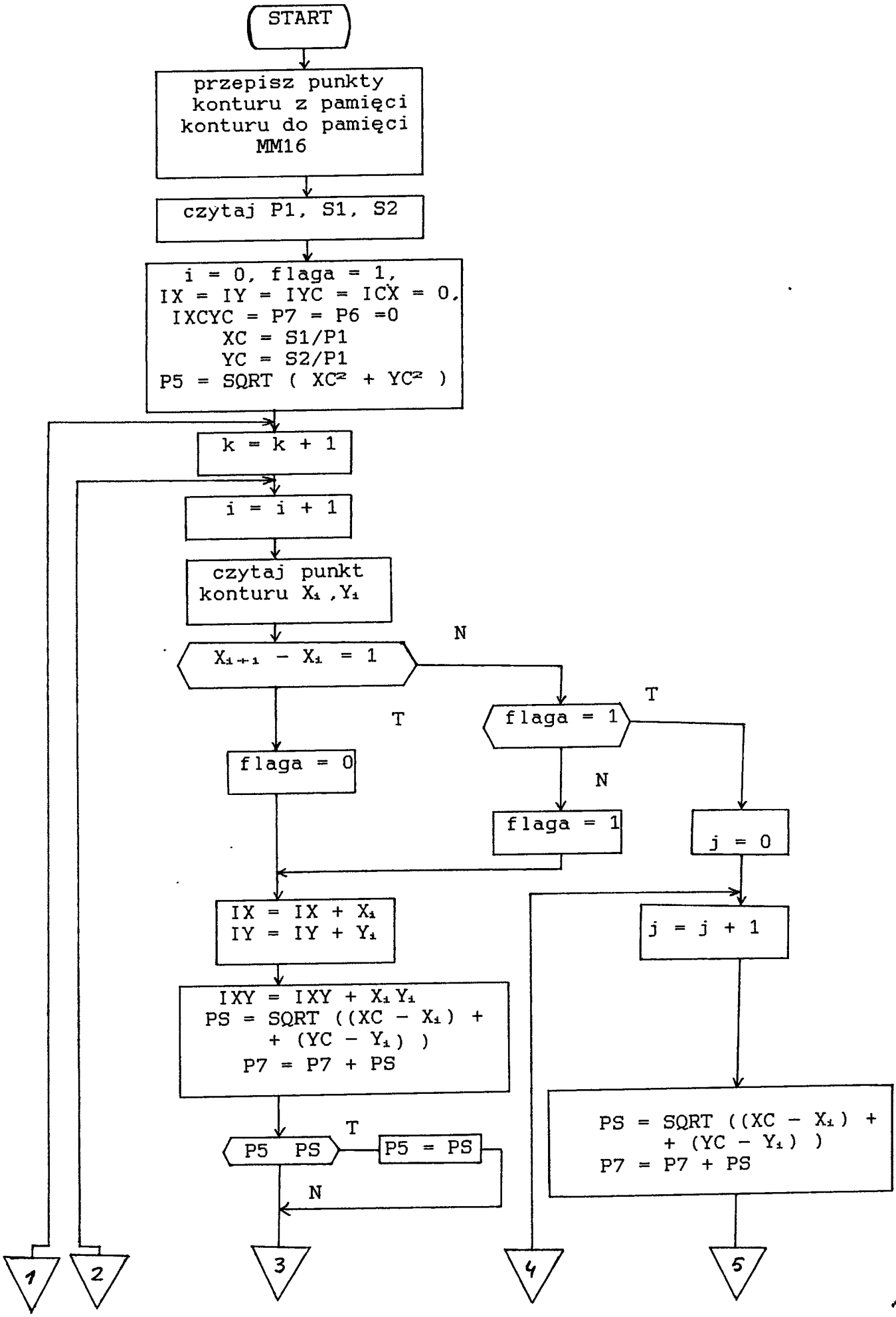
gdzie:

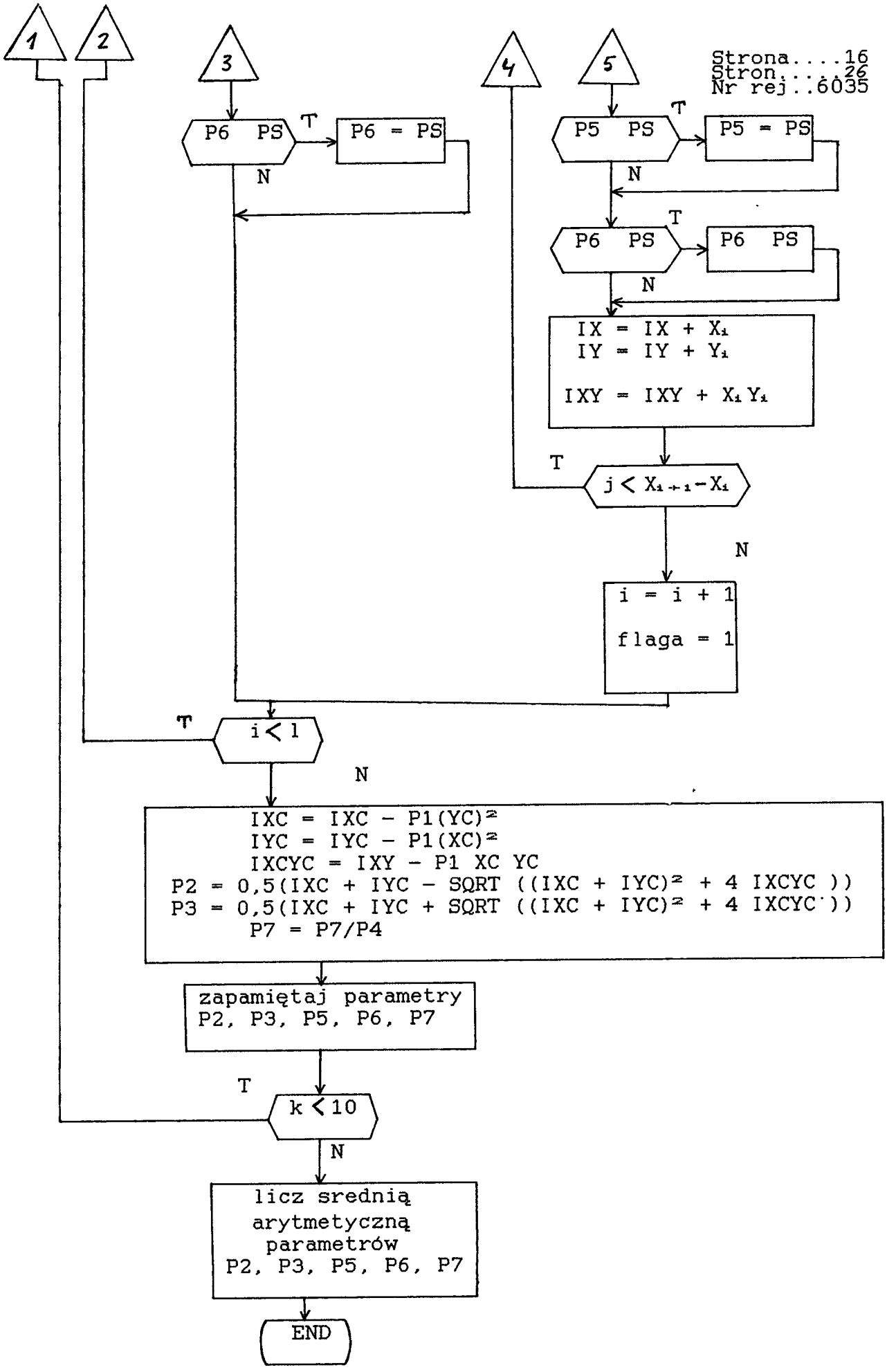
i - numer klasy

N - ilość klas

X - wektor parametrów klasyfikowanego obiektu

$\|X - X_k\| = \sum_{j=1}^{7} |x_j - x_{kj}|$, X_i - wektor parametrów klasy i


START
przepisz punkty konturu z pamięci konturu do pamięci MM16
czytaj P1, S1, S2
i = 0, flaga = 1,
IX = IY = IYC = ICX = 0,
IXCYC = P7 = P6 =0
XC = S1/P1
YC = S2/P1
P5 = SQRT (XC² + YC²)
k = k + 1
i = i + 1
czytaj punkt konturu Xi , Yi
Xi+1 − Xi = 1
N
T
T
flaga = 1
flaga = 0
N
flaga = 1
j = 0
IX = IX + Xi
IY = IY + Yi
j = j + 1
IXY = IXY + Xi Yi
PS = SQRT ((XC − Xi) +
+ (YC − Yi))
P7 = P7 + PS
T
P5 PS
P5 = PS
N
PS = SQRT ((XC − Xi) +
+ (YC − Yi))
P7 = P7 + PS
1
2
3
4
5



Rys.5 Schemat blokowy programu KLASYFIKACJA dla fazy uczenia.

2.3) W etapie trzecim przetwarzania obrazu dokonywane jest określanie orientacji obiektu zdefiniowanego w fazie rozpoznawania, tzn. obliczanie kąta obrotu obiektu w stosunku do położenia przyjętego za bazowe.

Etap ten poprzedzony jest również fazą uczenia.
Na rys.6 przedstawiony został schemat blokowy programu ORIENTACJA dla fazy uczenia.

Założenia do programu są takie same jak w przypadku programu KLASYFIKACJA.

Informacje wejściowe:
- promienie okręgów (dobrane przy wykorzystaniu programu SOKNO),
- kontur obiektu,
- obwód p4.

Informacje wyjściowe:
- tablica wzorcowych wartości kątów dla danej klasy w postaci:

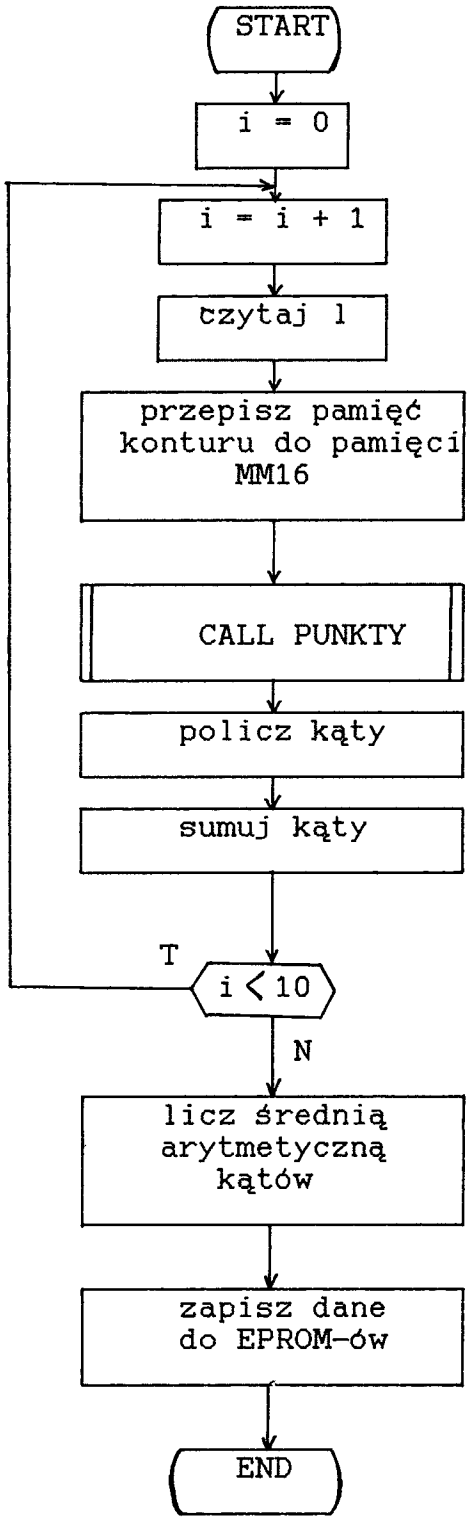
ilość bajtów: 1 1 1 1 2 1 1 2

kl	ir	r1	l1	α 11	. . .	rn	ln	α 1n ···

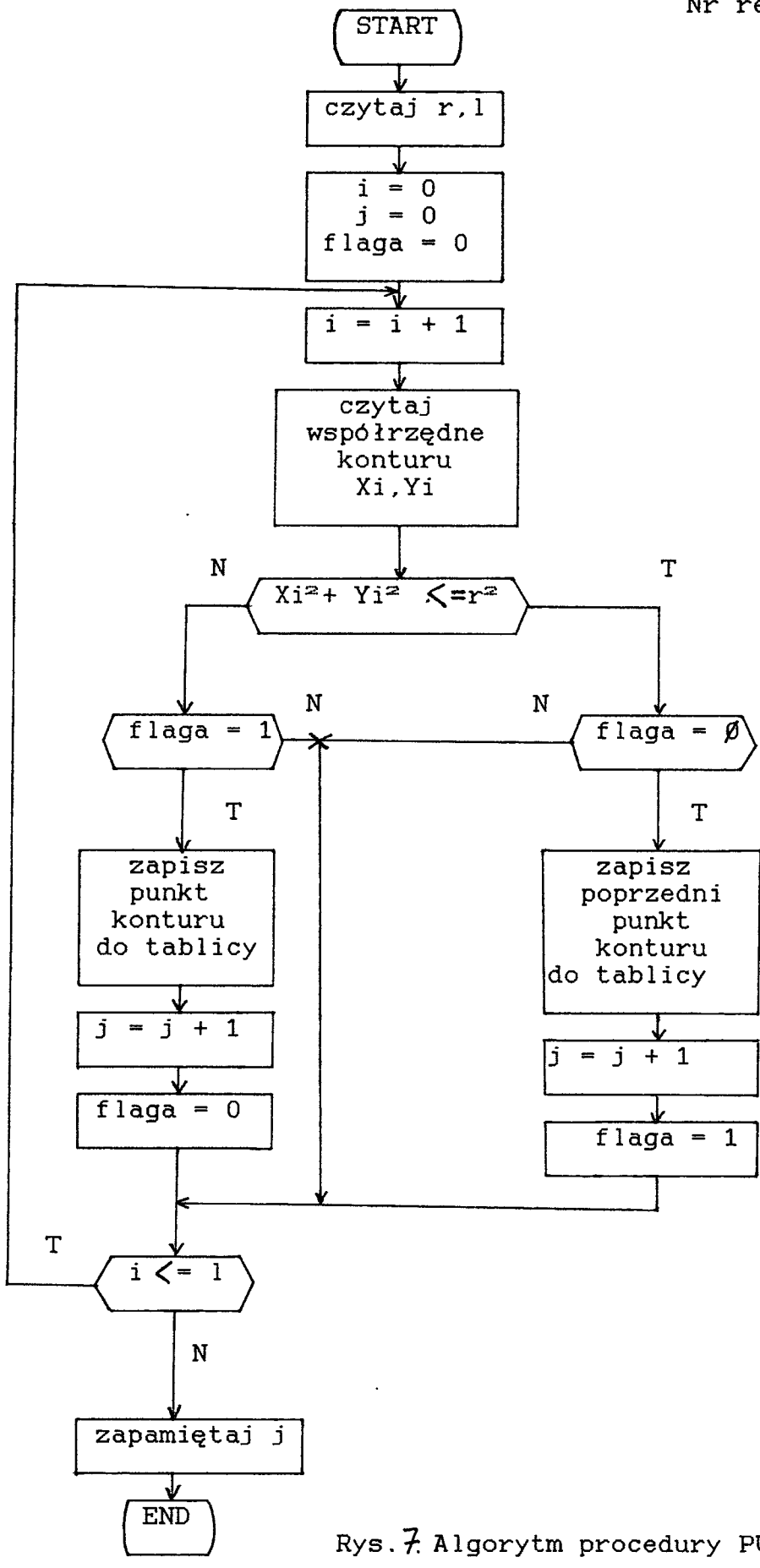
gdzie:

kl - numer klasy,
ir - ilość promieni,
r1 - pierwszy promień,
l1 - ilość kątów dla pierwszego promienia,
α 11 - pierwszy kąt pierwszego promienia,

przy czym pierwszy kąt jest kątem dla pierwszego punktu wspólnego konturu i okręgu w stosunku do osi x, natomiast pozostałe są kątami pomiędzy poszczególnymi punktami wspólnymi.



Rys.6 Schemat blokowy programu ORIENTACJA dla etapu uczenia.



Rys. 7. Algorytm procedury PUNKTY.

W fazie pracy automatycznej w pierwszej kolejności ustalana jest kolejność obliczonych kątów (bez pierwszego) w odniesieniu do kolejności dla kątów wzorcowych. Wykorzystuje się tutaj również metodę najbliższego sąsiada, przy czym wektor parametrów X stanowią w tym wypadku kąty wzorcowe, natomiast wektorami Xk są kąty obliczone z każdorazowym przesunięciem otatniego z nich na pierwszą pozycję. Następnie od wartości kąta dla punktu, który odpowiada pierwszemu punktowi dla położenia przyjętego za bazowe odejmowana jest wartość pierwszego kąta z tablicy kątów wzorcowych.

3.)KOMUNIKACJA SYSTEMU WIZYJNEGO 2-D.

1.) W skład systemu wizyjnego wchodzą:

- pakiet binaryzacji obrazu PB-21 - grubość pakietu 1T,
- pakiet preprocesora PP-21 - grubość pakietu 1T,
- pakiet procesora MM-16 - grubość pakietu 2T.

Łącznie system wizyjny zajmuje 3 stanowiska w kasecie Inteldigit-Proway układu sterowania robota.

2.) Komunikacja pomiędzy pakietami systemu wizyjnego.

Komunikacja pomiędzy pakietem binaryzacji a pakietem preprocesora odbywa się poprzez niewykorzystane styki na płaterze magistrali przeznaczone do połączeń indywidualnych. Wykorzystywane są następujące styki:

Bb8 : sygnał aktywny podczas wyprowadzania linii matrycy H,
Bb9 : znaczniki nowego obrazu ZERVH,
Bb10: takt bramkowany sygnałem H zawierający dodatkowo 1024 impulsy po końcu obrazu,
Bb11: sygnał RESET (z pakietu preprocesora).

Sygnał wizyjny podawany jest do pakietu preprocesora kablem koncentrycznym z łącza BNC płyty czołowej pakietu binaryzacji.

Komunikacja pomiędzy pakietem preprocesora a pakietem procesora MM-16 odbywa się poprzez magistralę rezydentną, która w pakiecie jednostki centralnej MM-16 wyprowadzona jest złączem szufladowym umieszczonym na płycie czołowej. Pakiet preprocesora wyposażony jest w związku z tym w takie samo wyprowadzenie. Oba wyprowadzenia połączone są kablem płaskim. Wymiana informacji magistralą rezydentną - zgodna z protokołem wymiany informacji dla magistrali Inteldigit-

Proway - zarządzana jest przez pakiet MM-16; pakiet preprocesora jest pakietem pasywnym.

3.) Komunikacja z układem sterowania robota.

Pakiety jednostek centralnych systemu wizyjnego i układu sterowania robota komunikują się poprzez magistralę Inteldigit-Proway. Ponieważ wykorzystywane pakiety MM-16 zostały zaprojektowane jako pakiety aktywne, nie ma możliwości ich bezpośredniej komunikacji. Wymiana informacji pomiędzy nimi odbywa się zatem przez wspólny obszar pamięci. Wyższy priorytet jednostki centralnej układu sterowania w dostępie do magistrali jest zapewniony poprzez arbitraż szeregowy magistrali zrealizowany sprzętowo. Wymiana danych odbywa się zgodnie z protokołem wymiany informacji pomiędzy układem sterowania robota a układami sensorycznymi. Wymiana informacji ma na celu korekcję trajektorii ruchu ramienia robota w zależności od wektora korekcji wyznaczonego przez system wizyjny w układzie współrzędnych związanych z kamerą.

4.) Komunikacja z pozostałymi urządzeniami współpracującymi z systemem wizyjnym.

4.1)Kamera matrycowa CCD.

Z kamery matrycowej CCD sygnały podawane są kablem na łącze typu BNC na płycie czołowej pakietu binaryzacji.
Są to następujące sygnały:

- video, w standardzie TV, częstotliwość zegara 8 MHz,
- synchronizacja linii H,
- synchronizacja ramki V,
- zasilanie +15V, GND (z pakietu binaryzacji).

4.2)IBM PC.

Z komputera klasy IBM PC przesyłane będą w fazie uruchamiania systemu kody programów do pamięci RAM pakietu MM-16. Wykorzystywany będzie do tego celu pakiet transmisji szere-

gowej MI-24 lub łącze szeregowe pakietu MM-16, jeżeli instrukcja LOAD monitora będzie obsługę tego łącza zapewniała.

4.3)Monitor TV.

Do monitora TV przesyłany będzie zbinaryzowany obraz kablem koncentrycznym z gniazda na płycie czołowej pakietu binaryzacji.

4) WIZUALIZACJA WYNIKÓW PRZETWARZANIA OBRAZÓW I BADANIA ALGORYTMÓW FILTRACJI.

Dla potrzeb wizualizacji wyników przetwarzania obrazów opracowano bibliotekę procedur graficznych realizujących proste funkcje graficzne jak np. rysowanie punktów, linii, okręgów. Ponieważ wizualizacja służy jedynie do badania skuteczności wybranych algorytmów, nie ma natomiast wpływu na czas przetwarzania systemu wizyjnego, procedury zostały napisane w języku C (dla kompilatora Microsoft C 4.0); krytyczne czasowo programy realizujące omówione w części 3 algorytmy rozpoznawania obrazów będą napisane w asemblerze procesora 8086. Funkcje te przewidziane są dla karty grafiki kolorowej wysokiej rozdzielczości EGA, która w trybie 10 zapewnia rozdzielczość 640x350 punktów. Wywołanie danej procedury graficznej odbywa się poprzez umieszczenie w programie jej nazwy z odpowiednim zestawem parametrów.

Oto przygotowane funkcje:

EGA

funkcja powoduje przełączenie trybu pracy karty graficznej.

wywołanie: `ega(1);` - powoduje przełączenie karty w tryb graficzny,
`ega(0);` - powoduje przełączenie karty w tryb tekstowy.

CLRSCR

funkcja powoduje wyzerownie zawartości pamięci ekranu.

wywołanie: `clrscr();`

MOVE

funkcja przesuwa kursor graficzny do zadanego punktu.

wywołanie: `move(x,y);`

POINT

funkcja rysuje na ekranie monitora punkt o zadanych współrzędnych i barwie.

wywołanie: point(x,y,barwa); - barwa jest liczbą od 0 do 16 odpowiadającą kolejnej barwie ze standardowej palety kolorów karty EGA.

LINE

funkcja rysuje linię pomiędzy punktami o podanych współrzędnych.

wywołanie: line(x1,y1,x2,y2);

CIRCLE

funkcja rysuje okrąg o podanym środku i promieniu.

wywołanie: circle(x,y,promień);

PLOT

funkcja wyświetla zawartość podanej tablicy o pojemności 64K bitów w wycinku ekranu 256x256 punktów.

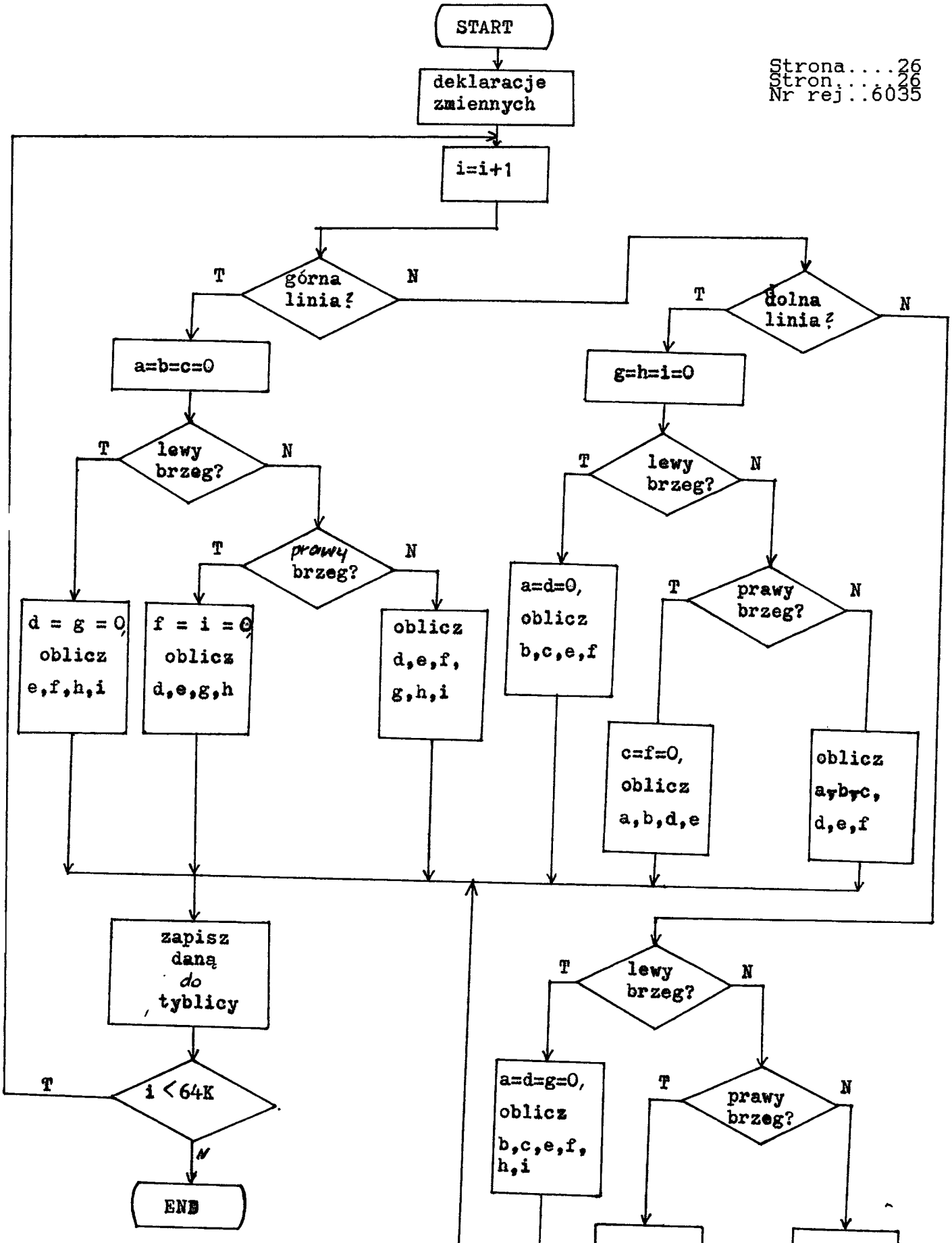
wywołanie: plot(tablica,barwa);

TEKST

funkcja pisze tekst poczynając od zadanego położenia kursora.

wywołanie: tekst(x,y,komunikat);

Na rys.8 przedstawiono dodatkowo algorytm określania punktów linii opóźniającej dla odpowiednich punktów obrazu.



Rys. 8 Algorytm procedury okienko.

BADANIA ALGORYTMÓW FILTRACJI.

Wykorzystując przedstawione powyżej procedury dla karty grafiki kolorowej EGA napisano program do badania efektywności działania różnych algorytmów filtracji obrazu.

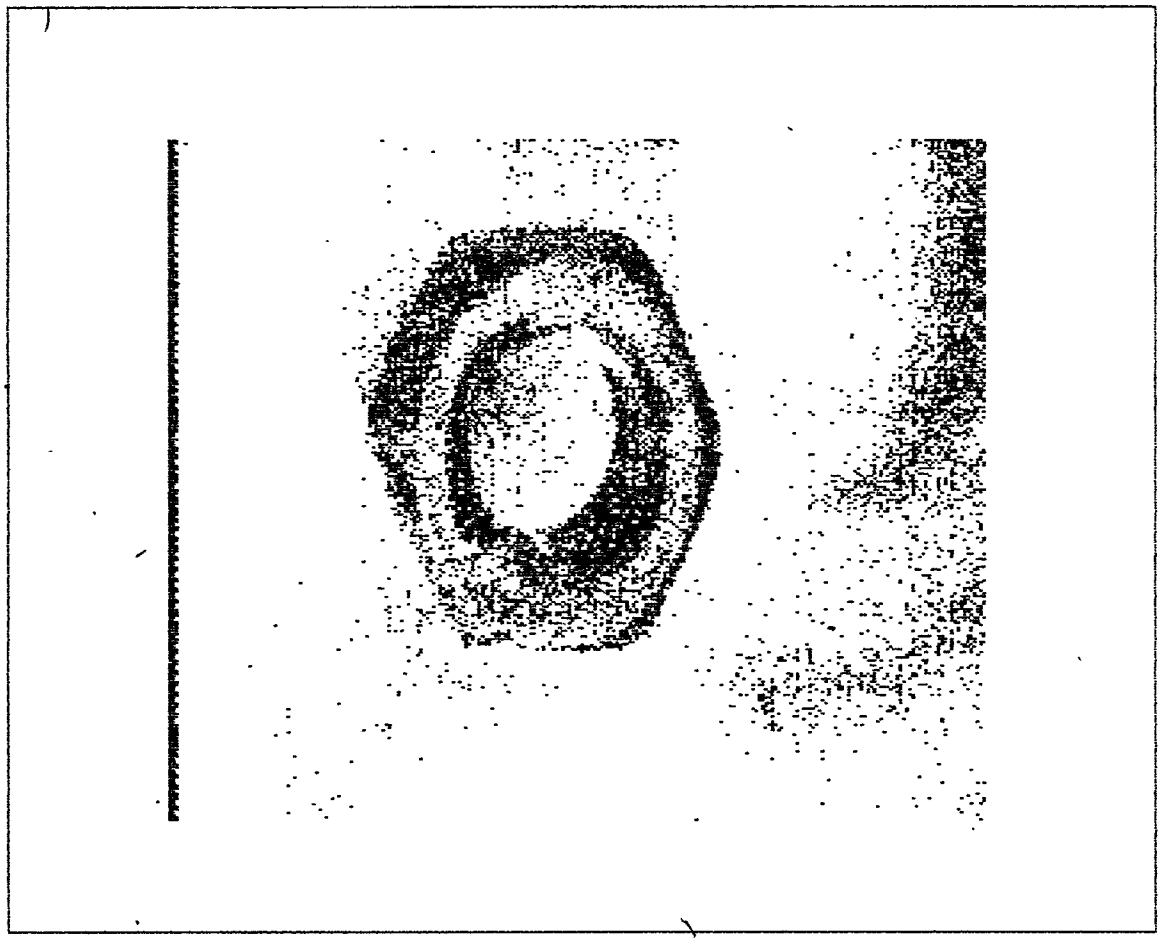
Program umożliwia przetworzenie obrazu za pomocą sekwencji 1 do 3 różnych filtracji i wydzielenie konturu. Pozwala to wybrać najbardziej odpowiedni zestaw filtracji dla danej klasy detali. Na podstawie tych badań będą odpowiednio zaprogramowane pamięci PROM pakietu preprocesora, przy pomocy których filtracje te będą realizowane sprzętowo.

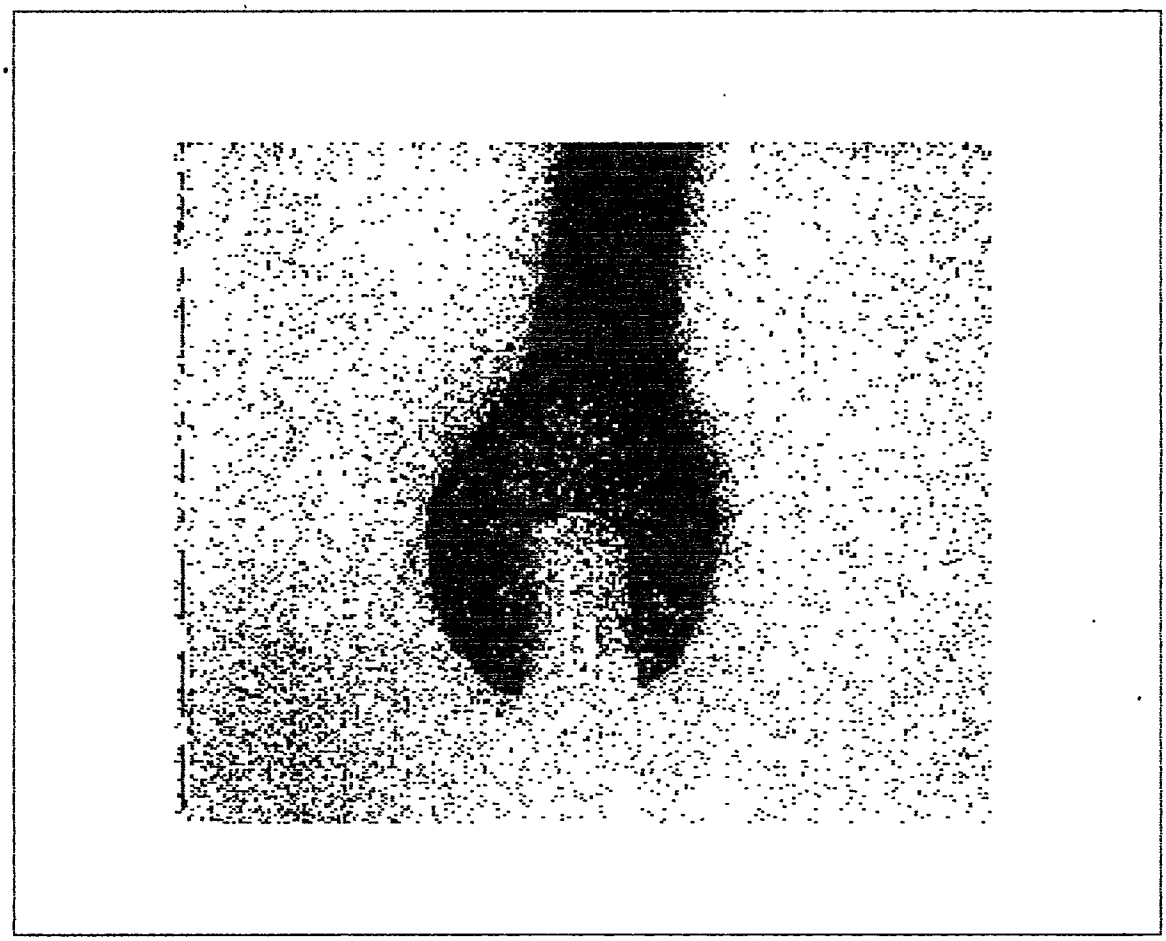
Załączniki nr 1 i 2 przedstawiają obrazy, które były wykorzystywane do badania algorytmów filtracji.

Załączniki nr 3-6 przedstawiają kontury detali otrzymanych po złożeniu dwóch filtracji logicznych oraz arytmetycznej i operacji wydzielenia konturu (zgodnie ze wzorami zamieszczonymi w sprawozdaniu z zadania 1.2). Ta sekwencja filtracji dała dla tych detali najlepsze rezultaty. Obrazy podzielone zostały na dwie części w celu umożliwienia ich wydruku (dyrektywa GRAPHICS systemu operacyjnego MS-DOS pozwala na drukowanie ekranu graficznego z największą rozdzielczością 640x200 punktów).

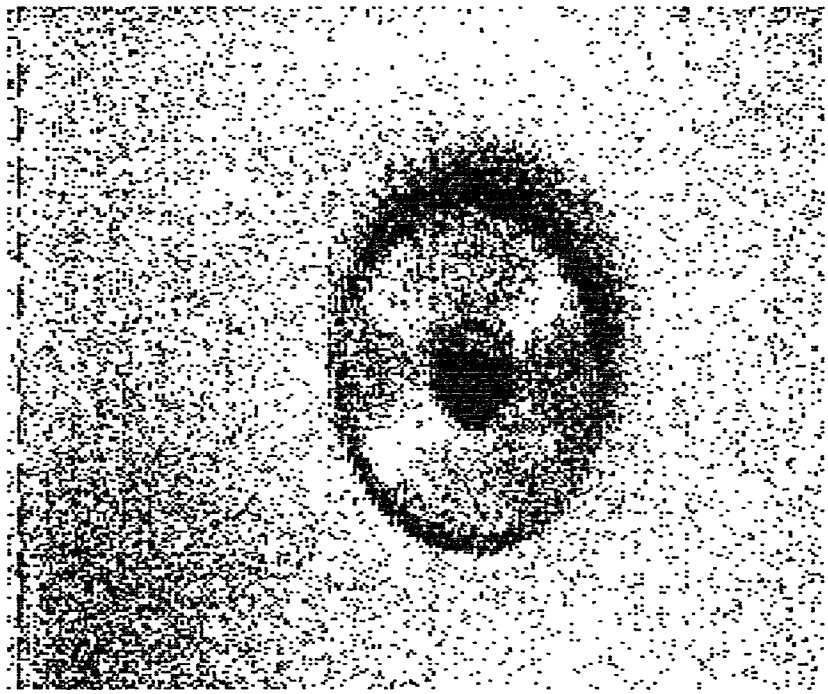
Należy zaznaczyć, że obrazy odczytywane były bez odpowiedniego oświetlenia, co ma dyży wpływ na wyniki przetwarzania.

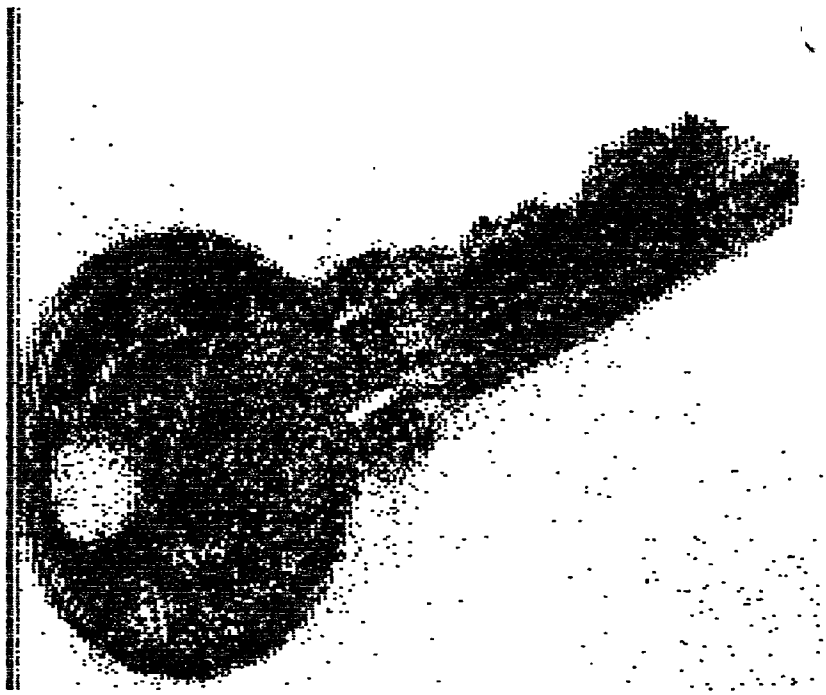
Załącznik nr 1





Załącznik nr 2

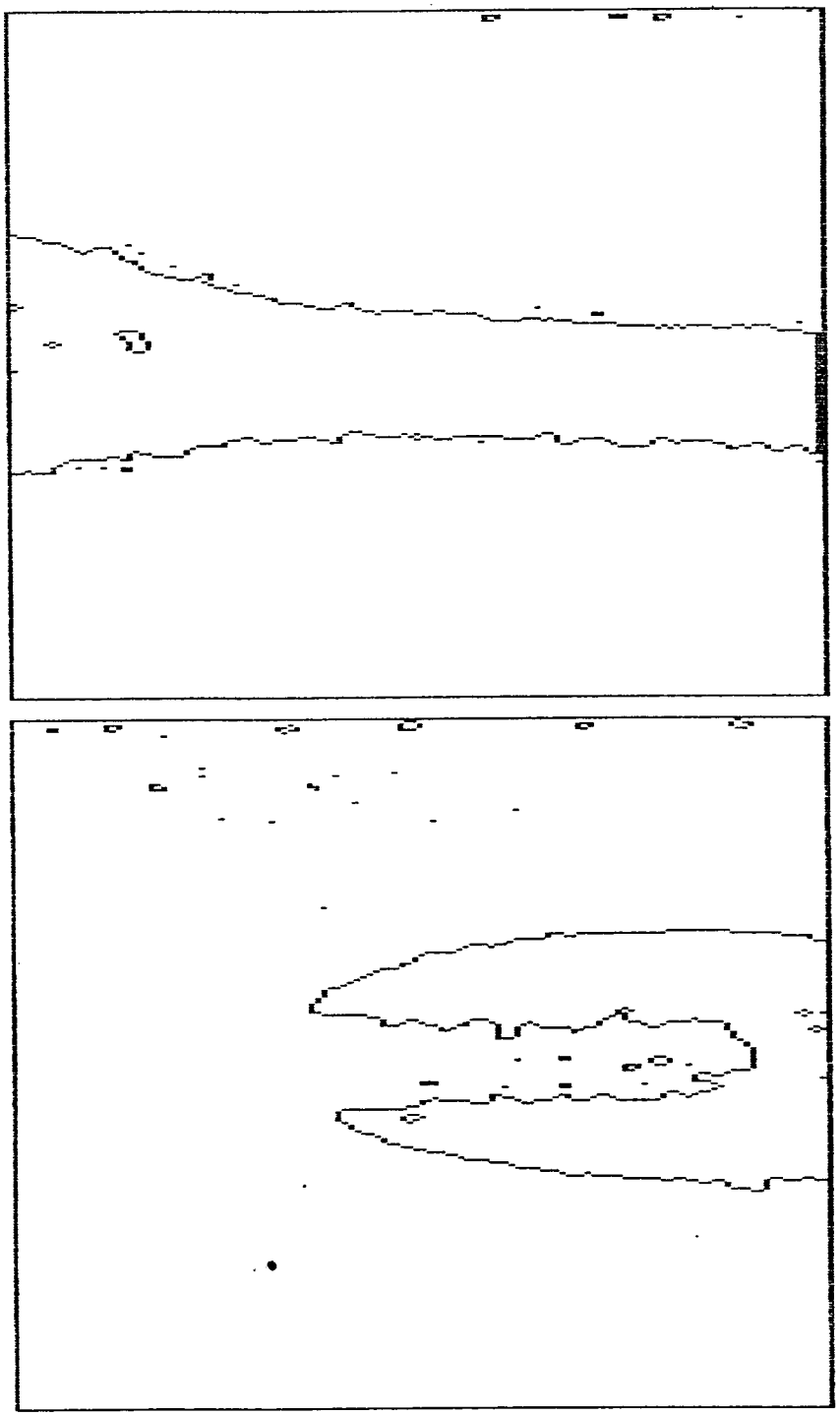




Załącznik nr 3

Klucz: próg komparacji - 5

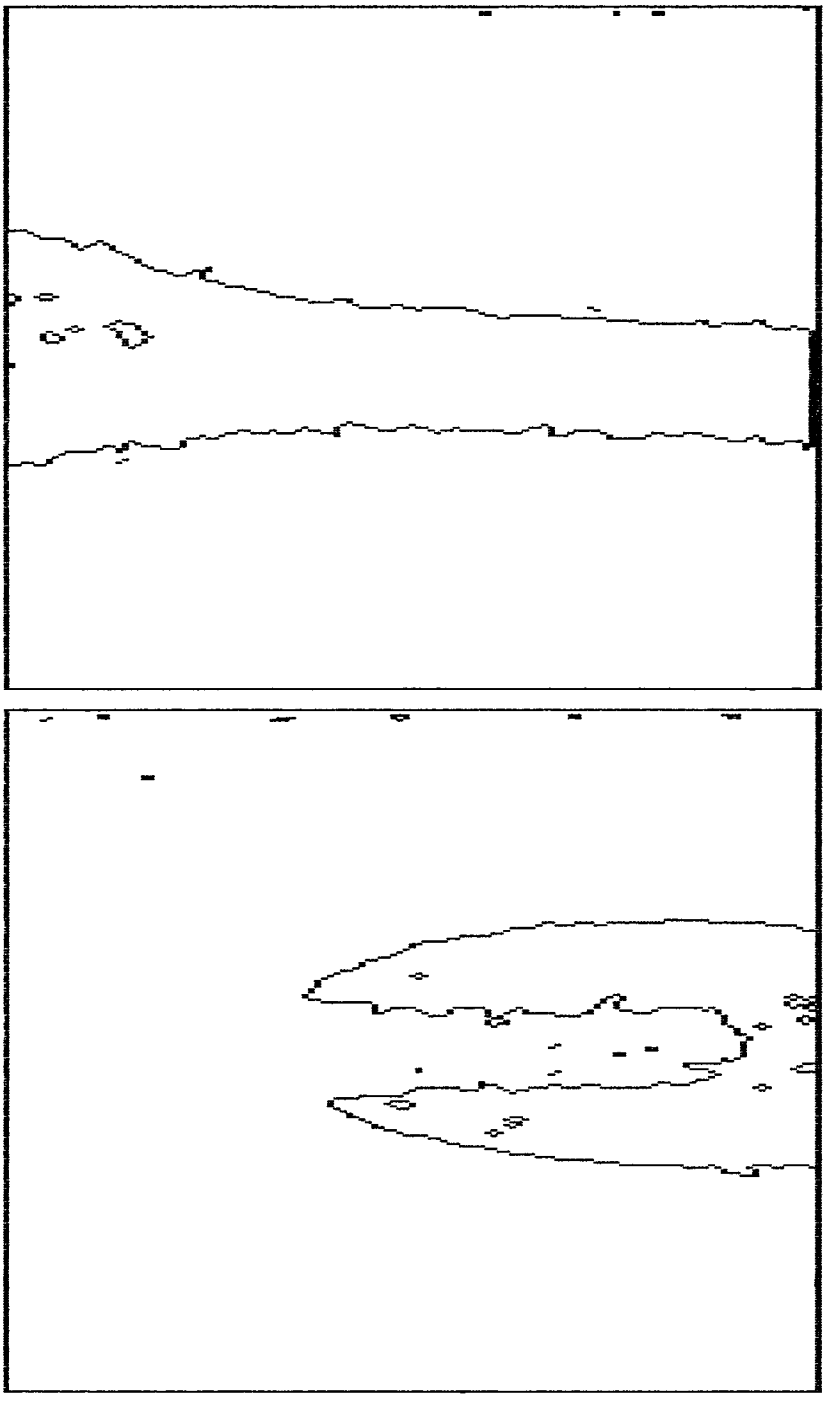
par. filtr. arytm. n=1



Załącznik nr 4

Klucz: próg komparacji - 6

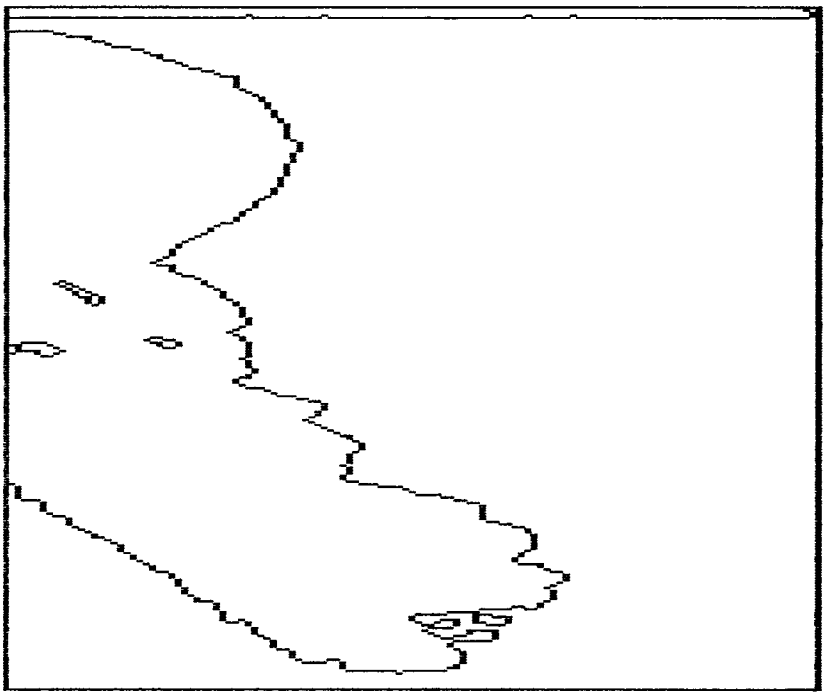
par. filtr. arytm. n=2

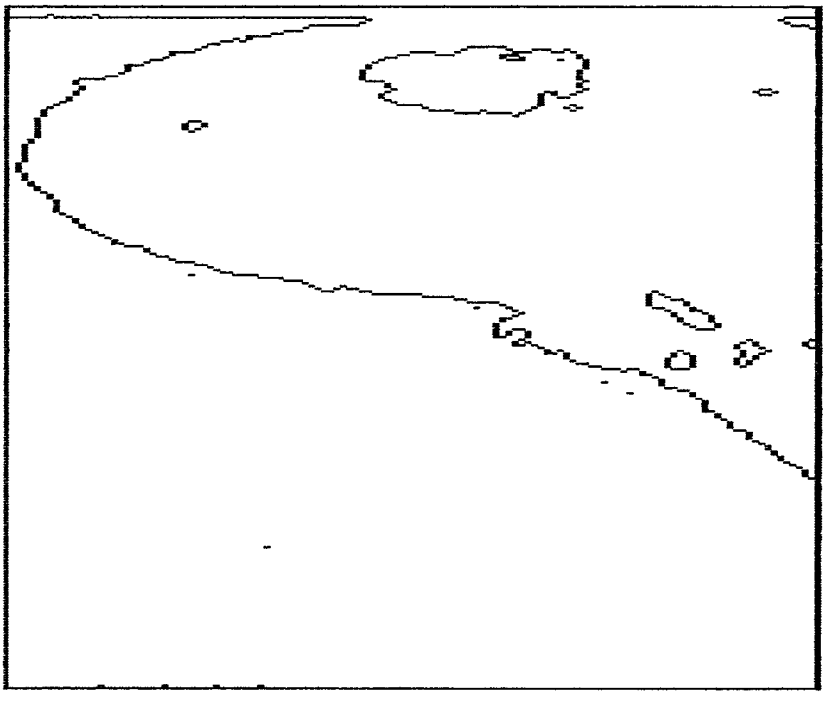


Załącznik nr 5

Klucz: próg komparacji - 7

par. filtr. arytm. n=1

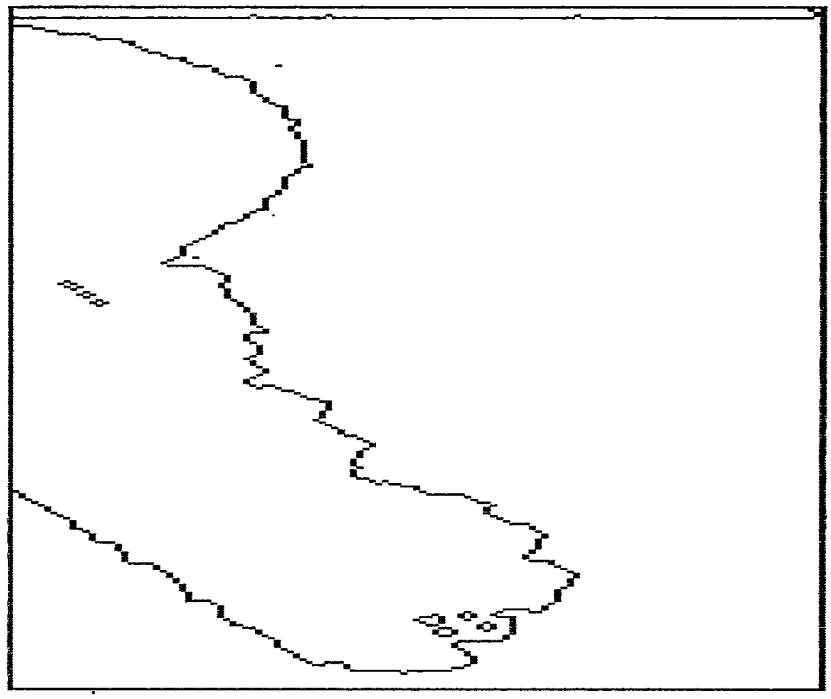


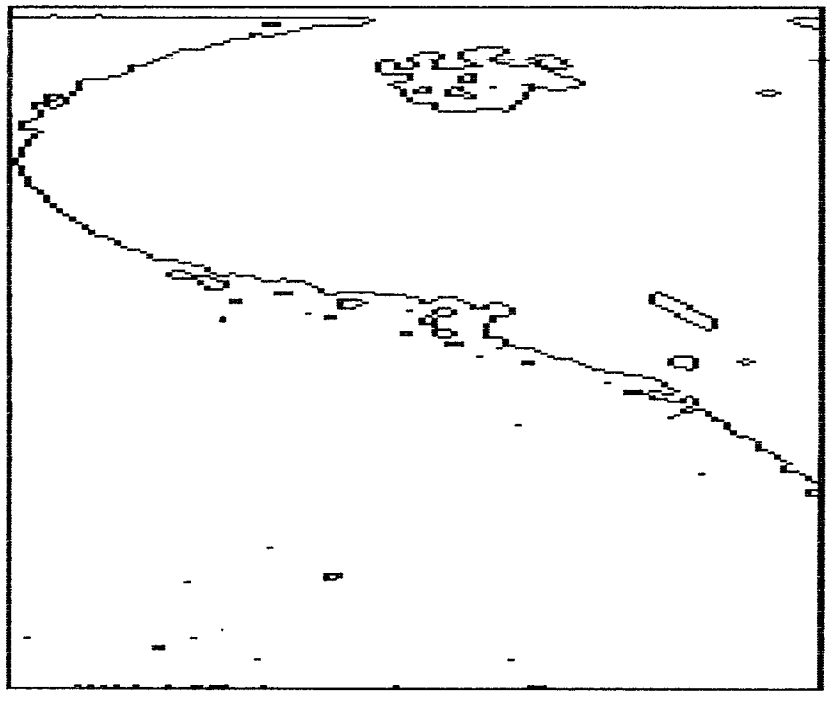


Załącznik nr 6

Klucz: próg komparacji - 8

par. filtr. arytm. n=1





Załącznik nr 9

LISTA ELEMENTOW PAKIETU PB-21.

Układy scalone:

Intel 8031	U1
Intel 6116 (lub 6132)	U3
Intel 2716 (lub 2732)	U2
Intel 8282	U4
Intel 8253	U10,U11,U12
Intel 8205	U5
K1107PW1	U32
K574YD1A	U33
AD 7520	U7
AD 741K	U8
ULY 7710N	U9
74LS393	U23,U24
74LS193	U13-U20,U25
74LS155	U22
74LS132	U32
74LS75	U6
74LS74	U21
74LS32	U31
74LS08	U26,U27
74LS07	U28
74LS04	U29
7404	U30

Tranzystory:

BC 107	T2,T3
BC 413	T1

Diody:

BZP630-C6V8	D1

Rezonator kwarcowy:

RS-3011 9984,0 kHz Q1,Q2

Rezystory:

10k /0,25 W	R4
1k /0,25 W	R5,R6,R7,R8,R9-R13
0,5k/0,25 W	R1
400 /0,25 W	R2
100 /0,25 W	R3
33k /0,25 W	R14
470k/0,25 W	R15
6,8k/0,25 W	R16,R18
390 /o,25 W	R17

Kondensatory:

10 F	C1,C6
1 nF	C4
20 pF	C2,C3
0,1 F	C5

Złącza:

BNC 50/N1	D
881009	C
811096	A,B

Załącznik nr 10

LISTA ELEMENTOW PAKIETU PP-21.

Układy scalone:

Intel 6116	U1,U2
Intel 8253	U27
Intel 8205	U28
Intel 4731	U39(4),U40(4)
KP556PT5	U38(4)
74LS393	U3-U12
74LS245	U13-U24,U50,U51
74LS193	U34
74LS181	U29-U32
74LS174	U34,U35,U36,U33(4)
74LS151	U41(4)
74LS74	U42
74LS30	U50
74LS25	U42
74LS11	U45
74LS10	U44
74LS08	U46,U47,U48
74LS07	U43
74LS04	U49

Rezystory:

1k /0,25 W	R1,R2

Złącza:

BNC 50/N	D
811096	A,B,C

```
/* Program SOKNO.C sluzy do badania skutecznosci filtracji   *
 * logicznych i arytmetycznych operujacych na okienku "3x3". *
 *                                                           *
 *  Wywolanie programu: SOKNO plik1 plik2 par1 par2          *
 *                                                           *
 *       gdzie:                                              *
 *              plik1 - plik zawierajacy obraz do filtracji, *
 *              plik2 - plik, do ktorego jest zapisywany     *
 *                      po filtracjach,                      *
 *              par1 - parametr dla filtracji arytmetycznej, *
 *              par2 - prog komparacji.                      *
 *                                                           */

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>
#include <process.h>

#define MASK1 128;
#define MASK2 32640;
#define MASK3 127;
#define MASK4 255;
#define MASK5 0xf;
#define REC  32768;
#define GRAPH   1;
#define TEXT    0;

extern ega ( int );
extern point ( int, int, char );
extern delay();
extern circle ( int, int, int, char );
extern dcircle ( int, int, int, char );
extern clrscr();
extern line ( int, int, int, int, char );

char tabk_x[2000];
char tabk_y[2000];
int l = 0;
static int i;

char flog1 ( tb )
char *tb;
{
char a;
        a = (tb[4]&(tb[0]|tb[1]|tb[2]|tb[3]|tb[5]|tb[6]|tb[7]|tb[8]))|
((~tb[4])&(tb[0]&tb[1]&tb[2]&tb[3]&tb[5]&tb[6]&tb[7]&tb[8]));
        return ( a );
}

char flog2( tb )
char *tb;
{
char a;
        a = tb[4]&(tb[0]&tb[1]|tb[2]&tb[5]|tb[7]&tb[8]|tb[3]&tb[6])|
        tb[4]&(tb[0]|tb[1])&(tb[3]|tb[5])&(tb[7]|tb[8])&(tb[3]|tb[6]);
        return( a );
}

char fatm( tb, n )
char *tb, n;
{
char a, n1;
        a = 0;
```

```
                                tab2[3] = mbajt ( ptr1[i-1] );
                                tab1[4] = mbajt ( ptr1[i] );
                                tab2[4] = sbajt ( ptr1[i] );
                                tab1[5] = sbajt ( ptr1[i] );
                                tab2[5] = mbajt ( ptr1[i+1] );
                                tab1[6] = sbajt ( ptr1[i+127] );
                                tab2[6] = mbajt ( ptr1[i+128] );
                                tab1[7] = mbajt ( ptr1[i+128] );
                                tab2[7] = sbajt ( ptr1[i+128] );
                                tab1[8] = sbajt ( ptr1[i+128] );
                                tab2[8] = mbajt ( ptr1[i+129] );
                                }
                        }
        }

        else
        {
                if ( i >= m2 ){
                        tab1[6] = 0;
                        tab2[6] = 0;
                        tab1[7] = 0;
                        tab2[7] = 0;
                        tab1[8] = 0;
                        tab2[8] = 0;
                        if ((( i & m4 ) == m1) || (( i & m4 ) == 0 )){
                                tab1[0] = 0;
                                tab2[0] = 0;
                                tab1[3] = 0;
                                tab2[3] = 0;
                                tab1[1] = mbajt ( ptr1[i-128] );
                                tab2[1] = sbajt ( ptr1[i-128] );
                                tab1[2] = sbajt ( ptr1[i-128] );
                                tab2[2] = mbajt ( ptr1[i-127] );
                                tab1[4] = mbajt ( ptr1[i] );
                                tab2[4] = sbajt ( ptr1[i] );
                                tab1[5] = sbajt ( ptr1[i] );
                                tab2[5] = mbajt ( ptr1[i+1] );
                                }
                        else
                                {
                                if ((( i & m4 ) == m3 ) || (( i & m4 ) == m4 )){
                                        tab1[2] = 0;
                                        tab2[2] = 0;
                                        tab1[5] = 0;
                                        tab2[5] = 0;
                                        tab1[0] = sbajt ( ptr1[i-129] );
                                        tab2[0] = mbajt ( ptr1[i-128] );
                                        tab1[1] = mbajt ( ptr1[i-128] );
                                        tab2[1] = sbajt ( ptr1[i-128] );
                                        tab1[3] = sbajt ( ptr1[i-1] );
                                        tab2[3] = mbajt ( ptr1[i] );
                                        tab1[4] = mbajt ( ptr1[i] );
                                        tab2[4] = sbajt ( ptr1[i] );
                                        }
                                else
                                        {
                                        tab1[0] = sbajt ( ptr1[i-129] );
                                        tab2[0] = mbajt ( ptr1[i-128] );
                                        tab1[1] = mbajt ( ptr1[i-128] );
                                        tab2[1] = sbajt ( ptr1[i-128] );
                                        tab1[2] = sbajt ( ptr1[i-128] );
                                        tab2[2] = mbajt ( ptr1[i-128] );
                                        tab1[3] = sbajt ( ptr1[i-1] );
                                        tab2[3] = mbajt ( ptr1[i] );
                                        tab1[4] = mbajt ( ptr1[i] );
                                        tab2[4] = sbajt ( ptr1[i] );
                                        tab1[5] = sbajt ( ptr1[i] );
                                        tab2[5] = mbajt ( ptr1[i+1] );
                                        }
                                }
                        }
                else
```

```
        n1 = tb[0]+tb[1]+tb[2]+tb[3]+tb[4]+tb[5]+tb[6]+tb[7]+tb[8];
        if( n1 > n )  a = 1;
        if( n1 == n ) a = tb[4];
        return( a );
}

char kontur ( tb )
char *tb;
{
char a;
        a = tb[4]&(( char )(!( tb[1]&tb[3]&tb[4]&tb[5]&tb[7] )));
        return( a );
}

char mbajt ( c )
char c;
{
char m5 = MASK5;
        return( c & m5 );
}

char sbajt(c)
char c;
{
char m5 = MASK5;
        return(( c >> 4 ) & m5 );
}

void fbinary ( tab1, tab2, poz )
char *tab1, *tab2, poz;
{
int i;
char a, b;
unsigned int lrc = REC;
        a = 0; b = 0;
        for ( i = 0; i < lrc ; i++ ){
                if ( mbajt ( tab1[i] ) <= poz )
                        a = 1;
                else
                        a = 0;
                if ( sbajt ( tab1[i] ) <= poz )
                        b = 1;
                else
                        b = 0;
                tab2[i] = a | ( b << 4 );
                }
}

void plot ( tab, barwa )
char *tab, barwa;
{
int k, i, j;
        k = 0;
        for ( i = 0; i < 256; i++ ){
                for ( j = 0; j < 128; j++ ){
                        if( mbajt( tab[k] ) == 1)
                                point( 192 + (2 * j), 47 + i, barwa );
                        k++;
                        }
                }
        k = 0;
        for ( i = 0; i < 256; i++ ){
                for ( j = 0; j < 128; j++ ){
                        if ( sbajt ( tab[k] ) == 1)
                                point( 193 + (2 * j), 47 + i, barwa );
                        k++;
                        }
                }
}

void punkt( tab )
char *tab;
```

```
{
int k, i, j;
        k = 0; l = 0;
        for( i = 0; i < 256; i++ ){
                for( j = 0; j < 128; j++ ){
                        if( mbajt( tab[k] ) == 1 ){
                        tabk_x[l] = 2 * j;
                        tabk_y[l] = i;
                        }
                        k++; l++ ;
                }
        }
        k = 0;
        for( i = 0; i <= 255; i++ ){
                for( j = 0; j < 128; j++ ){
                        if( sbajt( tab[k] ) == 1){
                        tabk_x[l] = 2 * j + 1;
                        tabk_y[l] = i;
                        }
                        k++; l++;
                }
        }
}

extern int i;

void okienko ( tab1, tab2, ptr1 )
char *tab1, *tab2, *ptr1;
{
int m1, m2, m3, m4;
m1 = MASK1; m2 = MASK2; m3 = MASK3; m4 = MASK4;

        if ( i,< m1 ){
                tab1[0] = 0;
                tab2[0] = 0;
                tab1[1] = 0;
                tab2[1] = 0;
                tab1[2] = 0;
                tab2[2] = 0;
                if ((( i & m4 ) == m1 ) || (( i & m4 ) == 0)){
                        tab1[3] = 0;
                        tab2[3] = 0;
                        tab1[6] = 0;
                        tab2[6] = 0;
                        tab1[4] = mbajt ( ptr1[i] );
                        tab2[4] = sbajt ( ptr1[i] );
                        tab1[5] = sbajt ( ptr1[i] );
                        tab2[5] = mbajt ( ptr1[i+1] );
                        tab1[7] = mbajt ( ptr1[i+128] );
                        tab2[7] = sbajt ( ptr1[i+128] );
                        tab1[8] = sbajt ( ptr1[i+129] );
                        tab2[8] = mbajt ( ptr1[i+129] );
                        }
                else
                        {
                        if ((( i & m4 ) == m3 ) || (( i & m4 ) == m4 )){
                                tab1[5] = 0;
                                tab2[5] = 0;
                                tab1[8] = 0;
                                tab2[8] = 0;
                                tab1[3] = sbajt ( ptr1[i-1] );
                                tab2[3] = mbajt ( ptr1[i] );
                                tab1[4] = mbajt ( ptr1[i] );
                                tab2[4] = sbajt ( ptr1[i] );
                                tab1[6] = sbajt ( ptr1[i+127] );
                                tab2[6] = mbajt ( ptr1[i+128] );
                                tab1[7] = mbajt ( ptr1[i+128] );
                                tab2[7] = sbajt ( ptr1[i+128] );
                                }
                        else
                                {
                                tab1[3] = sbajt ( ptr1[i-1] );
```

```
                if (((i & m4) == m3) || ((i & m4) == m4)){
                        tab1[0] = 0;
                        tab2[0] = 0;
                        tab1[3] = 0;
                        tab2[3] = 0;
                        tab1[6] = 0;
                        tab2[6] = 0;
                        tab1[1] = mbajt ( ptr1[i-128] );
                        tab2[1] = sbajt ( ptr1[i-128] );
                        tab1[2] = sbajt ( ptr1[i-128] );
                        tab2[2] = mbajt ( ptr1[i-127] );
                        tab1[4] = mbajt ( ptr1[i] );
                        tab2[4] = sbajt ( ptr1[i] );
                        tab1[5] = sbajt ( ptr1[i] );
                        tab2[5] = mbajt ( ptr1[i+1] );
                        tab1[7] = sbajt ( ptr1[i+128] );
                        tab2[7] = mbajt ( ptr1[i+128] );
                        tab1[8] = sbajt ( ptr1[i+128] );
                        tab2[8] = mbajt ( ptr1[i+129] );
                        }
                else
                        {
                        if (((i & m4) == m3) || ((i & m4) == m3)){
                                tab1[2] = 0;
                                tab2[2] = 0;
                                tab1[5] = 0;
                                tab2[5] = 0;
                                tab1[8] = 0;
                                tab2[8] = 0;
                                tab1[0] = sbajt ( ptr1[i-129] );
                                tab2[0] = mbajt ( ptr1[i-128] );
                                tab1[1] = mbajt ( ptr1[i-128] );
                                tab2[1] = sbajt ( ptr1[i-128] );
                                tab1[3] = sbajt ( ptr1[i-1] );
                                tab2[3] = mbajt ( ptr1[i] );
                                tab1[4] = mbajt ( ptr1[i] );
                                tab2[4] = sbajt ( ptr1[i] );
                                tab1[6] = sbajt ( ptr1[i+127] );
                                tab2[6] = mbajt ( ptr1[i+128] );
                                tab1[7] = mbajt ( ptr1[i+128] );
                                tab2[7] = sbajt ( ptr1[i+128] );
                                }
                        else
                                {
                                tab1[0] = sbajt ( ptr1[i-129] );
                                tab2[0] = mbajt ( ptr1[i-128] );
                                tab1[1] = mbajt ( ptr1[i-128] );
                                tab2[1] = sbajt ( ptr1[i-128] );
                                tab1[2] = sbajt ( ptr1[i-128] );
                                tab2[2] = mbajt ( ptr1[i-127] );
                                tab1[3] = sbajt ( ptr1[i-1] );
                                tab2[3] = mbajt ( ptr1[i] );
                                tab1[4] = mbajt ( ptr1[i] );
                                tab2[4] = sbajt ( ptr1[i] );
                                tab1[5] = sbajt ( ptr1[i] );
                                tab2[5] = mbajt ( ptr1[i+1] );
                                tab1[6] = sbajt ( ptr1[i+127] );
                                tab2[6] = mbajt ( ptr1[i+128] );
                                tab1[7] = mbajt ( ptr1[i+128] );
                                tab2[7] = sbajt ( ptr1[i+128] );
                                tab1[8] = sbajt ( ptr1[i+128] );
                                tab2[8] = mbajt ( ptr1[i+129] );
                                }
                        }
                }
        }
}

main ( argc, argv )
int argc;
char *argv[];
```

```
                ,---",
                }
        delay();
        circle( 128, 128, 70, 4 );
        delay();
                if ( fwrite ( alloc2, 1, rc, f2 ) !=  rc )
                        perror ("Write");
                fclose( f1 );
                fclose( f2 );
        ega( tx );
        system( "CLS" );
}
```

```
)
```

```
/*  PROCEDURY GRAFICZNE */


#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <math.h>
#include <malloc.h>

void ega( mode )
int mode;
{
union REGS regist;
        if ( mode ){
                regist.h.ah = 0;
                regist.h.al = 0x10;
                int86 ( 0x10, &regist, &regist );
                }
        else
                {
                regist.h.ah = 0;
                regist.h.al = 3;
                int86 ( 0x10, &regist, &regist );
                }
}

void point ( x, y, kolor )
char kolor;
int x, y;
{
union   REGS    regist;
                regist.h.ah = 0xC;
                regist.h.bh = 0;
                regist.x.dx = y;
                regist.x.cx = x;
                regist.h.al = kolor;
                int86 ( 0x10, &regist, &regist );
}

void delay()
{
        getchar();
}

void circle ( x, y, r, barwa )
int x, y, r;
char barwa;
{
int i, length, clength;
int x1,srodek, *tab1, *tab2, *tab3, *tab4;
x1 = 0; i = 0;
        tab1 = (int *)malloc (500 * sizeof ( int ));
        tab2 = (int *)malloc (500 * sizeof ( int ));
        tab3 = (int *)malloc (2000 * sizeof ( int ));
        tab4 = (int *)malloc (2000 * sizeof ( int ));
        srodek = (int)(ceil ( sqrt ((double)(( r * r ) / 2 ))));
        while ( ++x1 <= srodek ){
                tab1[i] = x1;
                tab2[i] = (int)(ceil ( sqrt ((double)(( r * r ) - ( x1 * x1 )))));
                i++;
                }
        if ( tab2[srodek ] == srodek ){
                for ( i = 0; i < srodek; i++ ){
                        tab1[(2 * srodek) - i - 2] = tab2[i];
                        tab2[(2 * srodek) - i - 2] = tab1[i];
```

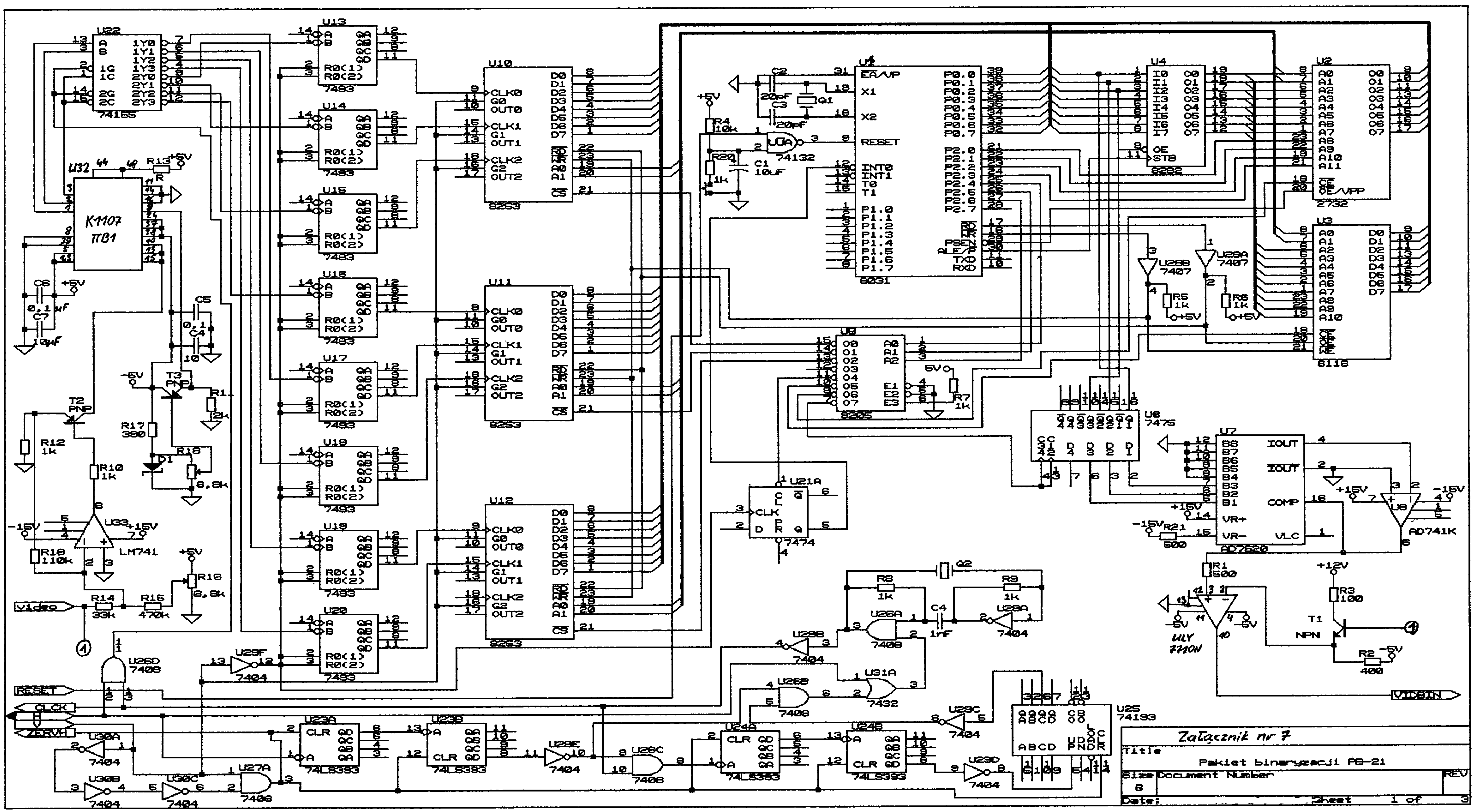
```
                length = 2 * srodek - 1;
                }
        else
                {
                for ( i = 0; i < srodek; i++ ){
                        tab1[(2 * srodek) - i - 1] = tab2[i];
                        tab2[(2 * srodek) - i - 1] = tab1[i];
                        }
                length = 2 * srodek;
                }
        for ( i= 0; i < length; i++ ){
                tab3[i] = x + tab1[i];
                tab4[i] = y + tab2[i];
                tab3[length + i] = x + tab1[i];
                tab4[length + i] = y - tab2[i];
                tab3[(2 * length) + i] = x - tab1[i];
                tab4[(2 * length) + i] = y - tab2[i];
                tab3[(3 * length) + i] = x - tab1[i];
                tab4[(3 * length) + i] = y + tab2[i];
                }
        clength = 4 * length;
        tab3[clength] = x;
        tab4[clength] = y + r;
        tab3[clength + 1] = x + r;
        tab4[clength + 1] = y;
        tab3[clength + 2] = x;
        tab4[clength + 2] = y - r;
        tab3[clength + 3] = x - r;
        tab4[clength + 3] = y;
        for ( i = 0; i < clength + 4; i++ )
                point ( 192 + tab3[i] , 302 - tab4[i], barwa );
}

void dcircle ( x, y, r, color )
int x, y, r;
char color;
{
int i, l, dl, cl,length, clength;
int x1,srodek, *tab1, *tab2, *tab3, *tab4, *tab5, *tab6;
x1 = 0; i = 0;
        tab1 = (int *)malloc (500 * sizeof ( int ));
        tab2 = (int *)malloc (500 * sizeof ( int ));
        tab3 = (int *)malloc (2000 * sizeof ( int ));
        tab4 = (int *)malloc (2000 * sizeof ( int ));
        tab5 = (int *)malloc (2000 * sizeof ( int ));
        tab6 = (int *)malloc (2000 * sizeof ( int ));
        srodek = (int)(ceil ( sqrt ((double)(( r * r ) / 2 ))));
        while ( ++x1 <= srodek ){
                tab1[i] = x1;
                tab2[i] = (int)(ceil ( sqrt ((double)(( r * r ) - ( x1 * x1 )))));
                i++;
                }
        l = 0;
        for ( i = 0; i < srodek - 1; i++ ){
                if ( tab2[i] != tab2[i + 1] ){
                        tab3[l] = tab1[i];
                        tab4[l] = tab2[i];
                        l++;
                        tab3[l] = tab1[i] + 1;
                        tab4[l] = tab2[i];
                        l++;
                        }
                else
                        {
                        tab3[l] = tab1[i];
                        tab4[l] = tab2[i];
                        l++;
                        }
                }
        tab3[ l ] = tab1[srodek - 1];
        tab4[ l ] = tab2[srodek - 1];
```

```
                tab3[(2 * l) + i -i] = tab4[i];
                tab4[(2 * l) + i -i] = tab3[i];
                }
        tab3[2 * l + 2] = srodek;
        tab4[2 * l + 2] = srodek;
        dl = 2 * l + 3;
        for ( i= 0; i < dl ; i++ ){
                tab5[i] = x + tab3[i];
                tab6[i] = y + tab4[i];
                tab5[dl + i] = x + tab3[i];
                tab6[dl + i] = y - tab4[i];
                tab5[2 * dl + i] = x - tab3[i];
                tab6[2 * dl + i] = y - tab4[i];
                tab5[3 * dl + i] = x - tab3[i];
                tab6[3 * dl + i] = y + tab4[i];
                }
        cl = 4 * dl;
        tab5[cl] = x;
        tab6[cl] = y + r;
        tab5[cl + 1] = x + r;
        tab6[cl + 1] = y;
        tab5[cl + 2] = x;
        tab6[cl + 2] = y - r;
        tab5[cl + 3] = x - r;
        tab6[cl + 3] = y;
        for ( i = 0; i < cl + 4; i++ )
                point ( 192 + tab5[i] , 302 - tab6[i], color );
}

void clrscr()
{
int i, j;
        for ( i = 0; i < 256; i++ ){
                for ( j = 0; j < 256; j++ )
                        point ( j + 192, 47 + i, 0 );
                }
}

void line ( x1, y1, x2, y2, barwa )
int x1,y1,x2,y2;
char barwa;
{
int x, y, xstep, ystep, kierx, dx;
        x = x1; y = y1; xstep = 1; ystep = 1;
        if( x1 > x2 ) xstep = -1;
        if( y1 > y2 ) ystep = -1;
        dx = abs ( x2 - x1 );
        kierx = dx ? 0 : -1;
        while ( ! (( x == x2 ) && ( y == y2 ))){
                point ( x, y, barwa );
                if ( kierx < 0 )
                        y += ystep;
                else
                        x += xstep;
                }
}
```


Załącznik nr 7
Title
Pakiet binaryzacji PB-21
Size Document Number
B
REV
Date:
Sheet 1 of 3