AKADEMIA GÓRNICZO-HUTNICZA
W KRAKOWIE
WYDZIAŁ ELEKTROTECHNIKI AUTOMATYKI
INFORMATYKI I ELEKTRONIKI
KATEDRA AUTOMATYKI

Krzysztof Szawłowski

Rejestrator holterowski z pamięcią cyfrową

Praca magisterska napisana
w Katedrze Automatyki pod
kierunkiem Pana
dr inż. Piotra Augustyniaka

Kraków 2001

Spis treści

1. Wstęp

Diagnostyka medyczna stawia sobie za cel wczesne rozpoznawanie różnych zaburzeń w organizmie człowieka. Od prawidłowego rozpoznania, które umożliwia zastosowanie właściwej terapii zależy los chorego. Ten istotny i trudny problem obecnie jest rozwiązywany poprzez zastosowanie coraz doskonalszej aparatury pomiarowo - rejestrującej ułatwiającej lekarzowi postawienie właściwej diagnozy. Aparaturze medycznej stawiane są bardzo wysokie wymagania związane z niezawodnością działania, bezpieczeństwem badanego pacjenta oraz możliwością w miarę jednoznacznej i prostej interpretacji zebranych podczas badania wyników pomiarów. Oznacza to konieczność wykorzystania przy konstruowaniu przyrządów pomiarowych aparatury medycznej szerokiej wiedzy z zakresu między innymi medycyny, elektroniki i informatyki. Zastosowanie najnowocześniejszych osiągnięć z zakresu tych dziedzin wiedzy sprawia, że projektowanie aparatury diagnostycznej o przeznaczeniu medycznym stało się trudną i bardzo interesującą dziedziną techniki. Zmusza to do tworzenia zespołów i ośrodków badawczych zajmujących się opracowywaniem takiej aparatury.

Przedstawione w niniejszej pracy urządzenie będące rejestratorem biopotencjałów generowanych przez serce przeznaczone jest do badania zaburzeń w jego funkcjonowaniu.

Serce, podobnie jak inne narządy zbudowane z tkanki mięśniowej, cechuje bioelektryczność, czyli zdolność do generowania potencjałów elektrycznych, które powstają w wyniku procesów życiowych tkanki mięśniowej. Rozrusznikiem, który inicjuje w sercu bioprądy, jest węzeł zatokowo - przedsionkowy. Rytmiczne efekty elektryczne, towarzyszące każdemu cyklowi pracy serca, mogą być zapisywane za pomocą urządzenia zwanego

elektrokardiografem. Badania elektrokardiograficzne są cenną pomocą w ustalaniu stanu serca. Istota tych badań polega głównie na rejestracji bioprądów sercowych, czyli potencjałów czynnościowych serca, które rozprzestrzeniają się przez układ przewodzący w całym sercu i pobudzają je do rytmicznych skurczów. Dzięki dobremu przewodnictwu elektrycznemu płynów ustrojowych i tkanek, bioprądy sercowe mogą być rejestrowane za pośrednictwem elektrod przyłożonych do różnych miejsc ciała. Odbierane przez te elektrody potencjały elektryczne są często rejestrowane na taśmie papierowej w postaci elektrokardiogramu (krzywej ekg), złożonej z zespołu tzw. załamków i odcinków. Dla pełniejszej oceny bioelektrycznej czynności serca analizujemy zwykle kilkanaście zapisów krzywych ekg, zarówno z odprowadzeń kończynowych, jak i przedsercowych, dających pośredni wgląd w poszczególne okolice i fazy pracy serca. Analiza tych zapisów daje wiele informacji o stanie serca i ewentualnych zmianach chorobowych. Na podstawie elektrokardiogramu, który może być rejestrowany jako spoczynkowy, wysiłkowy lub po określonych lekach, ocenia się także rytm akcji serca (przyspieszenie, zwolnienie, niemiarowość), zaburzenia przewodnictwa bodźców, czyli tzw. zawał mięśnia sercowego oraz tzw. przeciążenia i przerosty lub inne zmiany mięśniowe.

Funkcjonujące serce człowieka można traktować jako duży dipol. Przez znaczny czas trwania okresu depolaryzacji i repolaryzacji mięśnia serca biegun ujemny takiego dipola zwrócony jest w zasadzie ku górnej części serca (zazwyczaj skierowany jest w stronę barku prawego), biegun dodatni zaś ku części dolnej. Ponieważ wypadkowy dipol powstający w czasie cyklu sercowego jest równoważny z siłą elektromotoryczną, która ma określoną wartość, kierunek i zwrot, dlatego też może być on przedstawiony za pomocą wektora. Grot wektora wskazuje biegun dodatni dipola. Jeżeli taki dipol znajduje się w jednorodnym przewodniku objętościowym, to wówczas

powstanie pole elektryczne. Rozkład linii izopotencjalnych na klatce piersiowej człowieka nie jest jednak tak symetryczny jak w jednorodnym przewodniku objętościowym, gdyż tkanki ciała ludzkiego nie są jednorodnym przewodnikiem. W celu łatwiejszej interpretacji elektrokardiogramów przyjmuje się szereg uproszczeń, między innymi zakłada się, że: przewodnictwo tkankowe jest jednorodne, serce ma centralne położenie w klatce piersiowej. Zaakceptowanie tych uproszczonych założeń, które poczynił już w 1903 r. Willem Einthoven - twórca nowoczesnej elektrokardiografii spełnia pożyteczną rolę w praktycznej elektrofizjologii klinicznej. Zsumowanie potencjałów czynnościowych pojedynczych włókien tworzących mięsień sercowy wytwarza wokół serca pole elektryczne o zmieniającej się w trakcie poszczególnych faz cyklu sercowego lokalizacji źródła prądu i zmieniającym się napięciu. Prądy powstające podczas czynności serca rozprzestrzeniają się we wszystkich kierunkach i docierają do powierzchni ciała. Rozkład lini izopotencjalnych na tułowiu zależy od fazy cyklu sercowego, położenia serca, typu budowy ciała. Potencjał elektryczny spada w miarę oddalania się od biegunów. Połączenie przewodnikiem dwu różnych linii izopotencjalnych wywołuje przepływ prądu, który rejestruje się elektrokardiografem. Zarejestrowane zmiany potencjału wywołane czynnością serca nazywa się elektrokardiogramem.

Obecnie znane są liczne rodzaje odprowadzeń wykorzystywanych do monitorowania elektrokardiograficznego (monitorowania ekg). Najczęściej stosowane są dwubiegunowe odprowadzenia piersiowe (zwane przedsercowymi), których elektrody przykleja się do skóry w odpowiednich punktach klatki piersiowej. Kształt i amplituda załamków krzywych ekg zarejestrowanych w poszczególnych odprowadzeniach piersiowych zależą od rodzaju zastosowanego odprowadzenia i od wielkości rzutów chwilowych wektorów serca na oś odprowadzenia. Wybór rodzaju odprowadzenia zależy

od celu badania elektrokardiograficznego. Główne kliniczne wskazania do zastosowania monitorowania ekg to:

- ustalenie rozpoznania zaburzeń rytmu serca i zaburzeń przewodzenia,
- diagnostyka przejściowego niedotlenienia mięśnia sercowego,
- ocena skuteczności leczenia antyarytmicznego,
- ocena skuteczności leczenia przeciwdusznicowego,
- ocena funkcjonowania sztucznych rozruszników serca.

Odmianą elektrokardiografii jest tzw. **elektrokardiografia ambulatoryjna** (dynamiczna). Obejmuje ona:

- ciągłą, najczęściej dobową, rejestrację ekg metodą **Holtera,**
- ciągłą lub okresową transmisję ekg drogą radiową,
- okresową transmisję ekg przez telefon,
- okresową rejestrację ekg w dowolnych lub zaprogramowanych odstępach czasu.

Przedstawione w niniejszej pracy urządzenie spełnia wszystkie wymagania stawiane przez elektrokardiografię ambulatoryjną. Jest to rejestrator holterowski umożliwiający rejestrację ciągłą pracy serca o czasie trwania ograniczonym jedynie zastosowaną pojemnością pamięci RAM. Zastosowanie w urządzeniu standardowego portu szeregowego umożliwia współpracę rejestratora z dowolnym modemem umożliwiającym ciągłą transmisję danych na odległość. Urządzenie to, dzięki zastosowaniu mikroprocesora, umożliwia również przeprowadzanie okresowych rejestracji ekg w dowolnych odstępach czasu. Zaprogramować można zarówno dowolny czas rejestracji jak i jej długość. Do zaprojektowania rejestratora wykorzystano nowoczesne układy scalone, co umożliwiło stworzenie urządzenia przenośnego, o niewielkich wymiarach i wadze, zasilanego

bateryjnie. Zaawansowane oprogramowanie dla komputera klasy PC umożliwia przeprowadzanie badania elektrokardiograficznego w czasie rzeczywistym, jak też łatwą i szybką analizę zgromadzonych wcześniej rejestracji. Ze względu na wymienione powyżej właściwości rejestratora holterowskiego z pamięcią cyfrową możliwe jest stworzenie na jego bazie urządzenia w pełni funkcjonalnego, przeznaczonego na szeroki rynek aparatury medycznej. Sprzyja temu obserwowany obecnie rozwój technologii w zakresie pojemnych pamięci cyfrowych o niskich napięciach zasilania, umożliwiających znaczne wydłużenie czasu rejestracji urządzenia i zachowanie niskiego poboru mocy.

2. Opis rejestratora holterowskiego

Rejestrator holterowski z pamięcią cyfrową jest urządzeniem przeznaczonym do diagnostyki medycznej w elektrokardiografii. Może mieć on zastosowanie zarówno w elektrokardiografii klinicznej, jak też w tzw. elektrokardiografii ambulatoryjnej. Przy pomocy tego urządzenia dokonuje się w sposób ciągły lub okresowy monitorowania elektrokardiograficznego (monitorowania ekg), czyli rejestracji zmian potencjału wywołanych czynnością serca. Dane pomiarowe zebrane przez rejestrator mogą być wysyłane w czasie rzeczywistym bezpośrednio do komputera lub zapisywane w wewnętrznej pamięci cyfrowej. W przypadku pierwszym komputer klasy PC z zainstalowanym systemem operacyjnym Windows i programem rejestratora com232.exe spełnia funkcję monitora wyświetlając krzywą ekg. Współpraca urządzenia z komputerem jest w pełni automatyczna, co oznacza możliwość sterowania rejestratorem z poziomu aplikacji. Sygnał pobrany z elektrod jest wzmacniany i filtrowany w celu wyeliminowania zakłóceń oraz poddawany kwantyzacji w przetworniku analogowo-cyfrowym. Następnie dane z przetwornika poddawane są konwersji różnicowej i przesyłane przez łącze szeregowe standardu RS-232. Zostają one przetworzone cyfrowo w programie com232.exe. Umożliwia to kompensację składowej stałej i wyświetlanie elektrokardiogramu linią ciągłą. Możliwe jest także wybranie dokładności odwzorowania przebiegu przez zastosowanie różnych częstotliwości próbkowania przetwornika A/C. W przypadku drugim urządzenie dokonuje również konwersji różnicowej sygnału wejściowego i zapisuje dane pomiarowe w wewnętrznej pamięci RAM. Parametry pracy takie jak: częstotliwość próbkowania, czas rozpoczęcia i trwania rejestracji należy zaprogramować wcześniej. Programowania można dokonać z poziomu interfejsu urządzenia, które wyposażone zostało w klawiaturę i wyświetlacz

alfanumeryczny lub z poziomu aplikacji com232.exe. W tym przypadku przesyłanie parametrów pracy odbywa się przez wybrany w opcjach port szeregowy komputera. Urządzenie umożliwia zaprogramowanie do pięćdziesięciu następujących po sobie rejestracji o różnych czasach trwania. Po dokonaniu rejestracji dane przesyłane są do komputera i zapisywane do pliku binarnego w specjalnym formacie. Umożliwia to zapamiętanie zarówno danych, jak też wybranych ustawień i parametrów urządzenia. Podczas wyświetlania dane są poddawane również obróbce cyfrowej.

Szczegółowy opis działania programu jest przedstawiony w rozdziale "Opis programu com232.exe". Do niniejszej pracy dołączony został również w dodatku B kod źródłowy tego programu.

Sercem urządzenia jest mikroprocesor AT89C52 sterujący wszystkimi jego funkcjami. Mikroprocesor spełnia między innymi rolę zegara czasu rzeczywistego, co umożliwia wykonywanie rejestracji o określonej wcześniej godzinie.

Urządzenie może być zasilane napięciem stabilizowanym 9V lub baterią.

Schemat blokowy rejestratora holterowskiego z pamięcią cyfrową został przedstawiony na rysunku 1. Poszczególne bloki to:

1. Stopień wejściowy
2. Układ filtrów
3. Układ przetwornika A/C
4. Układ sterownika mikroprocesorowego
5. Układ wyświetlacza LCD
6. Układ klawiatury sterującej
7. Interfejs komunikacji szeregowej z komputerem
8. Blok zasilania

Przeznaczenie i funkcje wszystkich bloków urządzenia opisane są poniżej a osiągnięte parametry techniczne w rozdziale "Parametry techniczne".

2.1 Stopień wejściowy

Stopień wejściowy (rys. 2) zawiera wzmacniacz pomiarowy oraz wzmacniacz sygnału sprzężenia zwrotnego (ang. Right-Leg Drive). Wzmacniacz pomiarowy U11 zbudowany został na bazie układu scalonego INA118 firmy BURR-BROWN. Jest to wzmacniacz różnicowy przeznaczony między innymi do wykorzystania w urządzeniach medycznych. Charakteryzuje się niskim poborem mocy i dużym wzmocnieniem. Pojedynczy zewnętrzny rezystor umożliwia osiągnięcie wystarczająco dużego wzmocnienia układu przy jednoczesnym zapewnieniu doskonałych parametrów pracy. Do wejść układu U11 doprowadzone zostały elektrody RA (prawa) i LA (lewa)

Jednym z wymagań stawianych wzmacniaczom wejściowym aparatury medycznej jest duży współczynnik CMMR. Współczynnik ten jest miarą stopnia symetrii wzmacniacza różnicowego i jego zdolności do eliminowania sygnału sumacyjnego. Zastosowany układ spełnia powyższy warunek, gdyż charakteryzuje się bardzo wysokim współczynnikiem CMMR dla dużych wzmocnień sygnału różnicowego.

Układ wzmacniacza wejściowego nie wymagał zastosowania kompensacji składowej stałej z uwagi na wysoką rozdzielczość przetwornika analogowo - cyfrowego. Dane pomiarowe otrzymywane z przetwornika przed wysłaniem do komputera poddawane są konwersji różnicowej, która eliminuje z sygnału cyfrowego tą składową. Natomiast zmiany składowej stałej eliminowane są w programie com232.exe. Umożliwia to uaktywnienie opcji ‘Kompensacja składowej stałej’.

Sygnał sprzężenia zwrotnego wzmacniacza pomiarowego U11 zostaje następnie wielokrotnie wzmocniony i odwrócony w fazie we wzmacniaczu

operacyjnym U10B zbudowanym na bazie układu OPA2604AP. Tak wzmocniony sygnał podany na elektrodę obojętną RL zwiększa stosunek sygnału użytecznego do szumu. Umożliwia to zastosowanie mniejszych wzmocnień w układzie wzmacniacza pomiarowego U11.

Ten sam sygnał sprzężenia zwrotnego podawany jest również poprzez wtórnik U10A na ekran kabli odprowadzających sygnał z elektrod RA i LA.
Powoduje to zmniejszenie wartości zakłóceń zewnętrznych indukowanych w kablach przyłączeniowych.

2.2 Układ filtrów

W tym module rejestratora holterowskiego z pamięcią cyfrową (rys. 3) znajduje się filtr analogowy dolnoprzepustowy. Na jego wejście doprowadzony został sygnał z wyjścia wzmacniacza pomiarowego znajdującego się w stopniu wejściowym urządzenia. Jest to filtr aktywny czwartego rzędu zbudowany z dwóch ogniw drugiego rzędu. Filtr ten ma za zadanie stłumić wszystkie sygnały zakłócające sygnał pomiarowy, które pochodzą od pracujących urządzeń elektrycznych. Tłumione powinny być zarówno zakłócenia o częstotliwości 50 Hz powstające w sieci energetycznej, jak też wszelkiego rodzaju sygnały radiowe o wysokich częstotliwościach. Do zadań filtru należy również tłumienie wszystkich sygnałów wynikających z elektrycznej aktywności organizmu człowieka o częstotliwościach powyżej granicznej.
Ogniwa podstawowe filtru zbudowane są w oparciu o wzmacniacze operacyjne OPA2604AP. Wykorzystany układ scalony zawiera w swojej strukturze dwa identyczne wzmacniacze oznaczone na schemacie jako U9A i U9B. Obydwa ogniwa cechuje maksymalnie płaska charakterystyka a częstotliwość graniczna wynosi fg = 25 Hz. Podstawowe ogniwo to filtr o

wielokrotnym ujemnym sprzężeniu zwrotnym. Zaletą tego filtru jest to, że jego dobroć jest mało wrażliwa na skończoną wartość wzmocnienia wzmacniacza operacyjnego oraz na zmiany wartości zastosowanych elementów R i C. Dzięki temu stosunkowo łatwo można było uzyskać założone parametry filtru takie jak wzmocnienie i częstotliwość graniczną. Jednocześnie możliwe było zastosowanie do budowy urządzenia standardowych elementów elektronicznych. Oznacza to, że zmiany wartości rezystorów i kondensatorów spowodowane montażem (wysoką temperaturą podczas lutowania) oraz starzeniem się podzespołów (co dotyczy przede wszystkim kondensatorów) nie wpłyną istotnie na parametry filtru.

Zastosowany do budowy filtru niskoszumny podwójny wzmacniacz operacyjny OPA2604AP zapewnił uzyskanie założonych wcześniej parametrów częstotliwościowych. Charakterystyka filtru zastosowanego w torze pomiarowym rejestratora przedstawiona została na rysunku 4. Podczas pomiarów amplituda sygnału wejściowego wynosiła 0.5 V. Uruchomienie układu wymagało dokonania strojenia filtrów przy pomocy potencjometrów P1 i P2. Wzmocnienie wzmacniacza operacyjnego w ogniwie pierwszym wynosi 1. Sygnał poddany filtracji w tym ogniwie pozbawiony zostaje większości zakłóceń, co powoduje zwiększenie współczynnika sygnału użytecznego do szumu. Natomiast w ogniwie drugim sygnał pomiarowy jest jednocześnie filtrowany i wzmacniany jeszcze dziesięciokrotnie.

2.3 Układ przetwornika A/C

W tym bloku (rys. 5) znajduje się przetwornik analogowo-cyfrowy. Jest to ważny element toru pomiarowego, w którym następuje kwantyzacja wzmocnionego wcześniej sygnału bioelektrycznego. W urządzeniu zastosowany został przetwornik scalony ADS7808P firmy BURR-BROWN o rozdzielczości 12 bitów. Na schemacie posiada on oznaczenie U7.

Sygnał pomiarowy doprowadzony jest do wejścia R1IN przez opornik R5. Układ posiada wewnętrzne źródło napięcia referencyjnego a zastosowane elementy zewnętrzne R i C umożliwiają przetwarzanie sygnałów wejściowych o napięciach w zakresie ± 10 V. Przetwornik wymaga jednego napięcia zasilania + 5 V, jednak w celu osiągnięcia wysokiej jakości i dokładności przetwarzania należało zastosować oddzielne zasilacze dla części analogowej i cyfrowej. Konieczne było także blokowanie końcówek napięć zasilających odpowiednimi kondensatorami.

Układ charakteryzuje się niskim poborem mocy predysponującym go do zastosowań w urządzeniach przenośnych zasilanych bateryjnie. Posiada niezwykle użyteczną funkcję przełączania w stan obniżonego poboru mocy, która może być uaktywniona poprzez ustawienie wysokiego poziomu logicznego na końcówce 18 układu (PWRD). Według danych katalogowych zużywana wtedy przez układ moc zmniejsza się ze 100 mW do 50 μW. Rejestrator holterowski z pamięcią cyfrową wykorzystuje tą funkcję przetwornika oraz dodatkowo sygnalizuje jej włączenie lub wyłączenie za pomocą diody LED umieszczonej na obudowie. Podczas normalnej pracy urządzenia czerwona dioda jest zgaszona (DATA READ na rys. 5). Zostaje ona zapalona wyłącznie w czasie zbierania danych pomiarowych. Może to nastąpić zarówno podczas przełączenia rejestratora w tryb monitora ekg, jak też podczas automatycznej rejestracji według zaprogramowanych wcześniej w pamięci czasów.

Ważną cechą układu ADS7808, która zadecydowała również o jego zastosowaniu w przedstawianym urządzeniu, jest szeregowy interfejs cyfrowy. Umożliwia on prostą komunikację z mikroprocesorem przy wykorzystaniu niewielkiej liczby linii sterujących, co w sposób znaczny uprościło konstrukcję rejestratora. Oprócz wymienionego powyżej sygnału PWRD mikroprocesor do sterowania układem używa sygnałów CS, RC oraz

DCLK. Linia CS służy do wyboru (selekcji) układu podczas operacji konwersji i odczytu danych. Linia RC natomiast służy do wyboru trybu pracy przetwornika. Umożliwia rozpoczęcie konwersji analogowo-cyfrowej a następnie odczyt wyniku. Trzecia linia to zewnętrzny sygnał zegarowy generowany przez mikroprocesor. Przetwornik ADS7808 przystosowany jest do pracy z zegarem wewnętrznym, którego przebieg posiada ustalone parametry czasowe. Do sygnalizacji zakończenia konwersji i rozpoczęcia wysyłania bitów danych układ scalony używa linii SYNC. Może on również przesyłać dane przy wykorzystaniu zewnętrznego sygnału zegarowego. W urządzeniu zastosowano drugi sposób, gdyż umożliwia on dopasowanie prędkości transmisji danych z przetwornika do parametrów pracy mikroprocesora. Podczas odczytu danych generuje on sygnał zegarowy na linii DCLK

Przy wykorzystaniu taktowania zewnętrznym sygnałem zegarowym dane cyfrowe mogą być odczytywane z przetwornika na dwa sposoby. Pierwszy polega na odczycie kolejnych bitów danych podczas konwersji analogowo-cyfrowej. Drugi, wykorzystany w rejestratorze, polega na odczycie danych dopiero po wykonaniu konwersji. Taki sposób odczytu jest podyktowany koniecznością zapewnienia prawidłowego funkcjonowania poszczególnych modułów programowych sterownika AT89C51. Mikroprocesor spełnia w urządzeniu kilka niezależnych funkcji wykorzystujących jego system przerwań. System ten może przyjmować zgłoszenia obsługujące zegar czasu rzeczywistego, interfejs portu szeregowego i procedurę odczytu danych z przetwornika A/C. Umieszczenie przerwań na danym poziomie priorytetu decyduje o możliwości przerywania programów obsługi innych przerwań. Odczyt danych dopiero po zakończeniu konwersji pozwala na przerywanie procedury odczytu danych z przetwornika bez zakłócania pracy urządzenia. Ma to decydujący wpływ między innymi na

dokładność zegara czasu rzeczywistego. Opis procedur obsługi przerwań i ich zależności znajduje się w rozdziale 3.1 "Opis programu sterownika AT89C51".

Dane po konwersji analogowo-cyfrowej odczytywane są przez mikroprocesor z linii DATA. Przebiegi czasowe sygnałów sterujących przetwornikiem przedstawione zostały na rys. 6. Sterownie rozpoczyna się od ustawienia niskiego stanu logicznego na wejściu RC oznaczającego przełączenie układu w tryb konwersji. Jej rozpoczęcie powoduje opadające zbocze sygnału CS. Przetwornik sygnalizuje trwanie konwersji niskim stanem logicznym na linii BUSY, którą mikroprocesor testuje. Po maksymalnie ośmiu mikrosekundach pojawia się sygnał zakończenia przetwarzania. Jest nim zmiana poziomu logicznego końcówki BUSY na wysoki. Od tego momentu rozpoczyna się szeregowy odczyt danej pomiarowej. Mikroprocesor generuje sygnał zegarowy na linii DCLK a kolejne bity są wysyłane przez przetwornik linią DATA w kolejności od najstarszego do najmłodszego. Odebrane dane pomiarowe przechowywane są w postaci słowa 16-bitowego a następnie poddawane konwersji różnicowej, za którą odpowiedzialna jest jedna z procedur programu obsługi odczytu danych z przetwornika. Wynikiem jest 8-bitowa wartość przesyłana następnie do komputera lub zapisywana w pamięci statycznej RAM

2.4 Układ wyświetlacza LCD

Moduł wyświetlacza alfanumerycznego LCD (rys. 7) umożliwia komunikację urządzenia z użytkownikiem. Zapewnia on kontrolę nad wszystkimi funkcjami rejestratora holterowskiego. Pozwala na :

- ustawienie zegara czasu rzeczywistego,
- programowanie i kontrolę czasów rejestracji,
- ustawienie parametrów portu szeregowego,

■ ustawienie częstotliwości próbkowania przetwornika A/C,

■ testowanie pamięci cyfrowej RAM.

Po przełączeniu rejestratora w tryb automatycznej rejestracji wyświetlacz ten spełnia funkcję zegara. Dokładny opis sposobu programowania urządzenia znajduje się w rozdziale 3.1 “Opis programu sterownika AT89C52”.

W celu zapewnienia prostoty obsługi w rejestratorze zastosowano wyświetlacz LCD ze standardowym interfejsem zrealizowanym na bazie układu HD44780. Zapewnia on bezpośrednią komunikację z mikroprocesorem za pomocą trzech linii sterujących i czterech lub ośmiu linii danych. Posiada wewnętrzną pamięć zawierającą wzorce znaków alfanumerycznych i symboli graficznych. Największą zaletą tego wyświetlacza jest umieszczona w nim pamięć statyczna typu RAM (CG RAM - Charakter Generator RAM), dająca możliwość stworzenia i zaprogramowania własnych znaków graficznych. Dzięki tej funkcji możliwe było przystosowanie wyświetlacza do pracy z polskimi znakami diakrytycznymi.

Moduł LCD wymaga zastosowania dwóch napięć zasilających. Interfejs cyfrowy wyświetlacza wykorzystuje napięcie Vcc o wartości +5V. Napięcie Vee o wartości z zakresu od 0 do -5 V zasila matrycę LCD i umożliwia regulację kontrastu. Uzyskiwane jest przez zastosowanie układu scalonego oznaczonego na schemacie jako U5. Jest to prosty konwerter napięciowy z przełączanymi pojemnościami LCM7660IN umożliwiający inwersję napięcia wejściowego z zakresu 1.5 ÷ 10 V.

Do sterowania wyświetlaczem przeznaczone są trzy linie sygnałowe: RS, R/W i E. Linia RS o wysokim stanie logicznym (H) informuje wyświetlacz, że zapisywane dane mają być traktowane jako instrukcje sterujące jego pracą. Jeżeli na linii ustawiony zostanie niski stan logiczny (L), to zapisywane bajty umieszczane będą w wewnętrznej pamięci i wyświetlane.

Linia R/W w stanie H informuje moduł wyświetlacza, że dane mają być zapisywane do pamięci RAM. Gdy znajduje się ona w stanie L możliwy jest odczyt wcześniej zapisanych bajtów. Trzeci sygnał oznaczony jako E wyzwala operacje odczytu lub zapisu, informując jednocześnie o poprawności bitów danych ustawionych na liniach DB0 ÷ DB7.

Przedstawiony powyżej standardowy interfejs wyświetlacza wymagał przystosowania do współpracy z mikroprocesorem AT89C52. W tym celu został skonstruowany konwerter logiczny umożliwiający wykorzystanie linii mikroprocesora przeznaczonych do komunikacji z zewnętrzną pamięcią RAM. Dzięki takiemu rozwiązaniu znacznie ograniczono liczbę końcówek układu koniecznych do współpracy z wyświetlaczem. Konwerter logiczny (rys. 7) zbudowany został z trzech standardowych bramek typu NAND znajdujących się w układzie cyfrowym 74HCT00 (U2). Umożliwia on zapis lub odczyt danych z modułu wyświetlacza przez zastosowanie prostych rozkazów przesłań dotyczących zewnętrznej pamięci danych mikroprocesora. Zapisu danych do wyświetlacza dokonuje procedura:

```
SETB WRS
CLR WRW
SETB WCS
MOV A,CHAR
MOVX @R0,A
CLR WCS
```

Natomiast odczyt możliwy jest w sposób następujący:

```
SETB WRS
SETB WRW
SETB WCS
MOVX A,@R0
MOV CHAR,A
CLR WCS
```

Przedstawione powyżej procedury realizują zapis lub odczyt danych z pamięci wyświetlania DDRAM (Display Data RAM). Linie WRS i WRW pełnią identyczną funkcję jak przedstawione wcześniej RS i R/W. Sygnał WCS uaktywnia interfejs wyłącznie podczas operacji zapisu i odczytu. Jest to uzasadnione z uwagi na fakt wykorzystywania przez wyświetlacz oraz zewnętrzną pamięć RAM tych samych linii sygnałowych i danych. Mikroprocesor wykonując rozkaz zapisu do pamięci zewnętrznej (MOVX) wykorzystuje linię WR wysyłając jednocześnie dane przez port P0 (rys. 12). Podczas odczytu aktywna jest linia RD. W czasie trwania tych operacji, które dotyczą wyświetlacza dostęp do pamięci zewnętrznej blokowany jest sygnałem CHIP. Możliwa jest więc bezkonfliktowa współpraca obu bloków rejestratora.

Przebiegi czasowe przedstawionych powyżej sygnałów sterujących wyświetlaczem LCD znajdują się na załączonych do niniejszej pracy rysunkach. Rysunek 8 ilustruje w jaki sposób mikroprocesor blokuje dostęp do pamięci zewnętrznej RAM sygnałem CHIP, ustawia linie sterujące WRS i WRW oraz uaktywnia konwerter logiczny linią WCS. Podczas samej operacji odczytu pojawienie się danych na porcie P0 (linie D0 ÷ D7) potwierdzane jest sygnałem E, który powstaje przez zanegowanie aktywnej zerem linii RD mikroprocesora. Podobnie odbywa się zapis danych do wyświetlacza (rys. 9). Różnica polega na ustawieniu niskiego stanu logicznego linii WRW przed uaktywnieniem interfejsu.

Obsługa wyświetlacza wymaga również przesyłania instrukcji sterujących, które umożliwiają między innymi ustawienie trybu wyświetlania i sterowanie kursorem. Instrukcje dla modułu wyświetlacza przesyłane mogą być po ustawieniu niskiego stanu logicznego linii WRS instrukcją CLR WRS.

2.5 Układ klawiatury sterującej

W tym bloku (rys.10), który zrealizowany został jako oddzielny moduł dołączany do urządzenia za pomocą taśmy wielożyłowej, znajduje się zestaw klawiszy sterujących oraz elementy elektroniczne zapewniające współpracę mikroprocesora z przetwornikiem piezoelektrycznym umożliwiającym generację dźwięku. W układzie klawiatury zastosowano diody półprzewodnikowe zabezpieczające wyjścia portu P2 mikroprocesora przed zwarciem. Klawiaturą steruje procedura programowa TEST_KEY, której fragment został przedstawiony poniżej.

```
TEST_KEY:   SETB KEYIN
            LCALL SETKEYS

KEYL1:      CLR KEY1
            JNB KEYIN,PRESS1
            LCALL SETKEYS
KEYL2:      CLR KEY2
            JNB KEYIN,PRESS2
            LCALL SETKEYS
KEYL3:      CLR KEY3
            JNB KEYIN,PRESS3
            LCALL SETKEYS
KEYL4:      CLR KEY4
            JNB KEYIN,PRESS4
            LCALL SETKEYS
KEYL5:      CLR KEY5
            JNB KEYIN,PRESS5
            LCALL SETKEYS

            RET
```

Oznaczona jako KEYIN linia sygnałowa P1.0 (rys. 12) pełni funkcję wejścia i w związku z tym nadawany jej jest wysoki stan logiczny. Przed rozpoczęciem komunikacji z klawiaturą procesor wykonuje podprogram SETKEYS, którego

zadaniem jest ustawienie początkowego (wysokiego) stanu linii klawiszy KEY1 ÷ KEY5 (P2.2 ÷ P2.6). Testowanie klawiatury polega na ustawieniu niskiego stanu logicznego na linii odpowiadającej danemu klawiszowi i sprawdzeniu wejścia KEYIN. Rozkaz skoku warunkowego JNB zostanie wykonany tylko wtedy, gdy na wejściu pojawi się logiczne zero. W zależności od tego, który z klawiszy został wciśnięty wykonana zostanie jedna z procedur PRESS1 ÷ PRESS5. Procedury te zwracają odpowiedni kod klawisza zapisywany w rejestrze KEY do podprogramu, który wywołał procedurę TEST_KEY. Dotyczy to również przypadku, kiedy żaden z klawiszy nie został wciśnięty.

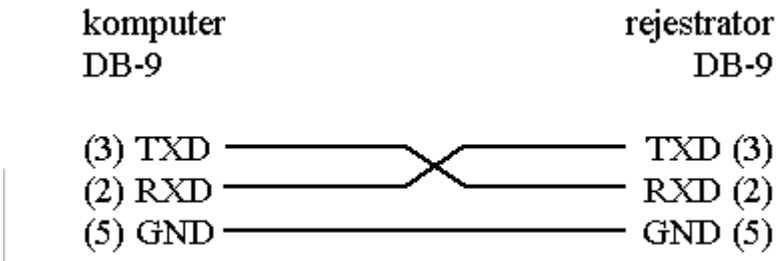
Moduł klawiatury wykorzystuje do współpracy z mikroprocesorem linie sygnałowe służące również jako linie adresowe zewnętrznej pamięci cyfrowej RAM. Dlatego nie jest możliwe testowanie stanu klawiatury podczas wykonywania operacji zapisu lub odczytu z pamięci. Takie rozwiązanie zapewniło pełną funkcjonalność urządzenia i umożliwiło jednocześnie ograniczenie liczby końcówek mikroprocesora koniecznych do komunikacji z modułem klawiatury.

Dodatkowo na płytce modułu umieszczone zostały elementy elektroniczne R1 i R2 umożliwiające podłączenie generatora piezoelektrycznego służącego do sygnalizacji dźwiękowej.

2.6 Interfejs komunikacji szeregowej z komputerem

Moduł interfejsu szeregowego (rys. 11) służy do komunikacji rejestratora holterowskiego z komputerem. Komunikacja odbywa się przez złącze standardu RS- 232 C znajdujące się w każdym komputerze klasy PC. Interfejs łączący urządzenia zapewnia dwukierunkową transmisją danych przez kabel typu NULL-MODEM z szybkością w zakresie od 9600 do 19200

bitów/s. Sposób połączenia linii transmisyjnych kabla, który zakończony jest z dwóch stron wtyczkami typu DB-9 został przedstawiony poniżej.



Interfejs portu szeregowego umożliwia zarówno transmisję danych pomiarowych, jak też bajtów sterujących pracą urządzenia. Funkcje bajtów sterujących zostały opisane w przedstawionej tabeli.

Transmitowany bajt (ASCII)	opis funkcji
"t"	przełączenie rejestratora w tryb rejestracji i wyświetlania czasu
"m"	przełączenie rejestratora w tryb ustawiania parametrów (menu)
"p"	przełączenie rejestratora w tryb programowania (do urządzenia przesyłany jest z programu com232.exe rekord sterujący TRecordEKG)
"e"	przełączenie rejestratora w tryb monitora ekg

Bajty sterujące umożliwiają interaktywną współpracę rejestratora z programem com232.exe, w tym wygodne ustawianie parametrów pracy urządzenia. Dokładny opis procedur programowych umożliwiających współpracę rejestratora z komputerem przez port szeregowy znajduje się w rozdziale 3.1 "Opis programu sterownika AT89C52".

W urządzeniu zastosowano układ scalony MAX232 będący standardowym podwójnym nadajnikiem i odbiornikiem interfejsu RS-232 C zasilanym pojedynczym napięciem +5V. Zawiera on w swojej strukturze dwa konwertery dostarczające napięcia +10V i -10V. Wejścia układu są kompatybilne z poziomami logicznymi standardu TTL/CMOS, co umożliwia współpracę z portem szeregowym mikroprocesora AT89C52. Układ charakteryzuje się również małym poborem mocy oraz wymaga do pracy niewielkiej ilości elementów zewnętrznych.

W celu zmniejszenia poziomu zakłóceń generowanych przez układy cyfrowe zastosowano prosty układ filtrujący składający się z dławika L1 i kondensatora C7. Dzięki takiemu rozwiązaniu możliwe stało się uzyskanie wysokich szybkości transmisji przy wykorzystaniu długiego kabla przyłączeniowego.

2.7 Układ sterownika mikroprocesorowego

Blok mikroprocesora AT89C52 z pamięcią cyfrową RAM (rys. 12) jest najważniejszą częścią urządzenia. Wykorzystano w nim układ AT89C52, który jest mikrokomputerem jednoukładowym rodziny MCS-51 produkowanym przez firmę Atmel. Posiada on w swojej strukturze reprogramowalną w systemie pamięć PEROM typu Flash o pojemności 8 kB. Rdzeń procesora jest kompatybilny ze standardem przemysłowym 80C52 i zawiera wewnętrzną pamięć danych RAM o rozmiarze 256 bajtów, trzy 16-bitowe liczniki, dwupoziomowy system przerwań, port szeregowy umożliwiający niezależne nadawanie i odbieranie transmisji szeregowej oraz cztery 8-bitowe porty wejścia-wyjścia. Ośmiobitowa jednostka centralna może wykonywać 111 rozkazów umożliwiających łatwą i efektywną realizację wszelkiego rodzaju algorytmów sterowania . Lista rozkazów mikroprocesora

zawiera między innymi rozkazy arytmetyczne i logiczne, rozkazy dotyczące operacji logicznych na bitach (procesor boolowski) oraz rozbudowane grupy rozkazów skoków warunkowych i wejścia-wyjścia. Prawie wszystkie rozkazy wykonują się w czasie jednego lub dwóch cykli maszynowych.

Zegar taktujący mikroprocesora jest stabilizowany zewnętrznym rezonatorem kwarcowym X1 o częstotliwości 11,059 MHz. Wartość ta ustalona została ze względu na konieczność dopasowania szybkości transmisji portu szeregowego do standardu wykorzystywanego w komputerach klasy PC.

Na rysunku 12 przedstawiono schemat systemu z pamięcią RAM 62256 (32K x 8) jako zewnętrzną pamięcią danych. Komunikacja procesora z tym układem odbywa się przez port P0, który pełni funkcję wejścia-wyjścia dwukierunkowej multipleksowanej magistrali adres/dane. Do portu P0 wysyłanych jest osiem mniej znaczących bitów adresu strobowanych sygnałem ALE. Ponieważ konieczne jest zapamiętanie wysyłanego z portu P0 adresu zastosowano dodatkowy rejestr adresowy U3. Wykorzystano w tym celu cyfrowy układ 74HCT573 posiadający w swojej strukturze osiem zatrzasków (przerzutników). Do końcówki 11 tego układu został doprowadzony sygnał ALE, który wysyłany jest dwukrotnie w każdym cyklu maszynowym. Jego opadające zbocze oznacza, że adres jest obecny na porcie P0 i dotyczy najbliższego przesłania. Natomiast osiem bardziej znaczących bitów adresu wysyłanych jest do portu P2. Na schemacie wykorzystywane linie adresowe oznaczone są jako A00 - A13. Przesyłanie danych sterowane jest za pomocą sygnałów RD (P3.7) i WR (P3.6). Są to sygnały odczytywania i zapisywania do zewnętrznej pamięci danych generowane w drugim cyklu maszynowym, wytwarzane w czasie wykonywania rozkazów MOVX. Gdy mikroprocesor wykonuje rozkaz MOVX A, @DPTR ustawiany jest aktywny (niski) stan logiczny linii RD doprowadzonej do końcówki 22 układu U4 i bajt danych z zewnętrznej pamięci o adresie podanym w DPTR przesyłany jest do

akumulatora. W przypadku zapisu cykl ten przebiega podobnie, lecz wykorzystywany jest analogicznie sygnał WR doprowadzony do końcówki 27 układu pamięci (U4). Odczyt i zapis odbywa się przez port P0 do którego dołączona została magistrala danych (D0 - D7). Ponieważ linie tego portu wykorzystywane są jako wyjścia konieczne było dołączenie zewnętrznych oporników polaryzujących RP1, które wymuszają stan wysoki na końcówkach linii, gdy na wyjście wysyłana jest jedynka. Układ mikrokontrolera nie zapewnia dla portu P0 właściwej polaryzacji, gdyż nie zastosowano w jego strukturze wewnętrznych oporników.

Układ pamięci zewnętrznej RAM dzieli magistralę danych i część magistrali adresowej z interfejsami wyświetlacza alfanumerycznego LCD i klawiatury. Dlatego konieczne było zastosowanie sygnału blokującego dostęp do pamięci podczas operacji transmisji danych do tych interfejsów. Za poprawną współpracę interfejsów odpowiada program mikroprocesora, który uaktywnia układ pamięci sygnałem CHIP (P2.7) tylko w czasie operacji odczytu lub zapisu danych. Również interfejsy wyświetlacza LCD i przetwornika A/C są uaktywniane sygnałami WCS i CS.

Końcówki zasilające wszystkich układów cyfrowych w tym bloku zostały przyłączone do masy przez kondensatory blokujące w celu stłumienia zakłóceń impulsowych wysokiej częstotliwości. Blokowanie zasilania w połączeniu z odpowiednim prowadzeniem ścieżek zasilających na płytce drukowanej umożliwiło poprawną pracę wszystkich układów cyfrowych i analogowych. Przyjęte rozwiązania konstrukcyjne zostały szczegółowo przedstawione w rozdziale 2.8 “Blok zasilania”.

2.8 Blok zasilania

W układzie rejestratora holterowskiego z pamięcią cyfrową występuje konieczność używania kilku napięć zasilających. Z tego powodu w urządzeniu tym przewidziane są trzy zasilacze: pierwszy dla części cyfrowej urządzenia o napięciu +5V, drugi dla części analogowej o napięciu +5V i trzeci dla części analogowej o napięciach ±9V (rys. 13). Do budowy dwóch pierwszych zasilaczy zostały zastosowane układy scalonych stabilizatorów typu LM7805 (U12 i U13). Kondensatory C20 ÷ C23 o wartości 100 nF oraz C36 ÷ C39 o wartości 100 μF znajdują się bezpośrednio przy końcówkach tych układów i pełnią funkcję filtrów przeciwzakłóceniowych. Zasilacz U12 dostarcza standardowego napięcia +5V do wszystkich układów cyfrowych rejestratora, tzn. mikroprocesora, układu pamięci RAM, układów logicznych, części cyfrowej przetwornika A/C, wyświetlacza alfanumerycznego LCD i układu nadajnika/odbiornika interfejsu szeregowego. Wszystkie wymienione układy generują zakłócenia impulsowe na liniach zasilających, które mogą uniemożliwić poprawną pracę urządzenia. W celu wyeliminowania tych zakłóceń wszystkie końcówki zasilające układów cyfrowych zostały przyłączone bezpośrednio do masy przez kondensatory blokujące 100 nF. Zasilacz U13 dostarcza napięcia +5V do części analogowej przetwornika A/C. Zastosowanie oddzielnego źródła zasilającego podyktowane zostało koniecznością zapewnienia wysokiej dokładności przetwarzania sygnałów analogowych, którą można osiągnąć wyłącznie przez rozdzielenie przepływu prądów powrotnych (prądów mas) sygnałów cyfrowych i analogowych. Takie rozwiązanie konstrukcyjne zapobiegło redukcji rzeczywistej rozdzielczości przetwornika.

Stopień wejściowy rejestratora holterowskiego oraz układ filtrów wymagał zastosowania wyższych napięć zasilających. Podyktowane było to

koniecznością zapewnienia szerokiego zakresu przetwarzania składowej stałej sygnału pomiarowego. Dzięki takiemu rozwiązaniu uniknięto konieczności regulacji napięcia odniesienia i zdecydowanie zredukowano ilość układów scalonych potrzebnych do konstrukcji urządzenia. Napięcie +9V wytwarzane jest przez zasilacz stabilizowany podłączany do gniazda GZ1 lub pobierane z baterii 6F22 znajdującej się wewnątrz rejestratora. W celu zablokowania niepożądanych zakłóceń zastosowano kondensator C41 o wartości 100 μF. Napięcie zasilające -9V otrzymane zostało przez inwersję napięcia +9V, która odbywa się w przedstawionym na schemacie układzie U6. Układ ten oznaczony jako LCM 7660IN jest prostym konwerterem napięciowym z przełączanymi pojemnościami, który do pracy wymaga tylko dwóch elementów zewnętrznych C11 i C12. Dodatkowo w celu eliminacji zakłóceń, które w sposób znaczny mogły by zniekształcić sygnał w torze pomiarowym zastosowano filtrację napięcia podawanego na końcówkę 8 układu przy pomocy kondensatorów C28 i C42. Napięcie -9V uzyskiwane po inwersji doprowadzane jest do układów wzmacniaczy stosunkowo długą ścieżką zasilającą i wymagało dodatkowego wygładzenia, które uzyskano stosując dławik L2.

Połączenie mas wszystkich napięć zasilających oraz sposób umieszczenia kondensatorów blokujących przedstawione jest na rysunku 14. Można zauważyć, że wszystkie ścieżki napięć wejściowych poszczególnych zasilaczy prowadzone są bezpośrednio od kondensatora C43 o wartości 470 μF. Takie rozwiązanie konstrukcyjne zapewnia rozdzielenie cyfrowych i analogowych prądów mas oraz eliminuje przenoszenie zakłóceń między blokami rejestratora. Dzięki temu stało się możliwe przetwarzanie sygnałów pomiarowych o małym poziomie napięciowym w układzie wykorzystującym dużą ilość układów cyfrowych.

3. Oprogramowanie urządzenia

Oprogramowanie rejestratora holterowskiego z pamięcią cyfrową składa się z dwóch części będących w rzeczywistości niezależnymi programami. W rozdziale tym opisane zostaną funkcje i sposób działania programu sterownika mikroprocesorowego AT89C52 oraz programu com232.exe pracującego w systemie operacyjnym Windows. Kody źródłowe tych programów znajdują się w rozdziale 8 “Dodatki” a także zapisane zostały w postaci plików projektów na dołączonej do niniejszej pracy dyskietce.

3.1 Opis programu sterownika AT89C52

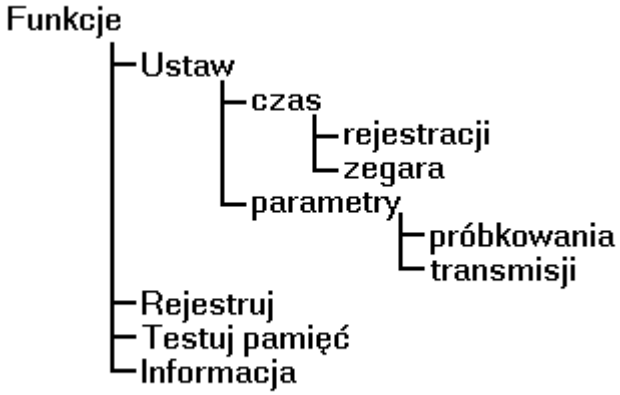
Program obsługujący pracę rejestratora holterowskiego z pamięcią cyfrową napisany został w asemblerze mikrokomputerów rodziny MCS-51 i po kompilacji do postaci binarnej zapisany do pamięci reprogramowalnej PEROM typu Flash sterownika AT89C52. Otrzymany plik wynikowy o rozmiarze 6,5 KB znajduje się na dyskietce załączonej do niniejszej pracy (A:\At89c52\Rejestr.bin). Kod źródłowy programu umieszczony został w rozdziale 8 “Dodatki”.

Przy tworzeniu oprogramowania mikroprocesora przyjęte zostały następujące założenia:

1. łatwy w obsłudze interfejs użytkownika umożliwiający programowanie i sterowanie rejestratora bezpośrednio z klawiatury lub zdalnie przez łącze szeregowe;
2. dane pomiarowe zapisywane są w pamięci statycznej RAM lub bezpośrednio przesyłane do komputera przez łącze szeregowe;
3. próbki wejściowe są kodowane i w takiej postaci przechowywane oraz transmitowane.

4. format nagłówka bloku danych zawiera wszystkie niezbędne dane umożliwiające prostą konwersję w celu wyświetlania i zapisu do pliku.

Interfejs użytkownika umożliwia bezpośrednie programowanie rejestratora i sterowanie jego pracą. Posiada interaktywne menu pozwalające w prosty sposób ustawiać wszystkie parametry urządzenia. Przedstawia je poniższy schemat.



Obsługę rejestratora holterowskiego należy rozpocząć od ustawienia zegara czasu rzeczywistego. W tym celu używa się klawiszy kursora klawiatury sterującej urządzenia i przechodzi do menu “Ustaw › czas › zegara ›” , by ustawić aktualny czas. Prawidłowe ustawienie potwierdza się klawiszem “ENTER” znajdującym się po prawej stronie klawiatury. Następnie należy ustawić wybraną częstotliwość próbkowania przetwornika A/C. Aby to osiągnąć,0 przechodzi się do pozycji “Ustaw › parametry › próbkowania ›” i wybraną wartość potwierdza również klawiszem “ENTER”. Rejestrator holterowski umożliwia dokonywanie wielu następujących po sobie rejestracji rozpoczynających się o dowolnej godzinie. Łączny czas trwania wszystkich rejestracji ograniczony jest wielkością pamięci RAM. Aby rozpocząć programowanie rejestracji należy przejść do pozycji “Ustaw › czas › rejestracji ›” i wybrać funkcję [dodaj], a następnie ustawić czas początku pierwszej

rejestracji. Po zatwierdzeniu prawidłowej wartości należy wybrać również czas trwania rejestracji. Urządzenie automatycznie kontroluje ilość dostępnej pamięci i uniemożliwia przekroczenie maksymalnego czasu trwania zapisu. Po potwierdzeniu wybranych wartości uaktywnione zostaje przewijane menu, które pozwala na przegląd dotychczasowych ustawień i ich zmianę. Można dodać kolejną rejestrację wybierając funkcję [dodaj] znajdującą się na końcu lub usunąć ostatnią klawiszem “LEFT”. Po wybraniu dowolnych ustawień należy wybrać funkcję [zakończ]. Rozpoczęcie wykonywania zaprogramowanych rejestracji możliwe jest po przejściu w menu do pozycji “Rejestruj ›”. Na wyświetlaczu urządzenia pojawi się zegar odmierzający aktualny czas. W każdej chwili można przerwać wykonywanie programu klawiszem “ENTER”. Kiedy czas zegara zrówna się z czasem początku rejestracji zaprogramowanym wcześniej urządzenie rozpocznie pobieranie danych pomiarowych, co sygnalizowane jest zapaleniem czerwonej diody LED. Po zakończeniu pomiaru dioda gaśnie a urządzenie przechodzi w tryb uśpiony, który trwa do czasu kolejnej rejestracji. Ponieważ rejestracje dokonywane są w tle głównego programu, podczas ich trwania odmierzany jest prawidłowo czas zegara.

Interfejs użytkownika umożliwia również dokonywanie ustawień parametrów transmisji przez port szeregowy. W tym celu należy wybrać w menu “Ustaw › parametry › transmisji ›” jedną z dostępnych szybkości transmisji. Maksymalną dostępną wartością jest 19200 bitów/s, jednak możliwe jest ustawienie innych wartości w celu dopasowania szybkości przesyłania danych do charakterystyki medium transmisyjnego, którym może być np. modem radiowy lub telefoniczny.

Ostatnią funkcją interfejsu jest możliwość testowania zainstalowanej w urządzeniu pamięci RAM. Po wybraniu w menu pozycji “Testuj pamięć ›”

program główny rejestratora dokonuje sprawdzenia poprawności zapisu i odczytu wszystkich bitów układu pamięci, po czym restartuje urządzenie.

Wszystkich ustawień dokonać można również z poziomu interfejsu programu com232.exe, który umożliwia wyświetlanie danych pomiarowych w postaci graficznej i sterowanie zdalne rejestratorem. Aplikacja pod Windows pozwala na przełączenie urządzenia w tryb pracy ciągłej, podczas którego pełni ono funkcję monitora ekg. W tym przypadku dane nie są zapisywane do pamięci RAM a przesyłane bezpośrednio łączem szeregowym do komputera, gdzie po poddaniu ich odpowiedniej konwersji przedstawione zostają na ekranie w postaci krzywej ekg. Kody bajtów sterujących rejestratorem oraz opisy wywoływanych nimi funkcji zamieszczone zostały w na stronie 22 w rozdziale 2.6 “Interfejs komunikacji szeregowej z komputerem”. Podczas programowania do rejestratora przesyłany jest rekord sterujący w postaci następującej struktury języka C:

```
struct TRecordEKG {
        char    HEADER1[12];        // "RECEKG v.1.0"
        char    Imie[20];
        char    Nazwisko[20];
        char    Numer[12];
        BYTE    IloscRekordow;
        BYTE    Fprobkowania;
        BYTE    TabCzasow[450];
        char    HEADER2[4];
        };
```

Poszczególne pola tej struktury zawierają dane umożliwiające pracę urządzenia. Pole “IloscRekordow” określa liczbę rejestracji do wykonania. W zmiennej “Fprobkowania” zawarta jest liczba określająca wspólną dla wszystkich pomiarów częstotliwość próbkowania. Natomiast dane znajdujące

się w tablicy “TabCzasow” decydują o czasie rozpoczęcia poszczególnych rejestracji i długości ich trwania oraz określają parametry bloku danych w pamięci RAM, takie jak adres początku i długość. Zawartość tej tablicy składa się z sekwencji 9-bajtowych bloków o strukturze przedstawionej poniżej.

Nr	***Nazwa***	***Opis wartości zapisanej w bajcie***
0	TRH1	początek rejestracji (godziny)
1	TRM1	początek rejestracji (minuty)
2	TRS1	początek rejestracji (sekundy)
3	TTM1	czas trwania rejestracji (minuty)
4	TTS1	czas trwania rejestracji (sekundy)
5	DPH	starszy bajt adresu początku zapisu w pamięci (DPTR)
6	DPL	młodszy bajt adresu początku zapisu w pamięci (DPTR)
7	LBH	liczba bajtów zapisu (H)
8	LBL	liczba bajtów zapisu (L)

Przyjęta wielkość tablicy (450 bajtów) ogranicza ilość możliwych do zaprogramowania rejestracji do pięćdziesięciu.
Struktura “TRecordEKG” przesyłana jest również podczas odczytu danych pomiarowych, a także zapisywana z blokiem danych do pliku na dysku. Takie rozwiązanie umożliwiło proste zapamiętanie wszystkich parametrów rejestracji.

Program sterownika mikroprocesorowego podczas startu inicjuje trzy procedury przerwań. Pierwsza o adresie 000BH obsługuje zgłoszenia pochodzące od licznika T0 i dokonuje rejestracji danych pomiarowych z wybraną częstotliwością 80, 100 lub 120 Hz. Integralną częścią tej procedury

jest podprogram rejestracji, który pobiera 12-bitową próbkę z przetwornika A/C i dokonuje jej konwersji różnicowej względem poprzedniej próbki, w wyniku której otrzymujemy liczbę 8-bitową zapisaną w kodzie U2 (ze znakiem). Jej wartość zawiera się w przedziale od –128 do 127 (80H do 7FH). Podczas konwersji dokonywane są obliczenia według wzoru:

$$P_{wy}(n) = P_{we}(n-1) - P_{we}(n)$$

gdzie: P_{wy} - próbka po konwersji (8-bitów w kodzie U2)

P_{we} - próbka z przetwornika A/C (12-bitowa bez znaku)

Przed rozpoczęciem pomiarów procedura pobiera próbkę odniesienia. Zaletą takiego rozwiązania jest wyeliminowanie zapamiętywania zbędnej informacji o składowej stałej przebiegu ekg, a przez to zmniejszenie ilości danych pomiarowych koniecznych do zapisania w pamięci RAM rejestratora i wydłużenie czasu rejestracji. Podprogram ten, w zależności od wybranej opcji, może również wysyłać tak przetworzone dane bezpośrednio kanałem RS-232 do aplikacji graficznej, która wyświetla je w czasie rzeczywistym na ekranie monitora komputera PC.

Procedura przerwań o adresie 0023H obsługuje zgłoszenia przychodzące z portu szeregowego. Jest ona odpowiedzialna za funkcje sterowania zdalnego rejestratorem oraz transmisje znaków kanałem interfejsu RS-232.

Trzecia procedura rozpoczyna się w pamięci programu w komórce o adresie 002BH i przyjmuje zgłoszenia przerwań z licznika T2. Odpowiada ona za funkcję zegara czasu rzeczywistego rejestratora holterowskiego oraz dokonuje sprawdzenia zgodności czasu rejestracji zaprogramowanego wcześniej z czasem aktualnym. W chwili stwierdzenia takiej zgodności uruchamiany jest podprogram rejestracji danych.

Wymienionym procedurom przydzielony został właściwy priorytet. Umieszczenie przerwania na danym poziomie priorytetu decyduje o możliwości przerywania programów obsługi innych przerwań. W czasie wykonywania programu obsługi przerwania z niższego poziomu priorytetu będzie przyjęte zgłoszenie przerwania z wyższego poziomu, a nie będzie przyjęte zgłoszenie przerwania z niższego poziomu. Natomiast, w czasie wykonywania programu obsługi przerwania z wyższego poziomu priorytetu nie będzie przyjęte żadne zgłoszenie przerwania. Program obsługi przerwania z wyższego poziomu jest nieprzerywalny, co zapewnia poprawną pracę urządzenia. Procedura zegara czasu rzeczywistego posiadająca najwyższy priorytet nie jest przerywana przez pozostałe programy i odmierza prawidłowo czas niezależnie od innych funkcji wykonywanych w tym samym czasie przez rejestrator. Ustawieniem właściwych priorytetów zajmuje się procedura inicjacji systemu.

Pozostałe podprogramy sterownika mikroprocesorowego wywoływane z pętli głównej programu odpowiedzialne są za obsługę klawiatury i wyświetlacza LCD. Ich działanie zostało opisane wcześniej w rozdziałach 2.4 i 2.5.

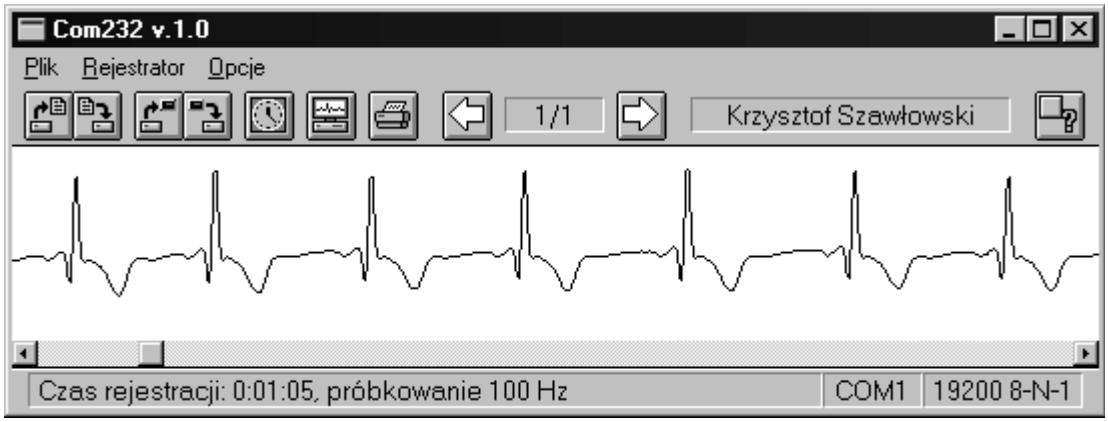
3.2 Opis programu com232.exe

Program com232.exe jest aplikacją systemu operacyjnego Windows i umożliwia wizualizację graficzną danych pomiarowych zebranych przez rejestrator holterowski z pamięcią cyfrową. Do napisania programu wykorzystano kompilator języka Borland C++ w wersji 4.52. Utworzony projekt wraz z plikami źródłowymi i wymaganymi bibliotekami umieszczony został na dyskietce załączonej do pracy.

Przy tworzeniu oprogramowania sterującego rejestratorem przyjęte zostały następujące założenia:

1. łatwy w obsłudze interfejs użytkownika umożliwiający zarówno sterowanie rejestratorem holterowskim przez kanał interfejsu szeregowego RS-232, jak też zdalne programowanie urządzenia i odczytywanie zgromadzonych wcześniej danych pomiarowych;
2. dane z rejestratora przechowywane w pamięci operacyjnej komputera mogą zostać zapisane na dysku twardym w pliku oraz ponownie wczytane w celu późniejszego przeglądania;
3. format binarny pliku zawiera wszystkie informacje umożliwiające właściwe odtworzenie graficzne danych pomiarowych;
4. wykres krzywej ekg tworzony na ekranie monitora może zostać wydrukowany na dowolnej drukarce.

Interfejs graficzny aplikacji został przedstawiony poniżej.



Zawiera on standardowe elementy umożliwiające uruchamianie poszczególnych funkcji programu, które dostępne są zarówno z rozwijanych menu, jak też za pośrednictwem ikon na pasku sterującym. Ikony te posiadają następujące znaczenie:

 odczytuje z dysku plik rejestratora zawierający dane pomiarowe;

 zapisuje dane pomiarowe na dysku w formacie binarnym;

 programuje rejestrator – zapisuje parametry pracy przez port szeregowy;

 odczytuje dane pomiarowe z rejestratora przez port szeregowy;

 wywołuje okno dialogowe umożliwiające ustawienie parametrów pracy urządzenia oraz czasów rejestracji;

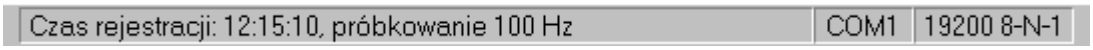
 uruchamia funkcję monitora ekg – umożliwia oglądanie krzywej ekg w czasie rzeczywistym na ekranie monitora;

 drukuje krzywą ekg na dowolnej drukarce;

 pokazuje poprzedni zapisany rekord danych;

 pokazuje następny zapisany rekord danych;

U dołu okna aplikacji znajduje się pasek statusu, który informuje o niektórych parametrach pracy rejestratora i czasie w jakim zapisana została oglądana krzywa ekg.

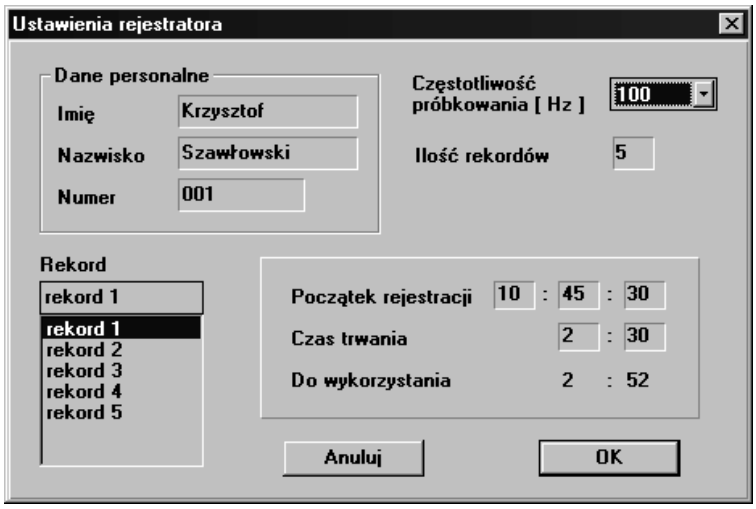


Powyższa ilustracja umożliwia stwierdzenie, że oglądany na monitorze przebieg został zarejestrowany o godzinie 12:15 z częstotliwością próbkowania wynoszącą 100 Hz. Po prawej stronie dodatkowo znajduje się informacja o aktualnie wykorzystywanym porcie szeregowym komputera oraz jego parametrach transmisji: szybkość 19200 bit/s, 8 bitów danych bez kontroli parzystości (N), jeden bit stopu.

W celu umożliwienia poprawnej komunikacji rejestratora z komputerem PC należy ustawić w programie com232.exe takie same wartości parametrów transmisji jak w rejestratorze. W menu „Opcje” należy wybrać funkcję „Port szeregowy”, która udostępnia następujące okno dialogowe.

Zaprogramowanie rejestratora może zostać przeprowadzone na dwa sposoby. Pierwszy z nich polega na zmianie parametrów transmisji bezpośrednio w urządzeniu i został przedstawiony w rozdziale 3.1. Drugi polega na uaktywnieniu ikony uruchamiającej funkcję programowania. Wszystkie ustawione parametry i informacja o czasach rejestracji zostaną przesłane łączem szeregowym do urządzenia. Poprawne przeprowadzenie transmisji sygnalizowane jest przez rejestrator krótkim dźwiękiem.

W celu zmiany parametrów rejestracji i czasów rejestracji należy uaktywnić ikonę udostępniającą okno dialogowe przedstawione poniżej.



Po prawej stronie u góry znajduje się rozwijana lista umożliwiająca wybranie częstotliwość próbkowania przetwornika A/C. Dostępne są trzy wartści: 80, 100 i 120 Hz. W przypadku wybrania częstotliwości wyższej zwiększa się jakość odwzorowania krzywej ekg, lecz zmniejsza dostępna ilość czasu rejestracji.

W trybie rejestratora holterowskiego urządzenie posiada zdolność dokonywania wielokrotnych zapisów o ustalonym czasie rozpoczęcia i czasie trwania. W celu zaprogramowania rejestracji holterowskich w polu „Ilość rekordów” należy wpisać właściwą liczbę, która decyduje o ilości pozycji na liście po lewej stronie. Zaznaczając kolejno każdą z tych pozycji należy następnie ustawić właściwy czas początku i czas trwania danego rekordu. Program informuje użytkownika o ilości dostępnego czasu rejestracji w polu „Do wykorzystania”. Prawidłowe ustawienia należy potwierdzić przyciskiem „OK” a następnie przesłać je do urządzenia w sposób opisany powyżej, przez uaktywnienie ikony programowania  . Jeżeli w rjestratorze został właściwie ustawiony zegar czasu rzeczywistego, to po wybraniu funkcji “Rejestruj” urządzenie rozpocznie pracę w zaprogramowanych okresach.

Po zakończeniu wszystkich zaprogramowanych rejestracji należy przesłać dane łączem szeregowym do programu. W tym celu uaktywniamy ikonę odczytu danych  . W odpowiednim okienku dialogowym aplikacja informuje użytkownika o ilości przesłanych przez kanał RS-232 bajtów informacji. Po zakończeniu operacji pobierania danych można przystąpić do przeglądania wyników pomiarów w oknie głównym programu. Poszczególne rejestracje dostępne są przez ikony zmiany rekordu  i  .

Program zawiera funkcje pozwalające na modyfikowanie wykresu danych pomiarowych w celu uzyskania lepszej wizualizacji. Jedną z nich jest możliwość zastąpienia wykresu punktowego wykresem liniowym. Aby ją uaktywnić należy zaznaczyć w menu „Opcje” pozycję „Wykres liniowy”. Drugą funkcją jest kompensacja zmian składowej stałej sygnału próbkowanego przez przetwornik A/C. Podczas dokonywania pomiarów może zdarzyć się, że poruszenie elektrod spowoduje zmianę położenia wykresu krzywej ekg na ekranie monitora. W celu utrzymania jednakowego poziomu

oglądanego przebiegu należy zaznaczyć w menu „Opcje” pozycję „Kompensacja składowej stałej”.

Program umożliwia dodatkowo zmianę kolorów tła i wykresy, co osiągamy przez wybranie w tym samym menu funkcji „Kolor tła” i „Kolor wykresu”. Możliwe jest również wydrukowanie oglądanego wykresu na dowolnej drukarce przez uaktywnienie ikony  , lub wybranie funkcji „Drukuj” z menu „Plik”. Tam również użytkownik może wywołać funkcję „Ustawienia strony”, umożliwiającą ustawienie parametrów drukarki i formatu wydruku.

Ostatnią funkcją przedstawianego programu jest wyświetlanie krzywej ekg na ekranie monitora w czasie rzeczywistym. W tym trybie, uruchamianym za pomocą ikony  , urządzenie zamienia się w pełnosprawny monitor ekg. Dane pobierane z przetwornika A/C kodowane są w sposób opisany wcześniej w rozdziale 3.1 „Opis programu sterownika AT89C52”, jednak nie są zapisywane do pamięci RAM rejestratora a przesyłane bezpośrednio łączem szeregowym do komputera PC. Tam po odpowiedniej konwersji na współrzędne ekranu zostają wyświetlone. Wykres krzywej ekg rysowany jest od lewej strony do prawej, przy czym nowe dane graficzne nadpisują poprzednio wyświetlone. W celu lepszego rozróżnienia tych dwóch obszarów wprowadzono opcję rysowania pionowej przesuwającej się wraz z wykresem linii. Jej uaktywnienie jest możliwe przez zaznaczenie pozycji „Pionowa linia” w menu „Opcje”.

Dokładna procedura przeprowadzania pomiarów za pomocą rejestratora holterowskiego zostanie przedstawiona w rozdziale następnym.

3.3 Instrukcja obsługi urządzenia

Rejestrator holterowski z pamięcią cyfrową posiada możliwość pracy w dwóch trybach: monitora EKG i rejestratora holterowskiego. Pierwszy z nich umożliwia obserwowanie w czasie rzeczywistym przebiegu krzywej ekg na ekranie monitora komputerowego, drugi pozwala na dokonywanie rejestracji w określonym wcześniej czasie.

W celu wykorzystania urządzenia jako monitora EKG należy wykonać następujące czynności:

1. Podłączyć wchodzący w skład urządzenia zasilacz do gniazdka zasilania rejestratora lub umieścić w urządzeniu baterię 9V;
2. Podłączyć przewód transmisyjny RS-232 do gniazdka DB-9 komputera PC;
3. Drugi koniec przewodu podłączyć do gniazdka DB-9 rejestratora;
4. Podłączyć końcówki elektrod wraz z elektrodami;
5. Włączyć zasilanie rejestratora przełącznikiem znajdującym się nad gniazdkiem zasilania;
6. W menu rejestratora wybrać pozycję „Ustaw > parametry > transmisji >” i zaznaczyć właściwą szykość portu (domyślne ustawienie: 19200 bit/s) i potwierdzić klawiszem „ENTER”;
7. Wybrać pozycję „Ustaw > parametry > próbkowania >” i ustawić częstotliwość próbkowania przetwornika A/C (domyślne ustawienie: 100 Hz);
8. Uruchomić aplikację przez wybranie z menu „Start” systemu operacyjnego Windows skrótu „Programy\Rejestrator holterowski\ com232.exe”;
9. Z menu „Opcje” wybrać pozycję „Port szeregowy” i ustawić wykorzystywany port szeregowy: COM1 lub COM2;
10. Uaktywnić ikonę  na belce sterującej lub wybrać z menu „Rejestrtor” funkcję „Monitor EKG”.

W wyniku powyższych czynności na wyświetlaczu LCD urządzenia pojawi się napis „Monitor EKG” a w oknie głównym programu zacznie rysować się krzywa ekg. Aby zmienić częstotliwość próbkowania przetwornika z poziomu aplikacji Windows należy uaktywnić ikonę  , ustawić w oknie dialogowym ilość rekordów co najmniej na 1 oraz wybrać właściwą jej wartość. Następnie uaktywniając ikonę  zaprogramować rejestrator. Zmiana będzie widoczna po ponownym uaktywnieniu ikony  na belce sterującej.

W celu wykorzystania urządzenia jako rejestratora holterowskiego należy wykonać następujące czynności:

1. Wykonać czynności opisane wyżej w punktach 1 do 9 poprzedniej instrukcji odnoszącej się do monitora EKG;
2. W rejestratorze dodatkowo ustawić właściwy czas zegara przechodząc do menu „Ustaw > czas > zegara >”. Ustawień dokonać należy klawiszami strzałek kursora i potwierdzić przyciskiem „ENTER” znajdującym się po prawej stronie klawiatury urządzenia;
3. Uaktywnić ikonę  a następnie w oknie dialogowym ustawić kolejno: częstotliwość próbkowania, liczbę rekordów oraz czas rozpoczęcia i trwania rejestracji (osobno dla każdego rekordu). Selekcji rekordów dokonać należy wybierając właściwą pozycję z listy „Rekord” po lewej stronie okna. Poprawne ustawienia potwierdza się przyciskiem „OK”;
4. Uaktywnić ikonę  w celu zaprogramowania rejestratora;
5. W rejestratorze wybrać funkcję „Rejestruj >”;

Po przeprowadzeniu wszystkich zaprogramowanych rejestracji, których trwanie sygnalizowane jest zapaleniem się czerwonej diody LED, należy wykonać następujące czynności:

6. W rejestratorze przycisnąć klawisz „ENTER”, w celu przejścia do głównego menu „Funkcje”;
7. W programie com232.exe uaktywnić ikonę  , w celu odczytania danych z urządzenia lub wybrać pozycję „Odczytaj” z menu „Rejestrator”;
8. Po zakończeniu transmisji rezultat poszczególnych pomiarów można przeglądać za pomocą ikon zmiany rekordu  i  .
9. Uaktywnić ikonę  , w celu zapisania zgromadzonych danych pomiarowych do pliku na dysku;
10. Uaktywnić ikonę  w celu wydrukowania krzywej ekg na drukarce.

Rejestrator holterowski można również zaprogramować z poziomu urządzenia, bez wykorzystania aplikacji Windows. Sposób przeprowadzenia procedury programowania za pomocą klawiatury został przedstawiony w rozdziale 3.1 „Opis programu sterownika AT89C52”.

4. Założenia i wzory projektowe

4.1 Wzmacniacz wejściowy

Wzmacniacz wejściowy zastosowany w rejestratorze holterowskim z pamięcią cyfrową jest układem o różnicowym wejściu z zamkniętą pętlą sprzężenia zwrotnego. Przeznaczony jest on do dokładnego wzmacniania małych sygnałów różnicowych występujących w obecności dużego sygnału współbieżnego.

W wybranym rozwiązaniu układowym (rys. 15) wzmacniacz pomiarowy zbudowany jest z jednego wzmacniacza scalonego INA118, w którym znajdują się trzy wzmacniacze operacyjne. Zastosowany układ scalony tworzy stopień wejściowy, który dzięki odpowiednio dobranym elementom charakteryzuje się dobrą symetrią napięcia niezrównoważenia i jego dryftu. Na zewnątrz układu wyprowadzono końcówki umożliwiające podłączenie opornika R_G, którego wartość decyduje o wzmocnieniu napięciowym. Obliczamy je według następującego wzoru:

$$G = 1 + \frac{50k\Omega}{R_G}$$

Ponieważ różnicowe napięcie wejściowe będzie zawierać się w granicach od 0.1 do 5 mV przyjęto teoretyczne wzmocnienie stopnia wejściowego 100V/V. Dla takiego wzmocnienia wartość rezystancji R_G wynosi 0,5 kΩ. W celu umożliwienia swobodnego dobrania stopnia wzmocnienia do poziomu mierzonego sygnału wejściowego zastosowano potencjometr obrotowy.

Scalony wzmacniacz INA118 posiada w swojej strukturze oporniki wysokiej klasy, których wartości są dobierane w procesie produkcji pod kątem zminimalizowania błędu wzmocnienia oraz zapewnienia jak najlepszego

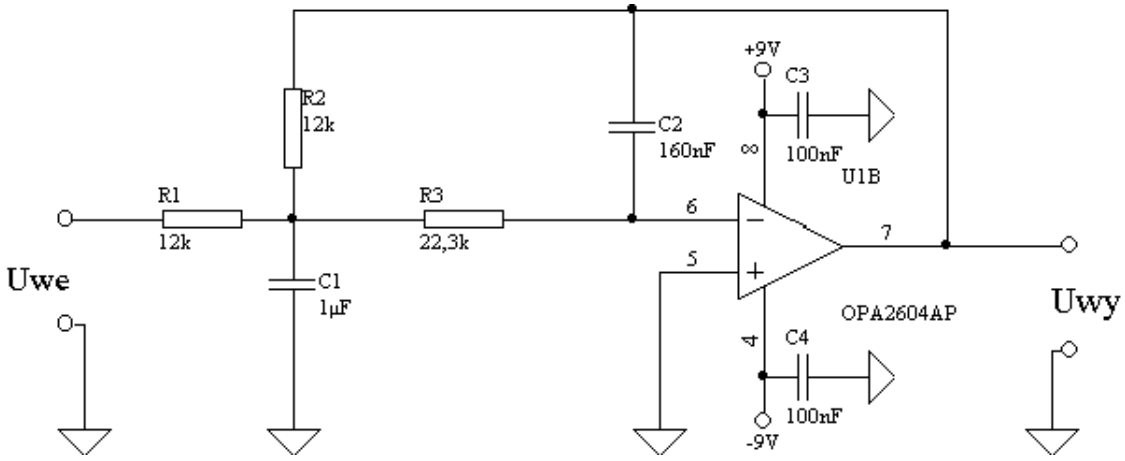
tłumienia sygnału współbieżnego. Według danych katalogowych producenta napięcie niezrównoważenia wzmacniacza wynosi maksymalnie 50μV. Współczynnik CMRR wynosi minimalnie 110 dB i dla przyjętych niskich częstotliwości sygnału pomiarowego jest stały.

4.2 Filtr dolnoprzepustowy

Częstotliwość sygnału z elektrod powinna zawierać się typowo w przedziale 5 – 25 Hz. Ponieważ mogą one wykraczać poza ten przedział przyjęto następujące parametry filtru:

- maksymalnie płaska charakterystyka amplitudowa;
- częstotliwość graniczna f_0 = 25 Hz;
- wzmocnienie w paśmie przepustowym –1V/V;
- tłumienie sygnałów powyżej częstotliwości granicznej –20 dB/dek.

Zastosowane rozwiązanie układowe filtru jest przedstawione na rysunku 3. Jest to układ drugiego rzędu z wielokrotnym ujemnym sprzężeniem zwrotnym. Ogniwo podstawowe z oznaczeniami elementów wykorzystanymi w obliczeniach zostało przedstawione poniżej.



Ogólna postać transmitancji filtru dolnoprzepustowego II-rzędu jest następująca:

$$T_u(s) = \frac{A_{uf} {w_0}^2}{s^2 + s\frac{w_0}{Q} + {w_0}^2} \tag{1}$$

gdzie A_{uf} - wzmocnienie w paśmie przepustowym,

w_0 - pulsacja graniczna,

Q - dobroć bieguna transmitancji.

W zastosowanym filtrze:

$$A_{uf} = -\frac{R_2}{R_1} \tag{2}$$

$$w_0 = \frac{1}{\sqrt{C_1 C_2 R_2 R_3}} \tag{3}$$

$$Q = \frac{\sqrt{\frac{C_1}{C_2}}}{\frac{\sqrt{R_2 R_3}}{R_1} + \sqrt{\frac{R_3}{R_2}} + \sqrt{\frac{R_2}{R_3}}} \tag{4}$$

Dla filtru o maksymalnie płaskiej charakterystyce wartość dobroci wynosi

$$Q = \sqrt{\frac{1}{2}} = 0.707$$

Podstawiając do zależności (2), (3), (4) przyjęte wartości parametrów:

A_{uf} = -1, f_0 = 25 Hz oraz Q = 0.707 wyznaczone zostały wartości elementów pasywnych filtru. Są one następujące:

R1 = R2 = 12 kΩ, R3 = 22.3 kΩ, C1 = 1 μF, C2 = 160 nF.

Ze względu na rozrzut wartości elementów zamiast opornika R3 zastosowano opornik o mniejszej wartości (20 kΩ) połączony szeregowo z potencjometrem obrotowym o wartości 4.7 kΩ. Dla lepszego wytłumienia niepożądanych sygnałów o częstotliwości powyżej 25 Hz, w torze pomiarowym zostały umieszczone dwa ogniwa podstawowe połączone kaskadowo.

5. Parametry techniczne

W rozdziale tym przedstawiono zmierzone parametry techniczne rejestratora holterowskiego z pamięcią cyfrową oraz ich porównanie z założonymi lub wyliczonymi teoretycznie. Niektóre parametry wzmacniacza INA118 nie były badane, a zostały jedynie przytoczone na podstawie katalogów ich producenta.

5.1 Warunki przeprowadzenia pomiarów

W stopniu wejściowym zmierzone zostały następujące parametry: współczynnik CMRR wzmacniacza pomiarowego, jego wejściowe napięcie niezrównoważenia i rezystancję wejściową. Przy pomiarze współczynnika CMRR wzmacniacza pomiarowego podano na obydwa wejścia sygnał sinusoidalny o częstotliwości 10 Hz i amplitudzie 1V. Charakterystykę amplitudową filtra dolnoprzepustowego badano podając na wejście sygnał sinusoidalny o amplitudzie 0.5 V i częstotliwości z zakresu 10 – 1000 Hz. Wyniki tych pomiarów zestawione są w tabeli 1 a charakterystyka przedstawiona na rysunku 4 w rozdziale szóstym.

5.2 Wyniki pomiarów

Zasilanie

Napięcie zasilania układów analogowych:	± 9V
Pobór prądu:	15 mA
Napięcie zasilania układów cyfrowych:	+ 5V
Pobór prądu:	32 mA

Wzmacniacz wejściowy

Zakres napięć wejściowych (max):	± 9 V
Wejściowe napięcie niezrównoważenia:	2 mV
Współczynnik CMRR:	106 dB
Rezystancja wejściowa:	> 10 MΩ
Wzmocnienie:	100 V/V
Zakres częstotliwości sygnału wejściowego:	0 - 25 Hz

Filtr dolnoprzepustowy

Charakterystyka maksymalnie płaska, rząd 4

Częstotliwość graniczna:	25 Hz
Wzmocnienie w paśmie przepustowym:	1 V/V
Tłumienie:	20 dB/dek

Tabela 1. Charakterystyka FDP (amplituda Uwe = 0,5 V)

częstotliwość f [Hz]	wzmocnienie A [dB]
10	0
20	0
25	0
30	- 0,3
35	- 1,9
40	- 5,2
50	- 11,9
60	- 17,5
70	- 22,5
80	- 27,1
90	- 30,8
100	-33,9

Tor pomiarowy

Zabezpieczenie linii wejściowych wzmacniacza	± 40 kV
Rozdzielczość przetwarzania A/C	12 bit
Zasilanie	± 9 V
Pobór prądu	15 mA

Interfejs cyfrowy

Zasilanie	+5V
Pobór prądu	32 mA
Kanał transmisji	RS-232 C

Transmisja dwukierunkowa

- szybkość maksymalna 19200 bit/s
- szybkość minimalna 4800 bit/s
- dane: 8 bit, 1 bit stopu, bez bitu parzystości

Spis schematów i rysunków

Schematy umieszczone zostały w dodatku C

Literatura

1. Stanisław Kuta, G. Krajewski - “Układy elektroniczne cz. II”, Wydawnictwo AGH , Kraków 1994
2. Andrzej Rydzewski - “Mikrokomputery jednoukładowe rodziny MCS-51”,WNT, Warszawa 1995
3. Nadachowski M., Kulka Z. - “Analogowe układy scalone”, WKŁ , Warszawa 1985
4. Mielczarek W. - “Szeregowe interfejsy cyfrowe”,Wydawnictwo Helion , Gliwice 1993
5. Gajewski P., Turczyński J. - “Cyfrowe układy scalone CMOS”,WKŁ , Warszawa 1990
6. Metzger P. - “Anatomia PC” , Wydawnictwo Helion , Gliwice 1993
7. Robin Jones, Ian Stewart - “Sztuka programowania w języku C” WNT , Warszawa
8. Piotr Wróblewski - “Programowanie w Windows dla praktyków”, Wydawnictwo Helion , 1997
9. Stanisław Bogdanowicz - “Najłatwiejsza elektrokardiografia - wprowadzenie do elektrokardiografii klinicznej”, Oficyna Wydawnicza “Impuls”, Kraków 1993
10. Noty aplikacyjne firm: Atmel, National Semiconductor, BURR-BROWN, Sony

Dodatek A

```
;*********************************************************************
;*                    Kod źródłowy programu sterownika 89C52          *
;*       dla procesora Atmel AT89C52     A:\AT89C52\Rejestr.asm       *
;*********************************************************************
;DELAY1         ( MOV L,#XX ) Opóźnienie L*1 MS
;DELAY100       ( MOV L,#XX ) Opóźnienie L*100 MS
; Funkcje obsługi wyświetlacza LCD
;LCD_INI        Inicjalizacja wyświetlacza LCD
;LCD_WR         ( MOV CHAR,#'X', MOV DPTR,#XXXX ) Wpis znaku do wyświetlacza
;LCD_RD         ( wynik w CHAR ) Odczyt znaku z wyświetlacza
;LCD_CSR        Przesunięcie kursora w prawo
;LCD_CSL        Przesunięcie kursora w lewo
;LCD_CAH        Kursor “at home”
;LCD_CON        Włączenie kursora
;LCD_COFF       Wyłączenie kursora
;LCD_DISPL      ( MOV DISPL,#P ) Ustawianie parametrów wyświetlania i kursora
;                1.Display On/Off Control:
;                  P: 00001DCB   D: 1 - all display ON   0 - display OFF
;                                C: 1 - cursor ON        0 - cursor OFF
;                                B: 1 - blink of cursor  0 - blink OFF
;                2.Entry Mode Set:
;                  P: 000001DS   D: 1 - increment 0 - decrement counter
;                                S: 1 - with display shift
;                3.Function Set:
;                  P: 001LNF00   L: 1 - 8-bit interf     0 - 4-bit interf
;                                N: 1 - 1/16Duty         0 - 1/8Duty,1/11Duty
;                                F: 1 - 5x10 dots        0 - 5x7 dots
;LCD_DSR        Przesunięcie napisu w prawo (Display shift right)
;LCD_DSL        Przesunięcie napisu w lewo (Display shift left)
;LCD_CLEAR      Czyszczenie wyświetlacza
;LCD_WRS        ( MOV DPTR,#XXXX ) Wpis tekstu, którego początek znajduje
;                  się w pamięci programu pod adresem w DPTR.
;                  Tekst musi być zakończony liczba 8.
;LCD_DDRAM      ( MOV ADR,#XX ) ustawienie adresu DDRAM dla LCD
;                  0-39 linia 1, 64-103 linia 2
;LCD_CGRAM      ( MOV ADR,#XX ) ustawienie adresu CGRAM dla LCD
;LCD_SETCHAR    ( MOV ADR,#XX - adres początku wpisywania w CGRAM [00H]
;                  MOV DPTR,#XXXX , adres początku tablicy wzorców zakończonej
;                  zerem znajdującej się w pamięci programu ) Wczytanie
;                  wzorców znaków do CGRAM.
;BEEP           ( MOV L,#XX - długość trwania dźwięku ) Sygnał dźwiękowy
;TEST_KEY          testowanie wciśnięcia przycisku, CHAR - wynik (numer
;                  przycisku)
;SET_TIME       ( MOV STPAR,#XX - 1 - zmiana czasu w rejestrach TSH,TSM,TSS
;                  2 - zmiana czasu w rejestrach TRH,TRM,TRS)
;                  podprogram ustawiania czasu systemowego
;CONV_BCDL      ( MOV CHAR,#XX, wynik w CHAR ) konwersja BCD L na ASCII
;CONV_BCDH      ( MOV CHAR,#XX, wynik w CHAR ) konwersja BCD H na ASCII
;CONV_BCD2BIN   ( MOV CHAR,#XX (BCD), wynik w CHAR (BIN)) konwersja BCD na BIN
;DISPLAY_HEX    ( MOV CHAR,#XX (BIN)) wyświetla wynik w postaci HEX na
;                  aktualnej pozycji kursora zawartość CHAR
;---------------------------------------------------------------------
; Rejestry systemu
RS_T_FLAG       BIT       00H          ;FLAGA KONCA TRANSMISJI BAJTU PRZEZ RS
RS_R_FLAG       BIT       01H          ;FLAGA ODEBRANIA BAJTU PRZEZ RS
MBREAK          BIT       02H          ;FLAGA BLOKADY KOMENDY 'm'
RECORD_FLAG     BIT       03H          ;FLAGA ZEZWOLENIA NA REJESTRACJE
REC_FIRSTPR     BIT       04H          ;FLAGA POBRANIA PROBKI ODNIESIENIA

RBUF            REG       21H          ;ADRESOWALNY BITOWO BUFOR REJESTRU
CHAR            REG       22H          ;BUFOR ZNAKU DLA LCD
ADR             REG       23H          ;BUFOR ADRESU
DISPL           REG       24H          ;REJESTR PARAMETROW WYSWIETLANIA
KEY             REG       25H          ;BUFOR NUMERU KLAWISZA
NMENU0          REG       26H          ;LICZNIK MENU 0
NMENU1          REG       27H          ;LICZNIK MENU 1
NMENU2          REG      28H          ;LICZNIK MENU 2
```

```
TSH             REG     29H             ;CZAS SYSTEMU: GODZINY
TSM             REG     2AH             ;CZAS SYSTEMU: MINUTY
TSS             REG     2BH             ;CZAS SYSTEMU: SEKUNDY
TRH             REG     2CH             ;CZAS REJESTRACJI: GODZINY
TRM             REG     2DH             ;CZAS REJESTRACJI: MINUTY
TRS             REG     2EH             ;CZAS REJESTRACJI: SEKUNDY
STATUS          REG     2FH             ;STATUS: t-WYŚWIETLANIE CZASU,REJESTRACJA
                                        ;m-WYŚWIETLANIE MENU, p-PROGRAMOWANIE,
                                        ;s-WYSYŁANIE DANYCH, e-MONITOR EKG
BUFH            REG     30H             ;BUFOR GODZINY
BUFM            REG     31H             ;BUFOR MINUTY
BUFS            REG     32H             ;BUFOR SEKUNDY
STPAR           REG     33H             ;PARAMETRY PODPROGRAMU SET_TIME
L               REG     34H             ;LICZNIK POWTORZEN PETLI
BSBUF           REG     35H             ;BUFOR ODEBRANEGO ZNAKU Z PORTU SZEREG.
RSBAUD          REG     36H             ;AKTUALNIE UŻYWANA SZYBKOŚĆ PORTU RS
                                        ;4800  - 1, 9600  - 2, 19200 - 3
BDPH            REG     37H             ;BUFOR DPH
BDPL            REG     38H             ;BUFOR DPL
NREC            REG     39H             ;NUMER AKTUALNIE NAGRYWANEGO REKORDU
LBARH           REG     3AH             ;HB-LICZBA BAJTÓW AKTUALNEJ REJESTRACJI
LBARL           REG     3BH             ;LB
BADRH           REG     3CH             ;HB-ADRES ZAPISYWANEJ DO RAM PRÓBKI
BADRL           REG     3DH             ;LB
BPROBH          REG     3EH             ;LH-BUFOR PRÓBKI
BPROBL          REG     3FH             ;LB
;-------------------------------------------------------------------------
; Definicja bitów obsługujących wyświetlacz
RS              BIT     P1.7
RW              BIT     P1.1
CS              BIT     P1.2            ;aktywne 1
;-------------------------------------------------------------------------
; Definicja bitów obsługujących pamięć RAM
RAM             BIT     P3.4            ;blokada pamięci
CHIP            BIT      P2.7           ;selekcja układu
;-------------------------------------------------------------------------
; Definicja bitów obsługujących klawiaturę
KEYIN           BIT     P1.0            ;P1.7
KEY1            BIT     P2.2            ;P1.0  LEFT
KEY2            BIT     P2.3            ;P1.1  UP
KEY3            BIT     P2.4            ;P1.6  DOWN
KEY4            BIT     P2.5            ;P1.5  RIGHT
KEY5            BIT     P2.6            ;P1.4  ENTER
;-------------------------------------------------------------------------
; Definicja bitów obsługujących przetwornik piezoelektryczny
BUZ          BIT    P3.3
;-------------------------------------------------------------------------
; Definicja bitów obsługujących przetwornik A/C
AC_PWRD         BIT     P3.5
AC_BUSY         BIT     P1.3
AC_CS           BIT     P3.2
AC_RC           BIT     P1.4
AC_DATA         BIT     P1.5
AC_DATACLK      BIT     P1.6
;-------------------------------------------------------------------------
; Parametry dla LCD_DISPL  ( np.: MOV DISPL,#DC_CURON )
; 1.Display On/Off Control:
DC_CURON        EQU     00001111B       ;Cursor ON blink ON
DC_CUROFF       EQU     00001100B       ;Cursor OFF blink OFF
DC_DISPLON      EQU     00001111B       ;Display ON cursor ON blink ON
DC_DISPLOFF     EQU     00001000B       ;Display OFF
DC_DISPLONBOFF  EQU     00001110B       ;Display ON cursor ON blink OFF
; 2.Entry Mode Set:
EMS_NOSHIFT     EQU     00000100B       ;
EMS_SHIFT       EQU     00000101B       ;Przesuwanie całego ekranu
; 3.Function Set
FS_2L           EQU     00111100B       ;Dwie linie
FS_1L           EQU     00110000B       ;Jedna linia
;-------------------------------------------------------------------------
;Definicje nazw klawiszy
KEY_LEFT     EQU   00110001B
KEY_RIGHT    EQU    00110010B
```

```
KEY_UP        EQU    00110011B
KEY_DOWN      EQU    00110100B
KEY_ENTER     EQU    00110101B
KEY_NOPRESS   EQU    00000000B

T2CON         EQU   0C8H
RLDH          EQU   0CBH
RLDL          EQU   0CAH

              ORG 0000H
              MOV SP,#40H
              LJMP START
              ORG 000BH           ;początek obsługi przerwania z T0
              LJMP AC_PROBE
              ORG 0023H           ;początek obsługi przerwania z RS
              LJMP RS_INT
              ORG 002BH           ;początek obsługi przerwania z T2
              LJMP CLOCK
              ORG 0033H
;*************************************
;*            Program główny          *
;*************************************
PMENU00:      DB 'Funkcje        ',8
PMENU11:      DB ' Ustaw         ',8
PMENU12:      DB '  czas         ',8
PMENU13:      DB '   rejestracji ',8
PMENU14:      DB '   zegara      ',8
PMENU15:      DB '  parametry    ',8
PMENU16:      DB '   pr',5,'bkowania ',8
PMENU17:      DB '   transmisji  ',8
PMENU18:      DB '  dane pacjenta',8
PMENU21:      DB ' Rejestruj     ',8
PMENU31:      DB ' Testuj pami',2,1,' ',8
PMENU41:      DB ' Informacja    ',8

START:        LCALL STARTUP
              LCALL LCD_COFF

              MOV NMENU0,#2

PMENU0:       LCALL LCD_CLEAR           ;kasowanie wyświetlacza
              LCALL LCD_CAH

TESTNMENU0:   MOV DPTR,#PMENU00
              LCALL LCD_WRS
              MOV A,NMENU0
              CJNE A,#2,NOM02
              MOV DPTR,#PMENU11
              LJMP PR0
NOM02:        CJNE A,#3,NOM03
              MOV DPTR,#PMENU21
              LJMP PR0
NOM03:        CJNE A,#4,NOM04
              MOV DPTR,#PMENU31
              LJMP PR0
NOM04:        CJNE A,#5,PR0
              MOV DPTR,#PMENU41
              LJMP PR0
PR0:          LCALL PRINTARR

KEYLOOP0:     LCALL TEST_KEY
              MOV A,KEY
              CJNE A,#KEY_UP,NOUP0
              LJMP UP0
NOUP0:        CJNE A,#KEY_DOWN,NODOWN0
              LJMP DOWN0
NODOWN0:      CJNE A,#KEY_RIGHT,NORIGHT0
              LJMP RIGHT0

NORIGHT0:     MOV A,STATUS
              CJNE A,#'e',NO_EKG0
              LJMP MONITOR_EKG
```

```
NO_EKG0:      CJNE A,#'p',NO_PROGRAM0
              LJMP PROGRAM
NO_PROGRAM0:  CJNE A,#'s',NO_SEND0
              LJMP SEND_DATA
NO_SEND0:     LJMP KEYLOOP0

UP0:          DEC NMENU0
              MOV A,NMENU0
              CJNE A,#4,UP04
              MOV DPTR,#PMENU31
              LCALL PRINTARR
              LJMP UP01
UP04:         CJNE A,#3,UP03
              MOV DPTR,#PMENU21
              LCALL PRINTARR
              LJMP UP01
UP03:         CJNE A,#2,UP02
              MOV DPTR,#PMENU11
              LCALL PRINTARR
              LJMP UP01
UP02:         CJNE A,#1,UP01
              MOV L,#1
              LCALL BEEP
              INC NMENU0
UP01:         LCALL KEYDELAY
              LJMP KEYLOOP0

DOWN0:        INC NMENU0
              MOV A,NMENU0
              CJNE A,#3,DOWN03
              MOV DPTR,#PMENU21
              LCALL PRINTARR
              LJMP DOWN06
DOWN03:       CJNE A,#4,DOWN04
              MOV DPTR,#PMENU31
              LCALL PRINTARR
              LJMP DOWN06
DOWN04:       CJNE A,#5,DOWN05
              MOV DPTR,#PMENU41
              LCALL PRINTARR
              LJMP DOWN06
DOWN05:       CJNE A,#6,DOWN06
              MOV L,#1
              LCALL BEEP
              DEC NMENU0
DOWN06:       LCALL KEYDELAY
              LJMP KEYLOOP0

RIGHT0:       MOV A,NMENU0
              CJNE A,#2,NOPMENU02       ;->Menu: Ustaw
              MOV NMENU1,#2
              LJMP PMENU1
NOPMENU02:    CJNE A,#3,NOPMENU03       ;->Menu: Rejestruj

              LCALL LCD_CLEAR
              MOV ADR,#4
              LCALL LCD_DDRAM
              LCALL TIMES_DISPL

              LCALL NOPMENU_SET

              MOV STATUS,#'t'           ;Tryb wyświetlania czasu
TNOM:         MOV R4,STATUS

              CJNE R4,#'m',TNOM

              MOV NMENU0,#3
              LJMP PMENU0

NOPMENU03:    CJNE A,#4,NOPMENU04       ;->Menu: Testuj pamięć

              LCALL RAM_TEST
```

```
NOPMENU_RTL: LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_ENTER,NOPMENU_RTL
NOPMENU_NP:  LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_NOPRESS,NOPMENU_NP
             LJMP START

NOPMENU04:   CJNE A,#5,NOPMENU05       ;->Menu: Informacja
             MOV NMENU1,#2
             LJMP PMENUINFO

NOPMENU05:   LJMP KEYLOOP0

NOPMENU_SET: MOV NREC,#1
             MOV BDPH,#00H
             MOV BDPL,#42H
             MOV R3,#0               ;TRH1(NREC)
             LCALL SRT_LICZ
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOVX A,@DPTR
             MOV TRH,A

             MOV BDPH,#00H
             MOV BDPL,#42H
             MOV R3,#1               ;TRM1(NREC)
             LCALL SRT_LICZ
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOVX A,@DPTR
             MOV TRM,A

             MOV BDPH,#00H
             MOV BDPL,#42H
             MOV R3,#2               ;TRS1(NREC)
             LCALL SRT_LICZ
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOVX A,@DPTR
             MOV TRS,A

             MOV BDPH,#00H
             MOV BDPL,#42H
             MOV R3,#7               ;LBH1(NREC)
             LCALL SRT_LICZ
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOVX A,@DPTR
             MOV LBARH,A

             MOV BDPH,#00H
             MOV BDPL,#42H
             MOV R3,#8               ;LBL1(NREC)
             LCALL SRT_LICZ
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOVX A,@DPTR
             MOV LBARL,A

             MOV BDPH,#00H
             MOV BDPL,#42H
             MOV R3,#5               ;DPH1(NREC)
             LCALL SRT_LICZ
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOVX A,@DPTR
             MOV BADRH,A
             MOV BDPH,#00H
             MOV BDPL,#42H
             MOV R3,#6               ;DPL1(NREC)
             LCALL SRT_LICZ
```

```
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOVX A,@DPTR
             MOV BADRL,A

             RET
;--------------------------------------
PRINTARR:    MOV ADR,#40
             LCALL LCD_DDRAM
             LCALL LCD_WRS

             MOV ADR,#40
             LCALL LCD_DDRAM
             MOV CHAR,#01111110B       ;strzałka w prawo
             LCALL LCD_WR
             RET

KEYDELAY:    MOV L,#2                  ;opóźnienie klawiatury
             LCALL DELAY100
             RET
;--------------------------------------
;|           TEST PAMIECI RAM          |
;--------------------------------------
RAMTESTI:    DB 'Test pami',2,'ci RAM',8
RAMTESTOK:   DB '32kB OK.',8
RAMTESTER:   DB 'Error: ',8

RAM_TEST:    LCALL LCD_CLEAR
             LCALL LCD_CAH

             MOV DPTR,#RAMTESTI
             LCALL LCD_WRS
             MOV BDPH,#07FH         ;test 32kB RAM
             MOV BDPL,#0FFH

RAMT_NEXTTEST:
             MOV DPH,BDPH
             MOV DPL,BDPL
             MOV A,#11111111B
             MOVX @DPTR,A
             MOV A,#0
             MOVX A,@DPTR
             CJNE A,#11111111B,RAMT_ERROR
             MOV A,#00000000B
             MOVX @DPTR,A
             MOV A,#0
             MOVX A,@DPTR
             CJNE A,#00000000B,RAMT_ERROR
             MOV A,#10101010B
             MOVX @DPTR,A
             MOV A,#0
             MOVX A,@DPTR
             CJNE A,#10101010B,RAMT_ERROR
             MOV A,#01010101B
             MOVX @DPTR,A
             MOV A,#0
             MOVX A,@DPTR
             CJNE A,#01010101B,RAMT_ERROR

             CLR C
             MOV A,BDPL
             SUBB A,#1
             MOV BDPL,A
             MOV A,BDPH
             SUBB A,#0
             MOV BDPH,A
             JNC RAMT_NEXTTEST
             MOV ADR,#64
             LCALL LCD_DDRAM
             MOV DPTR,#RAMTESTOK
             LCALL LCD_WRS
             RET
```

```
RAMT_ERROR:  MOV ADR,#64
             LCALL LCD_DDRAM
             MOV DPTR,#RAMTESTER
             LCALL LCD_WRS
             MOV CHAR,BDPH
             LCALL DISPLAY_HEX
             MOV CHAR,BDPL
             LCALL DISPLAY_HEX
             MOV CHAR,#'H'
             LCALL LCD_WR
             RET
;-------------------------------------
;            Menu: Ustaw
;-------------------------------------
PMENU1:      LCALL LCD_CLEAR           ;Menu: Ustaw
             LCALL LCD_CAH

             MOV DPTR,#PMENU11+1
             LCALL LCD_WRS
             MOV A,NMENU1
             CJNE A,#2,NOM12
             MOV DPTR,#PMENU12+1
             LJMP PR1
NOM12:       CJNE A,#3,NOM13
             MOV DPTR,#PMENU15+1
             LJMP PR1
NOM13:       MOV DPTR,#PMENU18+1
PR1:         LCALL PRINTARR

             LCALL KEYDELAY

KEYLOOP1:    LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_UP,NOUP1
             LJMP UP1
NOUP1:       CJNE A,#KEY_DOWN,NODOWN1
             LJMP DOWN1
NODOWN1:     CJNE A,#KEY_LEFT,NOLEFT1
             LJMP PMENU0               ;Powrót do PMENU0
NOLEFT1:     CJNE A,#KEY_RIGHT,NORIGHT1
             LJMP RIGHT1
NORIGHT1:    MOV A,STATUS
             CJNE A,#'e',NO_EKG1
             LJMP MONITOR_EKG
NO_EKG1:     CJNE A,#'p',NO_PROGRAM1
             LJMP PROGRAM
NO_PROGRAM1: CJNE A,#'s',NO_SEND1
             LJMP SEND_DATA
NO_SEND1:    LJMP KEYLOOP1

UP1:         DEC NMENU1
             MOV A,NMENU1
             CJNE A,#2,UP12
             MOV DPTR,#PMENU12+1
             LCALL PRINTARR
             LJMP UP11
UP12:        CJNE A,#1,UP11
             MOV L,#1
             LCALL BEEP
             INC NMENU1
UP11:        LCALL KEYDELAY
             LJMP KEYLOOP1

DOWN1:       INC NMENU1
             MOV A,NMENU1
             CJNE A,#3,DOWN13
             MOV DPTR,#PMENU15+1
             LCALL PRINTARR
             LJMP DOWN15
DOWN13:      CJNE A,#4,DOWN15
             MOV L,#1
```

```
             LCALL BEEP
             DEC NMENU1
DOWN15:      LCALL KEYDELAY
             LJMP KEYLOOP1

RIGHT1:      MOV A,NMENU1
             CJNE A,#2,NOPMENU12
             MOV NMENU2,#2
             LJMP PMENU1_1       ;->Menu: Ustaw->czas
NOPMENU12:   CJNE A,#3,NOPMENU13
             MOV NMENU2,#2
             LJMP PMENU1_2              ;->Menu: Ustaw->parametry
NOPMENU13:   LJMP KEYLOOP1
;--------------------------------------
;          Menu: Ustaw->czas
;--------------------------------------
PMENU1_1:    LCALL LCD_CLEAR
             LCALL LCD_CAH

             MOV DPTR,#PMENU11+1
             LCALL LCD_WRS
             MOV ADR,#6
             LCALL LCD_DDRAM
             MOV DPTR,#PMENU12+2
             LCALL LCD_WRS

             MOV A,NMENU2
             CJNE A,#2,NOM1_12
             MOV DPTR,#PMENU13+2
             LJMP PR1_1
NOM1_12:     MOV DPTR,#PMENU14+2
PR1_1:       LCALL PRINTARR

             LCALL KEYDELAY

KEYLOOP1_1:  LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_UP,NOUP1_1
             LJMP UP1_1
NOUP1_1:     CJNE A,#KEY_DOWN,NODOWN1_1
             LJMP DOWN1_1
NODOWN1_1:   CJNE A,#KEY_LEFT,NOLEFT1_1
             LJMP PMENU1                    ;Powrót do PMENU1
NOLEFT1_1:   CJNE A,#KEY_RIGHT,NORIGHT1_1
             LJMP RIGHT1_1
NORIGHT1_1:  MOV A,STATUS
             CJNE A,#'e',NO_EKG1_1
             LJMP MONITOR_EKG
NO_EKG1_1:   CJNE A,#'p',NO_PROGRAM1_1
             LJMP PROGRAM
NO_PROGRAM1_1:
             CJNE A,#'s',NO_SEND1_1
             LJMP SEND_DATA
NO_SEND1_1:  LJMP KEYLOOP1_1

UP1_1:       DEC NMENU2
             MOV A,NMENU2
             CJNE A,#2,UP1_12
             MOV DPTR,#PMENU13+2
             LCALL PRINTARR
             LJMP UP1_11
UP1_12:      CJNE A,#1,UP1_11
             INC NMENU2
UP1_11:      LCALL KEYDELAY
             LJMP KEYLOOP1_1

DOWN1_1:     INC NMENU2
             MOV A,NMENU2
             CJNE A,#3,DOWN1_13
             MOV DPTR,#PMENU14+2
             LCALL PRINTARR
             LJMP DOWN1_14
```

```
DOWN1_13:    CJNE A,#4,DOWN1_14
             DEC NMENU2
DOWN1_14:    LCALL KEYDELAY
             LJMP KEYLOOP1_1

RIGHT1_1:    MOV A,NMENU2
             CJNE A,#2,NOPMENU1_12     ;->Ustaw czas rejestracji
             LJMP SET_RECTIME
NOPMENU1_12: CJNE A,#3,NOPMENU1_13     ;->Ustaw czas zegara
             MOV STPAR,#1
             LCALL SET_TIME
             LJMP PMENU1_1
NOPMENU1_13: LJMP KEYLOOP1_1

;--------------------------------------
;    Menu: Ustaw->czas->rejestracji
;--------------------------------------
SET_RECTIME: LCALL LCD_CLEAR
             LCALL LCD_CAH
             MOV NREC,#1

SRT_TESTNREC:
             MOV DPTR,#0040H         ;ilość rekordów
             MOVX A,@DPTR
             INC A
             CJNE A,NREC,SRT_NOZERO
             LJMP SRT_WDODAJ
SRT_NOZERO:  LCALL SRT_WREKORD

SRT_LOOPNP1: LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_NOPRESS,SRT_LOOPNP1

SRT_KEYLOOP: LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_UP,SRT_NOUP
             MOV A,#1
             CJNE A,NREC,SRT_NOBEG
             SJMP SRT_KEYLOOP
SRT_NOBEG:   DEC NREC
             LJMP SRT_TESTNREC
SRT_NOUP:    CJNE A,#KEY_DOWN,SRT_NODOWN
             MOV DPTR,#0040H
             MOVX A,@DPTR
             INC A
             CJNE A,NREC,SRT_INC
             SJMP SRT_KEYLOOP
SRT_INC:     INC NREC
             LJMP SRT_TESTNREC
SRT_NODOWN:  CJNE A,#KEY_RIGHT,SRT_NORIGHT
             MOV DPTR,#0040H
             MOVX A,@DPTR
             CJNE A,NREC,SRT_NZMIENOST
             LCALL SRT_DODAJ
             LCALL SRT_WREKORD
             SJMP SRT_KEYLOOP
SRT_NZMIENOST:
             SJMP SRT_KEYLOOP
SRT_NORIGHT:
             CJNE A,#KEY_LEFT,SRT_NOLEFT
             MOV DPTR,#0040H
             MOVX A,@DPTR
             CJNE A,NREC,SRT_NOLEFT
             LJMP SRT_USUN
SRT_NOLEFT:  SJMP SRT_KEYLOOP
;--------------------------------------
SRT_DODAJ:   MOV BDPH,#00H       ;Ustawianie czasu początku rejestracji
             MOV BDPL,#42H
             MOV R3,#0
             LCALL SRT_LICZ
             MOV DPH,BDPH
             MOV DPL,BDPL
```

```
MOVX A,@DPTR
MOV TRH,A

MOV BDPH,#00H
MOV BDPL,#42H
MOV R3,#1
LCALL SRT_LICZ
MOV DPH,BDPH
MOV DPL,BDPL
MOVX A,@DPTR
MOV TRM,A

MOV BDPH,#00H
MOV BDPL,#42H
MOV R3,#2
LCALL SRT_LICZ
MOV DPH,BDPH
MOV DPL,BDPL
MOVX A,@DPTR
MOV TRS,A

MOV STPAR,#2
LCALL SET_TIME

MOV BDPH,#00H
MOV BDPL,#42H
MOV R3,#0
LCALL SRT_LICZ
MOV DPH,BDPH
MOV DPL,BDPL
MOV A,TRH
MOVX @DPTR,A

MOV BDPH,#00H
MOV BDPL,#42H
MOV R3,#1
LCALL SRT_LICZ
MOV DPH,BDPH
MOV DPL,BDPL
MOV A,TRM
MOVX @DPTR,A

MOV BDPH,#00H
MOV BDPL,#42H
MOV R3,#2
LCALL SRT_LICZ
MOV DPH,BDPH
MOV DPL,BDPL
MOV A,TRS
MOVX @DPTR,A

MOV BDPH,#00H              ;Ustawianie czasu trwania rejestracji
MOV BDPL,#42H
MOV R3,#3
LCALL SRT_LICZ
MOV DPH,BDPH
MOV DPL,BDPL
MOVX A,@DPTR
MOV TRM,A

MOV BDPH,#00H
MOV BDPL,#42H
MOV R3,#4
LCALL SRT_LICZ
MOV DPH,BDPH
MOV DPL,BDPL
MOVX A,@DPTR
MOV TRS,A

LCALL SET_TIMET

MOV BDPH,#00H
```

```
              MOV BDPL,#42H
              MOV R3,#3
              LCALL SRT_LICZ
              MOV DPH,BDPH
              MOV DPL,BDPL
              MOV A,TRM
              MOVX @DPTR,A

              MOV BDPH,#00H
              MOV BDPL,#42H
              MOV R3,#4
              LCALL SRT_LICZ
              MOV DPH,BDPH
              MOV DPL,BDPL
              MOV A,TRS
              MOVX @DPTR,A

              RET
;--------------------------------------
; Ustawianie czasu trwania rejestracji
; A,R4,R5,R6

SET_TIMET:    MOV ADR,#00H              ;adres początku w CGRAM (00H)
              MOV DPTR,#CHAR_SPEC       ;początek tablicy w pamięci programu
              LCALL LCD_SETCHAR         ;ustawienie wzorców
              LCALL LCD_CLEAR
              LCALL LCD_CAH
              MOV BUFS,TRS
              MOV BUFM,TRM
              MOV R4,#71                ;początkowa pozycja kursora

              LCALL TIMET_DISPL_U

              MOV ADR,R4
              LCALL LCD_DDRAM
              MOV CHAR,#00000000B
              LCALL LCD_WR
              INC R4
              MOV ADR,R4
              LCALL LCD_DDRAM
              MOV CHAR,#00000000B
              LCALL LCD_WR

STT_TEST:     LCALL KEYDELAY

STT_LOOPNP1:  LCALL TEST_KEY
              MOV A,KEY
              CJNE A,#KEY_NOPRESS,STT_LOOPNP1

STTLOOP1:     LCALL TEST_KEY
              MOV A,KEY
              CJNE A,#KEY_ENTER,STTNOENTER
              LCALL STT_UPDATE
              MOV TRS,BUFS
              MOV TRM,BUFM
              LCALL LCD_CLEAR
              LCALL LCD_CAH
              MOV ADR,#00H
              MOV DPTR,#CHAR_PL
              LCALL LCD_SETCHAR         ;ustawienie poprzednich wzorców
              RET

STTNOENTER:   CJNE A,#KEY_RIGHT,STTNORIGHT
              LJMP STTRIGHT
STTNORIGHT:   CJNE A,#KEY_LEFT,STTNOLEFT
              LJMP STTLEFT
STTNOLEFT:    CJNE A,#KEY_UP,STTNOUP
              LJMP STTDOWN
STTNOUP:      CJNE A,#KEY_DOWN,STTNODOWN
              LJMP STTUP
STTNODOWN:    LJMP STTLOOP1
```

```
STTRIGHT:    CJNE R4,#75,STTSHIFTR
             LJMP STTLOOP1
STTSHIFTR:   DEC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#' '
             LCALL LCD_WR
             INC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#' '
             LCALL LCD_WR
             INC R4
             INC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#0
             LCALL LCD_WR
             INC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#0
             LCALL LCD_WR

             LCALL KEYDELAY
             LJMP STTLOOP1

STTLEFT:     CJNE R4,#72,STTSHIFTL
             LJMP STTLOOP1
STTSHIFTL:   MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#' '
             LCALL LCD_WR
             DEC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#' '
             LCALL LCD_WR
             DEC R4
             DEC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#0
             LCALL LCD_WR
             DEC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#0
             LCALL LCD_WR
             INC R4
             LCALL KEYDELAY
             LJMP STTLOOP1

STTUP:       CJNE R4,#72,STTUNO72
             MOV A,BUFM
             MOV R5,BUFM
             ANL A,#00001111B
             CLR CY
             SUBB A,#1
             JNC STTUNOCY1M
             MOV A,#00001001B
             MOV R6,A
             MOV A,R5
             ANL A,#11110000B
             SWAP A
             CLR CY
             SUBB A,#1
             JNC STTUNOCY2M
             MOV A,#5
STTUNOCY2M:  SWAP A
             ORL A,R6
             PUSH BUFM
```

```
                MOV BUFM,A
                LCALL STT_COMP
                MOV A,CHAR
                CJNE A,#1,STTU_NOCH721
                MOV A,BUFM
                POP BUFM
                MOV BUFM,A
                SJMP STTUCONT1
STTU_NOCH721:   POP BUFM
STTUCONT1:      LJMP STTUNO72
STTUNOCY1M:     MOV R6,A
                MOV A,R5
                ANL A,#11110000B
                ORL A,R6
                PUSH BUFM
                MOV BUFM,A
                LCALL STT_COMP
                MOV A,CHAR
                CJNE A,#1,STTU_NOCH722
                MOV A,BUFM
                POP BUFM
                MOV BUFM,A
                SJMP STTUNO72
STTU_NOCH722:   POP BUFM
STTUNO72:       CJNE R4,#75,STTUNO75
                MOV A,BUFS
                MOV R5,BUFS
                ANL A,#00001111B
                CLR CY
                SUBB A,#1
                JNC STTUNOCY1S
                MOV A,#00001001B
                MOV R6,A
                MOV A,R5
                ANL A,#11110000B
                SWAP A
                CLR CY
                SUBB A,#1
                JNC STTUNOCY2S
                MOV A,#5
STTUNOCY2S:     SWAP A
                ORL A,R6
                PUSH BUFS
                MOV BUFS,A
                LCALL STT_COMP
                MOV A,CHAR
                CJNE A,#1,STTU_NOCH751
                MOV A,BUFS
                POP BUFS
                MOV BUFS,A
                SJMP STTUCONT2
STTU_NOCH751:   POP BUFS
STTUCONT2:      LJMP STTUNO75
STTUNOCY1S:     MOV R6,A
                MOV A,R5
                ANL A,#11110000B
                ORL A,R6
                PUSH BUFS
                MOV BUFS,A
                LCALL STT_COMP
                MOV A,CHAR
                CJNE A,#1,STTU_NOCH752
                MOV A,BUFS
                POP BUFS
                MOV BUFS,A
                SJMP STTUNO75
STTU_NOCH752:   POP BUFS
STTUNO75:       LCALL TIMET_DISPL_U
                LCALL KEYDELAY

                LJMP STTLOOP1
```

```
STTDOWN:      CJNE R4,#72,STTDNO72
              MOV A,BUFM
              ADD A,#1
              DA A
              CJNE A,#01100000B,STTDNO60
              MOV A,#0
STTDNO60:     PUSH BUFM
              MOV BUFM,A
              LCALL STT_COMP
              MOV A,CHAR
              CJNE A,#1,STTD_NOCH72
              MOV A,BUFM
              POP BUFM
              MOV BUFM,A
              SJMP STTDNO72
STTD_NOCH72:  POP BUFM
STTDNO72:     CJNE R4,#75,STTDNO75
              MOV A,BUFS
              ADD A,#1
              DA A
              CJNE A,#01100000B,STTDNO60S
              MOV A,#0
STTDNO60S:    PUSH BUFS
              MOV BUFS,A
              LCALL STT_COMP
              MOV A,CHAR
              CJNE A,#1,STTD_NOCH75
              MOV A,BUFS
              POP BUFS
              MOV BUFS,A
              SJMP STTDNO75
STTD_NOCH75:  POP BUFS
STTDNO75:     LCALL TIMET_DISPL_U
              LCALL KEYDELAY

              LJMP STTLOOP1

STT_UPDATE:       LCALL STT_LBREC         ;zapisanie ilości bajtów rekordu
                  MOV TRM,BDPH
                  MOV TRS,BDPL
                  MOV BDPH,#00H
                  MOV BDPL,#42H
                  MOV R3,#8               ;LBL1(NREC)
                  LCALL SRT_LICZ
                  MOV DPH,BDPH
                  MOV DPL,BDPL
                  MOV A,TRS
                  MOVX @DPTR,A
                  MOV BDPH,#00H
                  MOV BDPL,#42H
                  MOV R3,#7               ;LBH1(NREC)
                  LCALL SRT_LICZ
                  MOV DPH,BDPH
                  MOV DPL,BDPL
                  MOV A,TRM
                  MOVX @DPTR,A

                  MOV A,NREC
                  CJNE A,#50,STTUPD_WR
                  RET
STTUPD_WR:        MOV BDPH,#00H           ;zapisanie adresu początku następnego
                  MOV BDPL,#42H           ;rekordu
                  MOV R3,#6               ;DPL1(NREC)
                  LCALL SRT_LICZ
                  MOV DPH,BDPH
                  MOV DPL,BDPL
                  MOVX A,@DPTR
                  MOV R4,A
                  MOV BDPH,#00H
                  MOV BDPL,#42H
                  MOV R3,#5               ;DPH1(NREC)
                  LCALL SRT_LICZ
```

```
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV R5,A
                CLR C
                MOV A,R4
                ADD A,TRS
                MOV R4,A
                MOV A,R5
                ADDC A,TRM
                MOV R5,A

                INC NREC
                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#6               ;DPL1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOV A,R4
                MOVX @DPTR,A
                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#5               ;DPH1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOV A,R5
                MOVX @DPTR,A
                DEC NREC

                RET
;-------------------------------------
SRT_LICZ:       MOV A,NREC              ;BDPH.BDPL<-BDPH.BDPL+9*(NREC-1)+R3
                DEC A
                MOV B,#9
                MUL AB
                CLR C
                ADD A,BDPL
                MOV BDPL,A
                MOV A,B
                ADDC A,BDPH
                MOV BDPH,A
                CLR C
                MOV A,BDPL
                ADD A,R3
                MOV BDPL,A
                MOV A,BDPH
                ADDC A,#0
                MOV BDPH,A
                RET
;-------------------------------------
; Sprawdza zakres dostępnego do zapisu czasu,
; wynik: CHAR=1 => czas zapisu w dostępnym zakresie, CHAR=0 zakres przekroczony
; wejście: BUFM,BUFS w BCD
; ENDRAM(32K)=7FFFH
; wyjście: (ENDRAM-DPH1.DPL1) >= (M*60+S)*Fp => CHAR=1 else CHAR=0

STT_COMP:       MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#6               ;DPL1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV TRS,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#5               ;DPH1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
```

```
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV TRM,A

                MOV A,#0FFH
                CLR C
                SUBB A,TRS
                MOV TRS,A

                MOV A,#07FH             ;32kB
                SUBB A,TRM
                MOV TRM,A

                LCALL STT_LBREC

                CLR C                   ;porównanie
                MOV A,TRS
                SUBB A,BDPL
                MOV A,TRM
                SUBB A,BDPH
                MOV A,#0
                SUBB A,TRH              ;jeżeli CY=1 to zakres przekroczony

                JNC STTC_NOCY
                MOV CHAR,#0
                RET
STTC_NOCY:      MOV CHAR,#1
                RET
;--------------------------------------
; Liczba bajtów zapisu: TRH.BDPH.BDPL<-(BUFM*60+BUFS)*Fp
; BUFM,BUFS w kodzie BCD

STT_LBREC:      MOV CHAR,BUFS
                LCALL CONV_BCD2BIN
                MOV R1,CHAR
                MOV CHAR,BUFM
                LCALL CONV_BCD2BIN
                MOV A,CHAR
                MOV B,#60
                MUL AB
                CLR C
                ADD A,R1
                MOV BDPL,A
                MOV A,B
                ADDC A,#0       ;dodanie znacznika CY
                MOV BDPH,A

                MOV DPH,#00H
                MOV DPL,#41H
                MOVX A,@DPTR    ;Fp
                MOV R1,A

                MOV A,BDPL
                MOV B,R1
                MUL AB
                MOV BDPL,A

                MOV R2,B
                MOV A,BDPH
                MOV B,R1
                MUL AB
                ADD A,R2
                MOV BDPH,A
                MOV A,B
                ADDC A,#0
                MOV TRH,A

                RET
;--------------------------------------
SRTTEXT1:       DB ' zako',4,'cz [dodaj]',8
SRTTEXT2:       DB '[zako',4,'cz] dodaj ',8
```

```
SRT_WDODAJ:     LCALL SRTU_ZERUJN
                LCALL LCD_CLEAR
                LCALL LCD_CAH

                MOV ADR,#64
                LCALL LCD_DDRAM
                MOV DPTR,#SRTTEXT1
                LCALL LCD_WRS
                MOV ADR,#0
                LCALL LCD_DDRAM
                MOV A,NREC
                MOV B,#10
                DIV AB
                MOV R3,A
                MOV CHAR,A
                LCALL CONV_BCDL
                LCALL LCD_WR
                MOV ADR,#1
                LCALL LCD_DDRAM
                MOV CHAR,B
                LCALL CONV_BCDL
                LCALL LCD_WR
                MOV R3,#1

SRTWD_KEYLOOP:  LCALL TEST_KEY
                MOV A,KEY
                CJNE A,#KEY_LEFT,SRTWD_NOLEFT
                MOV ADR,#64
                LCALL LCD_DDRAM
                MOV DPTR,#SRTTEXT2
                LCALL LCD_WRS
                MOV R3,#2
                LCALL KEYDELAY
                SJMP SRTWD_KEYLOOP
SRTWD_NOLEFT:   CJNE A,#KEY_RIGHT,SRTWD_NORIGHT
                MOV ADR,#64
                LCALL LCD_DDRAM
                MOV DPTR,#SRTTEXT1
                LCALL LCD_WRS
                MOV R3,#1
                LCALL KEYDELAY
                SJMP SRTWD_KEYLOOP
SRTWD_NORIGHT:  CJNE A,#KEY_ENTER,SRTWD_NOENTER
                CJNE R3,#1,SRTWD_NO1
                MOV DPTR,#0040H
                MOVX A,@DPTR
                CJNE A,#50,SRTWD_NO50           ;maksymalnie 50 rekordów
                MOV L,#1
                LCALL BEEP
                MOV R3,#1
SRTWD_LOOPNP1:  LCALL TEST_KEY
                MOV A,KEY
                CJNE A,#KEY_NOPRESS,SRTWD_LOOPNP1
                LJMP SRTWD_KEYLOOP
SRTWD_NO50:     INC A
                MOVX @DPTR,A
                LCALL SRT_DODAJ
                LCALL SRT_WREKORD
                LCALL KEYDELAY
                LJMP SRT_KEYLOOP
SRTWD_NO1:      CJNE R3,#2,SRTWD_NOENTER
                LJMP PMENU1_1
SRTWD_NOENTER:  CJNE A,#KEY_UP,SRTWD_NOUP
                MOV A,#1
                CJNE A,NREC,SRTWD_NOBEG
                SJMP SRTWD_KEYLOOP
SRTWD_NOBEG:    DEC NREC
                LJMP SRT_TESTNREC
SRTWD_NOUP:     SJMP SRTWD_KEYLOOP
;-------------------------------------
SRTUTEXT1:      DB ' Usun',0,1,' zapis ? ',8
SRTUTXTTAK:     DB '    nie [tak]   ',8
```

```
SRTUTXTNIE:     DB '   [nie] tak    ',8

SRT_USUN:       LCALL LCD_CLEAR
                LCALL LCD_CAH
                MOV DPTR,#SRTUTEXT1
                LCALL LCD_WRS
                MOV ADR,#64
                LCALL LCD_DDRAM
                MOV DPTR,#SRTUTXTNIE
                LCALL LCD_WRS
                MOV R3,#1

SRTU_KEYLOOP:   LCALL TEST_KEY
                MOV A,KEY
                CJNE A,#KEY_LEFT,SRTU_NOLEFT
                MOV ADR,#64
                LCALL LCD_DDRAM
                MOV DPTR,#SRTUTXTNIE
                LCALL LCD_WRS
                MOV R3,#1
                LCALL KEYDELAY
                SJMP SRTU_KEYLOOP
SRTU_NOLEFT:    CJNE A,#KEY_RIGHT,SRTU_NORIGHT
                MOV ADR,#64
                LCALL LCD_DDRAM
                MOV DPTR,#SRTUTXTTAK
                LCALL LCD_WRS
                MOV R3,#2
                LCALL KEYDELAY
                SJMP SRTU_KEYLOOP
SRTU_NORIGHT:   CJNE A,#KEY_ENTER,SRTU_NOENTER
                CJNE R3,#1,SRTU_NO1
                LCALL SRT_WREKORD
                LJMP SRT_KEYLOOP
SRTU_NO1:       CJNE R3,#2,SRTU_NOENTER
                MOV A,#1
                CJNE A,NREC,SRTU_NOBEG
                SJMP SRTU_KEYLOOP
SRTU_NOBEG:     LCALL SRTU_ZERUJ

                DEC NREC
                LCALL SRT_WREKORD
                LJMP SRT_KEYLOOP
SRTU_NOENTER:   LJMP SRTU_KEYLOOP
;---------------------------------------
SRTU_ZERUJ:     MOV DPTR,#0040H
                MOVX A,@DPTR
                DEC A
                MOVX @DPTR,A

SRTU_ZERUJN:    MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#0
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOV A,#0
                MOVX @DPTR,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#1
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOV A,#0
                MOVX @DPTR,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#2
                LCALL SRT_LICZ
```

```
                    MOV DPH,BDPH
                    MOV DPL,BDPL
                    MOV A,#0
                    MOVX @DPTR,A

                    MOV BDPH,#00H
                    MOV BDPL,#42H
                    MOV R3,#3
                    LCALL SRT_LICZ
                    MOV DPH,BDPH
                    MOV DPL,BDPL
                    MOV A,#0
                    MOVX @DPTR,A

                    MOV BDPH,#00H
                    MOV BDPL,#42H
                    MOV R3,#4
                    LCALL SRT_LICZ
                    MOV DPH,BDPH
                    MOV DPL,BDPL
                    MOV A,#0
                    MOVX @DPTR,A

                    RET
;-------------------------------------
SRT_WREKORD:    LCALL LCD_CLEAR

                    MOV ADR,#12
                    LCALL LCD_DDRAM
                    MOV CHAR,#' '
                    LCALL LCD_WR

                    MOV ADR,#0
                    LCALL LCD_DDRAM
                    MOV A,NREC
                    MOV B,#10
                    DIV AB
                    MOV R3,A
                    MOV CHAR,A
                    LCALL CONV_BCDL
                    LCALL LCD_WR
                    MOV ADR,#1
                    LCALL LCD_DDRAM
                    MOV CHAR,B
                    LCALL CONV_BCDL
                    LCALL LCD_WR

                    MOV BDPH,#00H   ;wyświetlanie czasu rejestracji TRHx TRMx TRSx
                    MOV BDPL,#42H   ;wg. aktualnego numeru NREC
                    MOV R3,#0
                    LCALL SRT_LICZ
                    MOV DPH,BDPH
                    MOV DPL,BDPL
                    MOVX A,@DPTR
                    MOV BUFH,A

                    MOV BDPH,#00H
                    MOV BDPL,#42H
                    MOV R3,#1
                    LCALL SRT_LICZ
                    MOV DPH,BDPH
                    MOV DPL,BDPL
                    MOVX A,@DPTR
                    MOV BUFM,A

                    MOV BDPH,#00H
                    MOV BDPL,#42H
                    MOV R3,#2
                    LCALL SRT_LICZ
                    MOV DPH,BDPH
                    MOV DPL,BDPL
                    MOVX A,@DPTR
```

```
                MOV BUFS,A

                LCALL TIMEB_DISPL

                MOV BDPH,#00H    ;wyświetlanie czasu trwania rej. TTMx TTSx
                MOV BDPL,#42H
                MOV R3,#3
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV BUFM,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#4
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV BUFS,A

                LCALL TIMET_DISPL_D

                LCALL KEYDELAY
                RET
;--------------------------------------
TIMET_DISPL_D:  MOV R5,BUFM
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#64+7
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#64+8
                LCALL LCD_DDRAM
                LCALL LCD_WR

                MOV ADR,#64+9
                LCALL LCD_DDRAM
                MOV CHAR,#':'
                LCALL LCD_WR

                MOV R5,BUFS
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#64+10
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#64+11
                LCALL LCD_DDRAM
                LCALL LCD_WR
                RET
;--------------------------------------
TIMET_DISPL_U:  MOV R5,BUFM
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#7
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#8
                LCALL LCD_DDRAM
                LCALL LCD_WR

                MOV ADR,#9
                LCALL LCD_DDRAM
                MOV CHAR,#':'
```

```
                LCALL LCD_WR

                MOV R5,BUFS
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#10
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#11
                LCALL LCD_DDRAM
                LCALL LCD_WR

                RET
;--------------------------------------
;       Menu: Ustaw->parametry
;--------------------------------------
PMENU1_2:    LCALL LCD_CLEAR
             LCALL LCD_CAH

             MOV DPTR,#PMENU11+1
             LCALL LCD_WRS
             MOV ADR,#6
             LCALL LCD_DDRAM
             MOV DPTR,#PMENU15+2
             LCALL LCD_WRS

             MOV A,NMENU2
             CJNE A,#2,NOM1_22
             MOV DPTR,#PMENU16+2
             LJMP PR1_2
NOM1_22:     MOV DPTR,#PMENU17+2
PR1_2:       LCALL PRINTARR

             LCALL KEYDELAY

KEYLOOP1_2:  LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_UP,NOUP1_2
             LJMP UP1_2
NOUP1_2:     CJNE A,#KEY_DOWN,NODOWN1_2
             LJMP DOWN1_2
NODOWN1_2:   CJNE A,#KEY_LEFT,NOLEFT1_2
             LJMP PMENU1         ;Powrót do PMENU1
NOLEFT1_2:   CJNE A,#KEY_RIGHT,NORIGHT1_2
             LJMP RIGHT1_2
NORIGHT1_2:  MOV A,STATUS
             CJNE A,#'e',NO_EKG1_2
             LJMP MONITOR_EKG
NO_EKG1_2:   CJNE A,#'p',NO_PROGRAM1_2
             LJMP PROGRAM
NO_PROGRAM1_2:  CJNE A,#'s',NO_SEND1_2
                LJMP SEND_DATA
NO_SEND1_2:     LJMP KEYLOOP1_2

UP1_2:       DEC NMENU2
             MOV A,NMENU2
             CJNE A,#2,UP1_22
             MOV DPTR,#PMENU16+2
             LCALL PRINTARR
             LJMP UP1_21
UP1_22:      CJNE A,#1,UP1_21
             INC NMENU2
UP1_21:      LCALL KEYDELAY
             LJMP KEYLOOP1_2

DOWN1_2:     INC NMENU2
             MOV A,NMENU2
             CJNE A,#3,DOWN1_23
             MOV DPTR,#PMENU17+2
             LCALL PRINTARR
```

```
             LJMP DOWN1_24
DOWN1_23:    CJNE A,#4,DOWN1_24
             ;MOV L,#1
             ;LCALL BEEP
             DEC NMENU2
DOWN1_24:    LCALL KEYDELAY
             LJMP KEYLOOP1_2

RIGHT1_2:    MOV A,NMENU2
             CJNE A,#2,NOPMENU1_22     ;->Menu: Ustaw->parametry->próbkowania
             SJMP PARAM_PROB
NOPMENU1_22: CJNE A,#3,NOPMENU1_23     ;->Menu: Ustaw->parametry->transmisji
             LJMP PARAM_TRANS
NOPMENU1_23: LJMP KEYLOOP1_2
;--------------------------------------
;  Menu: Ustaw->parametry->próbkowania
;--------------------------------------
PPROB0:          DB ' Cz',2,'stotliwo',6,1,8
PPROB1:          DB '   [  80 Hz ]   ',8
PPROB2:          DB '   [ 100 Hz ]   ',8
PPROB3:          DB '   [ 120 Hz ]   ',8


PARAM_PROB:      LCALL LCD_CLEAR
                 LCALL LCD_CAH
                 MOV DPTR,#PPROB0
                 LCALL LCD_WRS

                 MOV DPTR,#0041H
                 MOVX A,@DPTR
                 CJNE A,#80,PP_NO80
                 MOV R3,#1
                 SJMP PPSETDISPL
PP_NO80:         CJNE A,#100,PP_NO100
                 MOV R3,#2
                 SJMP PPSETDISPL
PP_NO100:        CJNE A,#120,PPSETDISPL
                 MOV R3,#3

PPSETDISPL:      MOV A,R3
                 CJNE A,#1,PPNO80
                 MOV DPTR,#PPROB1
                 SJMP PPPRINT
PPNO80:          CJNE A,#2,PPNO100
                 MOV DPTR,#PPROB2
                 SJMP PPPRINT
PPNO100:         CJNE A,#3,PPKEYLOOP
                 MOV DPTR,#PPROB3

PPPRINT:         MOV ADR,#64
                 LCALL LCD_DDRAM
                 LCALL LCD_WRS
                 LCALL KEYDELAY

PPKEYLOOP:       LCALL TEST_KEY
                 MOV A,KEY
                 CJNE A,#KEY_UP,PP_NOUP
                 MOV A,R3
                 CJNE A,#3,PPINC
                 SJMP PP_NOUP
PPINC:           INC R3
                 SJMP PPSETDISPL
PP_NOUP:         CJNE A,#KEY_DOWN,PP_NODOWN
                 MOV A,R3
                 CJNE A,#1,PPDEC
                 SJMP PP_NODOWN
PPDEC:           DEC R3
                 SJMP PPSETDISPL
PP_NODOWN:       CJNE A,#KEY_ENTER,PP_NOENTER
                 LJMP PP_ENTER
PP_NOENTER:      SJMP PPKEYLOOP
```

```
PP_ENTER:       MOV DPTR,#0041H
                MOV A,R3
                CJNE A,#1,PP_NO1
                MOV A,#80
                MOVX @DPTR,A
                SJMP PP_NO3
PP_NO1:         CJNE A,#2,PP_NO2
                MOV A,#100
                MOVX @DPTR,A
                SJMP PP_NO3
PP_NO2:         CJNE A,#3,PP_NO3
                MOV A,#120
                MOVX @DPTR,A
PP_NO3:         MOV DPTR,#0040H           ;ilość rekordów = 0
                MOV A,#0
                MOVX @DPTR,A
                LJMP PMENU1_2
;-------------------------------------
;  Menu: Ustaw->parametry->transmisji
;-------------------------------------
PTRAN0:         DB ' Szybko',6,1,' portu',8
PTRAN1:         DB '   [  4800  ]   ',8
PTRAN2:         DB '   [  9600  ]   ',8
PTRAN3:         DB '   [  19200 ]   ',8

PARAM_TRANS:    LCALL LCD_CLEAR
                LCALL LCD_CAH

                MOV DPTR,#PTRAN0
                LCALL LCD_WRS
PTSETDISPL:     MOV A,RSBAUD
                CJNE A,#1,PTNO4800
                MOV DPTR,#PTRAN1
                SJMP PTPRINT
PTNO4800:       CJNE A,#2,PTNO9600
                MOV DPTR,#PTRAN2
                SJMP PTPRINT
PTNO9600:       CJNE A,#3,PTKEYLOOP
                MOV DPTR,#PTRAN3

PTPRINT:        MOV ADR,#64
                LCALL LCD_DDRAM
                LCALL LCD_WRS

                LCALL KEYDELAY

PTKEYLOOP:      LCALL TEST_KEY
                MOV A,KEY
                CJNE A,#KEY_UP,PT_NOUP
                MOV A,RSBAUD
                CJNE A,#3,PTINC
                SJMP PT_NOUP
PTINC:          INC RSBAUD
                SJMP PTSETDISPL
PT_NOUP:        CJNE A,#KEY_DOWN,PT_NODOWN
                MOV A,RSBAUD
                CJNE A,#1,PTDEC
                SJMP PT_NODOWN
PTDEC:          DEC RSBAUD
                SJMP PTSETDISPL
PT_NODOWN:      CJNE A,#KEY_ENTER,PT_NOENTER
                LJMP PT_ENTER
PT_NOENTER:     SJMP PTKEYLOOP

PT_ENTER:       MOV A,RSBAUD
                CJNE A,#1,PT_NO1
                ANL PCON,#01111111B          ;PCON.7=0 /2
                MOV TH1,#0FAH                ;wartość początkowa T1 - 4800
                SJMP PT_NO3
PT_NO1:         CJNE A,#2,PT_NO2
                ANL PCON,#01111111B          ;PCON.7=0 /2
                MOV TH1,#0FDH                ;wartość początkowa T1 - 9600
```

```
                SJMP PT_NO3
PT_NO2:         CJNE A,#3,PT_NO3
                ORL PCON,#10000000B           ;PCON.7=1
                MOV TH1,#0FDH                 ;wartość początkowa T1 - 19200
PT_NO3:         LJMP PMENU1_2
;--------------------------------------
;           Menu: Informacja
;--------------------------------------
PM4TEXT1:       DB '        Rejestrator holterowski z pami',2,'c',8
                                              ;erowski z pamiec
PM4TEXT2:       DB 'erowski z pami',2,'ci',0,' cyfrow',0,'. CPU 89C52 32',8
PM4TEXT3:       DB 'w',0,'. CPU 89C52,32kB RAM,DAC 12bit.       ',8

PMENUINFO:      LCALL LCD_CLEAR         ;Menu: Informacja

KEYBW41:        LCALL LCD_CAH
                MOV DPTR,#PM4TEXT1
                LCALL LCD_WRS

                MOV R4,#24
KWR4LOOP1:      LCALL LCD_DSL
                LCALL KEYDELAY
                LCALL TEST_KEY
                MOV A,KEY
                CJNE A,#KEY_ENTER,KWR4NOENTER1
                LJMP PMENU0 ;?
KWR4NOENTER1:   DJNZ R4,KWR4LOOP1

                LCALL LCD_CAH
                MOV DPTR,#PM4TEXT2
                LCALL LCD_WRS

                MOV R4,#24
KWR4LOOP2:      LCALL LCD_DSL
                LCALL KEYDELAY
                LCALL TEST_KEY
                MOV A,KEY
                CJNE A,#KEY_ENTER,KWR4NOENTER2
                LJMP PMENU0
KWR4NOENTER2:   DJNZ R4,KWR4LOOP2

                LCALL LCD_CAH
                MOV DPTR,#PM4TEXT3
                LCALL LCD_WRS

                MOV R4,#24
KWR4LOOP3:      LCALL LCD_DSL
                LCALL KEYDELAY
                LCALL TEST_KEY
                MOV A,KEY
                CJNE A,#KEY_ENTER,KWR4NOENTER3
                LJMP PMENU0
KWR4NOENTER3:   DJNZ R4,KWR4LOOP3
                SJMP KEYBW41
;**************************************
;*          Procedury systemowe       *
;**************************************
; A,B,DPTR,R0,R1,R2 - zbiór rejestrów 0

DELAY1:       MOV R0,L                  ;PETLA OPOZNIAJACA L*1 ms
LOOP3:        MOV R1,#2                 ;MOV L,#XX
LOOP2:        MOV R2,#250               ;LCALL DELAY1
LOOP1:        DJNZ R2,LOOP1
              DJNZ R1,LOOP2
              DJNZ R0,LOOP3
              RET

DELAY100:     MOV R0,L                  ;PETLA OPOZNIAJACA L*100 ms
LOOP6:        MOV R1,#194               ;MOV L,#XX
LOOP5:        MOV R2,#0FFH              ;LCALL DELAY100
LOOP4:        DJNZ R2,LOOP4
              DJNZ R1,LOOP5
```

```
            DJNZ R0,LOOP6
            RET

LCD_INI:    MOV L,#15
            LCALL DELAY1              ;WAIT 15 MS
            SETB CHIP                 ;blokada pamięci RAM
            CLR RAM
            CLR RS                    ;RS =0  P1.0
            CLR RW                    ;R/W=0  P1.1
            SETB CS                   ;CS=1  P1.2 odblokowanie wyświetlacza
            MOV A,#00110000B
            MOVX @R0,A
            MOV L,#5                  ;5 MS
            LCALL DELAY1
            MOV A,#00110000B
            MOVX @R0,A
            MOV L,#1                  ;1 MS
            LCALL DELAY1
            MOV A,#00110000B
            MOVX @R0,A

            MOV DISPL,#FS_2L
            LCALL LCD_DISPL           ;Function Set
            MOV DISPL,#DC_DISPLON
            LCALL LCD_DISPL           ;Display ON
            MOV DISPL,#EMS_NOSHIFT
            LCALL LCD_DISPL           ;Entry Mode Set
            LCALL LCD_CAH             ;Cursor At Home
            LCALL LCD_CLEAR           ;Clear Display
            CLR CS                    ;blokada wyświetlacza
            SETB RAM                  ;odblokowanie pamięci RAM
            RET
LCD_WR:     LCALL WAIT_BF             ;WRITE CHAR
            SETB CHIP
            CLR RAM
            SETB RS
            CLR RW
            SETB CS
            MOV A,CHAR
            MOVX @R0,A
            CLR CS
            SETB RAM
            RET
LCD_RD:     LCALL WAIT_BF           ;READ CHAR
            SETB CHIP
            CLR RAM
            SETB RS
            SETB RW
            SETB CS
            MOVX A,@R0
            MOV CHAR,A
            CLR CS
            SETB RAM
            RET
LCD_CSR:    LCALL WAIT_BF           ;CURSOR SHIFT RIGHT
            SETB CHIP
            CLR RAM
            CLR RS
            CLR RW
            SETB CS
            MOV A,#00010100B
            MOVX @R0,A
            CLR CS
            SETB RAM
            RET
LCD_CSL:    LCALL WAIT_BF           ;CURSOR SHIFT LEFT
            SETB CHIP
            CLR RAM
            CLR RS
            CLR RW
            SETB CS
            MOV A,#00010000B
```

```
            MOVX @R0,A
            CLR CS
            SETB RAM
            RET
LCD_CLEAR:  LCALL WAIT_BF
            SETB CHIP
            CLR RAM
            CLR RS
            CLR RW
            SETB CS
            MOV A,#00000001B        ;CLEAR DISPLAY
            MOVX @R0,A
            CLR CS
            SETB RAM
            RET
LCD_WRS:    MOV A,#0
            MOVC A,@A+DPTR
            MOV CHAR,A
            CLR C
            SUBB A,#8
            JZ KONIECWRS
            LCALL LCD_WR
            INC DPTR
            SJMP LCD_WRS
KONIECWRS:  RET
LCD_DDRAM:  LCALL WAIT_BF
            SETB CHIP
            CLR RAM
            CLR RS
            CLR RW
            SETB CS
            MOV A,ADR
            ADD A,#10000000B
            MOVX @R0,A
            CLR CS
            SETB RAM
            RET
LCD_CGRAM:  LCALL WAIT_BF
            SETB CHIP
            CLR RAM
            CLR RS
            CLR RW
            SETB CS
            MOV A,ADR
            ADD A,#01000000B
            MOVX @R0,A
            CLR CS
            SETB RAM
            RET
LCD_SETCHAR: LCALL LCD_CGRAM
            MOV A,#0
            MOVC A,@A+DPTR
            JZ ENDSETCHAR
            MOV CHAR,A
            LCALL LCD_WR
            INC DPTR
            INC ADR
            LJMP LCD_SETCHAR
ENDSETCHAR:
            MOV ADR,#00H
            LCALL LCD_DDRAM
            RET
WAIT_BF:    MOV L,#1
            LCALL DELAY1
            RET

LCD_DISPL:  LCALL WAIT_BF           ;Display set
            SETB CHIP
            CLR RAM
            CLR RS
            CLR RW
            SETB CS
```

```
             MOV A,DISPL
             MOVX @R0,A
             CLR CS
             SETB RAM
             RET
LCD_CAH:     LCALL WAIT_BF              ;Cursor at home
             SETB CHIP
             CLR RAM
             CLR RS
             CLR RW
             SETB CS
             MOV A,#00000010B
             MOVX @R0,A
             CLR CS
             SETB RAM
             RET
LCD_CON:     MOV DISPL,#DC_CURON        ;włączenie kursora
             LCALL LCD_DISPL
             RET
LCD_COFF:    MOV DISPL,#DC_CUROFF       ;wyłączenie kursora
             LCALL LCD_DISPL
             RET
LCD_DSR:     LCALL WAIT_BF              ;Display shift right
             SETB CHIP
             CLR RAM
             CLR RS
             CLR RW
             SETB CS
             MOV A,#00011100B
             MOVX @R0,A
             CLR CS
             SETB RAM
             RET
LCD_DSL:     LCALL WAIT_BF              ;Display shift left
             SETB CHIP
             CLR RAM
             CLR RS
             CLR RW
             SETB CS
             MOV A,#00011000B
             MOVX @R0,A
             CLR CS
             SETB RAM
             RET

;Tablice wzorców znaków:
CHAR_PL:     DB 100000000B    ;0 a
             DB 10000000B
             DB 10001110B
             DB 10000001B
             DB 10001111B
             DB 10010001B
             DB 10001111B
             DB 10000001B

             DB 10000010B    ;1 c
             DB 10000100B
             DB 10001110B
             DB 10010000B
             DB 10010000B
             DB 10010001B
             DB 10001110B
             DB 10000000B

             DB 10000000B    ;2 e
             DB 10000000B
             DB 10001110B
             DB 10010001B
             DB 10011111B
             DB 10010000B
             DB 10001111B
             DB 10000001B
```

```
                DB 10001100B    ;3 l
                DB 10000100B
                DB 10000110B
                DB 10000100B
                DB 10001100B
                DB 10000100B
                DB 10001110B
                DB 10000000B

                DB 10000010B   ;4 n
                DB 10000100B
                DB 10010110B
                DB 10011001B
                DB 10010001B
                DB 10010001B
                DB 10010001B
                DB 10000000B

                DB 10000010B   ;5 o
                DB 10000100B
                DB 10001110B
                DB 10010001B
                DB 10010001B
                DB 10010001B
                DB 10001110B
                DB 10000000B

                DB 10000010B   ;6 s
                DB 10000100B
                DB 10001110B
                DB 10010000B
                DB 10001110B
                DB 10000001B
                DB 10011110B
                DB 10000000B

                DB 10000100B   ;7
                DB 10000000B
                DB 10011111B
                DB 10000010B
                DB 10000100B
                DB 10001000B
                DB 10011111B
                DB 10000000B

                DB 0

CONV_BCDL:      MOV A,CHAR                ;MOV CHAR,#XX, wynik w CHAR
                ANL A,#00001111B
                ORL A,#00110000B
                MOV CHAR,A
                RET
CONV_BCDH:      MOV A,CHAR
                SWAP A
                ANL A,#00001111B
                ORL A,#00110000B
                MOV CHAR,A
                RET

CONV_BCD2BIN:  MOV A,CHAR                 ;MOV CHAR,#XX (BCD), wynik w CHAR (BIN)
                ANL A,#00001111B
                MOV R0,A
                MOV A,CHAR
                ANL A,#11110000B
                SWAP A
                MOV B,#10
                MUL AB
                ADD A,R0
                MOV CHAR,A
                RET
```

```
CHAR_SPEC:   DB 10011111B    ;0 left
             DB 10011111B
             DB 10000000B
             DB 10000000B
             DB 10000000B
             DB 10000000B
             DB 10000000B
             DB 10000000B

             DB 10011111B    ;1 right
             DB 10011111B
             DB 10000000B
             DB 10000000B
             DB 10000000B
             DB 10000000B
             DB 10000000B
             DB 10000000B
             DB 10000000B

             DB 10000001B    ;2
             DB 10000001B
             DB 10000010B
             DB 11000010B
             DB 10100100B
             DB 10010100B
             DB 10001000B
             DB 10000000B

             DB 0

DISPLAY_HEX: MOV R1,#2
             MOV R0,CHAR
             LCALL CONV_BCDH
             SJMP DH_TEST1
DH_TEST2:    MOV CHAR,R0
             LCALL CONV_BCDL
DH_TEST1:    MOV A,CHAR
             CJNE A,#00111010B,DH1_NOA
             MOV CHAR,#'A'
DH1_NOA:     CJNE A,#00111011B,DH1_NOB
             MOV CHAR,#'B'
DH1_NOB:     CJNE A,#00111100B,DH1_NOC
             MOV CHAR,#'C'
DH1_NOC:     CJNE A,#00111101B,DH1_NOD
             MOV CHAR,#'D'
DH1_NOD:     CJNE A,#00111110B,DH1_NOE
             MOV CHAR,#'E'
DH1_NOE:     CJNE A,#00111111B,DH1_NOF
             MOV CHAR,#'F'
DH1_NOF:     PUSH 00H
             PUSH 01H
             LCALL LCD_WR
             POP 01H
             POP 00H
             DJNZ R1,DH_TEST2
             RET
;*************************************
;*    PODPROGRAM INICJACJI SYSTEMU     *
;*************************************
INIT1:       DB 'AT89C52 32kB RAM',8
INIT2:       DB '11.059 MHz RS232',8

STARTUP:
             SETB RAM
             SETB CHIP                ;blokada pamięci RAM
             CLR CS                   ;CS=0 [P1.2] wyświetlacz nieaktywny

             CLR RS_T_FLAG       ;kasowanie bitów transmisji znaków
             CLR RS_T_FLAG
             CLR MBREAK               ;kasowanie flagi zezwolenia na kom.
             CLR RECORD_FLAG

             CLR AC_PWRD              ;ustawienie bitów obsługujących
```

```
SETB AC_BUSY              ;przetwornik
SETB AC_CS
SETB AC_RC
SETB AC_DATA
CLR AC_DATACLK

MOV TSH,#0                ;zerowanie buforów zegara
MOV TSM,#0
MOV TSS,#0
MOV TRH,#0
MOV TRM,#0
MOV TRS,#0

MOV STATUS,#'m'           ;włączenie trybu MENU

MOV NMENU0,#2
MOV NMENU1,#2
MOV NMENU2,#2

MOV NREC,#0

MOV TMOD,#00100001B       ;T0: M0=1 M1=0,tryb 1-licznik 16bit
                          ;T1: M0=0 M1=1,tryb 2-licznik 8bit
                          ;z automat. wpisywaniem wart. pocz.
MOV T2CON,#10000100B      ;T2: tryb 1
MOV RLDH,#00H             ;wartość początkowa T2
MOV RLDL,#00H
SETB IE.5                 ;ET2=1 od maskowanie przerwań z T2

MOV 01FH,#14              ;R7 zbiór rej. 3

MOV TL0,#000H
MOV TH0,#000H
SETB ET0                  ;od maskowanie przerwań z T0
SETB TR0                  ;włączenie licznika T0
SETB IP.1                 ;PT0=1 T0 na wyższy poziom

                          ;ustawienie param. portu szer. (T1)
MOV SCON,#01010000B       ;tryb 1,REN=1 - uaktywnienie odbioru
ORL PCON,#10000000B       ;PCON.7=1 19200
MOV TL1,#0FDH
MOV TH1,#0FDH      ;wartość początkowa T1 ( 11.059 MHz )
CLR TF1                   ;TCON.7=0 - znacznik przepełnienia T1
SETB TR1                  ;włączenie licznika T1
SETB ES                   ;od maskowanie przerwań portu RS

SETB EA                   ;EA=1 - globalne zezwolenie na przerw.

MOV RSBAUD,#3             ;RS standardowo 19200

LCALL SET_HEADER          ;zerowanie nagłówka pliku rejestracji

SETB REC_FIRSTPR

LCALL LCD_INI             ;inicjalizacja wyświetlacza

LCALL LCD_CLEAR           ;kasowanie wyświetlacza
LCALL LCD_CAH

MOV ADR,#00H              ;adres początku w CGRAM (00H)
MOV DPTR,#CHAR_PL         ;początek tablicy w pamięci programu
LCALL LCD_SETCHAR         ;ustawienie wzorców

LCALL LCD_COFF            ;wyłączenie kursora

MOV DPTR,#INIT1           ;wydruk komunikatów
LCALL LCD_WRS             ;systemowych
MOV ADR,#40
LCALL LCD_DDRAM
MOV DPTR,#INIT2
LCALL LCD_WRS
```

```
            MOV L,#20
            LCALL DELAY100

            MOV L,#1
            LCALL BEEP                ;sygnał dźwiękowy

            SETB AC_PWRD              ;przetwornik w stanie obniżonego
                                      ;poboru mocy
            RET
;--------------------------------------
; Zerowanie nagłówka pliku rejestracji
;--------------------------------------
HEADER:         DB 'RECEKG
v.1.0brak',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,'brak',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
'1',0,0,0,0,0,0,0,0,0,0,0,00H,64H,0,0,0,0,0,02H,08H

SET_HEADER: MOV BDPH,#0
            MOV BDPL,#0
            MOV R3,#73
            MOV R4,#0
SH_LOOP1:   MOV DPTR,#HEADER
            MOV A,R4
            MOVC A,@A+DPTR
            INC R4
            MOV DPH,BDPH
            MOV DPL,BDPL
            MOVX @DPTR,A
            INC DPTR
            MOV BDPH,DPH
            MOV BDPL,DPL
            DJNZ R3,SH_LOOP1

            MOV DPTR,#0049H
            MOV R3,#2
SH_LOOP2:   MOV R4,#0DEH
SH_LOOP3:   MOV A,#0
            MOVX @DPTR,A
            INC DPTR
            DJNZ R4,SH_LOOP3
            DJNZ R3,SH_LOOP2

            MOV DPTR,#0204H
            MOV A,#'D'
            MOVX @DPTR,A
            INC DPTR
            MOV A,#'A'
            MOVX @DPTR,A
            INC DPTR
            MOV A,#'T'
            MOVX @DPTR,A
            INC DPTR
            MOV A,#'A'
            MOVX @DPTR,A
            RET
;**************************************
;*    PODPROGRAM OBSŁUGI KLAWIATURY    *
;**************************************

TEST_KEY:   SETB KEYIN                ;P1.7 jako wejście, w KEY wynik
            LCALL SETKEYS

KEYL1:      CLR KEY1
            JNB KEYIN,PRESS1
            LCALL SETKEYS
KEYL2:      CLR KEY2
            JNB KEYIN,PRESS2
            LCALL SETKEYS
KEYL3:      CLR KEY3
            JNB KEYIN,PRESS3
            LCALL SETKEYS
KEYL4:      CLR KEY4
            JNB KEYIN,PRESS4
```

```
             LCALL SETKEYS
KEYL5:       CLR KEY5
             JNB KEYIN,PRESS5
             LCALL SETKEYS

             MOV KEY,#KEY_NOPRESS
             RET

PRESS1:      MOV KEY,#KEY_LEFT
             LCALL SETKEYS
             RET
PRESS2:      MOV KEY,#KEY_DOWN
             LCALL SETKEYS
             RET
PRESS3:      MOV KEY,#KEY_UP
             LCALL SETKEYS
             RET
PRESS4:      MOV KEY,#KEY_RIGHT
             LCALL SETKEYS
             RET
PRESS5:      MOV KEY,#KEY_ENTER
             LCALL SETKEYS
             RET

SETKEYS:     SETB KEY1                ;ustawienia początkowe bitów klawiatury
             SETB KEY2
             SETB KEY3
             SETB KEY4
             SETB KEY5
             RET
;*************************************
;*     PDPROGRAM GENERACJI DŹWIĘKU    *
;*************************************
; R0,R1,R2,R3 - zbiór rejestrów 0

BEEP:        MOV R2,L
BLOOPR2:     MOV R3,#100
BLOOPR3:     SETB P3.3
             LCALL BUZDEL
             CLR P3.3
             LCALL BUZDEL
             DJNZ R3,BLOOPR3
             DJNZ R2,BLOOPR2
             RET
BUZDEL:      MOV R0,#2
BLOOPR1:     MOV R1,#130
BLOOPR0:     DJNZ R1,BLOOPR0
             DJNZ R0,BLOOPR1
             RET
;*************************************
;*     PODPROGRAM USTAWIANIA CZASU    *
;*************************************
; Menu Ustaw->czas->(rejestracji,zegara)
; A,R4,R5,R6

SET_TIME:    MOV ADR,#00H             ;adres początku w CGRAM (00H)
             MOV DPTR,#CHAR_SPEC      ;początek tablicy w pamięci programu
             LCALL LCD_SETCHAR        ;ustawienie wzorców

             LCALL LCD_CLEAR
             LCALL LCD_CAH

             MOV A,STPAR
             CJNE A,#1,STNOR1
             MOV BUFS,TSS
             MOV BUFM,TSM
             MOV BUFH,TSH
STNOR1:      CJNE A,#2,STNOR2
             MOV BUFS,TRS
             MOV BUFM,TRM
             MOV BUFH,TRH
```

```
STNOR2:      MOV R4,#68                ;początkowa pozycja kursora

             LCALL TIMEB_DISPL

             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#00000000B
             LCALL LCD_WR

             INC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#00000000B
             LCALL LCD_WR

             LCALL KEYDELAY

ST_LOOPNP1:  LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_NOPRESS,ST_LOOPNP1

STLOOP1:     LCALL TEST_KEY
             MOV A,KEY
             CJNE A,#KEY_ENTER,STNOENTER
             MOV A,STPAR
             CJNE A,#1,STNOW1
             MOV TSS,BUFS
             MOV TSM,BUFM
             MOV TSH,BUFH
STNOW1:      CJNE A,#2,STNOW2
             MOV TRS,BUFS
             MOV TRM,BUFM
             MOV TRH,BUFH
STNOW2:      LCALL LCD_CLEAR
             LCALL LCD_CAH
             MOV ADR,#00H
             MOV DPTR,#CHAR_PL
             LCALL LCD_SETCHAR         ;ustawienie poprzednich wzorców

             RET
STNOENTER:   CJNE A,#KEY_RIGHT,STNORIGHT
             LJMP STRIGHT
STNORIGHT:   CJNE A,#KEY_LEFT,STNOLEFT
             LJMP STLEFT
STNOLEFT:    CJNE A,#KEY_UP,STNOUP
             LJMP STDOWN
STNOUP:      CJNE A,#KEY_DOWN,STNODOWN
             LJMP STUP
STNODOWN:    LJMP STLOOP1

STRIGHT:     CJNE R4,#75,STSHIFTR
             LJMP STLOOP1
STSHIFTR:    DEC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#' '
             LCALL LCD_WR
             INC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#' '
             LCALL LCD_WR
             INC R4
             INC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#0
             LCALL LCD_WR
             INC R4
             MOV ADR,R4
             LCALL LCD_DDRAM
             MOV CHAR,#0
```

```
                LCALL LCD_WR

                LCALL KEYDELAY
                LJMP STLOOP1

STLEFT:         CJNE R4,#69,STSHIFTL
                LJMP STLOOP1
STSHIFTL:       MOV ADR,R4
                LCALL LCD_DDRAM
                MOV CHAR,#' '
                LCALL LCD_WR
                DEC R4
                MOV ADR,R4
                LCALL LCD_DDRAM
                MOV CHAR,#' '
                LCALL LCD_WR
                DEC R4
                DEC R4
                MOV ADR,R4
                LCALL LCD_DDRAM
                MOV CHAR,#0
                LCALL LCD_WR
                DEC R4
                MOV ADR,R4
                LCALL LCD_DDRAM
                MOV CHAR,#0
                LCALL LCD_WR
                INC R4
                LCALL KEYDELAY
                LJMP STLOOP1

STUP:           CJNE R4,#69,STUNO69
                MOV A,BUFH
                MOV R5,BUFH
                ANL A,#00001111B
                CLR CY
                SUBB A,#1
                JNC STUNOCY1H
                MOV A,#00001001B
                MOV R6,A
                MOV A,R5
                ANL A,#11110000B
                SWAP A
                CLR CY
                SUBB A,#1
                JNC STUNOCY2H
                MOV A,#2
                MOV R6,#00000011B
STUNOCY2H:      SWAP A
                ORL A,R6
                MOV BUFH,A
                LJMP STUNO69
STUNOCY1H:      MOV R6,A
                MOV A,R5
                ANL A,#11110000B
                ORL A,R6
                MOV BUFH,A
STUNO69:        CJNE R4,#72,STUNO72
                MOV A,BUFM
                MOV R5,BUFM
                ANL A,#00001111B
                CLR CY
                SUBB A,#1
                JNC STUNOCY1M
                MOV A,#00001001B
                MOV R6,A
                MOV A,R5
                ANL A,#11110000B
                SWAP A
                CLR CY
                SUBB A,#1
                JNC STUNOCY2M
```

```
             MOV A,#5
STUNOCY2M:   SWAP A
             ORL A,R6
             MOV BUFM,A
             LJMP STUNO72
STUNOCY1M:   MOV R6,A
             MOV A,R5
             ANL A,#11110000B
             ORL A,R6
             MOV BUFM,A
STUNO72:     CJNE R4,#75,STUNO75
             MOV A,BUFS
             MOV R5,BUFS
             ANL A,#00001111B
             CLR CY
             SUBB A,#1
             JNC STUNOCY1S
             MOV A,#00001001B
             MOV R6,A
             MOV A,R5
             ANL A,#11110000B
             SWAP A
             CLR CY
             SUBB A,#1
             JNC STUNOCY2S
             MOV A,#5
STUNOCY2S:   SWAP A
             ORL A,R6
             MOV BUFS,A
             LJMP STUNO75
STUNOCY1S:   MOV R6,A
             MOV A,R5
             ANL A,#11110000B
             ORL A,R6
             MOV BUFS,A
STUNO75:     LCALL TIMEB_DISPL
             LCALL KEYDELAY
             LJMP STLOOP1

STDOWN:      CJNE R4,#69,STDNO69
             MOV A,BUFH
             ADD A,#1
             DA A
             CJNE A,#00100100B,STDNO24
             MOV A,#0
STDNO24:     MOV BUFH,A
STDNO69:     CJNE R4,#72,STDNO72
             MOV A,BUFM
             ADD A,#1
             DA A
             CJNE A,#01100000B,STDNO60
             MOV A,#0
STDNO60:     MOV BUFM,A
STDNO72:     CJNE R4,#75,STDNO75
             MOV A,BUFS
             ADD A,#1
             DA A
             CJNE A,#01100000B,STDNO60S
             MOV A,#0
STDNO60S:    MOV BUFS,A
STDNO75:     LCALL TIMEB_DISPL
             LCALL KEYDELAY
             LJMP STLOOP1
;--------------------------------------
;    WYŚWIETLANIE CZASU SYSTEMOWEGO
;--------------------------------------
; R5 - zbiór rejestrów 0

TIMES_DISPL: MOV R5,TSH
             MOV CHAR,R5
             LCALL CONV_BCDH
             MOV ADR,#4
```

```
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#5
                LCALL LCD_DDRAM
                LCALL LCD_WR

                MOV ADR,#6
                LCALL LCD_DDRAM
                MOV CHAR,#':'
                LCALL LCD_WR

                MOV R5,TSM
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#7
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#8
                LCALL LCD_DDRAM
                LCALL LCD_WR

                MOV ADR,#9
                LCALL LCD_DDRAM
                MOV CHAR,#':'
                LCALL LCD_WR

                MOV R5,TSS
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#10
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#11
                LCALL LCD_DDRAM
                LCALL LCD_WR
                RET

TIMEB_DISPL:    MOV R5,BUFH
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#4
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#5
                LCALL LCD_DDRAM
                LCALL LCD_WR

                MOV ADR,#6
                LCALL LCD_DDRAM
                MOV CHAR,#':'
                LCALL LCD_WR

                MOV R5,BUFM
                MOV CHAR,R5
                LCALL CONV_BCDH
                MOV ADR,#7
                LCALL LCD_DDRAM
                LCALL LCD_WR
                MOV CHAR,R5
                LCALL CONV_BCDL
                MOV ADR,#8
                LCALL LCD_DDRAM
                LCALL LCD_WR
```

```
            MOV ADR,#9
            LCALL LCD_DDRAM
            MOV CHAR,#':'
            LCALL LCD_WR

            MOV R5,BUFS
            MOV CHAR,R5
            LCALL CONV_BCDH
            MOV ADR,#10
            LCALL LCD_DDRAM
            LCALL LCD_WR
            MOV CHAR,R5
            LCALL CONV_BCDL
            MOV ADR,#11
            LCALL LCD_DDRAM
            LCALL LCD_WR
            RET
;*************************************
;*  PODPROGRAM OBSŁUGI PRZERWANIA Z T0 *
;*************************************
; zb. rej. 1
; STOS: PSW,ACC,DPH,DPL

AC_UEND:    POP DPL
            POP DPH
            POP ACC
            POP PSW
            RETI
AC_PROBE:   PUSH PSW            ;PSW->STOS

            SETB PSW.3          ;RS0=1 przełączenie na zbiór rej. 1
            CLR PSW.4           ;RS1=0

            PUSH ACC            ;A->  STOS
            PUSH DPH            ;DPH->STOS
               PUSH DPL  ;DPL->STOS

               MOV DPTR,#0041H
               MOVX A,@DPTR
               CJNE A,#80,TU_NO80
               ORL TL0,#000H        ;80Hz
               MOV TH0,#0B8H
               SJMP TU_PROBE
TU_NO80:       CJNE A,#100,TU_NO100 ;100Hz
               ORL TL0,#000H
               MOV TH0,#0DCH
               SJMP TU_PROBE
TU_NO100:      ORL TL0,#000H        ;120Hz
               MOV TH0,#0E2H

TU_PROBE:      MOV A,#'e'
               CJNE A,STATUS,AC_NOE

               LCALL REJ_DACPROBE
               JNB REC_FIRSTPR,TU_SENDB
               CLR REC_FIRSTPR
               SJMP AC_ENDI

TU_SENDB:      MOV SBUF,R5             ;wysyłanie danych monitora EKG
               SJMP AC_ENDI

AC_NOE:        JNB RECORD_FLAG,AC_ENDI
               LCALL REJESTRUJ

AC_ENDI:       POP DPL
               POP DPH
               POP ACC
               POP PSW
               RETI
;---------------------------------------
;         PODPROGRAM REJESTRACJI
;---------------------------------------
```

```
REJESTRUJ:      MOV DPTR,#0040H
                MOVX A,@DPTR
                INC A
                CJNE A,NREC,REJ_NONREND
                CLR RECORD_FLAG
                RET

REJ_NONREND:    JB REC_FIRSTPR,REJ_GETDATA
                CLR C
                MOV A,LBARL
                SUBB A,#1
                MOV LBARL,A
                MOV A,LBARH
                SUBB A,#0
                MOV LBARH,A
                JNC REJ_GETDATA
                LJMP REJ_ENDSET

REJ_GETDATA:    LCALL REJ_DACPROBE      ;pobranie próbki z przetwornika
                JNB REC_FIRSTPR,REJ_WRITE
                CLR REC_FIRSTPR
                RET

REJ_WRITE:      MOV DPH,BADRH           ;zapisanie próbki pod adres BADRH.BADRL
                MOV DPL,BADRL
                MOV A,R5                ;różnica próbek WYNIK
                MOVX @DPTR,A
                INC DPTR
                MOV BADRH,DPH
                MOV BADRL,DPL

                LJMP REJ_END

REJ_ENDSET:     INC NREC

                MOV BDPH,#00H           ;ustawienie nowego czasu początku
                MOV BDPL,#42H           ;rejestracji
                MOV R3,#0               ;TRH1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV TRH,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#1               ;TRM1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV TRM,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#2               ;TRS1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV TRS,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#7               ;LBH1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV LBARH,A

                MOV BDPH,#00H
```

```
                MOV BDPL,#42H
                MOV R3,#8               ;LBL1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV LBARL,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#5               ;DPH1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV BADRH,A

                MOV BDPH,#00H
                MOV BDPL,#42H
                MOV R3,#6               ;DPL1(NREC)
                LCALL SRT_LICZ
                MOV DPH,BDPH
                MOV DPL,BDPL
                MOVX A,@DPTR
                MOV BADRL,A

                CLR RECORD_FLAG
                SETB REC_FIRSTPR
                SETB AC_PWRD            ;wyłączenie przetwornika
REJ_END:        RET
;-----------------------------------------
; Pobranie próbki 12-bitowej z przetwornika
;-----------------------------------------
REJ_DACPROBE:   MOV RBUF,#0

                CLR AC_RC
                CLR AC_CS
                SETB AC_CS
                SETB AC_RC

REJ_BNOUP:      JNB AC_BUSY,REJ_BNOUP

                CLR AC_CS

                SETB AC_DATACLK
                CLR AC_DATACLK
                SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA11
                SETB RBUF.3
REJ_DATA11:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA10
                SETB RBUF.2
REJ_DATA10:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA09
                SETB RBUF.1
REJ_DATA09:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA08
                SETB RBUF.0
REJ_DATA08:     SETB AC_DATACLK
                CLR AC_DATACLK

                MOV R6,RBUF
                MOV RBUF,#0
```

```
                JNB AC_DATA,REJ_DATA07
                SETB RBUF.7
REJ_DATA07:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA06
                SETB RBUF.6
REJ_DATA06:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA05
                SETB RBUF.5
REJ_DATA05:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA04
                SETB RBUF.4
REJ_DATA04:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA03
                SETB RBUF.3
REJ_DATA03:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA02
                SETB RBUF.2
REJ_DATA02:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA01
                SETB RBUF.1
REJ_DATA01:     SETB AC_DATACLK
                CLR AC_DATACLK

                JNB AC_DATA,REJ_DATA00
                SETB RBUF.0
REJ_DATA00:     SETB AC_DATACLK
                CLR AC_DATACLK

                MOV R5,RBUF
                SETB AC_CS

                MOV A,R6
                MOV R1,A
                MOV A,R5
                MOV R0,A

                JNB REC_FIRSTPR,RDP_CONV
                MOV BPROBH,R6           ;próbka odniesienia
                MOV BPROBL,R5
                RET

RDP_CONV:       MOV A,R5                ;konwersja różnicowa 12 na 8-bitow
                XRL A,#11111111B        ;negacja
                MOV R5,A
                MOV A,R6
                XRL A,#11111111B
                MOV R6,A

                CLR C                   ;+1
                MOV A,R5
                ADD A,#1
                MOV R5,A
                MOV A,R6
                ADDC A,#0
                MOV R6,A

                CLR C                   ;próbka(n-1)-próbka(n) w kodzie U2
                MOV A,BPROBL
                ADD A,R5
                MOV R5,A
```

```
                MOV A,BPROBH
                ADDC A,R6
                MOV R6,A

                MOV BPROBL,R0
                MOV BPROBH,R1

                MOV RBUF,R6
                JB RBUF.7,RDP_BR67N0    ;ograniczenie wartości do 8 bitów
                CJNE R6,#00H,RDP_R6CHP  ;-128 do +127 (80H do 7FH)
                MOV RBUF,R5
                JB RBUF.7,RDP_R6CHP     ;wynik w R5
                RET
RDP_R6CHP:      MOV R5,#07FH
                RET
RDP_BR67N0:     CJNE R6,#0FFH,RDP_R6CHN
                MOV RBUF,R5
                JNB RBUF.7,RDP_R6CHN
                RET
RDP_R6CHN:      MOV R5,#80H
                RET
;*************************************
;*  PODPROGRAM OBSŁUGI PRZERWANIA RS   *
;*************************************
;
; zbiór rejestrów 2

RS_INT:         PUSH PSW
                PUSH ACC
                PUSH DPH
                PUSH DPL

              CLR PSW.3                 ;RS0=0 przełączenie na zbiór rej. 2
              SETB PSW.4                ;RS1=1

              JNB TI,RS_RECEIVE
              MOV A,STATUS
              CJNE A,#'e',RS_NOMONRUN
              CLR TI
              SJMP RS_END
RS_NOMONRUN:  CLR TI
              SETB RS_T_FLAG
              SJMP RS_END

RS_RECEIVE:   JNB RI,RS_END
              MOV BSBUF,SBUF
              MOV A,SBUF
                  JB MBREAK,RS_NO_SEND  ;blokada wszystkich komend
              CJNE A,#'e',RS_NO_MEKG
              MOV STATUS,#'e'           ;przełączenie rejestratora w tryb
              CLR RI                    ;monitora EKG
              SETB RS_R_FLAG
              SJMP RS_END
RS_NO_MEKG:   CJNE A,#'p',RS_NO_PROG
              MOV STATUS,#'p'           ;tryb programowania
              CLR RI
              SETB RS_R_FLAG
              SJMP RS_END
RS_NO_PROG:   CJNE A,#'m',RS_NO_MENU
              MOV STATUS,#'m'
              CLR RI
              SETB RS_R_FLAG
              SJMP RS_END
RS_NO_MENU:   CJNE A,#'s',RS_NO_SEND
              MOV STATUS,#'s'           ;tryb wysyłania danych
              CLR RI
              SETB RS_R_FLAG
              SJMP RS_END

RS_NO_SEND:   CLR RI
              SETB RS_R_FLAG

RS_END:       POP DPL
```

```
            POP DPH
            POP ACC
            POP PSW
            RETI
;*************************************
;* PODPROGRAM OBSŁUGI PRZERWANIA Z T2  *
;*************************************
; CLOCK: R7 zb. rej. 3
; STOS: PSW,ACC,DPH,DPL

CLK_TUEND:  POP DPL
            POP DPH
            POP ACC
            POP PSW
            CLR T2CON.7
            RETI

CLOCK:      PUSH PSW        ;PSW->STOS

            SETB PSW.3      ;RS0=1 przełączenie na zbiór rej. 3
            SETB PSW.4      ;RS1=1

            PUSH ACC     ;A->  STOS
            PUSH DPH     ;DPH->STOS
            PUSH DPL     ;DPL->STOS

            SETB KEYIN
            LCALL SETKEYS
            CLR KEY5
            JB KEYIN,CLK_TEND
            MOV STATUS,#'m'
CLK_TEND:   LCALL SETKEYS

            DJNZ R7,CLK_TUEND
            MOV R7,#14

            MOV A,TSS
            ADD A,#1
            DA A
            CJNE A,#01100000B,CLK_TUNO60S
            MOV A,#0
            MOV TSS,A

            MOV A,TSM
            ADD A,#1
            DA A
            CJNE A,#01100000B,CLK_TUNO60M
            MOV A,#0
            MOV TSM,A

            MOV A,TSH
            ADD A,#1
            DA A
            CJNE A,#00100100B,CLK_TUNO23H
            MOV A,#0
            MOV TSH,A

            MOV A,#'t'
            CJNE A,STATUS,CLK_DALEJ
            LCALL TIMES_DISPL
            LJMP CLK_DALEJ

CLK_TUNO23H: MOV TSH,A
            MOV A,#'t'
            CJNE A,STATUS,CLK_DALEJ
            LCALL TIMES_DISPL
            LJMP CLK_DALEJ
CLK_TUNO60M: MOV TSM,A
            MOV A,#'t'
            CJNE A,STATUS,CLK_DALEJ
            LCALL TIMES_DISPL
            LJMP CLK_DALEJ
```

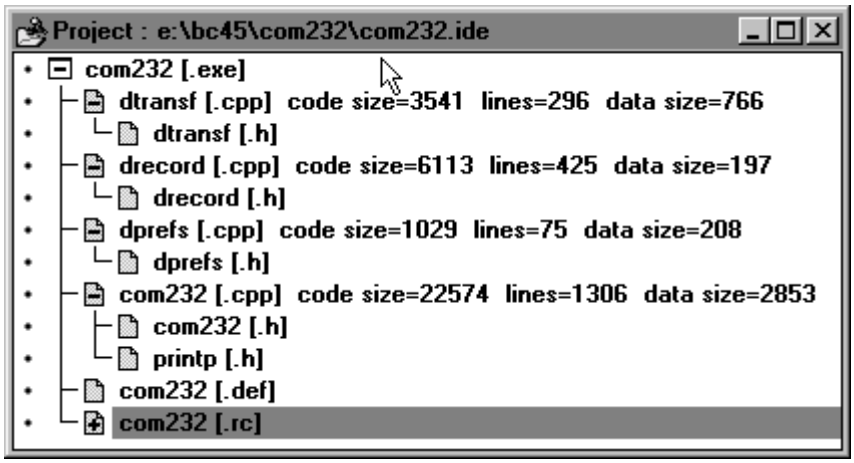
```
CLK_TUNO60S: MOV TSS,A
             MOV A,#'t'
             CJNE A,STATUS,CLK_DALEJ
             LCALL TIMES_DISPL
CLK_DALEJ:   MOV A,TSH          ;sprawdzenie zgodności z zaprogramowanym czasem
             CJNE A,TRH,CLK_TUNOALARM
             MOV A,TSM
             CJNE A,TRM,CLK_TUNOALARM
             MOV A,TSS
             CJNE A,TRS,CLK_TUNOALARM
             MOV A,#'t'
             CJNE A,STATUS,CLK_TUNOALARM

             CLR AC_PWRD        ;włączenie przetwornika

             SETB RECORD_FLAG

CLK_TUNOALARM: POP DPL
             POP DPH
             POP ACC
             POP PSW
             CLR T2CON.7        ;TF2=0 - dla T2 musi być zerowany programowo
             RETI
;--------------------------------------
;       Podprogram Monitora EKG
;--------------------------------------
MONITOR1:    DB '  Monitor EKG   ',8

MONITOR_EKG: CLR AC_PWRD        ;włączenie przetwornika

             LCALL LCD_CLEAR
             LCALL LCD_CAH

             MOV DPTR,#MONITOR1
             LCALL LCD_WRS

             CLR RS_R_FLAG              ;kasowanie zgłoszenia komendy 'e'

MON_LOOP:    MOV A,STATUS
             CJNE A,#'m',MON_LOOP

             SETB AC_PWRD               ;wyłączenie przetwornika

             LJMP PMENU0
;--------------------------------------
;   Podprogram obsługi programowania
;--------------------------------------
; R3 - zbiór rej. 0

PROGRAM1:    DB ' Programowanie  ',8

PROGRAM:     SETB MBREAK                ;zablokowanie komend
             CLR RS_R_FLAG              ;kasowanie zgłoszenia komendy 'p'

             MOV DPTR,#0000H

             MOV R3,#2                  ;ilość odbieranych bajtów (520)
PROG_RLOOP2: MOV R4,#8
PROG_RLOOP1: JBC RS_R_FLAG,PROG_FLAG
             SJMP PROG_RLOOP1

PROG_FLAG:   MOV A,BSBUF
             MOVX @DPTR,A
             INC DPTR

             DJNZ R4,PROG_RLOOP1
             DJNZ R3,PROG_RLOOP2

             MOV STATUS,#'m'
             CLR MBREAK
             MOV L,#5
             LCALL BEEP
```

```
            MOV NREC,#0
            LJMP PMENU0
;-------------------------------------
;  Podprogram obsługi wysyłania danych
;-------------------------------------
SENDDATA1:  DB 'Wysy',3,'anie danych',8

SEND_DATA:  LCALL LCD_CLEAR
            LCALL LCD_CAH

            MOV DPTR,#SENDDATA1
            LCALL LCD_WRS

            CLR RS_R_FLAG           ;kasowanie zgłoszenia komendy 's'

            MOV DPTR,#0000H
            MOV R3,#128
SEND_TLOOP2: MOV R4,#00H
SEND_TLOOP1: MOVX A,@DPTR

            MOV SBUF,A
SEND_FWAIT: JBC RS_T_FLAG,SEND_TEND
            SJMP SEND_FWAIT

SEND_TEND:  MOV A,STATUS
            CJNE A,#'m',SEND_NM
            SJMP SEND_END

SEND_NM:    INC DPTR
            DJNZ R4,SEND_TLOOP1
            DJNZ R3,SEND_TLOOP2

            MOV STATUS,#'m'
SEND_END:   LJMP PMENU0
;-------------------------------------
            END
```

Dodatek B

Kod źródłowy programu rejestratora holterowskiego dla Windows

Program został napisany w języku C++ i skompilowany w programie Borland C++ 4.52 dla systemu operacyjnego Windows. W skład programu wchodzą pliki przedstawione poniżej na wydruku okna projektu.



Projekt w całości został umieszczony na dyskietce załączonej do niniejszej pracy.

```
//----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik Com232.cpp
//----------------------------------------------------------------------------
#include <owl\owlpch.h>
#include <owl\applicat.h>
#include <owl\window.h>
#include <owl\dc.h>
#include <owl\gdiobjec.h>
#include <owl\decframe.h>
#include <owl\chooseco.h>
#include <owl\choosefo.h>
#include <owl\opensave.h>
#include <owl\printdia.h>
#include <owl\scroller.h>
#include <owl\controlb.h>
#include <owl\floatfra.h>
#include <owl\toolbox.h>
#include <owl\buttonga.h>
#include <owl\editfile.rh>
#include <dir.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "com232.rh"
#include "com232.h"
#include "dprefs.h"                          // TPrefsDialog
#include "drecord.h"                         // TRecordDialog
#include "dtransf.h"                         // TTransfDialog
#include "printp.h"
//----------------------------------------------------------------
const char plik_ini[]="com232.ini";          // Plik .ini
//----------------------------------------------------------------
TDecoratedFrame*    frame;                   // MainWindow

TStatusBar          *sb;
```

```
TTextGadget         *tg1,*tg2,*tg3,*tg4;

struct TPrefs Prefs;                    // Parametry portu szeregowego
struct TRecordEKG Record,*rec;

static TColor CustColors[32];

extern BYTE ConvBCD2B(BYTE lBCD);

class TRejestraror : public TWindow {
  public:
        TRejestraror();
        ~TRejestraror();

        bool IdleAction(long idleCount);

  protected:
      void   SetupWindow();
      void   EvSize(UINT SizeType, TSize& Size);
      void   EvDestroy();
      void   Paint(TDC& dc,bool,TRect&);

      TOpenSaveDialog::TData   FileData;
      HGLOBAL hgBufor,hgBuforW;         // wskaźnik do bufora pamięci
      BYTE*  BuforB;                    // wskaźnik do bufora pamięci typu BYTE
      WORD*  BuforBW;

      void   CmPlikZakoncz();
      void   CmOpcjePort();
      void   CmPlikInfo();
      void   CmPlikOtworz();
      void   CmPlikZapisz();
      void   CmPlikNastRekord();
      void   CmPlikPopRekord();
      void   CmPlikDrukuj();
      void   CmPlikUstawieniaStr();
      void   CmRejOdczytaj();
      void   CmRejProgramuj();
      void   CmRejCzasyOdczytow();
      void   CmRejMonitorEKG();
      void   CeRejMonitorEKG(TCommandEnabler& ce);
      void   CmRejWykresLiniowy();
      void   CeRejWykresLiniowy(TCommandEnabler& ce);
      void   CmRejKompSklSt();
      void   CeRejKompSklSt(TCommandEnabler& ce);
      void   CmRejPionLinia();
      void   CeRejPionLinia(TCommandEnabler& ce);
      void   CmOpcjeKolorTla();
      void   CmOpcjeKolorWykresu();

      void   ZapiszDoPliku(char *fileName);
      void   OdczytajZPliku(char *fileName);

      void   OpenPort();
      void   ClosePort();
      void   OdbZPortu(ULONG IloscB);
      void   GCError(int err);

      void   Konwertuj();

      TPrintDialog::TData PrintData;

      int idComDev,err;
      DCB s_dcb;
      COMSTAT s_comstat;

  private:
      TDC* dc;
      LOGFONT lf;
      TFont  *Arialfont;
      TFont  *ArialBfont;
```

```
      int cx,cy;                // rozmiar okna
      int status;         // stan portu COM 0-zamkniety 1-otwarty

      TChooseColorDialog::TData chooseT;
      TChooseColorDialog::TData chooseW;

      int NumWRec;
      int NPART;
      BOOL MONITOR;

      TPrinter* drukarka;

      TPen *penW,*penT;
      int idle;
      int bufsigw,prevbufsigw;

      BOOL WRITELINE;
      BOOL PRINTSET;
      BOOL KOMP;
      BOOL VERTLINE;

      UINT adr,lp,rp;
      char CzasRej[40];

  DECLARE_RESPONSE_TABLE(TRejestraror);
};

DEFINE_RESPONSE_TABLE1(TRejestraror, TWindow)
  EV_WM_SIZE,
  EV_WM_DESTROY,
  EV_COMMAND(CM_PLIKOTWORZ,                   CmPlikOtworz),
  EV_COMMAND(CM_PLIKZAPISZ,                   CmPlikZapisz),
  EV_COMMAND(CM_PLIKNASTREKORD,               CmPlikNastRekord),
  EV_COMMAND(CM_PLIKPOPREKORD,                CmPlikPopRekord),
  EV_COMMAND(CM_PLIKDRUKUJ,                   CmPlikDrukuj),
  EV_COMMAND(CM_PLIKUSTAWIENIASTR,            CmPlikUstawieniaStr),
  EV_COMMAND(CM_PLIKOKNO,                     CmPlikInfo),
  EV_COMMAND(CM_PLIKZAKONCZ,                  CmPlikZakoncz),
  EV_COMMAND(CM_REJODCZYTAJ,                  CmRejOdczytaj),
  EV_COMMAND(CM_REJPROGRAMUJ,                 CmRejProgramuj),
  EV_COMMAND(CM_REJCZASYODCZYTOW,             CmRejCzasyOdczytow),
  EV_COMMAND(CM_REJMONITOR,                   CmRejMonitorEKG),
  EV_COMMAND_ENABLE(CM_REJMONITOR,            CeRejMonitorEKG),
  EV_COMMAND(CM_OPCJEWYKRESLINIOWY,           CmRejWykresLiniowy),
  EV_COMMAND_ENABLE(CM_OPCJEWYKRESLINIOWY,    CeRejWykresLiniowy),
  EV_COMMAND(CM_OPCJEKOMPSKLST,               CmRejKompSklSt),
  EV_COMMAND_ENABLE(CM_OPCJEKOMPSKLST,        CeRejKompSklSt),
  EV_COMMAND(CM_OPCJEPIONLINIA,               CmRejPionLinia),
  EV_COMMAND_ENABLE(CM_OPCJEPIONLINIA,        CeRejPionLinia),
  EV_COMMAND(CM_OPCJEPORT,                    CmOpcjePort),
  EV_COMMAND(CM_OPCJEKOLOR_TLA,               CmOpcjeKolorTla),
  EV_COMMAND(CM_OPCJEKOLOR_WYKRESU,           CmOpcjeKolorWykresu),
END_RESPONSE_TABLE;

TRejestraror::TRejestraror() : TWindow(0,0,0) // konstruktor
{
 char buf[10];
 status = 0;              //port zamkniety
 NPART = 0;
 MONITOR = FALSE;
 WRITELINE = TRUE;
 KOMP = FALSE;
 PRINTSET = FALSE;
 VERTLINE = TRUE;
 //--- Odczytanie ustawien programu ---
 Prefs.port=GetPrivateProfileInt("Ustawienia","PORT",1,plik_ini);
 Prefs.szybkosc=GetPrivateProfileInt("Ustawienia","SZYBKOSC",9600,plik_ini);
 Prefs.bitydanych=GetPrivateProfileInt("Ustawienia","BITYDANYCH",8,plik_ini);
 GetPrivateProfileString("Ustawienia","PARZYSTOSC","N",
                                 buf,sizeof(buf),plik_ini);
                                 Prefs.parzystosc=*buf;
 Prefs.bitystopu=GetPrivateProfileInt("Ustawienia","BITYSTOPU",1,plik_ini);
```

```
 GetPrivateProfileString("DIR","FILE","C:\\COM232\\REJESTR\\TEST.REC",

Prefs.DIR,sizeof(Prefs.DIR),plik_ini);
 //--- Ustawienie parametrow czcionki ---
 memset(&lf,0,sizeof(LOGFONT));
 strcpy(lf.lfFaceName,"Arial");
              lf.lfHeight=15;
              lf.lfUnderline=0;
              lf.lfItalic=0;
              lf.lfStrikeOut=0;
 Arialfont=new TFont(&lf);
 strcpy(lf.lfFaceName,"Arial Black");
              lf.lfHeight=15;
              lf.lfUnderline=0;
              lf.lfItalic=0;
              lf.lfStrikeOut=0;
 ArialBfont=new TFont(&lf);
//-------------- Alokacja pamięci bufora rekordu ------------------
 hgBufor = GlobalAlloc(GHND,65536);
       if(hgBufor==NULL){
              MessageBox(" Za mało pamięci do alokacji           \n bufora
rekordu! ",
                                    "Błąd",MB_OK|MB_ICONSTOP);}
 void FAR *lpvBufor = GlobalLock(hgBufor);
       if(lpvBufor==NULL){
              MessageBox(" Błąd funkcji GlobalLock(). ",
                                    "Błąd",MB_OK|MB_ICONSTOP);}
       BuforB = (BYTE*)lpvBufor;
//-------------- Alokacja w pamięci bufora wyświetlania ------------------
 hgBuforW = GlobalAlloc(GHND,131070); //128 kB (2*64kB)
       if(hgBuforW==NULL){
              MessageBox(" Za mało pamięci do alokacji           \n bufora
wyswietlania danych! ",
                                    "Błąd",MB_OK|MB_ICONSTOP);}
 void FAR *lpvBuforW = GlobalLock(hgBuforW);
       if(lpvBuforW==NULL){
              MessageBox(" Błąd funkcji GlobalLock() (WYSW). ",
                                    "Błąd",MB_OK|MB_ICONSTOP);}
       BuforBW = (WORD*)lpvBuforW;
//-----------------------------------------------------------

 Attr.Style |= WS_HSCROLL;
 drukarka = new TPrinter;

 penW = 0;
 penT = 0;
 idle = 0;
}

TRejestraror::~TRejestraror()
{
if(hgBufor){
       GlobalUnlock(hgBufor);
       GlobalFree(hgBufor);
       }
 delete dc;
 delete drukarka;
 delete penW;
 delete penT;
}

void TRejestraror::SetupWindow()
{
  TWindow::SetupWindow();
  dc = new TClientDC(HWindow);

  Scroller = new TScroller(this, 1, 1, 0, 0);

  char text[100];
  wsprintf(text,"COM%d",Prefs.port);
  tg1->SetText(text);
  wsprintf(text,"%d %d-%c-%d",   Prefs.szybkosc,
```

```
      Prefs.bitydanych,

      Prefs.parzystosc,

      Prefs.bitystopu);
  tg2->SetText(text);

  wsprintf(text,"%s %s",Record.Imie,Record.Nazwisko);
  tg3->SetText(text);

  NumWRec = 0;

if(hgBufor!=NULL){
      HFILE hffile = _lopen("com232.cfg",READ);
      if(hffile != HFILE_ERROR) {
            TChooseColorDialog::TData *chT=&chooseT;
            TChooseColorDialog::TData *chW=&chooseW;
            void *wv;
            BYTE *bajt;
            int sizeT = sizeof(chooseT);
            int sizeW = sizeof(chooseW);
            int sizecc= sizeof(CustColors);
            _hread(hffile, BuforB, sizeT+sizeW+sizecc);
            wv=chT;
            bajt = (BYTE*)wv;
            for(int i=0;i<sizeT;i++){
              *(bajt+i)=BuforB[i];
             }
            wv=chW;
            bajt = (BYTE*)wv;
            for(i=0;i<sizeW;i++){
              *(bajt+i)=BuforB[i+sizeT];
             }
            wv=&CustColors;
            bajt = (BYTE*)wv;
            for(i=0;i<sizecc;i++){
              *(bajt+i)=BuforB[i+sizeT+sizeW];
             }
       _lclose(hffile);
       TRejestraror::SetBkgndColor(COLORREF(chooseT.Color));
       penW = new TPen(COLORREF(chooseW.Color),1,PS_SOLID);
       penT = new TPen(COLORREF(chooseT.Color),1,PS_SOLID);
      }else{
       chooseT.Flags = CC_RGBINIT;
       chooseT.CustColors = CustColors;
       chooseT.Color = TColor::Black;
       chooseW.Flags = CC_RGBINIT;
       chooseW.CustColors = CustColors;
       chooseW.Color = TColor::LtGreen;
       TRejestraror::SetBkgndColor(COLORREF(TColor::Black));
       penW = new TPen(COLORREF(chooseW.Color),1,PS_SOLID);
       penT = new TPen(COLORREF(chooseT.Color),1,PS_SOLID);
      }
 }
}

void TRejestraror::EvDestroy()
{
 char buf[200];
 // Zapis ustawień portu szeregowego do pliku .ini
 wsprintf(buf,"%d",Prefs.port);
 WritePrivateProfileString("Ustawienia","PORT",buf,plik_ini);
 wsprintf(buf,"%d",Prefs.szybkosc);
 WritePrivateProfileString("Ustawienia","SZYBKOSC",buf,plik_ini);
 wsprintf(buf,"%d",Prefs.bitydanych);
 WritePrivateProfileString("Ustawienia","BITYDANYCH",buf,plik_ini);
 wsprintf(buf,"%c",Prefs.parzystosc);
 WritePrivateProfileString("Ustawienia","PARZYSTOSC",buf,plik_ini);
 wsprintf(buf,"%d",Prefs.bitystopu);
 WritePrivateProfileString("Ustawienia","BITYSTOPU",buf,plik_ini);
```

```
 WritePrivateProfileString("DIR","FILE",Prefs.DIR,plik_ini);

 if(hgBufor!=NULL){
       HFILE hffile = _lcreat("com232.cfg",0);
       if(hffile != -1) {
              TChooseColorDialog::TData *chT=&chooseT;
              void *wv;
              BYTE *bajt;
              wv=chT;
              bajt = (BYTE*)wv;
              int sizeT = sizeof(chooseT);
              for(int i=0;i<sizeT;i++){
                BuforB[i]=*(bajt+i);
               }
              TChooseColorDialog::TData *chW=&chooseW;
              wv=chW;
              bajt = (BYTE*)wv;
              int sizeW = sizeof(chooseW);
              for(i=0;i<sizeW;i++){
                BuforB[i+sizeT]=*(bajt+i);
               }
              wv=&CustColors;
              bajt = (BYTE*)wv;
              int sizecc = sizeof(CustColors);
              for(i=0;i<sizecc;i++){
                BuforB[i+sizeT+sizeW]=*(bajt+i);
               }

              _hwrite(hffile, BuforB, sizeT+sizeW+sizecc);
              _lclose(hffile);
        }
 }
 TWindow::EvDestroy();
}

void TRejestraror::EvSize(UINT SizeType, TSize& Size)
{
 TWindow::EvSize(SizeType, Size);
 if(SizeType != SIZEICONIC){
       Invalidate();
 }
 if(MONITOR==TRUE)
       idle=0;
}

void TRejestraror::Paint(TDC& dc,bool,TRect&)
{
 UINT lb;
 char buf[200];
 long rx;
 int tmpS,tmpM,tmpH;

 TRect crect = GetClientRect();

 if(MONITOR == FALSE){

 int index = NumWRec-1;
 if(index >= 0){
  if(NPART==0){ //lb < 32768){
       adr = Record.TabCzasow[9*index+5]*256+Record.TabCzasow[9*index+6];
       //DPH*256+DPL
       lb =  Record.TabCzasow[9*index+7]*256+Record.TabCzasow[9*index+8];
       //LBH*256+LBL
  }else{
       if(NPART==1){
         adr = Record.TabCzasow[9*index+5]*256+Record.TabCzasow[9*index+6];
         //DPH*256+DPL
         lb =  32768;
        }
       if(NPART==2){
         adr = (Record.TabCzasow[9*index+5]*256+Record.TabCzasow[9*index+6])
                     +32768;      //DPH*256+DPL
```

```
        lb =  (Record.TabCzasow[9*index+7]*256+Record.TabCzasow[9*index+8])
                    -32768;      //LBH*256+LBL
       }
  }

 if(lb >= crect.Width()){
      rx = lb-crect.Width();
      Scroller->SetRange(rx, 0);
  }else
      Scroller->SetRange( 0, 0);

  lp=(UINT)crect.left+(UINT)Scroller->XPos;

  if(lb >= (UINT)crect.Width()){
       rp=(UINT)crect.left+(UINT)Scroller->XPos+(UINT)crect.Width();
      }else
       rp=(UINT)crect.left+(UINT)Scroller->XPos+lb;

      tmpS = ConvBCD2B(Record.TabCzasow[9*index+2]) + (lp/Record.Fprobkowania);

      if(tmpS < 59)
            tmpM = ConvBCD2B(Record.TabCzasow[9*index+1]);
      else{
            tmpM = ConvBCD2B(Record.TabCzasow[9*index+1]) + tmpS/60;
            tmpS = tmpS - (tmpS/60)*60;
      }

      if(tmpM < 59)
            tmpH = ConvBCD2B(Record.TabCzasow[9*index+0]);
      else{
            tmpH = ConvBCD2B(Record.TabCzasow[9*index+0]) + tmpM/60;
            tmpM = tmpM - (tmpM/60)*60;
      }

      wsprintf(CzasRej,"%d:%02d:%02d", tmpH, tmpM, tmpS);
      wsprintf(buf,"Czas rejestracji: %s, próbkowanie %d Hz", CzasRej,
                                                    Record.Fprobkowania);
      sb->SetText(buf);

      PRINTSET = TRUE;    //ustawione zostały właściwe parametry wydruku

 for(UINT i=lp;i<rp;i++){
      if(WRITELINE == FALSE){
            dc.SetPixel(i,(int)BuforBW[adr+i],COLORREF(chooseW.Color));
      }
      else{
            dc.SelectObject(*penW);
            if(abs((int)BuforBW[adr+i]-(int)BuforBW[adr+i+1]) > 1){
                  dc.MoveTo(i,(int)BuforBW[adr+i]);
                  dc.LineTo(i,(int)BuforBW[adr+i+1]);
            }
            else{
                  dc.SetPixel(i,(int)BuforBW[adr+i],COLORREF(chooseW.Color));
            }
      }
   }
  }
 }
}

bool TRejestraror::IdleAction(long idleCount)
{
 char buf[100];
 TClientDC dc(*this);
 TRect cr = GetClientRect();

if(MONITOR==TRUE){

 err = ReadComm(idComDev, BuforB+idle, 600);
 dc.SelectObject(*penT);
 if(err > 0){
  for(int x=0; x<err; x++){
```

```
      dc.MoveTo(idle,cr.top);
      dc.LineTo(idle,cr.bottom);
      signed char sigc = (signed char)BuforB[idle];
      int sigw = (int)sigc;
      bufsigw = bufsigw - sigw;

      if(WRITELINE == FALSE){
            dc.SetPixel(idle,(cr.bottom/2)+(bufsigw),COLORREF(chooseW.Color));
      }
      else{
            if(abs(prevbufsigw - bufsigw) > 1){
                  dc.SelectObject(*penW);
                  dc.MoveTo(idle,(cr.bottom/2)+(prevbufsigw));
                  dc.LineTo(idle,(cr.bottom/2)+(bufsigw));
                  dc.SelectObject(*penT);
            }
            else{

      dc.SetPixel(idle,(cr.bottom/2)+(prevbufsigw),COLORREF(chooseW.Color));
            }
            prevbufsigw = bufsigw;
      }

      if(idle < cr.Width())
        idle++;
       else{
        idle=0;
        bufsigw = 0;
        }
      }
  }

 if(VERTLINE == TRUE){                  //Wyświetlanie pionowej linii
      dc.SelectObject(*penW);
      dc.MoveTo(idle,cr.top);
      dc.LineTo(idle,cr.bottom);
      dc.SelectObject(*penT);
 }
 Invalidate(FALSE);
 }
 TWindow::IdleAction(idleCount);
}

class Tcom232App : public TApplication {
  public:
        Tcom232App() : TApplication("Com232 v.1.0") {}
        void InitMainWindow();

  protected:
        TGadgetWindow::THintMode HintMode;
        TControlBar* ControlBar;

        void CmOpcjeOpisy();
        void CeOpcjeOpisy(TCommandEnabler&);

DECLARE_RESPONSE_TABLE(Tcom232App);
};
DEFINE_RESPONSE_TABLE1(Tcom232App, TApplication)
      EV_COMMAND(CM_OPCJEOPISY,              CmOpcjeOpisy),
      EV_COMMAND_ENABLE(CM_OPCJEOPISY, CeOpcjeOpisy),
END_RESPONSE_TABLE;

static TControlBar*
BuildControlBar(TWindow* parent, TControlBar::TTileDirection direction)
{
  TControlBar* cb = new TControlBar(parent, direction);
  cb->Insert(*new TButtonGadget(BMP_PLIKOTWORZ,CM_PLIKOTWORZ ));
  cb->Insert(*new TButtonGadget(BMP_PLIKZAPISZ,CM_PLIKZAPISZ ));
  cb->Insert(*new TSeparatorGadget(6));
  cb->Insert(*new TButtonGadget(BMP_REJPROGRAMUJ,CM_REJPROGRAMUJ ));
  cb->Insert(*new TButtonGadget(BMP_REJODCZYTAJ,CM_REJODCZYTAJ ));
  cb->Insert(*new TSeparatorGadget(6));
```

```
  cb->Insert(*new TButtonGadget(BMP_CZASYREJ,CM_REJCZASYODCZYTOW ));

  cb->Insert(*new TSeparatorGadget(6));
  cb->Insert(*new
TButtonGadget(BMP_MONITOR,CM_REJMONITOR,TButtonGadget::Exclusive));
  cb->Insert(*new TSeparatorGadget(6));
  cb->Insert(*new TButtonGadget(BMP_PLIKDRUKUJ,CM_PLIKDRUKUJ ));

  cb->Insert(*new TSeparatorGadget(6));
  cb->Insert(*new TSeparatorGadget(6));

  cb->Insert(*new TButtonGadget(BMP_RECDOWN,CM_PLIKPOPREKORD));
  cb->Insert(*new TSeparatorGadget(6));

  tg4 = new
TTextGadget(ID_TXTGADGET1,TTextGadget::Recessed,TTextGadget::Center,4,"0/0");
  cb->Insert(*tg4);
  cb->Insert(*new TSeparatorGadget(6));

  cb->Insert(*new TButtonGadget(BMP_RECUP,CM_PLIKNASTREKORD));
  cb->Insert(*new TSeparatorGadget(6));
  cb->Insert(*new TSeparatorGadget(6));

  tg3 = new
TTextGadget(ID_TXTGADGET1,TTextGadget::Recessed,TTextGadget::Center,15,"");
  cb->Insert(*tg3);
  cb->Insert(*new TSeparatorGadget(6));
  cb->Insert(*new TSeparatorGadget(6));
  cb->Insert(*new TButtonGadget(BMP_TOGGLE,CM_OPCJEOPISY));

  cb->Attr.Style |= WS_CLIPSIBLINGS;
  cb->Attr.Id = IDW_TOOLBAR;
  return cb;
}
void Tcom232App::InitMainWindow()
{
  EnableBWCC(TRUE);

  TRejestraror& client = *new TRejestraror;
  frame = new TDecoratedFrame(0, Name,&client,TRUE);

// Odczytanie pozycji i rozmiaru okna z pliku .ini
  frame->Attr.X = GetPrivateProfileInt("Startup","Left",140,plik_ini);
  frame->Attr.Y = GetPrivateProfileInt("Startup","Right",100,plik_ini);
  frame->Attr.W = GetPrivateProfileInt("Startup","Width",340,plik_ini);
  frame->Attr.H = GetPrivateProfileInt("Startup","Height",200,plik_ini);

  MainWindow = frame;
  frame->Attr.AccelTable = IDA_COM232;
  frame->SetMenuDescr(TMenuDescr(IDM_COM232, 1, 1, 1, 1, 1, 1));
  MainWindow->SetIcon(this, "ICONA1");

// Konstrukcja TStatusBar
       sb = new TStatusBar(0, TGadget::Recessed);
       frame->Insert(*sb, TDecoratedFrame::Bottom);
       tg1 = new TTextGadget(10,TTextGadget::Recessed,
                             TTextGadget::Center,4);
       tg2 = new TTextGadget(11,TTextGadget::Recessed,
                             TTextGadget::Center,7);
       sb->Insert(*tg1);
       sb->Insert(*tg2);

// Konstrukcja control bar -> frame.
       HintMode = TGadgetWindow::PressHints;
       ControlBar = BuildControlBar(frame, TControlBar::Horizontal);
       frame->Insert(*ControlBar, TDecoratedFrame::Top);

       for(int m=0;m<sizeof(Record.TabCzasow);m++){
             Record.TabCzasow[m]=0;
       }

// Ustawienie wartości domyślnych struktury Record
```

```
            wsprintf(Record.HEADER1,"RECEKG v.1.0");
            wsprintf(Record.Imie,"");
            wsprintf(Record.Nazwisko,"");
            wsprintf(Record.Numer,"0");
            Record.IloscRekordow=(BYTE)0;
            Record.Fprobkowania =(BYTE)100;
            for(int x=0;x<=sizeof(Record.TabCzasow);x++){
                  Record.TabCzasow[x]=(BYTE)0;
            }
            Record.TabCzasow[5]=(BYTE)0x02; // DPH pierwszego rekordu (520)
            Record.TabCzasow[6]=(BYTE)0x08;  // DPL pierwszego rekordu

            Record.TabCzasow[7]=(BYTE)0x00;
            Record.TabCzasow[8]=(BYTE)0x00;

            wsprintf(Record.HEADER2,"DATA");
}

bool TApplication::CanClose()
{
 char buf[20];
 // Zapisanie parametrów okna do pliku .ini
 wsprintf(buf,"%d",MainWindow->Attr.X);
 WritePrivateProfileString("Startup","Left",buf,plik_ini);
 wsprintf(buf,"%d",MainWindow->Attr.Y);
 WritePrivateProfileString("Startup","Right",buf,plik_ini);
 wsprintf(buf,"%d",MainWindow->Attr.W);
 WritePrivateProfileString("Startup","Width",buf,plik_ini);
 wsprintf(buf,"%d",MainWindow->Attr.H);
 WritePrivateProfileString("Startup","Height",buf,plik_ini);
 return true;
}

void Tcom232App::CeOpcjeOpisy(TCommandEnabler& ce)
{
  ce.SetCheck(HintMode == TGadgetWindow::EnterHints);
}

void Tcom232App::CmOpcjeOpisy()
{
  HintMode = HintMode==TGadgetWindow::PressHints ?
                                  TGadgetWindow::EnterHints :
                                  TGadgetWindow::PressHints;
  ControlBar->SetHintMode(HintMode);
  ControlBar->SetHintCommand(-1);
}

int OwlMain(int /*argc*/, char* /*argv*/ [])
{
 if(Tcom232App().hPrevInstance)
       return 0;
  else
       return Tcom232App().Run();
}

void TRejestraror::CmPlikDrukuj()
{
  if((PRINTSET == TRUE) && (MONITOR == FALSE)){
       rec = &Record;
       if(drukarka){
             DokumentPrn printout("Wykres EKG", this, rec, BuforBW, adr, lp,
                                                                  rp, CzasRej);
             printout.TrybDruku(TRUE);
             drukarka->Print( this, printout, TRUE);
       }
  }
}

void TRejestraror::CmPlikUstawieniaStr()
{
 if (drukarka)
        drukarka->Setup(this);
```

```
}

void TRejestraror::CmPlikZakoncz()
{
 CloseWindow(0);
}

void TRejestraror::CmOpcjePort()
{
      TPrefsDialog(this,Prefs.port,
                                          Prefs.szybkosc,
                                          Prefs.bitydanych,
                                          Prefs.parzystosc,
                                          Prefs.bitystopu).Execute();

      char text[30];
      wsprintf(text,"COM%d",Prefs.port);
      tg1->SetText(text);
      wsprintf(text,"%d %d-%c-%d",     Prefs.szybkosc,

      Prefs.bitydanych,

      Prefs.parzystosc,

      Prefs.bitystopu);
      tg2->SetText(text);
      Invalidate();
}

void TRejestraror::CmOpcjeKolorTla()
{
  chooseT.Flags = CC_RGBINIT;
  chooseT.CustColors = CustColors;

 if (TChooseColorDialog(this, chooseT).Execute() == IDOK) {
      TRejestraror::SetBkgndColor(COLORREF(chooseT.Color));
      if(penT)
       delete penT;
       penT = new TPen(COLORREF(chooseT.Color),1,PS_SOLID);
      Invalidate();
  }
}

void TRejestraror::CmOpcjeKolorWykresu()
{
  chooseW.Flags = CC_RGBINIT;
  chooseW.CustColors = CustColors;

 if (TChooseColorDialog(this, chooseW).Execute() == IDOK) {
  dc->SelectObject(TPen(chooseW.Color,8));
  if(penW)
       delete penW;
       penW = new TPen(COLORREF(chooseW.Color),1,PS_SOLID);

  Invalidate();
  }
}

void TRejestraror::CmPlikOtworz()
{
 *FileData.FileName = 0;
 TOpenSaveDialog::TData FileData (
      OFN_HIDEREADONLY|OFN_FILEMUSTEXIST|OFN_NOREADONLYRETURN,
      "Pliki rejestratora (*.rec)|*.rec|Wszystkie pliki (*.*)|*.*",
      0,
      Prefs.DIR,
      "rec");

  if(TFileOpenDialog(frame, FileData).Execute() == IDOK) {
       OdczytajZPliku(FileData.FileName);
       wsprintf(Prefs.DIR,"%s",FileData.FileName);
       char fileTitle[30],buf[200];
```

```
        TOpenSaveDialog::GetFileTitle(FileData.FileName, fileTitle, 30);
        Scroller->ScrollTo(0, 0);
        Konwertuj();
        Invalidate();
       }
}

void TRejestraror::CmPlikZapisz()
{
 *FileData.FileName = 0;
 TOpenSaveDialog::TData FileData (
        OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
        "Pliki rejestratora (*.rec)|*.rec|Wszystkie pliki (*.*)|*.*",
        0,
        Prefs.DIR,
        "rec");
  if (TFileSaveDialog(frame, FileData).Execute() == IDOK) {
        wsprintf(Prefs.DIR,"%s",FileData.FileName);
              char fileTitle[30],buf[200];;
              TOpenSaveDialog::GetFileTitle(FileData.FileName, fileTitle, 30);
              wsprintf(buf,"Wykonany zapis do pliku %s",fileTitle);
              sb->SetText(buf);
              ZapiszDoPliku(FileData.FileName);
         }
}
void TRejestraror::ZapiszDoPliku(char *fileName)
{
 if(hgBufor==NULL){
       return;
       }
 HFILE hffile = _lcreat(fileName, 0);
 if(hffile == -1) {
       return;
       }
       struct TRecordEKG *rekord=&Record;
       void *wv;
       BYTE *bajt;
       wv=rekord;
       bajt = (BYTE*)wv;
       unsigned int size=sizeof(Record);
       for(unsigned int i=0;i<size;i++){
              BuforB[i]=*(bajt+i);
              }

              unsigned int adr = Record.TabCzasow[5]*256+Record.TabCzasow[6];
              unsigned int lb;

       unsigned int adrb = adr;
       for(int x=0;x<Record.IloscRekordow;x++){
         adr=adr+Record.TabCzasow[9*x+7]*256+Record.TabCzasow[9*x+8];
        }
        size=adr+3;

        BuforB[adr+0]='E';
        BuforB[adr+1]='N';
        BuforB[adr+2]='D';

       _hwrite(hffile, BuforB, size);
       _lclose(hffile);
}

void TRejestraror::OdczytajZPliku(char *fileName)
{
 if(hgBufor==NULL){
       return;
       }
 HFILE hffile = _lopen(fileName,READ);
 if(hffile == -1) {
       return;
       }
       struct TRecordEKG *rekord=&Record;
       void *wv;
```

```
        BYTE *bajt;
        wv=rekord;
        bajt = (BYTE*)wv;
        unsigned int size=65535;
        _hread(hffile, BuforB, size);
        for(unsigned int i=0;i<sizeof(Record);i++){
              *(bajt+i)=BuforB[i];
              }
        _lclose(hffile);

        char bufor[200];
        NumWRec = 1;
        int index = 0;
        UINT lb =  Record.TabCzasow[9*index+7]*256+Record.TabCzasow[9*index+8];
                                                                    //LBH*256+LBL
        if(lb > 32768){
          NPART=1;
          wsprintf(bufor,"%d.%d/%d",NumWRec,NPART,Record.IloscRekordow);
         }else{
          NPART=0;
          wsprintf(bufor,"%d/%d",NumWRec,Record.IloscRekordow);
         }
        tg4->SetText(bufor);

        wsprintf(bufor,"%s %s",Record.Imie,Record.Nazwisko);
        tg3->SetText(bufor);
}

void TRejestraror::CmPlikNastRekord()
{
 char bufor[200];
 if(NumWRec < Record.IloscRekordow){
  if(NPART==0)
       NumWRec++;
  }

 int index = NumWRec-1;
 if(index >= 0){
  UINT lb =  Record.TabCzasow[9*index+7]*256+Record.TabCzasow[9*index+8];
                                                                    //LBH*256+LBL
  if(lb > 32768){
              switch(NPART){
                     case 0:
                      NPART=1;
                      break;
                     case 1:
                      NPART=2;
                      break;
                     case 2:
                      if(NumWRec < Record.IloscRekordow){
                            NumWRec++;
                            NPART=0;
                       }
                      break;
               }
       }
  }

  if(NPART == 0)
        wsprintf(bufor,"%d/%d",NumWRec,Record.IloscRekordow);
       else
        wsprintf(bufor,"%d.%d/%d",NumWRec,NPART,Record.IloscRekordow);

  tg4->SetText(bufor);
  Scroller->ScrollTo(0, 0);
  Invalidate();
}

void TRejestraror::CmPlikPopRekord()
{
 char bufor[200];
 if(NumWRec > 1){
```

```
  if(NPART==0){
       NumWRec--;
      }
  }

 int index = NumWRec-1;
 if(index >= 0){
  UINT lb =  Record.TabCzasow[9*index+7]*256+Record.TabCzasow[9*index+8];
                                                                    //LBH*256+LBL
  if((lb > 32768)&&(NPART==0)){
       NPART=2;
      }else{
            switch(NPART){
                  case 2:
                   NPART=1;
                   break;
                  case 1:
                   if(NumWRec > 1){
                        NumWRec--;
                        NPART=0;
                    }
                   break;
             }
      }
  }
  if(NPART == 0)
       wsprintf(bufor,"%d/%d",NumWRec,Record.IloscRekordow);
      else
       wsprintf(bufor,"%d.%d/%d",NumWRec,NPART,Record.IloscRekordow);

  tg4->SetText(bufor);
  Scroller->ScrollTo(0, 0);
  Invalidate();
}

void TRejestraror::CmRejCzasyOdczytow()
{
      TRecordDialog(this,Record.IloscRekordow,
                                              Record.Imie,
                                              Record.Nazwisko,
                                              Record.Numer,
                                              Record.Fprobkowania,
                                              Record.TabCzasow).Execute();
}

void TRejestraror::CmRejOdczytaj()
{
 int Tryb = 1;
 TTransfDialog(this,Tryb,Prefs,Record,BuforB).Execute();
 Konwertuj();

 char text[150];
      wsprintf(text,"%s %s",Record.Imie,Record.Nazwisko);
      tg3->SetText(text);

      NumWRec = 1;
      int index = 0;
      UINT lb =  Record.TabCzasow[9*index+7]*256+Record.TabCzasow[9*index+8];
                                                                    //LBH*256+LBL
      if(lb > 32768){
        NPART=1;
        wsprintf(text,"%d.%d/%d",NumWRec,NPART,Record.IloscRekordow);
       }else{
        NPART=0;
        wsprintf(text,"%d/%d",NumWRec,Record.IloscRekordow);
       }
      tg4->SetText(text);
      Invalidate();
}

void TRejestraror::Konwertuj()
{
```

```
 int index,i,j;
 DWORD SUM,SR,SRODN,dSR;

 TRect cr = GetClientRect();
 bufsigw = cr.bottom/2;

 for(index=0; index < Record.IloscRekordow; index++){
      UINT lb =  Record.TabCzasow[9*index+7]*256+Record.TabCzasow[9*index+8];
                                                                  //LBH*256+LBL
      UINT adr = Record.TabCzasow[9*index+5]*256+Record.TabCzasow[9*index+6];
                                                                  //DPH*256+DPL
      for(i=0; i<lb; i++){
            signed char sigc = (signed char)BuforB[adr+i];
            int sigw = (int)sigc;
            bufsigw = bufsigw - sigw;
            BuforBW[adr+i]=bufsigw;
      }

      if(KOMP == TRUE){                     //kompensacja składowej stałej
            for(i=0; i<lb; i++){
                  SUM = 0;
                  for(j=0; j<500; j++){
                        SUM = SUM + BuforBW[adr+i+j];
                  }
                  SR = SUM/500;
                  if(i == 0)
                        SRODN = SR;

                  dSR = SRODN - SR;

                  if(i < lb+500-1)
                        BuforBW[adr+i+500-1] = BuforBW[adr+i+500-1] + dSR;
            }
      }
 }
}

void TRejestraror::CmRejProgramuj()
{
 int Tryb = 0;                          // Tryb: 0 - nadawanie, 1 - odbieranie
 TTransfDialog(this,Tryb,Prefs,Record,BuforB).Execute();
 Invalidate();
}

void TRejestraror::CmRejMonitorEKG()
{
 TRect cr = GetClientRect();

 if(MONITOR){
       TransmitCommChar(idComDev, 'm');
       ClosePort();
       OdczytajZPliku("temp.rec");
       idle = 0;
       MONITOR=FALSE;
       Invalidate();
  }else{
       sb->SetText("Monitor EKG");
       ZapiszDoPliku("temp.rec");
       OpenPort();
       Scroller->SetRange( 0, 0);

       for(UINT idle = 0; idle < cr.Width(); idle++){
            BuforB[idle]=cr.Height()/2+20;
        }

       idle = 0;
       bufsigw = 0;
       prevbufsigw = 0;
       MONITOR=TRUE;
       TransmitCommChar(idComDev, 'm');
       TransmitCommChar(idComDev, 'e');
       Invalidate();
```

```
       }
}

void TRejestraror::CmRejWykresLiniowy()
{
 if(WRITELINE == TRUE)
       WRITELINE = FALSE;
 else
       WRITELINE = TRUE;

 idle = 0;
 bufsigw = 0;
 Invalidate();
}

void TRejestraror::CmRejKompSklSt()
{
       if(MONITOR == FALSE){
              if(KOMP == TRUE)
                     KOMP = FALSE;
              else
                     KOMP = TRUE;

              Konwertuj();
              Invalidate();
       }
}

void TRejestraror::CmRejPionLinia()
{
 if(VERTLINE == TRUE)
       VERTLINE = FALSE;
 else
       VERTLINE = TRUE;

 idle = 0;
 bufsigw = 0;
 Invalidate();
}

void TRejestraror::CeRejMonitorEKG(TCommandEnabler& ce)
{
  ce.SetCheck(MONITOR ?
       TCommandEnabler::Checked : TCommandEnabler::Unchecked);
}

void TRejestraror::CeRejWykresLiniowy(TCommandEnabler& ce)
{
  ce.SetCheck(WRITELINE ?
       TCommandEnabler::Checked : TCommandEnabler::Unchecked);
}

void TRejestraror::CeRejKompSklSt(TCommandEnabler& ce)
{
  ce.SetCheck(KOMP ?
       TCommandEnabler::Checked : TCommandEnabler::Unchecked);
}

void TRejestraror::CeRejPionLinia(TCommandEnabler& ce)
{
  ce.SetCheck(VERTLINE ?
       TCommandEnabler::Checked : TCommandEnabler::Unchecked);
}

void TRejestraror::OpenPort()
{
 char    buf[100];
 char terr[40]="";

 wsprintf(buf,"COM%d",Prefs.port);
 idComDev = OpenComm(buf, 12288, 12288); //bufor 12 kB
 if(idComDev < 0){
```

```
      if(idComDev == IE_OPEN){wsprintf(terr,"Port jest już otwarty.");}
      if(idComDev == IE_HARDWARE){wsprintf(terr,"Port jest już zajęty.");}
      wsprintf(buf,"Blad otwarcia COM%d.\n%s\n%d",Prefs.port,terr,idComDev);
      MessageBox(buf,"Błąd",MB_ICONSTOP);
      }
 wsprintf(buf,"COM%d:%d,%c,%d,%d",     Prefs.port,
                                       Prefs.szybkosc,
                                       Prefs.parzystosc,
                                       Prefs.bitydanych,
                                       Prefs.bitystopu);
 err = BuildCommDCB(buf,&s_dcb);
 if(err < 0){
      MessageBox("Błąd BulidCommDCB.","Błąd",MB_ICONSTOP);
      }
 err = SetCommState(&s_dcb);
 if(err < 0){
      MessageBox("Błąd SetCommState.","Błąd",MB_ICONSTOP);
      }
}

void TRejestraror::ClosePort()
{
 err = CloseComm(idComDev);
 if(err != 0){
      MessageBox("Blad zamknięcia portu.","Błąd",MB_ICONSTOP);
      }
}

void TRejestraror::OdbZPortu(ULONG IloscB)
{
 char buf[100];
 ULONG cbRead=IloscB;

      err = ReadComm(idComDev, BuforB, cbRead);
      err = GetCommError(idComDev, &s_comstat);  GCError(err);

      wsprintf(buf," inQ = %u, outQ = %u",s_comstat.cbInQue,
                                          s_comstat.cbOutQue);
      sb->SetText(buf);
}

void TRejestraror::GCError(int err)
{
 char buf[100];
 char terr[100];
 if(err != 0){
      wsprintf(terr,"?");
      if(err == CE_BREAK){wsprintf(terr,"Hardware detected a break condition.");}
      if(err == CE_CTSTO){wsprintf(terr,"CTS (clear-to-send) timeout.");}
      if(err == CE_DSRTO){wsprintf(terr,"DSR (data-set-ready) timeout.");}
      if(err == CE_FRAME){wsprintf(terr,"Hardware detected a framing error.");}
      if(err == CE_OVERRUN){wsprintf(terr,"The character was lost.");}
      if(err == CE_RXOVER){wsprintf(terr,"Receiving queue overflowed.");}
      if(err == CE_RXPARITY){wsprintf(terr,"Hardware detected a parity error.");}
      if(err == CE_TXFULL){wsprintf(terr,"Transmission queue was full");}

      wsprintf(buf,"Błąd transmisji.\n%s \n%d",terr,err);
      MessageBox(buf,"Błąd GetCommError.",MB_ICONSTOP);
      }
}

void TRejestraror::CmPlikInfo()
{
  TDialog *info = new TDialog(this,IDD_INFORMACJA);
  info->Execute();
}

//------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik Com232.h
//------------------------------------------------------------------------
```

```
#ifndef __COM232_H
#define __COM232_H

struct TPrefs{                                  // Parametry portu szeregowego
              int   port;
              int   szybkosc;
              int   bitydanych;
              char  parzystosc;
              int   bitystopu;
              char  DIR[80];
              };

struct TRecordEKG {
              char  HEADER1[12];                // "RECEKG v.1.0"
              char  Imie[20];
              char  Nazwisko[20];
              char  Numer[12];
              BYTE  IloscRekordow;
              BYTE  Fprobkowania;
              BYTE  TabCzasow[450];
              char  HEADER2[4];
              };

#endif

//                    0     1     2     3     4     5     6     7     8
// TabCzasow :       |TRH1 |TRM1 |TRS1 |TTM1 |TTS1 |DPH  |DPL  |LBH  |LBL  |
// HEADER2:          "DATA",9*50 = 450 bajtów  max. 50 zapisów po około 12 s

// TRH1 - początek rejestracji ( godziny )
// TRM1 - początek rejestracji ( minuty )
// TRS1 - początek rejestracji ( sekundy )
// TTM1 - czas trwania rejestracji ( minuty )
// TTS1 - czas trwania rejestracji ( sekundy )
// DPH  - starszy bajt adresu początku zapisu w pamięci ( DPTR )
// DPL  - młodszy bajt adresu początku zapisu w pamięci ( DPTR )
// LBH  - liczba bajtów zapisu ( H )
// LBL  - liczba bajtów zapisu ( L )

//-----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik printp.h
//-----------------------------------------------------------------------------

#ifndef __PRINTP_H
#define __PRINTP_H

#include <owl/owlpch.h>
#include "com232.h"

class DokumentPrn : public TPrintout
{
 public:
  DokumentPrn(const char* title, TWindow* okno, TRecordEKG *rec,
              WORD* bufB, UINT adr, UINT lp, UINT rp, char* CzasRej);
  void GetDialogInfo(int& minPage, int& maxPage,int& selFromPage,
                     int& selToPage);
  void PrintPage(int page, TRect& rect, unsigned flags);
  void TrybDruku(BOOL b) {Banding = b;}
  BOOL HasPage(int pageNumber){
       return pageNumber == 1;
       }
 protected:
       TWindow* Okno;
       TRecordEKG *Rec;
       WORD* BufB;
       TPen* Pen;
       TBrush* Brush;
       UINT badr,blp,brp;
       char* bCzasRej;
};
```

```
DokumentPrn::DokumentPrn( const char* title, TWindow* okno, TRecordEKG *rec,
                          WORD* bufB, UINT adr, UINT lp, UINT rp,
                          char* CzasRej):TPrintout(title)
{
 Okno = okno;
 Rec = rec;
 BufB = bufB;
 badr = adr;
 blp = lp;
 brp = rp;
 bCzasRej = CzasRej;
}

void DokumentPrn::PrintPage(int, TRect& rect, unsigned)
{
int    old_mode;
TSize  oldVExt, oldWExt;
// arbitralnie skalujemy osie: 1 jedn. na osi  x =  1 jedn. na osi y
old_mode = DC->SetMapMode(MM_ANISOTROPIC);

TRect rozm_okna = Okno->GetClientRect();      // pobieramy obszar roboczy

int iloscpix = rozm_okna.right;
rozm_okna.right = rozm_okna.right + 2*12;
rozm_okna.bottom = rozm_okna.bottom + 2*12;

DC->SetViewportExt(PageSize, &oldVExt);// ustalamy nowy rozmiar strony

// ustalamy rozmiar okna związanego z DC
DC->SetWindowExt(rozm_okna.Size(), &oldWExt);

DC->Rectangle(rozm_okna.left+10,rozm_okna.top+10,
              rozm_okna.right-10,rozm_okna.bottom-10);
DC->Rectangle(rozm_okna.left+12,rozm_okna.top+12,
              rozm_okna.right-12,rozm_okna.bottom-12);

Pen = new TPen(TColor::Black,1);
DC->SelectObject(*Pen);
Brush = new TBrush(TColor::Black);
DC->SelectObject(*Brush);

DC->MoveTo(rozm_okna.right-12,rozm_okna.bottom-12-20);
DC->LineTo(rozm_okna.right-12-300,rozm_okna.bottom-12-20);
DC->LineTo(rozm_okna.right-12-300,rozm_okna.bottom-12);
DC->MoveTo(rozm_okna.right-12-150,rozm_okna.bottom-12-20);
DC->LineTo(rozm_okna.right-12-150,rozm_okna.bottom-12);
DC->MoveTo(rozm_okna.right-12-75,rozm_okna.bottom-12-20);
DC->LineTo(rozm_okna.right-12-75,rozm_okna.bottom-12);

delete Pen;
Pen = new TPen(TColor::Black,2);
DC->SelectObject(*Pen);

DC->MoveTo(13,(int)BufB[badr+blp]);

for(UINT i=1; i<iloscpix-1; i++){
      DC->LineTo(13+i,(int)BufB[badr+blp+i]);
  }

LOGFONT lf;
memset(&lf,0,sizeof(LOGFONT));
strcpy(lf.lfFaceName,"Times New Roman CE");
              lf.lfHeight=10;
              lf.lfWidth =6;
              lf.lfUnderline=0;
              lf.lfItalic=0;
              lf.lfStrikeOut=0;
              lf.lfCharSet=DEFAULT_CHARSET;   // gwarantuje polskie znaki
              lf.lfQuality=DRAFT_QUALITY;

TFont *ptfont=new TFont(&lf);
DC->SelectObject(*ptfont);
```

```
char buf[200];
wsprintf(buf,"%s %s",Rec->Imie,Rec->Nazwisko);
DC->TextOut(rozm_okna.right-12-290,rozm_okna.bottom-12-15,buf);
wsprintf(buf,"%s",Rec->Numer);
DC->TextOut(rozm_okna.right-12-140,rozm_okna.bottom-12-15,buf);
wsprintf(buf,"%s",bCzasRej);
DC->TextOut(rozm_okna.right-12-65,rozm_okna.bottom-12-15,buf);

DC->RestoreObjects();

DC->SetWindowExt(oldWExt);       // odtwarzamy poprzedni kontekst DC
DC->SetViewportExt(oldVExt);
DC->SetMapMode(old_mode);

DC->RestorePen();
DC->RestoreFont();
DC->RestoreBrush();
delete Pen;
delete ptfont;
delete Brush;
}

void DokumentPrn::GetDialogInfo(int& minPage, int& maxPage,
                                int& selFromPage, int& selToPage)
{
      minPage = maxPage = 0;
      selFromPage = selToPage = 0;
}

#endif

//----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik dprefs.cpp
//----------------------------------------------------------------------------

#include <owl\owlpch.h>
#include <owl\applicat.h>
#include <owl\button.h>
#include "dprefs.h"
#include "com232.rh"

DEFINE_RESPONSE_TABLE1(TPrefsDialog, TDialog)
  EV_BN_CLICKED(IDC_RBCOM1,  BnCom1),
  EV_BN_CLICKED(IDC_RBCOM2,  BnCom2),
  EV_BN_CLICKED(IDC_RB4800,  Bn4800),
  EV_BN_CLICKED(IDC_RB9600,  Bn9600),
  EV_BN_CLICKED(IDC_RB19200, Bn19200),
  EV_BN_CLICKED(IDC_RBBD7,   BnBD7),
  EV_BN_CLICKED(IDC_RBBD8,   BnBD8),
  EV_BN_CLICKED(IDC_RBKBRAK, BnKBrak),
  EV_BN_CLICKED(IDC_RBKPARZ, BnKParz),
  EV_BN_CLICKED(IDC_RBBS1,   BnBS1),
  EV_BN_CLICKED(IDC_RBBS2,   BnBS2),
  EV_CHILD_NOTIFY_ALL_CODES(IDOK,BOk),
END_RESPONSE_TABLE;

TPrefsDialog::TPrefsDialog(TWindow* parent,
                           int& port,
                           int& szybkosc,
                           int& bitydanych,
                           char& parzystosc,
                           int& bitystopu,
                           TModule* module)
 : TDialog(parent,IDD_DPREFS, module),
      Port(port),Szybkosc(szybkosc),Bitydanych(bitydanych),
      Parzystosc(parzystosc),Bitystopu(bitystopu)
{
}

void TPrefsDialog::SetupWindow()
```

```
{
  TDialog::SetupWindow();
  tPort=Port;
  tSzybkosc=Szybkosc;
  tBitydanych=Bitydanych;
  tParzystosc=Parzystosc;
  tBitystopu=Bitystopu;

  CheckRadioButton(IDC_RBCOM1, IDC_RBCOM2,
              Port == 1 ? IDC_RBCOM1 : IDC_RBCOM2);
  CheckRadioButton(IDC_RB4800, IDC_RB19200,
              Szybkosc == 4800 ? IDC_RB4800 : Szybkosc == 9600 ?
                                              IDC_RB9600 : IDC_RB19200);
  CheckRadioButton(IDC_RBBD7, IDC_RBBD8,
              Bitydanych == 7 ? IDC_RBBD7 : IDC_RBBD8);
  CheckRadioButton(IDC_RBKBRAK, IDC_RBKPARZ,
              Parzystosc == 'N' ? IDC_RBKBRAK : IDC_RBKPARZ);
  CheckRadioButton(IDC_RBBS1,IDC_RBBS2,
              Bitystopu == 1 ? IDC_RBBS1 : IDC_RBBS2);
}

BOOL TPrefsDialog::CanClose()
{
  return TRUE;
}

void TPrefsDialog::BOk(UINT /*notify*/)
{
  Port=tPort;
  Szybkosc=tSzybkosc;
  Bitydanych=tBitydanych;
  Parzystosc=tParzystosc;
  Bitystopu=tBitystopu;

  this->Destroy();
}

//----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik dprefs.h
//----------------------------------------------------------------------------

#ifndef __DPREFS_H
#define __DPREFS_H

#include <owl\dialog.h>

class TPrefsDialog : public TDialog {
  public:
       TPrefsDialog(TWindow* parent,int& port,int& szybkosc,
                    int& bitydanych,char& parzystosc,int& bitystopu,
                    TModule* module = 0);

  private:
        void SetupWindow();
        BOOL CanClose();

        void BnCom1()      {tPort = 1;}
        void BnCom2()      {tPort = 2;}
        void Bn4800()      {tSzybkosc = 4800;}
        void Bn9600()      {tSzybkosc = 9600;}
        void Bn19200()     {tSzybkosc = 19200;}
        void BnBD7(){tBitydanych = 7;}
        void BnBD8(){tBitydanych = 8;}
        void BnKBrak()     {tParzystosc = 'N';}
        void BnKParz()     {tParzystosc = 'E';}
        void BnBS1(){tBitystopu = 1;}
        void BnBS2(){tBitystopu = 2;}

        void BOk(UINT /*notify*/);

        int&  Port,        tPort;
```

```
        int&  Szybkosc,    tSzybkosc;
        int&  Bitydanych,  tBitydanych;
        char& Parzystosc,  tParzystosc;
        int&  Bitystopu,   tBitystopu;

  DECLARE_RESPONSE_TABLE(TPrefsDialog);
};

#endif

//----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik drecord.cpp
//----------------------------------------------------------------------------

#include <owl\owlpch.h>
#include <owl\applicat.h>
#include <owl\button.h>
#include <owl\combobox.h>
#include "drecord.h"
#include "com232.rh"

BYTE ConvB2BCD(BYTE lB)
{
 BYTE buf;
 buf=lB/10;
 lB=lB-(buf*10);
 buf=buf<<4;
 lB=buf|lB;
 return lB;
}

BYTE ConvBCD2B(BYTE lBCD)
{
 BYTE buf;
 buf=lBCD&0xf0;
 buf=buf>>4;
 buf=buf*10;
 lBCD=lBCD&0x0f;
 lBCD=buf+lBCD;
 return lBCD;
}

DEFINE_RESPONSE_TABLE1(TRecordDialog, TDialog)
  EV_CHILD_NOTIFY_ALL_CODES(IDOK,BOk),
  EV_CHILD_NOTIFY_ALL_CODES(IDCANCEL,BCancel),
  EV_CBN_SELCHANGE(IDC_CBOX1,UpdateCBOX1),
  EV_CHILD_NOTIFY_ALL_CODES(IDC_EDITILOSCREC,ZmIlosciRekordow),
  EV_CBN_SELCHANGE(IDC_CBOXCZESTOTP,UpdateCBOXCZESTOTP),
  EV_CHILD_NOTIFY_ALL_CODES(IDC_EDITPOCZREJG,UpdateEDITPOCZREJG),
  EV_CHILD_NOTIFY_ALL_CODES(IDC_EDITPOCZREJM,UpdateEDITPOCZREJM),
  EV_CHILD_NOTIFY_ALL_CODES(IDC_EDITPOCZREJS,UpdateEDITPOCZREJS),
  EV_CHILD_NOTIFY_ALL_CODES(IDC_EDITCZASTRWM,UpdateEDITCZASTRWM),
  EV_CHILD_NOTIFY_ALL_CODES(IDC_EDITCZASTRWS,UpdateEDITCZASTRWS),
END_RESPONSE_TABLE;

TRecordDialog::TRecordDialog(    TWindow*      parent,
                                 BYTE&  iloscRekordow,
                                 char*  imie,
                                 char*  nazwisko,
                                 char*   numer,
                                 BYTE&  fprobkowania,
                                 BYTE*   tabCzasow,
                                 TModule* module)
 : TDialog(parent,IDD_DRECORD, module),
       IloscRekordow(iloscRekordow),Imie(imie),Nazwisko(nazwisko),Numer(numer),
       Fprobkowania(fprobkowania),TabCzasow(tabCzasow)
{
}

void TRecordDialog::SetupWindow()
{
```

```
 TDialog::SetupWindow();

 int size = sizeof(tTabCzasow);
 for(int i=0;i<size;i++){
      tTabCzasow[i]=*(TabCzasow+i);
  }
      index = 0;
      ComboBOX1 = new TComboBox(this,IDC_CBOX1);
      ComboBOX1->Create();
      ComboBOX1->SetFocus();
      ComboBOX1->ClearList();
      UstawIloscComboBOX1(IloscRekordow);
      ComboBOXCzestotP = new TComboBox(this,IDC_CBOXCZESTOTP);
      ComboBOXCzestotP->Create(); ComboBOXCzestotP->SetFocus();
      ComboBOXCzestotP->AddString("80");
      ComboBOXCzestotP->AddString("100");
      ComboBOXCzestotP->AddString("120");
      if(Fprobkowania == 80)
            ComboBOXCzestotP->SetSelIndex(0);
            else{
                  if(Fprobkowania == 100)
                        ComboBOXCzestotP->SetSelIndex(1);
                        else
                        ComboBOXCzestotP->SetSelIndex(2);}

      SetDlgItemText(IDC_EDITIMIE,Imie);
      SetDlgItemText(IDC_EDITNAZWISKO,Nazwisko);
      SetDlgItemText(IDC_EDITNUMER,Numer);

      for(int x=0;x<50;x++){
            TabCzasow[9*x+0] = ConvBCD2B(TabCzasow[9*x+0]);
            TabCzasow[9*x+1] = ConvBCD2B(TabCzasow[9*x+1]);
            TabCzasow[9*x+2] = ConvBCD2B(TabCzasow[9*x+2]);
            TabCzasow[9*x+3] = ConvBCD2B(TabCzasow[9*x+3]);
            TabCzasow[9*x+4] = ConvBCD2B(TabCzasow[9*x+4]);
            }

      SetDlgItemInt(IDC_EDITILOSCREC,IloscRekordow);
}

BOOL TRecordDialog::CanClose()
{
  return TRUE;
}

void TRecordDialog::BOk(UINT /*notify*/)
{
  // końcowa aktualizacja danych
  char buf[60];
  GetDlgItemText(IDC_EDITILOSCREC,buf,sizeof(buf));
  IloscRekordow = (BYTE)atoi(buf);
  GetDlgItemText(IDC_EDITIMIE,Imie,20);
  GetDlgItemText(IDC_EDITNAZWISKO,Nazwisko,20);
  GetDlgItemText(IDC_EDITNUMER,Numer,20);

  DWORD ls,lb,adr;
  int i;
  for(i=0;i<IloscRekordow;i++){
       ls=60*TabCzasow[9*i+3]+TabCzasow[9*i+4];
       lb=Fprobkowania*ls;
       TabCzasow[9*i+7]=lb/256;                              // LBH
       TabCzasow[9*i+8]=lb-(TabCzasow[9*i+7]*256);           // LBL
       adr = TabCzasow[9*i+5]*256+TabCzasow[9*i+6];          // DPH*256+DPL
       adr = adr + lb;
       if(i<49){
        TabCzasow[9*(i+1)+5] = adr/256;                          //DPH
        TabCzasow[9*(i+1)+6] = adr - TabCzasow[9*(i+1)+5]*256;  //DPL
        }
       }

  for(int x=0;x<50;x++){
       TabCzasow[9*x+0] = ConvB2BCD(TabCzasow[9*x+0]);
```

```
      TabCzasow[9*x+1] = ConvB2BCD(TabCzasow[9*x+1]);
      TabCzasow[9*x+2] = ConvB2BCD(TabCzasow[9*x+2]);
      TabCzasow[9*x+3] = ConvB2BCD(TabCzasow[9*x+3]);
      TabCzasow[9*x+4] = ConvB2BCD(TabCzasow[9*x+4]);
      }

  this->Destroy();
}

void TRecordDialog::UpdateCBOX1()
{
  index = ComboBOX1->GetSelIndex();    // pobranie nowego indeksu rekordu
  if(index != -1){
       SetDlgItemInt(IDC_EDITPOCZREJG,TabCzasow[9*index+0]);
       SetDlgItemInt(IDC_EDITPOCZREJM,TabCzasow[9*index+1]);
       SetDlgItemInt(IDC_EDITPOCZREJS,TabCzasow[9*index+2]);
       SetDlgItemInt(IDC_EDITCZASTRWM,TabCzasow[9*index+3]);
       SetDlgItemInt(IDC_EDITCZASTRWS,TabCzasow[9*index+4]);
       }
}

void TRecordDialog::UpdateEDITPOCZREJG(UINT notify)
{
if(notify == EN_CHANGE)  // Sprawdzenie czy nastąpiła zmiana danych
 {
 char buf[15];
 if(index != -1){
      GetDlgItemText(IDC_EDITPOCZREJG,buf,15);
      TabCzasow[9*index] = atoi(buf);
            if(TabCzasow[9*index] > 23){
                  TabCzasow[9*index] = 23;
                  SetDlgItemInt(IDC_EDITPOCZREJG,23);
                  }
      }
 }
}

void TRecordDialog::UpdateEDITPOCZREJM(UINT notify)
{
if(notify == EN_CHANGE)
 {
 char buf[15];
 if(index != -1){
      GetDlgItemText(IDC_EDITPOCZREJM,buf,15);
      TabCzasow[9*index+1] = atoi(buf);
      if(TabCzasow[9*index+1] > 59){
                  TabCzasow[9*index+1]=59;
                  SetDlgItemInt(IDC_EDITPOCZREJM,59);
                  }
      }
 }
}

void TRecordDialog::UpdateEDITPOCZREJS(UINT notify)
{
if(notify == EN_CHANGE)
 {
 char buf[15];
 if(index != -1){
      GetDlgItemText(IDC_EDITPOCZREJS,buf,15);
      TabCzasow[9*index+2] = atoi(buf);
      if(TabCzasow[9*index+2] > 59){
                  TabCzasow[9*index+2]=59;
                  SetDlgItemInt(IDC_EDITPOCZREJS,59);
                  }
      }
 }
}

void TRecordDialog::UpdateEDITCZASTRWM(UINT notify)
{
if(notify == EN_CHANGE)
```

```
 {
 char buf[15];
 if(index != -1){
       GetDlgItemText(IDC_EDITCZASTRWM,buf,15);
       TabCzasow[9*index+3] = atoi(buf);
       if(TabCzasow[9*index+3] > 99){
         TabCzasow[9*index+3]=99;
         SetDlgItemInt(IDC_EDITCZASTRWM,99);
        }
        SprLB();
  }
 }
}

void TRecordDialog::UpdateEDITCZASTRWS(UINT notify)
{
if(notify == EN_CHANGE)
 {
 char buf[15];
 if(index != -1){
       GetDlgItemText(IDC_EDITCZASTRWS,buf,15);
       TabCzasow[9*index+4] = atoi(buf);
       if(TabCzasow[9*index+4] > 59){
         TabCzasow[9*index+4]=59;
         SetDlgItemInt(IDC_EDITCZASTRWS,59);
        }
       SprLB();
  }
 }
}

void TRecordDialog::UpdateCBOXCZESTOTP()
{
  char  buf[10];
  index = ComboBOXCzestotP->GetSelIndex();
  if (index != -1) {
        ComboBOXCzestotP->GetString(buf, index);
        Fprobkowania = atoi(buf);
        DWORD ls,lb;
        lb=0;
        for(int i=0;i<IloscRekordow;i++){
              ls=60*TabCzasow[9*i+3]+TabCzasow[9*i+4];
              lb=lb+Fprobkowania*ls;
              if(lb > 65012){
                index=i;
                for(int x=i+1;x<IloscRekordow;x++){
                      TabCzasow[9*x+3]=0;
                      TabCzasow[9*x+4]=0;
                     }
                SprLB();
                ComboBOX1->SetSelIndex(index);
                break;
               }
         }
         SprLB();
        }
}

void TRecordDialog::ZmIlosciRekordow(UINT notify)
{
 if(notify == EN_CHANGE)
       {
        ComboBOX1->ClearList();
        HWND hwndIloscRekordow = GetDlgItem(IDC_EDITILOSCREC);
        int n = (::GetWindowTextLength(hwndIloscRekordow));
        HWND hwndIDOK = GetDlgItem(IDOK);
        HWND hwndEDITPOCZREJG = GetDlgItem(IDC_EDITPOCZREJG);
        HWND hwndEDITPOCZREJM = GetDlgItem(IDC_EDITPOCZREJM);
        HWND hwndEDITPOCZREJS = GetDlgItem(IDC_EDITPOCZREJS);
        HWND hwndEDITCZASTRWM = GetDlgItem(IDC_EDITCZASTRWM);
        HWND hwndEDITCZASTRWS = GetDlgItem(IDC_EDITCZASTRWS);
        char buf[20];
```

```
        GetDlgItemText(IDC_EDITILOSCREC,buf,sizeof(buf));
        IloscRekordow = (BYTE)atoi(buf);
        if((n == 0)||(IloscRekordow == 0)){
              ::EnableWindow(hwndIDOK,FALSE);
              ::EnableWindow(hwndEDITPOCZREJG,FALSE);
              ::EnableWindow(hwndEDITPOCZREJM,FALSE);
              ::EnableWindow(hwndEDITPOCZREJS,FALSE);
              ::EnableWindow(hwndEDITCZASTRWM,FALSE);
              ::EnableWindow(hwndEDITCZASTRWS,FALSE);
                    SprLB();
              }else{
                    ::EnableWindow(hwndIDOK,TRUE);
                    ::EnableWindow(hwndEDITPOCZREJG,TRUE);
                    ::EnableWindow(hwndEDITPOCZREJM,TRUE);
                    ::EnableWindow(hwndEDITPOCZREJS,TRUE);
                    ::EnableWindow(hwndEDITCZASTRWM,TRUE);
                    ::EnableWindow(hwndEDITCZASTRWS,TRUE);
                    // Ustawienie aktualnej liczby rekordów
                    UstawIloscComboBOX1(IloscRekordow);

                    SetDlgItemInt(IDC_EDITPOCZREJG,TabCzasow[0]);
                    SetDlgItemInt(IDC_EDITPOCZREJM,TabCzasow[1]);
                    SetDlgItemInt(IDC_EDITPOCZREJS,TabCzasow[2]);
                    SetDlgItemInt(IDC_EDITCZASTRWM,TabCzasow[3]);
                    SetDlgItemInt(IDC_EDITCZASTRWS,TabCzasow[4]);
              }
        }
}

void TRecordDialog::UstawIloscComboBOX1(BYTE ilosc)
{
 char buf[30];
 if(ilosc > 50){                          // ograniczenie liczby rekordów
        ilosc=50;  IloscRekordow=(BYTE)50;
        SetDlgItemInt(IDC_EDITILOSCREC,IloscRekordow);
  }
        for(int i=1;i<=ilosc;i++){
          wsprintf(buf,"rekord %d",i);
          ComboBOX1->AddString(buf);
         }
        ComboBOX1->SetSelIndex(0);   index = 0;
}

void TRecordDialog::SprLB()
{
 DWORD ls,lb,maxleft;
 lb=0;
 DWORD TM,TS;

 for(int i=0;i<IloscRekordow;i++){
        ls=60*TabCzasow[9*i+3]+TabCzasow[9*i+4];
        lb=lb+Fprobkowania*ls;
  }

 if(lb <= 32244){
        maxleft=32244-lb;
  }else{
        TabCzasow[9*index+3] = 0;
        TabCzasow[9*index+4] = 0;
        lb=0;
        for(int i=0;i<IloscRekordow;i++){
          ls=60*TabCzasow[9*i+3]+TabCzasow[9*i+4];
          lb=lb+Fprobkowania*ls;
         }
         if(lb <= 32244){
              maxleft=32244-lb;
              TM = (maxleft/Fprobkowania)/60;
              TS = (maxleft-(60*TM*Fprobkowania))/Fprobkowania;
              SetDlgItemInt(IDC_EDITCZASTRWM,TM);
              SetDlgItemInt(IDC_EDITCZASTRWS,TS);
              TabCzasow[9*index+3] = TM;
              TabCzasow[9*index+4] = TS;
```

```
            maxleft=0;
         }
  }

  TM = (maxleft/Fprobkowania)/60;
  TS = (maxleft-(60*TM*Fprobkowania))/Fprobkowania;

  SetDlgItemInt(ID_STATTM,TM);
  SetDlgItemInt(ID_STATTS,TS);
}

void TRecordDialog::BCancel(UINT /*notify*/)
{
 int size = sizeof(tTabCzasow);
 for(int i=0;i<size;i++){
      TabCzasow[i]=*(tTabCzasow+i);
  }
 this->Destroy();
}

//----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik drecord.h
//----------------------------------------------------------------------------

#ifndef __DRECORD_H
#define __DRECORD_H

#include <owl\dialog.h>

class TRecordDialog : public TDialog {
  public:
       TRecordDialog(TWindow* parent,  BYTE&  iloscRekordow,
                                       char*  imie,
                                       char*  nazwisko,
                                       char*  numer,
                                       BYTE&  fprobkowania,
                                       BYTE*   tabCzasow,
                                       TModule* module = 0);
  private:
       void  SetupWindow();
       BOOL  CanClose();
       void  BOk(UINT /*notify*/);
       void  BCancel(UINT /*notify*/);
       void  UpdateCBOX1();
       void  ZmIlosciRekordow(UINT notify);
       void  UstawIloscComboBOX1(BYTE ilosc);
       void  UpdateCBOXCZESTOTP();
       void  UpdateEDITPOCZREJG(UINT notify);
       void  UpdateEDITPOCZREJM(UINT notify);
       void  UpdateEDITPOCZREJS(UINT notify);
       void  UpdateEDITCZASTRWM(UINT notify);
       void  UpdateEDITCZASTRWS(UINT notify);

       BYTE& IloscRekordow;
       char* Imie;
       char* Nazwisko;
       char* Numer;
       BYTE& Fprobkowania;
       BYTE* TabCzasow;
       BYTE tTabCzasow[450];

      protected:
            TComboBox* ComboBOX1;
            TComboBox* ComboBOXCzestotP;
            int index;                        //wskaźnik wybranego rekordu
            void SprLB();

  DECLARE_RESPONSE_TABLE(TRecordDialog);
};

#endif
```

```
//----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik dtransf.cpp
//----------------------------------------------------------------------------

#include <owl\owlpch.h>
#include <owl\applicat.h>
#include <owl\button.h>
#include <stdio.h>
#include <math.h>

#include "dtransf.h"
#include "com232.rh"

DEFINE_RESPONSE_TABLE1(TTransfDialog, TDialog)
 EV_COMMAND(IDC_START,CmStart),
END_RESPONSE_TABLE;

TTransfDialog::TTransfDialog(    TWindow* parent,
                                 int& tryb,
                                 TPrefs& prefs,
                                 TRecordEKG& record,
                                 BYTE* buforB,
                                 TModule* module)
 : TDialog(parent,IDD_DTRANSFER, module),
       Tryb(tryb),Prefs(prefs),Record(record),BuforB(buforB)
{
 Port = new TStatic( this, ID_STPORT);
 Status = new TStatic( this, ID_STSTATUS);
}

void TTransfDialog::SetupWindow()
{
 TDialog::SetupWindow();

 SetCaption("Programowanie rejestratora");

 wsprintf(buf,"COM%d:%d,%c,%d,%d",Prefs.port,Prefs.szybkosc,Prefs.parzystosc,
                                  Prefs.bitydanych,Prefs.bitystopu);
 Port->SetText(buf);
 Status->SetText("");
}

void TTransfDialog::CmStart()
{
 void *wv;
 LPSTR lpvBuf;
 wv=BuforB;
 lpvBuf = (LPSTR)wv;

 struct TRecordEKG *rekord=&Record;
 BYTE *bajt;
 wv = rekord;
 bajt = (BYTE*)wv;
 unsigned int i;

 if(Tryb == 0){                         //tryb nadawania danych
  for(i=0;i<sizeof(Record);i++){
             BuforB[i]=*(bajt+i);
             }
 ULONG cbWrite;
 cbWrite=768;
 OpenPort();
 TransmitCommChar(idComDev, 'm');
 TransmitCommChar(idComDev, 'p');
 Wysylaj(cbWrite);
 TransmitCommChar(idComDev, 'm');
 ClosePort();
 CloseWindow(0);
 }
 if(Tryb == 1){                         //tryb odbierania danych
```

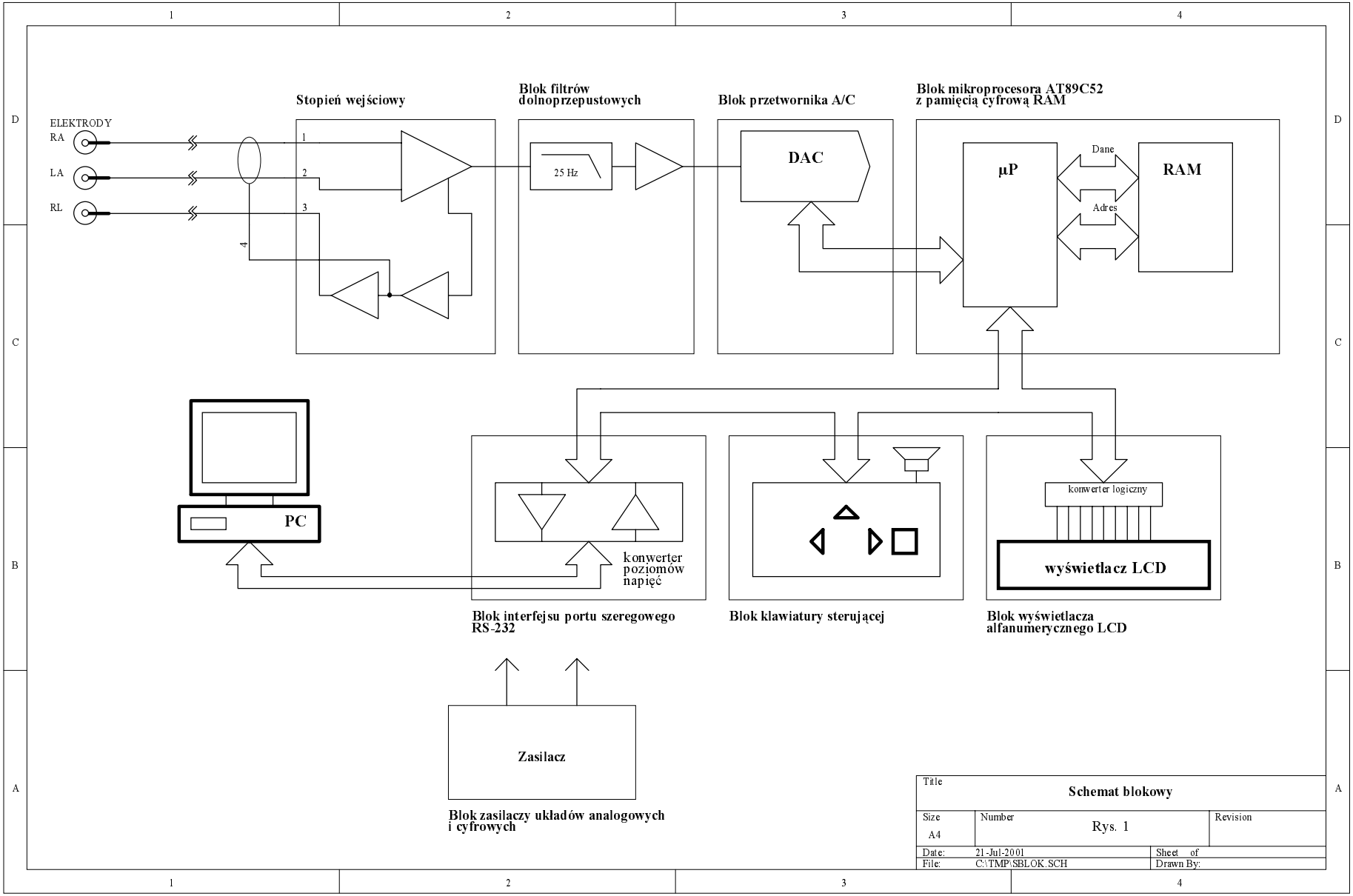
```
 ULONG cbRead;
 cbRead=32768;
 OpenPort();
 TransmitCommChar(idComDev, 'm');
 TransmitCommChar(idComDev, 's');
 Odbieraj(cbRead);
 ClosePort();
       for(i=0;i<sizeof(Record);i++){
             *(bajt+i)=BuforB[i];
             }
 CloseWindow(0);
 }
}

//Funkcje komunikacji przez port szeregowy

void TTransfDialog::OpenPort()
{
 char terr[40]="";
 wsprintf(buf,"COM%d",Prefs.port);
 idComDev = OpenComm(buf, 12288, 12288); // bufor 12 kB
 if(idComDev < 0){
       if(idComDev == IE_OPEN){wsprintf(terr,"Port jest już otwarty.");}
       if(idComDev == IE_HARDWARE){wsprintf(terr,"Port jest już zajęty.");}
       wsprintf(buf,"Blad otwarcia COM%d.\n%s\n%d",Prefs.port,terr,idComDev);
       MessageBox(buf,"Błąd",MB_ICONSTOP);
       this->Destroy();
       }
 wsprintf(buf,"COM%d:%d,%c,%d,%d",Prefs.port,Prefs.szybkosc,Prefs.parzystosc,
                                                  Prefs.bitydanych,Prefs.bitystopu);
 err = BuildCommDCB(buf,&s_dcb);
 if(err < 0){
       MessageBox("Błąd BulidCommDCB.","Błąd",MB_ICONSTOP);
       }
 err = SetCommState(&s_dcb);
 if(err < 0){
       MessageBox("Błąd SetCommState.","Błąd",MB_ICONSTOP);
       }
}

void TTransfDialog::ClosePort()
{
 err = CloseComm(idComDev);
 if(err != 0){
       MessageBox("Blad zamknięcia portu.","Błąd",MB_ICONSTOP);
       }
}

void TTransfDialog::Wysylaj(ULONG IloscB)
{
 int BWrite,b,kB;
 float f;
 ULONG k,l,lkB;
 BYTE* tBuforB;
 tBuforB = BuforB;
 ULONG cbWrite=IloscB;

 Status->SetText("Programowanie rejestratora.");

 b=0;  kB = 0;  l=1024;
 while(cbWrite != 0){
  if(cbWrite > 12288){
         BWrite=12288;
         cbWrite=cbWrite-12288;}
        else{
         BWrite=(ULONG)cbWrite;
         cbWrite=0;}

       err = WriteComm(idComDev, tBuforB, BWrite);
       err = GetCommError(idComDev, &s_comstat);  GCError(err);
```

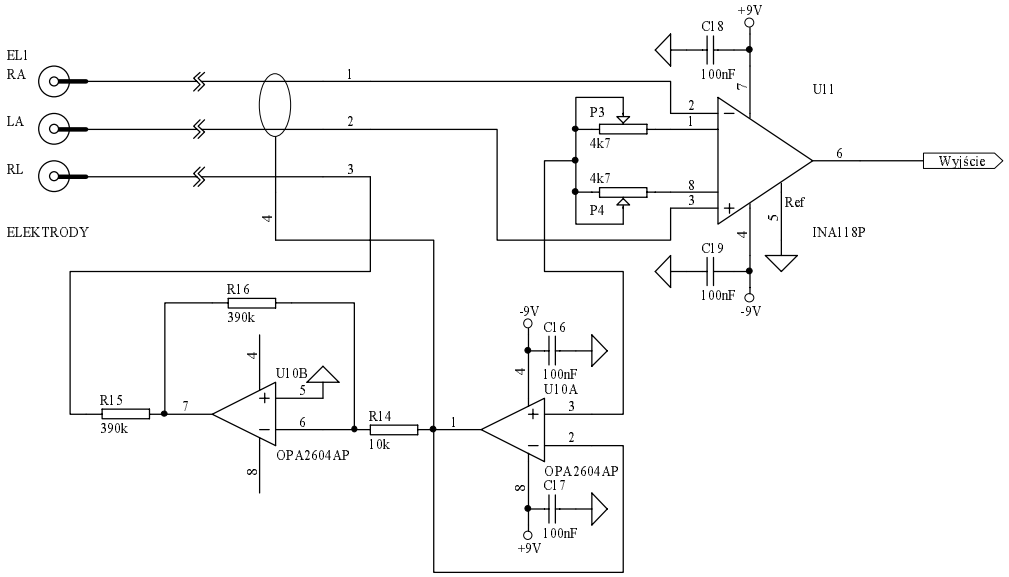
```
      if(cbWrite != 0)
            for(int i=0; i<12288; i++){
                  tBuforB++;
                  }

      while(s_comstat.cbOutQue > 0){
            err = GetCommError(idComDev, &s_comstat); GCError(err);
                  k=(ULONG)((ULONG)b*(ULONG)12288+(ULONG)((ULONG)BWrite-
                                    (ULONG)s_comstat.cbOutQue));
                  if( k >= l ){
                        l=l+1024;
                        kB++;
                        lkB=IloscB/1024;
                        f=(float)(kB*100)/lkB;
                        sprintf(buf,"Wysłane %3d kB, %.0f %%",kB,f);
                        Status->SetText(buf);
                        }
             }
             b++;
        }
        sprintf(buf,"Programowanie zakończone.");
        Status->SetText(buf);
}

void TTransfDialog::GCError(int err)
{
char terr[100];
if(err != 0){
      wsprintf(terr,"?");
      if(err == CE_BREAK){wsprintf(terr,"Hardware detected a break
                                                condition.");}
      if(err == CE_CTSTO){wsprintf(terr,"CTS (clear-to-send) timeout.");}
      if(err == CE_DSRTO){wsprintf(terr,"DSR (data-set-ready) timeout.");}
      if(err == CE_FRAME){wsprintf(terr,"Hardware detected a framing error.");}
      if(err == CE_OVERRUN){wsprintf(terr,"The character was lost.");}
      if(err == CE_RXOVER){wsprintf(terr,"Receiving queue overflowed.");}
      if(err == CE_RXPARITY){wsprintf(terr,"Hardware detected a parity
                                                error.");}
      if(err == CE_TXFULL){wsprintf(terr,"Transmission queue was full");}

      wsprintf(buf,"Błąd transmisji.\n%s \n%d",terr,err);
      MessageBox(buf,"Błąd GetCommError.",MB_ICONSTOP);
      }
}

void TTransfDialog::Odbieraj(ULONG IloscB)
{
 int kB;
 float f;
 ULONG lkB,BRead,suma;
 BYTE* tBuforB;
 tBuforB = BuforB;
 ULONG cbRead=IloscB;

 wsprintf(buf,"Odczytywanie rejestratora");
 SetCaption(buf);
 Status->SetText("Odczytywanie rejestratora.");

 suma=0;
 kB = 0;
 while(cbRead != 0){
  if(cbRead >= 1024){
      BRead=1024;
  }else{
      BRead=cbRead;
  }
      GetCommError(idComDev, &s_comstat);

      while( s_comstat.cbInQue < BRead ){
        GetCommError(idComDev, &s_comstat);
       }
        err = ReadComm(idComDev, tBuforB, 12288);
```

```
        suma=suma+err;
        cbRead=IloscB-suma;

        if(err < 0){
             err = GetCommError(idComDev, &s_comstat);
             GCError(err);
             err = 0;
             }

        kB++;
        lkB=IloscB/1024;
        f=(float)(kB*100)/lkB;
        sprintf(buf,"Odebrane %3d kB, %.0f %%",kB,f);
        Status->SetText(buf);

      if(cbRead != 0)
       for(int i=0; i<err; i++){
             tBuforB++;
       }
 }
 sprintf(buf,"Odczytywanie zakończone.");
 Status->SetText(buf);
 MessageBeep(MB_OK);
}

//----------------------------------------------------------------------------
//     Program rejestratora holterowskiego z pamięcią cyfrową
//     Plik dtransf.h
//----------------------------------------------------------------------------

#ifndef __DTRANSF_H
#define __DTRANSF_H

#include <owl\dialog.h>
#include "com232.h"  // struktura TRecordEKG

class TTransfDialog : public TDialog {
  public:
          TTransfDialog(  TWindow* parent,
                          int& tryb,
                          TPrefs& prefs,
                          TRecordEKG& record,
                          BYTE* buforB,
                          TModule* module = 0);

  private:
       void SetupWindow();
       void OpenPort();
       void ClosePort();
       void Wysylaj(ULONG cbWrite);
       void Odbieraj(ULONG cbWrite);
       void CmStart();
       void GCError(int err);

       int&         Tryb;
       TPrefs&      Prefs;
       TRecordEKG& Record;
       BYTE*        BuforB;

      protected:
       TStatic*    Status;
       TStatic*    Port;
       char        buf[100];
       int         idComDev,err;
       DCB         s_dcb;
       COMSTAT     s_comstat;

  DECLARE_RESPONSE_TABLE(TTransfDialog);
};

#endif
```

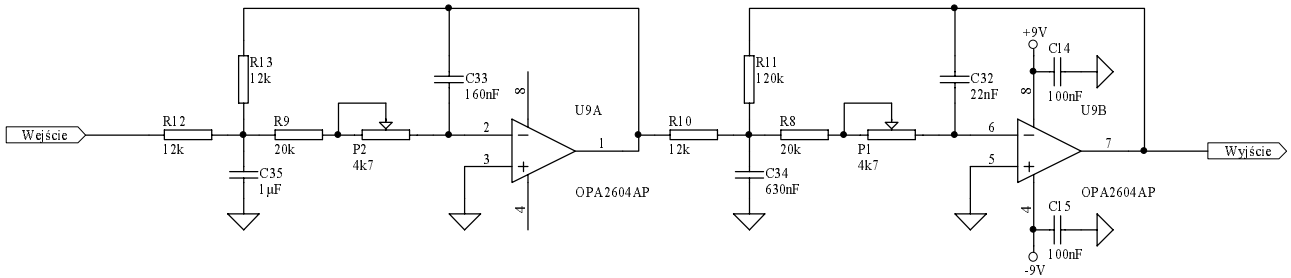
Dodatek C

Schematy ideowe urządzenia


Stopień wejściowy
Blok filtrów dolnoprzepustowych
Blok przetwornika A/C
Blok mikroprocesora AT89C52 z pamięcią cyfrową RAM
ELEKTRODY
RA
LA
RL
1
2
3
4
25 Hz
DAC
µP
Dane
Adres
RAM
PC
konwerter poziomów napięć
konwerter logiczny
wyświetlacz LCD
Blok interfejsu portu szeregowego RS-232
Blok klawiatury sterującej
Blok wyświetlacza alfanumerycznego LCD
Zasilacz
Blok zasilaczy układów analogowych i cyfrowych
Title
Schemat blokowy
Size
A4
Number
Rys. 1
Revision
Date:
21-Jul-2001
Sheet of
File:
C:\TMP\SBLOK.SCH
Drawn By:

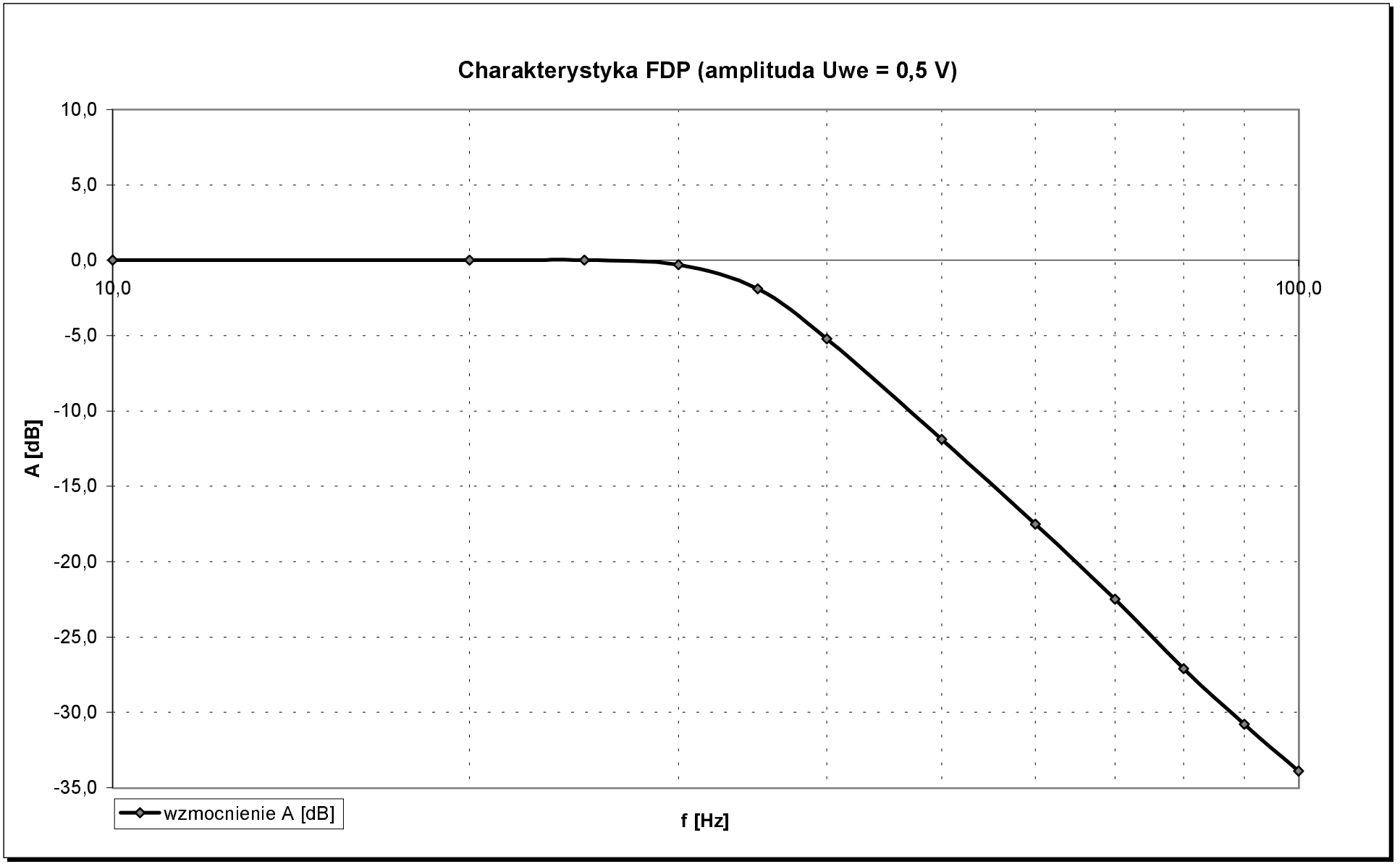


Title	**Schemat stopnia wejściowego**	
Size: A4	Number: Rys. 2	Revision:
Date: 21-Jul-2001		Sheet of
File: C:\TMP\STWEJ.SCH		Drawn By:

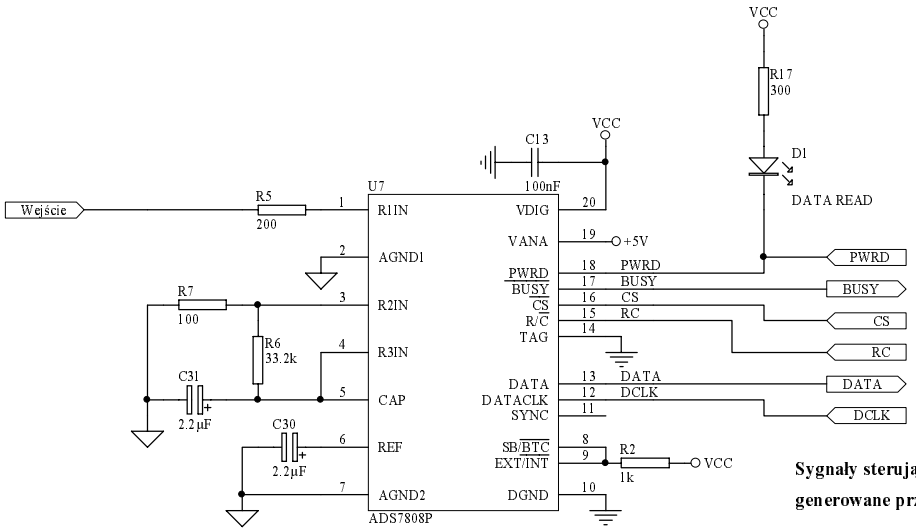


Częstotliwość graniczna fg = 25 Hz

Title: **Schemat bloku filtrów dolnoprzepustowych**		
Size: A4	Number: Rys. 3	Revision:
Date: 21-Jul-2001		Sheet of
File: C:\TMP\BLFDP.SCH		Drawn By:


Charakterystyka FDP (amplituda Uwe = 0,5 V)
10,0
5,0
0,0
-5,0
-10,0
-15,0
-20,0
-25,0
-30,0
-35,0
10,0
100,0
A [dB]
f [Hz]
wzmocnienie A [dB]

Rys. 4

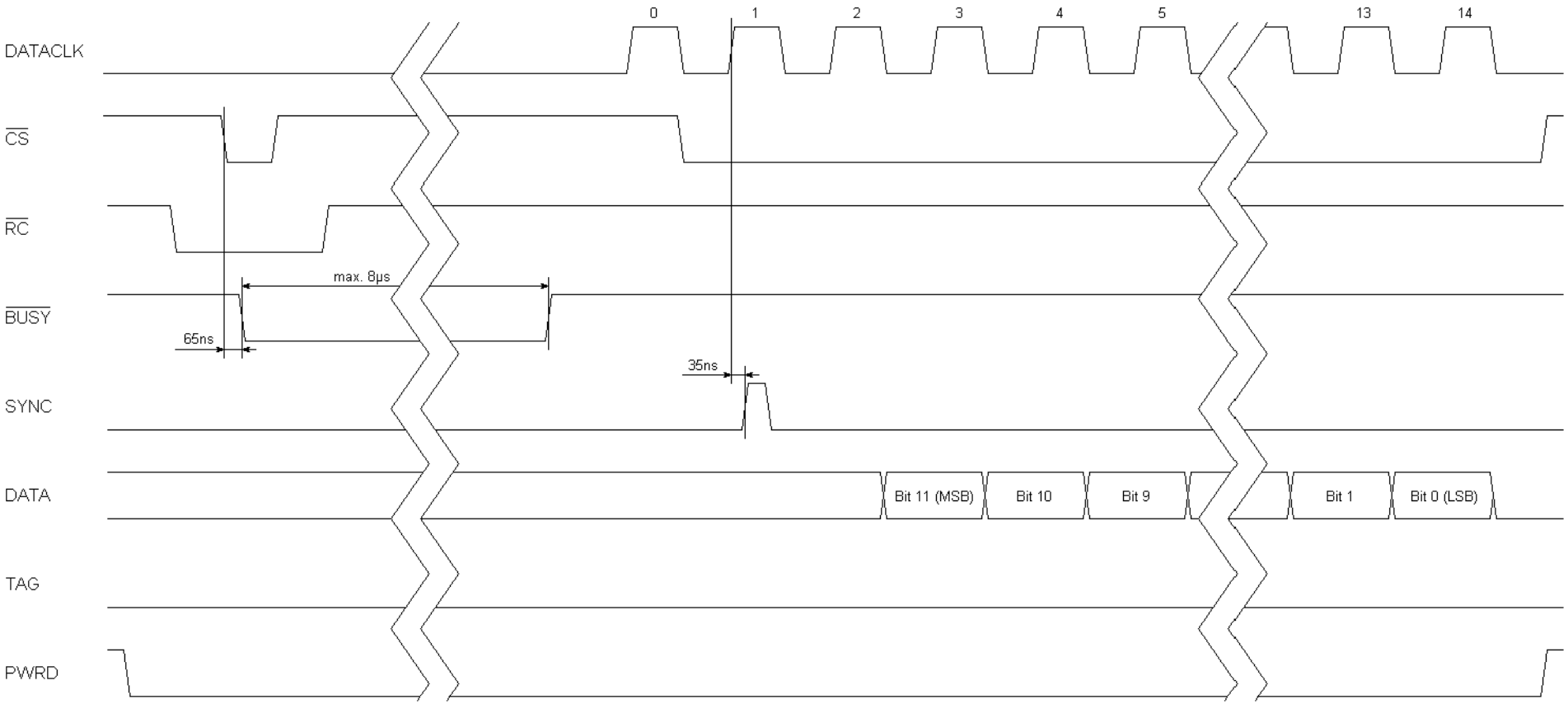


Sygnały sterujące generowane przez procesor

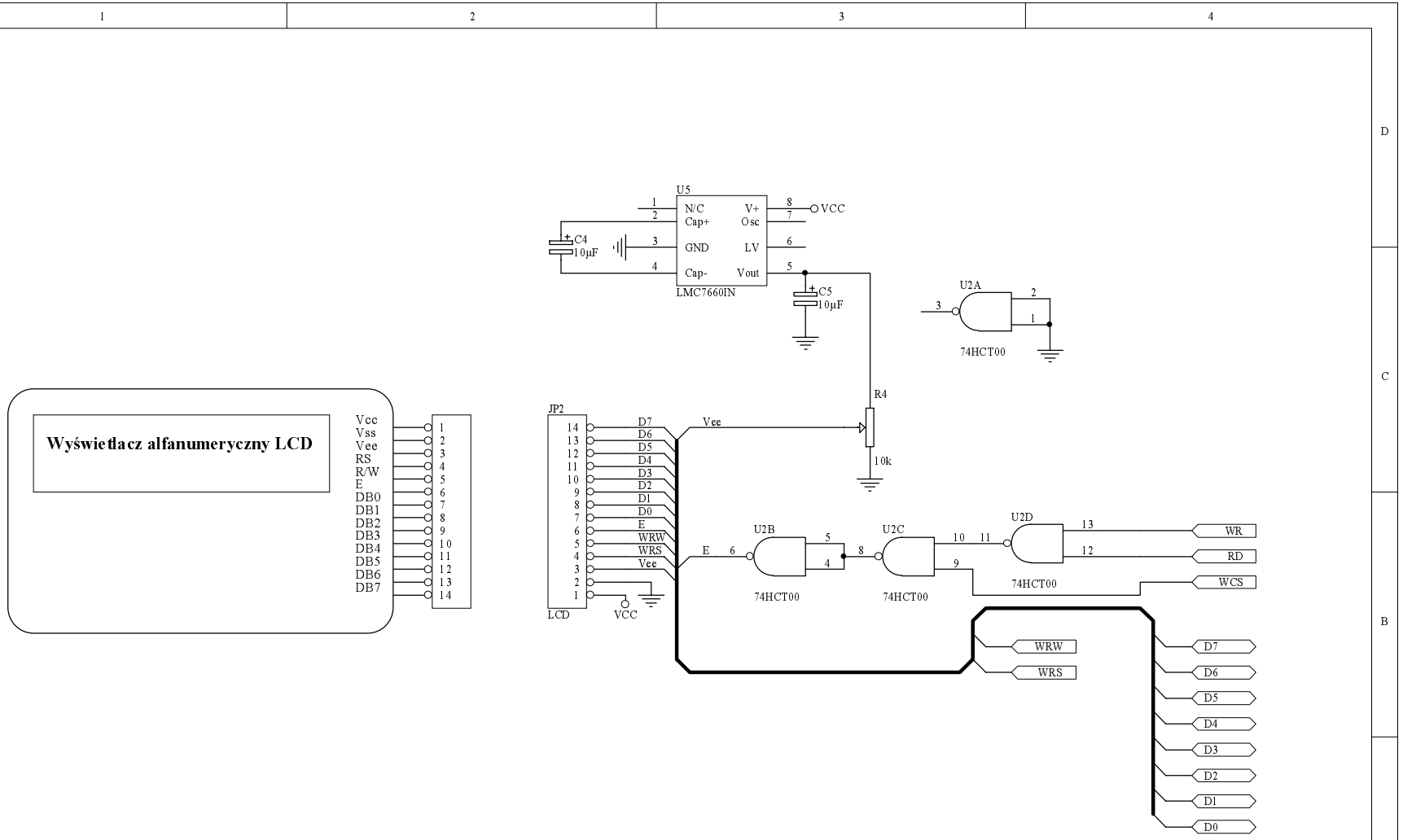
Dla elementów na schemacie zakres napięć wejściowych wynosi ± 10 V

Title: **Schemat bloku przetwornika A/C**		
Size: A4	Number: Rys. 5	Revision:
Date: 21-Jul-2001		Sheet of
File: C:\TMP\PRZAC.SCH		Drawn By:

Przebiegi czasowe sygnałów sterujących przetwornikiem A/C ADS7808



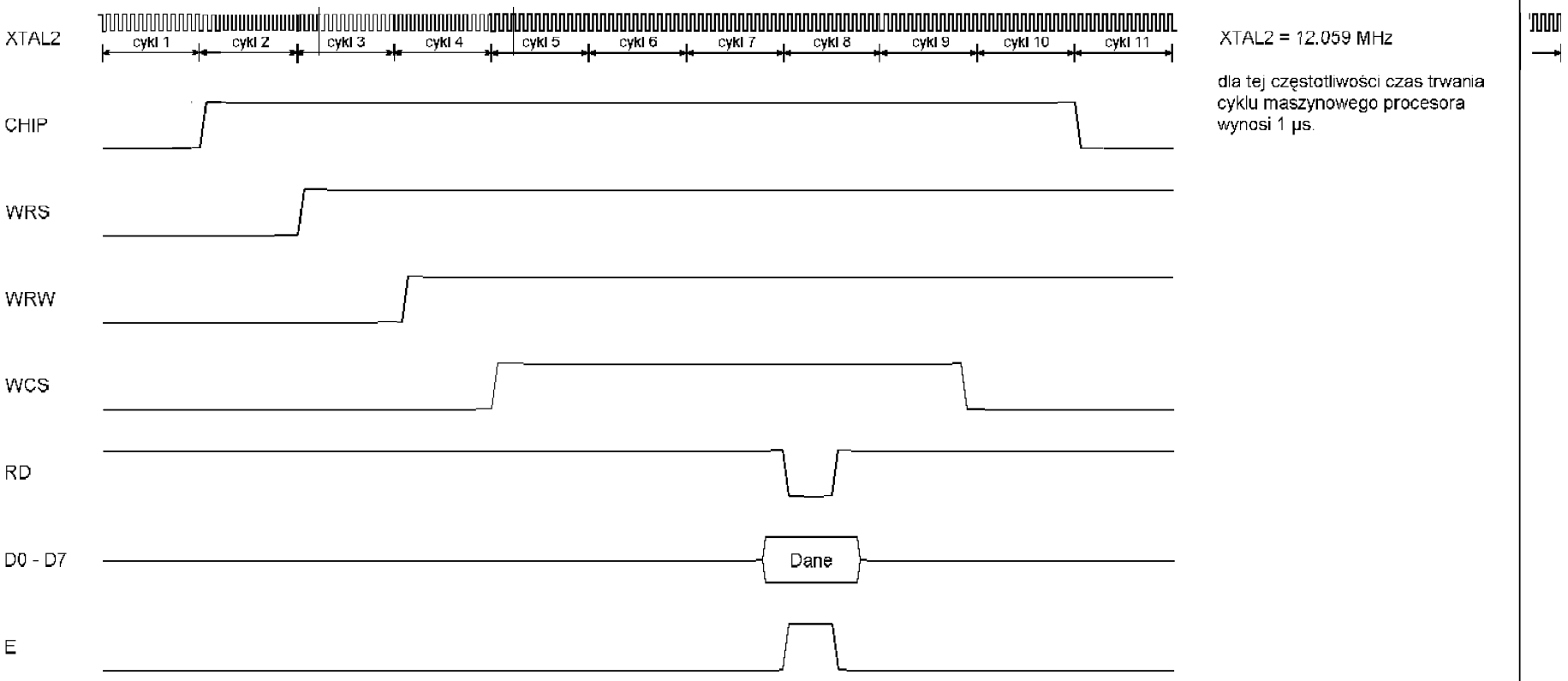
Rys. 6



Title	Schemat bloku wyświetlacza LCD		
Size A4	Number Rys. 7		Revision
Date: 21-Jul-2001		Sheet of	
File: C:\TMP\WLCD.SCH		Drawn By:	

Przebiegi czasowe sygnałów sterujących wyświetlaczem LCD z interfejsem HD44780

Odczyt danych z wyświetlacza



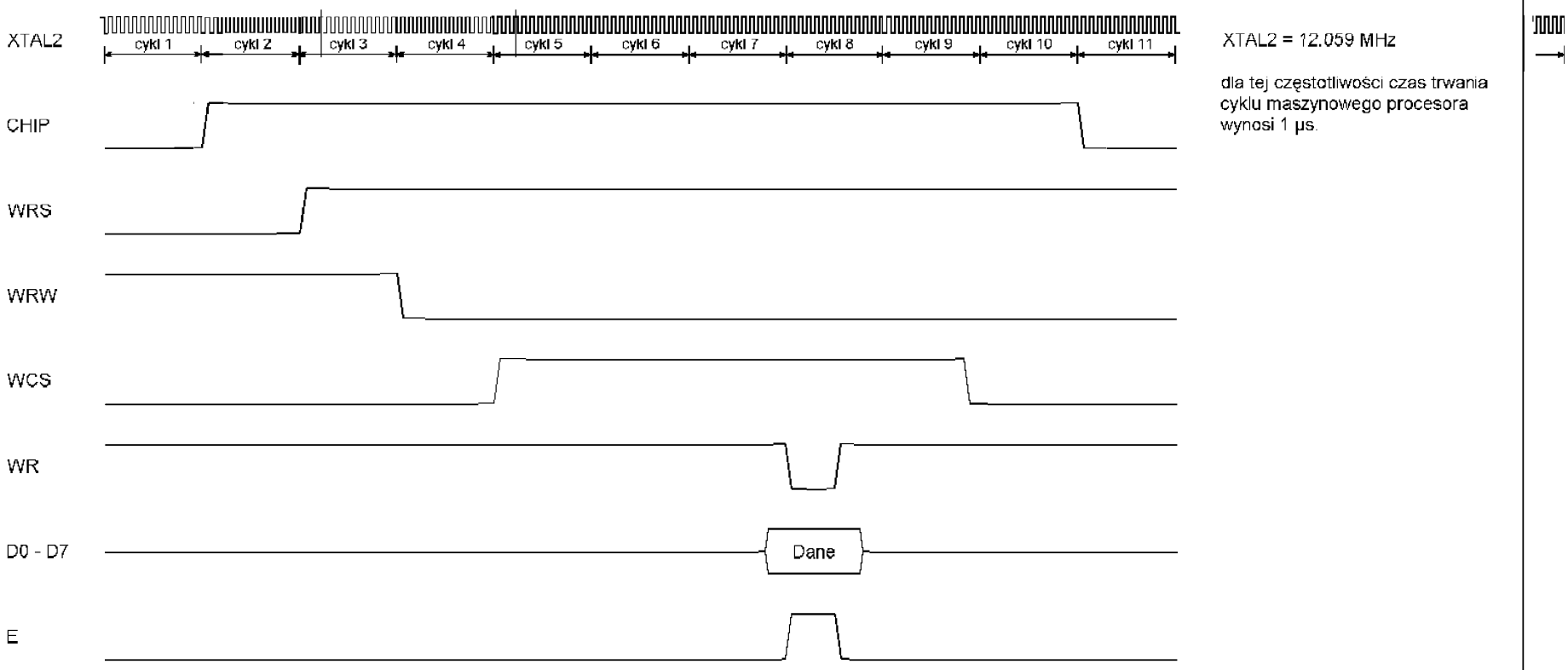
XTAL2 = 12.059 MHz

dla tej częstotliwości czas trwania cyklu maszynowego procesora wynosi 1 µs.

Rys. 8

Przebiegi czasowe sygnałów sterujących wyświetlaczem LCD z interfejsem HD44780

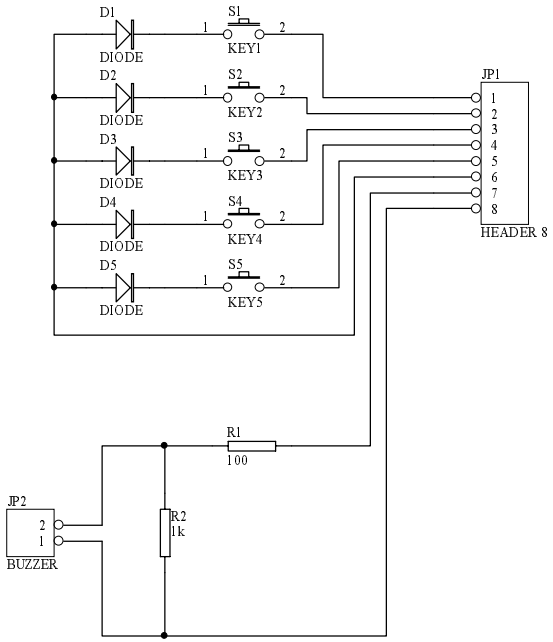
Zapis danych do wyświetlacza



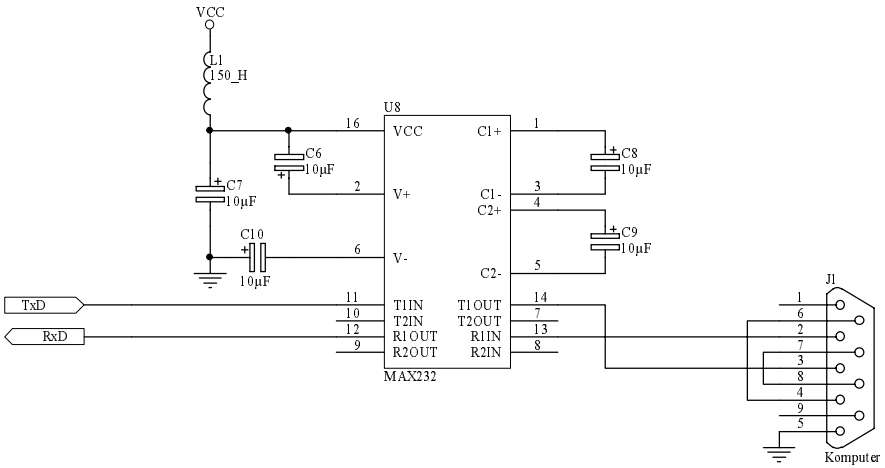
XTAL2 = 12.059 MHz

dla tej częstotliwości czas trwania cyklu maszynowego procesora wynosi 1 µs.

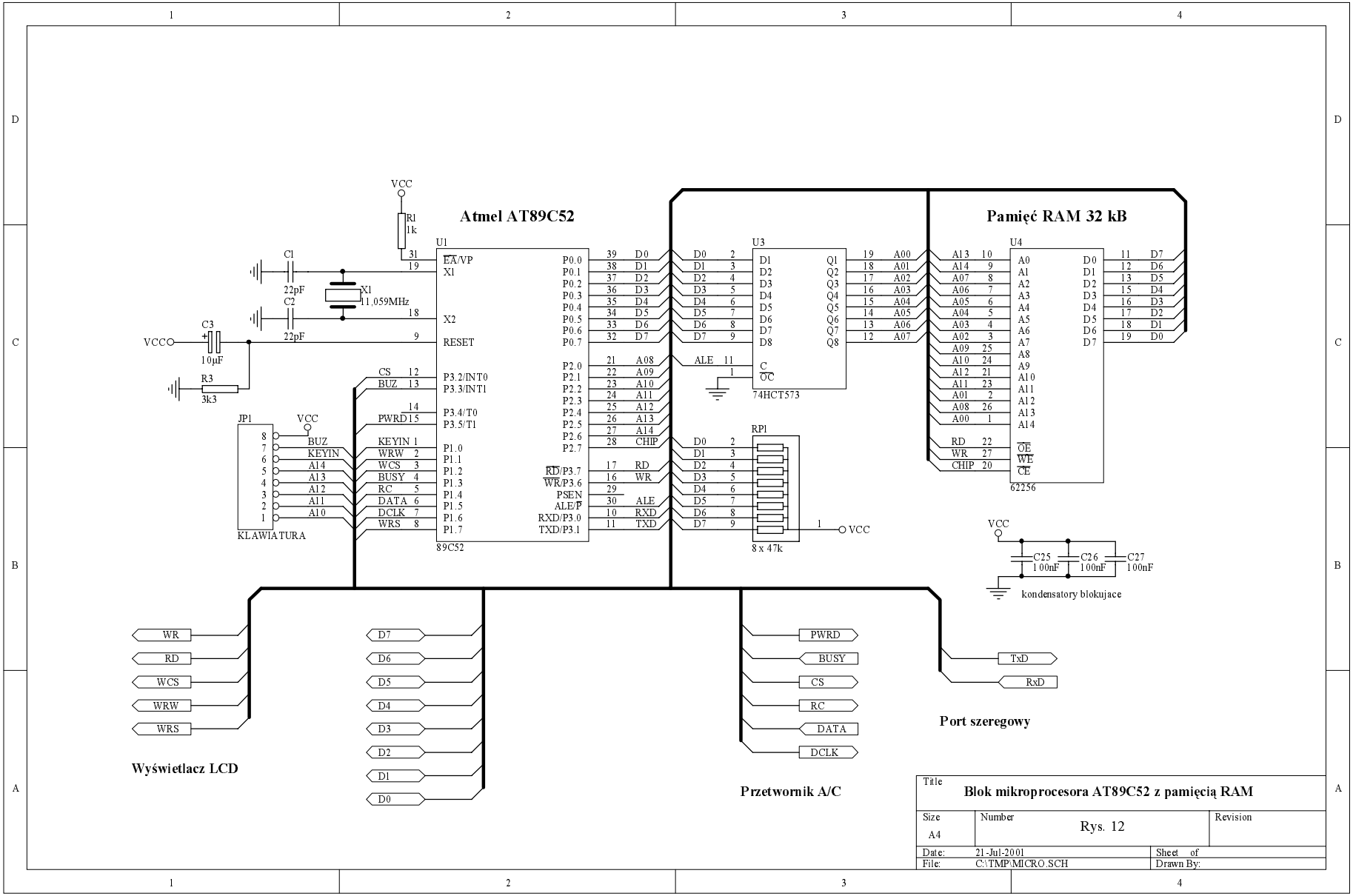
Rys. 9



Title: **Schemat bloku klawiatury sterującej**		
Size: A4	Number: Rys. 10	Revision:
Date: 21-Jul-2001		Sheet of
File: C:\TMP\KLAW.SCH		Drawn By:

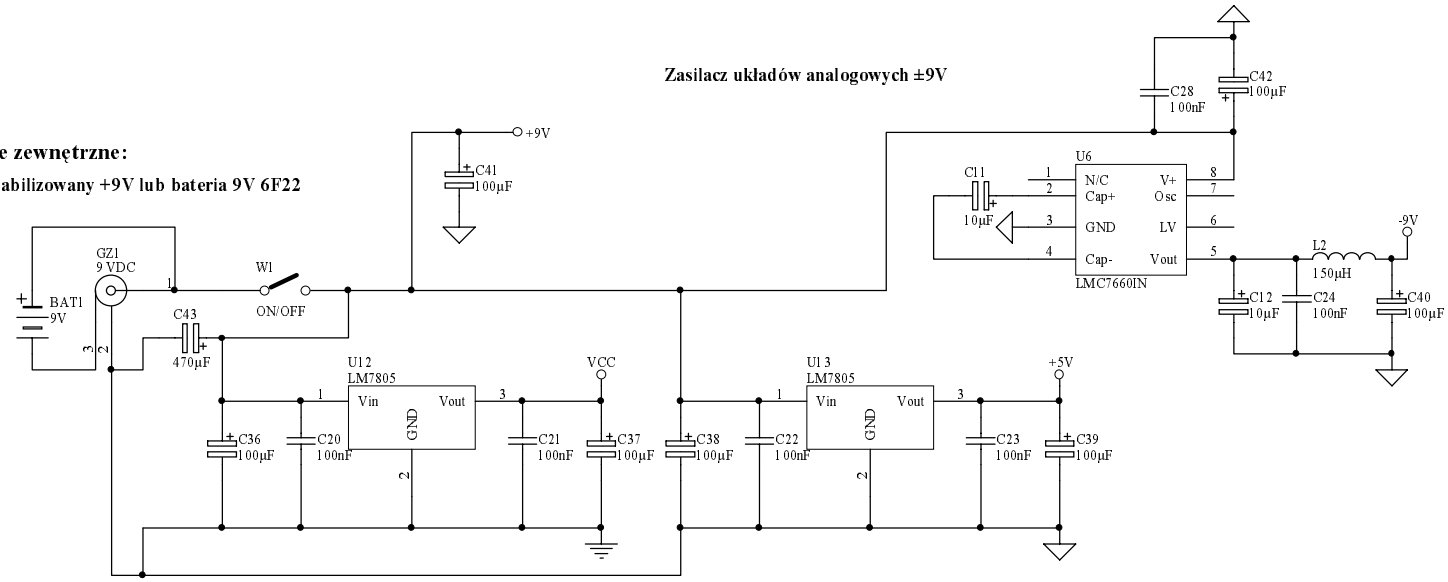


Title	**Schemat bloku interfejsu portu szeregowego RS-232**		
Size A4	Number Rys. 11		Revision
Date: 21-Jul-2001		Sheet of	
File: C:\TMP\PSZER.SCH		Drawn By:	

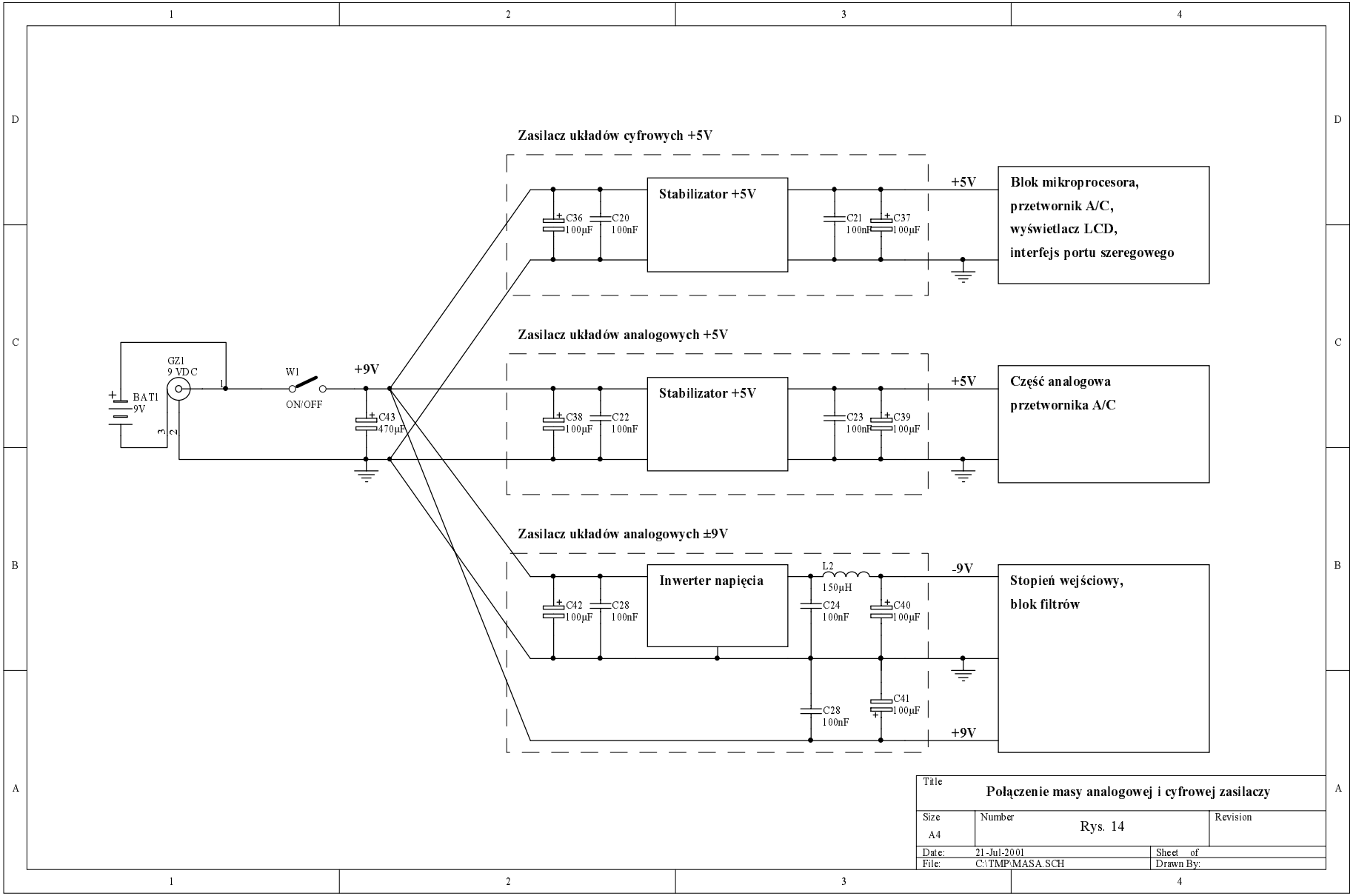

Atmel AT89C52
Pamięć RAM 32 kB
U1
89C52
U3
74HCT573
U4
62256
JP1
KLAWIATURA
RP1
8 x 47k
VCC
R1
1k
C1
22pF
C2
22pF
X1
11,059MHz
C3
10µF
R3
3k3
kondensatory blokujace
C25
100nF
C26
100nF
C27
100nF
WR
RD
WCS
WRW
WRS
Wyświetlacz LCD
D7
D6
D5
D4
D3
D2
D1
D0
PWRD
BUSY
CS
RC
DATA
DCLK
Przetwornik A/C
TxD
RxD
Port szeregowy
Title
Blok mikroprocesora AT89C52 z pamięcią RAM
Size
A4
Number
Rys. 12
Revision
Date:
21-Jul-2001
Sheet of
File:
C:\TMP\MICRO.SCH
Drawn By:

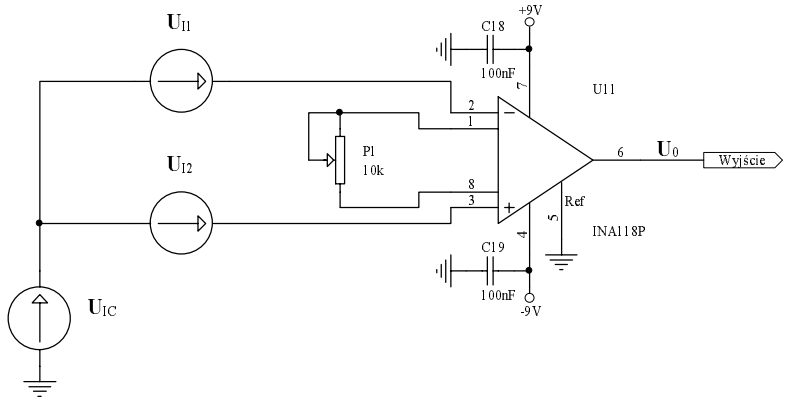
Zasilanie zewnętrzne:

zasilacz stabilizowany +9V lub bateria 9V 6F22

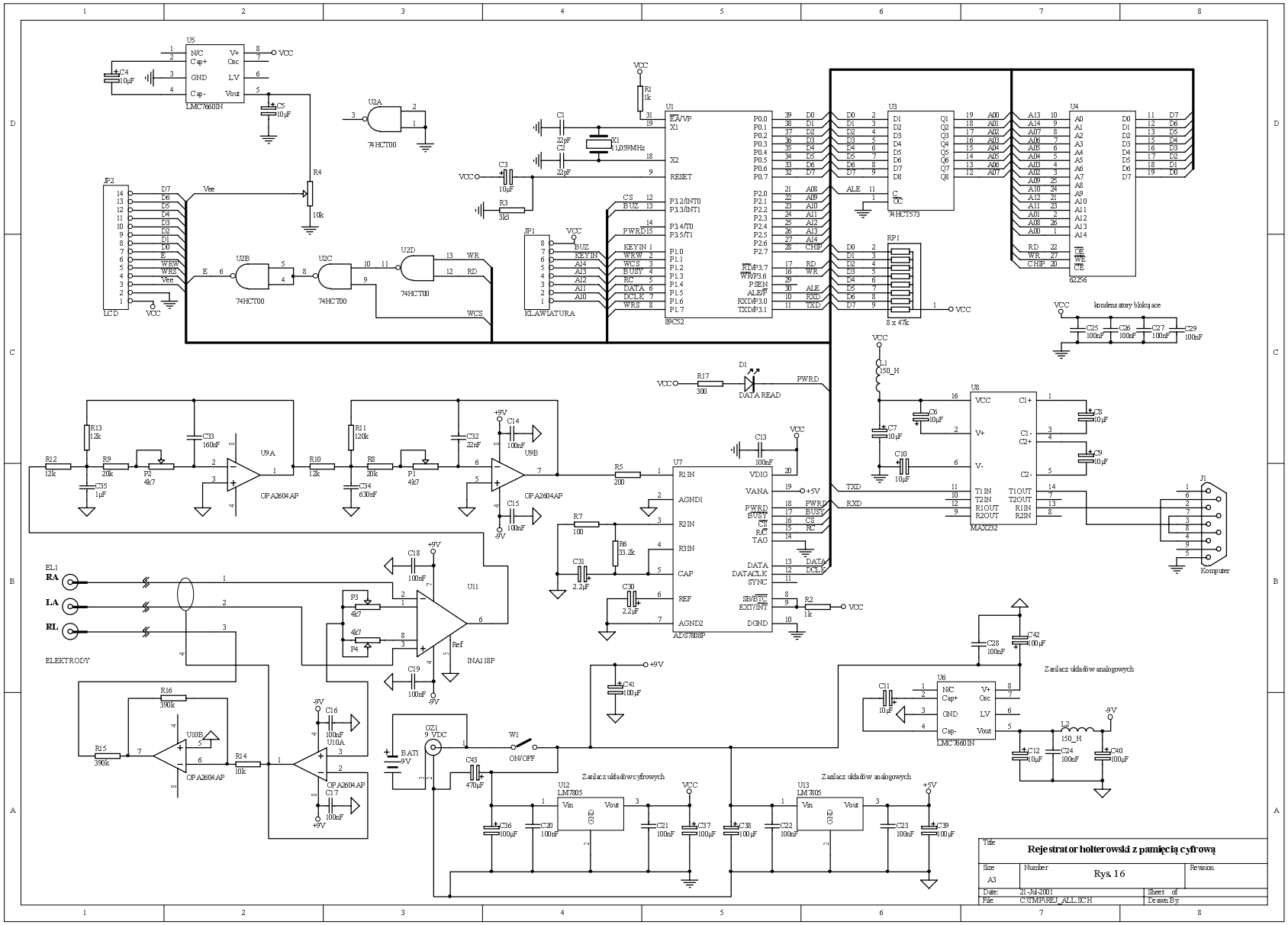


Title: **Blok zasilaczy układów cyfrowych i analogowych**		
Size: A4	Number: Rys. 13	Revision:
Date: 21-Jul-2001		Sheet of
File: C:\TMP\ZASIL.SCH		Drawn By:


Zasilacz układów cyfrowych +5V
Stabilizator +5V
C36 100µF
C20 100nF
C21 100nF
C37 100µF
+5V
Blok mikroprocesora,
przetwornik A/C,
wyświetlacz LCD,
interfejs portu szeregowego
Zasilacz układów analogowych +5V
Stabilizator +5V
C38 100µF
C22 100nF
C23 100nF
C39 100µF
+5V
Część analogowa
przetwornika A/C
Zasilacz układów analogowych ±9V
Inwerter napięcia
L2 150µH
C42 100µF
C28 100nF
C24 100nF
C40 100µF
C28 100nF
C41 100µF
-9V
+9V
Stopień wejściowy,
blok filtrów
GZ1
9 VDC
BAT1
9V
W1
ON/OFF
+9V
C43 470µF
Połączenie masy analogowej i cyfrowej zasilaczy
Rys. 14
A4
21-Jul-2001
C:\TMP\MASA.SCH



Title	**Wzmacniacz pomiarowy**	
Size: A4	Number: Rys. 15	Revision:
Date: 21-Jul-2001		Sheet of
File: C:\TMP\WZPOM.SCH		Drawn By:


U5
LMC7660IN
C4 10µF
C5 10µF
VCC
U2A
74HCT00
U1
89C52
X1 11,059MHz
C1 22pF
C2 22pF
C3 10µF
R1 1k
R3 5k3
R4 10k
JP2
LCD
JP1
KLAWIATURA
U2B
U2C
U2D
74HCT00
74HCT00
74HCT00
WR
RD
WCS
U3
74HCT573
U4
62256
RP1
8 x 47k
kondensatory blokujące
C25 100nF
C26 100nF
C27 100nF
C29 100nF
L1 150_H
D1
DATA READ
R17 300
PWRD
U8
MAX232
C6 10µF
C7 10µF
C8 10µF
C9 10µF
C10 10µF
TXD
RXD
J1
Komputer
U9A
U9B
OPA2604AP
R13 12k
R12 12k
R9 20k
P2 4k7
C33 160nF
C35 1µF
R10 12k
R11 120k
R8 20k
P1 4k7
C34 630nF
C32 22nF
C14 100nF
C15 100nF
+9V
-9V
U7
ADS7808P
C13 100nF
R5 200
R7 100
R6 33,2k
C31 2,2µF
C30 2,2µF
R2 1k
+5V
EL1
RA
LA
RL
ELEKTRODY
U11
INA118P
C18 100nF
C19 100nF
P3 4k7
P4 4k7
Ref
R16 390k
R15 390k
R14 10k
U10B
U10A
C16 100nF
C17 100nF
GZ1 9 VDC
BAT1 9V
W1 ON/OFF
C43 470µF
C41 100nF
U12 LM7805
U13 LM7805
Zasilacz układów cyfrowych
Zasilacz układów analogowych
C36 100µF
C20 100nF
C21 100nF
C37 100µF
C38 100µF
C22 100nF
C23 100nF
C39 100µF
C28 100nF
C42 100µF
C11 10µF
C12 10µF
C24 100nF
C40 100µF
L2 150_H
U6
LMC7660IN
Zasilacz układów analogowych
Title
Rejestrator holterowski z pamięcią cyfrową
Size
A3
Number
Rys. 16
Revision
Date: 21-Jul-2001
Sheet of
File: C:\TMP\REJ_ALL.SCH
Drawn By

Dodatek D

Karty katalogowe układów scalonych

INA118
OPA2604
ADS7808
AT89C52
LMC7660

Uwaga, karty katalogowe układów scalonych zostały usunięte z pliku pdf w celu ograniczenia jego objętości
P. Augustyniak