EGMONT INSTRUMENTS

Informacji na temat kart produkowanych dawniej przez Ambex udziela firma Egmont Instruments.

Pod adresem **http://www.ambex.com.pl** powstaje archiwum instrukcji i oprogramowania do kart komputerowych produkowanych dawniej przez Ambex, a obecnie jeszcze w części oferty przez Egmont Instruments. Archiwum to jest systematycznie uzupełniane. Jeśli poszukują Państwo informacji do starych lub aktualnych wyrobów, prosimy kierować się właśnie pod powyższy adres w Internecie. Jeśli nie znajdą tam Państwo potrzebnej informacji, prosimy o bezpośredni kontakt z nami.

Strony http://www.ambex.com.pl są prowadzone bezpośrednio przez firmę Egmont Instruments.

EGMONT INSTRUMENTS	**tel. (0-22) 850-62-05, 850-64-30**
00-867 Warszawa, ul. Chłodna 39, pawilon 11	**fax (0-22) 654-02-48**
http://www.egmont.com.pl	**e-mail egmont@egmont.com.pl**
http://www.ambex.com.pl	**e-mail ambex@ambex.com.pl**

EGMONT INSTRUMENTS

D O K U M E N T A C J A D R I V E R A

M O D U Ł U K O N T R O L N O - P O M I A R O W E G O

L C - 0 1 2 - 1 6 1 2

Wersja: październik 1997

EGMONT INSTRUMENTS
02-304 Warszawa , Aleje Jerozolimskie 141/90
http://www.egmont.com.pl
tel. (0-22) 823-30-17, 668-69-75
fax (0-22) 659-26-11
e-mail egmont@egmont.com.pl

SPIS TREŚCI

1. Informacje ogólne.

Przy tworzeniu oprogramowania modułów przyjęto zasadę, że cała komunikacja z modułem prowadzona jest za pośrednictwem rezydentnego programu dostępnego dla programów użytkowych poprzez przerwanie programowe. Takie rozwiązanie ma następujące zalety:
- użytkownik jest zwolniony ze znajomości szczegółów technicznych tak modułu jak i używanego komputera,
- rozwiązanie to jest niezależne od używanej implementacji języka wyższego poziomu.

Program obsługi został napisany w standardzie driver'ów systemu operacyjnego MS-DOS (wersja 3.1. i wyższe). Główną przyczyną wyboru takiego rozwiązania jest umożliwienie prostego badania obecności driver'a w systemie. Jedyną wykorzystywaną standardową funkcją driver'a jest funkcja inicjalizacji wykonywana w trakcie ładowania systemu. Po zainstalowaniu driver służy tylko jako obsługa danego przerwania programowego.

Driver'y modułów serii LC-011 i LC-020 podporządkowane zostały standardowi przyjętemu przez firmę. W standardzie tym przyjęto, że w komputerze może być zainstalowanych kilka (do czterech) modułów danego rodzaju obsługiwanych przez jeden driver. Ponieważ standard ten został zaprojektowany do różnych typów modułów, niektóre parametry wydawać się mogą nadmiarowe.

2. Instalacja driver'a.

Instalacji driver'a dokonuje się za pomocą programu instalacyjnego - INSTALL.
Przed instalacją należy skopiować:
- zbiór LC*.DRV (odpowiedni dla rodzaju komputera oraz modułu),
- program instalacyjny INSTALL.EXE,

do wybranego katalogu na dysku twardym, bądź, w przypadku pracy z komputerem nie posiadającym dysku twardego, na dyskietkę systemową.

Po uruchomieniu programu instalacyjnego zleceniem INSTALL wybieramy opcję instalacji: dysk twardy bądź dyskietka (musi być to dyskietka systemowa). Następnie użytkownik powinien ustalić konfigurację modułów danego typu (jedorazowy przebieg programu umożliwia instalację driver'a dla modułów tylko jednego typu). Bieżący parametr wskazywany jest podświetlonym paskiem. Zmiana wartości parametru następuje zawsze po wskazaniu go i naciśnięciu klawisza Enter. W zależności od typu parametru następuje wówczas nadanie parametrowi kolejnej wartości z listy wartości dopuszczalnych lub na ekranie pojawia się ramka, w której użytkownik powinien wpisać nową wartość. W tym drugim przypadku zaakceptowanie nowej wartości następuje po naciśnięciu klawisza Enter, zaś jej odrzucenie - po naciśnięciu klawisz Esc: parametr zachowuje wówczas swoją poprzednią wartość.

Program instalacyjny INSTALL jest wspólny dla wszystkich rodzajów modułów analogowych produkowanych przez firmę AMBEX. Na ekranie pojawiają się tylko parametry istotne dla wybranego typu modułu; w zależności od typu modułu są one w różny sposób ograniczone. Poniżej podany zostanie skrótowy opis wszystkich parametrów.

Parametry ogólne:
a) typ modułu: LC-010-1612, LC-011-1612, LC-020-0812, LC-020-3212, LC-030-1612;
b) liczba zainstalowanych modułów danego typu: od 1 do 4 8);
c) typ komputera: XT, AT1);
d) pełna nazwa ścieżki do katalogu, gdzie skopiowano zbiór LC*.DRV (łącznie z nazwą napędu dyskowego);
e) wybór trybu pracy: z pamięcią rozszerzoną lub bez 2) 3);
f) adres pamięci rozszerzonej w postaci szesnastkowej;
g) rozmiar pamięci rozszerzonej w kilobajtach;
h) maksymalny czas obsługi przerwania, które może mieć miejsce w trakcie trwania długich pomiarów - podawany w mikrosekundach (dodatnia liczba mieszcząca się na 16 bitach) 3) 8).

Parametry dotyczące jednego modułu 4):
i) nazwa modułu: od A do D (nazwa związana jest z adresem bazowym modułu);
j) tryb pracy modułu: master lub slave (istotne przy współpracy kilku modułow) 8);
k) częstotliwość zegara modułu: 4 lub 8 MHz 8);
l) zakres napięć dla wejść analogowych;
m) liczba kanałów wejść analogowych 5);
n) wyposażenie modułu we wzmacniacz Sample&Hold 6);
o) zakres napięć dla wyjścia analogowego - kanał 1 7);
p) zakres napięć dla wyjścia analogowego - kanał 2 3) 7) 8).
1) dla modułu LC-030-1612 - tylko AT.
2) nie dotyczy pracy z komputerem typu XT.
3) nie dotyczy pracy z modułem LC-010-1612.
4) przy instalacji kilku modułów jednego typu parametry dotyczące każdego z modułów różnią się między sobą np. moduły muszą różnić się nazwą.
5) moduł LC-010-1612: liczba kanałów zależna jest od tego czy na module zainstalowano multiplekser i wynosi 16 bądź 1,
moduły LC-011-1612 i LC-030-1612: liczba kanałów jest stała i wynosi 16,
moduły LC-020-0812 i LC-020-3212: liczba kanałów jest zależna od liczby zainstalowanych wzmacniaczy Sample&Hold
6) dotyczy tylko modułu LC-010-1612.
7) nie dotyczy pracy z modułem LC-020-3212.
8) nie dotyczy pracy z modułem LC-030-1612.

Po wprowadzeniu żądanych parametrów pracy modułu (modułów) należy uruchomić funkcję "instalacja" (F10). Program modyfikuje zbiór CONFIG.SYS lub

w przypadku jego braku - tworzy zbiór o tej nazwie. Następnie program inicjalizuje system operacyjny (reboot) i sprawdza szybkość przetwornika a/c dla każdego z instalowanych modułów. Po teście szybkości system jest ponownie ładowany, co kończy instalację. W przypadku nieumieszczenia modułu w komputerze, bądź jego wadliwej pracy, instalacja nie jest wykonana a użytkownik jest o tym informowany stosownym komunikatem.

Uwaga: podłączenie modułu do przerwania sprzętowego oraz numer tego przerwania wykrywane są automatycznie przez driver w momencie ładowania systemu. To samo dotyczy kanałów DMA.

3. Opis driver'a.

3.1. Informacje wstępne.

Przed driver'em postawione zostały następujące zadania:
- pełne wykorzystanie możliwości sprzętowych oferowanych przez obsługiwane moduły
- rozszerzenie w sposób programowy możliwości modułów o funkcje, które nie są lub nie mogą być realizowane sprzętowo; do funkcji takich należą:
- różnorodne warunki startu operacji wejścia / wyjścia (odczyt / zapis portów cyfrowych, przetwarzanie a/c i c/a); w grę wchodzi tu warunkowanie startu operacji sygnałami cyfrowymi (poziomem, zboczem, kombinacją sygnałów), oraz upływem czasu rzeczywistego (data i odcinek czasu);
- długie transmisje DMA - przekraczające 64kB; jak wiadomo, w komputerach klasy IBM PC układy DMA są 16-bitowe i nie są połączone bezpośrednio z tzw. rejestrem strony rozszerzającym adres transmisji do 20 (XT) lub 24 (AT) bitów; powoduje to, że jeżeli w trakcie transmisji adres transmisji ma przejść przez granicę 64kB to wymaga to śledzenia transmisji na bieżąco i odpowiedniego przeprogramowywania układów DMA; porcja danych, którą można w danej sytuacji przesłać za pomocą pojedynczej transmisji będzie dalej nazywana blokiem transmisji DMA;
- realizacja pewnych funkcji, dzięki którym możliwe jest pisanie uniwersalnych programów, niezależnych od instalacji konkretnego modułu

Driver rozpoczyna swoją pracę już w momencie ładowania systemu. Wówczas to pobiera i analizuje parametry instalacji podane w poleceniu "DEVICE=..." z pliku CONFIG.SYS - dzięki tym informacjom możliwe jest pisanie programów niezwiązanych z konkretną instalacją modułu. Następnie wykonywane jest tzw. twarde zerowanie wszystkich zadeklarowanych modułów, w trakcie którego wykonywane jest wstępne programowanie - m.in. ustawianie wyjść analogowych na 0V a wyjść cyfrowych - na 1. Następnie wykonywane są testy,

których celem jest sprawdzenie czy i do jakich przerwań i kanałów DMA podłączony jest każdy z zainstalowanych modułów.

Driver'y modułów analogowych produkowanych przez firmę AMBEX zostały zaprojektowane w sposób jednolity. Dzięki temu, jeżeli tylko program nie korzysta jawnie z cech czy funkcji modułu specyficznych tylko dla niego, to może być bez zmiany wykorzystywany do współpracy z różnymi typami modułów. Oczywiście z powodu takich założeń pewne funkcje czy tryby pracy driver'a są dostępne dla jednych typów modułów, dla innych - nie.

3.2. Standardy oznaczeń, numeracja, dane charakterystyczne.

Przy pisaniu tak oprogramowania jak i niniejszej dokumentacji przyjęto następujące zasady:
- wszystkie nazwy pól rekordów, stałych itp. (z wyjątkiem nazw funkcji) opatrzone są przedrostkiem LC0_;
- nazwy występujące w dokumentacji są identyczne z nazwami występującymi w plikach źródłowych dla języków C, Pascal, assembler (z dokładnością do rozróżnienie małe / duże litery).

Numery przerwań programowych związanych z driver'ami poszczególnych typów modułów:

typ modułu	nazwa przerwania	numer przerwania	
		szesnastkowy	dziesiętny
LC-011-1612	LC011_16	91	145
LC-020-0812	LC020_08_2	94	148
LC-020-3212	LC020_32	95	149

Wszystkie driver'y widziane są w systemie DOS jako urządzenia. Nazwy tych urządzeń są następujące:

typ modułu	nazwa driver'a	nazwa urządzenia
LC-011-1612	LC1116x.DRV	LC1116^^
LC-020-0812	LC2008x.DRV	LC2008^2
LC-020-3212	LC2032x.DRV	LC2032^^

Kodowanie numerów modułów:

moduł	nazwa kodu	wartość
A	LC0_MODA	1
B	LC0_MODB	2
C	LC0_MODC	3
D	LC0_MODD	4

Kodowanie typów urządzeń w modułach:

urządzenie	nazwa kodu	wartość
porty cyfrowe wejściowe	LC0_DINPUT	1
porty cyfrowe wyjściowe	LC0_DOUTPUT	2
przetworniki a/c	LC0_AINPUT	3
przetworniki c/a	LC0_AOUTPUT	4
kanały CTC	LC0_CTC	5

Wszystkie wejścia, wyjścia, kanały itp. numerowane są od 1.
Długości buforów, pomiarów, marginesów itp. podawane są zawsze w próbkach.

3.3. Komunikacja z driver'em.

Funkcje driver'a wywoływane są poprzez przerwanie programowe (numery przerwań - patrz p. 3.2.). Przesyłanie informacji pomiędzy programem użytkowym a driver'em odbywa się poprzez rekord opisu zlecenia, którego adres przekazywany jest w rejestrach DX:DI. Rekord ten służy do przekazywania informacji zarówno do jak i od driver'a.

Rekord opisu zlecenia ma strukturę zależną od rodzaju zlecenia. Jedynie trzy pierwsze pola są niezmienne i mają następujące znaczenie:
adres rekordu: DX:DI

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach

LC0_CODE określa funkcję jaka ma być wykonana przez driver a zarazem sposób interpretacji ciągu bajtów znajdujących się pod adresem DX:DI.

LC0_STATUS informuje program wywołujący driver o poprawności wykonania funkcji:

LC0_STATUS = 0: wykonanie poprawne
LC0_STATUS < 0: wykonanie błędne
LC0_STATUS > 0: wykonanie poprawne z zastrzeżeniami (ostrzeżenia)

LC0_ERR_STAT (jeżeli jest mniejszy od zera) niesie pewne dodatkowe informacje komentujące zwrócony w LC0_STATUS kod błędu. Dotyczy to dwóch sytuacji:
- błędnie podane parametry warunku startu / stopu (LC0_STATUS = LC0_ILL_START / LC0_ILL_STOP); LC0_ERR_STAT precyzuje co zostało podane błędnie
- funkcja przerwana wywołaniem funkcji BREAK (LC0_STATUS = LC0_BROKEN); LC0_ERR_STAT określa wtedy, czy funkcja została przerwana w trakcie oczekiwania na spełnienie warunku startu czy już w trakcie przetwarzania.

Istnieje jeden wyjątek od tej zasady: funkcja BREAK może zwrócić LC0_STATUS = 0 i LC0_ERR_STAT <> 0 (patrz opis funkcji, p.4.9).

W rozdziale 5 podano tabele wszystkich kodów zwracanych przez LC0_STATUS i LC0_ERR_STAT.

3.4. Typy warunków startu / zakończenia operacji.

W driverze zaimplementowano następujące startu operacji:
- start natychmiastowy: bez czekania na spełnienie jakichkolwiek warunków
- poziom sygnału cyfrowego: warunek jest spełniony, gdy sygnał cyfrowy ma zadaną wartość;
- zbocze sygnału cyfrowego: warunek jest spełniony, gdy sygnał cyfrowy zmieni swoją wartość w określony sposób;
- kombinacja sygnałów cyfrowych: warunek jest spełniony, gdy kombinacja sygnałów cyfrowych jest równa (różna) zadanej; w pierwszym przypadku wszystkie zadeklarowane sygnały muszą mieć zadany poziom, w drugim - wystarczy, że jeden z sygnałów ma poziom różny od zadanego;
- data: warunek jest spełniony, gdy bieżąca data zrówna się z zadaną;
- czas: warunek jest spełniony po upływie zadanego odcinka czasu.

Należy pamiętać o tym, że powyżej opisane warunki startu realizowane są programowo w związku z czym początek koniec pomiaru jest zawsze nieco opóźniony względem momentu spełnienia warunku. Opóźnienie to (rzędu mikrosekund) zależne jest od szybkości komputera.

Warunki startu związane z pomiarem czasu realizowane są w oparciu o czas systemowy. Z tego powodu czas podawany jest z dokładnością do sekundy.

Dla zakończenia operacji zdefiniowano następujące warunki:
- przetworzenie określonej liczby próbek;

Kodowanie typu warunku startu operacji:

kod typu /	parametry warunku

znaczenie	bajt 1	bajt 2	bajt 3	bajt 4	bajt 5
0 (LC0_SIMMED natychmiast	---	---	---	---	---
1 (LC0_SHARD) od sygnału sprzętowego	---	---	---	---	---
2 (LC0_SLEVEL) 1) od poziomu sygnału cyfrowego	numer modułu	numer portu	numer wejścia	poziom	---
3 (LC0_SSLOPE) 2) od zbocza sygnału cyfrowego	numer modułu	numer portu	numer wejścia	zbocze	---
4 (LC0_SDIG_EQ) 3) od kombinacji wejść cyfrowych warunek równości	numer modułu	numer portu	maska	wzorzec	---
5 (LC0_SDIG_NE) 3) od kombinacji wejść cyfrowych warunek nierówności	numer modułu	numer portu	maska	wzorzec	---
6 (LC0_STIME) po upłynięciu określonego czasu	odcinek czasu w sekundach				---
7 (LC0_SDATE) 4) o podanym czasie	sekunda	minuta	godzina	dzień miesiąca	---

1) "Poziom" wskazuje oczekiwany stan wejścia cyfrowego (0/1). Wszystkie wartości różne od zera traktowane są jak "1".
2) Oczekiwane zbocze wejścia cyfrowego kodowane jest następująco:
0 - zbocze opadające, 1 - zbocze narastające.
3) Bajt maski wskazuje, które bity portu wejściowego brane są pod uwagę przy badaniu warunku:
1 wskazuje bit badany, 0 - ignorowany.
4) "Dzień miesiąca" dotyczy bieżącego miesiąca. Jeżeli numer dnia jest mniejszy niż bieżący - następnego miesiąca.

Kodowanie typu warunku stopu (zatrzymania) operacji (kody podane szesnastkowo):

kod typu (szesnastkowo)/	parametry warunku				
znaczenie	bajt 1	bajt 2	bajt 3	bajt 4	bajt 5

00 (LC0_ZSAMPLES) po zmierzeniu bloku danych	liczba próbek do zmierzenia	---

4. Funkcje driver'a.

W poniższych rozdziałach opisano wszystkie funkcje driver'a. Każdy rozdział ma następującą strukturę:
- tabela zawierająca strukturę rekordu opisu zlecenia; w tabeli tej opisano każde pole rekordu w sposób następujący:
- nazwa pola; nazwa ta używana jest konsekwentnie w plikach źródłowych dotyczących języka C, Pascal i asemblera (patrz rozdział 6)
- rozmiar w bajtach; typ danej reprezentowanej przez to pole (np. czy jest to liczba ze znakiem czy bez) wynika ze znaczenia pola; w razie wątpliwości należy porównać z odpowiednim dla danego języka plikiem źródłowym deklarującym struktury danych (dodatki A, C i E)
- znaczenie
- przeznaczenie funkcji
- szczegółowy opis parametrów funkcji (pól rekordu opisu zlecenia); ten punkt został zamieszczony tylko wtedy, gdy uznano, że znaczenie parametru podane w tabeli jest niewystarczająco oczywiste
- ostrzeżenia; lista ostrzeżeń zwracanych przez funkcję w parametrze LC0_STATUS; jeżeli punkt ten nie występuje to oznacza to, że dana funkcja nie zwraca żadnych ostrzeżeń
- błędy; lista błędów zwracanych przez funkcję w parametrze LC0_STATUS; jeżeli punkt ten nie występuje to oznacza to, że dana funkcja nie zwraca żadnych błędów
- dodatkowe informacje o błędach; lista dodatkowych informacji zwracanych przez funkcję w parametrze LC0_ERR_STAT; jeżeli punkt ten nie występuje to oznacza to, że dana funkcja nie zwraca żadnych dodatkowych informacji (LC0_ERR_STAT zawsze równe LC0_E_OK: 0)

4.1. Inicjalizacja (MODULE_INIT).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 0
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_IMODULE	1	mapa modułów

Przeznaczenie:

Funkcja powoduje zainicjalizowanie pracy (zerowanie) wyspecyfikowanych modułów.
Moduły podlegające inicjalizacji specyfikuje się na poszczególnych bitach parametru LC0_IMODULE:

```
b8       b1
- - - - D C B A
0 0 0 0 x x x x
```

x = 1 - zeruj moduł, x = 0 - nie zeruj modułu

Dodatkową czynnością wykonywaną przez funkcję jest przechwycenie przerwania 1C16 (przerwanie usługowe, wywoływane przez procedurę obsługi przerwania zegarowego) aby w bezpieczny sposób odmierzać upływający czas astronomiczny. Po przechwyceniu przerwania następuje synchronizacja wewnętrznego zegara czasu rzeczywistego z zegarem komputera. Synchronizacja polega na przepisaniu do niego czasu i daty systemowej, pobranej za pomocą funkcji DOS'a (2C16 i 2A16). Z tego też powodu funkcji MODULE_INIT nie należy wykonywać w procedurach obsługi przerwania - może się to skończyć zawieszeniem pracy systemu operacyjnego.

UWAGA:

Inne programy wykorzystujące przerwanie 1C16 mają obowiązek - po przechwyceniu tego przerwania - wykonywać również poprzednią procedurę obsługi tego przerwania.

Po wykonaniu funkcji nadal są pamiętane parametry ostatniego przetwarzania a/c i c/a, w związku z czym po inicjalizacji modułu można wykonać przetwarzanie ze zgaszonym bitem trybu pracy LC0_MOD_NEWPAR (patrz opis funkcji ANALOG_INPUT i ANALOG_OUTPUT).

Ostrzeżenia (LC0_STATUS):
LC0_NON_EX_MOD - zażądano inicjalizacji nie istniejących modułów ale co najmniej 1 moduł został zainicjalizowany

Błędy (LC0_STATUS):
LC0_NO_MODULE - nie istnieje żaden z żądanych modułów

4.2. Informacja o konfiguracji ogólnej (GET_TOTAL_CONFIGURATION).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 1
LC0_STATUS	1	kod zakończenia funkcji = LC0_OK
LC0_ERR_STAT	1	dodatkowe informacje o błędach
parametry wyjściowe:		
LC0_TONF	1	konfiguracja modułów
LC0_TIAD	1	liczba dostępnych przetworników a/c
LC0_TIDA	1	liczba dostępnych przetworników c/a
LC0_TCTC	1	liczba dostępnych kanałów CTC
LC0_TIDI	1	liczba dostępnych portów we. cyfrowych
LC0_TIDO		liczba dostępnych portów wy. cyfrowych
LC0_TMEMA	4	adres bufora w pamięci rozszerzonej (adres absolutny)
LC0_TMEML	4	długość bufora w pamięci rozszerzonej (w próbkach)

Przeznaczenie:

Funkcja zwraca informację o sumarycznej konfiguracji wszystkich modułów danego typu. Bajt konfiguracji modułów ma następujący format:

```
b8       b1
D C B A D C B A
y y y y x x x x
```

x: x = 1 - moduł zainstalowany, x = 0 - modułu nie ma
y: y = 1 - moduł "master", y = 0 - moduł "slave" (dla modułów serii LC-011 i LC-020 wszystkie moduły są typu "master")

4.3. Informacja o konfiguracji modułu (GET_MODULE_INFORMATION).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 2
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_MMODULE		numer modułu
parametry wyjściowe:		
LC0_MBASE1	2	adres bazowy modułu - pakiet 1
LC0_MBASE2[1)]	2	adres bazowy modułu - pakiet 2
LC0_MIAD	1	liczba dostępnych przetworników a/c
LC0_MIDA	1	liczba dostępnych przetworników c/a
LC0_MCTC	1	liczba dostępnych kanałów CTC
LC0_MIDI	1	liczba dostępnych portów we. cyfrowych
LC0_MIDO	1	liczba dostępnych portów wy. cyfrowych
LC0_MCLOCK	2	częstotliwość zegara modułu w kHz; częstotliwość próbkowania tworzona jest przez podział 1/4 częstotliwości zegara modułu
LC0_MINT	1	numer przerwania (programowy)

1) dla modułów serii LC-011 i LC-020 - nieokreślone (moduły są jednopakietowe)

Przeznaczenie:
Funkcja zwraca informację o konfiguracji wyspecyfikowanego modułu.

Błędy (LC0_STATUS):
LC0_NO_MODULE - nie ma takiego modułu

4.4. Informacja o szczegółach technicznych (GET_INFO).

4.4.1. Wejścia cyfrowe.

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 3
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_GTYPE	1	rodzaj urządzenia = 1
LC0_GMODULE	1	numer modułu
LC0_GNUM	1	numer portu wejściowego
parametry wyjściowe:		
LC0_GCHAN	1	liczba bitów portu

Przeznaczenie:
Funkcja zwraca informację o liczbie bitów badanego portu wejściowego.

Błędy (LC0_STATUS):
LC0_NONEX_DEV - port wejściowy o tym numerze nie istnieje
LC0_NO_MODULE - nie ma takiego modułu

4.4.2. Wyjścia cyfrowe.

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 3
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_GTYPE	1	rodzaj urządzenia = 2
LC0_GMODULE	1	numer modułu
LC0_GNUM	1	numer portu wyjściowego
parametry wyjściowe:		
LC0_GCHAN	1	liczba bitów portu

Przeznaczenie:
Funkcja zwraca informację o liczbie bitów badanego portu wyjściowego.

Błędy (LC0_STATUS):
LC0_NONEX_DEV - port wyjściowy o tym numerze nie istnieje
LC0_NO_MODULE - nie ma takiego modułu

4.4.3. Wejścia analogowe.

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 3
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_GTYPE	1	rodzaj urządzenia = 3
LC0_GMODULE	1	numer modułu
LC0_GNUM	1	numer przetwornika
parametry wyjściowe:		
LC0_GCHAN	1	liczba kanałów przetwornika
LC0_GRES	1	liczba bitów przetwornika
LC0_GTIME	2	czas konwersji przetwornika w ns
LC0_GMINV	1	dolna granica zakresu napięć w dziesiątych częściach wolta
LC0_GMAXV	1	górna granica zakresu napięć w dziesiątych częściach wolta
LC0_GDMA	1	numer kanału DMA
LC0_GMINP	64	tablica minimalnych okresów próbkowania

Przeznaczenie:

Funkcja zwraca informację o konfiguracji badanego toru analogowo-cyfrowego.

Tablica LC0_GMINP zawiera wartości minimalnych okresów próbkowania w 1, 2, 3 ,..., 32 kanałach. Wypełnionych jest pierwszych n pozycji, gdzie n jest maksymalną liczbą kanałów modułu. Wartości okresów dotyczą transmisji pojedynczego bloku DMA. Okresy podane są w dziesiątych częściach mikrosekundy.

Błędy (LC0_STATUS):

LC0_NONEX_DEV - przetwornik a/c o tym numerze nie istnieje
LC0_NO_MODULE - nie ma takiego modułu

4.4.4. Wyjścia analogowe.

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 3
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_GTYPE	1	rodzaj urządzenia = 4
LC0_GMODULE	1	numer modułu
LC0_GNUM	1	numer przetwornika
parametry wyjściowe:		
LC0_GCHAN	1	liczba kanałów przetwornika
LC0_GRES	1	liczba bitów przetwornika
LC0_GTIME	2	czas konwersji przetwornika w ns
LC0_GMINV	1	dolna granica zakresu napięć w dziesiątych częściach wolta
LC0_GMAXV	1	górna granica zakresu napięć w dziesiątych częściach wolta
LC0_GDMA	1	numer kanału DMA

Przeznaczenie:

Funkcja zwraca informację o konfiguracji badanego toru cyfrowo-analogowego.

Błędy (LC0_STATUS):

LC0_BAD_DEV_TYP - brak przetworników c/a
LC0_NONEX_DEV - przetwornik c/a o tym numerze nie istnieje
LC0_NO_MODULE - nie ma takiego modułu

4.4.5. Kanały układów licznikowo-czasowych (CTC).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 3
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_GTYPE	1	rodzaj urządzenia = 5
LC0_GMODULE	1	numer modułu

LC0_GNUM	1	numer kanału

Przeznaczenie:
Funkcja informuje czy badany kanał CTC istnieje (przez kod odpowiedzi).

Błędy (LC0_STATUS):
LC0_BAD_DEV_TYP - brak dostępnych kanałów CTC
LC0_NONEX_DEV - kanał CTC o tym numerze nie istnieje
LC0_NO_MODULE - nie ma takiego modułu

4.5. Zadeklarowanie częstotliwości zegara magistrali (SET_CLOCK).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 4
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_LCLOCK	2	częstotliwość zegara w kHz

Przeznaczenie:
Funkcja nie jest realizowana dla opisanych modułów.

4.6. Ustawienie zakresu napięć (SET_VOLTAGE_RANGE).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 5
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_VTYPE	1	rodzaj urządzenia (3 lub 4)
LC0_VMODULE	1	numer modułu
LC0_VNUM	1	numer przetwornika
LC0_VMINV	1	dolna granica zakresu napięć w dziesiątych częściach wolta
LC0_VMAXV	1	górna granica zakresu napięć w dziesiątych częściach wolta

Przeznaczenie:
Funkcja nie jest realizowana dla opisanych modułów.

4.7. Zadeklarowanie maksymalnego czasu obsługi obcych przerwań (SET_TIME).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 6
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_ETIME	2	maksymalny czas obsługi w dziesiątych częściach ###s

Przeznaczenie:
Funkcja nie jest realizowana dla opisanych modułów.

4.8. Oczekiwanie na zakończenie operacji (WAIT_FOR_END).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 7
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_WTYPE	1	rodzaj urządzenia (3 lub 4)
LC0_WMODULE	1	numer modułu
LC0_WNUM	1	numer przetwornika
LC0_WMODE	1	tryb pracy
parametry wyjściowe:		
LC0_WRMNUM	4	rzeczywista liczba zmierzonych / wysłanych próbek
LC0_WREMAR	2	rzeczywista długość marginesu końcowego

Przeznaczenie:
Funkcja nie jest realizowana dla opisanych modułów.

4.9. Przerwanie operacji (BREAK).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 8
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_BMODE	1	tryb pracy
LC0_BPROC	4	adres procedury obsługi Ctrl-Break

Przeznaczenie:

Przerywanie pracy driver'a i modułu, instalowanie i wyinstalowywanie procedur obsługi przerwania generowanego przez klawisze Ctrl-Break.

Tryby pracy:

nazwa	wartość	znaczenie
LC0_BREAK_EXEC	0	przerwij pracę driver'a i modułu
LC0_BREAK_INST	1	zainstaluj procedurę obsługi przerwania Ctrl-Break
LC0_BREAK_UNINST	2	wyinstaluj procedurę obsługi przerwania Ctrl-Break

Przerwanie pracy - tryb LC0_BREAK_EXEC:

Działanie funkcji polega na przerwaniu wszystkich operacji wykonywanych we wszystkich modułach. W momencie wywołania tej funkcji driver i moduł mogą znajdować się w jednym z dwóch stanów:

1. Nie jest wykonywana żadna operacja - driver zwraca błąd LC0_STATUS = LC0_NO_OPER, LC0_ERR_STAT = LC0_E_OK.
2. Moduł wykonuje pod kontrolą driver'a operację synchroniczną. Ponieważ w tym przypadku należy przerwać pracę driver'a, zazwyczaj funkcja wywoływana jest w procedurze obsługi przerwania, np. Ctrl-Break. Driver zwraca LC0_STATUS = LC0_OK i LC0_ERR_STAT = LC0_E_OK.

Zainstalowanie procedury obsługi - tryb LC0_BREAK_INST:

Funkcja instaluje obsługę Ctrl-Break przez przechwycenie przerwania 1B16 i podłożenie nowej procedury obsługi tego przerwania. Użytkownik może podać adres własnej procedury obsługi przerwania (LC0_BPROC, daleki adres offset-segment; jest to procedura typu interrupt) lub zlecić obsługę standardową - przez procedurę wewnętrzną driver'a (LC0_BPROC = 0:0). Procedura obsługi przerwania musi zawierać wykonanie funkcji BREAK w trybie pracy

LC0_BREAK_EXEC (patrz wyżej; to też jest jedyną treścią standardowej procedury obsługi, dostarczanej przez driver).

Wyinstalowanie procedury obsługi - tryb LC0_BREAK_UNINST:

Funkcja przywraca procedurę obsługi przerwania istniejącą przed pierwszym wywołaniem funkcji BREAK w trybie LC0_BREAK_INST. Istotne jest więc aby zadbać o wyinstalowanie obsługi Ctrl-Break przed zakończeniem programu!

Procedura obsługi wyinstalowywana jest również przez funkcję LEAVE_DRIVER (patrz).

Błędy (LC0_STATUS):

LC0_INTR_INST - procedura obsługi Ctrl-Break jest już zainstalowana (dla żądania zainstalowania procedury)

LC0_NO_OPER - z żadnym modułem nie jest związana żadna operacja w toku (dla trybu pracy LC0_BREAK_EXEC)

LC0_BAD_MODE - błędny tryb pracy

Dodatkowe informacje (LC0_ERR_STAT; tylko dla trybu pracy LC0_BREAK_EXEC):

LC0_E_BROKEN_RUN - funkcja wywołana w trakcie przetwarzania przy operacji asynchronicznej

LC0_E_BROKEN_WAIT - funkcja wywołana w trakcie oczekiwania na spełnienie warunku startu przy operacji asynchronicznej

4.10. Wejście cyfrowe (DIGITAL_INPUT).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 9
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_DMODULE	1	numer modułu
LC0_DNUM	1	numer portu
LC0_DSTST	1	typ warunku startu operacji
LC0_DVAL	1	odczytana wartość
LC0_DSTART	5	parametry warunku startu

Przeznaczenie:

Funkcja odczytuje stan wejść cyfrowych podanego portu.

Błędy (LC0_STATUS):

LC0_ILL_START - błędne parametry trybu startu operacji

LC0_ILL_START_CODE - błędny tryb startu operacji

LC0_NONEX_DEV - port wejściowy o tym numerze nie istnieje
LC0_NO_MODULE - nie ma takiego modułu

Dodatkowe informacje o błędnych parametrach warunku startu operacji (LC0_ERR_STAT):
LC0_E_BAD_CHAN - numer nieistniejącego kanału
LC0_E_BAD_DATE - zła specyfikacja daty
LC0_E_BAD_TIME - zły odcinek czasu
LC0_E_NONEX_DEV - nie istnieje urządzenie o tym numerze
LC0_E_NO_MODULE - nie ma takiego modułu

4.11. Wyjście cyfrowe (DIGITAL_OUTPUT).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 10
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_OMODULE	1	numer modułu
LC0_ONUM	1	numer portu
LC0_OSTST	1	typ warunku startu operacji
LC0_OVAL	1	zapisywana wartość
LC0_OSTART	5	parametry warunków startu

Przeznaczenie:
Wysłanie wartości na wyjścia cyfrowe podanego portu.

Błędy (LC0_STATUS):
LC0_ILL_START - błędne parametry trybu startu operacji
LC0_ILL_START_CODE - błędny tryb startu operacji
LC0_NONEX_DEV - port wejściowy o tym numerze nie istnieje
LC0_NO_MODULE - nie ma takiego modułu

Dodatkowe informacje o błędnych parametrach warunku startu operacji (LC0_ERR_STAT):
LC0_E_BAD_CHAN - numer nieistniejącego kanału
LC0_E_BAD_DATE - zła specyfikacja daty
LC0_E_BAD_TIME - zły odcinek czasu
LC0_E_NONEX_DEV - nie istnieje urządzenie o tym numerze
LC0_E_NO_MODULE - nie ma takiego modułu

4.12. Zapis CTC (CTC_WRITE).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 11
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_CMODULE	1	numer modułu
LC0_CMODE	1	tryb pracy funkcji
LC0_CFUN	1	tryb pracy kanału CTC
LC0_CVAL	2	wpisywana wartość licznika

Przeznaczenie:
Funkcja nie jest realizowana dla opisanych modułów.

4.13. Odczyt CTC (CTC_READ).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 12
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_UMODULE	1	numer modułu
LC0_UNUM	1	numer kanału CTC
parametry wyjściowe:		
LC0_UVAL	2	odczytana wartość licznika

Przeznaczenie:
Funkcja nie jest realizowana dla opisanych modułów.

4.14. Transmisja danych (DATA_TRANSMIT).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 13

LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_RMODE	1	tryb przesyłania
LC0_RADDR	4	adres bufora (offset-segment)
LC0_RLEN	4	długość bufora
LC0_RMEAS	4	numer próbki, od której należy zacząć
LC0_RNUM	4	liczba próbek do przesłania
LC0_RMEMA	4	adres absolutny bufora w pamięci rozszerzonej
parametry wyjściowe:		
LC0_RRNUM	4	rzeczywista liczba przesłanych próbek

Przeznaczenie:

Funkcja przepisuje ciąg próbek między buforem w pamięci podstawowej a buforem w pamięci rozszerzonej.

Znaczenie poszczególnych parametrów:

a. LC0_RMODE

Tryb pracy funkcji:

nr bitu	wartość	nazwa	znaczenie
1	1	LC0_TO_EXT_DIR	do pamięci rozszerzonej
	0	LC0_FROM_EXT_DIR	z pamięci rozszerzonej
2..8	---	---	zarezerwowane; zawsze 0

b. LC0_RADDR

Adres bufora w podstawowej pamięci komputera.

c. LC0_RLEN

Długość bufora LC0_RADDR podana w próbkach. Ten parametr nie jest związany z liczbą próbek do przesłania.

d. LC0_RMEAS

Numer pierwszej próbki do przesłania. Przesłane zostaną próbki o numerach LC0_RMEAS, LC0_RMEAS + 1 itd. Należy pamiętać, że pierwsza próbka ma numer 1. Jeżeli mierzonych było np. 5 kanałów to 10 próbka w pierwszym kanale ma numer 51.

e. LC0_RNUM

Całkowita liczba próbek do przesłania.

f. LC0_RMEMA

Parametr określa absolutny, 24-bitowy adres bufora w pamięci rozszerzonej. Adres ten nie może wychodzić poza obszar bufora zadeklarowanego przy instalacji driver'a.

g. LC0_RRNUM

Rzeczywista liczba przepisanych próbek. Parametr LC0_RRNUM przybiera wartość będącą minimum z LC0_RLEN, <n> i LC0_RNUM. <n> jest tu liczbą próbek zawartych między próbką LC0_RMEAS a końcem bufora w pamięci rozszerzonej.

Ostrzeżenia (LC0_STATUS):

LC0_OTHER_LEN - przepisano mniejszą liczbę próbek niż żądano

Błędy (LC0_STATUS):

LC0_BAD_BUF_ADR - błędny adres bufora w pamięci podstawowej (odnoszący się do nieistniejącej pamięci)
LC0_BAD_BUF_LEN - błędna długość bufora w pamięci podstawowej (powodująca wyjście bufora poza pamięć itp.)
LC0_BAD_EXTMEM - błędny adres bufora w pamięci rozszerzonej
LC0_BAD_MNUM - błędny numer pierwszej próbki (np. powodujący wyjście poza jeden z buforów)
LC0_BAD_MODE - błędny tryb pracy
LC0_BROKEN - transmisja przerwana z powodu wykonania funkcji BREAK
LC0_NO_EXTMEM - brak pamięci rozszerzonej

4.15. Przetwarzanie analogowo-cyfrowe (ANALOG_INPUT).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 14
LC0_STATUS	1	kod zakończenia funkcji

LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_AMODULE	1	numer modułu
LC0_ANUM	1	numer przetwornika
LC0_AMODE	2	tryb pracy funkcji
LC0_ASTST	1	typy warunków startu / stopu operacji
LC0_APER	4	okres próbkowania
LC0_APER2[1)]	2	wielokrotność okresu próbkowania dla kanałów dodatkowych
LC0_ACHAN	1	liczba kanałów / numer kanału
LC0_ACHAN2[1)]	1	liczba kanałów dodatkowych
LC0_AADDR	4	adres bufora (offset-segment) w pamięci podstawowej
LC0_ALEN	4	długość bufora w pamięci podstawowej
LC0_AMEMA	4	adres absolutny bufora w pamięci rozszerzonej
LC0_ABMAR[1)]	2	długość marginesu początkowego
LC0_AEMAR[1)]	2	długość marginesu końcowego
LC0_AHAND[1)]	2	numer handlera pliku dyskowego
LC0_ASTART	5	warunki startu
LC0_ASTOP	5	warunki stopu
parametry wyjściowe:		
LC0_ARDIV1	2	rzeczywisty podzielnik zegara modułu
LC0_ARDIV2	2	- " -
LC0_ARMNUM	4	rzeczywista liczba zmierzonych próbek
LC0_ARBMAR[2)]	2	rzeczywista dł. marginesu początkowego
LC0_AREMAR[2)]	2	rzeczywista dł. marginesu końcowego
LC0_ARLEN	4	rzeczywista liczba przepisanych próbek
LC0_ARBUF[2)]	2	początek bufora cyklicznego

[1)] Wielkość bez znaczenia dla modułów serii LC-011 i LC-020.

2) Wielkość niekreślona dla modułów serii LC-011 i LC-020.

Przeznaczenie:

Jest to podstawowa funkcja driver'a sterująca główną częścią modułu - torem przetwarzania analogowo - cyfrowego.

Generalnie pomiar (w trybie blokowym, za pośrednictwem transmisji DMA) można podzielić na następujące fazy:

1. Przygotowanie toru pomiarowego do pracy.
2. Oczekiwanie na spełnienie warunku startu przetwarzania.
3. Start właściwego pomiaru.
4. Oczekiwanie na spełnienie warunku stopu przetwarzania.
6. Zakończenie pomiaru.

Znaczenie poszczególnych parametrów:

a. LC0_AMODE

Parametr określa tryb pracy funkcji przy czym znaczenie mają kolejne bity parametru:

nazwa	nr bitu	uwagi
LC0_MOD_START	1	
LC0_MOD_NEW_PAR	2	
LC0_MOD_SYNCHR	3	1
LC0_MOD_INTR	4	ignorowane
LC0_MOD_INTR_TYPE	5	ignorowane
LC0_MOD_BLOCK	6	
LC0_MOD_CYCL	7	ignorowane
LC0_MOD_FILE	8	ignorowane
LC0_MOD_MEM_W	9	
LC0_MOD_EXT_CLK	10	ignorowane
LC0_MOD_EXT_MEM	11	
-------	12..16	zarezerwowane; zawsze 0

LC0_MOD_START:
- start pomiarów
  ustawienie tego bitu na 1 oznacza żądanie wykonania przetwarzania; wartość 0 powoduje jedynie analizę poprawności i zapamiętanie parametrów funkcji

LC0_MOD_NEW_PAR:
- ustawienie nowych parametrów
  1 oznacza, że parametry przetwarzania pobierane będą z rekordu opisu zlecenia; 0 oznacza, że parametry przetwarzania będą identyczne jak poprzednio - jeżeli do tej pory nie było wykonania funkcji ANALOG_INPUT lub ostatnie było niepoprawne to driver zgłosi błąd LC0_NO_PARAMS

LC0_MOD_SYNCHR
- rodzaj pracy: synchroniczna (1) / asynchroniczna (0):

- praca synchroniczna: driver zwraca sterowanie dopiero po całkowitym zakończeniu przetwarzania.
- praca asynchroniczna: tryb pracy z przerwaniem (LC0_MOD_INTR = 1) na koniec
  przetwarzania (LC0_MOD_INTR_TYPE = 0); driver zwraca sterowanie natychmiast po rozpoczęciu przetwarzania. W przypadku pomiaru przekraczającego pojedynczy blok transmisji DMA, lub przy pracy z buforem cyklicznym, należy w obsłudze przerwania z karty wywołać funkcję ANALOG_INPUT. Przy pracy w tym trybie możliwa jest praca on-line tzn.wyświetlanie i/lub analiza wyników pomiaru na bieżąco.

UWAGA: Praca asynchroniczna stanowi płatną opcję.

LC0_MOD_BLOCK:
- tryb przetwarzania: blokowy (1) / pojedynczy (0)
- tryb blokowy: tryb podstawowy pracy modułu, w którym transmisja bloku próbek do pamięci komputera prowadzona jest za pośrednictwem kanału DMA, możliwy pomiar do bufora w pamięci rozszerzonej;
- tryb pojedynczy: pomiar prowadzony jest bezpośrednio do bufora określonego przez użytkownika w pamięci podstawowej komputera; mierzone jest tylko po jednej próbce z każdego zadeklarowanego kanału; przy pierwszym wykonaniu funkcji w słowie trybu pracy (LC0_AMODE) powinien być ustawiony bit LC0_MOD_NEW_PAR (wówczas istotne są tylko parametry: LC0_AMODULE, LC0_ANUM, LC0_AMODE, LC0_ACHAN, LC0_AADDR, LC0_ALEN, LC0_ASTST i LC0_ASTART; nie jest analizowany okres próbkowania LC0_APER;

UWAGA: należy zwrócić uwagę, że jeżeli chcemy wykonać pomiar bloku próbek, gdzie warunek startu odnosi się do całego bloku to drugi i kolejne pomiary należy wykonywać z warunkiem startu LC0_SIMMED

LC0_MOD_MEM_W:
- przepisanie do pamięci (1) / bez przepisywania do pamięci (0) bit steruje działaniem funkcji po zakończeniu przetwarzania w trybie synchronicznym przy współpracy z pamięcią rozszerzoną; jeżeli bit jest równy 1 to po zakończeniu przetwarzania do pamięci podstawowej (pod adres LC0_AADDR) przepisywanych jest <n> pierwszych próbek gdzie <n> jest minimum z całkowitej liczby zmierzonych próbek (LC0_ARMNUM) i rozmiaru bufora (LC0_ALEN); całkowita liczba przepisanych próbek zwracana jest w parametrze LC0_ARLEN; jeżeli natomiast bit LC0_MOD_MEM_W jest równy 0 to po zakończeniu przetwarzania dane nie są przepisywane do pamięci (wówczas parametry LC0_AADDR i LC0_ALEN nie są analizowane); zarówno w jednym jak i drugim przypadku przepisanie danych do pamięci podstawowej jest możliwe poprzez wykonanie funkcji DATA_TRANSMIT; przy pracy z pamięcią podstawową (LC0_MOD_EXT_MEM = 0) bit LC0_MOD_MEM_W jest ignorowany.

LC0_MOD_EXT_MEM:
- praca z pamięcią rozszerzoną (1) / pamięcią podstawową (0):

przy pracy z pamięcią rozszerzoną zmierzone dane przesyłane są do bufora w pamięci rozszerzonej pod adres LC0_AMEMA; bufor ten musi się zawierać w buforze zadeklarowanym przy instalacji driver'a; zadeklarowany bufor można wykorzystywać jako kilka mniejszych ale należy pamiętać o tym, że driver nie kontroluje zachodzenia na siebie tak utworzonych buforów; dla pracy z pamięcią rozszerzoną adres (LC0_AADDR) i długość (LC0_ALEN) bufora w pamięci podstawowej są analizowane tylko wtedy gdy jednocześnie zapalony zostanie bit LC0_MOD_MEM_W (patrz wyżej); dla pracy z pamięcią podstawową adres bufora w pamięci rozszerzonej (LC0_AMEMA) nie jest analizowany;

b. LC0_ASTST

Parametr określa typy warunków startu i stopu (ten drugi tylko dla pracy blokowej) pomiaru. Jest on sumą odpowiednich kodów typów warunku startu i stopu (patrz p.3.4.).

c. LC0_ASTART, LC0_ASTOP

Parametry te określają szczegółowo warunki startu i stopu operacji. Interpretacja ich zależna jest od zadanych typów warunków startu i stopu operacji (LC0_ASTST). Szczegółowy opis - patrz p.“3.4.

d. LC0_APER

Okres próbkowania dla pracy blokowej z zegarem wewnętrznym (bit parametru LC0_AMODE LC0_MOD_BLOCK = 1). Okres ten podawany jest w dziesiątych częściach mikrosekundy (np. 10000 oznacza 1 ms). Okres ten nie może być mniejszy niż minimalny okres dla danej liczby kanałów. (O minimalnych okresach próbkowania można - należy - się dowiedzieć z programu za pomocą funkcji GET_INFO (parametr LC0_GMINP, patrz opis funkcji).) Liczba kanałów określająca minimalny okres próbkowania wyznaczana jest następująco:
- dla pracy jednokanałowej (najstarszy bit parametru LC0_ACHAN równy 1, patrz niżej opis tego parametru) - oczywiście 1
- dla pracy wielokanałowej (najstarszy bit parametru LC0_ACHAN równy 0, patrz niżej opis tego parametru) - wartość LC0_ACHAN

Z parametrem tym związane są parametry zwrotne driver'a LC0_ARDIV1 i LC0_ARDIV2 (patrz opis poniżej).

e. LC0_ACHAN

Parametr określający tryb pracy modułu: jednokanałowo / wielokanałowo oraz (odpowiednio): liczbę kanałów / numer kanału:

b8	tryb pracy	b1..b7
0	praca wielokanałowa	liczba kanałów (2..16)
1	praca jednokanałowa	numer kanału (1..16)

f. LC0_AADDR, LC0_ALEN

Adres i długość bufora w pamięci podstawowej. Parametry te są analizowane tylko wtedy, gdy bit LC0_MOD_EXT_MEM trybu pracy (LC0_AMODE) został ustawiony na 0 (praca z pamięcią podstawową) lub - w przeciwnym razie - gdy bit LC0_MOD_MEM_W = 1 i LC0_MOD_SYNCHR = 1(praca synchroniczna z pamięcią rozszerzoną z przepisaniem do pamięci podstawowej). Adres bufora jest adresem dalekim, tzn. podanym w postaci offset-segment, natomiast długość bufora podawana jest w próbkach (i jest to liczba długa tj. 32 bitowa). Długość bufora jest jednym z czynników określających liczbę próbek przepisywanych do pamięci podstawowej (patrz opis bitu LC0_MOD_MEM_W).

h. LC0_AMEMA

Parametr istotny tylko dla pracy z pamięcią rozszerzoną (LC0_MOD_EXT_MEM = 1). Oznacza absolutny, 24-bitowy adres bufora w pamięci rozszerzonej. Adres ten nie może wychodzić poza obszar bufora zadeklarowanego przy instalacji driver'a.

i. LC0_ARDIV1, LC0_ARDIV2

Częstotliwość próbkowania tworzona jest przez podział (przez całkowitą wartość) 1/4 częstotliwości generatora w module. Poza tym podzielnik ten jest faktycznie iloczynem dwóch liczb (dwa połączone liczniki 16-bitowe). W związku z tym okres próbkowania zadany przez użytkownika nie zawsze jest osiągalny. W takim przypadku wybierany jest najbliższy możliwy okres, początkowo mniejszy a jeżeli jest to niemożliwe to większy od zadanego. Parametr LC0_ARDIV1 przekazuje wartość pierwszego licznika natomiast LC0_ARDIV2 - drugiego (rzeczywisty podzielnik zegara modułu jest równy iloczynowi LC0_ARDIV1 * LC0_ARDIV2). Łącznie z częstotliwością zegara modułu (patrz funkcja GET_MODULE_INFORMATION) daje informację o faktycznej częstotliwości próbkowania.

j. LC0_ARMNUM

Rzeczywista liczba zmierzonych próbek.

Błędy (LC0_STATUS):

LC0_BAD_BUF_ADR - błędny adres bufora (odnoszący się do nieistniejącej pamięci lub do pamięci rozszerzonej w przypadku transmisji programowej)

LC0_BAD_BUF_LEN - błędna długość bufora (powodująca wyjście bufora poza pamięć, przejście pomiędzy pamięcią podstawową a rozszerzoną, przekroczenie rozmiarów bufora w pamięci rozszerzonej itp.)

LC0_BAD_CHAN - numer nieistniejącego kanału

LC0_BAD_CHAN_N - zła liczba kanałów

LC0_BAD_EXTMEM - błędny adres bufora w pamięci rozszerzonej (dla pracy z pamięcią rozszerzoną, LC0_MOD_EXT_MEM = 1)

LC0_BAD_MARGIN - błędna długość marginesu początkowego (nie będąca wielokrotnością liczby kanałów)
LC0_BAD_MODE - błędny tryb pracy
LC0_BROKEN - operacja przerwana z powodu wykonania funkcji BREAK; LC0_ERR_STAT podaje, w jakim momencie operacja została przerwana
LC0_BAD_PER - za krótki okres próbkowania
LC0_DEV_BUSY - urządzenie zajęte; próba wykonania następnej funkcji ANALOG_INPUT przed zakończeniem poprzedniej
LC0_ILL_START - błędne parametry warunku startu
LC0_ILL_STOP - błędne parametry warunku stopu; w obu przypadkach sprecyzowanie błędu podane jest w LC0_ERR_STAT
LC0_ILL_START_CODE - nielegalny typ warunku startu
LC0_ILL_STOP_CODE - nielegalny typ warunku stopu;
LC0_INTR_NOT_INST - procedura obsługi przerwania nie jest zainstalowana (dla pracy z przerwaniem)
LC0_NONEX_DEV - nie istnieje przetwornik o tym numerze
LC0_NO_DMA - z danym przetwornikiem nie jest związany żaden kanał DMA (dla pracy blokowej)
LC0_NO_EXTMEM - brak pamięci rozszerzonej (dla pracy z pamięcią rozszerzoną)
LC0_NO_IRQ - z danym modułem nie jest związane żadne przerwanie (dla pracy z przerwaniem)
LC0_NO_MODULE - nie ma takiego modułu
LC0_NO_PARAMS - zgaszono bit LC0_MOD_NEW_PAR (LC0_AMODE) ale wcześniej nie ustawiono żadnych parametrów (nie było wykonania funkcji ANALOG_INPUT lub ostatnie wykonanie było niepoprawne)
LC0_NO_SECOND_FREQ - moduł nie może prowadzić przetwarzania z dwiema częstotliwościami (LC0_ACHAN2 <> 0)
LC0_OVERRUN - zakończono przetwarzanie z powodu błędu OVERRUN
LC0_TOO_LONG_MARG - suma marginesów dłuższa od bufora

Dodatkowe informacje o błędach (LC0_ERR_STAT):
LC0_E_BAD_CHAN - numer nieistniejącego kanału
LC0_E_BAD_DATE - zła specyfikacja daty
LC0_E_BAD_TIME - zły odcinek czasu
LC0_E_BROKEN_RUN - funkcja przerwana w trakcie przetwarzania
LC0_E_BROKEN_WAIT - funkcja przerwana w trakcie oczekiwania na spełnienie warunku startu
LC0_E_NONEX_DEV - nie istnieje urządzenie o tym numerze
LC0_E_NO_MODULE - nie ma takiego modułu

4.16. Przetwarzanie cyfrowo-analogowe (ANALOG_OUTPUT).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 15
LC0_STATUS	1	kod zakończenia funkcji
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_NMODULE	1	numer modułu
LC0_NNUM	1	numer przetwornika
LC0_NMODE	2	tryb pracy funkcji
LC0_NSTST	1	typy warunków startu / stopu operacji
LC0_NCHAN	1	liczba kanałów / numer kanału
LC0_NPER	4	okres sterowania
LC0_NADDR	4	adres bufora
LC0_NLEN	4	długość bufora
LC0_NHAND	2	numer handlera pliku dyskowego
LC0_NSTART	5	warunki startu
LC0_NSTOP	5	warunki stopu

Przeznaczenie:

Funkcja służy wysyłania do danych na wyjście analogowe.

a. LC0_NMODE

Parametr określa tryb pracy funkcji przy czym znaczenie mają kolejne bity parametru:

nazwa	nr bitu	uwagi
LC0_MOD_START	1	
LC0_MOD_NEW_PAR	2	
LC0_MOD_SYNCHR	3	1
LC0_MOD_INTR	4	ignorowane
LC0_MOD_INTR_TYPE	5	zarezerwowane; zawsze 0
LC0_MOD_BLOCK	6	
LC0_MOD_CYCL	7	ignorowane
LC0_MOD_FILE	8	zarezerwowane; zawsze 0
LC0_MOD_MEM_W	9	zarezerwowane; zawsze 0
LC0_MOD_EXT_CLK	10	
LC0_MOD_EXT_MEM	11	ignorowane
-------	12..16	zarezerwowane; zawsze 0

LC0_MOD_START:
- start pomiarów
  ustawienie tego bitu na 1 oznacza żądanie wykonania przetwarzania; wartość 0 powoduje jedynie analizę poprawności i zapamiętanie parametrów funkcji

LC0_MOD_NEW_PAR:
- ustawienie nowych parametrów
  1 oznacza, że parametry przetwarzania pobierane będą z rekordu opisu zlecenia; 0 oznacza, że parametry przetwarzania będą identyczne jak poprzednio - jeżeli do tej pory nie było wykonania funkcji ANALOG_OUTPUT lub ostatnie było niepoprawne to driver zgłosi błąd LC0_NO_PARAMS

LC0_MOD_SYNCHR:
- rodzaj pracy: synchroniczna (1) / asynchroniczna (0):
- praca synchroniczna: driver zwraca sterowanie dopiero po całkowitym zakończeniu przetwarzania; - praca asynchroniczna: driver zwraca sterowanie - ogólnie rzecz ujmując - tak szybko jak tylko może; moment ten jest zależny od wielu czynników ale wszystkie one sprowadzają się do jednego: wiadomo dokładnie ile próbek należy jeszcze przetransmitować i rozpoczęto transmisję ostatniego bloku DMA (wówczas nie ma potrzeby programowej kontroli transmisji) -

UWAGA: Praca asynchroniczna stanowi płatną opcję.

LC0_MOD_BLOCK:
- tryb przetwarzania: blokowy (1) / pojedynczy (0)
- tryb blokowy: tryb podstawowy pracy modułu, w którym transmisja bloku danych z pamięci komputera do modułu prowadzona jest za pośrednictwem kanału DMA;
- tryb pojedynczy: wysyłanie danych prowadzone jest bezpośrednio z bufora określonego przez użytkownika w pamięci podstawowej komputera do rejestrów modułu; wysyłane jest tylko po jednej próbce do każdego zadeklarowanego kanału; przy pierwszym wykonaniu funkcji w słowie trybu pracy (LC0_NMODE) powinien być ustawiony bit LC0_MOD_NEW_PAR; nie jest analizowany okres wysyłania LC0_NPER ani warunek stopu przetwarzania (część LC0_NSTST oraz LC0_NSTOP) jako nie mające w tym kontekście sensu); przy następnych wykonaniach funkcji bit ten może być (choć nie jest to konieczne) zgaszony; wymagane jest zadeklarowanie adresu bufora z pamięci podstawowej;

UWAGA: należy zwrócić uwagę, że jeżeli chcemy wykonać wysłanie bloku próbek, gdzie warunek startu odnosi się do całego bloku to drugie i kolejne wysłanie należy wykonywać z warunkiem startu LC0_SIMMED

LC0_MOD_CYCL:
- praca z buforem cyklicznym (1) / buforem prostym (0):
  praca z buforem cyklicznym polega zaprogramowaniu specjalnego trybu pracy kanału DMA, w którym po dojściu do końca bufora adres transmisji ustawiany jest

na jego początek i transmisja jest kontynuowana; ze względu na ograniczenia sprzętowe układów DMA bufor cykliczny nie może leżeć na granicy 64kB (np. jeżeli zaczyna się pod adresem absolutnym 50kB to jego długość musi być mniejsza niż 14kB); przy pracy z przerwaniem jest ono generowane po każdym wysłaniu całego bufora

b. LC0_NSTST

Parametr określa typy warunków startu i stopu (ten drugi tylko dla pracy blokowej) funkcji. Jest on sumą odpowiednich kodów typów warunku startu i stopu (patrz p.3.4.). Warunek startu nie może być typu LC0_SHARD.

c. LC0_NSTART, LC0_NSTOP

Parametry te określają szczegółowo warunki startu i stopu operacji. Interpretacja ich zależna jest od zadanych typów warunków startu i stopu operacji (LC0_NSTST). Szczegółowy opis - patrz p. 3.4.

d. LC0_NCHAN

Parametr określający tryb pracy modułu: jednokanałowo / wielokanałowo oraz (odpowiednio): liczbę kanałów / numer kanału:

b8	tryb pracy	b1..b7
0	praca wielokanałowa	liczba kanałów (2)
1	praca jednokanałowa	numer kanału (1..n) gdzie n - liczba kanałów c/a w module

e. LC0_NPER

Okres wysyłania danych do przetwornika. Częstotliwość wysyłania tworzona jest przez podział 1/4 częstotliwości zegara modułu przez 16 bitowy podzielnik. Z tego względu okres LC0_NPER nie może być większy niż 65536 * 4 * okres zegara modułu.

f. LC0_NADDR, LC0_NLEN

LC0_NADDR oznacza adres w postaci offset-segment,
LC0_NLEN oznacza długość bufora w pamięci podstawowej.

Błędy (LC0_STATUS):

LC0_BAD_BUF_ADR - błędny adres bufora (odnoszący się do nieistniejącej pamięci, dla pracy z pamięcią podstawową)
LC0_BAD_BUF_LEN - błędna długość bufora (powodująca wyjście bufora poza pamięć, przejście pomiędzy pamięcią podstawową a rozszerzoną itp.; dla pracy z pamięcią podstawową)
LC0_BAD_CHAN - numer nieistniejącego kanału
LC0_BAD_CHAN_N - zła liczba kanałów
LC0_BAD_EXTMEM - błędny adres bufora w pamięci rozszerzonej (dla pracy z pamięcią rozszerzoną)

LC0_BAD_MODE - błędny tryb pracy
LC0_BAD_PER - za długi lub za krótki okres wysyłania
LC0_BROKEN - transmisja przerwana z powodu wykonania funkcji BREAK; LC0_ERR_STAT podaje, w jakim momencie operacja została przerwana
LC0_DEV_BUSY - urządzenie zajęte; próba wykonania następnej funkcji ANALOG_OUTPUT przed zakończeniem poprzedniej
LC0_ILL_START - błędne parametry sposobu startu
LC0_ILL_STOP - błędne parametry warunku stopu; w obu przypadkach sprecyzowanie błędu podane jest w LC0_ERR_STAT
LC0_ILL_START_CODE - nielegalny sposób startu
LC0_ILL_STOP_CODE - nielegalny sposób stopu
LC0_INTR_NOT_INST - procedura obsługi przerwania nie jest zainstalowana (dla pracy z przerwaniem)
LC0_NONEX_DEV - nie istnieje przetwornik o tym numerze
LC0_NO_DMA - z danym przetwornikiem nie jest związany żaden kanał DMA (dla pracy blokowej)
LC0_NO_EXTMEM - brak pamięci rozszerzonej (dla pracy z pamięcią rozszerzoną)
LC0_NO_IRQ - z danym modułem nie jest związane żadne przerwanie (dla pracy z przerwaniem)
LC0_NO_MODULE - nie ma takiego modułu
LC0_NO_PARAMS - zgaszono bit LC0_MOD_NEWPAR (LC0_NMODE) ale wcześniej nie ustawiono żadnych parametrów (nie było wykonania funkcji ANALOG_OUTPUT lub ostatnie wykonanie było niepoprawne)

Dodatkowe informacje o błędach (LC0_ERR_STAT):
LC0_E_BAD_CHAN - numer nieistniejącego kanału
LC0_E_BAD_DATE - zła specyfikacja daty
LC0_E_BAD_TIME - zły odcinek czasu
LC0_E_BROKEN_RUN - funkcja przerwana w trakcie przetwarzania
LC0_E_BROKEN_WAIT - funkcja przerwana w trakcie oczekiwania na spełnienie warunku startu
LC0_E_NONEX_DEV - nie istnieje urządzenie o tym numerze
LC0_E_NO_MODULE - nie ma takiego modułu

4.17. Zakończenie pracy z driver'em (LEAVE_DRIVER).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 16

LC0_STATUS	1	kod zakończenia funkcji = LC0_OK
LC0_ERR_STAT	1	dodatkowe informacje o błędach

Przeznaczenie:

Funkcja, którą należy wywołać przed zakończeniem programu. Powoduje "zapomnienie" przez driver wszystkiego co zostało mu przekazane w trakcie pracy programu. Zapobiega to błędnemu działaniu driver'a np. w sytuacji, gdy następny program żąda wykonania przetwarzania a/c wg. poprzednich parametrów (LC0_MOD_NEW_PAR = 0) - a żadnych parametrów nie podał. Wykonanie funkcji w takiej sytuacji spowodowałoby np. przetransmitowanie danych na obszar ciała programu (adres bufora odziedziczony po poprzednim programie).

Funkcja wyinstalowuje również obsługę przerwania zegarowego 1C16 przez driver.

Funkcja zeruje wszystkie zainstalowane moduły oraz wyinstalowywuje procedurę obsługi przerwania od Ctrl-Break zainstalowaną przez funkcję BREAK (patrz) a także procedurę obsługi przerwania z modułu zainstalowaną przez funkcję INTERRUPT_SERVICE (patrz).

4.18. Obsługa przerwań (INTERRUPT_SERVICE).

Rekord opisu zlecenia:

nazwa	rozmiar w bajtach	znaczenie
LC0_CODE	1	kod funkcji = 17
LC0_STATUS	1	kod zakończenia funkcji = LC0_OK
LC0_ERR_STAT	1	dodatkowe informacje o błędach
LC0_SMODULE	1	numer modułu
LC0_SPROC	4	adres procedury użytkownika
LC0_SSTAT	4	adres słowa komunikacyjnego

Przeznaczenie:

Funkcja w przygotowaniu (opcja).

5. Zestawienie kodów zakończenia funkcji.

nazwa	kod	znaczenie
LC0_OK	0	poprawne zakończenie funkcji

Błędy (LC0_STATUS):

nazwa	kod	znaczenie
LC0_UNKN_FUNC	-1	nieznany kod funkcji
LC0_NO_MODULE	-2	nie istnieje żaden z żądanych modułów; nie ma takiego modułu
LC0_BAD_DEV_TYP	-3	brak urządzeń danego typu
LC0_NONEX_DEV	-4	nie istnieje urządzenie o tym numerze
LC0_BAD_FREQ	-5	błędna częstotliwość
LC0_BAD_RANGE	-6	błędny zakres napięć
LC0_NO_OPER	-7	z wyspecyfikowanym urządzeniem nie jest związana żadna operacja w toku
LC0_BAD_MARGIN	-8	błędna długość marginesu początkowego (nie będąca wielokrotnością liczby kanałów
LC0_BAD_BUF_ADR	-9	błędny adres bufora (odnoszący się do nieistniejącej pamięci lub do pamięci rozszersonej w przypadku transmisji programowej)
LC0_BAD_BUF_LEN	-10	błędna długość bufora (powodująca wyjście bufora poza pamięć, przejście pomiędzy pamięcią podstawową a rozszerzoną, przekroczenie rozmiarów bufora w pamięci rozszerzonej itp.)
LC0_DEV_BUSY	-11	urządzenie zajęte
LC0_BAD_PER	-12	za długi lub za krótki okres
LC0_BAD_CHAN_N	-13	zła liczba kanałów
LC0_BAD_CHAN	-14	numer nieistniejącego kanału
LC0_BROKEN	-15	przetwarzanie przerwane z powodu wykonania funkcji BREAK
LC0_INTR_NOT_INST	-16	procedura obsługi przerwania nie jest zainstalowana
LC0_ILL_START_CODE	-17	nielegalny typ warunku startu
LC0_ILL_STOP_CODE	-18	nielegalny typ warunku stopu
LC0_BAD_PROC	-19	błędny adres procedury obsługi przerwania lub słowa komunikacyjnego (spoza pamięci podstawowej)

LC0_TOO_LONG_MARG	-20	suma marginesów dłuższa od bufora
LC0_ILL_START	-21	błędne parametry warunku startu
LC0_ILL_STOP	-22	błędne parametry warunku stopu
LC0_BAD_MNUM	-23	błędny numer pierwszej próbki
LC0_NOT_SUPPORTED	-24	dla danego modułu funkcja nie jest realizowana
LC0_BAD_CTC_MODE	-25	błędny tryb pracy CTC
LC0_NO_PARAMS	-26	nie podano parametrów przetwarzania a/c, c/a
LC0_OVERRUN	-27	zakończono przetwarzanie a/c z powodu błędu OVERRUN
LC0_NO_DMA	-28	z danym urządzeniem nie jest związany żaden kanał DMA
LC0_NO_IRQ	-29	z danym modułem nie jest związane żadne przerwanie lub procedura obsługi nie została zainstalowana
LC0_NOT_FULLY_SUP	-30	żądany tryb wykonania funkcji nie jest dla danego typu modułu realizowany lub jest w opracowaniu
LC0_NO_EXTMEM	-31	brak pamięci rozszerzonej
LC0_NO_SEC_FREQ	-32	moduł nie może prowadzić przetwarzania z dwiema częstotliwościami
LC0_INTR_INST	-33	procedura obsługi przerwania jest już zainstalowana
LC0_BAD_PER2[1)]	-34	błędna wielokrotność okresu próbkowania (0 lub 1)
LC0_BAD_MODE	-35	błędny tryb pracy
LC0_BAD_EXTMEM	-36	błędny adres bufora w pamięci rozszerzonej
LC0_CTC_NOT_PROGRAMMED	-37	zlecono zapis wartości licznika lecz nie zaprogramowano trybu pracy kanału
LC0_REJECTED	-38	za duzo równoczesnych odwołań do driver'a

1) Nie występuje dla modułów serii LC-011 i LC-020.

Dodatkowe informacje o błędach (LC0_ERR_STAT):

nazwa	kod	znaczenie
LC0_E_OK	0	brak dodatkowych informacji
błędy w warunkach startu/stopu		

LC0_E_NO_MODULE	-1	nie ma takiego modułu
LC0_E_NONEX_DEV	-2	nie istnieje urządzenie o tym numerze
LC0_E_BAD_CHAN	-3	numer nieistniejącego kanału
LC0_E_BAD_TIME	-4	zły odcinek czasu
LC0_E_BAD_DATE	-5	zła specyfikacja daty
LC0_E_BAD_THRE	-6	błędny próg wyzwalania analogoweg
moment przerwania przez funkcję BREAK:		
LC0_E_BROKEN_WAIT	-7	funkcja przerwana w trakcie oczekiwania na spełnienie warunku startu
LC0_E_BROKEN_RUN	-8	funkcja przerwana w trakcie przetwarzania

Ostrzeżenia (LC0_STATUS):

nazwa	kod	znaczenie
LC0_NON_EX_MOD	1	zażądano inicjalizacji nie istniejących modułów ale co najmniej 1 moduł został zainicjalizowany
LC0_OTHER_LEN	2	przepisano mniej próbek niż żądano
LC0_PREMATURE_END	3	przedwczesne zakończenie operacji z powodu przepełnienia / opróżnienia całego bufora
LC0_IN_PROGRESS	4	badana transmisja jeszcze trwa

6. Projektowanie programów użytkowych.

W poniższym rozdziale zostanie omówiony sposób komunikacji programów użytkowych z driver'em. Na wstępie opisane zostaną ogólne zasady komunikacji a w dalszych podrozdziałach - komunikacja z driver'em z poziomu programów napisanych w C, Pascalu i asemblerze.

Wykonanie dowolnej z funkcji driver' wymaga następujących czynności:
- wypełnienie odpowiedniego rekordu opisu zlecenia; odpowiednie struktury danych dostarczane są przez producenta w postaci zbiorów źródłowych
- wpisanie adresu rekordu do rejestrów DX i DI
- wykonanie odpwiedniego przerwania programowego; numer przerwania zależny jest od konkretnego driver'a; numery przerwań dla poszczególnych driver'ów podane zostały w rozdziale 3.2.

Dobrze napisany program powinien składać się z następujących części:
- część wstępna:

- stwierdzenie, czy driver jest zainstalowany; metodą wykrycia obecności driver'a w pamięci systemu jest próba otwarcia urządzenia o nazwie określonej przez driver (patrz p.3.2.; powodzenie tej próby świadczy o zainstalowamiu driver'a, niepowodzenie - o jego braku
- rozpoznanie konfiguracji modułu (GET_TOTAL_CONFIGURATION - ile i jakich modułów jest zainstalowanych, GET_MODULE_CONFIGURATION - czy moduł ma zainstalowany przetwornik c/a, czy jest podłączony do któregoś z przerwań itp., GET_INFO - czy tor pomiarowy a/c jest podłączony do kanału DMA, jakie są minimalne okresy próbkowania, jakie są zakresy napięc przetworników a/c i c/a itp.); ten etap jest szczególnie ważny, gdy projektowanyjest program uniwersalny, mający operować na kilku rodzajach modułów
- inicjalizacja modułu; jest konieczna zwłaszcza w sytuacji, gdy przewidywane jest używanie czasowych warunków startu operacji - inicjalizacja powoduje (między innymi) synchronizację programowego zegara czasu rzeczywistego zawartego w driverze z zegarem komputera (który musi być prawidłowo ustawiony!)
- instalacja procedury obsługi przerwania generowanego przez klawisz Ctrl-Break (jeżeli przewiduje się jego użycie)
- część wykonawcza: tu powinny się znaleźć funkcje wykonujące właściwe operacje modułu jak ANALOG_INPUT, DATA_TRANSMIT, ANALOG_OUTPUT, DIGITAL_INPUT, DIGITAL_OUTPUT itp.; należy zwrócić uwagę na dwie rzeczy:
- każda funkcja może być wywołana z błędnymi parametrami i zasygnalizować to w kodzie odpowiedzi (LC0_STATUS, LC0_ERR_STAT); należy koniecznie sprawdzać tę odpowiedź, szczególnie w dwóch sytuacjach: gdy program jest na etapie uruchamiania i gdy parametry funkcji są dostarczane interakcyjnie przez użytkownika
- moduł może ulec uszkodzeniu - wówczas niektóre funkcje nie mogą się zakończyć (przetwarzanie a/c, oczekiwanie na spełnienie warunków startu związanych z wejściami cyfrowymi czy sygnałem analogowym); poza tym funkcja może zostać wywołana z omyłkowo podanymi parametrami - należy - przewidując taką sytuację - albo umożliwić operatorowi przerwanie takiej funkcji przez naciśnięcie klawisza Ctrl-Break (i wykonanie w procedurze obsługi przerwania funkcji BREAK) albo samodzielnie odmierzać czas wykonania operacji i po przekroczeniu oszacowanego wcześniej limitu - automatycznie wykonywać funkcję BREAK; jest to istotne o tyle, że w przeciwnym razie operator zmuszony będzie powtórnie załadować system co może się wiązać ze stratą zmierzonych uprzednio - i być może unikatowych - danych
- część końcowa:
- przed zakończeniem programu należy koniecznie wykonać funkcję LEAVE_DRIVER; jest to szczególnie ważne wtedy, gdy na komputerze wykonywanych jest kilka programów korzystających z tego samego driver'a.
UWAGA: Jeżeli w swoim programie użytkowym przechwytujemy przerwanie zegarowe 1C16 to struktura programu powinna być następująca:
- wywołanie własnej obsługi przerwania,

- wykonanie funkcji MODULE_INIT (m.in przechwycenie przerwania zegarowego przez driver),
- program pomiarowy,
- wykonanie funkcji LEAVE_DRIVER (m.in wyinstalowanie obsługi przerwania zegarowego przez driver),
- wyinstalowanie własnej obsługi przerwania.

6.1. Programowanie w języku C.

Z poziomu języka C komunikacja z driver'em odbywa się za pośrednictwem procedury bibliotecznej int86. W przypadku implementacji firmy Borland (Turbo C) można korzystać również z pseudozmiennych _DX i _DI oraz makroinstrukcji geninterrupt (dos.h).

Producent dostarcza dwa pliki źródłowe: plik AMBEX-LC.H zawierający definicje wszystkich struktur danych i stałych potrzebnych do współpracy z driver'em oraz plik TEST.C zawierający przykłady wykorzystania poszczególnych funkcji driver'a.

6.2. Programowanie w języku Pascal.

Z poziomu języka Pascal komunikacja z driver'em odbywa się za pośrednictwem procedury bibliotecznej intr.

Producent dostarcza dwa pliki źródłowe: plik AMBEX-LC.PAS zawierający definicje wszystkich struktur danych i stałych potrzebnych do współpracy z driver'em oraz plik TEST.PAS zawierający przykłady wykorzystania poszczególnych funkcji driver'a.

6.3. Programowanie w języku assemblera.

Producent dostarcza pliki źródłowy AMBEX-LC.ASM zawierający definicje wszystkich struktur danych i stałych potrzebnych do współpracy z driver'em.

D O D A T E K A

d o d o k u m e n t a c j i d r i v e r ' a

m o d u ł u k o n t r o l n o - p o m i a r o w e g o

L C - 0 1 1 - 1 6 1 2

AMBEX - LC.H

struktury danych i stałe dla języka C

```
/*************************************************************************/
/*                                    */
/* definicje struktur danych i stalych dla wspolpracy z driver'ami modulow */
/* analogowych serii LC-... z poziomu jezyka C              */
/*                                    */
/*************************************************************************/

/*************** REKORDY OPISOW ZLECEN FUNKCJI DRIVER'A ******************/

/************** struktury opisujace warunki startu i stopu ****************/
struct lc0_cond_level      /* warunek "poziom sygnalu cyfrowego"   */
    {
    unsigned char  mod_nr;     /* numer modulu          */
    unsigned char  port_nr;    /* numer portu           */
    unsigned char  inp_nr;     /* numer wejscia          */
    unsigned char  value;     /* oczekiwana wartosc       */
    };

/*-----------------------------------------------------------------------*/
               /* warunek "zbocze sygnalu cyfrowego"   */
struct lc0_cond_slope
    {
    unsigned char  mod_nr;     /* numer modulu          */
    unsigned char  port_nr;    /* numer portu           */
    unsigned char  inp_nr;     /* numer wejscia          */
    unsigned char  slope;     /* rodzaj zbocza          */
    };
               /* kody rodzajow zbocza:          */
#define LC0_SLOPE_COND_UP    1    /* narastajace           */
#define LC0_SLOPE_COND_DOWN   0    /* opadajace            */

/*-----------------------------------------------------------------------*/
struct lc0_cond_dig       /* warunek "kominacja sygnalow cyfrowych" */
    {
    unsigned char  mod_nr;     /* numer modulu          */
    unsigned char  port_nr;    /* numer portu           */
    unsigned char  mask;      /* maska aktywnych wejsc      */
    unsigned char  pattern;    /* testowany wzorzec        */
    };


/*-----------------------------------------------------------------------*/
struct lc0_cond_date      /* warunek "data"             */
    {
    unsigned char  second;     /* sekunda             */
    unsigned char  minute;     /* minuta             */
    unsigned char  hour;      /* godzina             */
    unsigned char  day;      /* dzien miesiaca         */
    };

/*-----------------------------------------------------------------------*/
               /* warunek "sygnal analogowy"       */
struct lc0_cond_analog
    {
    unsigned char  mod_nr;     /* numer modulu          */
    unsigned char  converter;   /* nr przetwornika         */
    unsigned char  channel;    /* numer kanalu (b1..b7), kierunek */
                    /*  przekroczenia (b8)      */
    int       level;     /* prog wyzwalania         */
    };
               /* kody kierunku przekroczenia progu:   */
#define LC0_ANALOG_COND_UP   0x80  /* w kierunku wiekszych wartosci  */
#define LC0_ANALOG_COND_DOWN  0    /* w kierunku mniejszych wartosci */
#define LC0_ANALOG_COND_MASK  0x80  /* maska kodu kierunku       */

/*-----------------------------------------------------------------------*/
union lc0_start         /* ============ warunki startu =============*/
    {
    struct lc0_cond_level level;     /* LC0_SLEVEL */
    struct lc0_cond_slope slope;     /* LC0_SSLOPE */
    struct lc0_cond_dig  dig_eq;     /* LC0_SDIG_EQ */
    struct lc0_cond_dig  dig_ne;     /* LC0_SDIG_NE */
    unsigned long      time;      /* LC0_STIME  */
    struct lc0_cond_date  date;      /* LC0_SDATE  */
    struct lc0_cond_analog analog;     /* LC0_SANALOG */
    };
```

```
union lc0_stop         /* ========== warunki stopu ==============*/
    {
    unsigned long      samples;    /* LC0_ZSAMPLES */
    struct lc0_cond_level level;     /* LC0_ZLEVEL  */
    struct lc0_cond_slope slope;     /* LC0_ZSLOPE  */
    struct lc0_cond_dig  dig_eq;     /* LC0_ZDIG_EQ */
    struct lc0_cond_dig  dig_ne;     /* LC0_ZDIG_NE */
    unsigned long      time;      /* LC0_ZTIME  */
    struct lc0_cond_date  date;      /* LC0_ZDATE  */
    struct lc0_cond_analog analog;     /* LC0_ZANALOG */
    };

/*=========================================================================*/

/* Rekord opisu zlecenia funkcji MODULE_INIT ============================*/
struct lc0_init
    {
    unsigned char  LC0_CODE;    /* kod funkcji (0)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_IMODULE;  /* mapa modulow          */
    };

/* Rekord opisu zlecenia funkcji GET_TOTAL_CONFIGURATION ================*/
struct lc0_total
    {
    unsigned char  LC0_CODE;    /* kod funkcji (1)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_TONF;    /* konfiguracja modulow      */
    unsigned char  LC0_TIAD;    /* przetworniki a/c        */
    unsigned char  LC0_TIDA;    /* przetworniki c/a        */
    unsigned char  LC0_TCTC;    /* kanaly CTC           */
    unsigned char  LC0_TIDI;    /* porty wejsc cyfrowych      */
    unsigned char  LC0_TIDO;    /* porty wyjsc cyfrowych      */
    unsigned long  LC0_TMEMA;   /* adres bufora w pamieci     */
                    /*  rozszerzonej - absolutny   */
    unsigned long  LC0_TMEML;   /* dlugosc bufora w pamieci    */
                    /*  rozszerzonej (w probkach;   */
                    /*  0 - brak pamieci rozszerzonej)*/
    };

/* Rekord opisu zlecenia funkcji  GET_MODULE_CONFIGURATION ===============*/
struct lc0_module
    {
    unsigned char  LC0_CODE;    /* kod funkcji (2)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_MMODULE;  /* numer modulu          */
    unsigned int  LC0_MBASE1;   /* adres bazowy pakietu 1     */
    unsigned int  LC0_MBASE2;   /* adres bazowy pakietu 2     */
    unsigned char  LC0_MIAD;    /* przetworniki a/c        */
    unsigned char  LC0_MIDA;    /* przetworniki c/a        */
    unsigned char  LC0_MCTC;    /* kanaly CTC           */
    unsigned char  LC0_MIDI;    /* porty wejsc cyfrowych      */
    unsigned char  LC0_MIDO;    /* porty wyjsc cyfrowych      */
    unsigned int  LC0_MCLOCK;   /* czestotliwosc zegara w kHz   */
    unsigned char  LC0_MINT;    /* numer przerwania (programowy)  */
    };

/* Rekord opisu zlecenia funkcji GET_INFO ================================*/
struct lc0_info
    {
    unsigned char  LC0_CODE;    /* kod funkcji (3)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_GTYPE;   /* rodzaj urzadzenia        */
    unsigned char  LC0_GMODULE;  /* numer modulu          */
    unsigned char  LC0_GNUM;    /* numer przetwornika/portu/    */
                    /*          ukladu CTC */
    unsigned char  LC0_GCHAN;   /* liczba kanalow         */
    unsigned char  LC0_GRES;    /* liczba bitow przetwornika    */
    unsigned int  LC0_GTIME;   /* czas konwersji przetwornika w ns*/
    char      LC0_GMINV;   /* dolna granica zakresu napiec w */
                  /*  dziesiatych czesciach volta  */
```

```
    char      LC0_GMAXV;   /* gorna granica zakresu napiec w */
                  /*  dziesiatych czesciach volta  */
    unsigned char  LC0_GDMA;    /* numer kanalu DMA        */
    unsigned int  LC0_GMINP[32]; /* minimalne okresy probkowania  */
    };

/* Rekord opisu zlecenia funkcji SET_CLOCK ================================*/
struct lc0_clock
    {
    unsigned char  LC0_CODE;    /* kod funkcji (4)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned int  LC0_LCLOCK;   /* czestotliowsc zegara w kHz   */
    };

/* Rekord opisu zlecenia funkcji SET_VOLTAGE_RANGE ========================*/
struct lc0_volt
    {
    unsigned char  LC0_CODE;    /* kod funkcji (5)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_VTYPE;   /* rodzaj urzadzenia        */
    unsigned char  LC0_VMODULE;  /* numer modulu          */
    unsigned char  LC0_VNUM;    /* numer przetwornika       */
    char      LC0_VMINV;   /* dolna granica zakresu napiec w */
                  /*  dziesiatych czesciach volta  */
    char      LC0_VMAXV;   /* gorna granica zakresu napiec w */
                  /*  dziesiatych czesciach volta  */
    };

/* Rekord opisu zlecenia funkcji SET_TIME =================================*/
struct lc0_time
    {
    unsigned char  LC0_CODE;    /* kod funkcji (6)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned int  LC0_ETIME;   /* maksymalny czas obslugi     */
                  /*  przerwania, ktore moze pojawic*/
                  /*  sie w trakcie wykonywania   */
                  /*  dlugiego pomiaru podawany w  */
                  /*  mikrosekundach        */
    };

/* Rekord opisu zlecenia funkcji WAIT_FOR_END =============================*/
struct lc0_wait
    {
    unsigned char  LC0_CODE;    /* kod funkcji (7)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_WTYPE;   /* rodzaj urzadzenia        */
    unsigned char  LC0_WMODULE;  /* numer modulu          */
    unsigned char  LC0_WNUM;    /* numer przetwornika       */
    unsigned char  LC0_WMODE;   /* tryb pracy           */
    unsigned long  LC0_WRMNUM;   /* rzeczywista liczba probek    */
    unsigned int  LC0_WREMAR;   /* rzeczywista dlugosc marginesu  */
                  /* koncowego           */
    };
               /* tryby pracy funkcji           */
#define LC0_W_WAIT   0    /* oczekiwanie              */
#define LC0_W_TEST   1    /* test konca              */
#define LC0_W_FINISHED 2    /* powiadomienie o koncu          */

/* Rekord opisu zlecenia funkcji BREAK ====================================*/
struct lc0_break
    {
    unsigned char  LC0_CODE;    /* kod funkcji (8)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_BMODE;   /* tryb pracy           */
    void interrupt (*LC0_BPROC)(void);   /* adres procedury obslugi */
                  /*  przerwania 0x1B    */
    };
              /* bity trybu pracy funkcji       */
#define LC0_BREAK_EXEC     0 /* przerwanie              */
#define LC0_BREAK_INST     1 /* instalacja Ctrl_Break...       */
#define LC0_BREAK_PREV     4 /* ...z wywolaniem poprzedniej obslugi  */
```

```
#define LC0_BREAK_UNINST    2 /* wyinstalowanie wszystkiego      */

/* Rekord opisu zlecenia funkcji DIGITAL_INPUT ============================*/
struct lc0_digital_in
    {
    unsigned char  LC0_CODE;    /* kod funkcji (9)         */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_DMODULE;  /* numer modulu          */
    unsigned char  LC0_DNUM;    /* numer portu           */
    unsigned char  LC0_DSTST;   /* typ warunku startu       */
    unsigned char  LC0_DVAL;    /* odczytana wartosc        */
    union lc0_start LC0_DSTART;   /* parametry warunku startu    */
    };

/* Rekord opisu zlecenia funkcji DIGITAL_OUTPUT ===========================*/
struct lc0_digital_out
    {
    unsigned char  LC0_CODE;    /* kod funkcji (10)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_OMODULE;  /* numer modulu          */
    unsigned char  LC0_ONUM;    /* numer portu           */
    unsigned char  LC0_OSTST;   /* typ warunku startu       */
    unsigned char  LC0_OVAL;    /* wartosc do wyslania       */
    union lc0_start LC0_OSTART;   /* parametry warunku startu    */
    };

/* Rekord opisu zlecenia funkcji CTC_WRITE ================================*/
struct lc0_ctc_write
    {
    unsigned char  LC0_CODE;    /* kod funkcji (11)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_CMODULE;  /* numer modulu          */
    unsigned char  LC0_CMODE;   /* tryb pracy funkcji       */
    unsigned char  LC0_CFUN;    /* tryb pracy kanalu        */
    unsigned int  LC0_CVAL;    /* nowa wartosc licznika      */
    };
              /* tryby pracy funkcji           */
#define LC0_SET_CTC_MODE    1    /* zaprogramuj tryb pracy kanalu  */
#define LC0_SET_COUNTER_VALUE  2    /* zaladuj nowa wartosc licznika  */
#define LC0_CTC_ENABLE     4    /* zezwolenie CTC         */
/* Rekord opisu zlecenia funkcji CTC_READ =================================*/
struct lc0_ctc_read
    {
    unsigned char  LC0_CODE;    /* kod funkcji (12)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_UMODULE;  /* numer modulu          */
    unsigned char  LC0_UNUM;    /* numer kanalu          */
    unsigned int  LC0_UVAL;    /* odczytana wartosc licznika   */
    };

/* Rekord opisu zlecenia funkcji DATA_TRANSMIT ============================*/
struct lc0_transmit
    {
    unsigned char  LC0_CODE;    /* kod funkcji (13)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_RMODE;   /* tryb przesylania        */
    int far     *LC0_RADDR;   /* adres bufora w pam. podstawowej */
    unsigned long  LC0_RLEN;    /* dlugosc bufora w pam. podst.  */
    unsigned long  LC0_RMEAS;   /* numer pierwszej probki     */
    unsigned long  LC0_RNUM;    /* liczba probek do przeslania   */
    unsigned long  LC0_RMEMA;   /* adres bufora w pamieci     */
                    /*  rozszerzonej (absolutny)   */
    unsigned long  LC0_RRNUM;   /* rzeczywista liczba probek    */

    };
              /* tryby pracy funkcji           */
#define LC0_TO_EXT_DIR     1    /* do pamieci rozszerzonej     */
#define LC0_FROM_EXT_DIR    0    /* z pamieci modulu / rozszerzonej */

/* Rekord opisu zlecenia funkcji ANALOG_INPUT =============================*/
struct lc0_analog_in
```

```
    {
    unsigned char  LC0_CODE;    /* kod funkcji (14)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_AMODULE;  /* numer modulu          */
    unsigned char  LC0_ANUM;    /* numer przetwornika       */
    unsigned int  LC0_AMODE;   /* tryb pracy funkcji       */
    unsigned char  LC0_ASTST;   /* typy warunkow startu i stopu  */
    unsigned long  LC0_APER;    /* okres probkowania        */
    unsigned int  LC0_APER2;   /* krotnosc okresu probkowania   */
    unsigned char  LC0_ACHAN;   /* liczba kanalow podstawowych   */
                    /*  (b8 okresla czy praca     */
                    /*  wielokanalowa (0) czy     */
                    /*  jednokanalowa (1))      */
    unsigned char  LC0_ACHAN2;   /* liczba kanalow dodatkowych   */
    int far     *LC0_AADDR;   /* adres buforaw pam. podstawowej */
    unsigned long  LC0_ALEN;    /* dlugosc bufora         */
    unsigned long  LC0_AMEMA;   /* adres bufora w pam. rozszerzonej*/
                    /*  (absolutny)          */
    unsigned int  LC0_ABMAR;   /* dlugosc marginesu poczatkowego */
    unsigned int  LC0_AEMAR;   /* dlugosc marginesu koncowego   */
    unsigned int  LC0_AHAND;   /* numer handler'a zbioru     */
    union lc0_start LC0_ASTART;   /* parametry warunku startu    */
    union lc0_stop LC0_ASTOP;   /* parametry warunku stopu     */
    unsigned int  LC0_ARDIV1;   /* pierwszy podzielnik zegara   */
    unsigned int  LC0_ARDIV2;   /* drugi podzielnik zegara     */
    unsigned long  LC0_ARMNUM;   /* rzeczywista liczba probek    */
    unsigned int  LC0_ARBMAR;   /* rzeczywista dlugosc marg. pocz. */
    unsigned int  LC0_AREMAR;   /* rzeczywista dlugosc marg. konc. */
    unsigned long  LC0_ARLEN;   /* rzeczywista liczba przepisanych */
                    /*  probek           */
    unsigned int  LC0_ARBUF;   /* dla pracy z buforem cyklicznym: */
                    /*  numer probki okreslajacej   */
                    /*  poczatek bufora po zakonczeniu*/
                    /*  pomiaru           */
    };

/* Rekord opisu zlecenia funkcji ANALOG_OUTPUT ============================*/
struct lc0_analog_out
    {
    unsigned char  LC0_CODE;    /* kod funkcji (15)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_NMODULE;  /* numer modulu          */
    unsigned char  LC0_NNUM;    /* numer przetwornika       */
    unsigned int  LC0_NMODE;   /* tryb pracy funkcji       */
    unsigned char  LC0_NSTST;   /* typy warunkow startu i stopu  */
    unsigned char  LC0_NCHAN;   /* liczba kanalow (b8 okresla   */
                    /*  czy praca wielokanalowa (0)  */
                    /*  czy jednokanalowa (1))    */
    unsigned long  LC0_NPER;    /* okres probkowania */
    union  {
        int far    *base_memory;
        unsigned long  extended_memory;
        }LC0_NADDR;       /* adres bufora (segment:offset dla*/
                    /*  pamieci podstawowej, absolutny*/
                    /*  dla pamieci rozszerzonej)   */
    unsigned long  LC0_NLEN;    /* dlugosc bufora w pam. podst.  */
    unsigned int  LC0_NHAND;   /* numer handler'a zbioru     */
    union lc0_start LC0_NSTART;   /* parametry warunku startu    */
    union lc0_stop LC0_NSTOP;   /* parametry warunku stopu     */
    unsigned long  LC0_NRMNUM;   /* rzeczywista liczba wyslanych  */
                    /*  probek           */
    };

/* Rekord opisu zlecenia funkcji LEAVE_DRIVER =============================*/
struct lc0_leave
    {
    unsigned char  LC0_CODE;    /* kod funkcji (16)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    };

/* Rekord opisu zlecenia funkcji INTERRUPT_SERVICE ========================*/
struct lc0_interrupt
    {
```

```
    unsigned char  LC0_CODE;    /* kod funkcji (17)        */
    char      LC0_STATUS;   /* kod odpowiedzi driver'a     */
    char      LC0_ERR_STAT;  /* dodatkowe informacje o bledach */
    unsigned char  LC0_SMODULE;  /* numer modulu          */
    void far (*LC0_SPROC)(void);  /* adres procedury obslugi     */
    int far     *LC0_SSTAT;   /* adres slowa komunikacyjnego   */
    };
#define LC0_IS_START  1    /* przerwanie wystapilo z powodu      */
                /*  rozpoczecia pomiaru          */
#define LC0_IS_ONE   2    /* przerwanie nastapilo z powodu zmierzenia*/
                /*  kolejnej serii pomiarowej       */
#define LC0_IS_END_ADC 4    /* przerwanie nastapilo z powodu      */
                /*  zakonczenia przetwarzania a/c     */
#define LC0_IS_END_DAC 8    /* przerwanie nastapilo z powodu      */
                /*  zakonczenia przetwarzania c/a     */
#define LC0_IS_BROKEN  16   /* zakonczono przetwarzanie z powodu    */
                /* wykonania funkcji BREAK         */
#define LC0_IS_SAMPLE  32   /* przerwanie nastapilo z powodu zmierzenia*/
                /* kolejnej probki            */

/* ***************** kody funkcji driver'a *******************************/
#define MODULE_INIT           0
#define GET_TOTAL_CONFIGURATION     1
#define GET_MODULE_CONFIGURATION    2
#define GET_INFO            3
#define SET_CLOCK            4
#define SET_VOLTAGE_RANGE        5
#define SET_TIME            6
#define WAIT_FOR_END          7
#define BREAK              8
#define DIGITAL_INPUT          9
#define DIGITAL_OUTPUT         10
#define CTC_WRITE            11
#define CTC_READ            12
#define DATA_TRANSMIT          13
#define ANALOG_INPUT          14
#define ANALOG_OUTPUT          15
#define LEAVE_DRIVER          16
#define INTERRUPT_SERVICE        17

/*************** numery przerwan obslugiwanych przez driver ***************/
#define LC010_16    0x99      /* LC-010-1612           */
#define LC011_08    0x90      /* LC-011-0812           */
#define LC011_16    0x91      /* LC-011-1612           */
#define LC015_16    0x92      /* LC-015-1612           */
#define LC020_08_0   0x93      /* LC-020-0812 v.0         */
#define LC020_08_2   0x94      /* LC-020-0812 v.1 i v.2      */
#define LC020_32    0x95      /* LC-020-3212           */
#define LC030_16    0x96      /* LC-030-1612           */
#define LC060_06    0x97      /* LC-060-0612           */

/******************* kody odpowiedzi funkcji driver'a ********************/
#define LC0_OK         0

/* ======= ostrzezenia ======== */
#define LC0_NON_EX_MOD     1    /* nie istniejacy(e) modul(y)   */
#define LC0_OTHER_LEN      2    /* przepisano mniejsza liczbe   */
                  /*  pomiarow           */
#define LC0_PREMATURE_END    3    /* przedwczesne zakonczenie z   */
                  /*  powodu przepelnienia bufora  */
#define LC0_IN_PROGRESS     4    /* badana transmisja jeszcze trwa */

/* ========== bledy =========== */
#define LC0_UNKN_FUNC      -1   /* nieznany kod funkcji      */
#define LC0_NO_MODULE      -2   /* brak modulu(ow)         */
#define LC0_BAD_DEV_TYP     -3   /* bledny typ urzadzenie      */
#define LC0_NONEX_DEV      -4   /* nie istnieje urzadzenie o tym  */
                  /*  numerze            */
#define LC0_BAD_FREQ      -5   /* zla czestotliwosc zegara    */
#define LC0_BAD_RANGE      -6   /* zly zakres napiec        */
#define LC0_NO_OPER       -7   /* zadna operacja nie jest     */
                  /*  wykonywana          */
#define LC0_BAD_MARGIN     -8   /* bledna dlugosc marginesu    */
                  /*  poczatkowego         */
#define LC0_BAD_BUF_ADR     -9   /* bledny adres bufora       */
#define LC0_BAD_BUF_LEN     -10   /* bledna dlugosc bufora      */
```

```
#define LC0_DEV_BUSY      -11   /* urzadzenie jest zajete     */
#define LC0_BAD_PER       -12   /* zly okres probkowania      */
#define LC0_BAD_CHAN_N     -13   /* zla liczba kanalow       */
#define LC0_BAD_CHAN      -14   /* numer nie istniejacego kanalu  */
#define LC0_BROKEN       -15   /* przerwano funkcja BREAK     */
#define LC0_INTR_NOT_INST    -16   /* procedura obslugi przerwania nie*/
                    /*  jest zainstalowana      */
#define LC0_ILL_START_CODE   -17   /* nielegalny typ warunku startu  */
#define LC0_ILL_STOP_CODE    -18   /* nielegalny typ warunku stopu  */
#define LC0_BAD_PROC      -19   /* bledny adres procedury obslugi */
                    /*  przerwania lub slowa     */
                    /*  komunikacyjnego        */
#define LC0_TOO_LONG_MARG    -20   /* margines dluzszy od bufora   */
#define LC0_ILL_START      -21   /* bledne parametry warunku startu */
#define LC0_ILL_STOP      -22   /* bledne parametry warunku stopu */
#define LC0_BAD_MNUM      -23   /* bledny numer pierwszej probki  */
#define LC0_NOT_SUPPORTED    -24   /* dla danego modulu funkcja nie  */
                    /*  jest realizowana       */
#define LC0_BAD_CTC_MODE    -25   /* bledny tryb pracy CTC      */
#define LC0_NO_PARAMS      -26   /* nie podano parametrow      */
                    /*  przetwarzania         */
#define LC0_OVERRUN       -27   /* zakonczono przetwarzanie z   */
                    /*  powodu bledu OVERRUN     */
#define LC0_NO_DMA        -28   /* z danym urzadzeniem nie jest  */
                    /*  zwiazany zaden kanal DMA   */
#define LC0_NO_IRQ        -29   /* z danym modulem nie jest    */
                    /*  zwiazane zadne przerwanie   */
#define LC0_NOT_FULLY_SUP    -30   /* zadany tryb wykonania funkcji  */
                    /*  nie jest realizowany dla   */
                    /*  danego typu modulu lub funkcja*/
                    /*  w opracowaniu         */
#define LC0_NO_EXTMEM      -31   /* brak pamieci roszerzonej    */
#define LC0_NO_SEC_FREQ     -32   /* modul nie moze wykonywac    */
                    /*  pomiarow z podwojna      */
                    /*  czestotliwoscia        */
#define LC0_INTR_INST      -33   /* procedura obslugi przerwania  */
                    /*  juz zainstalowana       */
#define LC0_BAD_PER2       -34   /* bledna wielokrotnosc okresu   */
                    /*  probkowania (0 lub 1)     */
#define LC0_BAD_MODE       -35   /* bledny tryb pracy        */
#define LC0_BAD_EXTMEM     -36   /* zly adres bufora w pamieci   */
                    /*  rozszerzonej         */
#define LC0_CTC_NOT_PROGRAMMED -37   /* zapis licznika przy niezapro-  */
                    /*  gramowanym trybie pracy    */
#define LC0_REJECTED       -38   /* za wiele rownoleglych wejsc do */
                    /*  driver'a           */

/************************** dodatkowe informacje o bledach *****************/
#define LC0_E_OK        0    /* brak dodatkowych informacji   */
#define LC0_E_NO_MODULE     -1   /* nie ma takiego modulu      */
#define LC0_E_NONEX_DEV     -2   /* nie istnieje urzadzenie o tym  */
                    /*  numerze            */
#define LC0_E_BAD_CHAN     -3   /* numer nieistniejacego kanalu  */
#define LC0_E_BAD_TIME     -4   /* zly odcinek czasu        */
#define LC0_E_BAD_DATE     -5   /* zla specyfikacja daty      */
#define LC0_E_BAD_THRE     -6   /* bledny prog wyzwalania     */
                    /*  analogowego          */
#define LC0_E_BROKEN_WAIT    -7   /* funkcja przerwana w trakcie   */
                    /*  oczekiwania na spelnienie   */
                    /*  warunku startu        */
#define LC0_E_BROKEN_RUN    -8   /* funkcja przerwana w trakcie   */
                    /*  przetwarzania         */
#define LC0_E_BAD_LEN      -9   /* zadeklarowano za duzo probek  */

/************************** kody warunkow startu ***************************/
#define LC0_SIMMED   0    /* natychmiastowy             */
#define LC0_SHARD    1    /* od sygnalu sprzetowego         */
#define LC0_SLEVEL   2    /* od poziomu sygnalu cyfrowego      */
#define LC0_SSLOPE   3    /* od zbocza sygnalu cyfrowego       */
#define LC0_SDIG_EQ   4    /* od kombinacji bitow - rowne       */
#define LC0_SDIG_NE   5    /* od kombinacji bitow - rozne       */
#define LC0_STIME    6    /* po uplynieciu okreslonego czasu     */
#define LC0_SDATE    7    /* o podanym czasie            */
#define LC0_SANALOG   8    /* od sygnalu analogowego         */

/************************** kody warunkow stopu ****************************/
```

```
#define LC0_ZSAMPLES  0x00  /* po zmierzeniu okreslonej liczby probek */
#define LC0_ZBREAK   0x10  /* po wykonaniu funkcji BREAK       */
#define LC0_ZLEVEL   0x20  /* od poziomu sygnalu cyfrowego      */
#define LC0_ZSLOPE   0x30  /* od zbocza sygnalu cyfrowego       */
#define LC0_ZDIG_EQ   0x40  /* od kombinacji bitow - rowne       */
#define LC0_ZDIG_NE   0x50  /* od kombinacji bitow - rozne       */
#define LC0_ZTIME    0x60  /* po uplynieciu okreslonego czasu     */
#define LC0_ZDATE    0x70  /* o podanym czasie            */
#define LC0_ZANALOG   0x80  /* od sygnalu analogowego         */

/************************** kody numerow modulow *************************/
#define LC0_MODA    1    /* modul A                */
#define LC0_MODB    2    /* modul B                */
#define LC0_MODC    3    /* modul C                */
#define LC0_MODD    4    /* modul D                */

/************************** maski modulow w mapie ************************/
#define LC0_MODAMAP   1    /* modul A                 */
#define LC0_MODBMAP   2    /* modul B                 */
#define LC0_MODCMAP   4    /* modul C                 */
#define LC0_MODDMAP   8    /* modul D                 */

/************************** kody typow urzadzen ****************************/
#define LC0_DINPUT   1    /* wejsciowy port cyfrowy         */
#define LC0_DOUTPUT   2    /* wyjsciowy port cyfrowy         */
#define LC0_AINPUT   3    /* przetwornik a/c             */
#define LC0_AOUTPUT   4    /* przetwornik c/a             */
#define LC0_CTC     5    /* kanal CTC               */

/************ maski trybu pracy funkcji ANALOG_INPUT i ANALOG_OUTPUT ********/
#define LC0_MOD_START      1    /* start przetwarzania       */
#define LC0_MOD_NEW_PAR     2    /* nowe parametry          */
#define LC0_MOD_SYNCHR     4    /* praca synchroniczna        */
#define LC0_MOD_ASYNCHR     0    /* praca asynchroniczna       */
#define LC0_MOD_INTR      8    /* praca bez przerwan        */
#define LC0_MOD_INTR_TYPE    16   /* bit trybu przerwan        */
#define LC0_MOD_END_INTR    0    /* przerwanie po koncu       */
                  /*  przetwarzania         */
#define LC0_MOD_ONE_INTR    16   /* przerwanie po kazdej probce   */
#define LC0_MOD_BLOCK      32   /* praca blokowa           */
#define LC0_MOD_SINGLE     0    /* praca pojedyncza         */
#define LC0_MOD_CYCL      64   /* bufor cykliczny          */
#define LC0_MOD_FILE_W     128   /* zapis do pliku          */
#define LC0_MOD_FILE_R     128   /* odczyt z pliku          */
#define LC0_MOD_MEM_W      256   /* przepisanie do pamieci     */
#define LC0_MOD_EXT_CLK     512   /* zegar zewnetrzny        */
#define LC0_MOD_EXT_MEM     1024  /* pamiec rozszerzona        */

/************* wartosci fragmentow bajtu kontrolnego 8253/8254 *************/
#define CTC_NB     0    /* kod naturalny binarny          */
#define CTC_BCD     1    /* kod BCD                */

#define CTC_MODE0    0    /* tryb 0                */
#define CTC_MODE1    2    /* tryb 1                */
#define CTC_MODE2    4    /* tryb 2                */
#define CTC_MODE3    6    /* tryb 3                */
#define CTC_MODE4    8    /* tryb 4                */
#define CTC_MODE5    10   /* tryb 5                */

#define CTC_LSB     0x10  /* tylko mlodszy bajt           */
#define CTC_MSB     0x20  /* tylko starszy bajt           */
#define CTC_BOTH    0x30  /* mlodszy - starszy           */

#define CTC_COUNT0   0x00  /* licznik 0               */
#define CTC_COUNT1   0x40  /* licznik 1               */
#define CTC_COUNT2   0x80  /* licznik 2               */
```

D O D A T E K B

d o d o k u m e n t a c j i d r i v e r ' a

m o d u ł u k o n t r o l n o - p o m i a r o w e g o

L C - 0 1 1 - 1 6 1 2

TEST.C
program przykładowy w C

```
/* TEST.C ***************************************************************/
/* Program przykladowy pokazujacy sposob wykorzystania driver'ow modulow  */
/* serii LC-011 i LC-020 firmy AMBEX.                                     */
/************************************************************************/
/* Program zostal przygotowany do kompilacji w kazdym modelu pamieci    */
/* (wymuszenie dalekich adresow przesylanych do driver'a).              */
/* W katalogu zawierajacym standardowe naglowki C musi sie znajdowac plik */
/* AMBEX-LC.H.                                                          */
/* UWAGA: Przed rozpoczeciem kompilacji nalezy sprawdzic ustawienie opcji */
/*    kompilatora:                                                      */
/*       - alingment = byte,                                            */
/*       - default char type = signed.                                  */
/************************************************************************/

#include <ambex-lc.h>      /* struktury danych i definicje stalych do */
                /* komunikacji z driver'em          */
#include <conio.h>       /* funkcja getch              */
#include <dos.h>        /* funkcja int86              */
#include <io.h>         /* dla funkcji driverinstalled       */
#include <stdio.h>       /* wyswietlanie              */
#include <stdlib.h>       /* funkcja exit              */

/******************** deklaracje procedur ***********************************/
void  askdriver(void);
void  blocktransmission(void);
void  checkdrivers(void);
void  displaybuf(int *buf, int c, int s);
void  displayconfiguration(void);
void  drivererror(char status, char err_stat);
int   driverinstalled(char *name);
void  failblocktransmission(void);
void  installbreak(void);
void  interruptedafter(void);
void  interruptedbefore(void);
void  pressanykey(void);
void  quit(void);
void  singletransmission(void);
void  transmission(unsigned char start);

/******************** definicje stalych *************************************/
#define TRUE      1
#define FALSE      0
#define BUFLEN     (8 * 20)        /* dlugosc bufora     */
#define SAMPLES     20           /* liczba probek      */
#define CHANNELS    8            /* liczba kanalow     */

/******************** deklaracje zmiennych **********************************/
int   LCinterrupt;      /* numer przerwania driver'a        */
                /* rekordy opisu poszczegolnych zlecen   */
struct lc0_init        s_init    = {MODULE_INIT};
struct lc0_total        s_total    = {GET_TOTAL_CONFIGURATION};
struct lc0_module       s_module   = {GET_MODULE_CONFIGURATION};
struct lc0_info        s_info    = {GET_INFO};
struct lc0_break        s_break    = {BREAK};
struct lc0_analog_in      s_analog_in  = {ANALOG_INPUT};
struct lc0_leave        s_leave    = {LEAVE_DRIVER};
union  REGS  r;       /* parametr dla funkcji int86       */
int   modulenum;       /* numer badanego modulu          */
                /* ------- teksty bledow          */
char  *DriverErrors[] =
{
                        /* LC0_UNKN_FUNC    -1 */
"LC0_UNKN_FUNC: Nieznany kod funkcji",
                        /* LC0_NO_MODULE    -2 */
"LC0_NO_MODULE: Bledny numer modulu",
                        /* LC0_BAD_DEV_TYP   -3 */
"LC0_BAD_DEV_TYP: Brak urzadzen danego typu",
                        /* LC0_NONEX_DEV    -4 */
"LC0_NONEX_DEV: Bledny numer urzadzenia",
                        /* LC0_BAD_FREQ     -5 */
"LC0_BAD_FREQ: Bledny okres",
                        /* LC0_BAD_RANGE    -6 */
"LC0_BAD_RANGE: Bledny zakres napiec",
                        /* LC0_NO_OPER     -7 */
"LC0_NO_OPER: Brak operacji w toku",
                        /* LC0_BAD_MARGIN    -8 */
```

```
"LC0_BAD_MARGIN: Bledna dlugosc marginesu poczatkowego",
                        /* LC0_BAD_BUF_ADR   -9 */
"LC0_BAD_BUF_ADR: Bledny adres bufora",
                        /* LC0_BAD_BUF_LEN   -10 */
"LC0_BAD_BUF_LEN: Bledna dlugosc bufora",
                        /* LC0_DEV_BUSY    -11 */
"LC0_DEV_BUSY: Urzadzenie zajete",
                        /* LC0_BAD_PER     -12 */
"LC0_BAD_PER: Za krotki okres probkowania",
                        /* LC0_BAD_CHAN_N   -13 */
"LC0_BAD_CHAN_N: Bledna liczba kanalow",
                        /* LC0_BAD_CHAN    -14 */
"LC0_BAD_CHAN: Bledny numer kanalu",
                        /* LC0_BROKEN     -15 */
"LC0_BROKEN: Przetwarzanie przerwane funkcja BREAK",
                        /* LC0_INTR_NOT_INST  -16 */
"LC0_INTR_INST: Procedura obslugi przerwania nie jest zainstalowana",
                        /* LC0_ILL_START_CODE -17 */
"LC0_ILL_START_CODE: Nielegalny sposob startu",
                        /* LC0_ILL_STOP_CODE  -18 */
"LC0_ILL_STOP_CODE: Nielegalny sposob stopu",
                        /* LC0_BAD_PROC    -19 */
"LC0_BAD_PROC: Bledny adres procedury obslugi przerwania",
                        /* LC0_TOO_LONG_MARG  -20 */
"LC0_TOO_LONG_MARG: Za dlugi margines poczatkowy",
                        /* LC0_ILL_START    -21 */
"LC0_ILL_START: Bledne parametry warunku startu",
                        /* LC0_ILL_STOP    -22 */
"LC0_ILL_STOP: Bledne parametry warunku stopu",
                        /* LC0_BAD_MNUM    -23 */
"LC0_BAD_MNUM: Bledny numer pierwszej probki",
                        /* LC0_NOT_SUPPORTED  -24 */
"LC0_NOT_SUPPORTED: Funkcja nie jest realizowana",
                        /* LC0_BAD_CTC_MODE  -25 */
"LC0_BAD_CTC_MODE: Bledny tryb pracy CTC",
                        /* LC0_NO_PARAMS    -26 */
"LC0_NO_PARAMS: Nie podano parametrow przetwarzania",
                        /* LC0_OVERRUN     -27 */
"LC0_OVERRUN: Blad OVERRUN",
                        /* LC0_NO_DMA     -28 */
"LC0_NO_DMA: Urzadzenie nie jest podlaczone do DMA",
                        /* LC0_NO_IRQ     -29 */
"LC0_NO_IRQ: Z modulem nie jest zwiazane zadne przerwanie",
                        /* LC0_NOT_FULLY_SUP  -30 */
"LC0_NOT_FULLY_SUP: Funkcja w opracowaniu",
                        /* LC0_NO_EXTMEM    -31 */
"LC0_NO_EXTMEM: Brak pamieci dodatkowej",
                        /* LC0_NO_SEC_FREQ   -32 */
"LC0_NO_SEC_FREQ: Modul ma tylko jedna czestotliwosc",
                        /* LC0_INTR_INST    -33 */
"LC0_INTR_INST: Procedura obslugi przerwania juz zainstalowana",
                        /* LC0_BAD_PER2    -34 */
"LC0_BAD_PER2: Bledna wielokrotnosc okresu probkowania",
                        /* LC0_BAD_MODE    -35 */
"LC0_BAD_MODE: Bledny tryb pracy",
                        /* LC0_BAD_EXTMEM   -36 */
"LC0_BAD_EXTMEM: Bledny adres bufora w pamieci rozszerzonej",
                        /* LC0_NOT_PROGRAMMED -37 */
"LC0_CTC_NOT_PROGRAMMED: Zapis licznika przy niezaprogramowanym trybie pracy",
                        /* LC0_REJECTED    -38 */
"LC0_REJECTED: Za duzo jednoczesnych odwolan do driver'a",
};

                /* ------ teksty ostrzezen        */
char  *DriverWarnings[] =
{
                        /* LC0_NON_EX_MOD    1 */
"LC0_NON_EX_MOD: Zazadano inicjalizacji nieistniejacych modulow",
                        /* LC0_OTHER_LEN    2 */
"LC0_OTHER_LEN: Przepisano mniej probek niz zazadano",
                        /* LC0_PREMATURE_END  3 */
"LC0_PREMATURE_END: Zakonczenie operacji z powodu przepelnienia bufora",
                        /* LC0_IN_PROGRESS   4 */
"LC0_IN_PROGRESS: Badana transmisja jeszcze trwa",
};
```

```
                /* ------- teksty informacji dodatkowych */
char  *DriverAdditionalErrors[] =
{
                        /* LC0_E_NO_MODULE   -1 */
"LC0_E_NO_MODULE: Nie ma takiego modulu",
                        /* LC0_E_NONEX_DEV   -2 */
"LC0_E_NONEX_DEV: Nie istnieje urzadzenie o tym numerze",
                        /* LC0_E_BAD_CHAN    -3 */
"LC0_E_BAD_CHAN: Numer nieistniejacego kanalu",
                        /* LC0_E_BAD_TIME    -4 */
"LC0_E_BAD_TIME: Zly odcinek czasu",
                        /* LC0_E_BAD_DATE    -5 */
"LC0_E_BAD_DATE: Zla specyfikacja daty",
                        /* LC0_E_BAD_THRE    -6 */
"LC0_E_BAD_THRE: Bledny prog wyzwalania analogowego",
                        /* LC0_E_BROKEN_WAIT  -7 */
"LC0_E_BROKEN_WAIT: Przerwanie w trakcie oczekiwania na warunek startu",
                        /* LC0_E_BROKEN_RUN   -8 */
"LC0_E_BROKEN_RUN: Przerwanie w trakcie przetwarzania",
                        /* LC0_E_BAD_LEN    -9 */
"LC0_E_BAD_LEN: Zadeklarowano za duzo probek",
};

/*************************************************************************/
/*************************************************************************/
void main(void)
{
    checkdrivers();     /* rozpoznanie zainstalowanego driver'a  */
    askdriver();      /* odpytanie driver'a o konfiguracje    */
    displayconfiguration(); /* wyswietlenie konfiguracji modulu i   */
                /* toru a/c                */
    installbreak();     /* zainstalowanie procedury obslugi    */
                /* przerwania generowanego przez Ctrl-Break*/
    pressanykey();     /* oczekiwanie na operatora        */
    blocktransmission();  /* wykonanie transmisji blokowej +     */
                /* wyswietlenie zmierzonych wartosci    */
    failblocktransmission();/* wykonanie blednej transmisji blokowej  */
    interruptedbefore();  /* wykonanie transmisji blokowej przerwanej*/
                /* przez operatora przed startem      */
    interruptedafter();   /* wykonanie transmisji blokowej przerwanej*/
                /* przez operatora po starcie       */
    singletransmission();  /* wykonanie pomiaru w trybie pojedynczym */
    quit();         /* zakonczenie programu          */
}

/*************************************************************************/
/* Przeznaczenie:                              */
/*  Rozpoznanie, ktory driver (a co za tym idzie ktory modul) jest    */
/*  zainstalowany.                             */
/* Sposob:                                    */
/*  Przez sprawdzenie zainstalowania kazdego z potencjalnych driver'ow.  */
/* Uwagi:                                    */
/*  Funkcja dodatkowo wyswietla komunikat o testowanym module i podstawia */
/*  wlasciwa wartosc na zmienna LCinterrupt (wlasciwy numer przerwania  */
/*  driver'a). Jezeli zaden driver nie jest zainstalowany to po      */
/*  wyswietleniu wlasciwego komunikaru program konczy prace.       */
/*************************************************************************/
void checkdrivers(void)
{
    clrscr();
    gotoxy(1, 1);
    if(driverinstalled("LC1116^^"))
      {
      printf("Test modulu LC-011-1612\n");
      LCinterrupt = LC011_16;
      }
    else if(driverinstalled("LC2008^2"))
      {
      printf("Test modulu LC-020-0812\n");
      LCinterrupt = LC020_08_2;
      }
    else if(driverinstalled("LC2032^^"))
      {
      printf("Test modulu LC-020_3212\n");
      LCinterrupt = LC020_32;
      }
```

```
    else
      {
      printf("Driver nie jest zainstalowany!\n");
      exit(1);
      }
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Sprawdzenie obecnosci drivera.                       */
/* Sposob:                                */
/*  Przez probe otwarcia urzadzenia o nazwie okreslonej przez driver.   */
/* Parametry:                                */
/*  name - nazwa urzadzenia                          */
/* Wartosc:                                */
/*  TRUE - driver jest zainstalowany                    */
/*  FALSE - driver nie jest zainstalowany                  */
/*************************************************************************/
int driverinstalled(char *name)
{
    int   hd;

    hd = open(name, 0);
    if(hd == -1)
      return(FALSE);
    else
      {
      close(hd);
      return(TRUE);
      }
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Rozpoznanie konfiguracji badanego modulu.              */
/* Sposob:                                */
/*  Przez wykorzystanie funkcji driver'a GET_TOTAL_CONFIGURATION,     */
/*  GET_MODULE_CONFIGURATION, GET_INFO.                  */
/* Uwagi:                                */
/*  Funkcja nadaje wartosc zmiennej modulenum (numer badanego modulu)   */
/*  wypelnia struktury total, module, info i inicjalizuje zainstalowane  */
/*  moduly.                                */
/*************************************************************************/
void askdriver(void)
{
    r.x.dx = FP_SEG(&s_total);
    r.x.di = FP_OFF(&s_total);
    int86(LCinterrupt, &r, &r);       /* GET_TOTAL_CONFIGURATION */

                /* inicjalizacja zainstalowanych modulow  */
    s_init.LC0_IMODULE = s_total.LC0_TONF & 0xF;
    r.x.dx = FP_SEG(&s_init);
    r.x.di = FP_OFF(&s_init);
    int86(LCinterrupt, &r, &r);           /* MODULE_INIT   */

                  /* sprawdzenie, ktory modul jest  */
                  /*  zainstalowany: A, B, C czy D */
    for(modulenum = 1; modulenum <= 4; modulenum++)
      if(s_total.LC0_TONF & (1 << (modulenum - 1)))
        break;

                  /* spytanie o konfiguracje modulu */
    s_module.LC0_MMODULE = modulenum;
    r.x.dx = FP_SEG(&s_module);
    r.x.di = FP_OFF(&s_module);
    int86(LCinterrupt, &r, &r);       /* GET_MODULE_CONFIGURATION*/

                  /* spytanie o konfiguracje toru a/c*/
    s_info.LC0_GMODULE = modulenum;
    s_info.LC0_GTYPE  = LC0_AINPUT;
    s_info.LC0_GNUM  = 1;
    r.x.dx = FP_SEG(&s_info);
    r.x.di = FP_OFF(&s_info);
    int86(LCinterrupt, &r, &r);           /* GET_INFO    */
}
```

```
/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wyswietlenie konfiguracji badanego modulu i jego toru a/c.      */
/* Sposob:                                    */
/*  Przez wykorzystanie informacji zawartych w strukturach module i info. */
/*************************************************************************/
void displayconfiguration(void)
{
    int   i;
                  /* konfiguracja modulu       */
    printf("\n---------- Konfiguracja modulu:\n\n");
    printf("Adres modulu: %04X (hex)\n", s_module.LC0_MBASE1);
    printf("Liczba przetwornikow a/c: %d\n", s_module.LC0_MIAD);
    printf("Liczba przetwornikow c/a: %d\n", s_module.LC0_MIDA);
    printf("Liczba portow cyfrowych wejsciowych: %d\n",
        s_module.LC0_MIDI);
    printf("Liczba portow cyfrowych wyjsciowych: %d\n",
        s_module.LC0_MIDO);
    printf("Czestotliwosc zegara modulu: %d MHz\n",
                       s_module.LC0_MCLOCK / 1000);

                  /* konfiguracja toru a/c      */
    printf("\n---------- Konfiguracja toru a/c:\n\n");
    printf("Liczba kanalow: %d\n", s_info.LC0_GCHAN);
    printf("Rozdzielczosc: %d bitow\n", s_info.LC0_GRES);
    printf("Zakres napiec: %.1f..%.1f V\n",
        (float)s_info.LC0_GMINV / 10,
        (float)s_info.LC0_GMAXV / 10);
    if(s_info.LC0_GDMA == 0xFF)
      printf("Tor a/c nie jest podlaczony do kanalow DMA\n");
    else
      printf("Numer kanalu DMA podlaczonego do toru a/c: %d\n",
                             s_info.LC0_GDMA);
    printf("Minimalne okresy probkowania [æs]:\n");
    printf("   ");
    for(i = 0; i < s_info.LC0_GCHAN; i++)
      {
      printf("%.1f", (float)s_info.LC0_GMINP[i] / 10);
      if(i < s_info.LC0_GCHAN - 1)
        {
        printf(", ");
        if((i + 1) % 8 == 0)
          {
          printf("\n");    /* zlamanie linii co 8 wielkosci  */
          printf("   ");
          }
        }
      }
    printf("\n");
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Zainstalowanie procedury obslugi przerwania generowanego przez    */
/*  Ctrl-Break.                                  */
/* Sposob:                                    */
/*  Przez wywolanie funkcji BREAK.                     */
/*************************************************************************/
void installbreak(void)
{
    s_break.LC0_BMODE = LC0_BREAK_INST;
    s_break.LC0_BPROC = NULL;        /* podlozona zostanie   */
                      /*  procedura driver'a  */
    r.x.dx = FP_SEG(&s_break);
    r.x.di = FP_OFF(&s_break);
    int86(LCinterrupt, &r, &r);           /* BREAK      */
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Przerwa miedzy kolejnymi czesciami programu - oczekiwanie na reakcje */
/*  operatora.                                  */
/*************************************************************************/
void pressanykey(void)
{
    printf("\nNacisnij dowolny klawisz . . .");
```

```
    getch();
    clrscr();
    gotoxy(1, 1);
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wykonanie poprawnej transmisji blokowej i wyswietlenie zmierzonych  */
/*  wartosci.                                   */
/* Sposob:                                     */
/*  Przez wykonanie funkcji transmission.                */
/*************************************************************************/
void blocktransmission(void)
{
    printf("---------- Poprawne przetwarzanie blokowe\n\n");
    s_analog_in.LC0_APER = (long)s_info.LC0_GMINP[CHANNELS - 1];
    transmission(LC0_SIMMED);
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wykonanie blednej transmisji blokowej.               */
/* Sposob:                                     */
/*  Przez wykonanie funkcji ANALOG_INPUT z okresem probkowania mniejszym */
/*  niz minimalny wskazany przez driver.                */
/*************************************************************************/
void failblocktransmission(void)
{
    printf("---------- Bledne przetwarzanie blokowe\n\n");
    s_analog_in.LC0_APER = (long)s_info.LC0_GMINP[CHANNELS - 1] - 1;
    transmission(LC0_SIMMED);
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wykonanie poprawnej transmisji blokowej ale przerwanej przez operatora*/
/*  (Ctrl-Break) w trakcie czekania na spelnienie warunku startu.     */
/* Sposob:                                     */
/*  Przez wykonanie funkcji ANALOG_INPUT z warunkiem startu LC0_STIME   */
/*  (start po okreslonym czasie) i parametrem tego warunku - 1000s.    */
/*************************************************************************/
void interruptedbefore(void)
{
    printf("---------- Przetwarzanie blokowe z oczekiwaniem 1000s\n");
    printf("---------- Oczekiwanie nalezy przerwac Ctrl-Break\n\n");
    s_analog_in.LC0_APER = (long)s_info.LC0_GMINP[CHANNELS - 1];
    s_analog_in.LC0_ASTART.time = 1000;
    transmission(LC0_STIME);
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wykonanie poprawnej transmisji blokowej ale przerwanej przez operatora*/
/*  (Ctrl-Break) w trakcie pomiaru.                    */
/* Sposob:                                     */
/*  Przez wykonanie funkcji ANALOG_INPUT z okresem probkowania 100s.   */
/*************************************************************************/
void interruptedafter(void)
{
    printf("---------- Przetwarzanie blokowe ",
        "z okresem probkowania 100s\n");
    printf("---------- Pomiar nalezy przerwac Ctrl-Break\n\n");
    s_analog_in.LC0_APER = 1000000000;
    transmission(LC0_SIMMED);
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wykonanie pomiaru bloku probek za pomoca tranmsisji pojedynczej, przy */
/*  czym caly blok ma byc zmierzony po 10s od startu.          */
/* Sposob:                                     */
/*  Przez wykonanie funkcji ANALOG_INPUT.                */
/* Parametry:                                  */
/* Wartosc:                                    */
/* Uwagi:                                      */
/*************************************************************************/
```

```
void singletransmission(void)
{
    int   buf[BUFLEN];      /* bufor na probki         */
    int   i,
        j;

    printf("---------- Przetwarzanie pojedyncze - po 5 sekundach\n\n");
    s_analog_in.LC0_AMODULE = modulenum;
    s_analog_in.LC0_ANUM = 1;
    s_analog_in.LC0_AMODE = LC0_MOD_START |
              LC0_MOD_NEW_PAR |
              LC0_MOD_SINGLE;
               /* praca wielokanalowa, CHANNELS kanalow  */
    s_analog_in.LC0_ACHAN = CHANNELS;
    s_analog_in.LC0_AADDR = (int far *)buf; /* konieczna konwersja na */
                        /*  daleki adres     */
    s_analog_in.LC0_ALEN = CHANNELS;    /* dlugosc bufora (tylko  */
                        /*  na jeden pomiar)   */
    s_analog_in.LC0_ASTST = LC0_STIME;
    s_analog_in.LC0_ASTART.time = 5;    /* start po 5 sekundach  */
    r.x.dx = FP_SEG(&s_analog_in);
    r.x.di = FP_OFF(&s_analog_in);
    int86(LCinterrupt, &r, &r);           /* ANALOG_INPUT  */
    s_analog_in.LC0_ASTST = LC0_SIMMED;
    i = 1;
    s_analog_in.LC0_AADDR = (int far *)&buf[i++ * CHANNELS];
    r.x.dx = FP_SEG(&s_analog_in);
    r.x.di = FP_OFF(&s_analog_in);
    int86(LCinterrupt, &r, &r);           /* ANALOG_INPUT  */
    s_analog_in.LC0_AMODE &= ~LC0_MOD_NEW_PAR;   /* zgaszenie bitu */
                          /* LC0_MOD_NEW_PAR */
    for(j = 0; j < 18; j++)
      {
      s_analog_in.LC0_AADDR = (int far *)&buf[i++ * CHANNELS];
      r.x.dx = FP_SEG(&s_analog_in);
      r.x.di = FP_OFF(&s_analog_in);
      int86(LCinterrupt, &r, &r);         /* ANALOG_INPUT  */
      }
    displaybuf(buf, CHANNELS, SAMPLES);
    pressanykey();
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wykonanie transmisji blokowej.                       */
/* Sposob:                                */
/*  Przez wykonanie funkcji ANALOG_INPUT.                 */
/* Parametry:                                */
/*  start - typ warunku startu                        */
/* Wartosc:                                */
/* Uwagi:                                */
/*  Funkcja wywolujaca musi ustawic okres probkowania i parametry warunku */
/*  startu.                                */
/*************************************************************************/
void transmission(unsigned char start)
{
    int   buf[BUFLEN];      /* bufor na probki         */

    printf
    (
    "Pomiar blokowy, %d kanalow, %d probek, okres probkowania %.1f æs:\n",
    CHANNELS, SAMPLES, (float)s_analog_in.LC0_APER / 10
    );
    switch(start)
      {
      case LC0_SIMMED:
        printf("Warunek startu: natychmiast\n");
        break;
      case LC0_STIME:
        printf("Warunek startu: upływ %lu sekund\n",
                        s_analog_in.LC0_ASTART.time);
        break;
                /* pozostale warunki nie sa obslugiwane bo */
                /*  w programie nie sa wykorzystywane   */
      }
    s_analog_in.LC0_AMODULE = modulenum;
```

```
    s_analog_in.LC0_ANUM = 1;
    s_analog_in.LC0_AMODE = LC0_MOD_START |
               LC0_MOD_NEW_PAR |
               LC0_MOD_SYNCHR |
               LC0_MOD_BLOCK;
            /* stop po zmierzeniu okreslonej liczby probek   */
    s_analog_in.LC0_ASTST = start + LC0_ZSAMPLES;
                /* praca wielokanalowa, CHANNELS kanalow  */
    s_analog_in.LC0_ACHAN = CHANNELS;
    s_analog_in.LC0_AADDR = (int far *)buf; /* konieczna konwersja na */
                        /*  daleki adres     */
    s_analog_in.LC0_ALEN = BUFLEN; /* dlugosc bufora         */
    s_analog_in.LC0_ABMAR = 0;
    s_analog_in.LC0_AEMAR = 0;   /* oba marginesy zerowe      */
                    /* calkowita liczba probek     */
    s_analog_in.LC0_ASTOP.samples = SAMPLES * CHANNELS;
    r.x.dx = FP_SEG(&s_analog_in);
    r.x.di = FP_OFF(&s_analog_in);
    int86(LCinterrupt, &r, &r);           /* ANALOG_INPUT  */
    if(s_analog_in.LC0_STATUS != LC0_OK)
      drivererror(s_analog_in.LC0_STATUS, s_analog_in.LC0_ERR_STAT);
    else
      displaybuf(buf, CHANNELS, SAMPLES);
    pressanykey();
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wyswietlenie bufora z danymi pomiarowymi.              */
/* Parametry:                                  */
/*  buf - adres bufora                             */
/*  c  - liczba kanalow                            */
/*  s  - liczba probek na kanal                       */
/*************************************************************************/
void displaybuf(int *buf, int c, int s)
{
    int   i,
        j;

    for(i = 0; i < s; i++)
      {
      printf("%2d: ", i + 1);
      for(j = 0; j < c; j++)
        printf("%04X ", buf[i * c + j]);
      printf("\n");
      }
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Wypisanie komunikatow o bledzie / ostrzezeniu / dodatkowej informacji */
/*  bledzie.                                  */
/* Parametry:                                   */
/*  status  - LC0_STATUS                            */
/*  err_stat - LC0_ERR_STAT                          */
/*************************************************************************/
void drivererror(char status, char err_stat)
{
    if(status > 0)
      printf("--------- Ostrzezenie:\n%s\n",
          DriverWarnings[status - 1]);
    else
      printf("--------- Blad:\n%s\n", DriverErrors[-status - 1]);
    if(err_stat < 0)
      printf("--------- Informacje dodatkowe:\n%s\n",
          DriverAdditionalErrors[-err_stat - 1]);
}

/*************************************************************************/
/* Przeznaczenie:                                */
/*  Zakonczenie pracy programu.                        */
/* Sposob:                                  */
/*  Przez wykonanie funkcji exit. Wczesniej - rozstanie sie z driver'em za*/
/*  pomoca funkcji LEAVE_DRIVER.                       */
/*************************************************************************/
void quit(void)
```

```
{
    r.x.dx = FP_SEG(&s_leave);
    r.x.di = FP_OFF(&s_leave);
    int86(LCinterrupt, &r, &r);           /* LEAVE_DRIVER  */

    printf("Dziekuje, to wszystko!\n");
    exit(0);
}
```

D O D A T E K C

d o d o k u m e n t a c j i d r i v e r ' a

m o d u ł u k o n t r o l n o - p o m i a r o w e g o

L C - 0 1 1 - 1 6 1 2

AMBEX - LC.PAS

struktury danych i stałe dla Pascala

```
(*************************************************************************)
(*                                       *)
(* definicje struktur danych i stalych dla wspolpracy z driver'ami modulow *)
(* analogowych serii LC-... z poziomu jezyka Pascal            *)
(*                                       *)
(*************************************************************************)

(*************** REKORDY OPISOW ZLECEN FUNKCJI DRIVER'A ******************)

(************** struktury opisujace warunki startu i stopu ****************)
type
lc0_cond_level =        (* warunek "poziom sygnalu cyfrowego"   *)
    record
      mod_nr:   byte;      (* numer modulu          *)
      port_nr:  byte;      (* numer portu           *)
      inp_nr:   byte;      (* numer wejscia          *)
      value:   byte;      (* oczekiwana wartosc       *)
    end;

(*-----------------------------------------------------------------------*)
type
lc0_cond_slope =        (* warunek "zbocze sygnalu cyfrowego"   *)
    record
      mod_nr:   byte;      (* numer modulu          *)
      port_nr:  byte;      (* numer portu           *)
      inp_nr:   byte;      (* numer wejscia          *)
      slope:   byte;      (* rodzaj zbocza          *)
    end;

const
                (* kody rodzajow zbocza:          *)
LC0_SLOPE_COND_UP    = 1;  (* narastajace               *)
LC0_SLOPE_COND_DOWN   = 0;  (* opadajace                *)

(*-----------------------------------------------------------------------*)
type
lc0_cond_dig =     (* warunek "kominacja sygnalow cyfrowych"     *)
    record
      mod_nr:   byte;      (* numer modulu          *)
      port_nr:  byte;      (* numer portu           *)
      mask:    byte;      (* maska aktywnych wejsc      *)
      pattern:  byte;      (* testowany wzorzec        *)
    end;


(*-----------------------------------------------------------------------*)
type
lc0_cond_date =     (* warunek "data"                *)
    record
      second:   byte;      (* sekunda             *)
      minute:   byte;      (* minuta             *)
      hour:    byte;      (* godzina             *)
      day:    byte;      (* dzien miesiaca         *)
    end;

(*-----------------------------------------------------------------------*)
type
lc0_cond_analog =        (* warunek "sygnal analogowy"       *)
    record
      mod_nr:   byte;      (* numer modulu          *)
      converter: byte;      (* nr przetwornika         *)
      channel:  byte;      (* numer kanalu (b1..b7),     *)
                  (*  kierunek przekroczenia (b8)  *)
      level:   integer;    (* prog wyzwalania         *)
    end;

const
            (* kody kierunku przekroczenia progu:       *)
LC0_ANALOG_COND_UP   = $80; (* w kierunku wiekszych wartosci      *)
LC0_ANALOG_COND_DOWN  = 0;  (* w kierunku mniejszych wartosci     *)
LC0_ANALOG_COND_MASK  = $80; (* maska kodu kierunku           *)

(*-----------------------------------------------------------------------*)
type
lc0_start =           (* =========== warunki startu ============*)
    record
```

```
      case integer of
        1: (level:   lc0_cond_level);    (* LC0_SLEVEL *)
        2: (slope:   lc0_cond_slope);    (* LC0_SSLOPE *)
        3: (dig_eq:   lc0_cond_dig);     (* LC0_SDIG_EQ *)
        4: (dig_ne:   lc0_cond_dig);     (* LC0_SDIG_NE *)
        5: (time:    longint);        (* LC0_STIME  *)
        6: (date:    lc0_cond_date);     (* LC0_SDATE  *)
        7: (analog:   lc0_cond_analog)    (* LC0_SANALOG *)
    end;

type
lc0_stop =           (* ========== warunki stopu ==============*)
    record
      case integer of
        1: (samples:  longint);        (* LC0_ZSAMPLES *)
        2: (level:   lc0_cond_level);    (* LC0_ZLEVEL  *)
        3: (slope:   lc0_cond_slope);    (* LC0_ZSLOPE  *)
        4: (dig_eq:   lc0_cond_dig);     (* LC0_ZDIG_EQ *)
        5: (dig_ne:   lc0_cond_dig);     (* LC0_ZDIG_NE *)
        6: (time:    longint);        (* LC0_ZTIME  *)
        7: (date:    lc0_cond_date);     (* LC0_ZDATE  *)
        8: (analog:   lc0_cond_analog)    (* LC0_ZANALOG *)
    end;

(*=============================================================================*)

(* Rekord opisu zlecenia funkcji MODULE_INIT ============================*)
type
lc0_init =
    record
      LC0_CODE:      byte;  (* kod funkcji (0)         *)
      LC0_STATUS:     shortint;    (* kod odpowiedzi driver'a *)
      LC0_ERR_STAT:    shortint;    (* dodatkowe informacje o *)
                        (* bledach         *)
      LC0_IMODULE:    byte;  (* mapa modulow          *)
    end;

(* Rekord opisu zlecenia funkcji GET_TOTAL_CONFIGURATION ================*)
type
lc0_total =
    record
      LC0_CODE:      byte;  (* kod funkcji (1)         *)
      LC0_STATUS:     shortint;    (* kod odpowiedzi driver'a *)
      LC0_ERR_STAT:    shortint;    (* dodatkowe informacje o *)
                        (* bledach         *)
      LC0_TONF:      byte;  (* konfiguracja modulow      *)
      LC0_TIAD:      byte;  (* przetworniki a/c        *)
      LC0_TIDA:      byte;  (* przetworniki c/a        *)
      LC0_TCTC:      byte;  (* kanaly CTC           *)
      LC0_TIDI:      byte;  (* porty wejsc cyfrowych      *)
      LC0_TIDO:      byte;  (* porty wyjsc cyfrowych      *)
      LC0_TMEMA:     longint;(* adres bufora w pamieci     *)
                    (*  rozszerzonej - absolutny   *)
      LC0_TMEML:     longint;(* dlugosc bufora w pamieci    *)
                    (*  rozszerzonej (w probkach;   *)
                    (*  0 - brak pamieci rozszerzonej)*)
    end;

(* Rekord opisu zlecenia funkcji  GET_MODULE_CONFIGURATION ===============*)
type
lc0_module =
    record
      LC0_CODE:      byte;  (* kod funkcji (2)         *)
      LC0_STATUS:     shortint;    (* kod odpowiedzi driver'a *)
      LC0_ERR_STAT:    shortint;    (* dodatkowe informacje o *)
                        (* bledach         *)
      LC0_MMODULE:    byte;  (* numer modulu          *)
      LC0_MBASE1:     word;  (* adres bazowy pakietu 1     *)
      LC0_MBASE2:     word;  (* adres bazowy pakietu 2     *)
      LC0_MIAD:      byte;  (* przetworniki a/c        *)
      LC0_MIDA:      byte;  (* przetworniki c/a        *)
      LC0_MCTC:      byte;  (* kanaly CTC           *)
      LC0_MIDI:      byte;  (* porty wejsc cyfrowych      *)
      LC0_MIDO:      byte;  (* porty wyjsc cyfrowych      *)
      LC0_MCLOCK:     word;  (* czestotliwosc zegara w kHz   *)
      LC0_MINT:      byte;  (* numer przerwania (programowy)  *)
```

```
    end;

(* Rekord opisu zlecenia funkcji GET_INFO ================================*)
type
lc0_info =
    record
      LC0_CODE:      byte;  (* kod funkcji (3)         *)
      LC0_STATUS:     shortint;    (* kod odpowiedzi driver'a *)
      LC0_ERR_STAT:    shortint;    (* dodatkowe informacje o *)
                        (* bledach         *)
      LC0_GTYPE:     byte;  (* rodzaj urzadzenia        *)
      LC0_GMODULE:    byte;  (* numer modulu          *)
      LC0_GNUM:      byte;  (* numer przetwornika/portu/    *)
                    (*           ukladu CTC *)
      LC0_GCHAN:     byte;  (* liczba kanalow         *)
      LC0_GRES:      byte;  (* liczba bitow przetwornika    *)
      LC0_GTIME:     word;  (* czas konwersji przetwornika w ns*)
      LC0_GMINV:     shortint;    (* dolna granica zakresu  *)
                        (*  napiec w dziesiatych *)
                        (*  czesciach volta  *)
      LC0_GMAXV:     shortint;    (* gorna granica zakresu  *)
                        (*  napiec w dziesiatych *)
                        (*  czesciach volta  *)
      LC0_GDMA:      byte;  (* numer kanalu DMA        *)
                    (* minimalne okresy probkowania  *)
      LC0_GMINP:     array [1..32] of word;
    end;

(* Rekord opisu zlecenia funkcji SET_CLOCK ===============================*)
type
lc0_clock =
    record
      LC0_CODE:      byte;  (* kod funkcji (4)         *)
      LC0_STATUS:     shortint;    (* kod odpowiedzi driver'a *)
      LC0_ERR_STAT:    shortint;    (* dodatkowe informacje o *)
                        (* bledach         *)
      LC0_LCLOCK:     word;  (* czestotliowsc zegara w kHz   *)
    end;

(* Rekord opisu zlecenia funkcji SET_VOLTAGE_RANGE =======================*)
type
lc0_volt =
    record
      LC0_CODE:      byte;  (* kod funkcji (5)         *)
      LC0_STATUS:     shortint;    (* kod odpowiedzi driver'a *)
      LC0_ERR_STAT:    shortint;    (* dodatkowe informacje o *)
                        (* bledach         *)
      LC0_VTYPE:     byte;  (* rodzaj urzadzenia        *)
      LC0_VMODULE:    byte;  (* numer modulu          *)
      LC0_VNUM:      byte;  (* numer przetwornika       *)
      LC0_VMINV:     shortint;    (* dolna granica zakresu  *)
                        (*  napiec w dziesiatych *)
                        (*  czesciach volta  *)
      LC0_VMAXV:     shortint;    (* gorna granica zakresu  *)
                        (*  napiec w dziesiatych *)
                                        (*   czesciach volta   *)
        end;

(* Rekord opisu zlecenia funkcji  SET_TIME ================================*)
type
lc0_time =
        record
            LC0_CODE:           byte;   (* kod funkcji (6)                 *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                        (* bledach                 *)
            LC0_ETIME:          word;   (* maksymalny czas obslugi         *)
                                        (*   przerwania, ktore moze pojawic*)
                                        (*   sie w trakcie wykonywania     *)
                                        (*   dlugiego pomiaru podawany w   *)
                                        (*   mikrosekundach                *)
        end;

(* Rekord opisu zlecenia funkcji  WAIT_FOR_END ============================*)
type
lc0_wait =
```

```
        record
            LC0_CODE:           byte;   (* kod funkcji (7)                 *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_WTYPE:          byte;   (* rodzaj urzadzenia               *)
            LC0_WMODULE:        byte;   (* numer modulu                    *)
            LC0_WNUM:           byte;   (* numer przetwornika              *)
            LC0_WMODE:          byte;   (* tryb pracy                      *)
            LC0_WRMNUM:         longint;(* rzeczywista liczba probek       *)
            LC0_WREMAR:         word;   (* rzeczywista dlugosc marginesu   *)
                                        (*   koncowego                     *)
        end;

const
                                (* tryby pracy funkcji                     *)
LC0_W_WAIT      = 0;            (* oczekiwanie                             *)
LC0_W_TEST      = 1;            (* test konca                              *)
LC0_W_FINISHED  = 2;            (* powiadomienie o koncu                   *)

(* Rekord opisu zlecenia funkcji  BREAK ===================================*)
type
lc0_break =
        record
            LC0_CODE:           byte;   (* kod funkcji (8)                 *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_BMODE:          byte;   (* tryb pracy                      *)
            LC0_BPROC:          pointer;(* adres procedury obslugi         *)
                                        (*   przerwania $1B                *)
        end;

const
                                (* bity trybu pracy funkcji                *)
LC0_BREAK_EXEC          = 0;    (* przerwanie                              *)
LC0_BREAK_INST          = 1;    (* instalacja Ctrl_Break...               *)
LC0_BREAK_PREV          = 4;    (* ...z wywolaniem poprzedniej obslugi     *)
LC0_BREAK_UNINST        = 2;    (* wyinstalowanie wszystkiego              *)

(* Rekord opisu zlecenia funkcji  DIGITAL_INPUT ===========================*)
type
lc0_digital_in =
        record
            LC0_CODE:           byte;   (* kod funkcji (9)                 *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_DMODULE:        byte;   (* numer modulu                    *)
            LC0_DNUM:           byte;   (* numer portu                     *)
            LC0_DSTST:          byte;   (* typ warunku startu              *)
            LC0_DVAL:           byte;   (* odczytana wartosc               *)
            LC0_DSTART:         lc0_start;      (* parametry warunku startu*)
        end;

(* Rekord opisu zlecenia funkcji  DIGITAL_OUTPUT ==========================*)
type
lc0_digital_out =
        record
            LC0_CODE:           byte;   (* kod funkcji (10)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_OMODULE:        byte;   (* numer modulu                    *)
            LC0_ONUM:           byte;   (* numer portu                     *)
            LC0_OSTST:          byte;   (* typ warunku startu              *)
            LC0_OVAL:           byte;   (* wartosc do wyslania             *)
            LC0_OSTART:         lc0_start;      (* parametry warunku startu*)
        end;

(* Rekord opisu zlecenia funkcji  CTC_WRITE ===============================*)
type
lc0_ctc_write =
        record
            LC0_CODE:           byte;   (* kod funkcji (11)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
```

```
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_CMODULE:        byte;   (* numer modulu                    *)
            LC0_CMODE:          byte;   (* tryb pracy funkcji              *)
            LC0_CFUN:           byte;   (* tryb pracy kanalu               *)
            LC0_CVAL:           word;   (* nowa wartosc licznika           *)
        end;

const
                                (* tryby pracy funkcji                     *)
LC0_SET_CTC_MODE        = 1;    (* zaprogramuj tryb pracy kanalu           *)
LC0_SET_COUNTER_VALUE   = 2;    (* zaladuj nowa wartosc licznika           *)
LC0_CTC_ENABLE          = 4;    (* zezwolenie CTC                          *)

(* Rekord opisu zlecenia funkcji  CTC_READ ================================*)
type
lc0_ctc_read =
        record
            LC0_CODE:           byte;   (* kod funkcji (12)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_UMODULE:        byte;   (* numer modulu                    *)
            LC0_UNUM:           byte;   (* numer kanalu                    *)
            LC0_UVAL:           word;   (* odczytana wartosc licznika      *)
        end;

(* Rekord opisu zlecenia funkcji  DATA_TRANSMIT ===========================*)
type
lc0_transmit =
        record
            LC0_CODE:           byte;   (* kod funkcji (13)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_RMODE:          byte;   (* tryb przesylania                *)
            LC0_RADDR:          pointer;(* adres bufora w pam. podstawowej *)
            LC0_RLEN:           longint;(* dlugosc bufora w pam. podst.    *)
            LC0_RMEAS:          longint;(* numer pierwszej probki  *)
            LC0_RNUM:           longint;(* liczba probek do przeslania     *)
            LC0_RMEMA:          longint;(* adres bufora w pamieci          *)
                                        (*   rozszerzonej (absolutny)      *)
            LC0_RRNUM:          longint;(* rzeczywista liczba probek       *)

        end;

const
                                (* tryby pracy funkcji                     *)
LC0_TO_EXT_DIR          = 1;    (* do pamieci rozszerzonej                 *)
LC0_FROM_EXT_DIR        = 0;    (* z pamieci modulu / rozszerzonej         *)

(* Rekord opisu zlecenia funkcji ANALOG_INPUT =============================*)
type
lc0_analog_in =
        record
            LC0_CODE:           byte;   (* kod funkcji (14)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                                (* bledach                 *)
            LC0_AMODULE:        byte;   (* numer modulu                    *)
            LC0_ANUM:           byte;   (* numer przetwornika              *)
            LC0_AMODE:          word;   (* tryb pracy funkcji              *)
            LC0_ASTST:          byte;   (* typy warunkow startu i stopu    *)
            LC0_APER:           longint;(* okres probkowania               *)
            LC0_APER2:          word;   (* krotnosc okresu probkowania     *)
            LC0_ACHAN:          byte;   (* liczba kanalow podstawowych     *)
                                        (*   (b8 okresla czy praca         *)
                                        (*   wielokanalowa (0) czy         *)
                                        (*   jednokanalowa (1))            *)
            LC0_ACHAN2:         byte;   (* liczba kanalow dodatkowych      *)
            LC0_AADDR:          pointer;(* adres bufora w pam. podstawowej *)
            LC0_ALEN:           longint;(* dlugosc bufora                  *)
            LC0_AMEMA:          longint;(* adres bufora w pamieci          *)
                                        (*   rozszerzonej (absolutny)      *)
            LC0_ABMAR:          word;   (* dlugosc marginesu poczatkowego  *)
            LC0_AEMAR:          word;   (* dlugosc marginesu koncowego     *)
```

```
            LC0_AHAND:          word;   (* numer handler'a zbioru          *)
            LC0_ASTART:         lc0_start;      (* parametry warunku startu*)
            LC0_ASTOP:          lc0_stop;       (* parametry warunku stopu *)
            LC0_ARDIV1:         word;   (* pierwszy podzielnik zegara      *)
            LC0_ARDIV2:         word;   (* drugi podzielnik zegara         *)
            LC0_ARMNUM:         longint;(* rzeczywista liczba probek       *)
            LC0_ARBMAR:         word;   (* rzeczywista dlugosc marg. pocz. *)
            LC0_AREMAR:         word;   (* rzeczywista dlugosc marg. konc. *)
            LC0_ARLEN:          longint;(* rzeczywista liczba przepisanych *)
                                        (*   probek                        *)
            LC0_ARBUF:          word;   (* dla pracy z buforem cyklicznym: *)
                                        (*   numer probki okreslajacej     *)
                                        (*   poczatek bufora po zakonczeniu*)
                                        (*   pomiaru                       *)
        end;

(* Rekord opisu zlecenia funkcji  ANALOG_OUTPUT ============================*)
type
lc0_memory =
        record
            case integer of
                1: (base_memory:        pointer);
                2: (extended_memory:    longint)
        end;

type
lc0_analog_out =
        record
            LC0_CODE:           byte;   (* kod funkcji (15)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                        (* bledach                         *)
            LC0_NMODULE:        byte;   (* numer modulu                    *)
            LC0_NNUM:           byte;   (* numer przetwornika              *)
            LC0_NMODE:          word;   (* tryb pracy funkcji              *)
            LC0_NSTST:          byte;   (* typy warunkow startu i stopu    *)
            LC0_NCHAN:          byte;   (* liczba kanalow (b8 okresla      *)
                                        (*   czy praca wielokanalowa (0)   *)
                                        (*   czy jednokanalowa (1))        *)
            LC0_NPER:           longint;(* okres probkowania               *)
            LC0_NADDR:          lc0_memory;     (* adres bufora            *)
                                        (*   (segment:offset dla           *)
                                        (*   pamieci podstawowej, absolutny*)
                                        (*   dla pamieci rozszerzonej)     *)
            LC0_NLEN:           longint;(* dlugosc bufora w pam. podst.    *)
            LC0_NHAND:          word;   (* numer handler'a zbioru          *)
            LC0_NSTART:         lc0_start;      (* parametry warunku startu*)
            LC0_NSTOP:          lc0_stop;       (* parametry warunku stopu *)
            LC0_NRMNUM:         longint;(* rzeczywista liczba wyslanych    *)
                                        (*   probek                        *)
        end;

(* Rekord opisu zlecenia funkcji  LEAVE_DRIVER =============================*)
type
lc0_leave =
        record
            LC0_CODE:           byte;   (* kod funkcji (16)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                        (* bledach                         *)
        end;

(* Rekord opisu zlecenia funkcji  INTERRUPT_SERVICE ========================*)
type
lc0_interrupt =
        record
            LC0_CODE:           byte;   (* kod funkcji (17)                *)
            LC0_STATUS:         shortint;       (* kod odpowiedzi driver'a *)
            LC0_ERR_STAT:       shortint;       (* dodatkowe informacje o  *)
                                        (* bledach                         *)
            LC0_SMODULE:        byte;   (* numer modulu                    *)
            LC0_SPROC:          pointer;(* adres procedury obslugi         *)
            LC0_SSTAT:          pointer;(* adres slowa komunikacyjnego     *)
        end;

const
```

```
LC0_IS_START    = 1;            (* przerwanie wystapilo z powodu          *)
                                (*   rozpoczecia pomiaru                  *)
LC0_IS_ONE      = 2;            (* przerwanie nastapilo z powodu zmierzenia*)
                                (*   kolejnej serii pomiarowej            *)
LC0_IS_END_ADC  = 4;            (* przerwanie nastapilo z powodu          *)
                                (*   zakonczenia przetwarzania a/c        *)
LC0_IS_END_DAC  = 8;            (* przerwanie nastapilo z powodu          *)
                                (*   zakonczenia przetwarzania c/a        *)
LC0_IS_BROKEN   = 16;           (* zakonczono przetwarzanie z powodu      *)
                                (* wykonania funkcji BREAK                *)
LC0_IS_SAMPLE   = 32;           (* przerwanie wystapilo z powodu zmierzenia*)
                                (*   kolejnej probki                      *)


(********************** pozostale stale ***********************************)
const
(******************* kody funkcji driver'a ********************************)
MODULE_INIT                    =  0;
GET_TOTAL_CONFIGURATION        =  1;
GET_MODULE_CONFIGURATION       =  2;
GET_INFO                       =  3;
SET_CLOCK                      =  4;
SET_VOLTAGE_RANGE              =  5;
SET_TIME                       =  6;
WAIT_FOR_END                   =  7;
BREAK                          =  8;
DIGITAL_INPUT                  =  9;
DIGITAL_OUTPUT                 = 10;
CTC_WRITE                      = 11;
CTC_READ                       = 12;
DATA_TRANSMIT                  = 13;
ANALOG_INPUT                   = 14;
ANALOG_OUTPUT                  = 15;
LEAVE_DRIVER                   = 16;
INTERRUPT_SERVICE              = 17;

(*************** numery przerwan obslugiwanych przez driver ***************)
LC010_16        = $99;          (* LC-010-1612                            *)
LC011_08        = $90;          (* LC-011-0812                            *)
LC011_16        = $91;          (* LC-011-1612                            *)
LC015_16        = $92;          (* LC-015-1612                            *)
LC020_08_0      = $93;          (* LC-020-0812 v.0                        *)
LC020_08_2      = $94;          (* LC-020-0812 v.1 i v.2                  *)
LC020_32        = $95;          (* LC-020-3212                            *)
LC030_16        = $96;          (* LC-030-1612                            *)
LC060_06        = $97;          (* LC-060-0612                            *)

(******************** kody odpowiedzi funkcji driver'a ********************)
LC0_OK  = 0;

(* ======= ostrzezenia ======== *)
LC0_NON_EX_MOD          = 1;    (* nie istniejacy(e) modul(y)             *)
LC0_OTHER_LEN           = 2;    (* przepisano mniejsza liczbe pomiarow    *)
LC0_PREMATURE_END       = 3;    (* przedwczesne zakonczenie z powodu      *)
                                (*   przepelnienia bufora                 *)
LC0_IN_PROGRESS         = 4;    (* badana transmisja jeszcze trwa         *)

(* ========== bledy =========== *)
LC0_UNKN_FUNC           =  -1;  (* nieznany kod funkcji                   *)
LC0_NO_MODULE           =  -2;  (* brak modulu(ow)                        *)
LC0_BAD_DEV_TYP         =  -3;  (* bledny typ urzadzenie                  *)
LC0_NONEX_DEV           =  -4;  (* nie istnieje urzadzenie o tym numerze  *)
LC0_BAD_FREQ            =  -5;  (* zla czestotliwosc zegara               *)
LC0_BAD_RANGE           =  -6;  (* zly zakres napiec                      *)
LC0_NO_OPER             =  -7;  (* zadna operacja nie jest wykonywana     *)
LC0_BAD_MARGIN          =  -8;  (* bledna dlugosc marginesu poczatkowego  *)
LC0_BAD_BUF_ADR         =  -9;  (* bledny adres bufora                    *)
LC0_BAD_BUF_LEN         = -10;  (* bledna dlugosc bufora                  *)
LC0_DEV_BUSY            = -11;  (* urzadzenie jest zajete                 *)
LC0_BAD_PER             = -12;  (* zly okres probkowania                  *)
LC0_BAD_CHAN_N          = -13;  (* zla liczba kanalow                     *)
LC0_BAD_CHAN            = -14;  (* numer nie istniejacego kanalu          *)
LC0_BROKEN              = -15;  (* przerwano funkcja BREAK                *)
LC0_INTR_NOT_INST       = -16;  (* procedura obslugi przerwania nie jest  *)
                                (*   zainstalowana                        *)
LC0_ILL_START_CODE      = -17;  (* nielegalny typ warunku startu          *)
```

```
LC0_ILL_STOP_CODE       = -18;  (* nielegalny typ warunku stopu           *)
LC0_BAD_PROC            = -19;  (* bledny adres procedury obslugi         *)
                                (*   przerwania lub slowa komunikacyjnego  *)
LC0_TOO_LONG_MARG       = -20;  (* margines dluzszy od bufora             *)
LC0_ILL_START           = -21;  (* bledne parametry warunku startu        *)
LC0_ILL_STOP            = -22;  (* bledne parametry warunku stopu         *)
LC0_BAD_MNUM            = -23;  (* bledny numer pierwszej probki          *)
LC0_NOT_SUPPORTED       = -24;  (* dla danego modulu funkcja nie jest     *)
                                (*   realizowana                           *)
LC0_BAD_CTC_MODE        = -25;  (* bledny tryb pracy CTC                  *)
LC0_NO_PARAMS           = -26;  (* nie podano parametrow przetwarzania    *)
LC0_OVERRUN             = -27;  (* zakonczono przetwarzanie z powodu bledu *)
                                (*   OVERRUN                               *)
LC0_NO_DMA              = -28;  (* z danym urzadzeniem nie jest zwiazany  *)
                                (*   zaden kanal DMA                       *)
LC0_NO_IRQ              = -29;  (* z danym modulem nie jest zwiazane zadne *)
                                (*   przerwanie                            *)
LC0_NOT_FULLY_SUP       = -30;  (* zadany tryb wykonania funkcji nie jest  *)
                                (*   realizowany dla danego typu modulu lub*)
                                (*   funkcja w opracowaniu                 *)
LC0_NO_EXTMEM           = -31;  (* brak pamieci roszerzonej               *)
LC0_NO_SEC_FREQ         = -32;  (* modul nie moze wykonywac pomiarow z    *)
                                (*   podwojna czestotliwoscia              *)
LC0_INTR_INST           = -33;  (* procedura obslugi przerwania juz       *)
                                (*   zainstalowana                         *)
LC0_BAD_PER2            = -34;  (* bledna wielokrotnosc okresu probkowania *)
                                (*   (0 lub 1)                             *)
LC0_BAD_MODE            = -35;  (* bledny tryb pracy                      *)
LC0_BAD_EXTMEM          = -36;  (* zly adres bufora w pamieci rozszerzonej *)
LC0_CTC_NOT_PROGRAMMED  = -37;  (* zapis licznika przy niezaprogramowanym  *)
                                (*   trybie pracy                          *)
LC0_REJECTED            = -38;  (* za wiele rownoleglych wejsc do driver'a *)

(************************* dodatkowe informacje o bledach *****************)
LC0_E_OK                = 0;    (* brak dodatkowych informacji            *)
LC0_E_NO_MODULE         = -1;   (* nie ma takiego modulu                  *)
LC0_E_NONEX_DEV         = -2;   (* nie istnieje urzadzenie o tym numerze  *)
LC0_E_BAD_CHAN          = -3;   (* numer nieistniejacego kanalu           *)
LC0_E_BAD_TIME          = -4;   (* zly odcinek czasu                      *)
LC0_E_BAD_DATE          = -5;   (* zla specyfikacja daty                  *)
LC0_E_BAD_THRE          = -6;   (* bledny prog wyzwalania analogowego     *)
LC0_E_BROKEN_WAIT       = -7;   (* funkcja przerwana w trakcie oczekiwania *)
                                (*   na spelnienie warunku startu         *)
LC0_E_BROKEN_RUN        = -8;   (* funkcja przerwana w trakcie            *)
                                (*   przetwarzania                        *)
LC0_E_BAD_LEN           = -9;   (* zadeklarowano za duzo probek do        *)
                                (*   zmierzenia                           *)

(************************* kody warunkow startu ***************************)
LC0_SIMMED      = 0;            (* natychmiastowy                         *)
LC0_SHARD       = 1;            (* od sygnalu sprzetowego                 *)
LC0_SLEVEL      = 2;            (* od poziomu sygnalu cyfrowego           *)
LC0_SSLOPE      = 3;            (* od zbocza sygnalu cyfrowego            *)
LC0_SDIG_EQ     = 4;            (* od kombinacji bitow - rowne            *)
LC0_SDIG_NE     = 5;            (* od kombinacji bitow - rozne            *)
LC0_STIME       = 6;            (* po uplynieciu okreslonego czasu        *)
LC0_SDATE       = 7;            (* o podanym czasie                       *)
LC0_SANALOG     = 8;            (* od sygnalu analogowego                 *)

(************************* kody warunkow stopu ****************************)
LC0_ZSAMPLES    = $00;          (* po zmierzeniu okreslonej liczby probek *)
LC0_ZBREAK      = $10;          (* po wykonaniu funkcji BREAK             *)
LC0_ZLEVEL      = $20;          (* od poziomu sygnalu cyfrowego           *)
LC0_ZSLOPE      = $30;          (* od zbocza sygnalu cyfrowego            *)
LC0_ZDIG_EQ     = $40;          (* od kombinacji bitow - rowne            *)
LC0_ZDIG_NE     = $50;          (* od kombinacji bitow - rozne            *)
LC0_ZTIME       = $60;          (* po uplynieciu okreslonego czasu        *)
LC0_ZDATE       = $70;          (* o podanym czasie                       *)
LC0_ZANALOG     = $80;          (* od sygnalu analogowego                 *)

(************************* kody numerow modulow ***************************)
LC0_MODA        = 1;            (* modul A                                *)
LC0_MODB        = 2;            (* modul B                                *)
LC0_MODC        = 3;            (* modul C                                *)
LC0_MODD        = 4;            (* modul D                                *)
```

```
(************************* maski modulow w mapie ***************************)
LC0_MODAMAP     = 1;            (* modul A                                  *)
LC0_MODBMAP     = 2;            (* modul B                                  *)
LC0_MODCMAP     = 4;            (* modul C                                  *)
LC0_MODDMAP     = 8;            (* modul D                                  *)

(************************ kody typow urzadzen ****************************)
LC0_DINPUT      = 1;            (* wejsciowy port cyfrowy                   *)
LC0_DOUTPUT     = 2;            (* wyjsciowy port cyfrowy                   *)
LC0_AINPUT      = 3;            (* przetwornik a/c                          *)
LC0_AOUTPUT     = 4;            (* przetwornik c/a                          *)
LC0_CTC         = 5;            (* kanal CTC                                *)

(*********** maski trybu pracy funkcji ANALOG_INPUT i ANALOG_OUTPUT ********)
LC0_MOD_START           =    1; (* start przetwarzania                      *)
LC0_MOD_NEW_PAR         =    2; (* nowe parametry                           *)
LC0_MOD_SYNCHR          =    4; (* praca synchroniczna                      *)
LC0_MOD_ASYNCHR         =    0; (* praca asynchroniczna                     *)
LC0_MOD_INTR            =    8; (* praca bez przerwan                       *)
LC0_MOD_INTR_TYPE       =   16; (* bit trybu przerwan                       *)
LC0_MOD_END_INTR        =    0; (* przerwanie po koncu przetwarzania        *)
LC0_MOD_ONE_INTR        =   16; (* przerwanie po kazdej probce              *)
LC0_MOD_BLOCK           =   32; (* praca blokowa                            *)
LC0_MOD_SINGLE          =    0; (* praca pojedyncza                         *)
LC0_MOD_CYCL            =   64; (* bufor cykliczny                          *)
LC0_MOD_FILE_W          =  128; (* zapis do pliku                           *)
LC0_MOD_FILE_R          =  128; (* odczyt z pliku                           *)
LC0_MOD_MEM_W           =  256; (* przepisanie do pamieci                   *)
LC0_MOD_EXT_CLK         =  512; (* zegar zewnetrzny                         *)
LC0_MOD_EXT_MEM         = 1024; (* pamiec rozszerzona                       *)

(************* wartosci fragmentow bajtu kontrolnego 8253/8254 *************)
CTC_NB          = 0;            (* kod naturalny binarny                    *)
CTC_BCD         = 1;            (* kod BCD                                  *)

CTC_MODE0       = 0;            (* tryb 0                                   *)
CTC_MODE1       = 2;            (* tryb 1                                   *)
CTC_MODE2       = 4;            (* tryb 2                                   *)
CTC_MODE3       = 6;            (* tryb 3                                   *)
CTC_MODE4       = 8;            (* tryb 4                                   *)
CTC_MODE5       = 10;           (* tryb 5                                   *)

CTC_LSB         = $10;          (* tylko mlodszy bajt                       *)
CTC_MSB         = $20;          (* tylko starszy bajt                       *)
CTC_BOTH        = $30;          (* mlodszy - starszy                        *)

CTC_COUNT0      = $00;          (* licznik 0                                *)
CTC_COUNT1      = $40;          (* licznik 1                                *)
CTC_COUNT2      = $80;          (* licznik 2                                *)
```

D O D A T E K D

d o d o k u m e n t a c j i d r i v e r ' a

m o d u ł u k o n t r o l n o - p o m i a r o w e g o

L C - 0 1 1 - 1 6 1 2

TEST.PAS
program przykładowy w Pascalu

```
(* TEST.PAS ************************************************************)
(* Program przykladowy pokazujacy sposob wykorzystania driver'ow modulow   *)
(* serii LC-011 i LC-020 firmy AMBEX.                                      *)
(***************************************************************************)
(***************************************************************************)
(* UWAGA: Przed rozpoczeciem kompilacji nalezy sprawdzic ustawienie opcji  *)
(*      kompilatora:                                                       *)
(*           - alingment = byte,                                           *)
(***************************************************************************)

program test;

uses crt, dos;

{$i ambex-lc.pas}

type
    buftype = array[1..320] of integer;         (* bufor na probki         *)

var
    s_init:      lc0_init;
    s_total:     lc0_total;
    s_module:    lc0_module;
    s_info:      lc0_info;
    s_break:     lc0_break;
    s_analog_in: lc0_analog_in;
    s_leave:     lc0_leave;

    r: registers;                       (* parametr dla funkcji intr       *)
    LCinterrupt: integer;               (* numer przerwania driver'a       *)
    modulenum: integer;                 (* numer badanego modulu           *)

const
    BUFLEN      = 320;                          (* dlugosc bufora          *)
    SAMPLES     = 20;                           (* liczba probek           *)
    CHANNELS    = 8;                            (* liczba kanalow          *)

(***************************************************************************)
(* Przeznaczenie:                                                          *)
(*   Zamiana cyfry szesnastkowej na jej reprezentacje znakowa.             *)
(* Parametry:                                                              *)
(*   dig - cyfra do zamiany                                                *)
(* Wartosc:                                                                *)
(*   Reprezentacja znakowa cyfry.                                          *)
(***************************************************************************)
function hexdigit(dig: word) : char;
begin
    if dig < 10 then hexdigit := chr(ord('0') + dig)
    else             hexdigit := chr(ord('A') + dig - 10);
end;

(***************************************************************************)
(* Przeznaczenie:                                                          *)
(*   Wydrukowanie liczby calkowitej w postaci szesnastkowej.               *)
(* Parametry:                                                              *)
(*   val - wartosc do wydrukowania                                         *)
(***************************************************************************)
procedure writehex(val: word);
begin
    write(hexdigit((val and $F000) shr 12));
    write(hexdigit((val and $F00) shr 8));
    write(hexdigit((val and $F0) shr 4));
    write(hexdigit(val and $F));
end;

(***************************************************************************)
(* Przeznaczenie:                                                          *)
(*   Inicjalizacja zmiennych.                                              *)
(***************************************************************************)
procedure initprogram;
begin
    s_init.LC0_CODE             := MODULE_INIT;
    s_total.LC0_CODE            := GET_TOTAL_CONFIGURATION;
    s_module.LC0_CODE           := GET_MODULE_CONFIGURATION;
    s_info.LC0_CODE             := GET_INFO;
    s_break.LC0_CODE            := BREAK;
```

```
    s_analog_in.LC0_CODE        := ANALOG_INPUT;
    s_leave.LC0_CODE            := LEAVE_DRIVER;
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Sprawdzenie obecnosci drivera.                                      *)
(* Sposob:                                                               *)
(*   Przez probe otwarcia urzadzenia o nazwie okreslonej przez driver.   *)
(* Parametry:                                                            *)
(*   name - nazwa urzadzenia                                             *)
(* Wartosc:                                                              *)
(*   TRUE  - driver jest zainstalowany                                   *)
(*   FALSE - driver nie jest zainstalowany                               *)
(*************************************************************************)
function driverinstalled(name: string) : boolean;
var
    hd: text;
begin
{$I-}
    assign(hd, name);
    reset(hd);
    if IOResult <> 0 then
        driverinstalled := false
    else
        begin
            close(hd);
            driverinstalled := true
        end;
{$I+}
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Rozpoznanie, ktory driver (a co za tym idzie ktory modul) jest      *)
(*   zainstalowany.                                                      *)
(* Sposob:                                                               *)
(*   Przez sprawdzenie zainstalowania kazdego z potencjalnych driver'ow. *)
(* Uwagi:                                                                *)
(*   Funkcja dodatkowo wyswietla komunikat o testowanym module i podstawia *)
(*   wlasciwa wartosc na zmienna LCinterrupt (wlasciwy numer przerwania  *)
(*   driver'a). Jezeli zaden driver nie jest zainstalowany to po         *)
(*   wyswietleniu wlasciwego komunikaru program konczy prace.            *)
(*************************************************************************)
procedure checkdrivers;
begin
    clrscr;
    gotoxy(1, 1);
    if driverinstalled('LC1116^^') then
        begin
            writeln('Test modulu LC-011-1612');
            LCinterrupt := LC011_16;
        end
    else if driverinstalled('LC2008^2') then
        begin
            writeln('Test modulu LC-020-0812');
            LCinterrupt := LC020_08_2;
        end
    else if driverinstalled('LC2032^^') then
        begin
            writeln('Test modulu LC-020_3212');
            LCinterrupt := LC020_32;
        end
    else
        begin
            writeln('Driver nie jest zainstalowany!');
            halt;
        end;
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Rozpoznanie konfiguracji badanego modulu.                           *)
(* Sposob:                                                               *)
(*   Przez wykorzystanie funkcji driver'a GET_TOTAL_CONFIGURATION,       *)
(*   GET_MODULE_CONFIGURATION, GET_INFO.                                 *)
```

```
(* Uwagi:                                                                  *)
(*   Funkcja nadaje wartosc zmiennej modulenum (numer badanego modulu)     *)
(*   wypelnia struktury total, module, info i inicjalizuje zainstalowane   *)
(*    moduly.                                                              *)
(*************************************************************************)
procedure askdriver;
label   lab1;
begin
    r.dx := seg(s_total);
    r.di := ofs(s_total);
    intr(LCinterrupt, r);                       (* GET_TOTAL_CONFIGURATION *)
                              (* inicjalizacja zainstalowanych modulow   *)
    s_init.LC0_IMODULE := s_total.LC0_TONF and $F;
    r.dx := seg(s_init);
    r.di := ofs(s_init);
    intr(LCinterrupt, r);                                (* MODULE_INIT     *)

                                       (* sprawdzenie, ktory modul jest   *)
                                       (* zainstalowany: A, B, C czy D    *)
    for modulenum := 1 to 4 do
        if ((s_total.LC0_TONF and (1 shl (modulenum - 1))) <> 0) then
            goto lab1;

lab1:
                                       (* spytanie o konfiguracje modulu  *)
    s_module.LC0_MMODULE := modulenum;
    r.dx := seg(s_module);
    r.di := ofs(s_module);
    intr(LCinterrupt, r);                       (* GET_MODULE_CONFIGURATION*)

                                       (* spytanie o konfiguracje toru a/c*)
    s_info.LC0_GMODULE := modulenum;
    s_info.LC0_GTYPE   := LC0_AINPUT;
    s_info.LC0_GNUM    := 1;
    r.dx := seg(s_info);
    r.di := ofs(s_info);
    intr(LCinterrupt, r);                                (* GET_INFO        *)
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wyswietlenie konfiguracji badanego modulu i jego toru a/c.          *)
(* Sposob:                                                               *)
(*   Przez wykorzystanie informacji zawartych w strukturach module i info. *)
(*************************************************************************)
procedure displayconfiguration;
var
    i: integer;
begin
                                       (* konfiguracja modulu              *)
    writeln;
    writeln('---------- Konfiguracja modulu:');
    writeln;
    write('Adres modulu: ');
    writehex(s_module.LC0_MBASE1);
    writeln(' (hex)');
    writeln('Liczba przetwornikow a/c: ', s_module.LC0_MIAD);
    writeln('Liczba przetwornikow c/a: ', s_module.LC0_MIDA);
    writeln('Liczba portow cyfrowych wejsciowych: ', s_module.LC0_MIDI);
    writeln('Liczba portow cyfrowych wyjsciowych: ', s_module.LC0_MIDO);
    writeln('Czestotliwosc zegara modulu:  ',
            (s_module.LC0_MCLOCK / 1000):4:2, 'MHz');

                                       (* konfiguracja toru a/c            *)
    writeln;
    writeln('---------- Konfiguracja toru a/c:');
    writeln;
    writeln('Liczba kanalow: ', s_info.LC0_GCHAN);
    writeln('Rozdzielczosc: ', s_info.LC0_GRES, ' bitow');
    writeln('Zakres napiec: ', (s_info.LC0_GMINV / 10):5:1, '..',
            (s_info.LC0_GMAXV / 10):4:1, ' V');
    if s_info.LC0_GDMA = $FF then
        writeln('Tor a/c nie jest podlaczony do kanalow DMA')
    else
        writeln('Numer kanalu DMA podlaczonego do toru a/c: ',
                                                    s_info.LC0_GDMA);
```

```
    writeln('Minimalne okresy probkowania [æs]:');
    write('     ');
    for i := 1 to s_info.LC0_GCHAN do
        begin
            write((s_info.LC0_GMINP[i] / 10):5:1);
            if i < s_info.LC0_GCHAN then
                begin
                    write(', ');
                    if (i mod 8) = 0 then
                        begin
                            writeln;    (* zlamanie linii co 8 wielkosci   *)
                            write('     ');
                        end
                end
        end;
    writeln;
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Zainstalowanie procedury obslugi przerwania generowanego przez      *)
(*   Ctrl-Break.                                                         *)
(* Sposob:                                                               *)
(*   Przez wywolanie funkcji BREAK.                                      *)
(*************************************************************************)
procedure installbreak;
begin
    s_break.LC0_BMODE := LC0_BREAK_INST;
    s_break.LC0_BPROC := nil;   (* podlozona zostanie procedura driver'a   *)
    r.dx := seg(s_break);
    r.di := ofs(s_break);
    intr(LCinterrupt, r);                              (* BREAK           *)
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Przerwa miedzy kolejnymi czesciami programu - oczekiwanie na reakcje  *)
(*   operatora.                                                          *)
(*************************************************************************)
procedure pressanykey;
var
    c: char;
begin
    writeln;
    writeln('Nacisnij dowolny klawisz . . .');
    c := readkey;
    clrscr;
    gotoxy(1, 1);
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wyswietlenie bufora z danymi pomiarowymi.                           *)
(* Parametry:                                                            *)
(*   buf - adres bufora                                                  *)
(*   c   - liczba kanalow                                                *)
(*   s   - liczba probek na kanal                                        *)
(*************************************************************************)
procedure displaybuf(buf: buftype; c, s: integer);
var
    i: integer;
    j: integer;
begin
    for i := 1 to s do
        begin
            write(i:2, ': ');
            for j := 1 to c do
                begin
                    writehex(buf[(i - 1) * c + j]);
                    write('  ');
                end;
            writeln;
        end;
end;

(*************************************************************************)
```

```
(* Przeznaczenie:                                                          *)
(*   Wypisanie komunikatow o bledzie / ostrzezeniu / dodatkowej informacji *)
(*   bledzie.                                                              *)
(* Parametry:                                                              *)
(*   status   - LC0_STATUS                                                 *)
(*   err_stat - LC0_ERR_STAT                                               *)
(*************************************************************************)
procedure drivererror(status, err_stat: shortint);
begin
    if status > 0 then
        writeln('--------- Ostrzezenie: ', status)
    else
        writeln('--------- Blad: ', status);
    if err_stat < 0 then
        writeln('--------- Informacje dodatkowe: ', err_stat);
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wykonanie transmisji blokowej.                                      *)
(* Sposob:                                                               *)
(*   Przez wykonanie funkcji ANALOG_INPUT.                               *)
(* Parametry:                                                            *)
(*   start - typ warunku startu                                          *)
(* Wartosc:                                                              *)
(* Uwagi:                                                                *)
(*   Funkcja wywolujaca musi ustawic okres probkowania i parametry warunku *)
(*   startu.                                                             *)
(*************************************************************************)
procedure transmission(start: byte);
var
    buf: buftype;                       (* bufor na probki              *)
    c: char;
begin
    writeln('Pomiar blokowy, ', CHANNELS, ' kanalow, ', SAMPLES,
            ' probek, okres probkowania ',
            (s_analog_in.LC0_APER / 10):11:1, ' æs:');
    case start of
        LC0_SIMMED:
            writeln('Warunek startu: natychmiast');
        LC0_STIME:
            writeln('Warunek startu: uplyw ', s_analog_in.LC0_ASTART.time,
                    ' sekund');
                                (* pozostale warunki nie sa obslugiwane bo *)
                                (* w programie nie sa wykorzystywane       *)
        end;
    s_analog_in.LC0_AMODULE := modulenum;
    s_analog_in.LC0_ANUM := 1;
    s_analog_in.LC0_AMODE := LC0_MOD_START or
                             LC0_MOD_NEW_PAR or
                             LC0_MOD_SYNCHR or
                             LC0_MOD_BLOCK;
                                (* stop po zmierzeniu okreslonej liczby    *)
                                (* probek                                  *)
    s_analog_in.LC0_ASTST := start + LC0_ZSAMPLES;
                                (* praca wielokanalowa, CHANNELS kanalow   *)
    s_analog_in.LC0_ACHAN := CHANNELS;
    s_analog_in.LC0_AADDR := @buf;
    s_analog_in.LC0_ALEN := BUFLEN;     (* dlugosc bufora                  *)
    s_analog_in.LC0_ABMAR := 0;
    s_analog_in.LC0_AEMAR := 0;         (* oba marginesy zerowe            *)
                                        (* calkowita liczba probek         *)
    s_analog_in.LC0_ASTOP.samples := SAMPLES * CHANNELS;
    r.dx := seg(s_analog_in);
    r.di := ofs(s_analog_in);
    intr(LCinterrupt, r);                               (* ANALOG_INPUT    *)
    if s_analog_in.LC0_STATUS <> LC0_OK then
        begin
            if s_analog_in.LC0_STATUS = LC0_BROKEN then
                c := readkey;   (* oproznienie buf. klawiatury po Ctrl-Break*)
            drivererror(s_analog_in.LC0_STATUS, s_analog_in.LC0_ERR_STAT)
        end
    else
        displaybuf(buf, CHANNELS, SAMPLES);
    pressanykey;
end;
```

```
(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wykonanie poprawnej transmisji blokowej i wyswietlenie zmierzonych  *)
(*   wartosci.                                                           *)
(* Sposob:                                                               *)
(*   Przez wykonanie funkcji transmission.                               *)
(*************************************************************************)
procedure blocktransmission;
begin
    writeln('---------- Poprawne przetwarzanie blokowe');
    writeln;
    s_analog_in.LC0_APER := s_info.LC0_GMINP[CHANNELS];
    transmission(LC0_SIMMED);
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wykonanie blednej transmisji blokowej.                              *)
(* Sposob:                                                               *)
(*   Przez wykonanie funkcji ANALOG_INPUT z okresem probkowania mniejszym*)
(*   niz minimalny wskazany przez driver.                                *)
(*************************************************************************)
procedure failblocktransmission;
begin
    writeln('---------- Bledne przetwarzanie blokowe');
    writeln;
    s_analog_in.LC0_APER := s_info.LC0_GMINP[CHANNELS] - 1;
    transmission(LC0_SIMMED);
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wykonanie poprawnej transmisji blokowej ale przerwanej przez operatora*)
(*   (Ctrl-Break) w trakcie czekania na spelnienie warunku startu.       *)
(* Sposob:                                                               *)
(*   Przez wykonanie funkcji ANALOG_INPUT z warunkiem startu LC0_STIME   *)
(*   (start po okreslonym czasie) i parametrem tego warunku - 1000s.     *)
(*************************************************************************)
procedure interruptedbefore;
begin
    writeln('---------- Przetwarzanie blokowe z oczekiwaniem 1000s');
    writeln('---------- Oczekiwanie nalezy przerwac Ctrl-Break');
    writeln;
    s_analog_in.LC0_APER := s_info.LC0_GMINP[CHANNELS];
    s_analog_in.LC0_ASTART.time := 1000;
    transmission(LC0_STIME);
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wykonanie poprawnej transmisji blokowej ale przerwanej przez operatora*)
(*   (Ctrl-Break) w trakcie pomiaru.                                     *)
(* Sposob:                                                               *)
(*   Przez wykonanie funkcji ANALOG_INPUT z okresem probkowania 100s.    *)
(*************************************************************************)
procedure interruptedafter;
begin
    writeln('---------- Przetwarzanie blokowe z okresem probkowania 100s');
    writeln('---------- Pomiar nalezy przerwac Ctrl-Break');
    writeln;
    s_analog_in.LC0_APER := 1000000000;
    transmission(LC0_SIMMED);
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Wykonanie pomiaru bloku probek za pomoca tranmsisji pojedynczej, przy *)
(*   czym caly blok ma byc zmierzony po 5s od startu.                    *)
(* Sposob:                                                               *)
(*   Przez wykonanie funkcji ANALOG_INPUT.                               *)
(* Parametry:                                                            *)
(* Wartosc:                                                              *)
(* Uwagi:                                                                *)
(*************************************************************************)
procedure singletransmission;
```

```
var
    buf: buftype;                       (* bufor na probki                  *)
    i: integer;
    j: integer;
begin
    writeln('---------- Przetwarzanie pojedyncze - po 5 sekundach');
    writeln;
    s_analog_in.LC0_AMODULE := modulenum;
    s_analog_in.LC0_ANUM := 1;
    s_analog_in.LC0_AMODE := LC0_MOD_START or
                             LC0_MOD_NEW_PAR or
                             LC0_MOD_SINGLE;
                                (* praca wielokanalowa, CHANNELS kanalow   *)
    s_analog_in.LC0_ACHAN := CHANNELS;
    s_analog_in.LC0_AADDR := @buf;
    s_analog_in.LC0_ALEN := CHANNELS;   (* dlugosc bufora (tylko na jeden  *)
                                        (* pomiar)                         *)
    s_analog_in.LC0_ASTST := LC0_STIME;
    s_analog_in.LC0_ASTART.time := 5;   (* start po 5 sekundach            *)
    r.dx := seg(s_analog_in);
    r.di := ofs(s_analog_in);
    intr(LCinterrupt, r);                               (* ANALOG_INPUT    *)
    s_analog_in.LC0_ASTST := LC0_SIMMED;
    s_analog_in.LC0_AADDR := @buf[CHANNELS + 1];
    i := 3;
    r.dx := seg(s_analog_in);
    r.di := ofs(s_analog_in);
    intr(LCinterrupt, r);                               (* ANALOG_INPUT    *)
                                    (* zgaszenie bitu LC0_MOD_NEW_PAR  *)
    s_analog_in.LC0_AMODE := s_analog_in.LC0_AMODE and (not LC0_MOD_NEW_PAR);
    for j := 1 to 18 do
        begin
            s_analog_in.LC0_AADDR := @buf[(i - 1) * CHANNELS + 1];
            i := i + 1;
            r.dx := seg(s_analog_in);
            r.di := ofs(s_analog_in);
            intr(LCinterrupt, r);                       (* ANALOG_INPUT    *)
        end;
    displaybuf(buf, CHANNELS, SAMPLES);
    pressanykey;
end;

(*************************************************************************)
(* Przeznaczenie:                                                        *)
(*   Zakonczenie pracy programu.                                         *)
(* Sposob:                                                               *)
(*   Rozstanie sie z driver'em za pomoca funkcji LEAVE_DRIVER.           *)
(*************************************************************************)
procedure quit;
begin
    r.dx := seg(s_leave);
    r.di := ofs(s_leave);
    intr(LCinterrupt, r);                               (* LEAVE_DRIVER    *)

    writeln('Dziekuje, to wszystko!');
end;

begin
        initprogram;            (* inicjalizacja programu                  *)
        checkdrivers;           (* rozpoznanie zainstalowanego driver'a    *)
        askdriver;              (* odpytanie driver'a o konfiguracje       *)
        displayconfiguration;   (* wyswietlenie konfiguracji modulu i      *)
                                (* toru a/c                                *)
        installbreak;           (* zainstalowanie procedury obslugi        *)
                                (* przerwania generowanego przez Ctrl-Break*)
        pressanykey;            (* czekanie na operatora                   *)
        blocktransmission;      (* wykonanie transmisji blokowej +         *)
                                (* wyswietlenie zmierzonych wartosci       *)
        failblocktransmission;  (* wykonanie blednej transmisji blokowej   *)
        interruptedbefore;      (* wykonanie transmisji blokowej przerwanej*)
                                (* przez operatora przed startem           *)
        interruptedafter;       (* wykonanie transmisji blokowej przerwanej*)
                                (* przez operatora po starcie              *)
        singletransmission;     (* wykonanie pomiaru w trybie pojedynczym  *)
        quit;                   (* zakonczenie programu                    *)
end.
```

D O D A T E K E

d o d o k u m e n t a c j i d r i v e r ' a

m o d u ł u k o n t r o l n o - p o m i a r o w e g o

L C - 0 1 1 - 1 6 1 2

AMBEX - LC.ASM

struktury danych i stałe dla assemblera

```
                SUBTTL  Struktury parametrow driver'a
                PAGE +

                COMMENT #
Kazde zlecenie zaczyna sie od nizej opisanych pol.
#
LC0_HEADER      STRUC
LC0_CODE        DB      ?               ;kod funkcji
LC0_STATUS      DB      ?               ;kod bledu
LC0_ERR_STAT    DB      ?               ;dodatkowa informacja o bledach
LC0_HEADER      ENDS

                SUBTTL  Stale
                PAGE +

;===============================================================================
;
;               Stale niezbedne dla driver'a
;
;===============================================================================


;===============================================================================
;
;               Stale niezbedne dla biblioteki uzytkowej
;
;===============================================================================

;numery przerwan obslugiwanych przez driver

LC010_16        EQU     99h             ;LC-010-1612
LC011_08        EQU     90h             ;LC-011-0812
LC011_16        EQU     91h             ;LC-011-1612
LC015_16        EQU     92h             ;LC-015-1612
LC020_08_0      EQU     93h             ;LC-020-0812 v.0
LC020_08_2      EQU     94h             ;LC-020-0812 v.1 i v.2
LC020_32        EQU     95h             ;LC-020-3212
LC030_16        EQU     96h             ;LC-030-1612
LC060_06        EQU     97h             ;LC-060-0612

;-------------------------------------------------------------------------------
LC0_COND_LENGTH EQU     5       ;maksymalny rozmiar struktury opisujacej
                                ;warunek startu / stopu
;-------------------------------------------------------------------------------

;
;Kody funkcji driver'a:
;
MODULE_INIT                     EQU     0
GET_TOTAL_CONFIGURATION         EQU     1
GET_MODULE_CONFIGURATION        EQU     2
GET_INFO                        EQU     3
SET_CLOCK                       EQU     4
SET_VOLTAGE_RANGE               EQU     5
SET_TIME                        EQU     6
WAIT_FOR_END                    EQU     7
BREAK                           EQU     8
DIGITAL_INPUT                   EQU     9
DIGITAL_OUTPUT                  EQU     10
CTC_WRITE                       EQU     11
CTC_READ                        EQU     12
DATA_TRANSMIT                   EQU     13
ANALOG_INPUT                    EQU     14
ANALOG_OUTPUT                   EQU     15
LEAVE_DRIVER                    EQU     16
INTERRUPT_SERVICE               EQU     17

;-------------------------------------------------------------------------------
;
;kody odpowiedzi driver'a karty
;
;-------------------------------------------------------------------------------
;
;ostrzezenia:
;
LC0_OK                  EQU     0       ;OK
```

```
LC0_NON_EX_MOD          EQU     1       ;nie istniejacy(e) modul(y)
LC0_OTHER_LEN           EQU     2       ;przepisano mniejsza liczbe pomiarow
LC0_PREMATURE_END       EQU     3       ;przedwczesne zakonczenie z powodu
                                        ;przepelnienia bufora
LC0_IN_PROGRESS         EQU     4       ;badana transmisja jeszcze trwa

;-------------------------------------------------------------------------
;
;bledy:
;
LC0_UNKN_FUNC           EQU     -1      ;nieznany kod funkcji
LC0_NO_MODULE           EQU     -2      ;brak modulu(ow)
LC0_BAD_DEV_TYP         EQU     -3      ;bledny typ urzadzenie
LC0_NONEX_DEV           EQU     -4      ;nie istnieje urzadzenie o tym numerze
LC0_BAD_FREQ            EQU     -5      ;zla czestotliwosc zegara
LC0_BAD_RANGE           EQU     -6      ;zly zakres napiec
LC0_NO_OPER             EQU     -7      ;zadna operacja nie jest wykonywana
LC0_BAD_MARGIN          EQU     -8      ;bledna dlugosc marginesu poczatkowego
LC0_BAD_BUF_ADR         EQU     -9      ;bledny adres bufora
LC0_BAD_BUF_LEN         EQU     -10     ;bledna dlugosc bufora
LC0_DEV_BUSY            EQU     -11     ;urzadzenie jest zajete
LC0_BAD_PER             EQU     -12     ;zly okres probkowania
LC0_BAD_CHAN_N          EQU     -13     ;zla liczba kanalow
LC0_BAD_CHAN            EQU     -14     ;numer nie istniejacego kanalu
LC0_BROKEN              EQU     -15     ;przerwano funkcja BREAK
LC0_INTR_NOT_INST       EQU     -16     ;procedura obslugi przerwania nie jest
                                        ;zainstalowana
LC0_ILL_START_CODE      EQU     -17     ;nielegalny typ warunku startu
LC0_ILL_STOP_CODE       EQU     -18     ;nielegalny typ warunku stopu
LC0_BAD_PROC            EQU     -19     ;bledny adres procedury obslugi
                                        ;przerwania lub slowa komunikacyjnego
LC0_TOO_LONG_MARG       EQU     -20     ;margines dluzszy od bufora
LC0_ILL_START           EQU     -21     ;bledne parametry warunku startu
LC0_ILL_STOP            EQU     -22     ;bledne parametry warunku stopu
LC0_BAD_MNUM            EQU     -23     ;bledny numer pierwszej probki
LC0_NOT_SUPPORTED       EQU     -24     ;dla danego modulu funkcja nie jest
                                        ;realizowana
LC0_BAD_CTC_MODE        EQU     -25     ;bledny tryb pracy CTC
LC0_NO_PARAMS           EQU     -26     ;nie podano parametrow przetwarzania
LC0_OVERRUN             EQU     -27     ;zakonczono przetwarzanie z powodu
                                        ;bledu OVERRUN
LC0_NO_DMA              EQU     -28     ;z danym urzadzeniem nie jest zwiazany
                                        ;zaden kanal DMA
LC0_NO_IRQ              EQU     -29     ;z danym modulem nie jest zwiazane
                                        ;zadne przerwanie
LC0_NOT_FULLY_SUP       EQU     -30     ;zadany tryb wykonania funkcji nie
                                        ;jest realizowany dla danego typu
                                        ;modulu lub funkcja w opracowaniu
LC0_NO_EXTMEM           EQU     -31     ;brak pamieci roszerzonej
LC0_NO_SECFREQ          EQU     -32     ;modul nie moze wykonywac pomiarow z
                                        ;podwojna czestotliwoscia
LC0_INTR_INST           EQU     -33     ;procedura obslugi przerwania juz
                                        ;zainstalowana
LC0_BAD_PER2            EQU     -34     ;bledna wielokrotnosc okresu
                                        ;probkowania (0 lub 1)
LC0_BAD_MODE            EQU     -35     ;bledny tryb pracy
LC0_BAD_EXTMEM          EQU     -36     ;zly adres bufora w pam. rozszerzonej
LC0_CTC_NOT_PROGRAMMED  EQU     -37     ;zapis licznika przy
                                        ;niezaprogramowanym trybie pracy
LC0_REJECTED            EQU     -38     ;za wiele rownoleglych wejsc do
                                        ;driver'a
;-------------------------------------------------------------------------
;
;informacje dodatkowe:
;
LC0_E_OK                EQU     0       ;brak informacji dodatkowych
LC0_E_NO_MODULE         EQU     -1      ;nie ma takiego modulu
LC0_E_NONEX_DEV         EQU     -2      ;nie istnieje urzadzenie o tym numerze
LC0_E_BAD_CHAN          EQU     -3      ;numer nieistniejacego kanalu
LC0_E_BAD_TIME          EQU     -4      ;zly odcinek czasu
LC0_E_BAD_DATE          EQU     -5      ;zla specyfikacja daty
LC0_E_BAD_THRE          EQU     -6      ;bledny prog wyzwalania analogowego
LC0_E_BROKEN_WAIT       EQU     -7      ;funkcja przerwana w trakcie
                                        ;oczekiwania na spelnienie war. startu
LC0_E_BROKEN_RUN        EQU     -8      ;funkcja przerwana w trakcie
                                        ;przetwarzania
```

```
LC0_E_BAD_LEN           EQU     -9      ;za dlugi pomiar

;-------------------------------------------------------------------------
;
;struktury rekordow opisu zlecen i pozostale stale
;
;=========================================================================
; MODULE_INIT
;
LC0_INIT        STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_IMODULE     DB      ?               ;mapa modulow
LC0_INIT        ENDS
;
; Mapa modulow
;               D C B A
;               x x x x
; x = 1 zeruj modul, = 0 nie zeruj modulu
;

;=========================================================================
; GET_TOTAL_CONFIGURATION
;
LC0_TOTAL       STRUC
                DB      TYPE LC0_HEADER DUP ()
                ;PARAMETRY WYJSCIOWE
LC0_TONF        DB      ?       ;konfiguracja modulow
LC0_TIAD        DB      ?       ;liczba dostepnych przetwornikow AC
LC0_TIDA        DB      ?       ;liczba dostepnych przetwornikow CA
LC0_TCTC        DB      ?       ;liczba dostepnych kanalow CTC
LC0_TIDI        DB      ?       ;liczba dostepnych portow wejsc cyfrowych
LC0_TIDO        DB      ?       ;liczba dostepnych portow wyjsc cyfrowych
LC0_TMEMA       DD      ?       ;adres bufora w pamieci rozszerzonej
                                ;(absolutny)
LC0_TMEML       DD      ?       ;dlugosc bufora w pamieci rozszerzonej
                                ;(w probkach; 0 - brak pamieci rozszerzonej)
LC0_TOTAL       ENDS
;
; Format bajtu konfiguracji modulow
;       D B C A D C B A
;       y y y y x x x x
; x = 1 modul zaistalowany, = 0 modulu nie ma
; y = 1 modul master, = 0 modul slave
;
;Operacja zawsze konczy sie pomyslnie

;=========================================================================
; GET_MODULE_CONFIGURATION
;
LC0_MODULE      STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_MMODULE     DB      ?               ;numer modulu
                ;PARAMETRY WYJSCIOWE
LC0_MBASE1      DW      ?       ;adres bazowy rejestrow modulu (pakiet 1)
LC0_MBASE2      DW      ?       ;adres bazowy rejestrow modulu (pakiet 2)
LC0_MIAD        DB      ?       ;liczba dostepnych przetwornikow AC
LC0_MIDA        DB      ?       ;liczba dostepnych przetwornikow CA
LC0_MCTC        DB      ?       ;liczba dostepnych kanalow CTC
LC0_MIDI        DB      ?       ;liczba dostepnych portow wejsc cyfrowych
LC0_MIDO        DB      ?       ;liczba dostepnych portow wyjsc cyfrowych
LC0_MCLOCK      DW      ?       ;czestotliwosc zegara w kHz
LC0_MINT        DB      ?       ;numer przerwania (programowy)
LC0_MODULE      ENDS

;=========================================================================
; GET_INFO
;
LC0_INFO        STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_GTYPE       DB      ?               ;rodzaj urzadzenia
LC0_GMODULE     DB      ?               ;numer modulu
LC0_GNUM        DB      ?               ;numer przetwornika/bajtu/ukladu CTC
                ;PARAMETRY WYJSCIOWE
LC0_GCHAN       DB      ?               ;liczba kanalow 1)
LC0_GRES        DB      ?               ;liczba bitow przetwornika 2)
LC0_GTIME       DW      ?               ;czas konwersji przetwornika w ns 2)
```

```
LC0_GMINV       DB      ?               ;dolna granica zakresu napiec w
                                        ;dziesiatych czesciach volta 2)
LC0_GMAXV       DB      ?               ;gorna granica zakresu napiec w
                                        ;dziesiatych czesciach volta 2)
LC0_GDMA        DB      ?               ;numer kanalu DMA 2)
LC0_GMINP       DW      32 DUP (0)      ;minimalne okresy probkowania 3)
LC0_INFO        ENDS
;
; okreslonosc poszczegolnych pol dla roznych typow urzadzen:
; 1) wszystkie bez CTC
; 2) tylko a/c i c/a
; 3) tylko a/c
;

;=============================================================================
; SET_CLOCK
;
LC0_CLOCK       STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_LCLOCK      DW      ?               ;czestotliwosc zegara w kHz
LC0_CLOCK       ENDS

;=============================================================================
; SET_VOLTAGE_RANGE
;
LC0_VOLT        STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_VTYPE       DB      ?       ;rodzaj urzadzenia
LC0_VMODULE     DB      ?       ;numer modulu
LC0_VNUM        DB      ?       ;numer przetwornika
LC0_VMINV       DB      ?       ;dolna granica zakresu napiec w dziesiatych
                                ;czesciach volta
LC0_VMAXV       DB      ?       ;gorna granica zakresu napiec w dziesiatych
                                ;czesciach volta
LC0_VOLT        ENDS

;=============================================================================
; SET_TIME
;
LC0_TIME        STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_ETIME       DW      ?       ;maksymalny czas obslugi przerwania, ktore
                                ;moze pojawic sie w trakcie wykonywania
                                ;dlugiego pomiaru podawany w mikrosekundach
LC0_TIME        ENDS
;

;=============================================================================
; WAIT_FOR_END
;
LC0_WAIT        STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_WTYPE       DB      ?       ;rodzaj urzadzenia
LC0_WMODULE     DB      ?       ;numer modulu
LC0_WNUM        DB      ?       ;numer przetwornika
LC0_WMODE       DB      ?       ;tryb pracy: LC0_W_WAIT, LC0_W_TEST,
                                ;            LC0_W_FINISHED
LC0_WRMNUM      DD      ?       ;rzeczywista liczba probek
LC0_WREMAR      DW      ?       ;rzeczywista dlugosc marginesu koncowego
LC0_WAIT        ENDS
;
; kody trybu pracy:
;
LC0_W_WAIT      EQU     0       ;oczekiwanie
LC0_W_TEST      EQU     1       ;test konca transmisji
LC0_W_FINISHED  EQU     2       ;zasygnalizowanie konca

;=============================================================================
; BREAK
;
LC0_BREAK       STRUC
                DB      TYPE LC0_HEADER DUP (?)
LC0_BMODE       DB      ?               ;tryb pracy
LC0_BPROC       DD      ?               ;adres procedury obslugi przerwania
LC0_BREAK       ENDS
```

```
;
; tryby wykonania funkcji:
;
LC0_BREAK_EXEC          EQU     0       ;przerwanie
LC0_BREAK_INST          EQU     1b      ;instalacja Ctrl_Break ...
LC0_BREAK_PREV          EQU     100b    ;... z wykonaniem poprzedniej procedury
LC0_BREAK_UNINST        EQU     10b     ;wyinstalowanie wszystkiego

;=============================================================================
; DIGITAL_INPUT
;
LC0_DIGITAL_IN  STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_DMODULE     DB      ?               ;numer modulu
LC0_DNUM        DB      ?               ;numer bajtu
LC0_DSTST       DB      ?               ;typ warunku startu
LC0_DVAL        DB      ?               ;wartosc
LC0_DSTART      DB      LC0_COND_LENGTH DUP (?) ;parametry warunku startu
LC0_DIGITAL_IN  ENDS

;=============================================================================
; DIGITAL_OUTPUT
;
LC0_DIGITAL_OUT STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_OMODULE     DB      ?               ;numer modulu
LC0_ONUM        DB      ?               ;numer bajtu
LC0_OSTST       DB      ?               ;typ warunku startu
LC0_OVAL        DB      ?               ;wartosc
LC0_OSTART      DB      LC0_COND_LENGTH DUP (?) ;parametry warunku startu
LC0_DIGITAL_OUT ENDS

;=============================================================================
; CTC_WRITE
;
LC0_CTC_WRITE   STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_CMODULE     DB      ?               ;numer modulu
LC0_CMODE       DB      ?               ;tryb pracy funkcji
LC0_CFUN        DB      ?               ;tryb pracy kanalu
LC0_CVAL        DW      ?               ;wartosc licznika
LC0_CTC_WRITE   ENDS
;
; tryb pracy funkcji
;
; bit 1 - zaprogramuj tryb pracy kanalu
; bit 2 - zaladuj nowa wartosc licznika
; bit 3 - zezwolenie / zakaz CTC
;
LC0_SET_CTC_MODE        EQU     1
LC0_SET_COUNTER_VALUE   EQU     2
LC0_CTC_ENABLE          EQU     4
;
; Format bajtu trybu zegara
;     C C R R M M M B
; C: 0/1/2 - numer kanalu
; R: 1 - MSB, 2 - LSB, 3 - LSB/MSB
; M - tryb 8253/8254: 0..5
; B = 0 licznik binarny (16b), = 1 licznik BCD (4 dekady)

;=============================================================================
; CTC_READ
;
LC0_CTC_READ    STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_UMODULE     DB      ?               ;numer modulu
LC0_UNUM        DB      ?               ;numer kanalu
                ;PARAMETRY WYJSCIOWE
LC0_UVAL        DW      ?               ;wartosc licznika
LC0_CTC_READ    ENDS

;=============================================================================
; DATA_TRANSMIT
;
LC0_TRANSMIT    STRUC
                DB      TYPE LC0_HEADER DUP ()
```

```
LC0_RMODE       DB      ?       ;tryb przesylania
LC0_RADDR       DD      ?       ;adres bufora w pamieci podstawowej
LC0_RLEN        DD      ?       ;dlugosc bufora w pamieci podstawowej
LC0_RMEAS       DD      ?       ;numer pierwszej probki do przeslania
LC0_RNUM        DD      ?       ;liczba probek do przeslania
LC0_RMEMA       DD      ?       ;adres bufora w pamieci rozszerzonej
                                ;(absolutny)
                ;PARAMETRY WYJSCIOWE
LC0_RRNUM       DD      ?       ;rzeczywista liczba probek
LC0_TRANSMIT    ENDS
;
; tryb przesylania:
;
LC0_TO_EXT_DIR          EQU     1       ;podstawowa -> rozszerzona
LC0_FROM_EXT_DIR        EQU     0       ;rozszerzona / modul -> podstawowa

;=============================================================================
; ANALOG_INPUT
;
LC0_ANALOG_IN   STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_AMODULE     DB      ?       ;numer modulu
LC0_ANUM        DB      ?       ;numer przetwornika
LC0_AMODE       DW      ?       ;tryb pracy
LC0_ASTST       DB      ?       ;typy warunku startu i stopu
LC0_APER        DD      ?       ;okres probkowania
LC0_APER2       DW      ?       ;krotnosc okresu podstawowego
LC0_ACHAN       DB      ?       ;liczba kanalow podstawowych (b8 okresla czy
                                ;praca wielokanalowa (0) czy
                                ;jednokanalowa (1))
LC0_ACHAN2      DB      ?       ;liczba kanalow dodatkowych
LC0_AADDR       DD      ?       ;adres bufora w pamieci podstawowej
LC0_ALEN        DD      ?       ;dlugosc bufora
LC0_AMEMA       DD      ?       ;adres bufora w pamieci rozszerzonej
                                ;(absolutny)
LC0_ABMAR       DW      ?       ;dlugosc marginesu poczatkowego
LC0_AEMAR       DW      ?       ;dlugosc marginesu koncowego
LC0_AHAND       DW      ?       ;numer handler'a zbioru
LC0_ASTART      DB      LC0_COND_LENGTH DUP (?) ;parametry warunku startu
LC0_ASTOP       DB      LC0_COND_LENGTH DUP (?) ;parametry warunku stopu
                ;PARAMETRY WYJSCIOWE
LC0_ARDIV1      DW      ?       ;rzeczywista wartosc 1. licznika dzielacego
LC0_ARDIV2      DW      ?       ;rzeczywista wartosc 2. licznika dzielacego
LC0_ARMNUM      DD      ?       ;rzeczywista liczba probek
LC0_ARBMAR      DW      ?       ;rzeczywista dlugosc marginesu poczatkowego
LC0_AREMAR      DW      ?       ;rzeczywista dlugosc marginesu koncowego
LC0_ARLEN       DD      ?       ;rzeczywista liczba przepisanych probek
LC0_ARBUF       DW      ?       ;dla pracy z buforem cyklicznym: numer probki
                                ;okreslajacej poczatek bufora po zakonczeniu
                                ;pomiaru
LC0_ANALOG_IN   ENDS

;=============================================================================
; ANALOG_OUTPUT
;
LC0_ANALOG_OUT  STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_NMODULE     DB      ?       ;numer modulu
LC0_NNUM        DB      ?       ;numer przetwornika
LC0_NMODE       DW      ?       ;tryb
LC0_NSTST       DB      ?       ;typy warunku startu i stopu
LC0_NCHAN       DB      ?       ;liczba kanalow podstawowych (b8 okresla
                                ;czy praca wielokanalowa (0) czy
                                ;jednokanalowa (1))
LC0_NPER        DD      ?       ;okres probkowania
LC0_NADDR       DD      ?       ;adres bufora (segment:offset dla
                                ;pamieci podstawowej, absolutny dla
                                ;pamieci rozszerzonej)
LC0_NLEN        DD      ?       ;dlugosc bufora
LC0_NHAND       DW      ?       ;numer handler'a zbioru
LC0_NSTART      DB      LC0_COND_LENGTH DUP (?) ;parametry warunku startu
LC0_NSTOP       DB      LC0_COND_LENGTH DUP (?) ;parametry warunku stopu
                ;PARAMETRY WYJSCIOWE
LC0_NRMNUM      DD      ?       ;rzeczywista liczba wyslanych probek
LC0_ANALOG_OUT  ENDS
```

```
;===============================================================================
; LEAVE_DRIVER
;
LC0_LEAVE       STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_LEAVE       ENDS

;===============================================================================
; INTERRUPT_SERVICE
;
LC0_INTERRUPT   STRUC
                DB      TYPE LC0_HEADER DUP ()
LC0_SMODULE     DB      ?       ;numer modulu
LC0_SPROC       DD      ?       ;adres procedury obslugi lub 0:0
LC0_SSTAT       DD      ?       ;adres slowa komunikacyjnego
LC0_INTERRUPT   ENDS
;
; kody przyczyny przerwania
;
LC0_IS_START    EQU     1       ;przerwanie wystapilo z powodu rozpoczecia
                                ;pomiaru
LC0_IS_ONE      EQU     2       ;przerwanie nastapilo z powodu zmierzenia
                                ;kolejnej serii pomiarowej
LC0_IS_END_ADC  EQU     4       ;przerwanie nastapilo z powodu zakonczenia
                                ;przetwarzania a/c
LC0_IS_END_DAC  EQU     8       ;przerwanie nastapilo z powodu zakonczenia
                                ;przetwarzania c/a
LC0_IS_BROKEN   EQU     16      ;zakonczono przetwarzanie z powodu wykonania
                                ;funkcji BREAK
LC0_IS_SAMPLE   EQU     32      ;przerwanie nastapilo z powodu zmierzenia
                                ;kolejnej probki

;===============================================================================

;===============================================================================
;
; bity w mapie modulow
;
LC0_MODAMAP     EQU     1       ;modul A
LC0_MODBMAP     EQU     2       ;modul B
LC0_MODCMAP     EQU     4       ;modul C
LC0_MODDMAP     EQU     8       ;modul D
LC0_MODAMASTER  EQU     16      ;modul A - master
LC0_MODBMASTER  EQU     32      ;modul B - master
LC0_MODCMASTER  EQU     64      ;modul C - master
LC0_MODDMASTER  EQU     128     ;modul D - master
;
; numery modulow
;
LC0_MODA        EQU     1       ;modul A
LC0_MODB        EQU     2       ;modul B
LC0_MODC        EQU     3       ;modul C
LC0_MODD        EQU     4       ;modul D
;
; kody urzadzen
;
LC0_DINPUT      EQU     1       ;wejscie cyfrowe
LC0_DOUTPUT     EQU     2       ;wyjscie cyfrowe
LC0_AINPUT      EQU     3       ;wejscie analogowe
LC0_AOUTPUT     EQU     4       ;wyjscie analogowe
LC0_CTC         EQU     5       ;kanal CTC

;-------------------------------------------------------------------------------
;
; bity trybu pracy funkcji ANALOG_INPUT i ANALOG_OUTPUT
;
LC0_MOD_START           EQU       1     ;start pomiarow
LC0_MOD_NEW_PAR         EQU       2     ;ustawienie nowych parametrow
LC0_MOD_SYNCHR          EQU       4     ;praca synchroniczna
LC0_MOD_ASYNCHR         EQU       0     ;praca asynchroniczna
LC0_MOD_INTR            EQU       8     ;praca z przerwaniami
LC0_MOD_INTR_TYPE       EQU      10h    ;bit trybu przerwan
LC0_MOD_ONE_INTR        EQU      10h    ;przerwanie co probka
LC0_MOD_END_INTR        EQU       0     ;przerwanie na koniec
LC0_MOD_BLOCK           EQU      20h    ;praca blokowa
LC0_MOD_SINGLE          EQU       0     ;praca pojedyncza
```

```
LC0_MOD_CYCL            EQU      40h    ;bufor cykliczny
LC0_MOD_FILE_W          EQU      80h    ;zapis do pliku
LC0_MOD_FILE_R          EQU      80h    ;odczyt z pliku
LC0_MOD_MEM_W           EQU     100h    ;przepisanie do pamieci podstawowej
LC0_MOD_EXT_CLK         EQU     200h    ;zegar zewnetrzny
LC0_MOD_EXT_MEM         EQU     400h    ;bufor w pamieci rozszerzonej

;-------------------------------------------------------------------------
;
; warunki startu / stopu
;
;
; typy warunku startu:
;
LC0_SIMMED      EQU     0       ;natychmiastowy
LC0_SHARD       EQU     1       ;od sygnalu sprzetowego
LC0_SLEVEL      EQU     2       ;od poziomu sygnalu cyfrowego
LC0_SSLOPE      EQU     3       ;od zbocza sygnalu cyfrowego
LC0_SDIG_EQ     EQU     4       ;od kombinacji bitow - rowne
LC0_SDIG_NE     EQU     5       ;od kombinacji bitow - rozne
LC0_STIME       EQU     6       ;po uplynieciu okreslonego czasu
LC0_SDATE       EQU     7       ;o podanym czasie
LC0_SANALOG     EQU     8       ;od sygnalu analogowego
;
; typy warunku stopu:
;
LC0_ZSAMPLES    EQU     00h     ;po zmierzeniu okreslonej liczby probek
LC0_ZBREAK      EQU     10h     ;po wykonaniu funkcji BREAK
LC0_ZLEVEL      EQU     20h     ;od poziomu sygnalu cyfrowego
LC0_ZSLOPE      EQU     30h     ;od zbocza sygnalu cyfrowego
LC0_ZDIG_EQ     EQU     40h     ;od kombinacji bitow - rowne
LC0_ZDIG_NE     EQU     50h     ;od kombinacji bitow - rozne
LC0_ZTIME       EQU     60h     ;po uplynieciu okreslonego czasu
LC0_ZDATE       EQU     70h     ;o podanym czasie
LC0_ZANALOG     EQU     80h     ;od sygnalu analogowego

;
; opis warunku "poziom sygnalu cyfrowego"
;
LC0_COND_LEVEL  STRUC
LC0_CL_MOD      DB      ?       ;numer modulu
LC0_CL_PORT     DB      ?       ;numer portu
LC0_CL_INP      DB      ?       ;numer wejscia
LC0_CL_VALUE    DB      ?       ;oczekiwana wartosc
LC0_COND_LEVEL  ENDS

;
; opis warunku "zbocze sygnalu cyfrowego"
;
LC0_COND_SLOPE  STRUC
LC0_CS_MOD      DB      ?       ;numer modulu
LC0_CS_PORT     DB      ?       ;numer portu
LC0_CS_INP      DB      ?       ;numer wejscia
LC0_CS_VALUE    DB      ?       ;rodzaj zbocza
LC0_COND_SLOPE  ENDS
;
; kody rodzajow zbocza:
;
LC0_SLOPE_COND_UP       EQU     1       ;narastajace
LC0_SLOPE_COND_DOWN     EQU     0       ;opadajace

;
; opis warunku "kombinacja sygnalow cyfrowych (rowne/rozne)"
;
LC0_COND_DIG    STRUC
LC0_CD_MOD      DB      ?       ;numer modulu
LC0_CD_PORT     DB      ?       ;numer portu
LC0_CD_MASK     DB      ?       ;maska aktywnych wejsc
LC0_CD_PATTERN  DB      ?       ;testowany wzorzec
LC0_COND_DIG    ENDS

;
; opis warunku "data"
;
LC0_COND_DATE   STRUC
LC0_CD_SECOND   DB      ?       ;sekunda
```

```
LC0_CD_MINUTE   DB      ?       ;minuta
LC0_CD_HOUR     DB      ?       ;godzina
LC0_CD_DAY      DB      ?       ;dzien miesiaca; jezeli mniejszy niz biezacy
                                ;to nastepnego miesiaca
LC0_COND_DATE   ENDS

;
; opis warunku "sygnal analogowy"
;
LC0_COND_ANALOG         STRUC
LC0_CA_MOD              DB      ?       ;numer modulu
LC0_CA_CONVERTER        DB      ?       ;numer przetwornika
LC0_CA_CHANNEL          DB      ?       ;b1..b7 - numer kanalu
                                        ;b8 - kod kierunku przekroczenia
                                        ; 1 - w gore, 0 - w dol
LC0_CA_LEVEL            DW      ?       ;prog wyzwalania
LC0_COND_ANALOG         ENDS
;
; kody kierunku przekroczenia progu:
;
LC0_ANALOG_COND_UP      EQU     80h     ;w kierunku wiekszych wartosci
LC0_ANALOG_COND_DOWN    EQU     0       ;w kierunku mniejszych wartosci
LC0_ANALOG_COND_MASK    EQU     80h     ;maska kodu kierunku
```