Eemc ODRA 1003 / 1013

Program

STAŁY

Symbol ELWRO: 03-IV-35
Symbol ZMN UMCS: WR-Z-5

Rekonstrukcja

2023-05-25

2023-10-27

Klemens Czajka

Spis treści

Wprowadzenie

Elektroniczna maszyna cyfrowa ODRA 1013 to komputer zbudowany we Wrocławskich Zakładach Elektronicznych ELWRO, będący rozwinięciem ODRY 1003 poprzez wyposażenie w szybką pamięć ferrytową, pracującą w roli dwóch ścieżek pamięci operacyjnej, zrealizowanej generalnie na bębnie magnetycznym. Na ODRZE 1013 uczono w latach 1966-1972 programowania na Uniwersytecie Marii Curie-Skłodowskiej w Lublinie. Była to maszyna rozbudowana o obsługę czytników i perforatorów taśmy papierowej 8-kanałowej, oraz inne modyfikacje.

Ostatnia, 63. ścieżka pamięci bębnowej, zawierała zapisany na stałe, gdyż zapis na nią był zablokowany sprzętowo, program służący przede wszystkim do wczytywania do pamięci programów w sytuacji, gdy w maszynie nie ma jeszcze żadnego programu, gdyż tylko króciutkie sekwencje rozkazów da się wprowadzać bezpośrednio z pulpitu maszyny. Od tej cechy – niemożności skasowania czy uszkodzenia treści tej ścieżki – wzięła się nazwa STAŁY zapisanego na niej programu.

Program STAŁY zajmuje adresy od $17600_{(8)}$ do $17777_{(8)}$. Jest to też wg [1] jedyny program zoptymalizowany do pracy w obu reżimach adresacji: normalnej i przeplotowej. Wszystkie cechy opisane w [1], oraz drobne uzupełnienia w stosunku do oryginału, ma też wykonana przeze mnie, na potrzeby Emulatora emc ODRA 1003/1013, jego rekonstrukcja. Na zrekonstruowany program STAŁY składają się:

STAŁY – WPROWADZAJĄCY — Program

Program wczytywania do pamięci danych zapisanych na tasiemce perforowanej 5-kanałowej w kodzie PIĄTKOWYM

STAŁY – WPROWADZAJĄCY — Podprogram

Podprogram wczytywania do pamięci danych zapisanych na tasiemce perforowanej 5-kanałowej w kodzie PIĄTKOWYM – jest to rozszerzenie umożliwiające wywołanie programu WPROWADZAJĄCEGO jako podprogramu, nieistniejące w oryginale

STAŁY – WYPROWADZAJĄCY — Podprogram

Podprogram perforowania bloków pamięci w kodzie PIĄTKOWYM na taśmę perforowaną 5-kanałową

STAŁY – ZERO ZMIENNOPRZECINKOWE — Stała

Stała ZERO ZMIENNOPRZECINKOWE – liczba zero w formacie zmiennoprzecinkowym, czyli liczba stałoprzecinkowa 64 w skali 38, zapisana pod adresem $17777_{(8)}$

STAŁY – DRUKOWANIE WYKRESÓW Program

Program DRUKOWANIE WYKRESÓW jest uzupełnieniem programu STAŁEGO, nieistniejącym w oryginale. Program umożliwia wykonywanie czynności możliwej na oryginalnym samopisie MAW podczas pracy offline, tj. drukowanie wykresów wyprowadzonych wcześniej na taśmę perforowaną 8-kanałową (by nie tracić czasu maszyny przy pisaniu bezpośrednio na powolny samopis), a która to możliwość nie została zaimplementowana w emulatorze.

STAŁY – EDYTOR DALEKOPISOWY Program

Program EDYTOR DALEKOPISOWY jest uzupełnieniem programu STAŁEGO, nieistniejącym w oryginale. Program umożliwia wykonywanie czynności możliwych na oryginalnym dalekopisie podczas pracy offline, a które nie zostały zaimplementowane w emulatorze.

STAŁY – Informacja o konfiguracji systemu komputerowego Stała

Zakodowane informacje o systemie komputerowym, wśród nich takie, których w żaden inny sposób programy nie mogą uzyskać.

STAŁY – Rozpoznawanie reżimu adresowania Sposób

Wyznaczone adresy $17707_{(8)}$ i $17701_{(8)}$, pod którymi znajdują się odpowiednio ZERO i NIEZERO stałoprzecinkowe. Jest to rozszerzenie programu STAŁEGO, pozwalające rozpoznać aktualny reżim adresowania.

STAŁY – Komórki robocze programu Pamięć

Z góry wyznaczone komórki pamięci o adresach od $17572_{(8)}$ do $17577_{(8)}$, niezablokowane przed zapisem, których program STAŁY używa w trakcie pracy, jak również do przechowywania informacji pomiędzy kolejnymi wykonaniami.

Informacje o programie STAŁYM komputerów ODRA 1003 i ODRA 1013 pochodzą z:

[1] skrypt akademicki UMCS „Światomir Ząbek: Programowanie i obsługa maszyn cyfrowych ODRA 1003 i ODRA 1013, Wydanie II, LUBLIN 1974”

Autor rekonstrukcji: Klemens Czajka

Kod PIĄTKOWY

Jak powiedziano, program STAŁY służy przede wszystkim do wczytywania do pamięci operacyjnej programów w sytuacji, gdy jeszcze w maszynie nie ma żadnego programu. Dla takich programów (i danych) ustalono format ich binarnego zapisu na taśmie perforowanej w postaci kodu PIĄTKOWEGO. Program STAŁY – WPROWADZAJĄCY i WYPROWADZAJĄCY – operują na taśmach w tym formacie.

Format taśmy w kodzie PIĄTKOWYM

· ·	Przed pierwszym blokiem może być dowolna liczba znaków NU.
Blok danych	Taśma w kodzie PIĄTKOWYM zawiera bloki danych wczytywanych do spójnych obszarów pamięci operacyjnej.
Blok danych	Każdy blok zaczyna się pilotem zawierającym adres obszaru pamięci operacyjnej, do którego mają być wczytane dane bloku.
Blok danych	Pozostała część bloku zawiera serię słów danych, wczytywanych do kolejnych komórek pamięci operacyjnej, począwszy od adresu wskazanego w pilocie.
Blok danych	Serię jednego lub więcej bloków kończy blok sumy kontrolnej. Seria 8 znaków NU oznacza koniec danych bez sumy kontrolnej.
… …	Suma kontrolna jest słowem zawierającym sumę wszystkich słów danych (bez bitów nadmiarowych) i służy do kontroli poprawności wczytania danych.
Blok sumy kontrolnej	Każdy pilot, słowo danych i suma kontrolna są uzupełnione do 40 bitów jedynką z lewej strony, i zapisane na taśmie w postaci 8 kwintetów.
· ·	Po sumie kontrolnej może być dowolna liczba znaków NU – ewentualne dalsze dane nie są wczytywane.
	Pilot, dla odróżnienia od słów danych, poprzedzony jest znakiem NU. Moja rekonstrukcja zezwala na nawet do 7 takich znaków NU - seria 8 znaków NU oznacza, że nie ma kolejnego bloku danych czy sumy kontrolnej.

Format bloku danych

·	Znak NU poprzedzający pilota, odróżniający go od słowa danych.
Pilot	Pilot zawierający adres obszaru pamięci operacyjnej – jest to jedno słowo.
Słowo	Słowo danych, wczytywane do komórki pamięci o adresie podanym w pilocie.
Słowo	Następne słowo danych, wczytywane do następnej komórki pamięci.
Słowo	I następne słowo danych.
…	Itd. – kolejne słowa danych, wczytywane do kolejnych komórek pamięci.

Format bloku sumy kontrolnej

·
Pilot 17574
Suma

Znak NU poprzedzający pilota, odróżniający go od słowa danych.

Pilot zawierający adres $17574_{(8)}$ – ten adres odróżnia go od pilota bloku danych.

Słowo zawierające 39 najmłodszych bitów sumy wszystkich słów danych.

Format pilota

100·00
000·00
aaa·aa
aaa·aa
aaa·00
000·00
000·00
000·00

Pilot składa się z 8 kwintetów, od najstarszego do najmłodszego w słowie.
39-bitowe słowo jest uzupełnione z lewej bitem równym 1 (czerwony).
Na bitach aaaaaaaaaaaaa jest 13-bitowy adres obszaru pamięci operacyjnej.

Pozostałe bity pilota muszą być wyzerowane.

Format słowa danych

1xx·xx
xxx·xx
xxx·xx
xxx·xx
xxx·xx
xxx·xx
xxx·xx
xxx·xx

Słowo danych składa się z 8 kwintetów, od najstarszego do najmłodszego w słowie.
39-bitowe słowo jest uzupełnione z lewej bitem równym 1 (czerwony).

Format sumy kontrolnej

1ss·ss
sss·ss
sss·ss
sss·ss
sss·ss
sss·ss
sss·ss
sss·ss

Słowo sumy kontrolnej składa się z 8 kwintetów, od najstarszego do najmłodszego.
39-bitowe suma jest uzupełniona z lewej bitem równym 1 (czerwony).

8 znaków NU po ostatnim słowie danych oznacza brak sumy kontrolnej i jej pilota.

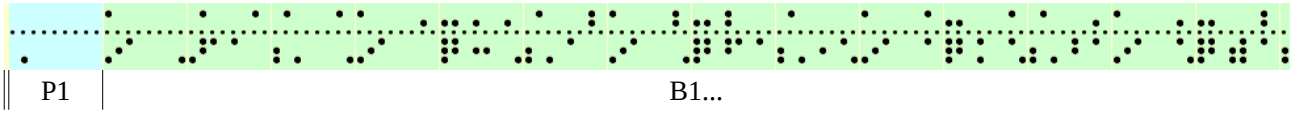
Podsumowanie taśmy w kodzie PIĄTKOWYM

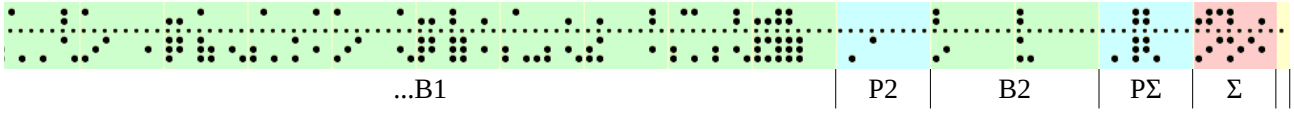
Kwintety poszczególnych słów ułożono tu w poziomie.

000·00 000·00 000·00 000·00 000·00 000·00 000·00 000·00	Na początku mogą być kody NU, a po nich pilot
000·00	Pilot jest zawsze poprzedzony jednym znakiem NU
100·00 000·00 aaa·aa aaa·aa aaa·00 000·00 000·00 000·00	Pilot: aaaaaaaaaaaaa = adres wprowadzania danych
1xx·xx xxx·xx xxx·xx xxx·xx xxx·xx xxx·xx xxx·xx xxx·xx	Słowo danych poprzedzone bitem równym 1
1xx·xx xxx·xx xxx·xx xxx·xx xxx·xx xxx·xx xxx·xx xxx·xx	Słowo danych poprzedzone bitem równym 1
... ...	...
000·00	Jeśli kolejne słowo nie powoduje nadmiaru, to jest to NU
100·00 000·00 bbb·bb bbb·bb bbb·00 000·00 000·00 000·00	a po NU kolejny Pilot wskazujący adres wprowadzania
1yy·yy yyy·yy yyy·yy yyy·yy yyy·yy yyy·yy yyy·yy yyy·yy	Słowo danych poprzedzone bitem równym 1
1yy·yy yyy·yy yyy·yy yyy·yy yyy·yy yyy·yy yyy·yy yyy·yy	Słowo danych poprzedzone bitem równym 1
... ...	...
000·00	
100·00 000·00 111·11 011·11 100·00 000·00 000·00 000·00	Ten pilot to Pilot sumy kontrolnej: adres = $17574_{(8)}$
1ss·ss sss·ss sss·ss sss·ss sss·ss sss·ss sss·ss sss·ss	Suma kontrolna = suma wszystkich słów danych
000·00	
000·00 000·00 000·00 000·00 000·00 000·00 000·00	albo po NU koniec taśmy (8 kodów NU po ostatnim słowie)

Przykładowa taśma w kodzie PIĄTKOWYM

Jest to obraz taśmy „test DzD.pgm.pt5” programu „test DzD” zamieszczonego w programotece:





W poszczególnych odcinkach taśmy znajdują się:

P1 – pilot pierwszego bloku danych (kodu programu), poprzedzony znakiem NU,

B1 – pierwszy blok danych (kod programu), zawierający 24 słowa rozkazowe,

P2 – pilot drugiego bloku danych (danych programu), poprzedzony znakiem NU,

B2 – drugi blok danych (przykładowe dane), zawierający 2 słowa liczbowe,

PΣ – pilot sumy kontrolnej, poprzedzony (a jakże) znakiem NU,

Σ – Suma kontrolna słów obu bloków danych, zawierająca 1 słowo liczbowe.

Kod THETA – 8-kanałowa alternatywa kodu PIĄTKOWEGO

Jak podaje się w [1], na maszynie UMCS, wyposażonej w czytniki i perforator taśm 8-kanałowych, programy i dane binarne przechowywano również na taśmach papierowych ośmiościeżkowych, w formacie o nazwie THETA, analogicznym do kodu PIĄTKOWEGO. W kodzie THETA taki sam był układ bloków danych i bloku sumy kontrolnej, oraz identyczny układ bitów pilota danych, pilota sumy kontrolnej, słowa danych i słowa sumy kontrolnej, zapisywanych jednak na pięciu oktetach (rządkach). Należy też przypuścić, że do zaznaczenia braku sumy kontrolnej i jej pilota wystarczyło 5 pustych rządków po ostatnim słowie danych.

Zalety kodu THETA:

- krótsze taśmy z danymi i tym samym krótszy czas wczytywania i perforowania danych
- lepsze wykorzystanie urządzeń 8-kanałowych: czytników i perforatora

Wady kodu THETA:

- brak w programie STAŁYM funkcji wczytywania i perforowania – konieczne było stosowanie zewnętrznych programów FORT (WR-OCT-1) i ANTYFORT (WR-OCT-2), które w czasie pracy zajmowały część cennej pamięci operacyjnej

Program WPROWADZAJĄCY
– rekonstrukcja

Program służy do wczytywania do pamięci danych, w tym programów, zapisanych na tasiemce perforowanej 5-kanałowej w kodzie PIĄTKOWYM.

Wczytywane są kolejne bloki danych. Każdy blok wczytywany jest do pamięci operacyjnej do kolejnych komórek począwszy od adresu podanego w pilocie bloku. Wprowadzanie trwa aż do napotkania sumy kontrolnej albo pustego odcinka taśmy złożonego z 8 pustych kwintetów.

Podczas wprowadzania zliczana jest suma wszystkich wczytywanych słów danych, przy czym ewentualny nadmiar jest ignorowany. Po napotkaniu na taśmie sumy kontrolnej, zliczona w A suma jest porównywana z sumą kontrolną poprzez obliczenie w rejestrze akumulatora A ich różnicy symetrycznej (XOR), po czym program kończy pracę. Jeżeli na taśmie nie ma sumy kontrolnej, to w A pozostaje zliczona suma wczytanych słów. Pozostawiona w A suma lub różnica jest tym samym widoczna na lampkach pulpitu. Niezerowa wartość akumulatora (świecą niektóre lampki) świadczy o niezgodności sum: zliczonej i wczytanej, albo o braku sumy kontrolnej na taśmie. W takim przypadku należy powtórzyć wczytywanie taśmy, by wykluczyć przekłamanie, albo zweryfikować istnienie i poprawność sumy na taśmie.

Program kończy pracę na rozkazie STOP:

```
:726 00000 17700 00
```

z adresem następnego rozkazu równym $17700_{(8)}$, dzięki czemu wczytanie kolejnej taśmy z kodem piątkowym wymaga jedynie założenia jej do czytnika nr 0 i wciśnięcia przycisku [StartCPU].

Sposób wczytania taśmy z kodem piątkowym z pulpitu

- założyć wczytywaną taśmę perforowaną z kodem piątkowym do czytnika nr 0
- ustawić adres $17700_{(8)}$ na panelu maszyny
- wcisnąć przycisk [ŁadR]
- wcisnąć przycisk [StartCPU]
- Po wczytaniu sumy kontrolnej lub końca taśmy program zatrzymuje się na rozkazie STOP
  `:726 00000 17700 00`
  wyświetlając na lampkach akumulatora różnicę symetryczną między sumą obliczoną, a zapisaną na taśmie. Prawidłowy stan, to zgaszone lampki akumulatora. Jeśli jakieś świecą, to przed szukaniem istotnego błędu należy upewnić się, czy to nie z powodu wciśniętego przycisku ZP na pulpicie, powodującego wyświetlanie cechy liczby zmiennoprzecinkowej na ostatnich siedmiu lampkach akumulatora.
- Aby wczytać następne bloki danych wystarczy po założeniu taśmy wcisnąć [StartCPU]

Podprogram WPROWADZAJĄCY
– rozszerzenie

Jest to rozszerzenie funkcji oryginalnego programu wprowadzającego, umożliwiające wywołanie go jako podprogramu. Może to być wygodne w razie dalszego prowadzenia obliczeń na danych wyprowadzonych przez wcześniejsze programy, pozwalając na łatwe ich wczytanie.

Możliwość wywołania programu WPROWADZAJĄCEGO jako podprogramu została dodana przeze mnie, gdyż nie istniała w oryginalnym programie STAŁYM.

Sposób wywołania z programu jako podprogram

- założyć taśmę perforowaną do czytnika 0
- wywołać jako podprogram rozkazem `:032 + 17770 60`
  w którym znak `+` można zastąpić adresem komórki, do której ma wrócić sterowanie.
  Sposób wywołania wymusza fakt, iż pamięć programu STAŁEGO jest chroniona przed zapisem, a więc niemożliwe jest użycie rozkazu SkS (TPG=746, czyli skoku ze śladem).
- Po wczytaniu sumy kontrolnej lub końca taśmy podprogram powraca po śladzie przekazanym w B6, zwracając w rejestrach:
  A – różnicę symetryczną między sumą obliczoną a zapisaną na taśmie
  B7 – sumę kontrolną danych zliczoną przy wczytywaniu
- Aby wczytać następny komplet bloków danych wystarczy ponownie wywołać podprogram

Rejestry na wejściu:

B6 – adres powrotu z podprogramu do programu wywołującego – nie naruszany
– przy uruchomieniu z pulpitu jako program, ustawienie B6 jest zbędne

Rejestry i komórki robocze:

A – wczytywane z taśmy słowo (pilot, dane, lub suma kontrolna)
B7 – zliczana przy wczytywaniu suma kontrolna danych
B1 – adres słowa zapamiętywanego w pamięci – odtwarzany
B2 – licznik wczytywanych kwintetów – odtwarzany
Komórki o adresach od $17574_{(8)}$ do $17575_{(8)}$ – tuż przed pamięcią stałą – nie używane
Komórki o adresach od $17576_{(8)}$ do $17577_{(8)}$ – tuż przed pamięcią stałą – nie odtwarzane

Rejestry na wyjściu:

A – suma obliczona, albo różnica symetryczna między nią a zapisaną na taśmie
B7 – zliczona przy wczytywaniu suma kontrolna danych
B6 – niezmieniony adres powrotu ustawiony przez program wywołujący,
– lub adres rozkazu „Stop;” jeśli program został uruchomiony z pulpitu
Rejestry B1 .. B5 pozostają nienaruszone
Jeśli wczytywane dane nie są zakończone pilotem sumy kontrolnej i sumą, lecz ośmioma znakami NU, to w rejestrze A znajdzie się to samo co w B7 – zliczona przy wczytywaniu suma kontrolna danych.

Przykład wywołania z programu jako podprogram

Lokata	Rozkaz	Komentarz
20000	`:032 + 17770 60`	Wczytanie pierwszego kompletu danych
1	`:032 + 17770 60`	Wczytanie drugiego kompletu danych z sumą kontrolną
2	`:400 + + 00`	Sprawdzenie, czy suma jest zgodna z sumą kontrolną
3	`:046 + niezg 00`	Gdy nie zero, to błąd – przejście pod adres `niezg`
4	`...`	Gdy zero – sumy są zgodne

Przykład pokazuje w poszczególnych rozkazach:

20000. Wczytanie kompletu bloków komórek (każdy blok z własnym pilotem). Nie wiemy, czy komplet ma sumę kontrolną bloków, czy nie.

20001. Wczytanie następnego kompletu bloków komórek (każdy blok z własnym pilotem). Komplet jest zakończony sumą kontrolną danych z wszystkich bloków, więc można sprawdzić poprawność wczytania.

20002. Sprawdzenie, czy w akumulatorze znajduje się zero, tj. różnica symetryczna sumy zliczonej i kontrolnej. Wartość 0 oznacza prawidłowe wczytanie drugiego kompletu bloków.

20003. Skok przy zerze do następnego rozkazu – gdy komplet bloków jest wczytany poprawnie, albo przejście pod adres `niezg` – gdy jest jakieś przekłamanie.

Logiczny tekst programu WPROWADZAJĄCEGO

Adres (8) / Etykieta	Rozkaz	Liczba kroków	Komentarz
17574	0	SumaK	robocza – nie używana
17575	0	Pilot	robocza – nie używana
17576	Robo6=	ŚladS	robocza – przechowalnia B1
17577	Robo7=	ŚladB	robocza – przechowalnia B2
			PROGRAM Z PULPITU
17700	:032 + 17770 60		B6 = CZ01; wywołanie podprogramu 17770
CZ01	:726 00000 17700 00		Stop;
			PODPROGRAM :032 + 17770 60
17770	:021 Robo6 + 10	4+	[Robo6] = B1; przechowanie B1
c11	:021 Robo7 + 20	4+	[Robo7] = B2; przechowanie B2
c12	:042 00000 + 70		B7 = A = 0; wyzerowanie A i sumy kontrolnej w B7
CZ13	:216 00005 + 00		A = A << 5, ustawienie Nd; szukanie pilota:
c14	:026 00000 + 00	4+	We0, Nd bez zmiany; wczytanie kwintetu
c15	:346 + CZ13 00		SkNd; jeśli nie nadmiar to szukać dalej
c16	:330 SumaK + 00		A - SumaK; jest pilot: czy to pilot sumy kontrolnej
c17	:046 CZ28 + 00		SkZ; gdy pilot sumy, to do wczytania sumy
c18	:402 00000 + 10		B1 = A; zwykły pilot: B1 = adres wprowadzania
CZ19	:072 00007 + 20		B2 = A = 7; śmieć w A zniknie
CZ20	:216 00005 + 00		A = A << 5; ustawienie Nd; czytanie słowa:
c21	:026 00000 + 00	4+	We0; Nd bez zmiany, ustawienie Z; wczytanie kwintetu
c22	:546 CZ20 + 20		SkLC B2--; Nd bez zmiany, Z bez zmiany
c23	:346 CZ25 + 00		SkNd; wczytano 8 kwintetów, gdy nadmiar zapisać słowo w pamięci
c24	:046 CZ32 CZ13 00		SkZ; nie nadmiar: gdy Z to koniec taśmy, gdy nie Z to pilot
CZ25	:401 00000 + 11	4+	[B1] = A; jest słowo, zapisanie go w pamięci
c26	:422 00000 + 70		B7 = A + B7; dodanie go do sumy kontrolnej
c27	:646 CZ19 CZ19 10		SkLC B1++; i do czytania następnego słowa do następnej komórki
CZ28	:072 00007 + 20		B2 = A = 7; śmieć w A zniknie
CZ29	:216 00005 + 00		A = A << 5; czytanie sumy
c30	:026 00000 + 00	4+	We0; wczytanie kwintetu
c31	:546 CZ29 + 20		SkLC B2--
CZ32	:760 00000 + 70		A = A ÷ B7; różnica symetryczna sum
c33	:012 Robo6 + 10	4+	B1 = [Robo6]; przywrócenie B1
c34	:012 Robo7 + 20	4+	B2 = [Robo7]; przywrócenie B2
c35	:000 00000 00000 61		RETURN po śladzie w B6 do programu głównego lub do Stop;

Podprogram WYPROWADZAJĄCY
– rekonstrukcja

Podprogram służy do wyprowadzania programów i/lub danych z pamięci operacyjnej na taśmę perforowaną w kodzie PIĄTKOWYM. Pozwala przy tym zadać adres wymagany w pilocie bloku różny od adresu bloku w pamięci. Jest to wygodne w odniesieniu do danych, które jeden program tworzy w pewnym obszarze pamięci operacyjnej, a kolejny wykonuje na tych danych dalsze obliczenia, ale wymaga umieszczenia ich pod innym adresem. Przesuwanie w ten sposób programów jest mało prawdopodobne, ze względu na absolutne adresy zawarte w rozkazach.

Podczas wyprowadzania podprogram zlicza sumę wyprowadzanych słów, którą dodaje do komórki pamięci o adresie $17574_{(8)}$. Przed wyprowadzeniem pierwszego bloku należy ustawić w tej komórce wartość początkową, zwykle zero. O tym, czy suma kontrolna zostanie wyprowadzona, decyduje adres wywołania podprogramu. Wywołanie poprzez adres:

- $17577_{(8)}$ wyprowadza blok danych bez sumy kontrolnej.
- $17576_{(8)}$ wyprowadza blok danych i sumę kontrolną bloków.

UWAGA: Podprogram nie wyprowadza ciągu ośmiu lub więcej znaków NU dla zaznaczenia końca danych bez sumy kontrolnej.

Wyprowadzane bloki danych mogą być poprzedzone wspólnym lub osobnym pilotem zawierającym adres wprowadzania, który może być różny od adresu wyprowadzania. Wymagany w pilocie adres wprowadzania należy podać w rejestrze B7. Jeżeli treść rejestru B7 różni się od zawartości komórki pamięci o adresie $17575_{(8)}$ (przechowującej adres wprowadzania), to zostanie wyprowadzony nowy pilot z adresem zadanym w B7 i ten nowy adres będzie od tej pory przechowywany w komórce pamięci. Jeżeli treść B7 jest zgodna z zawartością komórki $17575_{(8)}$, to pilot nie zostanie wyprowadzony – nowy blok stanie się kontynuacją poprzedniego. Przed pierwszym blokiem należy więc ustawić w komórce $17575_{(8)}$ adres różny od wymaganego w pilocie.

Sposób wywołania

- ustawić w komórce $17574_{(8)}$ początkową wartość sumy kontrolnej `s` (przed pierwszym blokiem zwykle zerową) powiększaną o sumy wyprowadzonych bloków
- ustawić w komórce $17575_{(8)}$ adres `qqqqq` w skali 21 różny od wymaganego adresu `ppppp` w pierwszym pilocie, np. liczbę -1 (potem będzie już ustawiony adres `ppppp`)
- załadować do rejestru B7 w skali 21 adres `ppppp` wymagany w pilocie
- załadować do rejestru A słowo rozkazowe `:000 ddddd nnnnn 00`,
  gdzie `ddddd` = adres pierwszej wyprowadzanej komórki pamięci,
  zaś `nnnnn` = liczba wyprowadzanych komórek
- gdy to nie jest ostatni blok i nie trzeba wyprowadzać sumy kontrolnej, to wykonać skok ze śladem do komórki $17577_{(8)}$. W tym przypadku:
  – w komórce $17575_{(8)}$ znajdzie się adres `ppppp` w skali 21, a rejestr B7 zostanie odtworzony (wartości tych nie ma potrzeby ustawiać przed wyprowadzeniem kolejnego bloku, jeśli bloki mają mieć wspólnego pilota),
  – w komórce $17574_{(8)}$ znajdzie się aktualna wartość sumy kontrolnej bloków danych (której nie należy ustawiać przed wyprowadzeniem kolejnego dosumowywanego bloku)
- gdy jest to ostatni blok i należy wyprowadzić sumę kontrolną bloków, to wykonać skok ze śladem do komórki $17576_{(8)}$. W tym przypadku wartości rejestru B7 oraz komórek $17575_{(8)}$ i $17574_{(8)}$ zostaną zniszczone.

Rejestry i komórki na wejściu:

A :000 ddddd nnnnn 00 – nigdy nie odtwarzany

B7 :000 ppppp 00000 00 – odtwarzany, …, po sumie nie odtworzony

Komórka $17574_{(8)}$ – suma kontrolna zliczana po wyprowadzeniu każdego bloku

Komórka $17575_{(8)}$ – pilot różny od pierwszego wymaganego pilota (potem równy)

Komórka $17576_{(8)}$ – ślad z wywołania podprogramu z wyprowadzaniem sumy

Komórka $17577_{(8)}$ – ślad z wywołania podprogramu bez wyprowadzania sumy

Rejestry i komórki robocze:

B1 – licznik perforowanych kwintetów – odtwarzany

B2 – adresowanie słów pobieranych z pamięci – odtwarzany

B3 – licznik wyprowadzanych słów – odtwarzany

Komórka $17572_{(8)}$ – robocza – nie odtwarzana

Komórka $17573_{(8)}$ – robocza – nie odtwarzana

Rejestry i komórki na wyjściu – gdy bez wyprowadzania sumy kontrolnej:

Rejestr B7 pozostaje nienaruszony

Komórka $17574_{(8)}$ – suma kontrolna powiększona o sumę wyprowadzonego bloku

Komórka $17575_{(8)}$ – wymagany wspólny pilot bloków danych

Pozostałe rejestry B pozostają nienaruszone

Rejestry i komórki na wyjściu – po wyprowadzeniu sumy kontrolnej:

Rejestr B7 zostanie zniszczony

Komórka $17574_{(8)}$ zostanie zniszczona

Komórka $17575_{(8)}$ zostanie zniszczona

Pozostałe rejestry B pozostają nienaruszone

Przykład 1 – wyprowadzenie bloku z pilotem i sumą

Wyprowadzenie bloku komórek $16400_{(8)}$–$16577_{(8)}$ z własnym pilotem 16400 i sumą kontrolną. Jak widać, adres wymagany w pilocie ma być zgodny z adresem wyprowadzanego bloku w pamięci operacyjnej.

Lokata	Rozkaz	Komentarz
20000	:001 17574 + 00	[17574] = 0 Suma = 0
1	:001 17575 + 00	[17575] = 0 pilot qqqqq = 0 (różny od wymaganego)
2	:032 16400 + 70	B7 = :000 16400 00000 00 pilot wymagany ppppp
3	:070 00200 + 00	A = :000 00200 00000 00 liczba komórek bloku
4	:116 00015 + 00	A = :000 00000 00200 00 przesunięcie o 13 bitów
5	:470 16400 + 00	A = :000 16400 00200 00 adres,długość bloku
6	:746 17576 + 00	Wyprowadzenie bloku z pilotem i sumą
	...	

Przykład 2 – wyprowadzenie bloku z pilotem, ale bez sumy

Wyprowadzenie bloku komórek $16400_{(8)}$–$16577_{(8)}$ z pilotem 16400 i bez sumy kontrolnej. Jak widać, adres wymagany w pilocie ma być zgodny z adresem wyprowadzanego bloku w pamięci operacyjnej. Pilot w komórce $17575_{(8)}$ ma być różny od pierwszego wymaganego pilota, ale różnica może tkwić na normalnie zerowych pozycjach bitowych poza adresem qqqqq. Skorzystamy z tego, dzięki czemu adres w komórce $17575_{(8)}$ będzie różny od dowolnego wymaganego pilota ppppp. Przykład ten różni się nadto od poprzedniego rozkazem o lokacie 20006.

Suma kontrolna być może będzie wyprowadzona wraz z następnym blokiem.

Lokata	Rozkaz	Komentarz
20000	:001 17574 + 00	[17574] = 0 Suma = 0
1	:131 17575 + 00	[17575] = -17575 qqqqq na pewno różne od ppppp
2	:032 16400 + 70	B7 = :000 16400 00000 00 pilot wymagany ppppp
3	:070 00200 + 00	A = :000 00200 00000 00 liczba komórek bloku
4	:116 00015 + 00	A = :000 00000 00200 00 przesunięcie o 13 bitów
5	:470 16400 + 00	A = :000 16400 00200 00 adres,długość bloku
6	:746 17577 + 00	Wyprowadzenie bloku z pilotem i bez sumy
7	...	

Przykład 3 – wyprowadzenie bloku bez pilota, ale z sumą

Wyprowadzenie następnego bloku komórek $12300_{(8)}$–$12377_{(8)}$ z pilotem 16400 (tym samym co w poprzednim przykładzie) i z sumą kontrolną. Adres wymagany w pilocie jest inny niż adres wyprowadzanego bloku w pamięci operacyjnej, ale zgodny adresem wymaganym w pilocie poprzedniego bloku. Zatem pilot tym razem zostanie pominięty, czyli ten blok zostanie „doklejony” do poprzedniego. Suma kontrolna obejmie całość: ten i poprzedni blok.

Ten przykład w połączeniu z poprzednim pokazuje sposób wyprowadzenia dwóch bloków komórek ze wspólnym pilotem i wspólną sumą.

Lokata	Rozkaz	Komentarz
		Suma i pilot już są ustawione w [17574], [17575] i B7
20007	:070 00100 + 00	A = :000 00100 00000 00 liczba komórek bloku
10	:116 00015 + 00	A = :000 00000 00100 00 przesunięcie o 13 bitów
11	:470 12300 + 00	A = :000 12300 00100 00 adres,długość bloku
12	:746 17576 + 00	Wyprowadzenie bloku bez pilota i z sumą obu bloków
	...	

Przykład 4 – wyprowadzenie bloków z pilotami i sumą

Wyprowadzenie dwóch bloków komórek $12300_{(8)}$–$12777_{(8)}$ i $00400_{(8)}$–$02177_{(8)}$, każdy z osobnym pilotem, i wspólną sumą. Pierwszy blok z pilotem $10000_{(8)}$ różnym od adresu bloku w pamięci, a drugi ze zgodnym. Pilot drugiego bloku zostanie wyprowadzony na tasiemkę, ponieważ różni się od pilota poprzedniego bloku.

Lokata	Rozkaz	Komentarz
20000	:001 17574 + 00	[17574] = 0 Suma = 0
1	:131 17575 + 00	[17575] = -17575 pilot qqqqq różny od ppppp
2	:032 10000 + 70	B7 = :000 10000 00000 00 pilot wymagany ppppp
3	:070 00500 + 00	A = :000 00500 00000 00 liczba komórek bloku
4	:116 00015 + 00	A = :000 00000 00500 00 przesunięcie o 13 bitów
5	:470 12300 + 00	A = :000 12300 00500 00 adres,długość bloku
6	:746 17577 + 00	Wyprowadzenie bloku z pilotem i bez sumy
		Suma i poprzedni pilot są ustawione w [17574], [17575]
20007	:032 00400 + 70	B7 = :000 00400 00000 00 pilot wymagany ppppp
10	:070 01600 + 00	A = :000 01600 00000 00 liczba komórek bloku
11	:116 00015 + 00	A = :000 00000 01600 00 przesunięcie o 13 bitów
12	:470 00400 + 00	A = :000 00400 01600 00 adres,długość bloku
13	:746 17576 + 00	Wyprowadzenie bloku z pilotem i z sumą obu bloków
	...	

Przykład 5 – wyprowadzenie samej sumy

Wyprowadzenie sumy naliczonej przy wyprowadzaniu bloków, bez wyprowadzania kolejnego bloku. Jest to przydatne, gdy w chwili wyprowadzania bloku nie wiemy, czy jest to ostatni blok.

Lokata	Rozkaz	Komentarz
		Suma i pilot już są ustawione w [17574], [17575] i B7
20014	:040 00000 + 00	A = :000 00000 00000 00 adres=0,długość bloku=0
5	:746 17576 + 00	Wyprowadzenie 0 słów bloku bez pilota i z sumą
	...	

Logiczny tekst podprogramów WYPROWADZAJĄCYCH – Z SUMĄ i BEZ SUMY KONTROLNEJ

Adres (8) / Etykieta	Rozkaz	Liczba kroków	Komentarz
17572	0	Robo2	robocza
17573	0	Robo3	robocza
17574	0	SumaK	suma kontrolna
17575	:000 qqqqq 00000 00	Pilot	adres w skali 21 różny od ppppp wymaganego w pierwszym pilocie
17576	:000 17576 powrS 00	ŚladS	ślad – wyprowadzenie bloku słów i sumy kontrolnej
17577	:000 17577 powrB 00	ŚladB	ślad – wyprowadzenie bloku słów bez sumy kontrolnej
			PODPROGRAM Z SUMĄ
17600	:746 ŚladB WY36 00	4+	SkS ŚladB; wywołanie podprogramu (BEZ SUMY)
			PODPROGRAM BEZ SUMY
17601	:401 Robo2 + 00	4+	[Robo2] = A; = :000 ddddd nnnnn 00 przechowanie A
w02	:060 00000 + 10		A = B1; = :000 11111 00000 00
w03	:416 00015 + 00		A = A >>< 13; = :000 00000 11111 00
w04	:460 00000 + 20		A = A + B2; = :000 22222 11111 00
w05	:416 00015 + 00		A = A >>< 13; = :111 00000 22222 11
w06	:421 Robo3 + 30	4+	[Robo3] = A+B3; = :111 33333 22222 11 przechowanie B3,B2,B1
w07	:052 Robo2 + 20	4+	B2 = A=[Robo2]; = :000 ddddd nnnnn 00 B2 = ddddd;
w08	:016 00015 + 00		A = A <<< 13; = :ddd nnnnn 00000 00
w09	:402 00000 + 30		B3 = A; = nnnnn B3 = nnnnn;
w10	:060 00000 + 70		A = B7; = :000 ppppp 00000 00
w11	:012 SumaK + 70	4+	B7 = [SumaK]; = suma kontrolna
w12	:310 Pilot + 70	4+	A - [Pilot]; jeśli ppppp == qqqqq
w13	:046 WY24 + 00		to wyprowadzić blok bez pilota
			WYPROWADZANIE PILOTA
w14	:526 00000 + 00	4+	Wy5; wyprowadzenie 00000
w15	:401 Pilot + 00	4+	[Pilot] = A; = :000 ppppp 00000 00 nowy pilot do pamięci
			WYPROWADZANIE KWINTETÓW PILOTA / SŁOWA
WY16	:116 00001 + 00		A = A >>> 1; $\Omega = A_{38}$
w17	:450 WY35 + 00	4+	$A_0 = A_0 + 1$; 1 na bit znaku
w18	:032 00007 + 10		B1 = 7; licznik kwintetów
WY19	:526 00000 + 00	4+	Wy5; wyprowadzenie kwintetu
w20	:016 00001 + 00		A = A <<< 1; zrobienie miejsca na bit Omega
w21	:426 00000 + 00		$A_{38} = A_{38} + \Omega$; i nieszkodliwe powielanie bitu Omega
w22	:016 00004 + 00		A = A <<< 4;
w23	:546 WY19 + 10		SkLC B1--;
			POBRANIE KOLEJNEGO SŁOWA
WY24	:546 + WY28 30		SkLC B3--; jeśli brak słowa do wyprowadzenia to do SUMY
w25	:050 00000 + 21	4+	A = [B2]; słowo do wyprowadzenia
w26	:646 + + 20		SkLC B2++; przesunięcie adresu na następne słowo
w27	:422 00000 WY16 70		B7 = A + B7; dodanie słowa do sumy kontrolnej

			ZAPAMIĘTANIE SUMY I ZAKOŃCZENIE
WY28	:021 SumaK + 70	4+	[SumaK] = B7; zapisanie sumy kontrolnej
w29	:052 Robo3 + 30		B3 = A=[Robo3]; = :111 33333 22222 11 przywrócenie B3
w30	:416 00015 + 00		A = A >>< 13; = :222 11111 33333 22
w31	:402 00000 + 10		B1 = A; przywrócenie B1
w32	:416 00015 + 00		A = A >>< 13; = :333 22222 11111 33
w33	:402 00000 + 20		B2 = A; przywrócenie B2
w34	:012 Pilot ŚladB 70	4+	B7 = [Pilot]; przywrócenie B7; RETURN
WY35	:400 00000 00000 00		jedynka na bicie znaku A_0
			CD. PODPROGRAMU Z SUMĄ
WY36	:050 WY39 + 00	4+	A = :000 SumaK 00001 00 adres i długość bloku sumy
w37	:032 SumaK + 70		B7= :000 SumaK 00000 00 pilot sumy
w38	:746 ŚladB ŚladS 00	4+	wywołanie podprogramu ŚladB; wyprowadzenie sumy i RETURN
			bez odtworzenia B7
WY39	:000 SumaK 00001 00		adres i długość bloku sumy kontrolnej

Stała ZERO ZMIENNOPRZECINKOWE

Jest to umieszczona dla wygody programów, pod adresem $17777_{(8)}$

– liczba zero w formacie zmiennoprzecinkowym,
– czyli liczba stałoprzecinkowa, dodatnia, o wartości 64 w skali 38.

Logiczny tekst stałej ZERO ZMIENNOPRZECINKOWE

Adres (8) / Etykieta	Rozkaz	Liczba kroków	Komentarz
17777	:000 00000 00004 00		liczba=0 zmiennoprzecinkowo, lub =64 stałoprzecinkowo w skali 38

Program DRUKOWANIE WYKRESÓW
– uzupełnienie

Jest to dodane przeze mnie uzupełnienie programu STAŁEGO, nieistniejące w oryginale. Program umożliwia wykonywanie na samopisie MAW operacji, które wykonywano offline, a które nie są zaimplementowane w emulatorze dalekopisu. Program istnieje jedynie w wersjach STAŁEGO na ODRY UMCS (wyposażone w urządzenia 8-kanałowe); w przeciwnym razie pod adresem $17657_{(8)}$ znajduje się rozkaz STOP.

Program składa się z trzech, działających w nieskończonej pętli, rozkazów przepisywania danych z czytnika 8-kanałowego PTR2’ na perforator i/lub samopis 8-kanałowy.

Sposób uruchomienia programu z pulpitu

- przygotować dane wejściowe w postaci taśmy 8-kanałowej w czytniku PTR2’
- uruchomić lub zatrzymać przyciskiem (O) stosowne urządzenia wyjściowe 8-kanałowe:
  - samopis MAW
  - i/lub perforator
- wcisnąć lub zwolnić przycisk (A) na samopisie MAW, stosownie do sposobu sterowania wysuwem papieru wymaganym przy wydruku wykresu:
  – wcisnąć gdy po każdym postawionym punkcie ma nastąpić automatyczny wysuw papieru
  – zwolnić gdy wysuwem papieru sterują kody znajdujące się na taśmie z danymi
- przyciskami rejestru R na panelu maszyny ustawić adres $17657_{(8)}$
- wcisnąć przycisk [ŁadR]
- wcisnąć przycisk [StartCPU]
- na zakończenie oderwać wynikowy wydruk wykresu i/lub wyperforowaną tasiemkę
- program wymaga zatrzymania przyciskiem [StopCPU], po czym trzeba będzie pozwolić na wczytanie jeszcze jednego, dowolnego oktetu z taśmy wejściowej

Logiczny tekst programu DRUKOWANIE WYKRESÓW

Adres $_{(8)}$ / Etykieta	Rozkaz	Liczba kroków	Komentarz
17657	:266 00000 + 00		XWe2; albo STOP :726 00000 17657 00
MAW1	:016 00037 + 00		LL 31;
MAW2	:566 00000 17657 00		XWy5;

Program EDYTOR DALEKOPISOWY
– uzupełnienie

Jest to dodane przeze mnie uzupełnienie programu STAŁEGO, nieistniejące w oryginale. Program umożliwia wykonywanie na dalekopisie operacji, które wykonywano offline, a które nie są zaimplementowane w emulatorze dalekopisu: przygotowywanie tekstów (np. programów źródłowych) na taśmach perforowanych, kopiowanie taśm 5-kanałowych, oddrukowywanie taśm, a wszystko to z możliwością usuwania błędów i korygowania treści. Jest on tak krótki, że z łatwością można wprowadzać go z pulpitu w dowolne miejsce pamięci.

Program składa się z jednego rozkazu czytania z dalekopisu, działającego w nieskończonej pętli. Czytane dane można wprowadzać z klawiatury wirtualnej dalekopisu, z klawiatury komputera-gospodarza, z tasiemki perforowanej założonej do czytnika dalekopisu, albo z pliku tekstowego wyręczającego klawiaturę. Dane mogą być jednocześnie automatycznie drukowane na dalekopisie i/lub perforowane na perforatorze dalekopisu. Ponieważ dalekopis umożliwia przełączanie w dowolnym momencie wejścia: klawiatura / czytnik / plik, oraz pozycjonowanie tasiemki w czytniku i tekstu w pliku, to można swobodnie manipulować danymi wejściowymi i uzyskiwać skorygowane dane na wyjściu. Emulator ułatwia takie manipulacje dzięki możliwości start/stopowej pracy ODRY, oraz przyspieszania i zwalniania pracy maszyny suwakiem prędkości, zatrzymywania i startowania urządzeń wejściowych, oraz wyłączania i włączania pracy drukarki i perforatora.

Szczególną uwagę należy zwrócić na kopiowanie treści z pliku tekstowego wyręczającego klawiaturę. Jest to specyficzna możliwość oferowana przez emulator, nieistniejąca w rzeczywistym sprzęcie. Plik tekstowy może zawierać znaki nie istniejące w repertuarze dalekopisu. Znaki takie są przez emulator zastępowane spacją, a nie ignorowane jak w przypadku pisania na klawiaturze. Jedynym wyjątkiem jest znak tabulacji, zamieniany na serię od 1 do 8 spacji, co ma zapewnić zachowanie układu graficznego tekstu. Należy przy tym pamiętać, że liczba tych spacji zależy od aktualnej pozycji karetki dalekopisu, a więc przed kopiowaniem nowego tekstu dobrze jest zapewnić właściwą pozycję karetki wyprowadzeniem kodów przejścia do nowego wiersza.

Sposób uruchomienia programu z pulpitu

- przygotować dane w urządzeniach wejściowych dalekopisu:
  – taśmę w czytniku
  – plik tekstowy wyręczający klawiaturę
  – albo przełączyć wejście dalekopisu na klawiaturę
- ustawić suwakiem odpowiednią prędkość emulatora, a potem odpowiednio zmieniać:
  – mniejszą dla danych z czytnika lub pliku tekstowego (nawet start/stopową),
  – większą przy pisaniu na klawiaturze
- uruchomić lub zatrzymać przyciskiem (O) odpowiednie urządzenia wyjściowe dalekopisu: drukarkę i/lub perforator
- przyciskami rejestru R na panelu maszyny ustawić adres $17654_{(8)}$
- wcisnąć przycisk [ŁadR]
- wcisnąć przycisk [StartCPU]
- wykonywać potrzebne manipulacje: przełączanie wejść, pozycjonowanie tasiemki i tekstu, korygowanie treści na klawiaturze
- na zakończenie oderwać wynikowy wydruk i/lub wyperforowaną tasiemkę
- program wymaga zatrzymania przyciskiem [StopCPU], po czym trzeba będzie wprowadzić jeszcze jeden znak, np. kod NU

Logiczny tekst programu EDYTOR DALEKOPISOWY

Adres $_{(8)}$ / Etykieta	Rozkaz	Liczba kroków	Komentarz
17654	:126 infor 17654 00		We1;

Rozkaz w komórce o adresie $17654_{(8)}$ ma na pierwszej części adresowej, AR, obojętnej dla działania, opisaną dalej, zakodowaną informację o konfiguracji systemu komputerowego.

Informacja o konfiguracji systemu komputerowego

Instalacje komputerowe mogą być różne – inne są modele komputera, czy typy dalekopisu. Odróżnienie ODRY UMCS od niezmodyfikowanej ODRY, albo alfabetu dalekopisu i sposobu sterowania nim, nie jest możliwe przez sam program komputerowy bez dodatkowego wsparcia. Emulator każdą instalację wyposaża w program STAŁY zawierający informacje o jej konfiguracji sprzętowej, w tym takie, których program komputerowy nie jest w stanie inaczej uzyskać. Informacja ta nie wpływa na działanie komputera i programu STAŁEGO, a umożliwia programom wygodne jej pobranie i sprawdzenie, czy np. komputer odpowiada wymaganiom.

Informacje o konfiguracji systemu komputerowego są zakodowane w komórce o adresie $17654_{(8)}$, na poszczególnych bitach pierwszej części adresowej, AR, słowa:

									AR																									
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
.	.	.	.	.	.	.	.	.	F	U	bez CR	.	.	.	.	Alfabet				Ster		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

F – model komputera:
- =0 ODRA 1003, czyli bez pamięci ferrytowej
- =1 ODRA 1013, czyli z pamięcią ferrytową i rozkazami przesyłania blokowego

U – modyfikacja UMCS komputera:
- =0 to nie jest ODRA UMCS, nie ma rozkazów we/wy 8-kanałowego i rozkazu SkBG
- =1 ODRA UMCS, ma rozkazy we/wy 8-kanałowego i rozkaz skoku przy braku gotowości

bez CR – jakie znaki końca wiersza w wydruku z dalekopisu są generowane przez kod LF:
- =0 para znaków (CR,LF) – wg dawnej konwencji, stosowanej też w systemach Windows
- =1 pojedynczy znak (LF) – wg konwencji stosowanej w systemach Unix/Linux

Alfabet – dalekopis ze znakami języka:

=0	polskiego (PL)	– MKD-2 PL2,	MKD-2 PL3,	MKD-2 PL4
=1	angielskiego brytyjskiego (GB)	– ITA2 GB2,	ITA2 GB3,	ITA2 GB4
=2	niemieckiego (DE)	– ITA2 DE2		
=3	francuskiego (FR)	– ITA2 FR2		
=4	skandynawskich (SC)	– ITA2 SC2		
=5	angielskiego amerykańskiego (US)	– ITA2 US2		
=7	rosyjskiego (RU)	– MTK-2 РУ3		

Ster – sposób sterowania pocztami znaków dalekopisu:
- =00 dalekopis z dwoma pocztami znaków, sterowany kodami LS i FS, jak np.: MKD-2 PL2, ITA2 GB2, ITA2 DE2, ITA2 FR2, ITA2 SC2, ITA2 US2
- =01 dalekopis z trzema pocztami znaków, sterowany kodami LS i FS, jak np.: MKD-2 PL3, ITA2 GB3
- =10 dalekopis z czterema pocztami znaków, sterowany kodami LS i FS, jak np.: MKD-2 PL4, ITA2 GB4
- =11 dalekopis z trzema pocztami znaków, sterowany kodami LS, FS, oraz NU, jak np.: MTK-2 РУ3

Możliwe są wszystkie 4 kombinacje bitów F i U. Pozostałe bity części AR słowa są zarezerwowane.

UWAGA: Informacja o konfiguracji pobrana przez program na bęben może być nieaktualna, jeśli przeniesiono bęben do innej instalacji komputerowej – tj. zatrzymano program, wyłączono maszynę i/lub zasilanie, zmieniono komputer i/lub dalekopis i wznowiono ciągle istniejący na bębnie program ☺

Sposób odróżnienia ODRY 1013 od ODRY 1003

Wystarczy sprawdzić bit F słowa w komórce o adresie $17654_{(8)}$:

Lokata	Rozkaz	Komentarz
20000	:050 17654 + 00	A = [17654]; pobranie treści komórki [17654] do A
1	:630 10000 + 00	A & 4096; czy bit A_9 = 1
2	:046 bęben + 00	SkZ bęben; skok gdy bit A_9 = 0
ferryt	...	Bit ustawiony – to jest ODRA 1013
bęben	...	Bit wyzerowany – to jest ODRA 1003

Można też bez odwoływania się do pamięci STAŁEJ, sprawdzając, czy w maszynie istnieje pamięć ferrytowa, tj. czy działa rozkaz przesyłania blokowego. Test taki nie jest związany z programem STAŁYM, ale zamieszczam go tu uzupełniająco.

Lokata	Rozkaz	Komentarz
20000	:131 17577 + 00	[17577] = -17577; liczba ujemna do kom. ferrytowej
1	:236 17777 + 00	Przesłanie blokowe zera zmp z komórki bębn. do ferr.
2	:010 17577 + 00	[17577]; czy zmieniła się treść komórki ferr.
3	:146 bęben + 00	SkU bęben; skok gdy nie zmieniła się
ferryt	...	Zmieniła się, więc jest pamięć ferrytowa
bęben	...	Nie zmieniła się – przesyłanie blokowe nie działa

Czy to ODRA UMCS, albo jaki jest dalekopis

Sprawdzenie modelu ODRY, modyfikacji UMCS, czy typu zainstalowanego dalekopisu, sprowadza się do pobrania i inspekcji odpowiednich bitów pierwszej części adresowej słowa o adresie $17654_{(8)}$ programu STAŁEGO, jak np.:

Lokata	Rozkaz	Komentarz
20000	:050 17654 + 00	A = [17654]; pobranie treści komórki [17654] do A
	bit A_9 = 1 – ODRA 1013 bit A_{10} = 1 – ODRA UMCS (może to być ODRA 1003 UMCS, lub ODRA 1013 UMCS) bity $A_{16..19}$ = nr języka – zestawu znaków dalekopisu bity $A_{20..21}$ = 00, 01, 10 – dalekopis z dwoma, trzema, czterema pocztami znaków, sterowany jak np. MKD-2 PL2, MKD-2 PL3, MKD-2 PL4 bity $A_{20..21}$ = 11 – dalekopis z trzema pocztami znaków, sterowany jak MTK-2 PY3	
1	:630 04000 + 00	A & 2048; czy to jest ODRA UMCS
2	:046 nieU + 00	SkZ nieU; skok gdy to nie jest ODRA UMCS
UMCS	...	Bit A_{10} niezerowy – to jest ODRA UMCS
nieU	...	Bit A_{10} wyzerowany – to nie jest ODRA UMCS

Rozpoznawanie reżimu adresowania
– rozszerzenie

W wykonanej rekonstrukcji programu STAŁEGO pozostały wolne komórki, które są wyzerowane. Adresy tych komórek w normalnym reżimie adresowania nie pokrywają się z adresami w reżimie przeplotowym. Istnienie niewykorzystanych komórek może posłużyć rozpoznaniu aktualnego reżimu adresowania. Do tego celu zostały przeznaczone dwa adresy: $17707_{(8)}$ i $17701_{(8)}$.

Komórka o adresie normalnym $17707_{(8)} = 17701_{(8)}$ (przeplotowo) zawsze będzie wyzerowana.

Komórki o adresach: przeplotowym $17707_{(8)}$ i normalnym $17701_{(8)}$, zawsze będą niezerowe.

Nie jest ustalone, jaka konkretnie ma być zawartość komórek niezerowych. Oto aktualna ich treść:

	Etykieta / adres$_{(8)}$ przeplotowo	Etykieta / adres$_{(8)}$ normalnie	Rozkaz		Komentarz
niezero	w11p 17767	17701	:012 17574 17617 70	4+	B7 = [SumaK]; = suma kontrolna
	...	...	...		
zero	**17701**	**17707**	**:000 00000 00000 00**		**zero stałoprzecinkowe**
	...	...	...		
niezero	17707	c31n 17761	:546 17732 17650 20		SkLC B2--

Sposób rozpoznania aktualnego reżimu adresowania

Sprawdzenie, czy zawartość komórki o adresie $17707_{(8)}$ jest zerowa:

Lokata	Rozkaz	Komentarz
20000	:010 17707 + 00	Pobranie [17707] do sumatora
1	:046 2norm + 00	Skok do 2norm, gdy pobrano zero
2	...	Adresowanie jest przeplotowe
2norm	...	Adresowanie jest normalne

Można też inaczej – sprawdzić, czy zawartość komórki o adresie $17701_{(8)}$ jest niezerowa:

Lokata	Rozkaz	Komentarz
20000	:010 17701 + 00	Pobranie [17701] do sumatora
1	:046 + 2norm 00	Skok do 2norm, gdy pobrano niezero
2	...	Adresowanie jest przeplotowe
2norm	...	Adresowanie jest normalne

Pamięć STAŁA – zawartość ścieżki numer 63
– rekonstrukcja

Komórki robocze

Ścieżka z programem STAŁYM jest zabezpieczona przed zapisem, więc do dyspozycji programu przeznaczono 6 zapisywalnych komórek pamięci znajdujących się tuż przed pamięcią stałą (w ODRZE 1013 są to komórki na końcu pamięci ferrytowej):

- $17572_{(8)}$ – komórka robocza
- $17573_{(8)}$ – komórka robocza
- $17574_{(8)}$ – komórka robocza, lub przechowalnia sumy kontrolnej bloków
- $17575_{(8)}$ – komórka robocza, lub przechowalnia ostatniego pilota
- $17576_{(8)}$ – komórka robocza, lub ślad (punkt wejścia) do podprogramu WYPROWADZAJĄCEGO blok słów Z SUMĄ KONTROLNĄ
- $17577_{(8)}$ – komórka robocza, lub ślad (punkt wejścia) do podprogramu WYPROWADZAJĄCEGO blok słów BEZ SUMY KONTROLNEJ

Każde użycie programu STAŁEGO ustawia lub niszczy zawartość tych komórek, ale programy użytkownika mogą umiejętnie używać ich do własnych celów. Komórki $17574_{(8)}$ i $17575_{(8)}$ przechowują dane pomiędzy kolejnymi wywołaniami **podprogramu wyprowadzającego**, zatem wolno ich używać tylko wtedy, gdy nie używa się tego podprogramu. Pozostałych komórek można używać w interwałach pomiędzy wywołaniami programu STAŁEGO, ale się tego nie zaleca.

Istotne adresy programu STAŁEGO

Oto adresy o ustalonym znaczeniu, niezależnym od wersji programu STAŁEGO:

- $17600_{(8)}$ – pierwszy rozkaz podprogramu WYPROWADZAJĄCEGO blok słów Z SUMĄ KONTROLNĄ – ślad w komórce $17576_{(8)}$
- $17601_{(8)}$ – pierwszy rozkaz podprogramu WYPROWADZAJĄCEGO blok słów BEZ SUMY KONTROLNEJ – ślad w komórce $17577_{(8)}$
- $17654_{(8)}$ – program STAŁY – EDYTOR DALEKOPISOWY umożliwiający oddrukowywanie, reperforowanie, korygowanie danych na dalekopisie
  – w rozkazie pod tym adresem są zakodowane informacje o systemie komputerowym
- $17657_{(8)}$ – program STAŁY – DRUKOWANIE WYKRESÓW umożliwiający kopiowanie wykresów z 8-kanałowej taśmy perforowanej, na samopis MAW (w systemach komputerowych bez samopisu MAW, jest tu rozkaz Stop)
- $17700_{(8)}$ – program STAŁY – WPROWADZAJĄCY blok słów z sumą lub bez sumy kontrolnej, uruchamiany z pulpitu
- $17707_{(8)}$ (nieprzeplotowo) = $17701_{(8)}$ (przeplotowo) – zero stałoprzecinkowe, w przeciwnych reżimach adresowania pod tymi adresami są niezera
- $17770_{(8)}$ – podprogram STAŁY – WPROWADZAJĄCY blok słów z sumą lub bez sumy kontrolnej, uruchamiany rozkazem `:032 + 17770 60`
- $17777_{(8)}$ – zero zmiennoprzecinkowe, czyli liczba stałoprzecinkowa 64 w skali 38

Niewykorzystane komórki ścieżki STAŁEJ są obecnie wyzerowane, ale może to się zmienić wraz ze zmianą wersji programu.

Zawartość ścieżki STAŁEJ

Programy i podprogramy działają w obu reżimach adresacji: normalnym i przeplotowym, a także są optymalizowane pod kątem czasu pracy, stąd rozkazy są rozrzucone po całej ścieżce.

Strefy komórek roboczych	Etykieta / adres$_{(8)}$ przeplotowo	Etykieta / adres$_{(8)}$ normalnie	Rozkaz		Komentarz
Robo2	Robo2 17572	Robo2 17572	0		robocza
Robo3	Robo3 17573	Robo3 17573	0		robocza
Robo4	SumaK 17574	SumaK 17574	0		suma kontrolna
Robo5	Pilot 17575	Pilot 17575	:000 qqqqq 00000 00		poprzedni pilot
Robo6	ŚladS 17576	ŚladS 17576	:000 17576 powrS 00		Ślad: wyprowadzenie bloku i sumy – robocza
Robo7	ŚladB 17577	ŚladB 17577	:000 17577 powrB 00		Ślad: wyprowadzenie bloku bez sumy – robocza
	WYS00 17600	**WYS00 17600**	**:746 17577 17725 00**	**4+**	**SkS ŚladB; wywołanie podprogr. (BEZ SUMY)**
	17667	**WYB1n 17601**	**:401 17572 17775 00**	**4+**	**[Robo2] = A; = :000 ddddd nnnnn 00**
	17756	w03n 17602	:416 00015 17622 00		A = A >>< 13; = :000 00000 11111 00
	w08p 17645	w08n 17603	:016 00015 17610 00		A = A <<< 13; = :ddd nnnnn 00000 00
	17734	w13n 17604	:046 17641 17640 00		to wyprowadzić blok bez pilota
	WY16p 17623	WY16n 17605	:116 00001 17625 00		A = A >>> 1, Ω=A_{38}
	c35p 17712	c35n 17606	:000 00000 00000 61		RETURN po śladzie w B6 do programu lub Stop;
	WYB1p 17601	**17607**	**:401 17572 17733 00**	**4+**	**[Robo2] = A; = :000 ddddd nnnnn 00**
	17670	w09n 17610	:402 00000 17675 30		B3 = A; = nnnnn B3 = nnnnn;
	c12p 17757	17611	:042 00000 17760 70		A = B7 = 0; wyzerowanie A i sumy w B7
	17646	c12n 17612	:042 00000 17621 70		A = B7 = 0; wyzerowanie A i sumy w B7
	c24p 17735	17613	:046 17741 17760 00		SkZ; nie nadmiar: Z koniec taśmy, nie Z pilot
	w30p 17624	17614	:416 00015 17763 00		A = A >>< 13; = :222 11111 33333 22
	17713	c24n 17615	:046 17650 17621 00		SkZ; nie nadmiar: Z koniec taśmy, nie Z pilot
	w03p 17602	17616	:416 00015 17736 00		A = A >>< 13; = :000 00000 11111 00
	c26p 17671	17617	:422 00000 17716 70		B7 = A + B7; dodanie go do sumy kontrolnej
	CZ13p 17760	17620	:216 00005 17650 00		A = A << 5, ustawienie Nd; szukanie pilota:
	17647	CZ13n 17621	:216 00005 17633 00		A = A << 5, ustawienie Nd; szukanie pilota:
	w04p 17736	w04n 17622	:460 00000 17626 20		A = A + B2; = :000 22222 11111 00
	w17p 17625	17623	:450 17761 17737 00	4+	A_0 = A_0 + 1; 1 na bit znaku
	17714	w30n 17624	:416 00015 17645 00		A = A >>< 13; = :222 11111 33333 22
	17603	w17n 17625	:450 17627 17631 00	4+	A_0 = A_0 + 1; 1 na bit znaku
	17672	w05n 17626	:416 00015 17636 00		A = A >>< 13; = :111 00000 22222 11
	WY35p 17761	WY35n 17627	:400 00000 00000 00		jedynka na bicie znaku A_0
	c14p 17650	17630	:026 00000 17765 00	4+	We0, Nd bez zmiany; wczytanie kwintetu
	w18p 17737	w18n 17631	:032 00007 17651 10		B1 = 7; licznik kwintetów
	w05p 17626	17632	:416 00015 17762 00		A = A >>< 13; = :111 00000 22222 11
	17715	c14n 17633	:026 00000 17665 00	4+	We0, Nd bez zmiany; wczytanie kwintetu
	w13p 17604	17634	:046 17627 17740 00		to wyprowadzić blok bez pilota
	17673	c26n 17635	:422 00000 17643 70		B7 = A + B7; dodanie go do sumy kontrolnej
	w06p 17762	w06n 17636	:421 17573 17776 30	4+	[Robo3] = A+B3 = :111 33333 22222 11
	WY19p 17651	17637	:526 00000 17615 00	4+	Wy5; wyprowadzenie kwintetu
	w14p 17740	w14n 17640	:526 00000 17752 00	4+	Wy5; wyprowadzenie 00000
	WY24p 17627	WY24n 17641	:546 17652 17653 30		SkLC B3-- jeśli brak słowa, to do SUMY
	c27p 17716	17642	:646 17613 17613 10		SkLC B1++ do czyt. nast.słowa do nast.kom.
	17605	c27n 17643	:646 17716 17716 10		SkLC B1++ do czyt. nast.słowa do nast.kom.

	CZ01p 17674		17644	:726 00000 17700 00		Stop;	
	w31p 17763	w31n 17645		:402 00000 17677 10		B1 = A;	przywrócenie B1
	w25p 17652		17646	:050 00000 17742 21	4+	A = [B2];	słowo do wyprowadzenia
	CZ32p 17741		17647	:760 00000 17642 70		A = A ÷ B7;	różnica symetryczna sum
	17630	CZ32n 17650		:760 00000 17773 70		A = A ÷ B7;	różnica symetryczna sum
	17717	WY19n 17651		:526 00000 17734 00	4+	Wy5;	wyprowadzenie kwintetu
	17606	w25n 17652		:050 00000 17656 21	4+	A = [B2];	słowo do wyprowadzenia
	17675	WY28n 17653		:021 17574 17661 70	4+	[SumaK] = B7;	zapisanie sumy kontrolnej
	17764	**ED00n 17654**		**:126 infor 17654 00**		**We1;**	
	WY28p 17653		17655	:021 17574 17607 70	4+	[SumaK] = B7;	zapisanie sumy kontrolnej
	w26p 17742	w26n 17656		:646 17676 17676 20		SkLC B2++	adres na następne słowo
	17631	**MAW0n 17657**		**:266 00000 17704 00**		**XWe2;**	**albo Stop;**
	17720		17660	:000 00000 00000 00			
	w29p 17607	w29n 17661		:052 17573 17624 30		B3 = A=[Robo3]	= :111 33333 22222 11
	w27p 17676		17662	:422 00000 17623 70		B7 = A + B7;	dodanie słowa do sumy kontr.
	c15p 17765		17663	:346 17766 17760 00		SkNd;	jeśli nie nadmiar to szukać dalej
	ED00p 17654		**17664**	**:126 infor 17654 00**		**We1;**	
	17743	c15n 17665		:346 17673 17621 00		SkNd;	jeśli nie nadmiar to szukać dalej
	MAW1p 17632		17666	:016 00037 17635 00		A = A <<< 31;	
	17721		17667	:000 00000 00000 00			
	w09p 17610		17670	:402 00000 17633 30		B3 = A; = nnnnn	B3 = nnnnn;
	w32p 17677		17671	:416 00015 17745 00		A = A >>< 13;	= :333 22222 11111 33
	c16p 17766		17672	:330 17574 17656 00		A - SumaK;	jest pilot: czy to pilot sumy
	17655	c16n 17673		:330 17574 17705 00		A - SumaK;	jest pilot: czy to pilot sumy
	17744	CZ01n 17674		:726 00000 17700 00		Stop;	
	w10p 17633	w10n 17675		:060 00000 17767 70		A = B7;	= :000 ppppp 00000 00
	17722	w27n 17676		:422 00000 17605 70		B7 = A + B7;	dodanie słowa do sumy kontr.
	17611	w32n 17677		:416 00015 17703 00		A = A >>< 13;	= :333 22222 11111 33
	CZ000 17700	**CZ000 17700**		**:032 17674 17770 60**		**B6 = CZ01;**	**wywołanie podprogramu 17770**
niezero	w11p 17767		17701	:012 17574 17617 70	4+	B7 = [SumaK];	= suma kontrolna
	c17p 17656		17702	:046 17771 17612 00		SkZ;	gdy pilot sumy, to do wczytania sumy
	w33p 17745	w33n 17703		:402 00000 17724 20		B2 = A;	przywrócenie B2
	17634	MAW1n 17704		:016 00037 17713 00		A = A <<< 31;	
	17723	c17n 17705		:046 17720 17712 00		SkZ;	gdy pilot sumy, to do wczytania sumy
	c18p 17612		17706	:402 00000 17613 10		B1 = A;	zwykły pilot: B1 = adres wprowadzania
zero	**17701**		**17707**	**:000 00000 00000 00**		**zero stałoprzecinkowe**	
	CZ10p 17770		**17710**	**:021 17576 17754 10**	**4+**	**[Robo6] = B1;**	**przechowanie B1**
	MAW0p 17657		**17711**	**:266 00000 17632 00**		**XWe2;**	**albo Stop;**
	17746	c18n 17712		:402 00000 17716 10		B1 = A;	zwykły pilot: B1 = adres wprowadzania
	MAW2p 17635	MAW2n 17713		:566 00000 17657 00		XWy5;	
	w34p 17724		17714	:012 17575 17577 70	4+	B7 = [Pilot];	przywrócenie B7; RETURN
	CZ19p 17613		17715	:072 00007 17747 20		A = B2 = 7;	śmieć w A zniknie
	17702	CZ19n 17716		:072 00007 17722 20		A = B2 = 7;	śmieć w A zniknie
	CZ28p 17771		17717	:072 00007 17637 20		A = B2 = 7;	śmieć w A zniknie
	17660	CZ28n 17720		:072 00007 17732 20		A = B2 = 7;	śmieć w A zniknie
	CZ20p 17747		17721	:216 00005 17772 00		A = A << 5, ustawienie Nd;	czytanie słowa:
	17636	CZ20n 17722		:216 00005 17727 00		A = A << 5, ustawienie Nd;	czytanie słowa:
	WY36p 17725		17723	:050 17750 17663 00	4+	A = :000 SumaK 00001 00	adres i dł.bloku sumy
	17614	w34n 17724		:012 17575 17577 70	4+	B7 = [Pilot];	przywrócenie B7; RETURN
	17703	WY36n 17725		:050 17730 17745 00	4+	A = :000 SumaK 00001 00	adres i dł.bloku sumy

Robo2p	c21p 17772	17726	:026 00000 17727 00	4+	We0, Nd bez zmiany, ustawienie Z; wczyt.kwint.
	17661	c21n 17727	:026 00000 17742 00	4+	We0, Nd bez zmiany, ustawienie Z; wczyt.kwint.
	WY39p 17750	WY39n 17730	:000 17574 00001 00		adres i długość bloku sumy kontrolnej
	CZ29p 17637	17731	:216 00005 17773 00		A = A << 5; czytanie sumy
	17726	CZ29n 17732	:216 00005 17736 00		A = A << 5; czytanie sumy
	w20p 17615	17733	:016 00001 17751 00		A = A <<< 1; zrobienie miejsca na bit Omega
	17704	w20n 17734	:016 00001 17740 00		A = A <<< 1; zrobienie miejsca na bit Omega
Robo3p	c30p 17773	17735	:026 00000 17620 00	4+	We0; wczytanie kwintetu
	17662	c30n 17736	:026 00000 17761 00	4+	We0; wczytanie kwintetu
	w21p 17751	17737	:426 00000 17705 00		$A_{38} = A_{38} + \Omega$; i nieszk. powielanie bitu Omega
	17640	w21n 17740	:426 00000 17744 00		$A_{38} = A_{38} + \Omega$; i nieszk. powielanie bitu Omega
	c22p 17727	17741	:546 17747 17664 20		SkLC B2--, Nd bez zmiany, Z bez zmiany
	17616	c22n 17742	:546 17722 17757 20		SkLC B2--, Nd bez zmiany, Z bez zmiany
	w22p 17705	17743	:016 00004 17641 00		A = A <<< 4;
SumaKp	17774	w22n 17744	:016 00004 17750 00		A = A <<< 4;
	w37p 17663	w37n 17745	:032 17574 17753 70		B7= :000 SumaK 00000 00 pilot sumy
	w15p 17752	17746	:401 17575 17623 00	4+	[Pilot] = A; = :000 ppppp 00000 00
	w23p 17641	17747	:546 17651 17627 10		SkLC B1--
	17730	w23n 17750	:546 17651 17641 10		SkLC B1--
	w12p 17617	w12n 17751	:310 17575 17604 70	4+	A - [Pilot]; jeśli ppppp == qqqqq
	17706	w15n 17752	:401 17575 17605 00	4+	[Pilot] = A; = :000 ppppp 00000 00
Pilotp	17775	w38n 17753	:746 17577 17576 00	4+	wywołanie ŚladB; wyprowadz. sumy i RETURN
	c23p 17664	17754	:346 17665 17735 00		SkNd; wczytano 8 kwint., gdy nadmiar zapisać
	w38p 17753	17755	:746 17577 17576 00	4+	wywołanie ŚladB; wyprowadz. sumy i RETURN
	c33p 17642	17756	:012 17576 17643 10	4+	B1 = [Robo6]; przywrócenie B1
	17731	c23n 17757	:346 17766 17615 00		SkNd; wczytano 8 kwint., gdy nadmiar zapisać
	c31p 17620	17760	:546 17637 17741 20		SkLC B2--
niezero	17707	c31n 17761	:546 17732 17650 20		SkLC B2--
ŚladSp	w07p 17776	17762	:052 17572 17645 20	4+	B2 = A=[Robo2] = :000 ddddd nnnnn 00
	CZ25p 17665	17763	:401 00000 17671 11		[B1] = A; jest słowo, zapisanie go w pamięci
	c11p 17754	17764	:021 17577 17757 20	4+	[Robo7] = B2; przechowanie B2
	c34p 17643	17765	:012 17577 17712 20	4+	B2 = [Robo7]; przywrócenie B2
	17732	CZ25n 17766	:401 00000 17635 11		[B1] = A; jest słowo, zapisanie go w pamięci
	17621	w11n 17767	:012 17574 17751 70	4+	B7 = [SumaK]; = suma kontrolna
	17710	**CZ10n 17770**	**:021 17576 17772 10**	**4+**	**[Robo6] = B1; przechowanie B1**
ŚladBp	**Zerop 17777**	**17771**	**:000 00000 00004 00**		**zero zmiennoprzecinkowe**
Robo2n	17666	c11n 17772	:021 17577 17612 20	4+	[Robo7] = B2; przechowanie B2
Robo3n	17755	c33n 17773	:012 17576 17774 10	4+	B1 = [Robo6]; przywrócenie B1
SumaKn	17644	c34n 17774	:012 17577 17606 20	4+	B2 = [Robo7]; przywrócenie B2
Pilotn	w02p 17733	w02n 17775	:060 00000 17602 10		A = B1; = :000 11111 00000 00
ŚladSn	17622	w07n 17776	:052 17572 17603 20	4+	B2 = A=[Robo2] = :000 ddddd nnnnn 00
ŚladBn	**17711**	**Zeron 17777**	**:000 00000 00004 00**		**zero zmiennoprzecinkowe**

UWAGA:

Zawartość komórek o adresach $17654_{(8)}$ i $17657_{(8)}$ – zarówno normalnych jak i przeplotowych – jest różna w różnych systemach komputerowych.