Zmienne w układzie sterowania CNC SINUMERIK Operate

The variables in the CNC control system SINUMERIK Operate

BOGUSŁAW PYTLAK*

DOI: https://doi.org/10.17814/mechanik.2017.3.44

Przedstawiono typy zmiennych występujące w układzie sterowania SINUMERIK Operate. Opisano zmienne systemowe **i zmienne użytkownika: R-parametry, globalne R-parametry, zmienne GUD, zmienne PUD i zmienne LUD, wraz z przykładami programowania. Omówiono programowanie pośrednie z wykorzystaniem zmiennych. Opracowano program do frezowania gwintu, na którego przykładzie pokazano różnice wynikające ze stosowania różnych typów zmiennych oraz zalety tych rozwiązań.**
SŁOWA KLUCZOWE: typy zmiennych, układ sterowania CNC, programowanie pośrednie, programowanie parametryczne

The types of variables occurring in the control system SINUMERIK Operate are presented. The system variables and user variables: R-parameters, global R parameters, GUD variables, PUD variables and LUD variables, with programming examples are described. The indirect programming with using of variables is discussed. The example of program for thread milling is prepared, which shows the differences of using of different variable types and advantages this solutions.
KEYWORDS: variable types, CNC control system, indirect programming, parametric programming

Programowanie parametryczne w układzie sterowania SINUMERIK opiera się na wykorzystaniu zmiennych, które w połączeniu z funkcjami obliczeniowymi i strukturami kontrolnymi zapewniają wysoką elastyczność programów, podprogramów i cykli. Celem artykułu jest przybliżenie użytkownikom układu sterowania SINUMERIK poszczególnych typów zmiennych oraz sposobów ich wykorzystania w programach obróbki. Zmienne w układzie sterowania SINUMERIK można podzielić na: systemowe i użytkownika [1].

Zmienne systemowe

Zmienne systemowe mają zdefiniowane znaczenie. Korzysta z nich oprogramowanie systemowe, można je też odczytywać i zapisywać w programach obróbki użytkownika. Pomimo wstępnie zdefiniowanego znaczenia zmiennych systemowych istnieje możliwość zmiany ich właściwości przez tzw. redefinicję. Zmienne systemowe pozwalają parametryzować układ sterowania i zapewniają dostęp do aktualnego stanu sterowania, maszyny i procesu obróbki.

* Dr inż. Bogusław Pytlak (boguslaw.pytlak.ext@siemens.com) – Siemens Sp. z o.o.

Zmienne te można podzielić na zmienne przebiegu wyprzedzającego i zmienne przebiegu głównego [1]. Zmienne przebiegu wyprzedzającego są czytane i zapisywane w momencie czytania (interpretacji) bloku programu, w którym zostały zaprogramowane. Nie powodują zatrzymania przebiegu wyprzedzającego. Z kolei zmienne przebiegu głównego są czytane i zapisywane w momencie wykonywania bloku programu, w którym zostały zaprogramowane. Do zmiennych tych zalicza się zmienne programowane w akcjach synchronicznych, zmienne zatrzymujące przebieg wyprzedzający oraz zmienne, których wartość jest określana w przebiegu wyprzedzającym, ale zapisywana dopiero w przebiegu głównym.

Standardowe oznaczenie zmiennych systemowych składa się ze znaku `$` + pierwsza litera (rodzaj danych) + druga litera (obszar obowiązywania). Zmienne systemowe przebiegu wyprzedzającego są następujące [1]: `$M` – dane maszynowe, `$S` – dane ustawcze, `$T` – dane narzędzi, `$P` – wartości programowane, `$C` – zmienne cykli ISO, `$O` – dane opcji, `R` – parametry obliczeniowe. Zmienne systemowe przebiegu głównego to [1]: `$$M` – dane maszynowe, `$$S` – dane ustawcze, `$A` – aktualne dane przebiegu głównego, `$V` – dane Serwo, `$R` – parametry obliczeniowe. Zmienne przebiegu głównego odróżnia dodatkowy znak `$`. Jest on zarezerwowany do zmiennych systemowych i nie wolno go używać w zmiennych użytkownika.

Wyróżnia się obszary obowiązywania zmiennych: `N` – zmienne globalne (NCK), `C` – zmienne kanałowe (Channel), `A` – zmienne osiowe (Axis). Szczegółowy wykaz i opis zmiennych systemowych znajdują się w dokumentacji układu sterowania SINUMERIK [2].

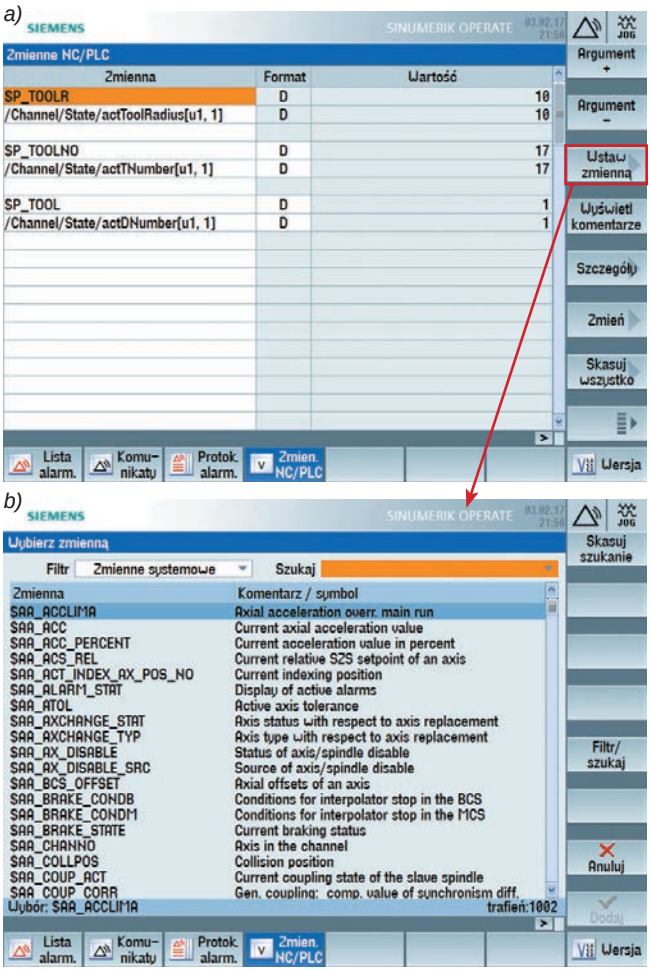
Do wyświetlania zmiennych systemowych przewidziano obszar 

→ 

(rys. 1*a*). Zmienną można wyświetlić w danym wierszu, wpisując bezpośrednio jej nazwę. Można także użyć przycisku 

, który otwiera okno wyszukiwania i wyboru zmiennych (rys. 1*b*).

W układzie sterowania SINUMERIK większość zmiennych systemowych występuje pod dwoma nazwami. Pierwsza z nich, np. `$P_TOOLR`, oznacza promień aktywnego narzędzia i można jej używać w programach. Natomiast druga, np. `/Channel/State/actToolRadius[u1,1]`, oznacza tę samą zmienną i jest przydatna w oprogramowaniu systemowym, np. w plikach COM definiujących okna dialogowe. Oznaczenie `u1` określa numer kanału, dla kanału numer 2 będzie to `u2` itd.



Rys. 1. Okna: *a*) obszaru zmiennych NC/PLC, *b*) wyszukiwania i wyboru zmiennych NC/PLC

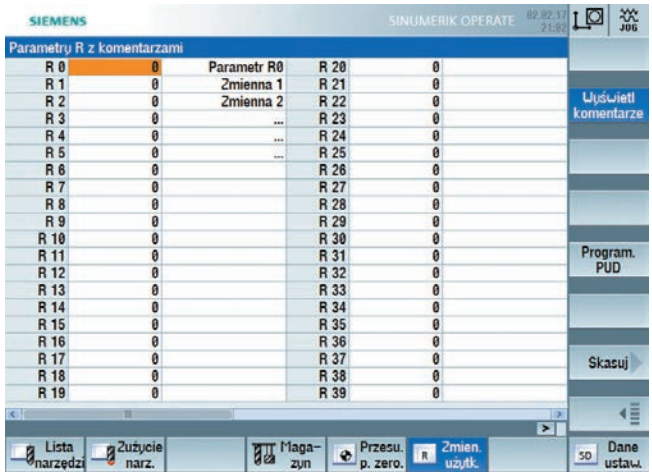
Zmienne użytkownika

Zmienne użytkownika obejmują z kolei zmienne wstępnie zdefiniowane w systemie oraz zmienne definiowane przez samego użytkownika. Typ zmiennych użytkownika wstępnie zdefiniowanych jest z góry określony, natomiast ich liczbę określa się w odpowiednich danych maszynowych. Do zmiennych tych zaliczyć można: R-parametry, globalne R-parametry RG, zmienne typu Link. Zmienne definiowane przez użytkownika są tworzone w systemie na stałe lub tylko na czas wykonywania programu – użytkownik ma nad nimi pełną kontrolę. Zmienne te można podzielić w zależności od zakresu obowiązywania na: globalne GUD (Global User Data), programowe PUD (Program User Data) i lokalne LUD (Local User Data).

■ **R-parametry** należą do najbardziej znanych predefiniowanych zmiennych użytkownika. Występowały już w pierwszych wersjach układu sterowania SINUMERIK i przez długi czas były jedynym sposobem na parametryzację programów obróbki oraz przekazanie wartości parametrów do cykli. Obecnie przy dostępności innych rodzajów zmiennych użytkownika znaczenie R-parametrów jest mniejsze, choć ciągle znajdują zastosowanie. R-paramery są tablicą zmiennych typu Real.

Dla lepszej czytelności R-parametrów w programie obróbki często umieszcza się dodatkowy opis znaczenia danego R-parametru, np. w formie komentarza w programie obróbki: `R10=200; wspolrzedna w osi X`. W wersji oprogramowania HMI Operate v.4.7 można obok kolumny R-parametrów wyświetlić dodatkową kolumnę z ich opisem (rys. 2).

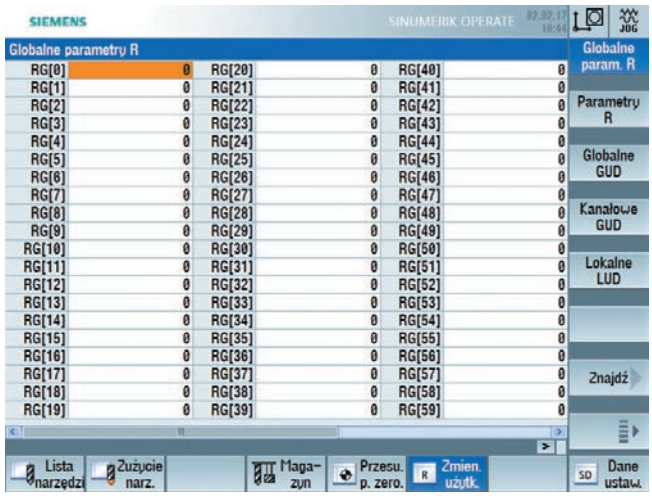
Formalnie odwołanie się do danego R-parametru powinno być odwołaniem do zmiennej tablicowej, np. `R[10]=...`, jednak ze względów historycznych jest dopuszczalny zapis uproszczony, np. `R10=...` R-parametry są zmiennymi kanałowymi, co oznacza, że w każdym kanale jest identyczny zestaw R-parametrów.



Rys. 2. Okno R-parametrów z dodatkową kolumną komentarzy

■ **Globalne R-parametry RG.** Do wymiany informacji pomiędzy kanałami można użyć globalnych R-parametrów RG dostępnych w wersji oprogramowania HMI Operate v.4.7 (rys. 3) lub globalnych zmiennych użytkownika GUD. Podobne globalne R-parametry istniały w układzie sterowania SINUMERIK 840C, gdzie R-parametry powyżej `R700` były traktowane jako globalne. Do globalnych R-parametrów RG należy się odwoływać tak jak do zmiennej tablicowej, np. `RG[10]=...`

■ **Zmienne typu Link** są wykorzystywane do cyklicznej wymiany informacji pomiędzy różnymi NCU połączonymi w sieć za pomocą funkcji NCU-Link. Zmienne typu Link mają charakter globalny – mogą być zapisywane i odczytywane przez programy obróbki na wszystkich połączonych w sieć NCU. W przypadku tylko jednego NCU zmienne typu Link można wykorzystać jako dodatkowe globalne zmienne użytkownika. Wyróżnia się następujące zmienne typu Link: `$A_DLB[<i>]` – bajt, `$A_DLW[<i>]` – słowo, `$A_DLD[<i>]` – podwójne słowo, `$A_DLR[<i>]` – Real, gdzie `<i>` oznacza indeks zmiennej. Indeks ten zmienia się od 0, co 1 dla bajtu, co 2 dla słowa, co 4 dla podwójnego słowa, co 8 dla Real.



Rys. 3. Okno globalnych R-parametrów RG

■ **Globalne zmienne użytkownika GUD** są zmiennymi, do których mają dostęp wszystkie programy. To właśnie one najczęściej wypierają popularne R-parametry. Zmienne GUD mogą być różnego typu i mieć dowolną nazwę; znajdują się one w nieulotnej pamięci NC. Te cechy decydują o ich dużej użyteczności. W układzie sterowania SINUMERIK korzystają z nich bardzo często producenci maszyn, dla których przewidziany jest plik definicji zmiennych MGUD.DEF, z kolei dla użytkowników przeznaczono plik definicji zmiennych UGUD.DEF. Zmienne GUD definiuje się w następujący sposób:

DEF <zakres> <stop_przebiegu_wyprzedzajacego> <prawa _do stepu> <klasa_danych> <typ> <jednostka_fizyczna> <wartosci_graniczne> <nazwa>[<wartosc_1>, <wartosc_2>, <wartosc_3>] = <wartosc_inicjalizacyjna>

gdzie:

<zakres> – zakres obowiązywania zmiennej: globalna NCK, kanałowa CHAN;
<stop_przebiegu_wyprzedzajacego> – moment zatrzymania przebiegu wyprzedzającego: zapis, odczyt, zapis + odczyt zmiennej;
<prawa_dostepu> – poziom dostępu do zmiennej: dla programu NC i panelu sterowania;
<klasa_danych> – określenie, do jakiej klasy danych należy dana zmienna (tylko dla 828D sl);
<typ> – typ zmiennej;
<jednostka_fizyczna> – jednostka, w jakiej będzie wyrażana dana zmienna (tylko dla typu REAL i INT);
<wartosci_graniczne> – ograniczenia dolne LLI i górne ULI wartości zmiennej (tylko dla typu REAL, INT i CHAR);
<nazwa> – nazwa zmiennej (nie można używać jako nazw poleceń języka programowania SINUMERIK i zmiennych już zdefiniowanych);
[<wartosc_1>, <wartosc_2>, <wartosc_3>] – wymiary tablicy dla zmiennej tablicowej;
<wartosc_inicjalizacyjna> – wartość, jaką zmienna przyjmuje w momencie inicjalizacji.

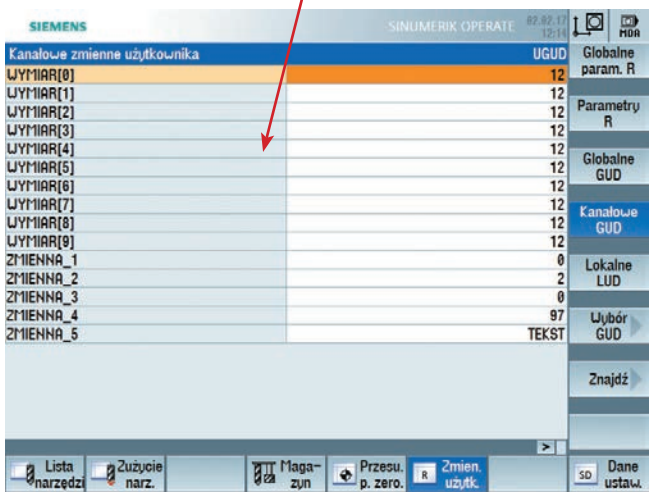
W układzie sterowania SINUMERIK można definiować zmienne następujących typów: INT (Integer), REAL, BOOL, CHAR, STRING[<max_dl.>], AXIS, FRAME. Większość tych typów występuje w językach programowania komputerów, jednak wyjaśnienia wymagają typy AXIS i FRAME, które pojawiają się tylko w języku programowania SINUMERIK.

AXIS jest to typ osiowy, co oznacza, że do zmiennej tego typu można przypisać dowolną oś maszyny. Z kolei do zmiennej FRAME można przypisać przekształcenia układu współrzędnych (TRANS, ROT, MIRROR, SCALE) lub ich kombinację.

Przy okazji omawiania typów zmiennych należy wspomnieć o wielkiej zalecie języka programowania SINUMERIK, jaką jest automatyczna konwersja zmiennych (o ile taka jest możliwa). Obecnie istnieje możliwość programowania zmiennych tablicowych aż trójwymiarowych, za wyjątkiem zmiennych typu STRING, które mogą być dwuwymiarowe (trzeci wymiar jest zawarty w samym typie STRING, który jest tablicą zmiennych typu CHAR). Przykładowa składnia pliku definicji globalnych zmiennych UGUD.DEF jest następująca:

```
DEF CHAN SYNJ APRP 1 APWP 1 REAL PHU 1 LLI 0 ULI 500
WYMIAR[10] = REP(12) ; pełna definicja zmiennej GUD
;CHAN – zmienna kanałowa,
;SYNJ – zatrzymanie przebiegu wyprzedzajacego przy zapisie,
;APRP 1 – odczyt z programu NC na poziomie producenta maszyny,
;APWP 1 – zapis z programu NC na poziomie producenta maszyny,
;REAL – typ rzeczywisty,
;PHU 1 – jednostka długosc, mm lub cal,
;LLI 0 – dolne ograniczenie rowne 0,
;ULI 500 – gorne ograniczenie rowne 500,
;WYMIAR[10] – zmienna tablicowa o 10 elementach,
;REP(12) – inicjalizacja wszystkich elementow tablicy z wartoscia 12,

DEF CHAN REAL ZMIENNA_1 ; definicja uproszczona
DEF CHAN INT ZMIENNA_2=2
DEF CHAN BOOL ZMIENNA_3=FALSE
DEF CHAN CHAR ZMIENNA_4="a" ;lub 97 wg kodu ASCII
DEF CHAN STRING[10] ZMIENNA_5="TEKST"
DEF CHAN AXIS ZMIENNA_6=(X)
DEF CHAN FRAME ZMIENNA_7
M30
```



Rys. 4. Okno kanałowych zmiennych użytkownika GUD

Jak widać na rys. 4, zmienne typu AXIS i FRAME nie są wyświetlane w oknie zmiennych użytkownika, ponadto zmiennej typu FRAME nie można przypisać wartości inicjalizacyjnej. Można to zrobić później, np. w programie obróbki, np. ZMIENNA_7=CTRANS(X,100,Y,200,Z,300):CROT(Z,45), co oznacza przesunięcie i obrót układu współrzędnych. Kolejność definicji zmiennych w pliku DEF decyduje o kolejności ich wyświetlania w oknie zmiennych GUD (rys. 4).

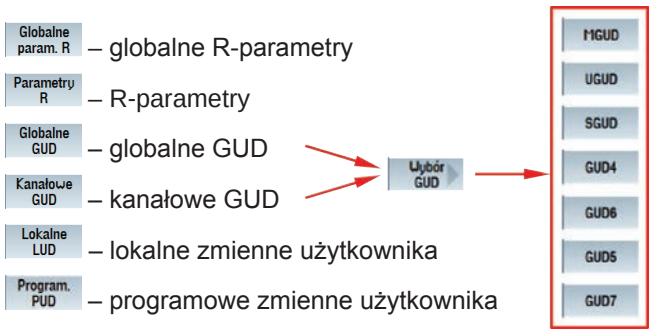
■ **Lokalne zmienne użytkownika LUD** są definiowane w obrębie danego programu (podprogramu) obróbki. Powstają w momencie wykonywania danego programu i są kasowane po jego zakończeniu. Definicji zmiennych lokalnych dokonuje się na samym początku programu obróbki za pomocą polecenia DEF (wcześniej może się znajdować tylko polecenie EXTERN). Składnia polecenia DEF jest następująca:

DEF <typ_zmiennej> <jednostka_fizyczna> <wartosci_graniczne> <nazwa> [<wartosc_1>, <wartosc_2>, <wartosc_3>] = <wartosc_inicjalizacyjna>

W obrębie jednej definicji DEF zdefiniowanych może być kilka zmiennych tego samego typu (należy tylko oddzielić je przecinkami). Każdy kolejny typ zmiennych musi być zdefiniowany w nowym bloku za pomocą ponownego polecenia DEF. Standardowo zmienne LUD zdefiniowane w programie głównym nie są widoczne dla podprogramu wywołanego z programu głównego.

Aby stały się widoczne, należy włączyć obsługę programowych zmiennych użytkownika PUD. W tym celu ustawia się daną maszynową $MN_LUD_EXTENDED_SCOPE=1. Po czym pojawia się dodatkowy przycisk ze zmiennymi typu PUD.

■ **Programowe zmienne użytkownika PUD** są zdefiniowane w programie głównym i dostęp mają do nich wszystkie podprogramy wywoływane z tego programu. Pozwala to na przekazywanie informacji pomiędzy programem głównym i wywołanym podprogramem. Podglądu i modyfikacji zmiennych użytkownika dokonuje się w obszarze: Parametry → Zmien. użytk. . Przy czym kolejne przyciski odpowiadają poszczególnym zmiennym:



■ **Redefinicja zmiennych.** Po zdefiniowaniu zmiennych użytkownika, podobnie jak w przypadku zmiennych systemowych, można dokonać zmiany ich atrybutów za pomocą polecenia redefinicji REDEF. W jednym poleceniu REDEF można zmienić tylko jeden atrybut zmiennej:

```
REDEF <nazwa> <atrybut>
```

W tym samym poleceniu można określić moment ponownej inicjalizacji zmiennej (tj. ustawienia zmiennej na wartość inicjalizacyjną: INIPO – po Power On, INIRE – po Reset, INICF – po poleceniu NEWCONFIG, PRLOC – inicjalizacja zmiennej, a dokładnie programowalnej danej ustawczej, następuje po resecie tylko w sytuacji, gdy została ona zmieniona z poziomu programu obróbki).

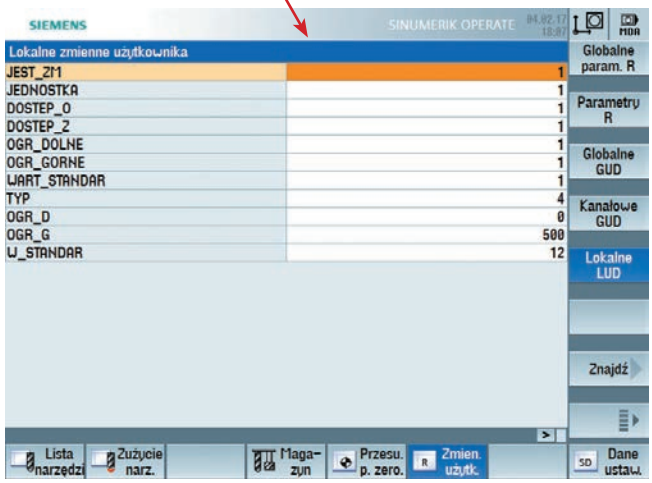
Przykładowo: dana ustawcza została zainicjowana wartością $SC_THREAD_START_ANGLE=0, co oznacza kąt startu 0° dla nacinania gwintu. Zaprogramowanie polecenia np. SF=45 zmienia wartość tej danej na 45°. Dzięki określeniu momentu ponownej inicjalizacji za pomocą PRLOC po resecie do danej ustawczej $SC_THREAD_START_ANGLE znów jest przypisana wartość 0.

■ **Uzyskiwanie informacji o zmiennych.** Podczas definiowania kolejnej zmiennej może się okazać, że taka zmienna już jest w układzie sterowania. Zaleca się, aby nazwa zmiennych użytkownika zaczynała się od znaku podkreślenia, jednak praktyka wskazuje, że lepiej nie stosować nazw typu podkreślenie + nazwa osi, np. _A, gdyż są one wewnętrznie używane w cyklach obróbki i czasem z tego powodu występują problemy z dublowaniem się definicji.

Aby uniknąć problemu podwójnej definicji tej samej zmiennej lub np. próby zapisu czy odczytu nieistniejącej zmiennej, można użyć polecenia ISVAR(<nazwa_zmiennej>), które sprawdza, czy zmienna wprowadzona w parametrze <nazwa_zmiennej> istnieje. Wynik tego sprawdzenia najlepiej zapisać do innej zmiennej typu BOOL. Oprócz sprawdzenia istnienia danej zmiennej można w układzie sterowania SINUMERIK uzyskać znacznie więcej informacji o niej: o jednostce (GETVARPHU), prawie dostępu (GETVARAP), ograniczeniach (GETVARLIM), wartości standardowej (GETVARDFT) oraz typie (GETVARTYP). Polecenia te, podobnie jak polecenie ISVAR, zwracają określone wartości liczbowe, które najlepiej zapisać do zmiennych typu INT i REAL (rys. 5).

```
DEF BOOL JEST_ZM
DEF INT JEDNOSTKA,DOSTEP_O,DOSTEP_Z,OGR_DOLNE,OGR_GOR
NE, WART_STANDAR,TYP
DEF REAL OGR_D,OGR_G,W_STANDAR

JEST_ZM=ISVAR("WYMIAR") ;sprawdzenie istnienia zmiennej
JEDNOSTKA=GETVARPHU("WYMIAR") ;odczyt jednostki
DOSTEP_O=GETVARAP("WYMIAR","RP") ;odczyt prawa odczytu
DOSTEP_Z=GETVARAP("WYMIAR","WP") ;odczyt prawa zapisu
OGR_DOLNE=GETVARLIM("WYMIAR","L",OGR_D) ;odczyt dolnego
ograniczenia do zmiennej OGR_D
OGR_GORNE=GETVARLIM("WYMIAR","U",OGR_G) ;odczyt gornego
ograniczenia do zmiennej OGR_G
WART_STANDAR=GETVARDFT("WYMIAR",W_STANDAR,3) ;odczyt
wartosci elementu tablicy o indeksie 3 do zmiennej W_STANDAR
TYP=GETVARTYP("WYMIAR") ;odczyt typu zmiennej
```



Rys. 5. Okno zmiennych lokalnych użytkownika LUD z odczytanymi informacjami o zmiennej WYMIAR

■ **Programowanie pośrednie.** Zmienne użytkownika są często wykorzystywane w tzw. programowaniu pośrednim. Polega ono na użyciu zmiennych w programowaniu adresów rozszerzonych. Przykładowo: aby zaprogramować i włączyć obroty wrzeciona, którego numer określa zmienna NR_WRZECIONA, wystarczy wprowadzić następujący kod programu:

```
DEF INT NR_WRZECIONA=1
S[NR_WRZECIONA]=2000 M[NR_WRZECIONA]=3
```

W sposób pośredni można wywoływać także podprogramy, np. za pomocą polecenia CALL, gdzie ścieżkę dostępu i nazwę programu (podprogramu) można podać za pomocą zmiennej typu STRING, do której jest np. dołączona zmienna typu INT z numerem programu:

```
DEF STRING[100] NAZWA
DEF INT NR_PROG=1
NAZWA="/_N_WKS_DIR/_N_PRZYKLAD"<<NR_PROG<<
"_WPD/_N_PRZYKLAD"<<NR_PROG<<"_MPF"
CALL NAZWA
```

W tym przykładzie wykorzystano operator łączenia łańcuchów znaków <<, który pozwala połączyć dwa łańcuchy

znaków typu STRING i dołączyć do tego zmienną typu INT. Konwersja typów odbywa się w sposób automatyczny, o czym była mowa wcześniej.

Inną ciekawą możliwością jest programowanie funkcji G w sposób pośredni. W tym celu wykorzystuje się informacje o tym, do jakiej grupy funkcja G jest przyporządkowana oraz jaką pozycję zajmuje w danej grupie. Dokładny opis grup funkcji G i pozycji zajmowanych w tych grupach można znaleźć w dokumentacji [3] w rozdziale 17.4. Przykładowo w tablicy przedstawiono funkcje G w grupie numer 8.

TABLICA. Grupa funkcji G numer 8: Ustawiane przesunięcie punktu zerowego

Polecenie G	Nr	Znaczenie
G500	1	Wyłączenie ustawianego przesunięcia punktu zerowego (G54 ... G57, G505 ... G599)
G54	2	1 ustawiane przesunięcie punktu zerowego
G55	3	2 ustawiane przesunięcie punktu zerowego
G56	4	3 ustawiane przesunięcie punktu zerowego
G57	5	4 ustawiane przesunięcie punktu zerowego
G505	6	5 ustawiane przesunięcie punktu zerowego
...	...	...
G599	100	99 ustawiane przesunięcie punktu zerowego

Aby wywołać przesunięcie punktu zerowego G54 i kolejne, należy zaprogramować:

```
DEF INT NR_GRUPY_G,NR_PPZ
NR_GRUPY_G=8
NR_PPZ=2
G[NR_GRUPY_G]=NR_PPZ ;wywołanie G54
STOPRE ;zatrzymanie przebiegu wyprzedzajacego
NR_PPZ=NR_PPZ+1 ;zwiekszenie NR_PPZ o 1
G[NR_GRUPY_G]=NR_PPZ ;wywołanie G55
STOPRE ;zatrzymanie przebiegu wyprzedzajacego
NR_PPZ=NR_PPZ+1 ;zwiekszenie NR_PPZ o 1
G[NR_GRUPY_G]=NR_PPZ ;wywołanie G56
;...
```

Istnieje także możliwość pośredniego programowania kodu NC za pomocą polecenia EXECSTRING G(<zmienna_string>). Polecenie to zamienia łańcuch znaków <zmienna_string> na kod NC, który jest wykonywany przez sterowanie, np.:

```
EXECSTRING("S1000 M3")
EXECSTRING("G4 F10")
EXECSTRING("M5")
```

■ **Programowanie parametryczne.** W układzie sterowania SINUMERIK w programach obróbki można korzystać z różnych rodzajów zmiennych. Aby przedstawić różnice pomiędzy poszczególnymi programami, opracowano prosty przykład frezowania pełnej spirali wewnętrznego gwintu M52×5 w półfabrykacie o grubości 50 mm. Współrzędne środka gwintu wynoszą X50 Y50. W pierwszym wariancie program jest tzw. programem stałym, który nie korzysta ze zmiennych (poza promieniem aktywnego narzędzia $P_TOOLR). W programie tym w pierwszym bloku G3 następuje dojazd do ścianki gwintu po półokręgu. W drugim bloku G3 jest frezowana pełna spirala gwintu. W ostatnim bloku G3 następuje powrót do środka gwintu po półokręgu:

```
T="THREAD CUTTER"
M6
G17 G54 G94
WORKPIECE("","","BOX",112,0,-50,-80,0,0,100,100)
G90 G0 X50 Y50 Z5 D1 F100 S2000 M3
G3 X=IC(26-$P_TOOLR) Y50 Z=IC(-5/4) CR=(26-$P_TOOLR)/2
G3 X=IC(0) Y=IC(0) Z=IC(-60) I=AC(50) J=AC(50) TURN=12
G3 X50 Y50 Z=IC(-5/4) CR=(26-$P_TOOLR)/2
G0 Z5
M2
```

W drugim wariancie dokonano parametryzacji programu za pomocą R-parametrów. W przykładzie tym należy zwrócić uwagę na parametryzację półfabrykatu widocznego podczas symulacji, tzw. WORKPIECE, którego wymiary zostały również sparametryzowane. Podobną parametryzację można wprowadzać w innych cyklach w kodzie G, a także ShopMill i ShopTurn. Jak widać, stosowanie R-parametrów wymaga dodatkowego opisu ich znaczenia w formie komentarzy:

```
R0=50 ;wspolrzedna srodka w osi X
R1=50 ;wspolrzedna srodka w osi Y
R2=5 ;wysokosc poczatkowa w osi Z
R3=52 ;srednica frezowanego gwintu
R4=5 ;skok gwintu
R5=12 ;liczba zwoi
T="THREAD CUTTER"
M6
G17 G54 G94
WORKPIECE("","","BOX",112,R2-5,-R4*(R5-2),-80,R0-50,R1-
-50,R0+50, R1+50)
G90 G0 X=R0 Y=R1 Z=R2 D1 F100 S2000 M3
G3 X=IC(R3/2-$P_TOOLR) Y=R1 Z=IC(-R4/4) CR=(R3/2-$P_TO
OLR)/2
G3 X=IC(0) Y=IC(0) Z=IC(-R4*R5) I=AC(R0) J=AC(R1) TURN=R5
G3 X=R0 Y=R1 Z=IC(-R4/4) CR=(R3/2-$P_TOOLR)/2
G0 Z=R2
STOPRE
R0=0 R1=0 R2=0 R3=0 R4=0 R5=0
M2
```

W kolejnym przykładzie zastosowano do parametryzacji lokalne zmienne użytkownika LUD definiowane na początku programu (w ich miejsce można użyć np. zmiennych GUD zdefiniowanych w pliku UGUD.DEF). Przypadek ten jest znacznie czytelniejszy, bo już po samej nazwie zmiennej można się zorientować, co ona oznacza:

```
DEF REAL WSPOL_X,WSPOL_Y,WSPOL_Z,SREDNICA,SKOK,
LICZBA_ ZWOI
WSPOL_X=50 ;wspolrzedna srodka w osi X
WSPOL_Y=50 ;wspolrzedna srodka w osi Y
WSPOL_Z=5 ;wysokosc poczatkowa w osi Z
SREDNICA=52 ;srednica frezowanego gwintu
SKOK=5 ;skok gwintu
LICZBA_ZWOI=12 ;liczba zwoi
T="THREAD CUTTER"
M6
G17 G54 G94
WORKPIECE("","","BOX",112,WSPOL_Z-5,-SKOK*(LICZBA_ZWOI-2),-
80, WSPOL_X-50,WSPOL_Y-50,WSPOL_X+50,WSPOL_Y+50)
G90 G0 X=WSPOL_X Y=WSPOL_Y Z=WSPOL_Z D1 F100 S2000 M3
G3 X=IC(SREDNICA/2-$P_TOOLR) Y=WSPOL_Y Z=IC(-SKOK/4)
CR=(SREDNICA/2-$P_TOOLR)/2
G3 X=IC(0) Y=IC(0) Z=IC(-SKOK*LICZBA_ZWOI) I=AC(WSPOL_X)
J=AC(WSPOL_Y) TURN=LICZBA_ZWOI
G3 X=WSPOL_X Y=WSPOL_Y Z=IC(-SKOK/4) CR=(SREDNICA/2-
-$P_TOOLR)/2
G0 Z=WSPOL_Z
M2
```

W czwartym przykładzie utworzono podprogram parametryczny FREZOWANIE_GWINTU umieszczony w katalogu Podprogramy (konieczność stosowania polecenia EXTERN), w którym odbywa się sam proces frezowania gwintu. Natomiast reszta poleceń niezbędnych do wykonania obróbki oraz wywołanie podprogramu parametrycznego FREZOWANIE_GWINTU są w programie głównym.

W przykładzie do podprogramu FREZOWANIE_GWINTU parametry przekazywane są na trzy sposoby: za pomocą stałych wartości, za pośrednictwem R-parametrów oraz za pośrednictwem zmiennych lokalnych LUD.

```
EXTERN FREZOWANIE_GWINTU(REAL,REAL,REAL,REAL,REAL,REAL)
DEF REAL WSPOL_X,WSPOL_Y,WSPOL_Z,SREDNICA,SKOK,
LICZBA_ZWOI
WSPOL_X=50 R0=50  ;wspolrzedna srodka w osi X
WSPOL_Y=50 R1=50  ;wspolrzedna srodka w osi Y
WSPOL_Z=5   R2=5  ;wysokosc poczatkowa w osi Z
SREDNICA=52 R3=52  ;srednica frezowanego gwintu
SKOK=5 R4=5  ;skok gwintu
LICZBA_ZWOI=12 R5=12 ;liczba zwoi
T="THREAD CUTTER"
M6
G17 G54 G94 D1 F100 S2000 M3
WORKPIECE(,"",,"BOX",112,WSPOL_Z-5,-SKOK*(LICZBA_ZWOI-2),-
80, WSPOL_X-50,WSPOL_Y-50,WSPOL_X+50,WSPOL_Y+50)
FREZOWANIE_GWINTU(50,50,5,52,5,12)  ;lub
FREZOWANIE_GWINTU(R0,R1,R2,R3,R4,R5)  ;lub
FREZOWANIE_GWINTU(WSPOL_X,WSPOL_Y,WSPOL_Z,SREDNICA
,SKOK, LICZBA_ZWOI)
R0=0 R1=0 R2=0 R3=0 R4=0 R5=0
M2
```

Natomiast treść podprogramu FREZOWANIE_GWINTU jest następująca:

```
PROC FREZOWANIE_GWINTU(REAL _WSPOL_X,REAL _WSPOL_Y
,REAL _WSPOL_Z,REAL _SREDNICA,REAL _SKOK,REAL _LICZBA_
ZWOI) SAVE
G90 G0 X=_WSPOL_X Y=_WSPOL_Y Z=_WSPOL_Z
G3 X=IC(_SREDNICA/2-$P_TOOLR) Y=_WSPOL_Y Z=IC(-SKOK/4)
CR=(_SREDNICA/2-$P_TOOLR)/2
G3 X=IC(0) Y=IC(0) Z=IC(-_SKOK*_LICZBA_ZWOI) I=AC(_WSPOL_X)
J=AC(_WSPOL_Y) TURN=_LICZBA_ZWOI
G3 X=_WSPOL_X Y=_WSPOL_Y Z=IC(-_SKOK/4)
CR=(_SREDNICA/2-$P_TOOLR)/2
G0 Z=_WSPOL_Z
RET
```

W podprogramie parametrycznym FREZOWANIE_GWINTU dla odróżnienia od programu głównego w nazwach parametrów podprogramu wprowadzono dodatkowy znak podkreślenia. Aby podprogram parametryczny FREZOWANIE_GWINTU stał się cyklem, wystarczy go skopiować do katalogu Cykle uzytkownika i zrestartować układ sterowania. Od tego momentu w programie głównym nie będzie już potrzebne polecenie EXTERN, gdy cykle go nie wymagają.

LITERATURA

1. „*SINUMERIK 840D sl/828D. Przygotowanie do pracy. Podręcznik programowania*”. 10/2015, 6FC5398-2BP40-5NA3.
2. „*SINUMERIK 840D sl/828D. System variables. Parameter Manual*”. 10/2015, 6FC5397-6AP40-5BA3.
3. „*SINUMERIK 840D sl/828D. Podstawy. Podręcznik programowania*”. 10/2015, 6FC5398-1BP40-5NA3.