Parametric programming of CNC machine tools

Rafał Gołębski [1,*]
[1] Czestochowa University of Technology, Institute for Mechanical Technologies, Al. Armii Krajowej 21, 42-201 Czestochowa, Poland

Abstract. The article presents the possibilities of parametric programming of CNC machine tools for the SINUMERIK 840D sl control system. The kinds and types of the definition of variables for the control system under discussion described. On the example of the longitudinal cutting cycle, parametric programming possibilities are shown. The program's code and its implementation in the control system is described in detail. The principle of parametric programming in a high-level language is also explained.

1 Introduction

CNC machine tools, including multi-axial multi-purpose machine tools, find increasingly wide application in many branches of industry. With the development of the design of CNC machine tools, a continuous development of their control systems, as well as CAD/CAM type software for their programming, follows. While the machining of geometrically simple items can be programmed manually in the EIA/ISO code or using dialogue programming (even not knowing G-codes), the programming of the machining of a geometrically complex part on a multi-axial CNC machine tool without the automatic generation of the machine tool control code using CAD/CAM software is very difficult, if not impossible. The ease of programming CNC machine tools allows their effective use in small-lot or even unit production.

In practice, many plants manufacture, in larger or smaller lots, different geometrically similar simple parts, for the programming of which parametric programming is the cheapest solution. The parametric programming of CNC machine tools is a type of programming, in which controlled axis positions, feed functions and velocities will be defined in the program as parametric expressions [1]. Introducing variables in a parametric form to the program allows the program being created to be made more flexible. The architecture of the created program needs to be reduced solely to the exchange of data in the declared parameters. It is also possible to perform computation on variables, to make use of conditional instructions, jump commands, branching, loops, or, for example, to use complex array definition instructions in the programming process [2]. The majority of CNC machine tool control systems enable the user to use the parametric programming method; the parametric program can be a complement to the main program, as a subprogram, or can also make a separate cycle for a given specificity of machining [3, 4].

* Corresponding author: rafal@itm.pcz.pl

Using the parametric code in the area of the machining program's iteration and geometric variable parametrization operations may substantially reduce the program's volume and, as a consequence, improve its performance [5, 6].

After building a parametric program, the user can create his own dialogue window to enable the efficient management of the program and, using relevant program structures, to reduce the likelihood of making errors.

2 Parametric programming in the SINUMERIK control system

Parametric programming in the architecture of the SINUMERIK 840D sl control system involves using the machine tool's control system and its available, usually limited memory for storing variables. The control system memory is furnished with a one-dimensional array, to the cells of which reference can be made via the cell addresses. In those cells, different types of data can be used as variables in the control system to perform, with their use, different (mathematical, structural) functions, like in other high-level languages (e.g. C, Pascal). Such a solution can be very convenient in the case of programs for machining parts with a limited number of parameters, which can be readily changed on the control panel, if necessary.

In the SINUMERIK control system, three types of variables can be defined:

– *User defined variables:*

LUD (Local User Data); this variable is applicable in the entire program.

GUD (Global User Data); the range of GUD variable influence refers to the control system's kernel (NCK – Numerical Control Kernel).

PUD (Program User Data); the PUD variable applies to programs and subprograms, in which it is defined; this variable should be declared under a unique name and it may not contain words reserved for control system commands. PG variables are created with the start of the machining program and removed upon its ending or resetting.

These variables can be accessed in the machining program, however, only upon meeting the condition, when the $MN_LUD_EXTENDED_SCOPE machine variable parameter is set to the value "1". Otherwise, user-defined variables defined in a given main program will only be active in that program.

– *Computational variables.* These are real-type variables predefined by the manufacturer, and referred to as so-called R parameters. The R variable is used for computations and transformations in the program. It has an R address reserved to it, with a successive number, e.g.: R512. Computation parameters, and essentially their values, can be assigned in the program to NC control addresses. In the basic system, the computation parameter number R is contained in the range 0-99, which can be extended through the modification of the control system machine data. R parameters were used in older SINUMERIK control systems; presently, however, using R parameters is gradually becoming unjustified. The use of the user's variable ensures better flexibility in programming, while the employed language is slowly beginning to resemble higher-level programming languages.

– *System variables.* These variables have an impact on the control system's configuration and its functioning in relation with the written program. System variables can be stored in, and read out from the control system's area, and the following can be distinguished here: zero point shift variables, tool correction definition variables, current position value variables, and measurement value variables. To be specially distinguished, system variables always start with the sign $. Thus, the relevant machine variable defining the starting position of the tool in a given axis prior to program simulation is defined as $MAS_SIM_START_POSITION and is available in the machine axis data tab and, like other variables, can be edited.

Defining variables in the program involves specifying the allowable data type [7]. For predefined R variables and system variables, their type is predetermined, while in the case of user variables, the variables can be defined – Table 1.

Table 1. The types of variables and the range of their values.

Type of variable	Definition	Allowable value range
INT	Integer numerical values	-2147483646 ... +2147483647
REAL	Fractional numbers with a decimal point	$\pm (2{,}2 * 10^{-308} \ldots 1{,}8 * 10^{+308})$
BOOL	Logic values: TRUE (1), FALSE (0)	1 or 0
CHAR	ASCII character	Respectively, according to the ASCII code, it is possible to assign Polish characters
STRING	Character chain in brackets [ ]	Up to 200 characters can be defined
AXIS	Axis identifiers (addresses)	Axis identifiers or spindles existing in a given channel
FRAME	Geometrical data of coordinate system transformations	

Defining user variables always starts with the DEF command. The definition syntax is as follows:

DEF <variable type> < variable name> = <variable value>

When defining a variable it should remembered that the definition must always take place in a separate block. Many variables of one type may be defined in a block. A variable name may not exceed 31 characters, and its first two characters must always be letters or underscores. The $ sign and commands characteristic of a given type of NC control may not be used as the name of a variable. A constant or computational value may always be assigned to a variable. In the case, where no value has been assigned to the variable in the variable definition, the system will automatically assign it a default value equal to 0.

The SINUMERIK control system is furnished, as standard, with a definite number of cycles for the most frequent turning, milling or drilling operations. Very often, cycles offered by the manufacturer are limited, and their extension by not advanced user by adding further technological solutions is practically impossible due to their undescribed structure. The SINUMERIK control system offers the capability to define the user's own cycles as subroutines with a parametric structure. They can be stored in the user cycle directory, and after restarting the control system these subroutines will be treated as cycles by the control system. It is also important to make sure that the EXTERN command be removed from the

program; in that case, the parametric subroutine will be then treated by the control system as a cycle.

2.1 Example 1

The creation of the user's own cycle is presented on the example of building a longitudinal turning subroutine. At the beginning of the program, all parameters, their names and their entering order must be defined.

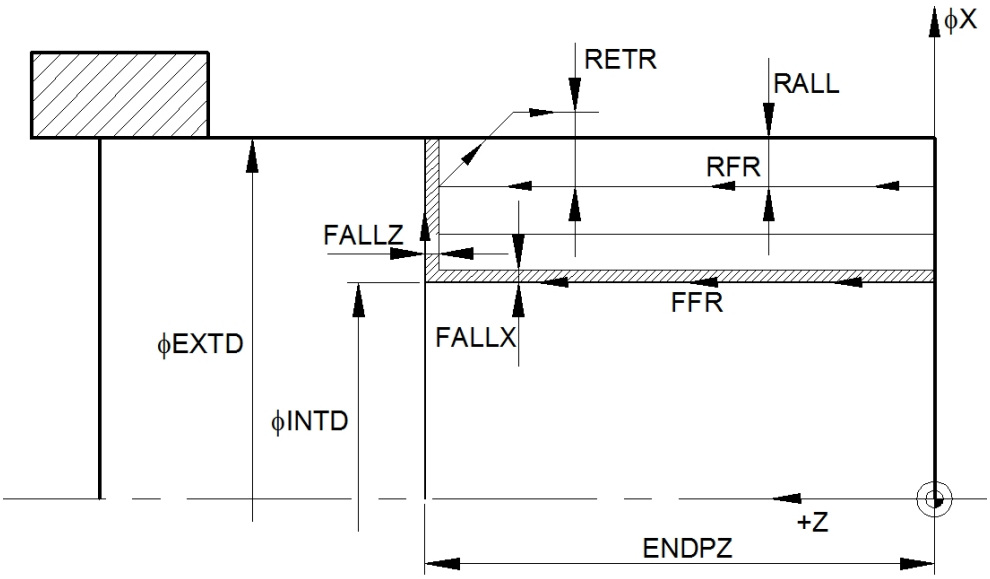

Fig. 1. Longitudinal turning parameters for the TURNING cycle.

For the cycle defined by the name TURNING, the following parameters have been defined in Figure 1:

- **EXTD – external diameter** (type REAL) defines the machining starting location in terms of diameter. For the correct division of the allowance to the first pass, this value should coincide with the blank diameter – the coordinate in the direction of the X axis is given in terms of diameter [mm].
- **INTD – internal diameter** (type REAL) defines the diameter after machining – the X coordinate is given in terms of diameter [mm].
- **ENDPZ - end point Z** type REAL, defines the turning length with reference to the zero point of the workpiece, the coordinate in the direction of the Z axis is given with the positive sign [mm].
- **RALL** – **roughing allowance** (type REAL) defines the thickness of the machined layer in the roughing pass; an absolute value on the radius [mm] .
- **FALLX** – **finishing allowance X** (type REAL) defines the amount of the allowance left for roughing in the direction of the X axis; an absolute value on the radius [mm].
- **FALLZ** – **finishing allowance Z** (type REAL) defines the amount of the allowance left for roughing in the direction of the Z axis; an absolute value in the Z axis direction [mm].
- **RFR** – **roughing feed rate** (type REAL) defines the magnitude of the synchronous feed for roughing; the value defined in [mm/rev].

- **FFR** – **finishing feed rate** (type REAL) defines the magnitude of the synchronous feed for finishing; the value defined in [mm/rev].
- **RETR – retract tool** (type REAL) defines the magnitude of tool retraction after performing the roughing pass in the X axis direction; an absolute value on the radius [mm].
- **LAYER_C** (type INT) defines the number of cycle layers; the parameter value calculated based on the parameters declared for the cycle.
- **COUNTER** (type INT) – the counter of the current cycle layer.

After defining the parameters (PUD variables), the structure of the subroutine is shown below. The program code can be transferred to SINUMERIK, where, as a file with the extension *.SPF (Sub Program File), it will perform the function of the subroutine:

```
PROC TURNING_CYCLE(REAL EXTD, REAL INTD, REAL ENDPZ, REAL RALL,
REAL FALLX, REAL FALLZ, REAL RFR, REAL RFR, REAL RETR) SAVE
; definition of the procedure TURNING_CYCLE and individual cycle parameters
DEF INT LAYER_C, COUNTER=0
        ; definition of the parameter of the total number of layers and of the layer counter
LAYER_C=((EXTD/2-INTD/2)-FALLX) DIV ()
    IF (((EXTD/2-INTD/2)-FALLX) MOD ()) <> 0
        ; computing the total number of cycle layers, LAYER_C, verifying if
        ; the thickness of the machined layer divides in a fractional manner, and if so:
    LAYER_C=LAYER_C+1
 ENDIF
 =(EXTD-INTD-ENDPZ)/LAYER_C
        ; otherwise, computing the number of layers without a fraction:
G90 G0 X=EXTD+1 Z=1 DIAMON
   G0 X=EXTD
        ; establishing the absolute dimensioning, activating diametral
        ; dimensioning, running the tool to the point before the workpiece
CYCLE_START:    ; starting the cycle procedure
MSG ("**ROUGHING** CURRENT LAYER:"<< COUNTER+1 << " from " <<
LAYER_C)
        ; displaying messages on the screen
        ; about the machining type and the currently machined layer
G91 G1 X=- F=RFR
   G1 Z=(1+ENDPZ-FALLZ)
   G1 X=RETR
   G1 Z=-(1+ENDPZ-FALLZ)
   G0 X=-RETR
        ; declaration of incremental dimensioning, parametric cycle description
GOTOF CYCLE_END
        ; procedure of forward jumping to procedure CYCLE_END
CYCLE_END:
COUNTER=COUNTER+1
IF COUNTER<LAYER_C GOTOB CYCLE_START
        ; program ending procedure, if the number of layers is greater than
        ; from the current layer, jumping back to procedure CYCLE_START
G90 X=EXTD+50 Z50
        ; absolute dimensioning, retract to a safe distance from material.
MSG ("FINISHING")       ; message "finishing"
```

```
T="FINISHING_T35 A"     ; declaration of the finishing machining tool
G90 G0 X=INTD Z=1
   G1Z=ENDPZ
   G1X=EXTD+1
   G0Z=1
            ; absolute dimensioning, parametric description of the finishing cycle
MSG()     ; message bar clearing
M17       ; end of subroutine
```

In the subprogram, blocks preceded by a semicolon are treated by the control system as comments. In the above code, the procedure of the finishing machining cycle has been incorporated. It has been preceded by the departure and the tool exchange function. This solution is correct. The finishing machining procedure can also be included in a separate subroutine, in that case, this would require the reference to the subprogram to be appropriately addressed in the main program.

The structure of the main program is shown below:

```
EXTERN TURNING_CYCLE(REAL, REAL, REAL, REAL, REAL, REAL, REAL,
REAL, REAL)
T1 D1 S2700 M3
            ; assigning the TURNING_CYCLE subroutine and handing over the parameters
G18 G54 G96 S300 LIMS=3200
            ; definition of the machining plane XZ of the workpiece's coordinate system
            ; switching on the constant turning speed, limiting
            ; spindle rotational speed
G90 G0 X50 Z50
            ; absolute dimensioning, running the tool to the material
            ; to a safe distance
 WORKPIECE(,,,"CYLINDER",192,0,-170,-150,100)
            ; blank declaration, a 100mm-diameter 170mm-long cylinder
            ; allowance for end face planning, 0mm
            ; available safe turning length smaller than 150mm

                        ;EXTD--INTD--ENDPZ-RALL-FALLX-FALLZ-RFR-FFR--RETR
TURNING_CYCLE (100.00,  50.00,  -100.00,  2.50,  0.25,   0.25,  0.25,  0.1,  1.0)
            ; declaration of cycle parameter values entered in the order
            ; of defining variables in the subprogram
            ;
M30       ; end of the main program
```

The above code can be copied to the control system and stored with the extension *.MPF (Main Program File) to perform the function of a program. At the next step it was necessary to test the running of the program. For this purpose, the SINUMERIK Operate 4.5.ed.II software was used. It has a graphical dialogue programming overlay, SHOP-Turn, within which simulation was carried out – Figure 2. On the top bar, information about the program's progress is displayed, including the machining type and information on the currently machined layer.

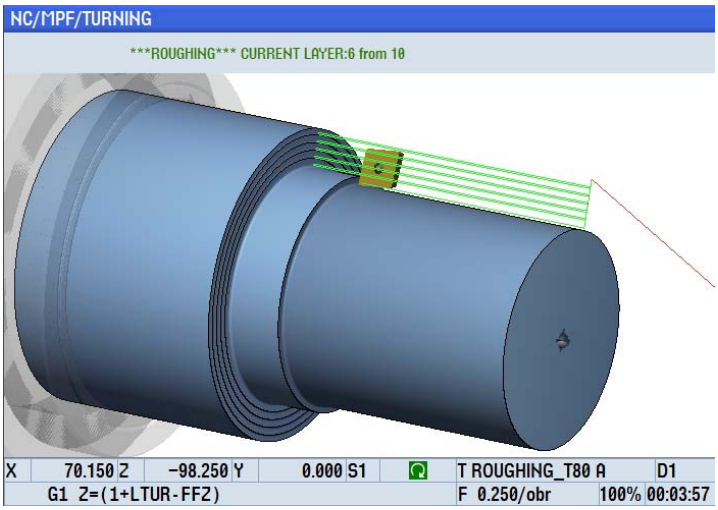

Fig. 2. A view of the cycle simulation window.

The program simulation progressed smoothly, which indicates that the program has been written correctly. The subprogram can be copied to the user cycle directory. Restarting the control system will make the program visible to the NCK. We remove the EXTERN command from the subprogram's code. At the next step, the program was run from the machine tool's control panel level. During the operation of the program, the currently executed machining block is displayed on the machine's screen. In the case of parametric programming, the program code in the parametric system will be displayed. For an operator, who has not created the program, this may pose a great impediment. The solution is to activate the 'BASIC BLOCKS' option, which will allow the operator to observe the functions used by the program and the coordinates after the translation of the program into the ISO code. The lower left window presents the currently executed block in the parametric code, while the lower right window, in the ISO code – Figure 3.

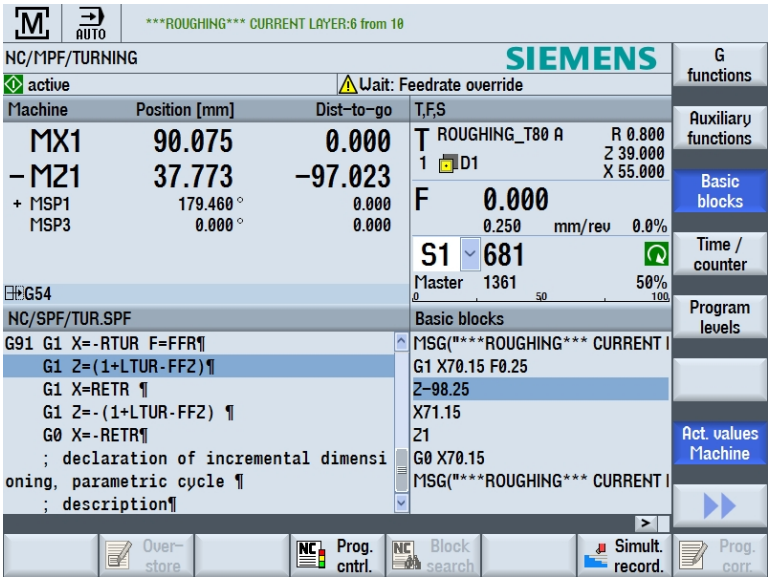

Fig. 3. The window of running the program on a machine tool.

3 Parametric programming in a high-level language using a PC

Considering the increasing capabilities and increasingly wide application of the advanced CAD/CAM software, as well as the centralization of CNC machine tool software, due to the costs and striving to construct Flexible Manufacturing Systems (FMS), CNC machine tool software is more and more often transferred outside the machine tools (also because of the Ethernet operation and remote data transmission capabilities).

In this case, the aim is to utilize the capabilities of the PC for engineering calculations and to generate the CNC machine tool code in the computation program [8]. It is possible to fully utilize the capability of the high-level programming language and to generate, from a single program, control codes for different machine tools in a given plant in the form of a parametric program. Only the indispensable selected parameters and the generated machine tool control code are entered to the final machine tool control program, whereas in the program's computational module, which can be very complex, the number of variables of multi-dimensional arrays may be very large.

3.1 Example 2

A program for machining worms of an arbitrary profile has been developed for VCN-type (multi-purpose machine tool) and QTN-type (turning center) Integrex CNC machine tools manufactured by MAZAK [9]. The program in Delphi includes a module for determining the tool positioning parameters, depending on the cutter diameter, the preset worm axial profile and the required machining accuracy.

The program - Figure 4 generates also the machine tool control code – Figure 5. It is important that the program allows the utilization of the capacity of a PC and enables it to run under the Windows environment, perform mathematical operations and use subprograms in any high-level programming language, and makes it possible to generate the control program for different machine tools, which have been used in the developed machine tool control program - Figure 5. In the example under discussion, the worm is machined by the step-by-step method – Figure 6a.

The machining performance depends on the cutting parameters and the number of tool passes.

```
for i:=number_n downto 1 do
  if bvnl[3,i]<>0.0 then
    begin
      l_mazak:=l_mazak+1;
      Writeln(outf,'G90 G0 X#'+IntToStr(600+l_mazak));
      Writeln(outf,'G95 G01 C0.0 F#122');
      Writeln(outf,'G0 Z[#'+IntTostr(500+l_mazak)+']');
      Writeln(outf,'G91 G95 G1 C2979.383 X[#115+#116+#117] F#105');
      Writeln(outf,'G90 G0 Z[#120+2*#114]');
      Writeln(outf,'G0 X#'+IntToStr(800+l_mazak));
      Writeln(outf,'G95 G01 C0.0 F#122');
      Writeln(outf,'G0 Z[#'+IntTostr(700+l_mazak)+']');
      Writeln(outf,'G91 G95 G01 C2979.383 X[#115+#116+#117] F#105');
      Writeln(outf,'G90 G0 Z[#120+2*#114]');
    end;
```

Fig. 4. A fragment of the machine tool control code generating program.

The tool positioning coordinates for each cutter pass are calculated in the computation program and transferred to the machine tool control system. These values are stored in the memory cells of the control system, and therefore their number is limited by the storage capacity and may not be arbitrary. An advantage of the adopted machining method is the capability to make a worm of any profile and an arbitrary module with a universal tool (a finger ball-end mill) – Figure 6b.

```
G90 G0 Z[#120+2*#114]
G90 G0 X#610
G95 G01 C0.0 F#122
G0 Z[#510]
G91 G95 G1 C2979.383 X[#115+#116+#117] F#105
G90 G0 Z[#120+2*#114]
G0 X#810
G95 G01 C0.0 F#122
G0 Z[#710]
.
.
.
G90 G0 Z[#120+2*#114]
G0 X#813
G95 G01 C0.0 F#122
G0 Z[#713]
G91 G95 G01 C2979.383 X[#115+#116+#117] F#105
```

Fig. 5. A fragment of the machine tool control program code.

The parametric variables are stored in the program's memory under addresses indicated by the sign #. The control program was read in to machine tool control system via a USB (the program can also be transmitted through the Ethernet). The parameters can be displayed and modified on the machine tool control panel. The machine tool control program generated automatically from the universal CAM software program will generally be much longer, and any modifications could be very difficult, if not impossible, to carry out. The CNC machine tool control program written in a high-level language using parametric variables will be a cheaper (there is no need for using the expensive CAM software) and optimal (easy to modify) solution.

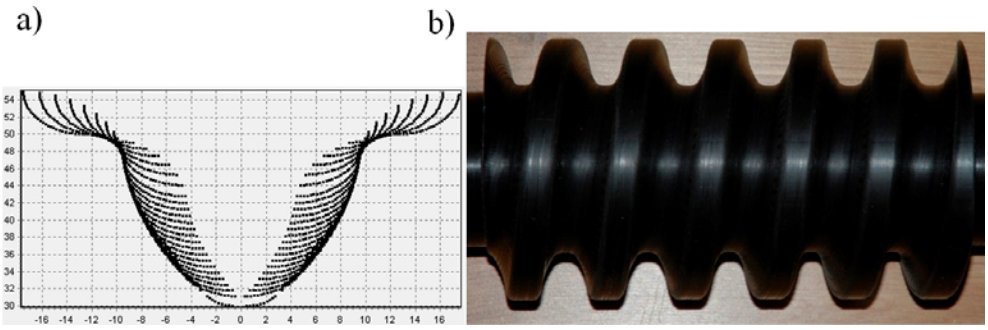

Fig. 6. An example of a worm cut by the step-by-step method (m=10): a) the worm profile and tool profiles in the worm axial section, b) a finished worm.

4 Conclusions

Parametric programming requires the knowledge of G-codes and does not require any additional tools to facilitate programming, often available optionally, (e.g. 3D visualization), and is cheap (is used in the automotive industry).

Parametric programming is convenient and easy to master, and is frequently used in machining geometrically simple parts. However, considering the capacities of the one-dimensional arrays of control systems to store variables, these programs have some limitations.

Parametric programming in a high-level language involves performing engineering calculations, while fully utilizing the capabilities of such calculations, and ultimately generating the parametric CNC machine tool control code. This is usually very difficult (the engineering calculation module may be very complex), but the capabilities of such programming are greater. The machine tool resources, the programming language capability, the ability to generate the code for several machine tools and the possibility of carrying out complicated calculations can all be utilized for machining geometrically more advanced parts.

For the discussed control system, parametric programming provides a perfect alternative to the SHOP-Turn and SHOP-Mill dialogue programming overlays being in use, which are a costly option for the SINUMERIK. Moreover, for a given parametric cycle, there is a possibility of building the user's own graphic window for declaring cycle parameters.

To sum up, with its open architecture, the SINUMERIK environment provides excellent opportunities for using parametric machine tool programming solutions.

References

1. M. Lynch, *Parametric programming for computer numerical control machine tools and touch probes* (Socienty of Manufacturing Engineers, 1997)
2. R. Stryczek, B. Pytlak, *Flexible programming of machine tools,* PWN (Scientific Publishing House, 2011)
3. D. Manocher, Comp. & Ind. Eng. **35**, 33, (1998)
4. G. Nikiel, Adv. in Man. Sci. and Techn. **33**, 33, (2009)
5. G. Nikiel, *Programming of CNC machine tools on the example of the Sinumerik 810D/840D control system* (ATH University of Bielska-Biala, 2004)
6. J. Szadkowski, R. Stryczek, G. Nikiel G, *The design of technological processes for numerically controlled machine tools* (ATH University of Bielska-Biala, 1995)
7. SINUMERIK 840D, *Job planning, Programming Manual* (2006)
8. T. Nieszporek, A. Piotrowski, *Robotics in Theory and Practice*. *Applied Mechanics and Materials,* 203, (Trans Tech Publications, Zurich, 2013)
9. Yamazaki, INTEGREX Revolutionary Multi-tasking Machine, https://www.mazak.com.sg/machines/process/multi-tasking/Yamazaki, (2006)