FAKULTET INŽENJERSKIH NAUKA
UNIVERZITETA U KRAGUJEVCU
Katedra za proizvodno mašinstvo
Kragujevac, Srbija

37. SAVETOVANJE PROIZVODNOG MAŠINSTVA SRBIJE
- SPMS 2018 -
37th INTERNATIONAL CONFERENCE ON PRODUCTION ENGINEERING OF SERBIA
- ICPE-S 2018 -

25 – 26 October 2018, Kragujevac, Serbia

ZBORNIK RADOVA
PROCEEDINGS

EDITORS: Bogdan Nedić, Slobodan Mitrović

ПРОИЗВОДНО МАШИНСТВО СРБИЈА

37. Savetovanje Proizvodnog Mašinstva Srbije - SPMS 2018
ZBORNIK
37th International Conference on Production Engineering Of Serbia - ICPE-S 2018
PROCEEDINGS

ISBN: 978-86-6335-057-1

Urednici: **Editors**:	**Bogdan Nedić** **Slobodan Mitrović** University of Kragujevac, Faculty of Engineering
Izdavač: **Publisher**:	**University of Kragujevac, Faculty of Engineering** Sestre Janjić 6, 34000 Kragujevac, Serbia
Za izdavača: **For the Publisher**:	**Dobrica Milovanović** University of Kragujevac, Faculty of Engineering
Tehnička obrada: **Technical editor**:	**Suzana Petrović Savić** **Dragan Džunić** **Marko Pantić** University of Kragujevac, Faculty of Engineering
Printed by:	**Inter Print** Jurija Gagarina 12, 34000 Kragujevac, Serbia
Circulation:	100 copies

The publication of this Proceedings was financially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

Society of Production Engineering

SPMS 2018
37. Savetovanje Proizvodnog mašinstva Srbije
ICPE-S 2018
37th International Conference on Production Engineering of Serbia

Faculty of Engineering
University of Kragujevac

Kragujevac, Serbia, 25 – 26. October 2018

CONTROL AND PROGRAMMING SYSTEM OF PARALLEL KINEMATIC MACHINE

Zoran DIMIĆ[1,*], Dragan MILUTINOVIĆ[2], Saša ŽIVANOVIĆ[2], Stefan MITROVIĆ[1]
[1]LOLA Institute, Belgrade, Serbia, zoran.dimic@li.rs, stefan.mitrovic@li.rs
[2]University of Belgrade, Faculty of Mechanical Engineering, Belgrade, Serbia,
dmilutinovic@mas.bg.ac.rs, szivanovic@mas.bg.ac.rs
*Corresponding author: zoran.dimic@li.rs

Abstract: *The paper presents research and development results of control and programming system of parallel kinematic machine (PKM). The control system is based on G-code and two separate units: a real-time control system and an off-line system for G-code writing and editing, checking programme syntax and semantics and discovering and avoiding singular positions by simulating the machining programme in accordance with machine tool kinematics.*

Keywords: *Parallel Kinematic machine tools, Control system, Programming system, Off-line programming, G-code*

1. INTRODUCTION

G-code programming has survived since the very beginning of numerical control until today. Numerous CAD/CAM tools used for 3D modelling and generating G-code programmes for machine tools of all types have been developed and are in use. G-code has become synonymous with machine tools and today and it represents an intuitive and widely accepted programming model [1]. Modern and fast hardware with an efficient real-time operating system is an ideal platform for software-oriented CNC [2, 3, 4].

The revolution in the development of machine tools was brought about by the appearance of parallel kinematics machine (PKM) [5]. Their lower weight compared to conventional machine tools for the same stiffness of mechanism due to their closed kinematics loops with the possibility of achieving far greater machining speeds is the main advantage that the technology of parallel kinematics machines brings [6]. On the other hand, the complexity of kinematic modelling largely impedes configuring the control of machine tools with parallel kinematics [7]. The inverse kinematics equations associated with singularities in the PKM workspace render the conventional machine tool control system useless. G-code interpretation during machining, without any previously performed simulations, could lead to unforeseen effects caused by mechanism singularities of PKM.

The paper proposes a control system based on G-code interpretation, which consists of a real-time software system for control of PKM and an off-line system for G-code writing and editing, checking programme syntax and semantics and simulating the machining programme in accordance with machine tool kinematics.

2. 2. CONTROL SYSTEM STRUCTURE 3.

The complex kinematic structure of PKM indicates the construction of a control system that encapsulates the system kinematics. This enables the programming of PKM to be kinematically independent. It must be taken into account here that, before being executed in the machine, G-code must pass appropriate checks aimed at:

- Discovering and avoiding singular positions;
- Positioning the workpiece within the boundaries of PKM workspace;
- Identifying and eliminating collision situations of the machine and the workpiece, the machine and its environment and the machine elements with each other.

To overcome the difficulties in PKM programming, a control system has been proposed that consists of two software and hardware units. Figure 1 shows the structure of the proposed control system that includes:

Off-line programming and simulation system with the following tasks:

1. G-code syntax and semantics check;
2. Generating appropriate motion instructions at the simulator level and interpolating the programmed path;
3. Emitting a P-code as set of machining instructions checked by simulation in the previous steps.

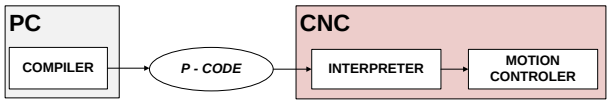

Figure 1. Control System Structure

The CNC system, which, by analogy with the off-line system, interprets the P-code in real time, translating the code instructions into:

1. Motion by means of real-time interpolators and inverse kinematics;
2. Sequential functions by means of real-time sequential automaton.

3. OFF-LINE PROGRAMMING AND SIMULATION SYSTEM

Figure 2 shows a Unified Modelling Language (UML) package diagram of developed off-line programming system. The functionality of all software packages in the system is integrated by the GUI trough several tasks delegated to it:

- G-code document browsing, opening, editing and saving;
- Interfacing G-code compiler (NC Compiler) functionality by using visual command elements;
- Displaying messages during the compilation, such as current compiling G-code programme line during the compilation, error messages etc.;
- Displaying results of interpolation task through graphical representation of programmed tool path;
- Hosting CNC Terminal which is transferring P-code file to controller and interfacing CNC controller functionality.

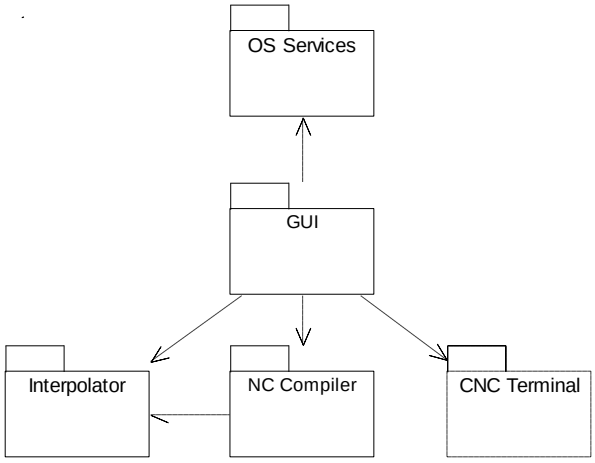

Figure 2. UML package diagram of NC Compiler developed off-line programming system

The compiler design is preceded by defining the Backus - Naur Form (BNF) notation for programming language, in our case G-code.

Tabel 1. BNF notation for G-code provides the BNF notation for the basic set of G-code instructions necessary and sufficient for programming one vertical milling machine.

Table 1. BNF notation for G-code

Rule No.	BNF notation	Limitations
1.	**Program** = ProgramBorder StartProgram ProgramBody ProgramEnd ProgramBorder	
2.	**ProgramBorde**r = "%"	
3.	**StartProgram** = "L" "INTEGER"	StartProgram = "L" "INTEGER"- "INTEGER" must contain four digits
4.	**ProgramBody** = Line {Line}	
5.	**ProgramEnd** = "M30"|"M02" "NEW_LINE"	In single line can be M02 or M30
6.	**Line** = "NEW_LINE" LineNumber [ExactStopInstruction | StopInstruction | InstructionQueue]	
7.	**LineNumber** = "N" "INTEGER"	LineNumber = "N" "INTEGER"- Line numbers must be in ascending
8.	**InstructionQueue** = { GInstruction | MInstruction | Coordinates | SCode | SRCode | FCode | HCode}	
9.	**ExactStopInstruction** = G09 | (G04 [PCode])	
10.	**StopInstruction** = "M00"	
11.	**SCode** = "S" " INTEGER l"	SCode = "S" "INTEGER"- SCode can be only one in single line - INTEGER range is: 00 - 99999
12.	**HCode** = "H" "INTEGER"	HCode = "H" "INTEGER" - HCode can be only one in single line - INTEGER range is: 00 - 99
13.	**FCode** = "F" "Real"	FCode = "F" "Real"- FCode can be only one in single line - "Real" is in format 5.1
14.	**PCode** = "P" "INTEGER"	PCode = "P" "INTEGER" - PCode can be only one in single line
15.	**Coordinate** = ("X" | "Y" | "Z" | "I" | "J" | "K" | "R") SimpleExpression	
16.	**SimpleExpression** = [Sign] Monom {AddOperatorAndMonom}	
17.	**Sign** = "+"|"-"	
18.	**Monom** = Factor {MulOperatorAndFactor}	
19.	**AddOperatorAndMonom** = AddOperator Monom	
20.	**MulOperatorAndFactor** = MulOperator Factor	
21.	**Factor** = SimpleExpressionWithBrackets | "Real" | Variable	
22.	**SimpleExpressionWithBrackets** = "OPEN_BRACKET" SimpleExpression "CLOSED_BRACKET"	
23.	**GInstruction** = P0Group | P1Group | P3Group | 4Group | P5Group | P6Group | P7Group | P8Group	
24.	**P0Group** = "G15" | "G16"	
25.	**P1Group** = "G00" | "G01" | "G02" |"G03"	
26.	**P3Group** = "G17" | "G18" | "G19"	
27.	**P4Group** = "G20" | "G21"	
28.	**P5Group** = "G43" | "G44"	
29.	**P6Group** = "G53" | "G54" | "G55" | "G56" | "G57" | "G58" | "G59"	
30.	**P7Group** = "G61" | "G62" | "G64"	
31.	**P8Group** = "G90" | "G91"	
32.	**MInstruction** = "M03" | "M04" | "M05" | "M07" | "M07"| "M08" | "M09" | "M10" | "M11" | "M60"	

A compiler generator is usually used when developing a compiler for structured programming languages. Due to the specific structure and semantics of G-code, the NC Compiler proposed in this paper was developed manually (Fig.3).

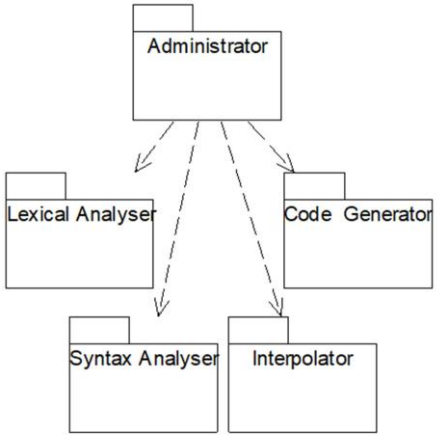

Figure 3. UML package diagram of NC Compiler

The recursive descent method was selected for developing the compiler [8,9]. An intermediate code (P-code) corresponding to the source G-code logic was defined to make the developed compiler portable and adaptable. The compiler translates the source code into the intermediate code, P-code, which is interpreted on the CNC during programme execution.

The compilation is executed in two passes (Fig. 4). In the first pass, the Lexical Analyser package, whose component model is shown in Fig. 5, reads character by character from the G-code file, forms words and, using the Hash algorithm, finds them in the Symbol Table. Afterwards, according to the rules defined in the BNF, a syntax and semantics analysis is performed and the intermediate code is generated.

In the second pass, look a heads are solved and the code generated in the first pass is optimised. The compilation steps, characteristic of higher-level programming languages, are kept to be able to expand G-code with the elements of structured programming languages. All of the above activities are coordinated by the Administrator programming package.

The recursive descent requires observing certain restrictions in language grammar. The language grammar must be LL(1)[1] grammar. Furthermore, the compiler must be developed in a higher-level language that allows recursion. The G-code compiler was developed in the spirit of object-oriented programming in C++.

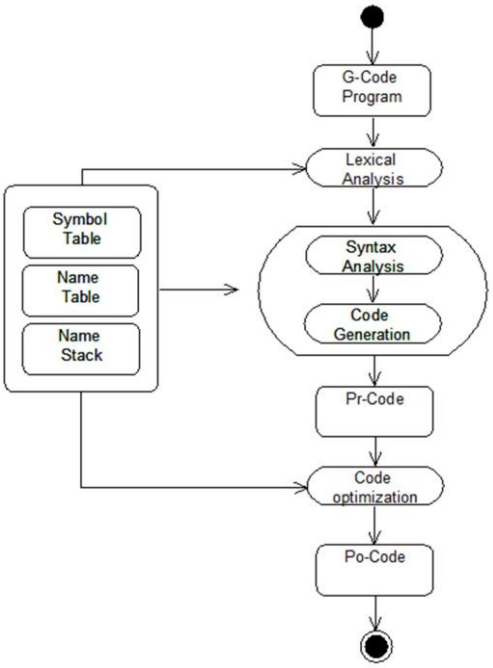

Figure 4. UML activity model of NC Compiler

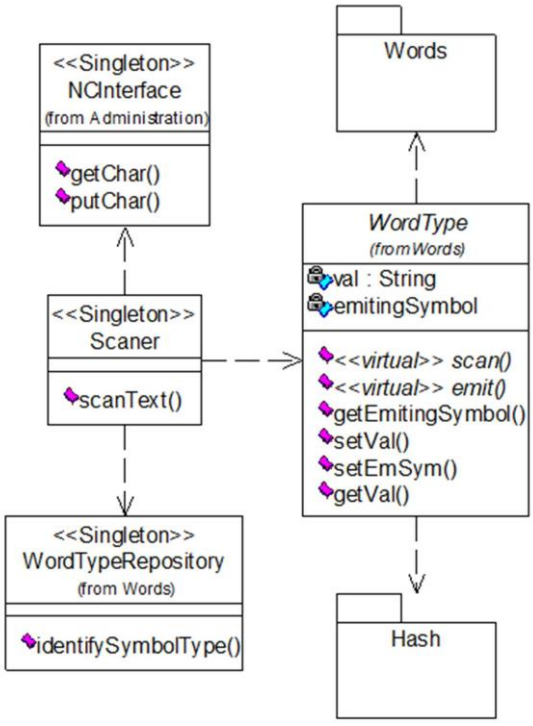

Figure 5. UML component model of Lexical Analyser

[1] Top-down parser which uses 1 token of lookahead when parsing a sentence

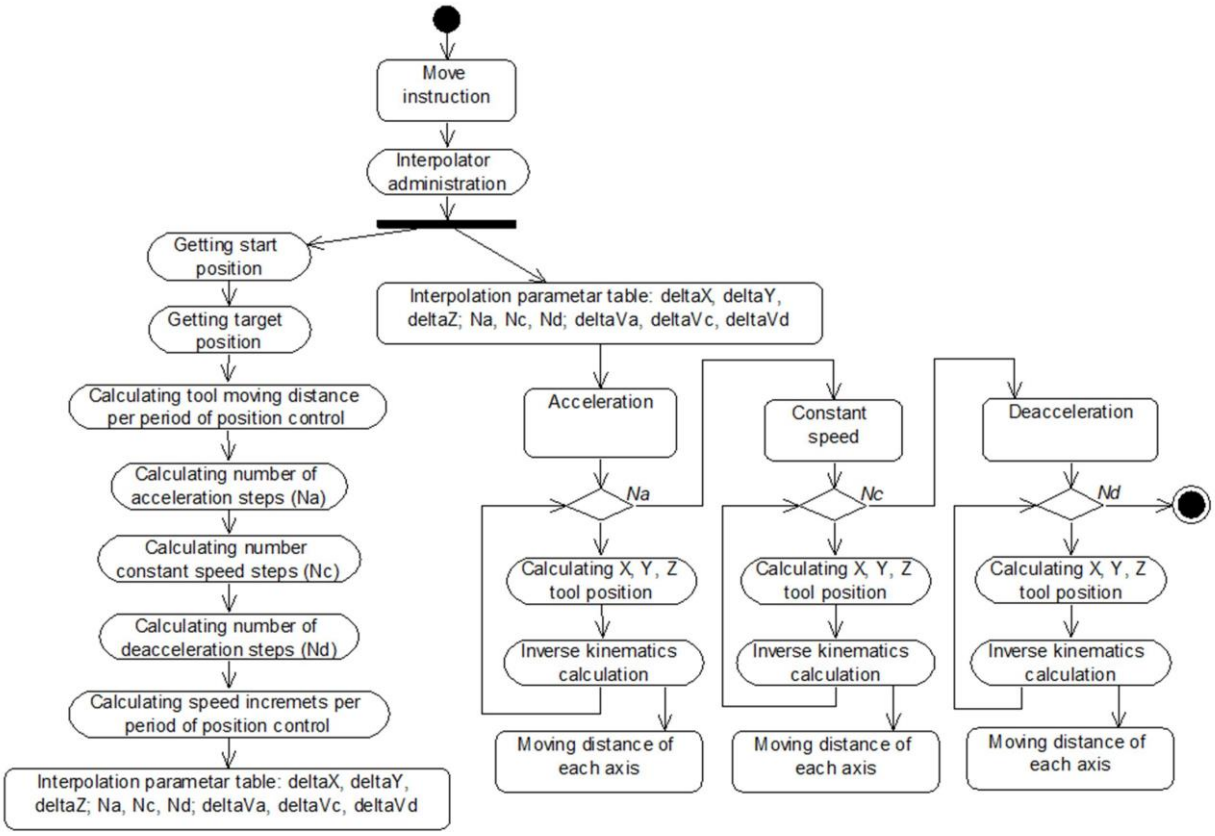

Figure 6. UML activity model of Interpolator

P_o-code was obtained by P_r-code optimisation. P_o-code was adapted to support real-time machining programme execution, motion commands and modularity. P-code was treated as a separate language having a code syntax that defines code commands and code sentences. When designing the code syntax, it was strived to make the code commands correspond directly to the concepts and the source syntax of G-code.

In order for P_o-code to be designated as valid for the execution on a PKM, it must pass one more check, Fig.6.

This check involves interpolation of motion instructions in the off-line system. Then the off-line system becomes a virtual CNC that executes the motion instructions by generating the tool tip path and the profiles of tool tip speeds and acceleration as well as individual joints of the PKM. It is also checked whether the joints are within their defined ranges of motion, speed and acceleration. In the case of detecting any collision situations, the compiler execution stops and an appropriate error message is created. The proposed control system interpolates the path using the Sampled-Data interpolation method [10].

System integrators are provided with the possibility to define maximum joints acceleration for the specific PKM as well as the method of connecting the two motions in the cases where continuous path interpolation is required. It is necessary to set a parameter that defines the radius of the approximate motion area, i.e. the radius of the area in which the tool deviates from the programmed path, in order to reduce inertial forces and maintain the programmed machining speed (Fig.7).

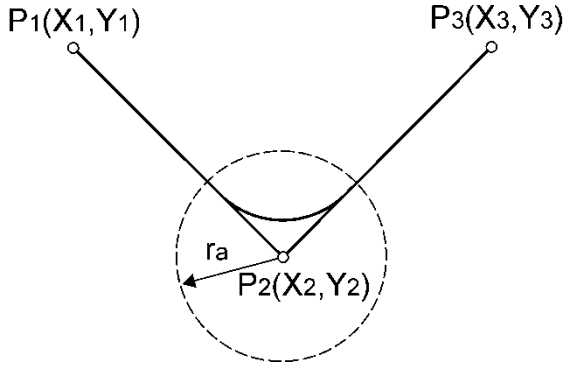

Figure 7. Approximate motion

The GUI of the off-line programming and simulation system is designed and executed under Windows (Fig.8). The CAM-generated G-code is compiled in the off-line system, and the results obtained by the compilation are a simulated tool path (Fig. 8) and a P_t-code, which is transferred by means of the CNC Terminal (Fig. 9) to the CNC, where it is interpreted in real time.

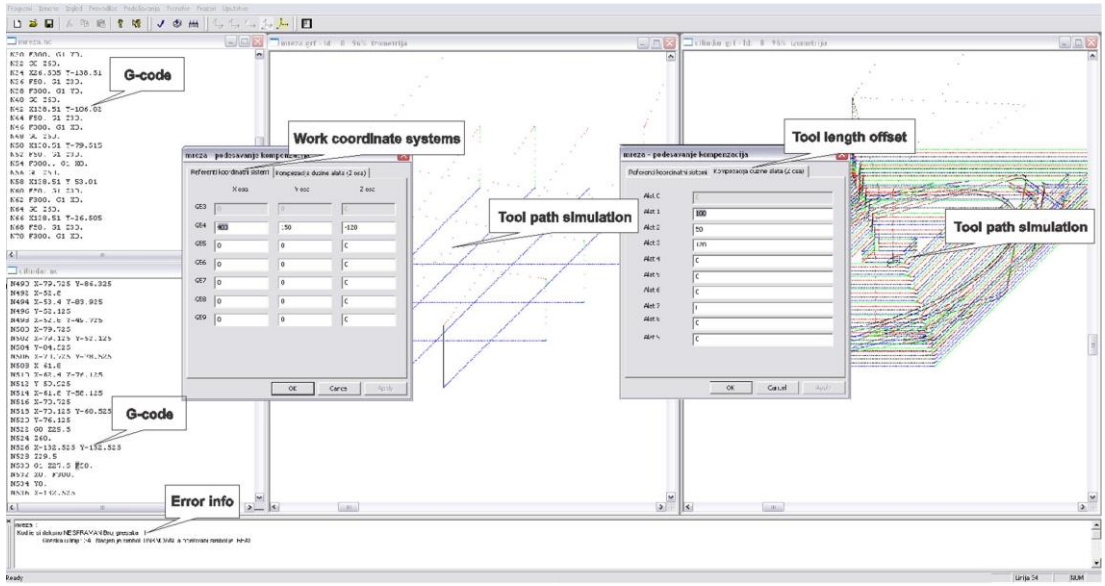

Figure 8. G-code programme and simulated tool path in off-line programming environment

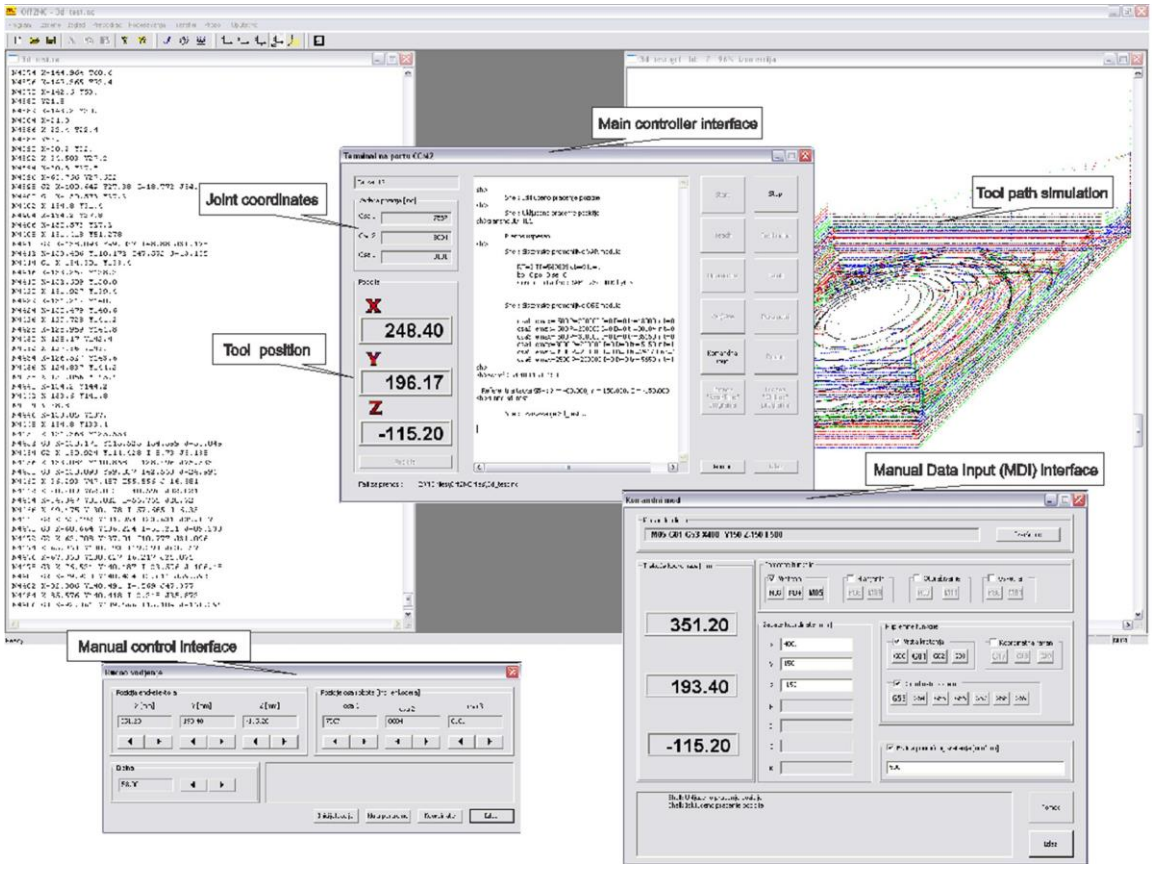

Figure 9. Off-line system with interface for communication with real-time controller

4. REAL-TIME PKM CONTROL SYSTEM

Real-time PKM control system is a software system segment responsible for:

- Maintenance of position servo loops;
- Real-time interpretation of P_t-code commands;
- Communication with the GUI through displaying basic status information about PKM;
- Manual control of PKM in a Cartesian coordinate system;
- Manual control of PKM in the joints coordinate system;
- PKM initialisation.

As software that operates in real time, the real-time PKM control system consists of several competitive tasks of different priorities (Fig.10). Real-time system tools such as signal and event were used for inter-task synchronisation and data modules were used for inter-task data exchange.

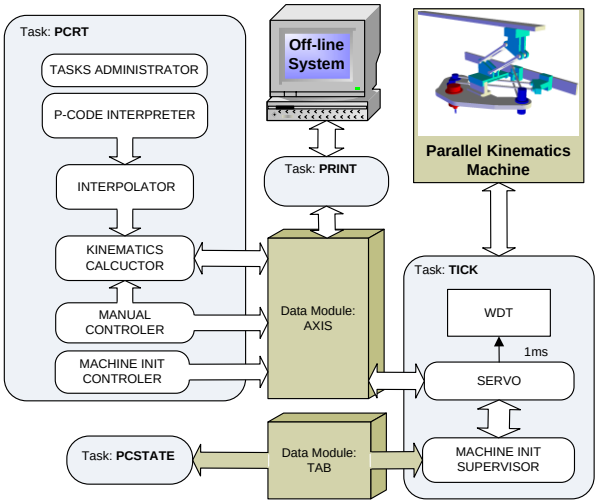

Figure 10. Structure of real-time PKM control system

PCRT (**P**KM **C**ontrol **R**eal-time **T**ask) is the main task in the system that is executed periodically every 5ms. It performs all necessary computer system hardware initialisations, creates necessary resources, sets system parameters and initiates other tasks:

- TICK task with servo control loops;
- PCSTATE task for scanning the status of digital inputs;
- PRINT task for communication with GUI.

TICK is the top priority task that is executed periodically every 1ms by using an installed cyclical alarm. Being the first task, it is started from PCRT and contains an initialisation algorithm and a servo loops.

The PRINT task creates a communication channel to the off-line system GUI. PRINT performs the following functions:

- Provides the protocol for communication with the off-line system GUI;
- Forwards information to the off-line system on the sensor status before and during the PKM initialisation;
- Periodically forwards data to the off-line system on the tool tip position as well as on the positions of PKM joints.

PCSTATE is a task that is activated every 5ms and performs the following functions:

- Scans the status of digital inputs and forwards the information to other tasks;
- Installs the FINISH intercept routine, which waits for the operation end signal from PCRT;

5. EXPERIMENTAL VERIFICATION OF THE CONTROL SYSTEM

For the purpose of experimental verification, the control system was configured to control LOLA pn101_4 V.1. vertical parallel kinematics milling machine (Fig. 11).

Figure 11. Vertical 3-axis parallel kinematics milling machine

Kinematic modelling was performed for the parallel mechanism of the machine tool. Since the system should enable the use of the existing CAD/CAM systems, it was necessary to:

- Establish coordinate systems of the machine, tool and workpiece according to the machine tool conventions (Fig. 12);
- Define position and joint coordinates;
- Solve the direct and inverse kinematics problem;
- Analyse singularities and define the workspace [11,12].

Figure 12 represents a geometric model of a vertical milling machine prototype for which the developed control system is configured.

Coordinate frames $\{B\}$ and $\{P\}$ attached to the base and mobile platform are always mutually parallel due to the mechanism's nature. Vectors $\boldsymbol{v}$ referenced in frames $\{B\}$ and $\{P\}$ are denoted by ${}^{B}\boldsymbol{v}$ and ${}^{P}\boldsymbol{v}$. Coordinate frame $\{P\}$ is attached to the mobile platform in a way that may later offer advantages for platform manufacturing and calibration of the machine.

Vectors defined by the machine's parameters:

- The position vectors of the midpoints P_i between joint centres at the mobile platform are defined in the frame $\{P\}$ as ${}^{P}\boldsymbol{p}_{pi}$, $i = 1,2,3$, where ${}^{P}\boldsymbol{p}_{p1} = [c_4, d, z_{p2}]^T$, ${}^{P}\boldsymbol{p}_{p2} = [0, 0, z_{p2}]^T$, ${}^{P}\boldsymbol{p}_{p3} = [x_{p3}, 0, z_{p3}]^T$;
- The position vector of the tool tip is defined in the frame $\{P\}$ as ${}^{P}\boldsymbol{p}_T = [x_T, y_T, z_T]^T$;
- The position vectors of driving axes reference points O_i and the position vectors of the midpoints of joint centres on the sliders are defined in the coordinate frame $\{B\}$ as ${}^{B}\boldsymbol{p}_{0i}$ and ${}^{B}\boldsymbol{d}_i$, $i = 1,2,3$, where ${}^{B}\boldsymbol{p}_{01} = {}^{B}\boldsymbol{p}_{02} = 0$, ${}^{B}\boldsymbol{p}_{03} = [0, -y_{03}, z_{03}]^T$, ${}^{B}\boldsymbol{d}_1 = {}^{B}\boldsymbol{d}_3 = 0$ and ${}^{B}\boldsymbol{d}_2 = [0, -d, 0]^T$.

Joint coordinates vector:

- $\boldsymbol{p} = [p_1, p_2, p_3]$, where $p_i, i = 1,2,3$ are scalar variables controlled by actuators while ${}^{B}\boldsymbol{a}_i = [1,0,0]^T$, $i = 1,2,3$ are unit vectors.

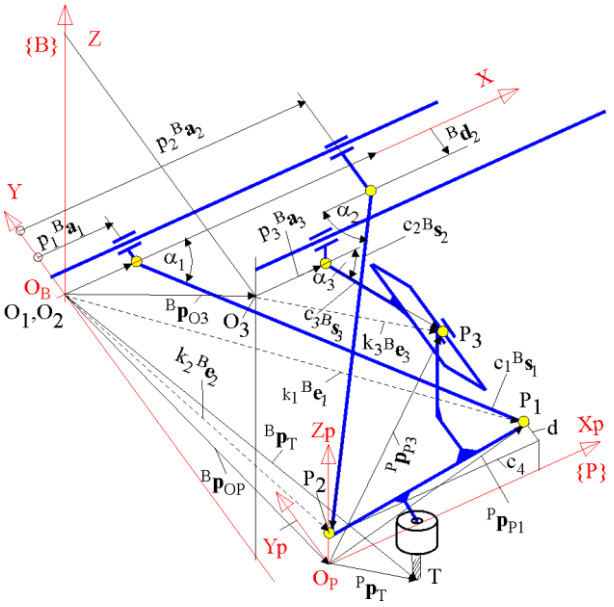

Figure 12. Kinematic model of parallel mechanism

World coordinates vector:

- ${}^{B}\boldsymbol{p}_T = [x_T, y_T, z_T]^T = \boldsymbol{x}$ represents the position vector of the tool tip, while ${}^{B}\boldsymbol{p}_{OP} = [x_p, y_p, z_p]^T$ represents location of the platform, i.e. origin O_P of the coordinate frame $\{P\}$ attached to it. The relationship between these two vectors is obvious since coordinates frames $\{B\}$ and $\{P\}$ are always mutually parallel, i.e.

$$ {}^{B}\boldsymbol{p}_T = {}^{B}\boldsymbol{p}_{O_P} + {}^{P}\boldsymbol{p}_T, \qquad (1) $$

Other vectors and parameters are defined as shown in Fig. 12, where ${}^{B}\boldsymbol{w}_i$ and ${}^{B}\boldsymbol{z}_i$, $i = 1,2,3$ are unit vectors while $c_i, i = 1,2,3$ are fixed lengths of joint parallelograms [13].

Based on geometric relations shown in Fig.12, the following equations are derived:

$$ k_i {}^{B}\boldsymbol{w}_i = {}^{B}\boldsymbol{p}_{OP} + {}^{P=B}\boldsymbol{p}_{pi} - {}^{B}\boldsymbol{p}_{0i} $$

$$ k_i {}^{B}\boldsymbol{w}_i = p_i {}^{B}\boldsymbol{a}_i + {}^{B}\boldsymbol{d}_i - c_i {}^{B}\boldsymbol{z}_i, \qquad (2) $$

As the vectors ${}^{B}\boldsymbol{a}$ and ${}^{B}\boldsymbol{d}_i$ are orthogonal to each other if the square of both sides in Eq. (3) is taken, the following relation is derived:

$$ c_i^2 = p_i^2 - 2p_i({}^{B}\boldsymbol{a}_i k_i {}^{B}\boldsymbol{w}_i) + (k_i {}^{B}\boldsymbol{w}_i - {}^{B}\boldsymbol{d}_i)^2, \qquad (3) $$

By substituting the machine's parameters in Eq. (4), the system of the following three equations is obtained:

$$p_1^2 - 2p_1(x_T - x_{TP} + c_4) + (x_T - x_{TP} + c_4)^2 + (y_T - y_{TP} + d)^2 + \left(z_T - z_{TP} + z_{p2}\right)^2 - c_1^2 = 0, \quad (5)$$

$$p_2^2 - 2p_2(x_T - x_{TP}) + (x_T - x_{TP})^2 + (y_T - y_{TP} + d)^2 + \left(z_T - z_{TP} + z_{p2}\right)^2 - c_2^2 = 0, \quad (6)$$

$$p_3^2 - 2p_3\left(x_T - x_{TP} + x_{p3}\right) + \left(x_T - x_{TP} + x_{p3}\right)^2 + \left(z_T - z_{TP} + z_{p3} - z_{03}\right)^2 - c_3^2 = 0, \quad (7)$$

from which are derived:

- inverse kinematics equations as

$$p_1^2 = x_T - x_{TP} + c_4 - \sqrt{c_1^2 - (y_T - y_{TP} + d)^2 - \left(z_T - z_{TP} + z_{p2}\right)^2}, \quad (8)$$

as well as

- direct kinematics equations as

$$x_T = x_{TP} + \frac{p_2^2 + c_1^2 - c_2^2 - (p_1 - c_4)^2}{2(p_2 - p_1 + c_4)}, \quad (9)$$

$$z_T = z_{TP} + z_{03} - z_{p3} - \sqrt{c_3^2 - (p_3 - \left(x_T - x_{TP} + x_{p3}\right))^2}, \quad (10)$$

$$y_T = y_{TP} - d - \sqrt{c_2^2 - (p_2 - x_T + x_{TP})^2 - \left(z_T - z_{TP} + z_{p2}\right)^2}, \quad (11)$$

The solution of the direct kinematics problem for the machine general case in explicit form is impossible, although this may be achieved, as shown, by the suitable selection of machine's parameters in order to simplify nominal geometric model [11, 12].

Bringing the machine to reference position was preceded by integrating the inverse and direct kinematics equations into the control system. The main objective of the experimental verification is to test the performance of the control system prototype. A PTC Creo CAD/CAM system was used for generating G-code of the programme and the idea was to programme the vertical parallel

Figure 13. Machined test pieces

kinematics milling machine in the same way as the conventional vertical milling machine. Prior to machining, the programme was first compiled and tested in an off-line system. Figure 13 shows CAD model and the first machined test piece.

5. CONCLUSIONS

This paper presents a control system designed for controlling PKM. Advantages have been confirmed of the control system that encapsulates the kinematics of parallel machine, which enables programming using G-code processed for conventional machining. It has been pointed out that it is necessary to have an off-line system for programming and checking the programming code in order to prevent breakdown situations in PKM. The software system designed as platform-independent provides the possibility for it to be integrated in an open-architecture hardware platform in future implementations.

ACKNOWLEDGMENT

The authors would like to thank the Ministry of Education, Science and Technological Development of the Republic of Serbia for providing financial support that made this work possible.

REFERENCES

[1] B. Arthaya, A. Setiawan, S. Sunardi: The design and development of G-code checker and cutting simulator for CNC turning operation, Journal of Mechanical Engineering Research Vol. 2, No. 3, pp. 58–70, 2010.

[2] H. Ji, Y. Li, J. Wang: A software oriented CNC system based on Linux/RTLinux, The international journal of advanced manufacturing technology, Vol. 39, pp. 291–301, 2008.

[3] T. Staroveski, D. Brezak, T. Udiljak, D. Majetic: Implementation of a Linux-based CNC open control system, in: *Proceedings of the 12th International Scientific Conference on Production Engineering – CIM2009*, 2009, Biograd, Croatia, Croatian Association of Production Engineering, Zagreb, pp 210–216.

[4] S. Park, S.H. Kim, H. Cho: Kernel software for efficiently building, re-configuring, and distributing an open CNC controller, The international journal of advanced manufacturing technology, Vol. 27, pp. 788–796, 2006.

[5] Y. Koren, S. Kota: *Reconfigurable Machine Tool.* US Patent 5 943 75, 1997.

[6] M. Weck, D. Staimer: Parallel Kinematic Machine Tools – Current State and Future Potentials. CIRP Annals – Manufacturing Technology, Vol. 51, No. 2, pp.671-683, 2002.

[7] J-P. Merlet, F. Pierrot: *Modeling of Parallel Robots. In Modeling, Performance Analysis and Control of Robot Manipulators*. London UK: ISTE Ltd, 2007.

[8] A. Aho, R. Sethi, J. Ullman: *Compilers Principle, Techniques and Tools*, Addison Wesley, 1986.

[9] H.P. Brinch: *Brinch Hansen on Pascal Compilers,* New Jersey, Prentice-Hall Inc, 1985.

[10] S-H. Suh, S-K. Kang, D-H. Chung, I. Stroud: *Interpolator. In: Theory and Design of CNC Systems*, pp. 69–106. Springer-Verlag London Limited, 2008.

[11] D. Milutinovic, M. Glavonjic, V. Kvrgic, S. Zivanovic: A New 3-DOF Spatial Parallel Mechanism for Milling Machines with Long X Travel. Annals of the CIRP, Vol. 54, No.1, pp.345–348, 2005.

[12] M. Glavonjic, D. Milutinovic: Parallel structured milling machines with long X travel. Robotics and Computer-Integrated Manufacturing Vol. 24, pp. 310–320. 2008.

[13] T. Arai, T. Tanikawa, J-P. Merlet, T. Sendai: Development of a new parallel manipulator with fixed linear actuator, in: *Proceedings of the Japan/USA Symposium on Flexible Automation (ASME 1996),* pp. 145–149, 1996.