FLEXIBLE CELL CONTROL IN "OPEN CIM SCREEN"

Petar Jakovljević[1]*,
[0009-0006-8002-3326]
Miloš Vujošević[1],
[0009-0001-3672-3302]
Đorđe Dihovični[2],
[0000-0003-0961-2540]
Nada Ratković Kovačević[2]
[0000-0001-6398-4391]

[1]Polytechnic – school for new technologies, Belgrade, Serbia

[2]The Academy of Applied Technical Studies Belgrade, Belgrade, Serbia

Abstract:

This paper describes flexible cell control using subprograms made in SCORBASE software. Subprograms represent the operations for each device in the cell. The goal is to move the robot arm to the desired positions for manipulating the workpieces and machine tending. In the SCORBASE software, the positions of the robots are saved, after the positions are saved they are called in the program for cell control, and in between called positions the subprograms for cell device operations are called. One example can be a robot moving towards the machine, subprogram for opening the door of the machine is called, then robot moving inside of the machine workspace, subprogram for opening chuck device is called, robot places the part in the machine, chuck is closed, robot moves out of the machine workspace, machine door closes and the cycle for manufacturing starts. Before each operation status of both the robot and machine is checked to avoid disturbing the process. Each machine in the cell is connected to an I/O controller which sends signals to the machine to operate, subprograms for machine operations consist of variables that control each machine and each operation.

Keywords:

Flexible cell, Automation, Programming, Robot, Manufacturing.

INTRODUCTION

In a flexible cell, all components establish a network where signals travel to achieve certain operations. This flexible cell consists of two CNC machines Emco concept mill 55 and Emco concept turn 55, robot arm Scorbot ER4, pneumatic and gravity feeder, two computers (for part control (ID41) and cell control (ID43)) and a main cell controller [1],[2],[3]. Both computers (ID41 and ID43) use SCORBASE software for whole-cell control and robot movement. ID41 computer has its program for controlling how many parts are tended and manufactured. Each machine has its control unit where CNC programs are loaded for manufacturing the workpieces and a controller that is responsible for sending signals to the machines to perform operations that would otherwise be operated manually [4],[5],[6]. Machines are equipped with a pneumatic system for door and clamping device control, a pneumatic feeder is also connected to the pneumatic system [7].

Correspondence:
Petar Jakovljević

e-mail:
pjakovljevic51@gmail.com

DOI: 10.15308/Sinteza-2024-191-198

The idea is to make subprograms for each operation of the cell and combine created subprograms with the robot movement program, each time the robot finishes the movement or certain task the subprogram for another operation is called [8]. For example, the robot moves to the feeder where the workpiece is held, the feeder is activated using a subprogram to push or detect the workpiece depending on which feeder is used, the robot picks the workpiece and moves to the machine, the machine door is open using a subprogram to activate the controller on the machine, the robot enters the workspace of the machine and the clamping device is open using the subprogram for activating the controller, the robot places the part into the machine and moves out of its workspace [9],[10].

2. FLEXIBLE CELL CONTROL

To start up the system it is necessary to turn on the PCs ID41 and ID43, control units for CNC Mill and CNC Turn. An air compressor is activated to bring pressured air to the machine's clamping devices to hold the workpiece and to the pneumatic feeder to move the Milling parts to the pickup position. CNC machines and the robot are activated and their control unit software, the robot is brought to its home position, machines are referenced.

To describe how this flexible cell is working it is necessary to develop certain algorithms which will represent the whole control of the cell (Figure 1).

The first algorithm shows that to activate the cell operations the script file FMS.VBS is loaded into the computer ID43 which contains the program for machine and peripheral device operations. After loading the script file the number of machine workpiece parts is set for each machine on computer ID41. Algorithm 1 is implemented as a code in the provided example (Listing 1):

The initialization program code is loaded to set variables for machine operations. The initialization program is used to prepare the flexible cell for the operations. Robot status is checked if the robot is brought to the home position using the computer ID43. The variables used to describe the machines are M for Mill and T for Turn. A counter is added to check how many parts the machines have produced and variables describing are the machines ready to receive or take out the finished part. The initialization program code example is provided in the example (Listing 2).

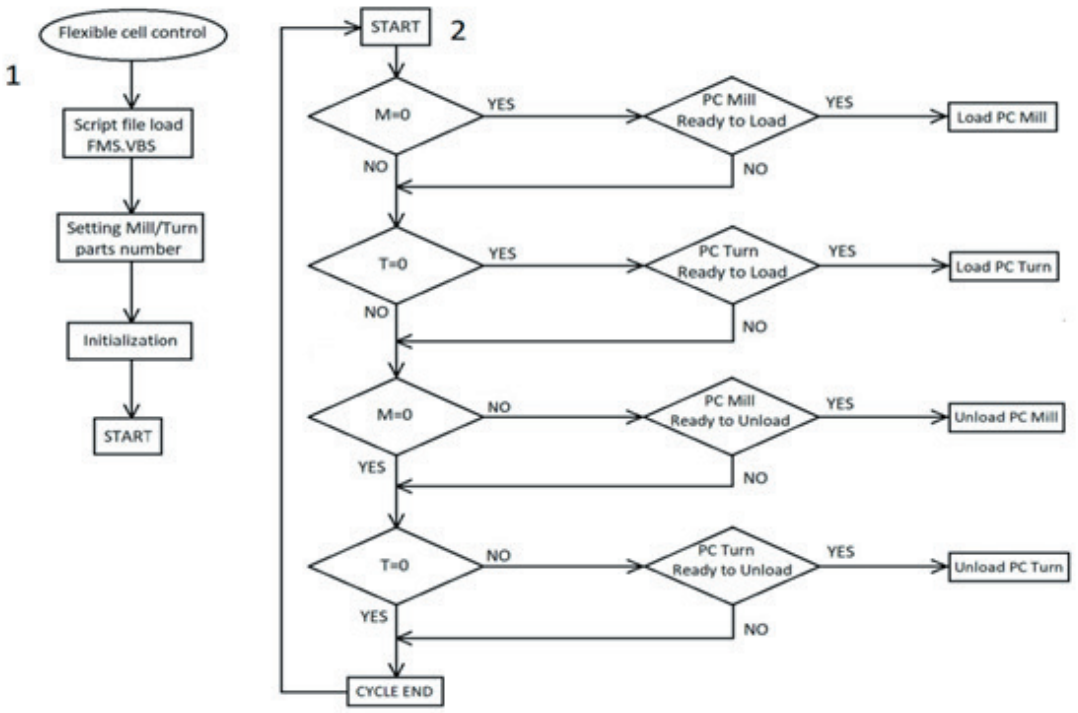

Figure 1. Algorithm of flexible cell control.

```
Set Variable M = SCRIPT.GET_NUMBER_MILL
Set Variable T = SCRIPT.GET_NUMBER_LATHE
Print to Screen: MILL Parts='M', LATHE Parts= 'T'
```

Listing 1. Program for entering the number of manufactured parts.

```
Set Variable ROBOT_STATUS = 0
Send Message RUN INITC to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Set Variable TOTAL_ORDER_STATUS = 0
Set Variable MILL_ORDER = M
Set Variable TURN_ORDER = T
Set Variable MILL_COUNTER = 1
Set Variable TURN_COUNTER = 1
Set Variable PCMILL55_READY_TO_LOAD = 1
Set Variable PCTURN55_READY_TO_LOAD = 1
Set Variable PCMILL55_READY_TO_UNLOAD = 0
Set Variable PCTURN55_READY_TO_UNLOAD = 0
Print to Screen & Log: START OF NEW ORDER
Print to Screen & Log: MILL ORDER = 'MILL_ORDER'
Print to Screen & Log: TURN ORDER = 'TURN_ORDER'
Set Variable STATUS = M + T
```

Listing 2. Program example of Initialization.

The second algorithm represents machine check if each machine has one or less than one workpiece set in the machine:

Are there 0 parts in the CNC Mill:

- Answer YES: Algorithm Checks is CNC Mill ready to load;
- Answer NO: Algorithm checks are there 0 parts in the CNC Turn.

Is CNC Mill ready to Load:

- Answer YES: Execute program to load CNC Mill;
- Answer NO: Algorithm checks are there 0 parts in the CNC Turn.

Are there 0 parts in the CNC Turn:

- Answer YES: Algorithm checks is CNC Turn ready to load;
- Answer NO: Algorithm checks are there 0 parts in the CNC Mill.

Is CNC Turn ready to load:

- Answer YES: Execute program to load CNC Turn;
- Answer NO: Algorithm checks are there 0 parts in the CNC Mill.

Are there 0 parts in the CNC Mill:

- Answer YES: Algorithm checks are there 0 parts in the CNC Turn;
- Answer NO: Algorithm Checks is CNC Mill ready to load.

Is CNC Mill ready to unload:

- Answer YES: Execute program to unload CNC Mill;
- Answer NO: Algorithm checks are there 0 parts in the CNC Turn.

Are there 0 parts in the CNC Turn:

- Answer YES: Cycle ends;
- Answer NO: Algorithm Checks is CNC Turn ready to unload.

Is CNC Turn ready to unload:

- Answer YES: Execute program to unload CNC Turn;
- Answer NO: Cycle ends.

3. ACTIVATING MACHINE OPERATIONS USING I/O CONTROLLER FUNCTIONS

I/O controllers are sending signals to the machine to complete different operations, to be activated the script file is loaded into the ID43 PC for cell control. The program code is provided in the example (Listing 3):

The numbers that are shown describe which machine is performing which operation. The first number represents the machine (1 for Mill, 2 for Turn), and the second number represents the operation (0 = door open, 1 = door closed, 2 = clamping open, 3 = clamping closed). The third number is the bit activation, 1 is for activation of the controller, after the operation is complete the controller must be set to 0 to be ready for a second operation.

Since the robot controller does not provide sufficient I/O for the complete operation of all automation functions USB I/O interface is used to provide the required interface for the machine door and clamping device function. Program code for the machine operations is provided in the example (Listing 4).

```
Load script file: USBIO.VBS
Set Variable RET = SCRIPT.SETOUTPUTSB(1,0,0)
Set Variable RET = SCRIPT.SETOUTPUTSB(1,1,0)
Set Variable RET = SCRIPT.SETOUTPUTSB(1,2,0)
Set Variable RET = SCRIPT.SETOUTPUTSB(1,3,0)
Set Variable RET = SCRIPT.SETOUTPUTSB(2,0,0)
Set Variable RET = SCRIPT.SETOUTPUTSB(2,1,0)
Set Variable RET = SCRIPT.SETOUTPUTSB(2,2,0)
Set Variable RET = SCRIPT.SETOUTBUTSB(2,3,0)
```

Listing 3. Program example of setting the controller status for each operation.

Table 1. Controller input activation.

Input	Type	Description
PA0	PNP	PC Mill/Turn Open Door
PA1	PNP	PC Mill/Turn Close Door
PA2	PNP	PC Mill/Turn Open Clamping
PA3	PNP	PC Mill/Turn Close Clamping

Table 2. Controller output activation.

Output	Type	Description
K0	Relay	PC Mill/Turn Open Door
K1	Relay	PC Mill/Turn Close Door
K2	Relay	PC Mill/Turn Open Clamping
K3	Relay	PC Mill/Turn Close Clamping

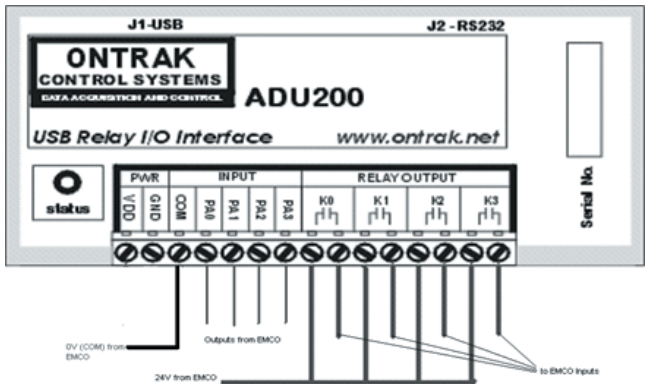

Figure 2. Controller used for machine operations.

```
Set Subroutine OPEN PCMILL55 DOOR
Print to Screen: SET OUTPUT TO OPEN PCMILL55 DOOR
Set Variable RET = SCRIPT.SETOUTPUTSB(1,0,1)
If RET == 1 Jump to NEXT
Call Subroutine ERROR_HANDLER
End
NEXT:
Wait 20 (10ths of seconds)
Set Variable RET = SCRIPT.SETOUTPUTSB(1,0,0)
Print to Screen: RESET OUTPUT TO OPEN PCMILL55 DOOR
Set Variable IO_BOX_NUMBER = 1
Set Variable INPUT_NUMBER = 0
Set Variable WAIT_TIME_SEC = 600
Set Variable EXPECTED_STATUS = 1
Call Subroutine WAIT_INPUT
Print to Screen: PCMILL55 DOOR OPEN
Return from Subroutine
```

Listing 4. Program for opening CNC Mill door.

Table 3. Variable differences for each machine operation.

Operation	Variable
Close Mill Door	SCRIPT.SETOUTPUTSB (1,1,1)
Open Mill Clamping	SCRIPT.SETOUTPUTSB (1,2,1)
Close Mill Clamping	SCRIPT.SETOUTPUTSB (1,3,1)
Open Turn Door	SCRIPT.SETOUTPUTSB (2,0,1)
Close Turn Door	SCRIPT.SETOUTPUTSB (2,1,1)
Open Turn Clamping	SCRIPT.SETOUTPUTSB (2,2,1)
Close Turn Clamping	SCRIPT.SETOUTPUTSB (2,3,1)

Table 4. Main controller input operations.

Input	Type	Description
1	LOW	PC Mill Cycle Status
2	LOW	PC Turn Cycle Status
3	LOW	Not in use
4	LOW	Not in use
5	LOW	Pneumatic feeder sensor
6	LOW	Gravity feeder sensor
7	LOW	Not in use
8	LOW	Not in use

Table 5. Main controller output operations.

Output	Type	Description
1	Relay	PC Mill Cycle Start
2	Relay	PC Turn Cycle Start
3	Relay	Not in use
4	Relay	Not in use
5	Source	Pneumatic feeder activation
6	SINK	Not in use
7	SINK	Not in use
8	SINK	Not in use

Program codes for other machine operations are similar so only the changes for the controller will be shown using tables 1 and 2.

The robot controller is used for starting up the machine cycles, for activating part feeders, and for robot control. The gravity feeder is using the piston to push milling workpieces to the pickup position.

The servo valve is activated and transfers pressurized air, extending the piston. The gravity feeder only has a micro switch identifying when the turning part is placed in the pickup position.

Program code for activating the pneumatic feeder is provided in the example (Listing 5).

```
Set Subroutine EXTEND_PISTON
Turn On Output 5
Wait 20 (10ths of seconds)
Return from Subroutine
*****************************************************************************************************
Set Subroutine RETRACT_PISTON
Turn Off Output 5
Wait 20 (10ths of seconds)
Return from Subroutine
```

Listing 5. Program for activating pneumatic feeder.

```
Set Subroutine LOAD PCMILL55
Set Variable ROBOT_STATUS = 0
Send Message RUN GET_PART_FROM_MILL_FEEDER to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Set Variable ROBOT_STATUS = 0
Send Message RUN PUT PART INTO PCMILL55 to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Return from Subroutine
*****************************************************************************************************
Set Subroutine LOAD PCTURN55
Set Variable ROBOT_STATUS = 0
Send Message RUN GET_PART_FROM_LATHE_FEEDER to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Set Variable ROBOT_STATUS = 0
Send Message RUN PUT PART INTO PCTURN55 to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Return from Subroutine
```

Listing 6. Programs for loading CNC Mill and CNC Turn.

```
Set Subroutine UNLOAD PCMILL55
Send Message SET INDEX_RACK_1='MILL_COUNTER' to Robot Device Driver ID=43
Set Variable ROBOT_STATUS = 0
Send Message RUN GET PART FROM PCMILL55 to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Set Variable ROBOT_STATUS = 0
Send Message RUN PUT_PART_TO_MILL_RACK to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Return from Subroutine
*****************************************************************************************************
Set Subroutine UNLOAD PCTURN55
Send Message SET INDEX_RACK_2='TURN_COUNTER' to Robot Device Driver ID=43
Set Variable ROBOT_STATUS = 0
Send Message RUN GET PART FROM PCTURN55 to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Set Variable ROBOT_STATUS = 0
Send Message RUN PUT_PART_TO_LATHE_RACK to Robot Device Driver ID=43
Call Subroutine CHECK_ROBOT_STATUS
Return from Subroutine
```

Listing 7. Programs for unloading CNC Mill and CNC Turn.

4. LOADING AND UNLOADING THE MACHINES

Movement instructions are given to the robot arm to reach certain positions for picking and placing workpieces. Each workpiece is located on its feeder: A pneumatic feeder for milling parts and a gravity feeder for turning parts. Subprograms for loading and unloading the machines are made which contain the operations of the machines and the robot positions and operations. Sub programs are provided in the examples (Listing 6) and (Listing 7).

Program RUN GET_PART_FROM_MILL/LATHE_FEEDER to Robot Device Driver ID=43 is activating saved robot positions for picking the workpiece from the part feeder for the desired machine, the positions for picking the workpiece from the feeder are: Approaching part feeders, approaching workpieces, picking the workpieces, retracting from the feeders.

Program RUN PUT PART INTO PCMILL55/TURN55 to Robot Device Driver ID=43 is activating saved robot positions for placing the workpiece in the machine, the positions for placing the workpiece in the machine are: Approaching the machine, placing the workpiece in the machine, retracting from the machine.

Program RUN GET PART FROM PCMILL/TURN55 to Robot Device Driver ID=43 is activating saved robot positions for removing the work piece from the machine, the positions for removing the work piece from the machine are: Approaching the machine, picking manufactured work piece, retracting from the machine,

Program RUN PUT_PART_TO_MILL/LATHE_RACK to Robot Device Driver ID=43 is activating saved robot positions for placing the manufactured part on the pallet, the positions for placing the manufactured work piece on the pallet are: Approaching the pallet, placing the finished part on the pallet, retracting from the pallet.

5. CONCLUSION

Creating subprograms for each operation in the cell and calling them in between the program for robot operations is one way of making the program easier to read and check for errors. Once the subprograms are made they can be further used in new in the described cell with only changing some small parts to achieve synchronization. Problems that were encountered are errors in the machine door and clamping device operations due to lack of pressurized air.

The solution was to increase waiting time to reduce the consumption of pressurized air and to reduce the flow throughout the hoses. Decreasing the wait time however, can cause the collision of the robot and the machine, and pressure forming in the pneumatic system can cause the hose to rupture and the pressurized air supply would be drastically reduced which would fail to perform machine operations, and pneumatic feeder piston control.

To achieve a better arrangement of the connection between the machines and the controllers each input and output was connected with different wire colors if suddenly some of the wires were disconnected it would make reconnecting the wires simpler and it would save time. Wires connecting the main controller are tied up in one plastic holder to avoid cable mixing and damage.

6. REFERENCES

[1] P. Jakovljevic, “Role and application of industrial robots in education,” M.S. thesis, Dept. Comput. Mech. Eng., The Academy of Applied Technical Studies, Belgrade, Serbia, 2022.

[2] P. Jakovljević, Đ. Dihovični, N. Ratković Kovačević, “Robot Movement Programming for Flexible Cell In "Open Cim Screen"," *Sinteza 2023 - International Scientific Conference on Information Technology, Computer Science, and Data Scienc*e, Belgrade, Singidunum University, Serbia, 2023, pp. 235-241. doi: 10.15308/Sinteza-2023-235-241

[3] T. Saric, Robot Programming, Accessed: Apr. 10, 2024. [Online], Available: https://dokumen.tips/documents/programiranje-robota.html

[4] Intelitek , Derry, NH, USA. Manual SCORBASE version 7 and higher for SCORBOT-ER 4U SCORBOT-ER 2U ER-400 AGV mobile robot, Catalog No. 100342 (2016). Accessed: Apr. 12, 2024. [Online]. Avaliable: https://downloads.intelitek.com/Manuals/Robotics/ER-4u/Scorbase_USB_I.pdf

[5] Intelitek, Derry, NH, USA. Manual on RoboCell for Motoman MHJF, Catalog #34-8000-0012 (2018). Accessed: Apr. 12, 2024. [Online]. Available: https://www.intelitekdownloads.com/Manuals/Robotics/MHJF/RoboCell_for_MHJF_v10.1_user_manual_revC.pdf

[6] Intelitek, Derry, NH, USA. Controller-USB User Manual, Catalog No. 100341, Accessed: Apr. 12, 2024. [Online]. Available: https://downloads.intelitek.com/Manuals/Robotics/ER-4u/Controller-USB-H.pdf

[7] L. Wang, "The Development of Flexible Manufacturing Systems in Automotive Production Under the Background of Industry 4.0: A Descriptive Study," *Academic Journal of Science and Technology,* vol. 8, no. 3., pp. 29–31, Dec. 28, 2023. doi: 10.54097/m2qyte40.

[8] Al Fahim, M. Mizanur Rahman, M. W. Hridoy, and K. R. Uddin, "Development of a PLC Based Automation Cell for Industry," *Journal of Integrated and Advanced Engineering (JIAE),* vol. 3, no. 2., pp. 87–100, Sep. 03, 2023. doi: 10.51662/jiae.v3i2.94.

[9] B. Wallner, B. Zwölfer, T. Trautner, and F. Bleicher, "Digital Twin Development and Operation of a Flexible Manufacturing Cell using ISO 23247," *Procedia CIRP*, vol. 120., pp. 1149–1154, 2023. doi: 10.1016/j.procir.2023.09.140.

[10] D. Mourtzis, J. Angelopoulos, and G. Dimitrakopoulos, "Design and development of a flexible manufacturing cell in the concept of learning factory paradigm for the education of generation 4.0 engineers," *Procedia Manufacturing,* vol. 45., pp. 361–366, 2020. doi: 10.1016/j.promfg.2020.04.035.