ZESZYTY NAUKOWE POLITECHNIKI ŁĄSKIEJ 1992
Seria: MECHANIKA z. 108 Nr kol. 1161

International Conference on
COMPUTER INTEGRATED MANUFACTURING
Internationale Konferenz über
RECHNERINTEGRIERTE FERTIGUNGSSYSTEME
Zakopane, March 24-27 1992

Janusz MADEJSKI

The Institute of Physical Metallurgy
Silesian Technical University, Gliwice, Poland

Ryszard ZDANOWICZ

The Institute of Machine Technology
Silesian Technical University, Gliwice, Poland

COMPUTER SUPERVISORY CONTROL OF SMALL ROBOTIZED MANUFACTURING SYSTEM

Abstract. The paper presents the logical layout of the overall control of the small manufacturing system. Assignment of control tasks of separate elements of the system top down to the application software level is presented. Specific control tasks are discussed and computer control system is described in the form of program graphs. New technological documentation - which enables the operator to switch from task to task quickly and error free is presented.

1.Introduction

Several years practice on hands-on job-shop level connected with designing and implementing of robotized manufacturing systems and many case studies have led us to some general conclusions valid for small robotized manufacturing systems [1,2,3], namely:

- all basic control tasks should be distributed to autonomous system elements,
- a system may consist of so many workcells as can be tended by an operator,
- workcells should be able to operate independently - thus being "assigned" to separate system layers,
- the bottom system level forming any of the layers consists of a set of

  - machine tools (automatic assembly devices, etc.),
  - quality inspection stands (ideally at least input and output ones [4]),
  - transfer system - in this case the least expensive and most versatile unit is an industrial robot (there may be some of them in one workcell),

- the intermediate system level is an overall system computer control - all signals between any of the system elements in any workcell should be handled by this level as the appropriate actions may be taken only having the whole system in the scope of view,
- the top system level is an operator as - still - a most flexible, intelligent and versatile system element - Fig.1.

2. Specific control tasks of the robotized system elements

Exemplary control algorithms for various system elements are shown in Fig.2 - note that an operator is being treated as one of the system elements. All information interchange takes place through the PC based computer control unit. One of the system elements - a robot, as a freely programmable device - may carry out complicated control and manipulation tasks by itself - but to make any modifications of its program easier strict programming rules should be observed:

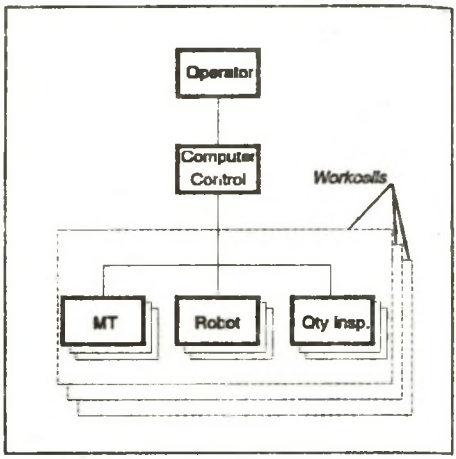


Figure 1 Logical layout of the small manufacturing system elements

- robot program must be fully structured employing all available robot program flow control programming means - by this we mean separate programs or at least subroutines for tending system elements,

- all robot/workcell settings must be fully documented - it is best to follow a uniform procedure in all systems. One of the exemplary technological documentation form is shown in Table 1[1]. Note the robot program flow graph - simply inevitable at the workcell setting up stage. The I/O signals' names reflect in each case real connections.

- in most cases autonomous task dedicated elements like machine tools' or quality inspection stands' programs may be treated as simple logical units each with one entry and return points with a logically simple although sometimes lengthy "do-it" in-between.

---

[1] Legend to **Table 1** abbreviations

Rb	- robot	**S**	- batch
IM_n	- input magazine *n*	**NP, SP, Δ**	- used in quality inspection stand N/A here
OM	- output magazine		
MT_n	- machine tool *n*	**N**	- tools
R_n	- reorientation stand	*cat. no.*	- tool id number
IM	- intermediate magazin	*n*	- tool has to be changed every *n* machining operation

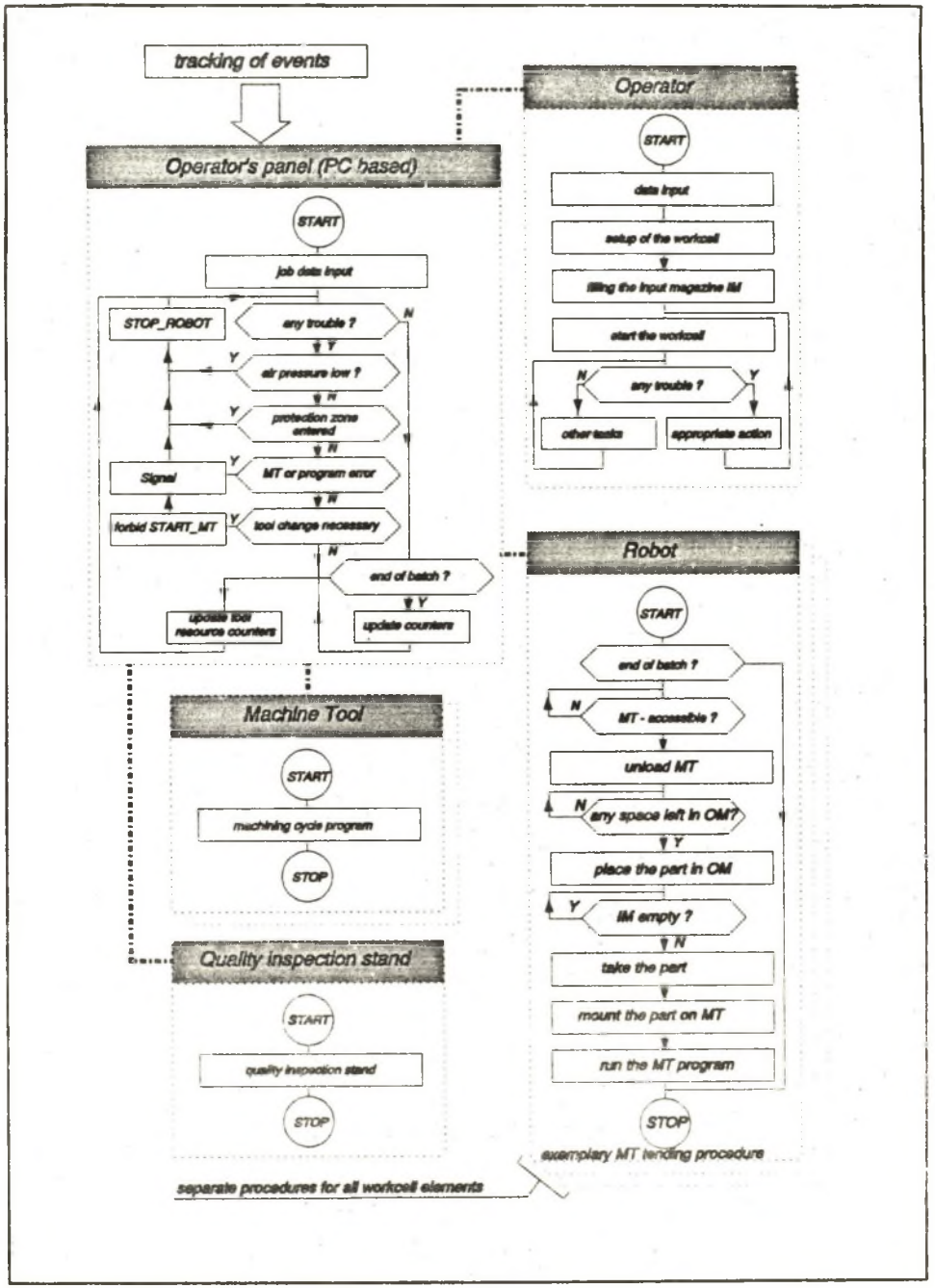


Fig.2 Interconnections of the control tasks

3. Computer control system

A computer control program flow chart shown in Fig.3 contains all the essential tasks carried out by the system. Its basic task is to scan the system state in an endless loop. Scanning is a way to gather all important data for:

- updating the system work log,
- analysis of the system state (i.e. tracking of the alarm conditions including END_OF_CYCLE for MT*n* elements, TOOL_CHANGE demands, updating tool resource counters, etc.)
- taking of program decisions (i.e. forbid to START_MT*n* if a tool has not been changed on demand when necessary, there may occur a situation when STOP_OPERATION condition is evaluated and all the system data is stored in the disk file for further use, etc.),
- interface procedures for operator-system dialogue.

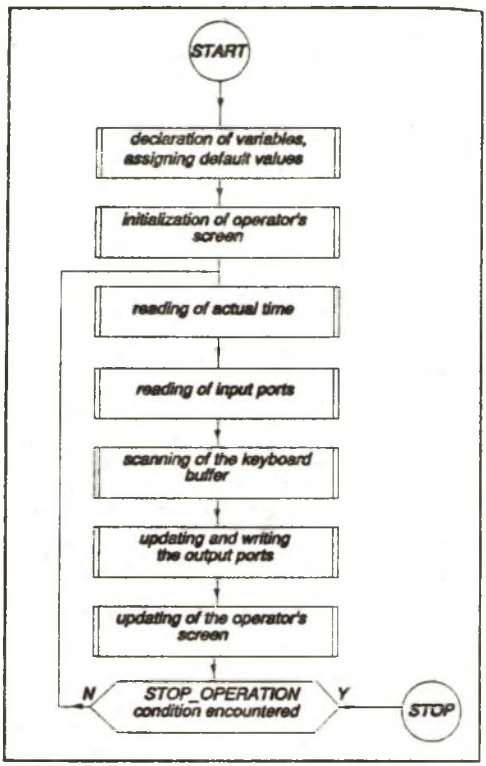


Fig.3 Flow chart of the system computer control

Fig.4 ÷ 7 show some important fragments of the control software. Note the links between the port 0x280 bit fields and the robot I/O signals in Table 1. This net of connections forms a basic hardware level upon which all the software may be developed. After defining the meaning of input ports' bit fields it is possible to evaluate the values of the system state variables. On the lower level it is useful to do this using versatile procedures like `var_st()` being called in the fig.5 program fragment.

Proper decisions are reached after some considerations including scanning many input signals (Fig.6), some of the decisions are reflected as an updated screen data, others are sent to the output port as well (Fig.7). The computer keyboard buffer is also treated as an ordinary input circuit so the decisions employing this data are being made in the same way as it has been described for the input port data. Thanks to this solution - as we have tested for a one workcell system - at least (depending on the complexity of the system) every 20ms each input signal is being scanned by the software. As systems like this are not response time dependent we can state that control system time lag up to 1 ÷ 2 secs would be still satisfactory.

```
/* bit fields ------------------------------------------------------*/

struct Port280  {unsigned STOP_POT_01   :1;      MT1 program stop
                 unsigned awaria_01     :1;      MT1 breakdown
                 unsigned praca_aut_01  :1;      MT1 automatic operation
                 unsigned drzwi_otw_01  :1;      MT1 door open
                 unsigned drzwi_zmk_01  :1;      MT1 door closed
                 unsigned uchwyt_zmk_01 :1;      MT1 chuck closed
                 unsigned hydraulika_01 :1;      MT1 hydraulic circuit O.K.
                 unsigned chlodziwo_01  :1;      MT1 coolant O.K.
                 int                    :8;      - bits not available
                ) ;/* 1 input card */

struct Port281  {unsigned R_START_POT_01 :1;     Robot demands start of MT1
                 unsigned R_uchwyt_01   :1;      Robot flip-flops MT1 chuck
                 unsigned R_alarm_01    :1;      Robot signals MT1 alarm cond.
                 unsigned R_koc_ckl_01  :1;      Robot reports MT1 EOC
                 unsigned R_START_POT_02 :1;     Robot demands start of MT2
                 unsigned R_uchwyt_02   :1;      Robot flip-flops MT2 chuck
                 unsigned R_alarm_02    :1;      Robot signals MT2 alarm cond.
                 unsigned R_koc_ckl_02  :1;      Robot reports MT2 EOC
                 int                    :8;      - bits not available
                ) ;/* 1 input card */           (EOC : end-of-cycle)
```

Fig.4 Example of bit fields declaration

```
/*********------------* port 0x280 *------* MT1 *------------*/   Computer checks step-by-step all
        p280 = inportb(0x280);                                    the incoming signals reading data
        p0   = &p280;                                             from ports 0x280 through 0x283.

        /*--- checking the m00 or m30 function ---------------*/
        01_STOP_POT = p0->STOP_POT_01;

        /*--- as above - checking breakdown cond. ------------*/
        wp          = p0->awaria_01;                              This port brings information about
        ver_st(wp,&01_awaria,44,1,cir[wp],8,awaria[wp]);          completing of the machining
                                                                  operation, eventual alarm
        /*--- as above - checking automatic op. --------------*/ conditions that may be caused by
        wp          = p0->praca_aut_01;                           low hydraulic circuit pressure, low
        ver_st(wp,&01_aut,44,4,cir[wp+1],6,stan_maszyny[wp]);     coolant level, open MT door, etc...

        /*--- as above - checking door open cond. ------------*/
        01_d_otw    = p0->drzwi_otw_01;

        /*--- as above - maybe they are open? ----------------*/ It is also possible to actuate the
        wp          = p0->drzwi_zmk_01;                           machine tool chuck, computer
        ver_st(wp,&01_d_zmk,44,8,cir[wp+1],9,drzwi[wp]);          control logs its state...

        /*--- as above - checking MT1 chuck (closed?) --------*/
        wp          = p0->uchwyt_zmk_01;
        ver_st(wp,&01_uchwyt,44,5,cir[wp+1],9,uchwyt[wp]);

        /*--- as above - MT1 hydr. circuit pressure ----------*/
        wp          = p0->hydraulika_01;
        ver_st(wp,&01_hydraulika,44,8,cir[wp+1],9,poziom[wp]);

        /*--- as above - MT1 coolant level -------------------*/
        wp          = p0->chlodziwo_01;
        ver_st(wp,&01_chlodziwo,44,7,cir[wp+1],9,poziom[wp]);;

        /*--- as above - maybe Rbt wants to Op/Cl MT1 chuck-*/
        R_uchw_02       = p1->R_uchwyt_02;

        /*--- as above - maybe RBT signals Alarm cond.2 ----*/
        R_02_alarm      = p1->R_alarm_02;

        /*--- as above - maybe Rbt signals MT1 EOC  ----------*/
        wp = p1->R_koc_ckl_02;
```

Fig.5 Reading of the input port data, updating of the system state definition variables

```
/******** ******...........................* status analysis *---------------*/

/* 1. reaction on the coolant level and MT hydraulic circuit pressure     */

sgn = (01_chlodziwo==0||02_chlodziwo==0||01_hydraulika==0
     ||02_hydraulika==0)?1:0;

/* 2. sending an EOC (end-of-cycle) signal - KNC_CKL results in updating the
cutting tools resources' record (for a given MT) - after the updating the
system checks whether any one of them reaches the resource limit          */

      if(R_01_knc_ckl==1||R_02_knc_ckl==1)   /* EOC signal coming          */
      (                         /* subsequent reactions to this signal not
                                 possible before proper reaction from the robot*/

      rob =(R_01_knc_ckl==1)?(R_01_knc_ckl=2,1):(R_02_knc_ckl=2,2);

         chn = (rob-1)*2+1;
         pom = (rob-1)*20+44;

         status[chn-1][8] += 1;      /*  updating of the workpiece counter */
         corritoa(str,status[chn-1][8],3);
         writef(20,21+rob,LIGHTGRAY,3,str);
         }
                               ... etc.
```

All the incoming signals are read by the operator's panel computer and proper reactions are evaluated and executed. In this example it is assumed that the workcell consists of two machine tools. Note that all the workcell elements communicate with each other only through the operator's panel computer.

Fig.6 Analysis of the state of the workcell

```
/*********.........................* port 0x285 *--* ->Rb ready  *--------
--*/

        if (sgnl || !01_STOP_POT)        /* switch off READY signal
*/
                p5->GOT_OBSL_01=FALSE;

        if (sgnl || !02_STOP_POT)
                p5->GOT_OBSL_02=FALSE;

        if (01_d_otw != p5->drzwi_otw_01) /* door open message
*/
                p5->drzwi_otw_01;

        if (02_d_otw != p5->drzwi_otw_02)
                p5->drzwi_otw_02;

        if (!sgnl)  /* this may be done only when not ALARM cond. !!!
*/
        (
                if (01_STOP_POT && !p5->GOT_OBSL_01)
                p5->GOT_OBSL_01=TRUE;

                if (01_STOP_POT && !p5->GOT_OBSL_02)
                p5->GOT_OBSL_02=TRUE;

        )
        if(status[0][8]==status[1][7]&&!p5->knc_serii_01)
        p5->knc_serii_01=TRUE;
        if(status[0][8]!=status[1][7]&&p5->knc_serii_01)
        p5->knc_serii_01=FALSE;
        if(status[2][8]==status[3][7]&&!p5->knc_serii_02)
        p5->knc_serii_02=TRUE;
        if(status[2][8]!=status[3][7]&&p5->knc_serii_02)
        p5->knc_serii_02=FALSE;

        outportb(0x285,outport285);   /* write information to the port
*/
```

This is an example of writing information to one of the output ports. This information partly was copied from input circuits, and partly was prepared as decisions made by the software.

All necessary condition check-ing is done i.e. blocking the MT_START signals when MT door is open, advicing robot control unit about batch completing, etc.

Fig.7 Writing information to the output port

Table 1 Example of one of the workcell set-up documentation forms

workcell no	01	ROBOT PROGRAM INSTRUCTION CHART	page no	x
robot no	01		program no	1

MT program no	Machining instrctn	Setup instrctn	Offset settings	Gripper setup
35-XX7/23 (A V B)	35/XX7/23-i1 (A V B)	35/XX7/23-i2	35/XX7/23-i3 (A V B)	XX/7/23-g1

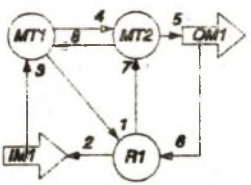
INITIAL STATE OF THE WORKCELL - SHAFT XX7/23

		OPERATOR'S PANEL DATA								
Rb	synchronization position	S = 60			NP= -					
IM1	full (check initial cut length ≥8mm)	SP = -			Δ = -					
IM2	-	N			1	2	3	4	5	6
OM1	empty	MT1	cat. no			12	14		17	
OM2	-		n			10	15		40	
MT1	door open, chuck open	MT2	cat. no		3	12		9		
MT2	door open, chuck open		n		10	10		35		
R1	empty									
R2	empty (used as an interm. storage-IM)	remarks			-					

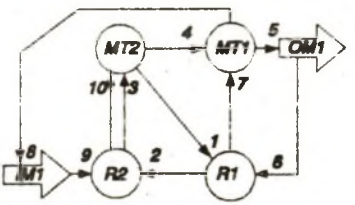
ROBOT I/O SIGNALS

	Input		Output		Memory flags
1	CONTACT_MT1	1	START_MT1	17	-
2	CONTACT_MT2	2	START_MT2	18	-
3	-	3	CHUCK_MT1	19	-
4	-	4	CHUCK_MT2	20	temporary storage before reorientation
5	-	5	-	21	startup of MT2 (1st machining operation)
6	-	6	-	22	machining of the 1st workpiece on MT1
7	IM not used?	7	-	23	door closed on MT2
8	-	8	-	24	door closed on MT1
9	-	9	-	25	IM used
10	IM1 full?	10	-	26	initial filling of IM
11	-	11	-	27	-
12	-	12	gripper change	28	-
13	MT1 door open?	13		29	-
14	MT2 door open?	14		30	-
15	MT1 - m00 \m30			31	-
16	MT2 - m00 \m30			32	-

program graphs



version A - IM not used
1st operation - MT2
2nd operation - MT1



version B - IM used

cassette id	setup time [h]	cycle time [min]	made	checked	approved
003-A,B	0.10	6.00	JM	RZ	LR

4. Conclusions

Experience gathered during the application of the robotized machining, welding and inspection workcells has enabled us to develop certain system software and hardware design procedures. Some of our experoence has been included in the presence itself and the form of the robot program flow graphs featuring a part of the 'Robot Program Instruction Chart' form. Numbered arrows show which tasks will be carried out by the robot following its program. It is the most convenient way to let the operator learn what is the sequence of events in a program he does not review before. In the version A program in the Table1 the robot takes a workpiece from the MT2 machine tool and leaves it in the reorientation stand R1 (note the bolded head of the arrow - it signifies that the robot performs its task loaded with the workpiece). Next the robot proceeds to the input magazine IM1 and feeds the MT1 machine tool with a new workpiece. By this time the machining on the MT2 should be over so the robot (unloaded) goes to the MT2 machine tool and takes a finished piece after the second operation to the output magazine OM1. Having completed this task the robot returns to the rerientation stand, takes a workpiece that was resting there for some time now and mounts it on MT2. At last it returns to the MT1 and after the first maching operation is over the cycle starts anew again. Some of these procedures deal with proper system functioning documentation, structuring of the robot programs (including procedures for automatization of their flow design) as well as of the overall computerized control system design - a man-automated machine system interface. This interface has been thoroughly remodelled in each case as the user demands vary significantly always. Nevertheless some basic procedures that are commonly employed form our specialized task oriented software library now.

REFERENCES

[1] Wójcikowski J., Madejski J. : Prototypowy układ nadzoru pracy zrobotyzowanego gniazda tokarskiego, Proceedings of the Institute of Machine Technology, Silesian Technical University, Gliwice 1986

[2] Zdanowicz R. : Komputerowy system nadzoru pracy zrobotyzowanego gniazda, III Krajowa Konferencja Robotyki, tom I str. 317 - 323 , Wrocław 1990

[3] Sobczyk W.J. i inni : Rozwój oprogramowania sterującego pracą zrobotyzowanych gniazd produkcyjnych, proceedings of the Institute of Machine Technology, Silesian Technical University, Gliwice 1991, unpublished

[4] Козырев, Роботизованные технологические системы - разновидность гибких производственных систем, СТАНКИ И ИНСТРУМЕНТ, str. 2-3, 10/1989

KOMPUTERUNTERSTÜTZTE ÜBERWACHUNG KLEINES ROBOTISIERTEN PRODUKTIONSSYSTEMS

Zusammenfassung. In der Arbeit wurde logische Struktur des überwaschungsystems kleines Produktionssystems dargestellt. Es wurde die Zuteilung der Steuerungsaufgaben den Systemsteilen vom oberste Niveau bis zum Nutzprogrammniveau herab dargestellt. Es wurden konkrete Aufgaben besprochen und Steuerungssystem genau beschrieben. Es wurde auch neue technologische Dokumentation, welche dem Bediener leichten und fehlerlosen Übergang zur neuen Aufgaben ermöglicht, vorgestellt.

KOMPUTEROWY NADZÓR NAD MAŁYM ZROBOTYZOWANYM SYSTEMEM PRODUKCYJNYM

Streszczenie. W referacie przedstawiono strukturę logiczną systemu nadzoru małego systemu produkcyjnego. Przedstawiono przydział zadań sterowania elementom systemu od poziomu najwyższego w dół, aż do poziomu oprogramowania użytkowego. Omówiono konkretne zadania i opisano dokładnie system sterowania. Zaprezentowano nową dokumentację technologiczną umożliwiającą operatorowi łatwe i bezbłędne przechodzenie do nowych zadań.

Wpłynęło do redakcji w styczniu 1992 r. Recenzent: Wit Grzesik