Robotica (1999) volume 17, pp. 23–32. Printed in the United Kingdom © 1999 Cambridge University Press

A modular control system for flexible robotized manufacturing cells

Peter Kopacek, Gernot Kronreif and Robert Probst

Institute for Handling Devices and Robotics, Vienna University of Technology, Floragasse 7A, A-1040 Vienna (Austria)
E-mail: {kopacek,kronreif,probst}@ihrt1.ihrt.tuwien.ac.at

SUMMARY
In co-operation with an Austrian small sized company, the Institute for Handling Devices and Robotics at the Vienna University of Technology started a project with the goal to reduce the time necessary for planning, programming and set-up of robot equipped manufacturing cells. In this contribution a modular, object-oriented control system for robotized cells is presented. This system 'C_CTRL' is responsible for all necessary tasks, like sequencing, supervising, controlling of basic functions, error handling, and recording of statistic data. The control system is 'self-generating' during program start-up – using hardware information from a simple ASCII configuration file. Regardless which and how many components used in a particular cell, there is no additional programming effort for generation of the entire control software.

KEYWORDS: Flexible manufacturing cells; Modular control system; Robots; Design considerations.

1. INTRODUCTION

Robotized assembly/disassembly automation has increased dramatically over the last years. Especially for small and medium sized companies, there is a great demand for flexible modular (dis-) assembly cells usually equipped with components like robots, grippers and tools, storage devices and part feeders, and other auxiliary devices. The costs and expertise required to set-up a cell, and subsequently to operate it, negates the many advantages of implementing such flexible systems. To guarantee high flexibility and efficiency, an appropriate distribution and a context sensitive supply of information between all components must be ensured. This paper deals with a new modular control software package developed at the Institute for Handling Devices and Robotics (IHRT) at the Vienna University of Technology. This control software for robotized manufacturing cells is designed to shorten the time for putting into operation of the cell and should result in a more efficient, flexible cell operation.

The possibility to assemble both particular instructions and sub-sequences to one manufacturing sequence opens a broad variety of cell programs. Due to the structure of the program 'database' defined in the 'Sequence Editor **C_SEQU**' the user can add new modules or delete obsolete ones to/from the program library in a very comfortable way. In addition, the editor allows easy selection of single program instructions and their combination to a manufacturing sequence.

The tasks of the second package, the 'Sequence Controller **C_CTRL**', are to check the availability of all components in each stage of the assembly process, to start the corresponding robot program sequence and to send all necessary commands to the connected subsystems. Furthermore, the sequence controller is connected to a man-machine interface, which allows comprehensible monitoring of the current cell state. All error and status conditions of the connected systems are interfaced to this module. For better error localisation all process relevant data are stored in an external record file.

Until now, the system has been used in an assembly system for automated assembling of flashlights and a system for assembling moisture proofed lamps. For these applications the advantages of **C_CTRL**, i.e. fast set-up of the cell control system, efficient integration of cell components, and monitoring functions proved to be successful.

Beside the description of the control system, this paper deals with the most recent application example: an assembly system for Geneva gears used for education in robotics.

2. DEFINITION OF 'CELL'

Currently, there is no standard – or even accepted – definition of the term 'cell'. In industry, one finds two types of cells: they contain either a collection of identical, or functionally identical, equipment or they are group technology cells. In the first case, a cell might contain a collection of drills, or a collection of milling machines, etc. In the second case, a cell might contain all of the equipment (sometimes a single piece) needed to manufacture a particular family of parts, assemble a specific family of circuit boards, etc. As a typical example of just this second type, robotized manufacturing cells can be mentioned. Depending on the particular manufacturing tasks, a robotized cell may include (beside one or more industrial robots) some peripheral devices, like:

- grippers and/or tools and changing devices
- auxiliary fixtures
- transfer systems
- part feeders
- checking devices, quality assurance systems
- additional PLC or CNC controlled facilities

Thus, an interpretation of the role of a cell control system is that it should manage the interaction between the particular

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

components under its charge, thereby changing and checking states according to some plan of action which might be generated by a supervisory system higher up the manufacturing hierarchy.

3. CONTROL STRUCTURES

Connected with the number of robots grouped in such a cell, there are different control structures in use.[1] For single systems, the robot controller might be responsible – beside its real tasks to control and co-ordinate the robots actions – for all co-ordination between the robot and connected peripheral devices. Increasing the number of robots used in one cell usually leads to an expansion of the co-ordination tasks between the peripheral devices as well as between the robots. To clear this obstacle a *Master-Slave-System* can be chosen, where one robot controller acts as the master, and the other one(s) as slave(s). The master co-ordinates the movement of all robots – each robot supervises the peripheral devices related to its task. On the other hand, a *cell controller with two* (*or more*) *slave robots* can be applied. In this connection, a supervising controller carries out all communication and co-ordination tasks between the particular subordinated components of the cell. To sum it up, control structures for robotized cells can be divided into three different strategies:

- The entire system is controlled by a robot controller.
- Some of the peripheral devices are controlled by a programmable controller (PLC). The robot controller supervises these subsystems and co-ordinates all tasks.
- The entire system is controlled by a supervisory system – each of the subordinated systems (peripheral devices, robot, etc.) is controlling its own basic functions.

4. CELL CONTROLLER DESIGN CONSIDERATIONS

When integrating the activities of a set of subsystems to perform a manufacturing task, a range of system design issues needs to be considered.

4.1 Modularity, configurability

Using tailored hardware and software solutions, a group of subsystems can be combined to perform a single, specific manufacturing application. However, in this instance, a redefinition of the application (which may include removing/adding one or more subsystems, re-synchronising the interactions of the subsystems, etc.) would involve significant engineering effort and costs and would result in yet another specific solution.

A hierarchical and modular cell control system must therefore be capable of redefining the presence as well as the interactions of the subsystems under its charge. These circumstances identify the need for cell controllers to offer flexibility with respect to modularity and configurability to control all the components used in a cell.

4.2 Sequence co-ordination capability

The sequence co-ordination function (SCF) is the interface between the cell controller and its subordinate equipment controllers. It releases jobs to subordinates and monitors subordinate feedback on those jobs. If the subordinate can only manage one job at a time, the SCF should allow one to release one job at a time. A new job for the same subsystem should only be released when the previous one has been completed. Considering this, the sequence co-ordination module of a cell control system must be able to handle the following control tasks:

- controlling the serial tasks
- control of tasks carried out simultaneously
- making decisions about continuing the cycle, based on the current state of the system
- making decisions about stopping or delaying the cycle, based on the current state of the system

4.3 User-interface

The user-interface of a cell control system has to assist the user by sending commands and inform the user about the current state of the cell. Some important features of such an interface should be:

- monitoring the state of the system (i.e. of all subordinated modules)
- defining and editing the manufacturing sequence
- debugging the manufacturing sequence (e.g. single-step mode, setting new starting points)
- automatic/manual mode (e.g. executing certain operating steps for manual mode)

5. CONCEPTION OF A SYSTEM-INDEPENDENT CONTROL SYSTEM

To realise a general, system-independent control system, two methods – the one does not preclude the other – can be taken into consideration:

1. development of configurable modules
2. parametric system

5.1 Configurable modules

Defining the term ‘configure’ as ‘forming a whole from the particular’, a configurable system is composed of different, pre-defined modules. Each of these modules accomplishes a well-defined function and contains a clear interface to other modules.

By means of such a modular system structure, a control system applicable for a set of different systems can be developed (Fig. 1).

5.2 Parametric system

A second method for development of a system-independent control system is the use of parameters, i.e. to put in variables at appropriate sectors in order to control execution. One of the most important applications for this technique is the use of *global constants* – occasionally stored in *configuration files* – in order to configure a system.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

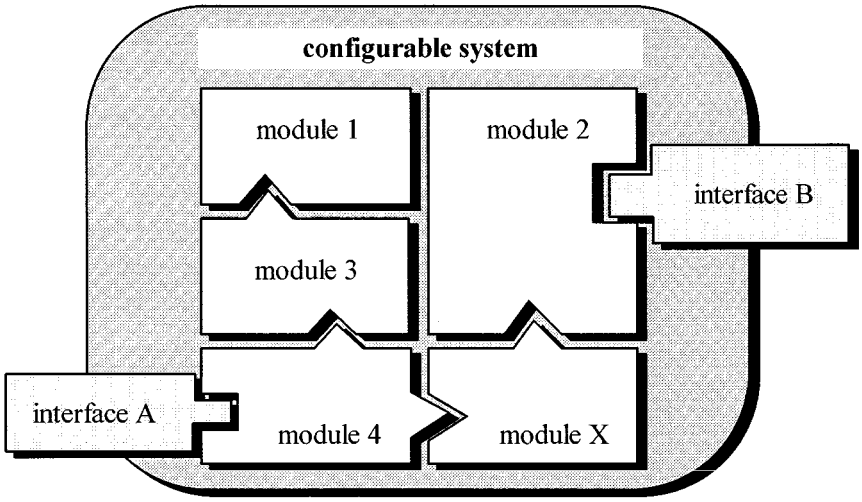


Fig. 1. Structure of a configurable system.

5.3 Object-oriented systems

Stelzner et al.[2] describe object-oriented programming (OOP) as 'the process of building a computational model of a system consisting of discrete objects that interact with one another by passing messages . . .'. Therefore, programs written in object-oriented languages involve the definition of objects and object-trees to 'rebuild' the real system.

Concepts of object-oriented systems: The essence of object-oriented systems is provided by the concepts of *encapsulation* and *inheritance*.[3] *Encapsulation*, the grouping of data together with the procedures that manipulate it, encourages modularity by isolating the execution of any service to be carried out from the object asking for the service. *Inheritance* promotes the reuse of modules by allowing objects to be specified as specialisation of other objects. Only when a new object differs from existing ones that it needs to be described.

A typical object class definition consists of a set of *slots* storing attributes of the object concerned, and a set of *methods* defining how the object behaves (modifying slots, or sending further messages) on receipt of messages. When an *instance* of this object class is created, it will have its own values of all the defined *slots* but will respond to incoming *messages* in exactly the same way as other *instances*.

In terms of the modelling of manufacturing cells, the object-oriented approach provides the capability to represent the system naturally, developing objects to correspond with the elements of the real system. Furthermore, the intercommunication between such objects can be represented by the methods of the particular object class.

Object-oriented models of a manufacturing cell: A manufacturing cell corresponds to this in the sense that it consists of a collection of entities which communicate to ask tasks of each other in order to achieve the overall objective of the entire system. In this connection, the principles of object-oriented systems can be used through definition of a *cell entity* as the building block from which the full cell definition can be constructed.

5.4 Object-oriented approach for controlling of a robotized manufacturing cell

For practical applications of the concepts of object-oriented modelling described above, the collection of cell entities might be arranged hierarchically. In this sense, one possible solution for definition of a control system is a two-level architecture with using the cell entity for both description of **low-level devices** and as **cell control module**.

The cell entity as a low-level device: At this level, each object corresponds to a particular cell 'hardware'-component; the cell entity can be used to describe the local control tasks present in the cell ('equipment controller'), such as robot controller or programmable logic controller (PLC), in the following manner.

- The device runs a controlling process which takes certain decisions on the basis of a model of the state of the device. This model includes elements describing the device's internal state and the signals received from its sensors interfacing with the physical world.
- Outgoing communications allow reporting of the device's local state back to the supervisory system.
- The device can be considered as an autonomous entity.
- Each device only executes one task at a time.

The cell entity as a cell control module: At this supervisory level, the cell entity is responsible for co-ordination of the operation of the particular subordinated low-level devices rather than for controlling their individual actions in detail.

6. MODULAR CONTROL SYSTEM C_CTRL

At the Institute for Handling Devices and Robotics (IHRT), a modular control concept for robotized manufacturing cells has been developed, which helps to reduce the amount of programming and to increase the efficiency and flexibility of the cell. This concept, called **C_CTRL**, is fully object-oriented and realised using the programming language *C++* and *Turbo-Vision*. Each module is defined as an object (in terms of object-oriented theory) which can be adapted for special behaviour.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

6.1 System architecture

The key aspect of any cell control system is to have sufficient functionality to allow an interaction with a vast array of cell components, i.e. of 'everything required to accomplish one or more manufacturing activities'.[4] In the control system **C_CTRL**, this interaction takes place within a two-level hierarchical control structure (see Fig. 2). The lower level is dedicated to **subordinated local control functions**, which are divided into *subsystems with their own controller*, such as robots, screwing and soldering devices, and *systems without own controller*, like single pneumatics elements, simple grippers or tools, and part feeders.

All these subsystems interface to the **sequence controller** i.e. to the supervisory level. To control, co-ordinate and manage the entire cell, this level is connected to a database containing the necessary information for performing each particular assembling step. One other characteristic of this system architecture includes a *master/slave* relationship which exists between the two levels with command data flowing downwards in the hierarchy and sensory data flowing upwards towards the supervisory level.

The subordinated systems are implemented as *objects* ('cell entities'). Concerning the principles of object-oriented structures, each object consists of different variables (e.g. defining the state) and commands related to the task of the subsystem (e.g. open/close for pneumatic grippers). Each object has to be initialized at run-time with all related connecting information, such as availability of sensor signals at a distinct input-line.

For sequence co-ordination (see also chapter 4.2) the control system **C_CTRL** uses the concept of 'synchronisation steps'. During definition of the operating steps, each particular step can be assigned a 'TabStop' (see Fig. 9), which ensures – during execution of the assembling sequence — that a new job is only released, when *each previous released job* (assigned to any subordinate) is already finished. In addition, the system provides elements for controlling the program sequence, like definition of labels, and conditional or unconditional branches (GOTO, IF).

6.2 System structure

The control system consists of two software modules – the *assembling sequence editor* '**C_SEQU**' and the *sequence controller* '**C_CTRL**'. Both modules are linked to each other via an ASCII *cell description file* (see Fig. 3).

Cell description file. All information necessary for definition of the cell structure is stored in a cell description file (**CDF**). The structure of this file is similar to the initialization files of MS-Windows (*.INI), i.e. the file consists of different sections (introduced by a key-word) followed by appropriate definitions (see Fig. 4).

Using this ASCII-file, the entire cell control system (including the man-machine interface) is generated automatically. According to object oriented programming technique, the control software includes the declaration of certain objects representing classes of 'real-world' cell components, which are reacting to specific combinations of input signals and activated methods by setting appropriate output signals. After starting the cell control software the CDF is scanned for definition sections of particular components and software objects ('cell entities') are created – all these objects together representing the 'real-world' system. Changing the hardware setup requires only a modification of the cell description file (and a restart of the control software). Consequently, no programming effort is necessary when installing/modifying a robotized cell.

Assembling sequence editor C_SEQU. To use the control system **C_CTRL**, the local control functions (for subsystems with own controller; controlling of the basic functions of subsystems not having their own controller is

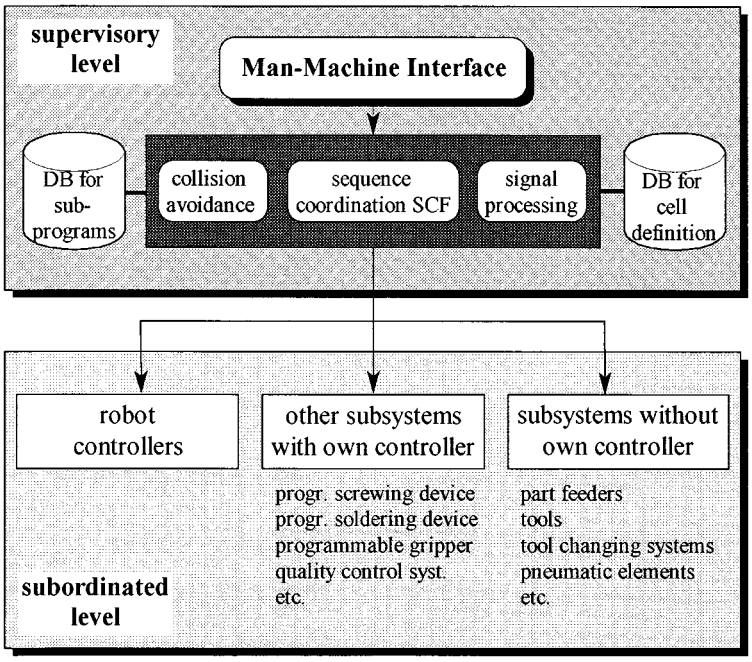


Fig. 2. Hierarchical structure of the control system C_CTRL.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

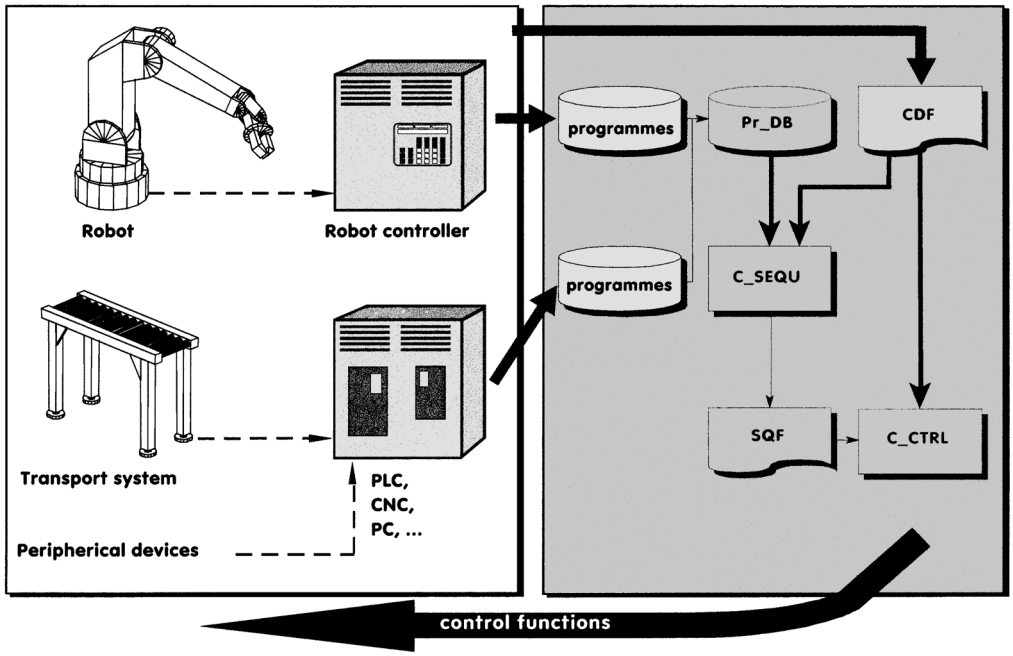


Fig. 3. Structure of the control system C_CTRL; Interfaces to the real manufacturing cell.

accomplished by the sequence controller itself) have to be programmed as a first step. According to the type of the subsystem, these programs might be stored in particular database systems (in the broadest sense) – i.e. the robot controller, PLC, and others. To manage these single programs and to link them together to an *assembling sequence*, a 'Program Database' (**Pr_DB**) is used, storing a short description, the 'address', and some additional data of each subprogram. To involve new assembling tasks or to change existing ones, it is necessary to change or add information to this database (for further details see chapter 7, 'step ④').

The user-interface of this system — called **C_SEQU** – has been designed as a three-window system (similar to MS-Windows file-manager). The left window offers a rough structure of the database (see Fig. 5). It consists of pre-defined groups depending on the subsystem objects ('ControlStatements', 'Robots', etc.) and on additional user defined groups and subgroups ('HT_Bosch', 'Feeders_Bosch', etc.).

The upper right window provides all assembly (or manufacturing) steps related to the currently chosen database topic. New process steps have to be defined within this window by input of appropriate data by means of dialogue-masks. Using the right lower window, the particular assembling sequence file (**SQF**) can be generated and/or edited by inserting/deleting/moving of process steps from the upper window.

In other words, the left and the upper right window are a representation of the 'Program Database' which includes all the (predefined) programs of 'intelligent' cell components. Arranging these together into a certain sequence (mixed with other basic functions for 'non-intelligent' components, and/or control statements) results in an assembly sequence which is represented in the lower right window of the user-interface.

```
;Comment
[TYPE OF OPERATION]        ;Starting of a new
                           ;section
  NAME = Identifier
  TYPE = Description of the object
  ;Additional specifications (depending
  ;on the type of the object)
  ;e.g. number and 'address' of I/O lines


;EXAMPLE: Definition of pallet index/lifting tables
[HUBTISCH]
  NAME = HubtischBOSCH
  TYPE = HTB
  HUBT_I_ADRESSE = 27      ;Start address input -  Number = 5 (fixed)
                           ;reading sensor information
  HUBT_O_ADRESSE = 3       ;Start address output - Number = 3 (fixed)
                           ;operating pneumatic elements
  TIME_OUT = 30            ;Time-out: waiting for sensor information
```

Fig. 4. Structure of the cell description file.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

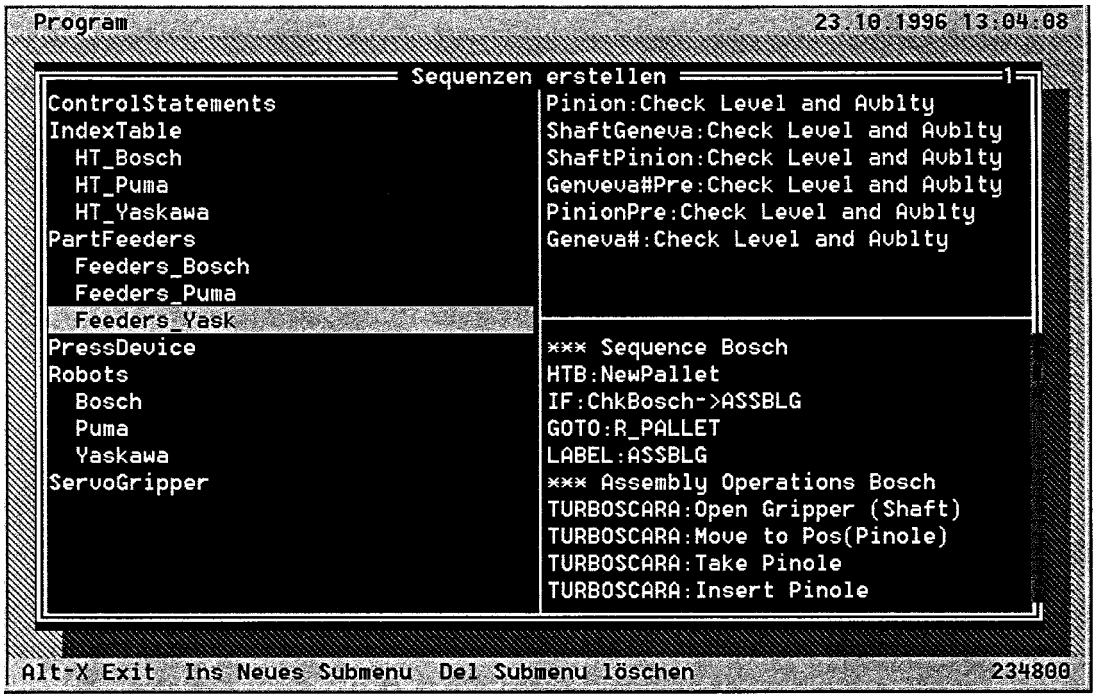


Fig. 5. Snapshot of the GUI of the sequence editor C_SEQU; The application programme (i.,e. the assembling sequence) can be 'programmed' by select/insert of pre-defined programme modules.

Sequence controller C_CTRL. Using the information given in the cell description file (ASCII file) and in the assembling sequence file (created with **C_SEQU**), the sequence controller serves as supervisory system for the entire cell. In detail, the tasks of the sequence controller are:

- starting the corresponding robot program sequence.
- sending all the necessary commands to connected subsystems (like grippers, tools, pneumatic elements, measuring devices),
- supervising all subsystems with own controller/controlling the basic functions of subsystems without own controller,
- error handling. There are two possible types of system errors: **Warnings** and **Errors**. For the first group (e.g. warnings generated by the level indicator of the part feeders) there is an acoustic warning and a message prompted on the screen – the assembly sequence continues without needing the user to confirm this warning. In case of an error (e.g. time-out messages, no part available in feeder, wrong effector attached to the robot) there is an acoustic error signal as well as an appropriate message window displayed on the console – the assembling process (for this cell) stops at this particular operation step and needs confirmation by the user for continuing.
- send all sensory data generated by the subordinated components to the connected man-machine interface.

After starting **C_CTRL**, the information stored in the cell description file is used to create modules for interaction (i.e. set operating modes, read sensory data, etc.) with the particular cell components as well as for generation of the man-machine interface. This interface module displays all error conditions and status information of the cell – like information about running subsystems, assembling steps, and all installed subsystems – in different windows (see Fig. 6) and generates report-files.

Furthermore, the interface allows to start the particular assembling sequence at a certain point, to start an individual operation on a subsystem (e.g. for debugging aids), and to execute different variants of assembling sequences.

6.3 Control of robotized manufacturing lines

Up to now, the capabilities of **C_CTRL** have been successfully demonstrated in several different test cases, like assembling of flashlights,[5] or assembling of moisture proofed lamps. Although using more than one robot in each of these assembly systems, the process is taking place in a robotized *cell*. To enlarge the area of application, control software **C_CTRL** is being provided with features for parallel control of multiple robotized cells linked together to a manufacturing *line* by means of transport systems.

Additional modules for **C_CTRL** are being built to solve the new tasks:

- Definition and loading of different sequences in each particular cell.
- Control of each individual cell.

As a first step, the main tasks of the sequence controller are split into different modules. Generation of all particular modules for the line is done globally during start-up phase of **C_CTRL**. This and the entire user interface with all monitoring functions are part of the main program. At the end of the start-up procedure, several '*cell-modules*' are created – according to the number of robotized cells linked together to the line and with a structure defined in the cell description file CDF. Each cell-module is now responsible for loading its own SQF and executing the particular sequence as described in the chapter before.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press



Fig. 6. Snapshot of the GUI of the sequence controller C_CTRL with information windows for cell status display, runtime errors, current active subsystems, and current running operations.

The second module is called '*Switcher*'. This module is designed for switching between the control functions of the cell-modules to ensure control for the whole line (Fig. 7). Using small cells linked together by means of a pallet transfer system, the task for this second module is very easy, one simply calls the control functions in a cyclic way. The start of a new assembly sequence depends only on the availability of a work piece in the buffers of the transportation line between the cells. In one of the next releases of **C_CTRL**, the *switcher* module will be improved for handling several autonomous cells linked together by other transport devices, like AGVs (autonomous guided vehicles).

7. APPLICATION EXAMPLE

The new features of **C_CTRL** are being tested in a manufacturing line for the automated assembling of Geneva gears. This assembly system is located at the robotics laboratory of the Institute for Handling Devices and Robotics (IHRT) at the Vienna University of Technology and used (beside serving as development and test environment for the new modules of the control software) for education in robotics and consists of

- robot Bosch TURBOSCARA SR60 equipped with a pneumatic gripping device
- robot Unimation PUMA 560 equipped with a pneumatic gripping device
- robot Yaskawa MOTOMAN K6511 equipped with a programmable servo-gripper (developed by IHRT)
- part feeders for
  - pinole + shaft (pre-assembled)
  - Geneva wheel + shaft (pre-assembled)
  - Geneva wheel
  - pinole
  - shaft for pinole

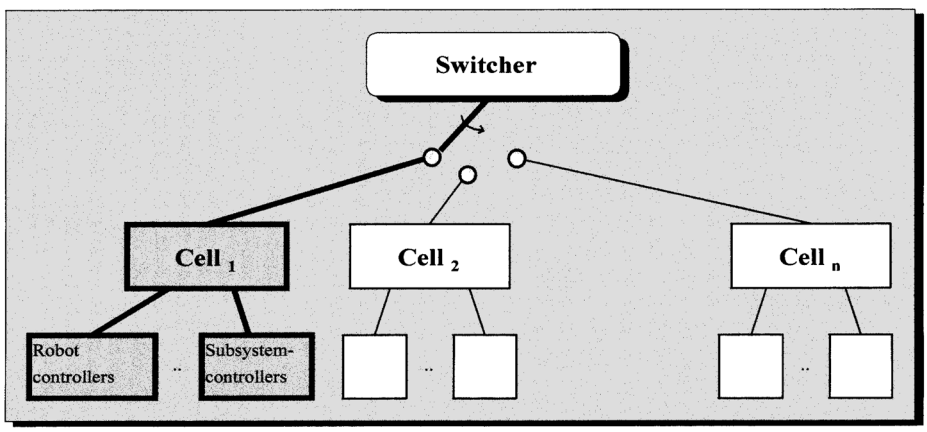


Fig. 7. Sequence co-ordination for control of manufacturing lines.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

  - shaft for Geneva wheel
  - case cap
- Pallet of gear boxes
- special operating pressing device (to pre-assemble the pinole/Geneva wheel with the shaft)
- manual assembly station equipped with a screwing device

All the components listed above are grouped together to three assembly cells linked with a BOSCH pallet transfer system with 3 pallet index/lifting tables and 2 pallet reversing devices (see Fig. 8). For final assembling (screwing of the case cap) and unloading of the finished parts, there is an additional manual assembly station. The structure of the cell and the assembling steps for each particular cell are listed in table I.

At this point, it should be mentioned that the problem in question could also be solved with only one cell and/or with a reduced number of robots. The layout of the assembly line and the allotment of the assembling operations to the particular cell was especially designed to develop and to test the new features of the control software **C_CTRL** and due to didactic purposes.

On the basis of this application example, the required steps for cell set-up and programming are as follows:

(i) Hardware set-up
To establish the flow of information between the supervisory controller (running on an IBM compatible personal computer, DOS operating system) and the subordinated ones, each component is linked to the main controller via digital input and output lines (24 Volts). These signals (for the example in question, there are 72 input and 72 output lines available) are connected to the personal computer via

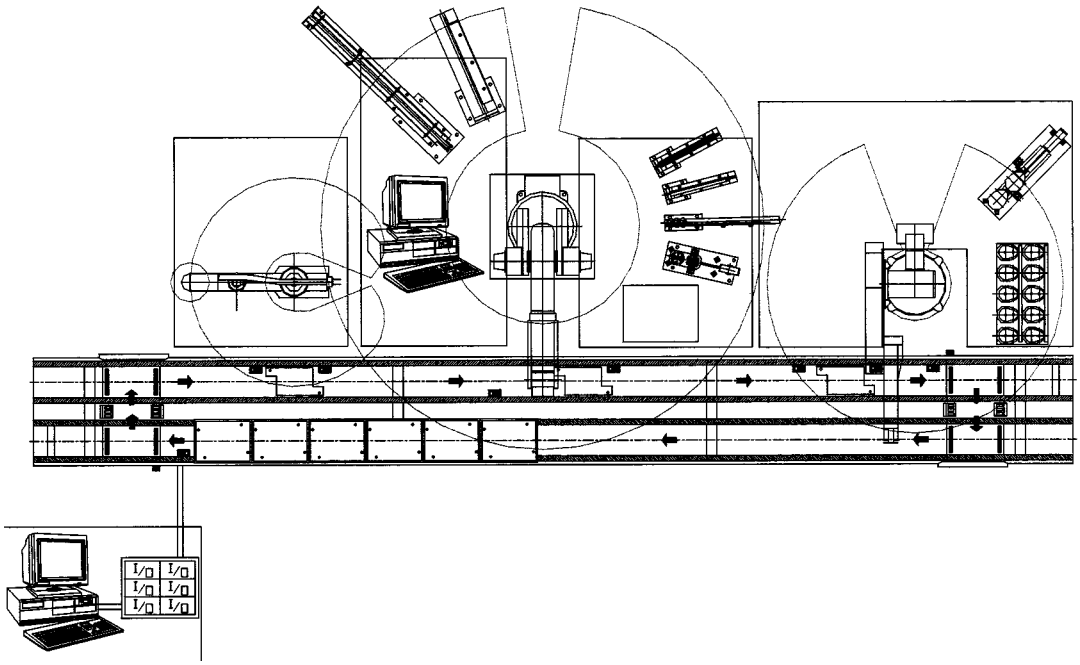


Fig. 8. Manufacturing line for automated assembling of Geneva gears – Layout.

Table I. Structure of the assembly line; Assembling operations.

Cell	Components	Assembling steps
1	• Robot TURBOSCARA SR60.	1) Insert pinole + shaft into gear box 2) Insert Geneva wheel + shaft into gear box
2	• Robot MOTOMAN K6SB. • Part feeders for pre-assembled pinole + shaft and Geneva wheel + shaft; Part feeders for Geneva wheel, pinole, shafts. • Special operating pressing device for pre-assembling.	1) Taking pre-assembled part from feeder or pre-assembling of pinole and shaft (if no part in feeder) 2) Taking pre-assembled part from feeder or pre-assembling of Geneva wheel and shaft (if no part in feeder) 3) loading pinole + shaft/Geneva wheel + shaft to pallet
3	• Robot PUMA 560. • Pallet of gear boxes. • Part feeder for case cap.	1) Loading gear box to pallet resp. 1) Assembling case cap

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

a digital input/output extension card as well as 3 input (equipped with isolated photo couplers) and 3 output extension modules (equipped with relays).

(ii) Programming the cell components
As described above, control software **C_CTRL** distinguishes between components with and others without own controlling capabilities. For components with own control system (e.g. robots, programmable gripper devices, etc.) the necessary programs have to be prepared in this step. To make use of the flexibility provided by **C_CTRL**, it is recommended to keep each single program as modular as possible (e.g. for the robot: basic movements, programs for opening/closing the gripper, etc.).

(iii) Cell description file
According to the hardware connection (See 'step (i)'), one has to prepare the cell description file. In this file, all the components needed to accomplish the assembly task must be defined with respect to starting address and number of input/output lines, or other specifications.

Example: Robot TURBOSCARA SR60
Output lines: *24, 25, 26, 27, 28, 29, 30, 31*
7 lines for sending the program number (binary coded) to the robot controller; 1 line for signal 'start program'.
Input lines: *0, 1, 2, 3, 4, 5, 6*
1 line 'busy'/'inactive'; 1 line 'standby_on'/ 'standby_off'; 2 lines for error signals; 3 lines for gripper status (not used).
Referring to this input/output configuration, the definition block in the cell description file has to be prepared as follows:

```
[ROBOTER]
NAME = TURBOSCARA
TYPE = BOSCH
PRG_I_ADRESSE = 0
PRG_I_NUMMER = 7
PRG_O_ADRESSE = 24
PRG_O_NUMMER = 8
WW_I_ADRESSE = 71    ; Tool changing device (not used)
WW_I_NUMMER = 0
ERR_I_ADRESSE = 71   ; Transfer robot error code (not used)
ERR_I_NUMMER = 0
```

(iv) Program database
Using the sequence editor C_SEQU, all the necessary subprograms (created in 'step (ii)') must be described (e.g. for the robot programs: program name, program number, appropriate effector, necessary gripper status, synchronization marker 'Y'/'N', etc.) and divided into groups and subgroups (see Fig. 9).
Concerning the components without local controller, all basic functions (e.g. 'New_Part' for the part feeders; 'Raise_Cylinder' for the pneumatic modules; etc.) have to be included into the program database as well. Control of these functions is accomplished by the cell controller itself.

(v) Assembling sequence
The next step is to arrange the assembling sequence of each particular cell using the program steps defined in the program database. In addition to these paticular subprograms, some control statements for program flow (IF statments, LABEL, GOTO statements) are available (see Fig. 5). As described in chapter 6.2 this step can be done in an interactive

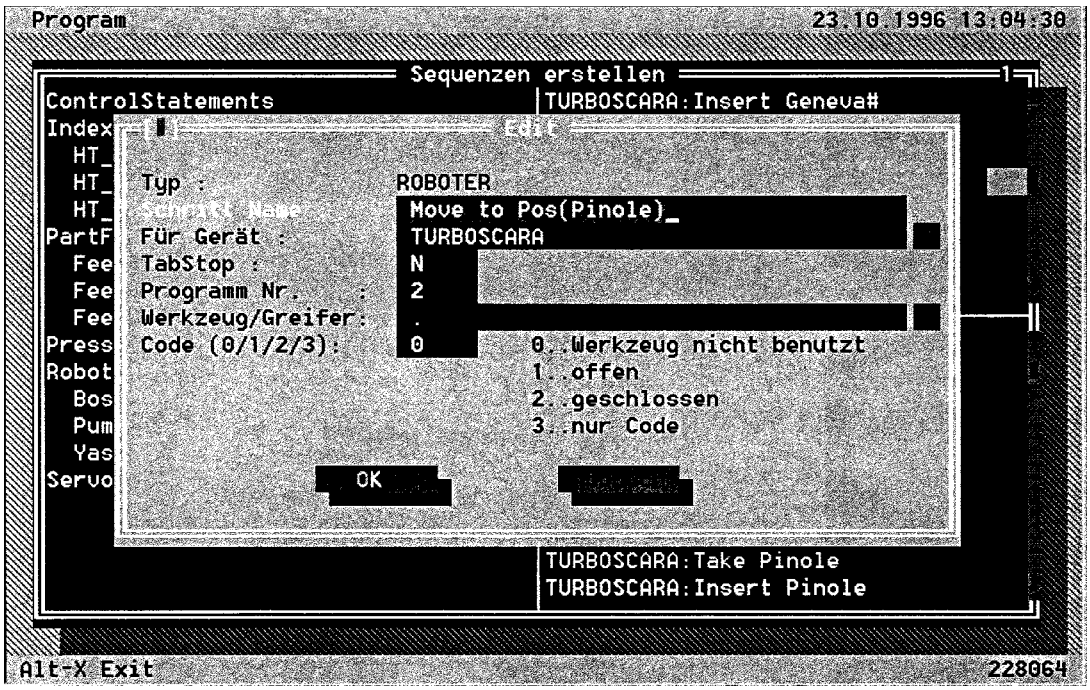


Fig. 9. Inserting a new subprogram into the 'Program database' Pr_DB.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press

way using software system **C_SEQU**. First, the user has to select the next operation step from the 'Program Database' – this operation can be linked to the assembling sequence by pressing the 'INS'-key. In addition, the user is able to modify the current assembling sequence by deleting/moving of assembling operations linked to the sequence.

(vi) Cell control
When starting the cell controller **C_CTRL**, the appropriate cell definition file as well as the particular assembling sequence for each single cell are loaded to the system. Based on this information, **C_CTRL** generates the necessary software objects for controlling and monitoring the entire assembly line (see Fig. 6). Beside cell control in automatic mode, there are several 'debugging aids', like single-step mode, executing of single program steps, etc. provided by the control software.

8. CONCLUSION

While prices for robot systems are decreasing more and more, the costs for integration of the cell components (and consequently the costs for the robotized cell) are still a considerable obstacle to the use of robotized (dis)-assembly – especially for small and medium sized companies. One of the major reasons for that is the need of tailored software solutions (and the significant engineering effort connected to this) for supervisory control functionality and monitoring.

In this paper, a new flexible, modular and 'low-cost' solution for hierarchical control of robotized manufacturing cells and lines is presented. The fully object-oriented system is based on a master-slave principle between the supervisory control unit and the subordinated systems on equipment level. Each subsystem defines related commands and is responsible for only local control functions (e.g. moving the robot from one point to another, gripping a distinct workpiece or testing a stored part). The supervising cell-controller module consists of a sequence controller, an advanced man-machine interface and a database system, and is responsible for all co-ordination, communication and visualization tasks. The concept is realized in *C++* using objects for all subsystems, the man-machine interface and the sequence controller. Each subsystem is designed to work in a given way, but can be easily adapted to special behavior.

The entire system – consisting of the cell control system **C_CTRL** and the assembling sequence editor **C_SEQU** – was tested in different applications, like a robotized cell for assembling of flashlights and a cell for assembling of moisture proofed lamps. As a result of the first applications, the merits of this new system, such as efficient description of the cell structure, high flexibility for definition/modification of the assembling sequence, and the improved monitoring system, could be proven successfully. The modular concept and the object-oriented approach realized in this control system allows us to reduce the time for the generation of a supervisory control system to a very short period of time. Whenever adding/deleting/modifying a certain component to the cell, this new device has only to be defined in the ASCII cell description file, and an updated cell control system is available after restarting **C_CTRL**.

Beside the applications mentioned above, the control system described in this paper is used in the robotics laboratory of the Institute for Handling Devices and Robotics at the Vienna University of Technology for the development of additional modules and for education in robotics. Resulting from this usage, **C_CTRL** is currently being extended to the control of robotized *lines* as well. In addition and according to the current research tasks, new groups of 'cell entities', e.g. objects for image processing or wrist force sensors, are in the stage of development. Other extensions, like improved generation of the cell configuration file by means of a special CAD-interface, are also planned.

References

1. R.S. Wiener and L.J. Pinson, *An Introduction to Object-Oriented Programming and C++* (Addison-Wesley, Reading, Mass., 1988).
2. M. Stelzner, J. Dynis and F. Cummins, "The SimKit System: Knowledge-Based Simulation and Modelling Tools in KEE", *Intellicorp Technical Article* (1987).
3. R. Probst and G. Kronreif, "Information Technology in Robotics", *Proceedings of the* 1*st World Congress on Intelligent Manufacturing*, Mayaguez, Puerto Rico (1995) **Vol. 1**, pp. 612–620.
4. R.H. Weston, I. Coutts, A. Hodgson, S. Murgatroud and J.D. Gascoigne, *Manufacturing Cells*: *control*, *programming and integration* (D.J. Williams and P. Rogers, Eds.), Butterworth-Heinemann, London, 1991) pp. 10–17.
5. P. Kopacek, R. Probst, G. Kronreif, "A Modular Control System for Flexible Robotized Manufacturing Cells", *Preprints of the* 26. *International Symposium on Industrial Robots ISIR*, Singapore, (1995), pp. 533–538.

https://doi.org/10.1017/S0263574799001022 Published online by Cambridge University Press