UNIVERSITY OF STOCKHOLM

INSTITUTE OF PHYSICS

A MULTIFUNCTION EDITOR FOR PROGRAMMING CONTROL SEQUENCES FOR A ROBOT BASED RADIOPHARMACEUTICAL SYNTHESIS SYSTEM

A USER AND REFERENCE MANUAL

G. APPELQUIST and C. BOHM

USIP Report 90 - 01

January 1990

USIP -- 90 - 01 .

A Multifunction Editor for programming Control Sequences for a Robot Based Radiopharmaceutical Synthesis System

A User and Reference Manual

G. Appelquist and C. Bohm

University of Stockholm, Department of Physics
Vanadisvägen 9
113 46 Stockholm

ABSTRACT

The Multifunction Editor is a development tool for building control sequences for a robotized production system for positron emitting radiopharmaceuticals. This system consists of a SCARA robot and a PC-AT personal computer as a controller together with general and synthesis specific chemistry equipment. The general equipment, which is common for many synthesis, is fixed to the wall of the hotcell, while the specific equipment, dedicated to the given synthesis, is located on a removable tray. The program recognizes commands to move the robot, to control valves and to control the computer screen.

From within the editor it is possible to run the control sequence forward or backward to test it and to use the single step feature to debug. The editor commands include insert, replace and delete of commands in the sequence. When programming or editing robot movements the robot may be controlled by the mouse, from the keyboard or from a remote control box. The robot control sequence consists of a succession of stored robot positions.

The screen control is used to display dynamic flowchart diagrams. This is achieved by displaying a modified picture on the screen whenever the system state has been changed significantly.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

When dealing with the production of positron emitting radionuclides, highly automated systems are required to satisfy the needs for radiation safety and production capacity. These needs led to the development of the CAMP system, where the acronym stands for "Computer Aided Manufacturing of Positron-emitters" system. The aim was to create an integrated environment for automated synthesis of radioactive tracers together with efficient tools for the development of synthesis applications.

The synthesis system has been designed around a robot arm manipulator to achieve maximum flexibility. It also contains a number of computer controlled valves, detectors, ovens etc. The equipment that is specific to a given synthesis is placed on a removable tray. The entire system is controlled by a personal computer via command files, unique for each synthesis.

The development tool for the often complex control sequences is a "Multi Function Editor" (MFE), by which the different types of control statements can be assembled using familiar editing commands. The editor also supports run and single stepping commands to facilitate debugging. Special care has been taken to provide a comprehensive user interface that does not require computer specialists to program the system for a new synthesis. This has been achieved by using a "mouse driven" and, as far as possible, a self explanatory command structure with help information available at all times.

1.1 The multifunction editor

The MFE is used for generating new, or editing existing, command files. A command file contains a description of the control sequence for a specific synthesis. It consists of a number of command steps, either requesting new arm positions, or requesting other services, e.g. valve control commands.

The MFE has the usual editing facilities like insert, delete and replace. It can execute the command sequence forward, backward or in a single step mode. The program can be run in two modes, edit and real. In edit mode only the robot positioning commands are executed, all other commands are just written on the screen when they occur in the control sequence. In real mode, on the other hand, all commands are executed, e.g. valves are controlled and pictures are shown. The reason for this is to speed up the development of new synthesis, where most of the problems have to do with the robot positioning.

1.1.1 Motion control

In order to insert new robot arm positions into the control sequence, the arm must first be moved to the position using the MFE commands, before the store position command is given. There are three coordinate systems available for specifying arm motions:

- o Laboratory based cylindrical coordinates (r, ϕ, z) with the origin at the base of the robot column
- o Laboratory based Cartesian coordinates (x, y, z) with the origin at the base of the robot column
- o Hand based Cartesian coordinates with the origin in the hand (left-right, back-forward, up-down)

In addition the movements of the hand can also be controlled (grip, roll, pitch, yaw).

In order to move the arm, the value of a coordinate must be modified by one of the following alternative methods:

- o With the mouse
- o Interactively from the keyboard

- o With a remote control box
- o By specifying new coordinates from the keyboard

When using the mouse a coordinate is selected with the mouse controlled cursor. The selected coordinate is then controlled by the mouse until the button is released.

With the up/down arrow keys on the keyboard a coordinate is selected (indicated with an asterisk to the left of the coordinate on the screen). By pressing either the left or the right shift keys the selected coordinate decreases or increases and the arm moves accordingly.

From the remote control only the hand based coordinate system and the hand movements are available. There are two buttons for each of the coordinates that can be controlled, one for each direction. While pressing a button the arm will move along the corresponding coordinate.

It is also possible to select a coordinate and enter its next numeric value from the keyboard.

After having moved the robot arm to a desired position, using one or several of the above methods, this position can be stored as part of the control sequence. Together with the position, a speed setting is also stored, giving the user control over the choice of speed for each individual movement of the arm. When a number of consecutive positions have been stored it is possible to run through the sequence and then, if necessary, modify it.

1.1.2 I/O control

In addition to arm positions it is also possible to store I/O control commands. These can either be direct, such as e.g. to open a valve, or they can be delayed. Examples of the latter category are to wait for a certain time period or wait for an event to occur, such as waiting for a counter to reach a maximum.

1.1.3 Flowchart display

With "show picture" commands inserted into the control sequence it is possible to display pictures, created by a graphics program called PAINTBRUSH (from Zsoft Inc.), on the screen. This feature can be utilized to display dynamic flowcharts on the screen, updating them with a modified picture representing the new system state, when required.

1.2 Command file execution

A control sequence, which has been generated with the MFE, can be stored more permanently in a command file. Such command files can then be executed either during an editing session using the "run" command (after the file has been loaded), or directly by starting the MFE with the "-run" switch and the name of the commandfile as argument. The latter method invokes the MFE in real mode and starts the execution of the command file specified. For more information on different ways to start the MFE see page 16.

2. User interface

The user interface of the Multi Function Editor is based on "pull down" menus that are activated by the mouse. All valid MFE commands can be given from these menus. However, some common command may also be activated from the keyboard.

The screen is divided into separate areas or "windows", as seen in figure 2-1.

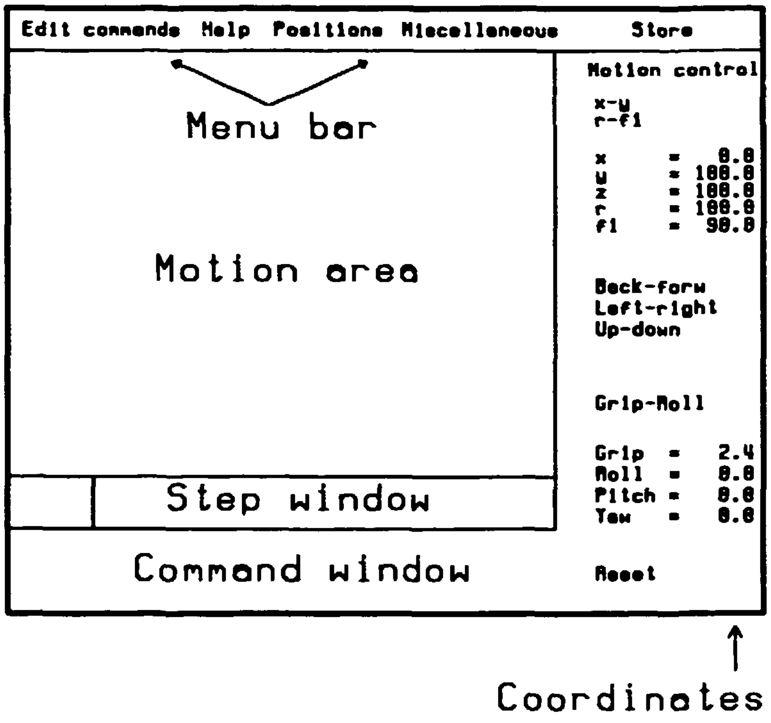

Figure 2-1 Screen layout of the Multi Function Editor

A menu bar, from which the different "pull down" menus are activated, is situated at the top of the screen. These menus contain all commands, that do not directly control robot movements (see section 2.1).

To the right there is a permanent menu for robot movements. This menu serves two purposes, it shows the current position of the robot arm and it is used for moving the arm to a new position (see section 2.1.6).

The large clear area in the middle of the screen is the motion control area. When the robot arm is guided by the mouse, this area represents the range of the currently controlled parameter(s).

Below the motion area is the step window where the current step number is displayed together with the command type and any parameters specified in the command. These parameters can be edited by pointing at them with the cursor. This invokes a line editor in a special editing window containing the parameter to be edited. It is also possible to go to any step in the sequence by pointing at the step number and then enter a new step number from the keyboard.

The area at the bottom of the screen is the command window. Here the user is prompted for input, when required. Messages to the user are also displayed here.

2.1 Menus

A menu is activated by pointing at the menu name in the menu bar and pressing the leftmost mouse button. As long as the mouse button is kept down, the selected menu stays on the screen. The menu contains a list of commands that can be selected by pointing with the cursor. The currently selected command is enclosed in a rectangle. When the mouse button is released the selected command is activated and the menu disappears. Now one of three things can happen. The command can perform some action without further user intervention, like e.g. executing the next step in the control sequence. It can display status information in the command window at the bottom of the screen, e.g. information about the current control sequence, or it can ask the user for some input, e.g. a new speed. In the latter case the user is prompted for the required information in the command window. The necessary information should then be typed in from the keyboard, terminated with a <Return>.

On-line help is available all the time by pressing, and holding down, the rightmost mouse button when an item in a menu is selected. The help information stays on the screen until the rightmost button is released. It is thus possible to check the action performed by (or the syntax of) a command just before the command is executed.

Below follows a description of each of the commands in the different menus. Here the function of the commands are discussed. For information on the exact syntax of each command, see section 3.2.

2.1.1 Edit

The commands in the edit menu are used to modify positions or commands already stored in the control sequence. Many of these commands depend on what is called the "current position". The current position pointer points at the next command to be executed. If a new command is stored it is inserted just before the next command in the sequence making the newly stored command the current step, that is shown in the step window. When single stepping, the pointe· is advanced one step forward or backward and the command just executed is shown in the step window.

File information

Gives information about the command file that is currently being edited. The total number of steps in the sequence together with the current step number and the file name are displayed.

New

Clears all previously stored steps from the memory and resets all parameters to their default values. This command should be used to initialize a new command file.

Load

Loads a command file from disk. The user is prompted for a file name (no extension is needed).

Save

Saves the current control sequence in a command file. The user is prompted for a name of the file. This name can be up to 8 characters long. It must start with a character (no extension is needed).

Previous

Executes the step just before the current step. The just executed step now becomes the current step. The "previous" command can also be activated with <ctrl>B.

Restore

If the robot has been moved interactively after the last positioning command was executed, e.g. with the mouse, this command restores the position of the arm to the position specified in the positioning command. In the step window the text "Arm moved" is shown each time the arm has been moved away from a position in the control sequence. This command can also be activated with <ctrl>R.

Next

Executes the next step in the control sequence. If the MFE is in edit mode only robot positioning commands are executed, all other commands are just displayed in the step window. The "next" command can also be activated with <ctrl>N.

Delete next

Deletes the step just after the current step in the control sequence. This command can also be activated with <ctrl>D.

Replace

This command is used for replacing existing robot positions in a control sequence. If the arm is moved to a new position, e.g. with the mouse, selecting this command replaces the current position in the control sequence with this new position. This command can also be activated with function key 3 (F3).

Run

Execution of the control sequence starting from the next command step. The sequence is then run through the number of times specified or until a key is pressed. If a key was pressed the current command is always completed. To start from the beginning of a control sequence, use the "reset" command before "run".

Reverse

Executes the commands in the control sequence in reverse order, starting at the current position. Continues until the first position is reached or a key is pressed. If a key is pressed the current command will be completed.

Quit

Leave the program. This command can also be activated by pressing the <Esc> key.

2.1.2 Help

This is not an ordinary menu. When selected a message window is shown with some general information about the program. Help on a specific command is available all the time by pressing the rightmost mouse button while the command is selected.

2.1.3 Positions

The position menu is used to memorize robot positions without storing them in the control sequence. This is useful if the same position occurs several times in a control sequence.

Remember

When selected, the user is asked to name the current position. This name will subsequently appear at the bottom of the positions menu. Selecting a position name will cause the robot arm to move to the corresponding position, independent of its previous position. The control sequence is not affected and the current position remains the same.

Load

Load positions from a file. The user is prompted for a filename (no extension is needed). The positions loaded appear at the bottom of the positions menu and can be selected as a normal menu command.

Save

Save the memorized positions in a file. The user is prompted for a file name in the command window. Positions that have been saved in a file can always be recovered with the load command.

Reset position

This is the only predefined memorized position. Selecting this entry moves the robot arm to the reset position, without affecting the control sequence.

2.1.4 Miscellaneous

This menu contains commands that do not directly deal with the control sequence and that do not fit into the other menus.

Dos

Temporarily leaves the program and enters a Dos shell where normal Dos commands are recognized. Control is returned to the MFE when an "exit" command is given to the Dos shell. No parameters in the MFE are altered or lost by using this facility.

Edit/Real mode

Toggles between edit and real mode. In the edit mode only the robot positioning commands in the control sequence are executed. All other commands are just displayed on the screen. In real mode, all commands are executed, including commands that control valves, show pictures and so forth.

Orientation

For almost every position in the horizontal plane there are two possible orientations of the robot arm, the elbow to the right or to the left. Selecting this command toggles between the two possible orientations. The current orientation is shown in the menu. It is important to note that not all positions in the plane can be reached with a given orientation of the arm. If a position cannot be reached, changing the orientation of the arm may solve the problem.

Autosave on/off

Switches autosave on and off. If autosave is on, the control sequence being edited is saved in a file at regular intervals, specified by the user when autosave was activated. Autosave is by default on and saves the control sequence at every third step added to the sequence. If the program crashes, the commands that were stored in the computer's memory can be restored by loading the command file "autosave" with the load command in the edit menu.

Initialize

The arm is initialized, which means that it moves to a well defined position and recallibrates the position of each motor. This command is useful if the robot position information has been corrupted, such as when the main power to the arm has been switched off, in which case the arm loses all knowledge about its position. An initialization command can also be put in the control sequence to improve accuracy in long or multiple runs.

2.1.5 Store

The store menu contains commands for inserting new steps into the control sequence. From within this menu the arms position can be stored as a part of the control sequence. Here are also the commands for controlling valves, showing pictures etc. A command is stored just after the current position in the control sequence. If the current position is in the middle of a sequence the command is inserted just after the current position but before the next step in the sequence.

Speed

Sets and displays the speed of the robot arm. The value shown in the menu is the current speed. If selected, the user is prompted for a new speed. The speed setting is stored together with the arm's position in the command file when a "store position" command is given. The speed must be in the range 1 - 10.

Show picture

Inserts a "show picture" command in the control sequence. The user is asked for a picture name (without extensions). This command can be used to display dynamic flowcharts on the screen.

Show counter

Inserts a "show counter" command in the control sequence. The show counter command is used to show the values of up to three counters on the screen when the program is in real mode. These values are updated between each step in the control sequence. They stay on the screen until they are switched off with this command. The counter display can be placed anywhere on the screen and can thus be part of a picture shown with the "show picture" command. For the exact syntax of this command, see section 3.2.

Wait for

Inserts a "wait for" command in the control sequence. This command orders the system to wait for a certain event to occur before continuing with the next step. The currently available events are: a count rate maximum on one of the counters connected to the system, exceeding a certain level at a counter or just wait for a certain amount of time. For the exact syntax of this command, see section 3.2

Do

Inserts a "do" command in the control sequence. This command orders certain predefined actions. At present, the only actions that can be performed is the opening and closing of valves. See section 3.2 for further details on this command.

Call

Inserts a "call" command in the control sequence. When a call command is encountered in the control sequence a subroutine, whose name is given as the parameter, is called. A subroutine can be any command file previously created with the MFE. See section 3.2 for further details on this command.

Error handling

With the "wait for" commands a time-out period can be specified. If the condition supplied is not met within the time-out period, an error is reported to the program. Normally, the program ignores these errors, but with this command errors can be taken care of. An error handling command can only be placed immediately after a "wait for" command in the control sequence. If an error occurs in the preceding command, one of two actions may be chosen. Either a user specified error message is written on the screen, or the program continues in a command file specified by the user.

Store position

Stores the position of the arm, together with the speed setting, just after the current position in the control sequence. This command can also be activated with function key 1 (F1).

2.1.6 *Motion control*

The Motion control menu stays on the screen all the time. It is used both for showing the current position of the arm, in all available coordinates, and to move the arm to a new position.

There are four methods for moving the arm:

o With the mouse

o By giving coordinates

- o Interactive control from the keyboard
- o With a remote control

Mouse control

The basic method of controlling the arm is using the mouse. By selecting one of the possible coordinates with the cursor and pressing the leftmost mouse button, that coordinate is controlled. The control continues until the mouse button is released. When a single coordinate is selected its value decreases when the cursor is moved to the left and increases when the cursor is moved to the right. When a double coordinate is selected (e.g. x and y) the first coordinate (in this case x) is controlled as usual, by moving the cursor horizontally, and the second coordinate (y) is controlled by moving the cursor vertically. Moving the mouse upwards increases the coordinate value. When a coordinate is selected, the cursor arrow is moved into the "motion area" (see fig. 2-1). As long as the arm is controlled by the mouse the cursor position indicates the value of the controlled coordinate. The left and right border of the motion area represents the minimum and maximum value of the currently controlled coordinate, respectively. The coordinate controlled by the mouse is marked with a white astensk to the left of the name of the coordinate.

When a coordinate is controlled with the mouse the rightmost mousebutton can be used to enter a high resolution mode (indicated with the text "HR" at the upper right corner of the Motion control menu). In high resolution the speed is set to 1 and the mouse movements are scaled by a factor of ten. Thus, a certain mouse movement now corresponds to a much smaller change in the controlled coordinate. The high resolution mode is intended for fine tuning of critical positions in the control sequence.

Specifying coordinates

The second method of controlling the arm is by specifying coordinate values. The current values of all the possible coordinates are listed in the coordinate window at the right of the screen in the MFE (see fig. 2-1). By pointing at one of these values with the cursor and pressing the leftmost mouse button, the user is prompted for a new value for that coordinate. When the value is entered the arm immediately moves to its new position.

Thus, by pointing at a *coordinate* (e.g. x) it will be controlled interactively with the mouse and by pointing at a *coordinate value*, a new value can be specified from the keyboard.

Interactive keyboard control

A coordinate can be selected with the up and down arrow keys on the keyboard . By pressing either the left or the right shift key this coordinate can be decreased or increased causing the arm to move accordingly. The currently selected coordinate is marked with a yellow asterisk to the left of the name of the coordinate.

Remote control

The fourth, and last, option in controlling the arm is by using a remote control that can be connected to the system. From this remote control not all coordinates are available, just the handoriented coordinates (see below) together with the hand movements. The remote control is intended for fine tuning of the arm positions in critical situations when it is necessary for the user to be very close to the hand to ensure accurate positioning.

Coordinate systems

There are three coordinate systems for moving the arm, one for moving the hand and one for moving both the arm and the hand.

The ranges for the coordinates given in fig. 2-2 to 2-5 gives the absolute maximum and minimum values that each coordinate can assume. These limits are not reachable at all times. It depends on the value of the other coordinates.

Cartesian coordinates

A Cartesian coordinate system is available for controlling the arm, allowing the arm to be moved along the x, y and z axes. The Cartesian coordinate system has its origin in the base of the robot colon, see fig. 2-2. x, y and z can be controlled individually and x and y can be controlled together (with the mouse).

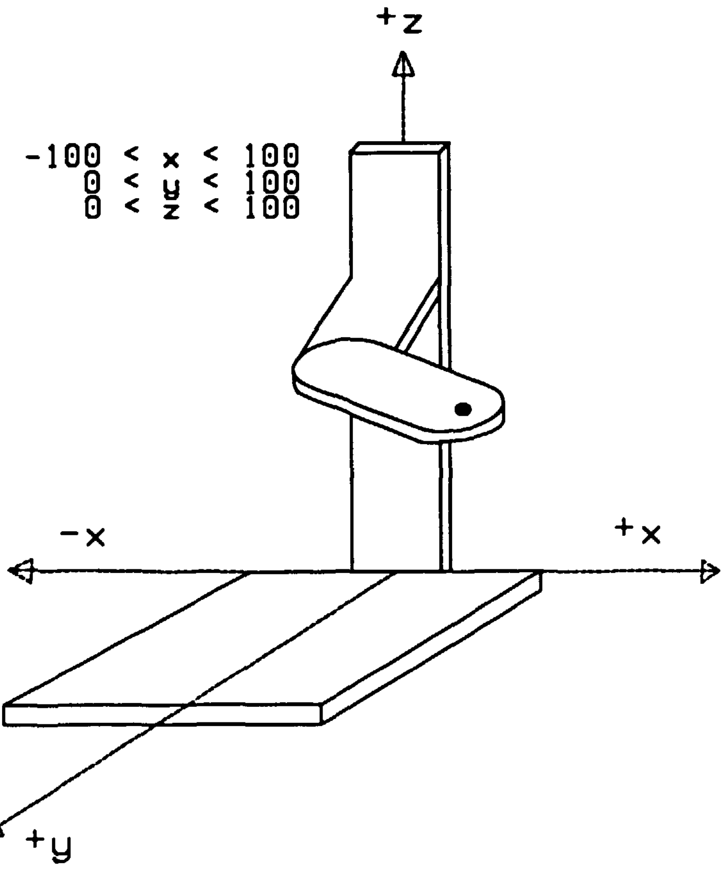

Figure 2-2 Cartesian coordinates

Cylindrical coordinates

The arm can also be controlled in a cylindrical coordinate system, i.e. r, ϕ and z. This is the natural coordinate system for the robot arm. Thus, controlling the arm with the mouse via cylindrical coordinates results in smooth movements.

Because of the design of the arm there are two possible orientations of the arm for almost every position in the horizontal plane. Therefore the program has to know which of these positions to use. The program uses a *preferable orientation*, chosen by the user, to select position. The preferred position can be either "left" or "right". In fig. 2-2 and 2-3 the arm is shown in a right oriented position. In the "Miscellaneous" menu (see sec. 2.1.5) there is an entry "Orient." which serves two purposes. It displays the current orientation and second, when selected toggles the preferred orientation to the other alternative. The next time the arm is moved, it is positioned according to the new preferred position. The orientation only has a meaning when the arm is controlled interactively. When a position is stored in the control sequence the positions of each of the motors in the arm are stored.

With a given orientation it is not possible to reach all positions. This is especially notable for the ϕ coordinate. The available range for ϕ depends on the orientation chosen. Thus, if a position can't be reached, changing the orientation may help.

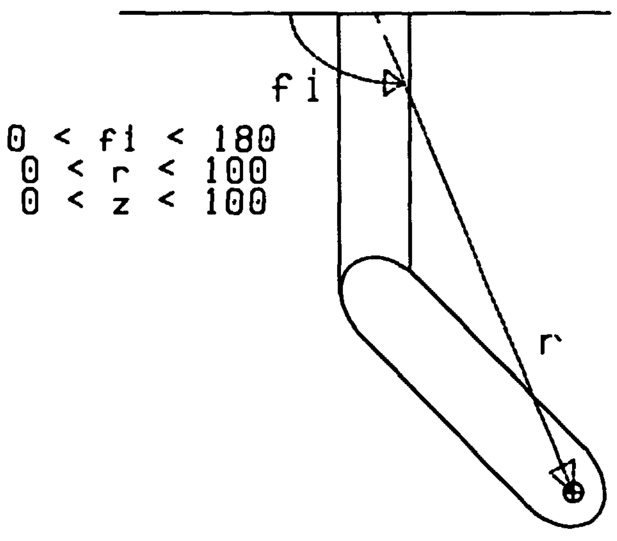

Figure 2-3 Cylindrical coordinates (top view)

Hand oriented coordinates

In the previous coordinate systems the arm has been moved independently of the hand's position. When moving the arm in one of these coordinate systems the hand just "comes along". In many cases though, one wants to move the arm relative to the hand. To satisfy this demand a hand oriented coordinate system has been provided . This is especially useful for the final tuning of the hand's position before gripping an object. In this case, move the hand in front of the object , open the gripper and move the hand forward until the desired position is reached. This is much more convenient than trying to figure out which of the other coordinates to change to reach the desired position.

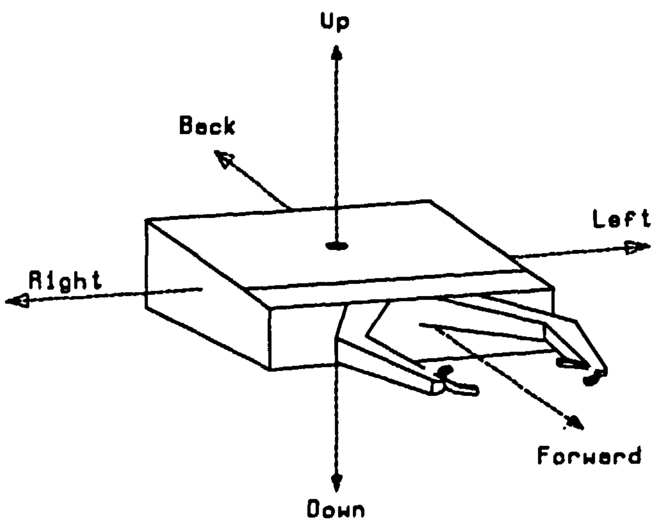

Figure 2-4 Hand oriented coordinates

Hand movements

The position of the hand can be changed independent of the arm. The hand has three degrees of freedom (yaw, pitch and roll) together with the gripper motion (see fig 2-5).

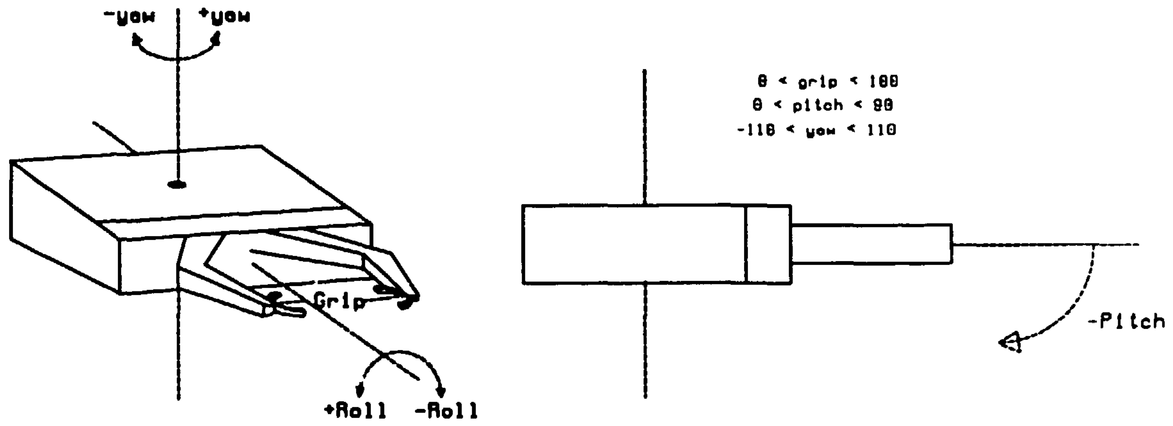

Figure 2-5 Hand movements

2.2 File structure

The files used and created by the CAMP programs are organized as shown in figure 2-6.

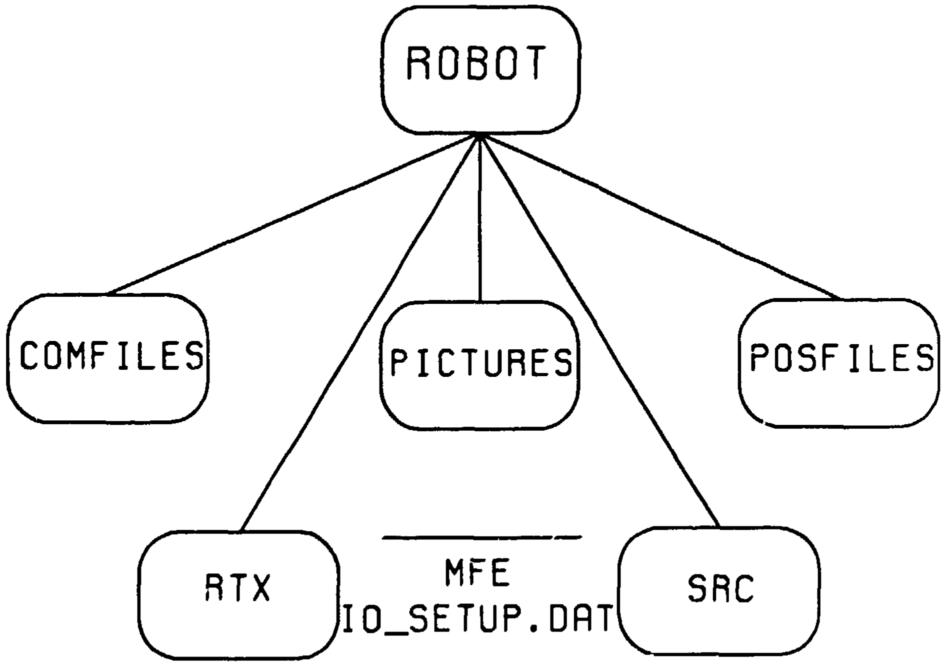

Figure 2-6 File structure of CAMP files and programs

The MFE program is situated together with the configuration file IO_SETUP.DAT, that describes the equipment that is connected to the system, in the main directory called ROBOT.

There are also a number of subdirectories in the ROBOT directory. The names and contents of these subdirectories are:

- o COMFILES: Here the command files generated by the MFE are stored. A command file must be placed in this directory and has an extension .ROB for the program to find it.
- o PICTURES: This is the directory for PaintBrush pictures. The picture files must have an extension .PCX.
- o POSFILES: Positions in the "position" menu are stored in this directory, when the save command in that menu is executed. Position files must have an extension .POS.
- o RTX: This directory contains some utility programs from the robot manufacturer.
- o SRC: This directory contains the source code for the MFE program.

3. Technical reference

This section is intended as a reference for the advanced user. Section 3.1 gives a description of the MFE. In this section the emphasis has been put on the architecture of the program and not its function. The aim is to provide the advanced user with enough information to be able to understand the architecture of the program in order to implement minor changes in the program code.

Section 3.2 gives a more thorough description of all the commands that can be given to the Multi Function Editor than was given in previous sections. The commands are listed in alphabetic order.

3.1 Multi Function Editor

The MFE program is written in Turbo Pascal from Borland Inc. Code that perform related tasks are put in separately compiled units.

Basic program architecture

The program is menu driven which means that program execution is controlled by menu selections. The MFE contain five pull down menus (Edit, Help, Positions, Miscellaneous and Store) and one static menu (Motion control). In the main loop the program waits for user input after which appropriate actions are taken, depending on the type of input. To each menu selection corresponds a well defined portion of the code.

Variables

Since the basic feature of the MFE is to generate control sequences the main variables in the program are concerned with storing these sequences. A control sequence is stored in three arrays:

- o COMMANDS: is an array of records. The index to the array is a step number in the sequence. Each element of the array contains two items. The first identifies the type of command, e.g. a robot position or a timedelay command. The second contains an index to another array that stores the data needed for this particular command. If the command is a robot position, then it contains an index to the array ROBOT_PARAM where the positions of each motor in the robot arm are stored. For other commands it contains an index to the array OTHER_COMMANDS. This is an array of strings that contains the command parameters.
- o ROBOT_PARAM: is an array of robot parameters. Each element contains the position of all motors in the arm together with the speed with which the arm will move to this position.

- o OTHER_COMMANDS: is an array of strings. Each element contain the command parameters.

At any time there is a current step number (shown in the "Step window", see fig. 2-1), which tells where in the control sequence the program is currently working. If no commands have been stored or no command files have been read from the disk the "Step window" will be empty. If the user wants to store a new step in the sequence the commands already stored in the COMMANDS array are pushed, starting at the current position, so that a new command can be inserted. When the command type has been stored the program obtains a free position in the adequate array (ROBOT_PARAM or OTHER_COMMANDS) by checking one of the variables max_robot_index or max_other_index. Thus, for each new command step that is stored, the parameters for the command are stored at the end of the respective arrays, independent of the command step's position in the sequence. This accelerates the insertion of new positions in the middle of already stored commands since the amount of data that has to be moved is reduced. The same is true when a command step is deleted. It is only deleted from the COMMANDS array, it remains in the ROBOT_PARAM or OTHER_COMMANDS arrays. This does not mean that deleted command steps are stored on disk when using the "Save" command.

All other variables are as far as possible, given self explanatory names. Single letter variable names have been avoided.

File structure

The source code of the MFE is distributed over several files, all situated in the SRC subdirectory:

- o MFE.PAS contains the source code for the main program. This file contains the code for initializing the system and the main loop with all the menus. There are no procedures in this file. In this file changes can be made to the initialization procedures and new commands can be added.

- o MFE.INC contains most of the procedures called upon from the main program in MFE.PAS. The most important of these procedures is probably the TRACE procedure that handles the interactive control of the arm with the mouse.

- o ROBOT.PAS is a unit that contains the low level procedures used for controlling the arm.

- o MENUE.PAS is a unit that contains the procedures used for generating the menus and help facilities.

- o BOARD.PAS is a unit that contains the procedures used to control the equipment on the tray and other "environmental" features, like showing pictures. This unit also reads the configuration file IO_SETUP.DAT that contains information on how the system is configured with valves, counters and other equipment.

- o EDITOR.PAS is a unit that contains the line editor procedure used for editing parameters given to commands like "show picture".

- o VERSION.INC is used for version control of the MFE program. As comments in this file, the version number of each of the above files, used for a specific version of the program, are shown. The file also contains the procedure "sign_on" that writes a message on the screen when the program is started. As soon as any of the above files have been changed VERSION.INC should be updated to record the changes.

Each of the procedures and functions in the above files are described in connection with the respective declarations in the files.

Recompilation

When a file has been changed the entire program has to be rebuilt. This is easily done. In the main directory (ROBOT) there is a configuration file for the Turbo Pascal compiler. If the compiler is started from within this directory all is set up correctly. To recompile the program, just give the "Make" command to the Turbo Pascal program. All the compiled files are stored in a subdirectory called OBJ in the ROBOT directory. The only file in this directory that is needed to run the program is MFE.EXE. The other files are kept to speed up recompilation.

3.2 Command reference

This section contains an alphabetic listing of all the commands that can be given from the menus in the Multi Function Editor. Text in italic must be entered by the user. Standard font text is written on the screen by the program when a command is selected. A vertical bar separates different choices. Square brackets indicates optional information. When there is no italic text in the command definition, indicating no input, but still two choices the selection of such a command will cause toggling between the two choices.

Autosave

Syntax: Autosave ON | OFF

Menu: Miscellaneous

Description: The autosave facility is used to automatically save the control sequence currently being edited in a file on the computer's harddisk. When autosave is on, the control sequence is saved at intervals specified by the user. By default autosave is performed every third new step added to the control sequence. The commands are stored in a file named "autosave" which, at a later time, can be read back with the load command. If the computer hangs this allows the sequence currently in the computer's memory to be restored. The current setting is shown in the menu.

Call

Syntax: Call: *filename*

Menu: Store

Description: The "call" command calls a subroutine, i.e. the file given as argument will be loaded and executed. When the subroutine is terminated the execution will resume in the main file. Subroutines can be nested in up to ten levels, i.e. a subroutine can call another subroutine, which can call another subroutine...(ten times).

Delete next

Syntax: Delete next or *<ctrl>D*

Menu: Edit

Description: This command deletes the next step in the control sequence. The "current position" pointer is not affected.

Do

Syntax: Do: *open* | *close* [*multi_valve_nr:*] *valve_nr*

Menu: Store

Description: Do controls the equipment on the tray. At present time the only action that can be performed is opening and closing of valves. Both single and multi valves can be controlled. The valve numbers used in this command are the ones assigned to each valve in the configuration file (see sec. 3.3.1).

Dos

Syntax: Dos

Menu: Miscellaneous

Description: This command is used to leave the Multi Function Editor temporarily and enter the Dos shell where MS - DOS commands can be given. To return to MFE type "exit" at the DOS level.

Edit/Real mode

Syntax: Edit | Real mode

Menu: Miscellaneous

Description: The Multi Function Editor can be used in two modes. In the edit mode the only command steps that are executed are robot positioning commands. This mode is used to develop the motion scheme of the arm without involving the equipment on the tray. In the real mode all command steps are executed, i.e. valves are controlled, pictures are shown and wait conditions are checked. The default mode is the edit mode but when the MFE is started with the "-run " switch the program starts in the real mode. The current setting is shown in the menu.

Error handling

Syntax: Error handling: *go filename* | *write 'text'*

Menu: Store

Description: Error handling is used to take care of errors that occur when a "wait for" function is executed, e.g. waiting for an activity maximum to be reached in a counter. An error means that the condition has not been met within the timeout period. An error handling command can only be placed immediately after a "wait for" command in the control sequence. With an error handling command one of two actions can be taken, either execution continues in another command file (the *go* statement) or a message is written on the screen (the *write* statement).

File information

Syntax: File information

Menu: Edit

Description: The file information command gives information about the command file that is being edited. The name of the file, the total number of steps and the current step is written in the command window.

Help

Syntax: Help

Menu: Help

Description: Help gives general information on how to operate the program. For help on a specific command the rightmost mouse button can be pressed when the command is highlighted.

Initialize

Syntax: Initialize

Menu: Miscellaneous

Description: This command initializes the arm. It should be used when the main power to the arm has been turned off or when the arm has lost it's orientation. It can also be used as a command in the control sequence to improve accuracy in multiple runs. After an initialization command has been given the user is asked if an initialize command should be stored in the control sequence. If the answer is affirmative, the arm is initialized each time the execution of the control sequence reaches this step.

Load command file

Syntax: Load: *filename*

Menu: Edit

Description: This command loads a Robot Command File into the Multi Function Editor so that it can be edited or run. It is assumed that the file is located in the COMFILES directory and has an extension .ROB.

Load position file

Syntax: Load: *filename*

Menu: Positions

Description: This command loads a position file into the "positions" menu. It is assumed that the file is located in the POSFILES directory and has an extension .POS.

MFE

Syntax: *MFE* [*filename* [*-run no_of_times*]]

Menu: At DOS level

Description: MFE starts the Multi Function Editor from a DOS shell. Optionally a *filename* (of a Robot Command File) can be given in which case that file is loaded into the MFE in the same way as with the "load" command in the edit menu. If the *-run no_of_times* option is used together with a filename the MFE starts in real mode, loads the specified file and executes the sequence the specified number of times. The automatic execution can be stopped by pressing any key. The MFE then completes the current command and enters edit mode.

New

Syntax: New

Menu: Edit

Description: New clears the control sequence currently being edited and returns the Multi Function Editor to the state where a new command sequence can be created.

Next

Syntax: Next or *<ctrl>N*

Menu: Edit

Description: Next executes the next command step. If in edit mode only robot positioning commands are executed, all other commands are just written on the screen in the step window. The "current position" pointer is advanced to the next step to be executed.

No. of runs

Syntax: No. of runs: *number*

Menu: Miscellaneous

Description: This command specifies the number of times the control sequence will be run when a "run" command is given. The default is that the sequence will be run only one time.

Orientation

Syntax: Orientation

Menu: Miscellaneous

Description: Orientation toggles the orientation of the arm between elbow left and elbow right. Since, in the horizontal plane, the arm can reach most positions in two ways there must be a way to choose between these two possibilities. This is done by selecting an orientation with this command. If a position can not be reached with one elbow position the orientation must be changed. When the orientation is changed the new setting will be used the next time the arm is moved interactively. If a robot positioning command in the control sequence is executed the orientation setting is updated to correspond to the orientation of the arm. The current setting is shown in the menu.

Previous

Syntax: Previous or *<ctrl>B*

Menu: Edit

Description: Previous executes the previous step in the control sequence. If in edit mode only robot positioning commands are executed, all other commands are just written on the screen in the step window. The "current position" pointer is moved back one step so that it points on the next step to be executed when stepping forward.

Quit

Syntax: Quit

Menu: Edit

Description: Exit the Multi Function Editor

Remember

Syntax: Remember: *name*

Menu: Positions

Description: This command memorizes the current position of the arm under the name specified. This name subsequently occurs at the bottom of the "positions" menu so that the position can be retrieved at a later time by selecting the name.

Replace

Syntax: Replace

Menu: Edit

Description: Replaces the current step (must be a robot position command) in the control sequence with the arm's current position.

Reset

Syntax: Reset

Menu: Motion control

Description: Moves the arm to the reset position and sets the current step pointer at the beginning of the control seq,ence. The next step that will be executed is the first step in the sequence.

Reset position

Syntax: Origin

Menu: Positions

Description: This command moves the arm to the reset position. This command does not affect the control sequence.

Restore

Syntax: Restore or *<ctrl>R*

Menu: Edit

Description: If the arm has been moved interactively this command restores the position of the arm to the position specified in the last executed robot positioning command in the control sequence. As soon as the arm has been moved interactively the text "Arm moved" is shown in the step window. It stays there until the position of the arm is stored, another positioning command in the control sequence is executed or a restore command is given.

Reverse

Syntax: Reverse

Menu: Edit

Description: Run through all steps in reverse order, starting at the current step, until the first step is reached. This command does not depend on the number of runs specified, it always stops when the first step is reached. When in edit mode only robot positioning commands are executed, all other commands are only written on the screen in the step window.

Run

Syntax: Run

Menu: Edit

Description: Run through the control sequence, starting at the next step. the number of times specified with the "no. of runs" command. The default is that execution stops when the last step is reached for the first time. If in edit mode only robot positioning commands are executed, all other commands are only written on the screen in the step window.

Save command file

Syntax: Save: *filename*

Menu: Edit

Description: This command saves the control sequence currently being edited. No extension is needed, the file is given an extension *.ROB* and is stored in the *COMFILES directory.*

Save position file

Syntax: Save: *filename*

Menu: Positions

Description: This command saves the positions currently present in the "positions" menu in a file so that they can be restored at a later time. No extension is needed, the file is given an extension .POS and is stored in the POSFILES directory.

Show counter

Syntax: Show counter: *counter_nr x_pos y_pos* [*color font size*]

Menu: Store

Description: Show counter controls the display of counter values on the screen. A specific counter can be turned on or off. When a counter has been turned on it will be displayed on the screen until it is turned of with another "show counter" command. Specify parameters as follows:

- o *counter_nr*: Specifies the counter the command should apply to. The counter number must be an integer in the range 1 - 3.
- o *x_pos*: The x coordinate (horizontal) of the counter value on the screen must be between 0 and 639, where 0 is the left edge and 639 the right edge of the screen.
- o *y_pos*: The y coordinate (vertical) of the counter value on the screen must be between 0 and 349, where 0 is at the top of the screen.
- o *color*: The display color of the counter value is an **optional** parameter. Default is that the counter value will be shown in the same color as all other text on the screen. The color is a number between 0 and 15. For further details on which colors belongs to what number, see the Turbo Pascal manual.
- o *font*: The font is an **optional** number that specifies the font of the counter value. The font number must be between 0 and 10, see Turbo Pascal manual for further details.
- o *size*: The size is an **optional** scaling number that specifies the size of the text. It is a number between 0 and 10. A size of 1 is the standard size you see on the screen in MFE. A size of 2 is twice that size and so forth. If the size is set to 0 the showing of the specified counter is turned off.

Show picture

Syntax: Show picture: *picturename*

Menu: Store

Description: This command shows a picture, previously created with the PC - PAINTBRUSH program, on the screen. The picture stays on the screen until a new picture is shown or the command execution is stopped by pressing a key. The picture name should be given without extension. It is assumed that pictures have an extension .PCX and that are located in the PICTURES directory. The MFE must be in real mode for this command to be executed.

Speed

Syntax: Enter new speed: *number*

Menu: Store

Description: Set the speed that the arm will use for subsequent arm movements when controlled interactively. When a robot position is stored as a step in the control sequence, the current speed setting is stored together with the arm's position. Thus when a robot position step is executed the speed stored in the step is used and the speed setting is changed.

Store position

Syntax: Store position

Menu: Store

Description: This command stores the current position of the arm as a step in the control sequence. This command can also be activated with function key 1 (F1).

Wait for

Syntax: Wait for: *max. counter_nr percent minlevel timeout* | *rate counter_nr level timeout* | *timeout time*

Menu: Store

Description: The wait for command is used to Insert a wait function in the control sequence. There are three possibilities. Wait for a maximum to be reached in one of the counters. Wait for one of the counters to reach a certain level and wait for a specified amount of time. The parameters should be specified as follows:

- For the *max.* function:
  - *counter_nr*: Specifies which counter to check. The counter number must be an integer in the range 1 - 3.
  - *percent*: Specifies the number of percents the counter reading has to decrease before it should be considered that a maximum has been reached.
  - *minlevel*: Specifies the minimum counter reading that must be reached before starting to look for a maximum.
  - *timeout*: The timeout time (in seconds) specifies the maximum time the program should wait for a maximum to occur. If the maximum does not occur within the timeout time an error is reported. The error can be taken care of with an "error handling" command just following the "wait for" command in the control sequence (see page 15 for more information on the error handling command).
- For the *rate* function:
  - *counter_nr*: Same as above
  - *level*: Specifies the level that should be reached
  - *timeout*: Same as above but the timeout error occurs if the specified rate is not reached within the timeout period.
- For the *timedelay* function:
  - *time*: Specifies a time in seconds. When this command is encountered in the control sequence command execution is suspended for the specified amount of time. The timedelay function can never trig an "error handling" command.

3.3 Hardware environment

The Multi Function Editor would be useless if it was not able to control something. Therefore it must be possible to connect different input and output devices to the system, e.g. valves, counters, mass flow regulators and ovens. How these devices connected to the system and how the software is informed about the environment is described in the following sections.

Configuration file

If you connect a new device to the system the software will not recognize the new device automatically. You must inform the program about what devices are connected to the system and where they are connected. The configuration file serves this purpose. It is called IO_SETUP.DAT and is placed in the same directory as the MFE. The configuration file is a ASCII file which can be edited by a standard editor.

In the configuration file there is one line for each device connected to the system (see appendix for an example of an configuration file). On this line the device type is specified together with information on where the device is connected (what pins on the I/O board). For some devices, e.g. valves, there is also a user id specified to be used as an identifier inside MFE. When giving commands like "open *number*". At present the only devices that are recognized are: singlevalve, multivalve and remote_ctrl. For the latest information on what equipment that can be connected to the system see the on-line help and the file IO_SETUP.DOC in the SRC directory.

Connecting I/O equipment to the system

All I/O equipment that is described in the configuration file is connected to a special parallel I/O extension board that has 32 inputs and 32 outputs. The pin numbers in the configuration file relates to pin numbers on the connectors on these boards. There can be up to five of these boards. For more information on the hardware used in the CAMP system, please contact the authors.

A. Example configuration file

This appendix contains an example of the configuration file IO_SETUP.DAT that the MFE program reads at startup to determine what I/O equipment that is connected to the computer. This file contains one line for each device. A line starting with # is ignored and can thus be used for comments. At present there are three different devices that can be connected to the system. The information needed for each of these is:

- **singlevalve**: A single valve needs just one output signal that controls if it is open or closed. A single valve is described in the setup file by starting a line with the keyword *singlevalve* Then there are three columns. The first specifies the number this valve has when referenced in the control sequence. The second column contains the board address that the valve is connected to and the third contains the pin number on that board to which the valve is connected.

- **multivalve**: A multi valve needs three kinds of control signals. First there is one output signal that is used to start the valve rotation. Then there is an input signal that indicates if the valve is in one of the eight possible positions and finally there are three inputs from which the valve number (0 - 7) can be read in binary form. A multivalve is described by starting a line with the keyword *multivalve*. Then there are seven columns. The first column specify the identity number of this valve when referred to in the control sequence. The next three columns specifies the board address of the boards to which the rotation control, position match and valve number signals are connected. The last three columns gives the pin numbers to which these signals are connected on each board. For the valve number signals the number given is the starting number to which the least significant bit is connected. The other two bits must be connected to the pins that follow directly after this pin.

- **remote_control**: The remote control uses five input signals and it is described by starting a line with the keyword *remote_ctrl*. Then there are two columns. The first specifies the address of the board to which the remote control is connected and the second gives the pin number of the pin to which the first signal from the remote control is connected. The other signals must be connected to the pins that follow consecutively.

Example configuration file:

```
# device   user number board number  pin number
#
 singlevalve   1           0            3
 singlevalve   2           0            4
 singlevalve   3           0            5
 singlevalve   4           0            6
 singlevalve   5           0            7
 singlevalve   6           0            8
 singlevalve   7           0            9
 singlevalve   8           0            10
 singlevalve   9           0            11
#
 multivalve    1         0 0 0        3 12 13
 multivalve    2         0 0 0        4 14 15
 multivalve    3         0 0 0        5 16 17
 multivalve    4         0 0 0        6 18 19
 multivalve    5         0 0 0        7 20 21
#
 remote_ctrl               0             22
```

INDEX

INDEX [continued]