ACLoff-line

Software Version 1.67

User's Manual

2nd Edition

Catalog #100051- Rev. C

ESHED ROBOTEC

Table of Contents

What is ACLoff-line?

ACLoff-line is a preprocessor software utility, which lets you access and use your own text editor to create and edit **ACL** programs even when the controller is not connected or not communicating with your computer.

After communication is established, the **Downloader** utility lets you transfer your program to the controller. The **Downloader** detects the preprocessor directives, and replaces them with a string or block of **ACL** program code.

ACLoff-line also enables activation of **ATS**, Advanced Terminal Software, for on-line programming and system operation.

ACLoff-line can be used to edit program files for either **Controller-A** or **Controller-B**. Be sure to use the proper **ACL** commands and format for the specific version of **ACL** used in your controller. Refer to the ***ACL** Reference Guide* supplied with the controller.

ACLoff-line version 1.67 is included in the **ATS** and the **Open-CIM** software packages, and provides the necessary tools for programming in the **Open-CIM** environment.

To help you learn how to operate **ACLoff-line**, it is suggested that you read through this manual, and perform the instructions given for the sample programs.

Activation

1. Be sure you have made all the required hardware connections, as described in the *User's Manual* supplied with your robot/controller.
2. Turn on the controller power switch if you want to download a file to the controller or if you want to activate **ATS**. You do not need to turn on the controller to edit a file.
3. **DOS:**

   Make the **ACLoff-line** directory or disk drive the current one.

   At the DOS prompt, activate **ACLoff-line**:

   - If the controller is connected to computer port COM1 (default), type:

     ```
     offline [Enter]
     ```

   - If the controller is connected to computer port COM2, type:

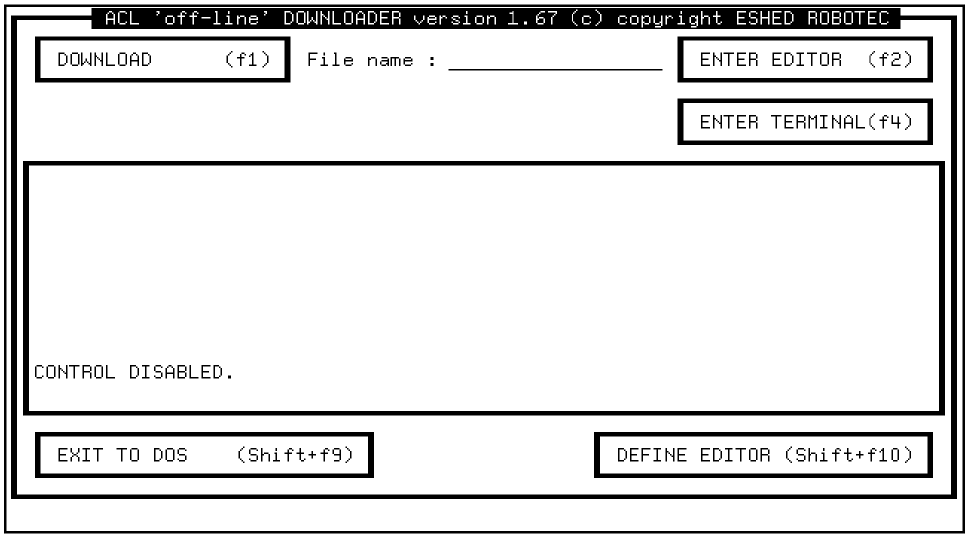
     ```
     offline /c2 [Enter]
     ```

   Windows:

   In Windows, create an icon for **ACLoff-line/Downloader**, and define the required properties and parameters for activation.

   Click on the **ACLoff-line/Downloader** icon.

   If **ACLoff-line** does not function properly in the Windows environment, make sure it is not running on the same serial port used by another software program (such as **ATS**).
4. Once the program has loaded, the **ACLoff-line** menu is displayed on the screen:

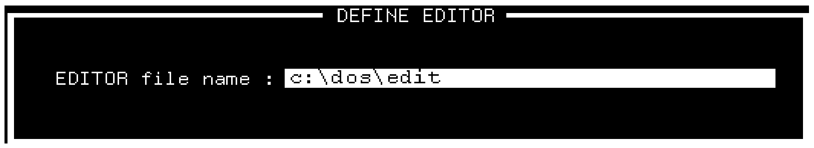

```
ACL 'off-line' DOWNLOADER version 1.67 (c) copyright ESHED ROBOTEC
DOWNLOAD      (f1)   File name : __________________   ENTER EDITOR  (f2)
                                                      ENTER TERMINAL(f4)

CONTROL DISABLED.

EXIT TO DOS    (Shift+f9)                DEFINE EDITOR (Shift+f10)
```

Defining the Text Editor

By default **ACLoff-line** uses EDIT.COM the text editor included in DOS. You may, however, use any other text editor with **ACLoff-line**.

To define a new text editor, press: **`[Shift]+F10`**

A small window appears on the screen and the text editor currently in use is displayed.



```
DEFINE EDITOR

EDITOR file name : c:\dos\edit
```

To change the editor, simply type in the name of the text editor you want to use, and press [Enter]. Be sure to include a drive and path if necessary. For example, type:

```
c:\brief\b  [Enter]
```

This editor now remains active until you exit to DOS.

To permanently change the editor, change the CONFIG.DLD file. Refer to the section "Additional **ACLoff-line** Files," later in this manual.

Activating the Editor

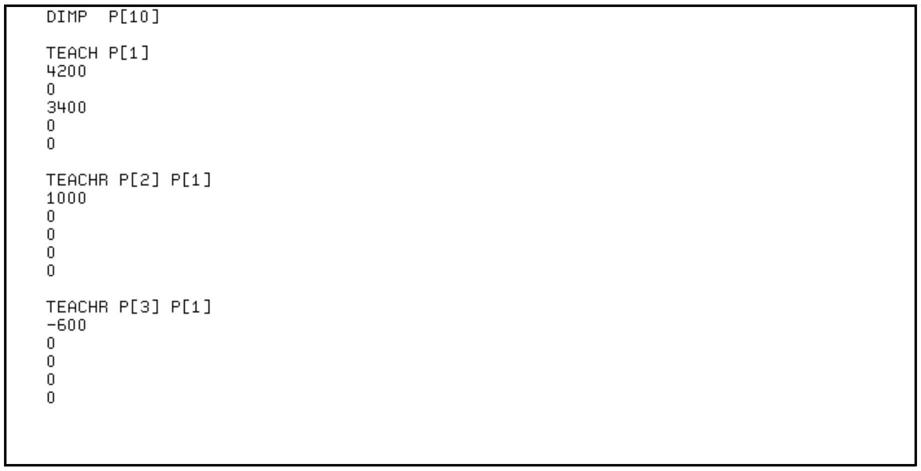
When the **ACLoff-line** menu is first activated, the cursor appears on the file name option line. Type the name of an existing file, or type a new name for the file you wish to create, and press [Enter].

To activate the text editor, press: **F2**

For example, to call up a file named DEMO.DNL, which is included in the **ACLoff-line** diskette, do the following:

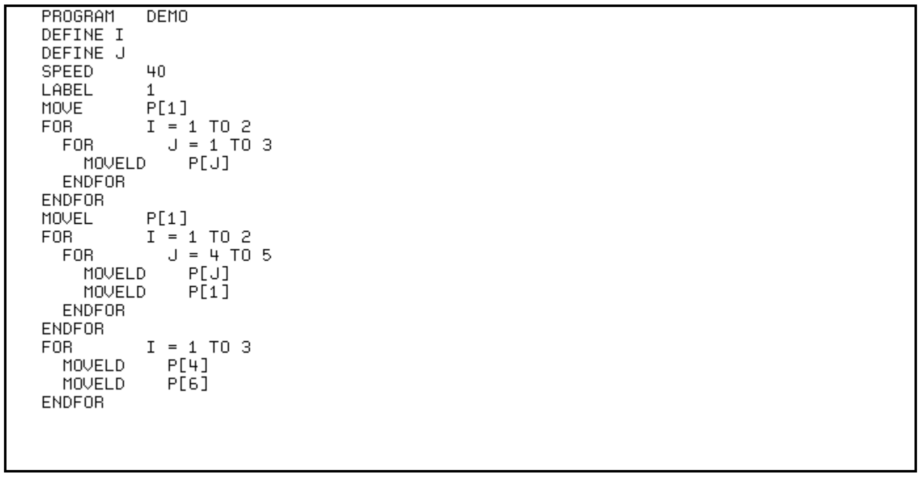
- Type DEMO.DNL and press [Enter].
- Press **F2** to activate the text editor.

The DEMO file will now be displayed, as shown below:



```
DIMP  P[10]

TEACH P[1]
4200
0
3400
0
0

TEACHR P[2] P[1]
1000
0
0
0
0

TEACHR P[3] P[1]
-600
0
0
0
0
```

If you are using DOS EDIT, or most any other text editor, use the [PgDn] key to scroll through the program text file until you find the line which reads PROGRAM DEMO. Use the up ↑ and down ↓ arrow keys to place that line at the top of the screen, as shown below.



```
PROGRAM   DEMO
DEFINE I
DEFINE J
SPEED     40
LABEL     1
MOVE      P[1]
FOR       I = 1 TO 2
  FOR       J = 1 TO 3
    MOVELD    P[J]
  ENDFOR
ENDFOR
MOVEL     P[1]
FOR       I = 1 TO 2
  FOR       J = 4 TO 5
    MOVELD    P[J]
    MOVELD    P[1]
  ENDFOR
ENDFOR
FOR       I = 1 TO 3
  MOVELD    P[4]
  MOVELD    P[6]
ENDFOR
```

As you can see, the format of a program written in **ACLoff-line** is similar to the format of a program when displayed by means of the **ACL** command LIST.

Note that the DEMO program was written for **Controller-A** (**SCORBOT-ER Vplus** and **SCORBOT-ER VII** robots). The position values defined in this program are not valid for **Controller-B**.

When you exit the editor, the **ACLoff-line** menu will reappear.

Prerequisite Program Data

ACL Commands

Before the **ACLoff-line** file is downloaded to the controller, you must be sure to define the positions and the global variables which will be called by the program(s) in the file.

ACL DIRECT mode commands can be included in your **ACLoff-line** file in order to enter position and variable data. It is recommended that you place these commands at the beginning of the file. These commands may not be used within a program; that is, they may not appear between a PROGRAM *name* command and its companion END command.

The data will be sent to the controller when the **ACLoff-line** file is downloaded to the controller.

This section briefly describes the commands used to prepare the data required by the controller in order to execute the program.

Refer to the ***ACL** Reference Guide* supplied with your controller for complete descriptions of the **ACL** commands and examples of use.

Define Position

The following commands are used to define positions.

`DEFP` *`pos`*	Defines a position.
`DIMP` *`pos`*`[`*`n`*`]`	Defines a position vector and its dimension.

Record Position

The following commands are used to set the coordinate values for the positions you have defined.

`SETPV` *`pos`*	Enters coordinates for position, in joint (encoder) values.
`TEACH` *`pos`*	Enters coordinates for position, in XYZ (Cartesian) values.
`HERER` *`pos`*	Enters coordinates for position, in joint (encoder) values, relative to current position of robot.
`TEACHR` *`pos`*	Enters coordinates for position, in XYZ (Cartesian) values, relative to current position of robot.

A program line containing a TEACH, TEACHR, SETPV or HERER command must be followed by a set of lines equivalent to the number of axes in the robot or device. For example:

- A TEACH command for a **SCORBOT-ER Vplus** position will be followed by five lines—values for each of the Cartesian, pitch and roll coordinates of the robot axes;
- A SETPV command for a **SCORA-ER 14** position will be followed by only four lines—values for axes 1, 2, 3 and roll coordinates of the robot axes.
- A position recording command for an XY-Table in group B will be followed by only two lines—values for two axes.

Note the difference in the expression of values in **Controller-A** and **Controller-B**.

Also note that pitch value is not defined for **SCORA-ER 14**.

Controller-A	**Controller-B (ACL 2.26 and later)**	**Result**
`DEFP PA`	`DEFP PA`	Defines position PA
`TEACH PA`	`TEACH PA`	Records coordinates for position PA
`2000`	`200`	X value = 200mm
`0`	`0`	Y value = 0mm
`400`	`40`	Z value = 40mm
`-900`	`-90`	Pitch value = -90°
`0`	`0`	Roll value = 0°

The following position recording commands are *not recommended* for use in **ACLoff-line**, because the position coordinates will be determined by the arbitrary location of the robot at the time the command is downloaded to the controller.

`HERE` *`pos`*

`HERER` *`pos2 pos1`*

`TEACHR` *`pos2 pos1`*

`HEREC` *`pos`* (available in **Controller-B** only)

Define Variables

The following commands are used to define global variables:

`GLOBAL` *`var`*	Defines a global variable.
`DIMG` *`var`*`[`*`n`*`]`	Defines a global variable array.

Note that the **Open-CIM** system contains variables whose names have the prefix $. The $ indicates it is a system-required variable which should not be manipulated by the user.

The following command format is used to give an initial value to the variable:

```
SET var = n
```

Constant ASCII Values

ACLoff-line allows you to assign a character's ASCII value to a variable. The command format requires the character to be within apostrophe marks, as follows.

`SET` *`var`* `= '`*`x`*`'`	*Var* is the variable; *x* is the character whose ASCII value is assigned to *var*.

For example:

`SET` *`var`* `= 'a'`	Sets the value of *var* to 65, the ASCII code for the character A. This command line will appear as SET VAR=65 after the program file is downloaded to the controller.

This feature is useful for preparing programs which include a GET command, as it enables you to see the actual characters expected or required by the command. For example:

```
GET KEY
IF KEY = 'X'        If X is pressed,
  GOTO 9            the program ends.
ENDIF
IF KEY = 'B'        If B is pressed, program B will run.
  RUN B
ENDIF
IF KEY = 'C'        If C is pressed, program C will run.
  RUN C
ENDIF
LABEL 9
END
```

Comments

ACLoff-line allows you to insert comments anywhere within a file.

Mark the beginning of a comment with a semicolon (;).

A comment is limited in length only by the number of spaces remaining until the end of the line.

Unlike **ACL** comment lines, which are preceded by an asterisk (*), **ACLoff-line** comments are *not sent* to the controller during downloading, and can be viewed only during off-line editing.

Preprocessing Directives

ACLoff-line offers a number of directives which function similarly to their counterparts in the C and Assembly languages. These are:

```
#DEFINE

#MACRO

#ENDM

#INCLUDE

#IF

#IFDEF

#IFNDEF

#ELSE

#ENDIF
```

When **ACLoff-line** text files are downloaded to the controller, these preprocessing directives are replaced by the associated program codes.

It is recommended that #DEFINE and #MACRO directions be contained within a .DMC file for global system access, and within a .DNL file for local use.

#INCLUDE directives should be included in a .DNL file. However, they may be included, and nested, in either .DMC or .DNL files. Refer to the section, “Saving a Program File” later in this manual.

#DEFINE and #UNDEF

#DEFINE directives enable the use of a symbolic constant instead of a string.

For example:

`#DEFINE FASTSP 80`	Preprocessing directive defines a value.
`SPEED .FASTSP`	**ACLoff-line** program line. You will use .FASTSP instead of a numeric speed value throughout your **ACLoff-line** programs.
`SPEED 80`	**ACL** program command line, after downloading to controller.

In this example, the #DEFINE directive allows you to globally change the value of fast speed. If you change the directive to #DEFINE FASTSP 60, all **ACLoff-line** references to .FASTSP will be be translated to a value of 60.

The period indicates a call to a symbolic constant. When the **Downloader** encounters the period, it substitutes the complete command string for the symbolic constant.

The defined name may contain 20 characters. The first characters must be a letter. Subsequent characters may be alphabetic or numeric, or an underscore. Other characters are not valid.

The length of the string to be replaced is limited to the number of spaces remaining until the end of the line.

For example:

`#DEFINE LIGHT  10`
`#DEFINE ON     1`
`#DEFINE OFF    0`
	Preprocessing directive: light is connected to output 10.
	1 = output on; 0 = output off.

`SET OUT[.LIGHT]=.ON`	**ACLoff-line** program line.
`SET OUT[10]=1`	**ACL** program command line, after downloading to controller.

In this example, the #DEFINE directive defines a light which is controlled by output 10. If the light is subsequently connected to another output, you need only to change one line in your **ACLoff-line** file—the #DEFINE directive. After downloading the file, all occurrences of .LIGHT will refer to the new output.

#UNDEF deletes the symbolic constant defined by a #DEFINE directive.

The **Open-CIM** software includes a file named DEVICE.DMC which contains the #DEFINE directives (symbolic constants) for all the devices (conveyor, robot, ASRS, etc.) used in the system.

#MACRO and #ENDM

#MACRO and #ENDM are the directives used to delimitate macro definitions.

For example:

```
#MACRO    LIGHTON
SET OUT   [4] = 1
DELAY 50
#ENDM

#MACRO    LIGHTOFF
SET OUT   [4] = 0
DELAY 50
#ENDM
```

These two macros define the output index and status which result in a light turning on and off, followed by a delay in program execution.

```
.LIGHTON
```

ACLoff-line program command line: turns on output 4; turns on light.

```
.LIGHTOFF
```

ACLoff-line program command line: turns off output 4; turns off light.

The period indicates a call to a macro. When the **Downloader** encounters the period, it substitutes the name of the macro with its complete definition.

The macro name may contain 20 characters. The first characters must be a letter. Subsequent characters may be alphabetic or numeric, or an underscore. Other characters are not valid.

The macro may include up to nine formal parameters: .1 .2 .3 .4 .5 .6 .7 .8 and .9.

The call to a macro may include up to nine parameters, each of which can have 20 characters. However, the length of the macro name and parameters is limited to the number of spaces remaining until the end of the line.

When the macro is downloaded, these parameters are replaced by corresponding strings.

For example:

```
#MACRO SETOUT
SET OUT [.1] = .2
#ENDM
```

The macro contains two formal parameters.

```
.SETOUT 5 1
```

ACLoff-line program line: when the SETOUT macro is called, the first parameter becomes output 5; the second parameter becomes 1 (output on).

```
SET OUT[5]=1
```

ACL program command line, after downloading to controller.

#INCLUDE

#INCLUDE statements contain the path and name of a file which will be downloaded to the controller together with the file which calls it.

For example:

```
#INCLUDE  C:\OPENCIM\SETUP\DEVICE.DMC

#INCLUDE  C:\OPENCIM\LIB\CIMSYS.DMC
```

#INCLUDE directives may appear anywhere in an **ACLoff-line** file. It is recommended that they reference DMC files and be placed at the beginning of DNL files.

#IF, #IFDEF, #IFNDEF, #ELSE, #ENDIF,

#IF is used for conditional program branching.

- If the result of the #IF expression is true, all lines included in the directive (until #ENDIF or #ELSE is encountered) will be downloaded to the controller.
- If the result of the #IF expression is false, none of the lines in the directive (until #ENDIF or #ELSE is encountered) will be downloaded to the controller.

For example:

```
#IF .__CONTROLLER = A
    SENCOM 1, 'S'
#ELSE
    SENDCOM 0, 'S'
#ENDIF
```

When included in a program which is suitable for either **Controller-A** or **Controller-B**, this routine will transmit a character on COM1 of **Controller-A**, or on COM0 of **Controller-B** (default COM ports). (Refer to the following section for an explanation of __CONTROLLER.)

#IFDEF and #IFNDEF can be used in place of #IF, as described above.

- #IFDEF is true if the symbolic constant has been defined.
- #IFNDEF is true if the symbolic constant has not been defined.

Do not use the period (.) in either the #IFDEF or #IFNDEF statement. Otherwise the symbolic constant will be expanded during downloadeding.

The expression used in an #IF directive may be a comparison of two strings or the result of a mathematical operation. If the result is 0, the expression is false; otherwise the expression true.

For example:

```
#DEFINE A XYZ
#IF .A > ABC
```

Defines A as "XYZ" and
compares the values of "XYZ" and "ABC."

`#IF (.A = 1) | (.A = XYZ)`	Condition is true when either A=1 or A=“XYZ”.

`#IF ._ _GROUP_B`
`    DEFPB POSB1`
`#ENDIFS`
	The position POSB1 for group B will be defined only if axis control group B is defined.

`#IF (.A+.B)>(.C*2)&(.B>0)`	The result of the expression is determined by the values of A, B and C.

Predefined Symbolic Constants

ACLoff-line provides a number of predefined symbolic constants which can be used to customize programs. These variables have two underscores as a prefix (_ _).

`_ _CONTROLLER`	Either A or B, according to the actual controller to which the program will be downloaded.
`_ _NEWTP`	Defined only when using **Controller-B** with **ACL** EPROM version 2.28 or later and teach pendant.
`_ _GROUP_A`	The number of axes in control group A.
`_ _GROUP_B`	The number of axes in control group B. If group B is not defined, _ _GROUP_B=0.
`_ _GROUP_C`	The number of axes in control group C. If no axes are installed in group C, _ _GROUP_C=0.
`_ _GRIPPER`	The number of the axis to which the gripper is connected, according to the controller configuration. If no gripper is configured, _ _GRIPPER=0.
`_ _ROBOT`	The type of robot being used, according to the controller configuration: 2 =PERFORMER-MK2 5 =SCORBOT-ER Vplus (or ER V) 7 =SCORBOT-ER VII 9 =SCORBOT-ER IX 14 =SCORA-ER 14

`_ _FILE`	Name of a downloaded file. For example:
	`PRINTLN "SOURCE FILE IS"`
	`PRINT " ._ _FILE"`
	When encountered, displays the name of the downloaded file which contains the **ACL** program currently being executed.

If the file is called by an #INCLUDE directive (or a nested #INCLUDE directive), the variable receives the name of the file where the nesting originates.

Program Editing

Once you have defined the required program data, you may proceed to write and edit programs. This phase of programming is comparable to editing a program when working on-line with the controller in the **ACL** EDIT mode.

ACLoff-line allows you to use all **ACL** EDIT mode commands. The command lines are not entered directly to the controller, but will be sent to the controller when the program file in which it is contained is downloaded to the controller.

Define Program

The following format is used to define and mark the beginning of a program:

```
PROGRAM name
```

To automatically allow or prevent the overwriting of programs during downloading, you may include an overwrite switch, /Y or /N, after program names within your text files.

`/Y`	Allows program to be overwritten during downloading.
`/N`	Program will not be overwritten during downloading.

For example:

```
PROGRAM PICK1 /N
```

When overwrite switches are included in the text file, the /Y and N/ overwrite switches used during command line activation of **ACLoff-line** will be ignored. See the section, **"ACLoff-line** Command Line Options," later in this manual.

Edit Program

Write and edit the program using any **ACL** EDIT mode command.

Tabulation (indentation) can be included, although it is not required. The controller will format the programs according to **ACL** syntax rules during the downloading procedure.

Private variables and variable arrays must be defined within the program itself. Use the commands:

`DEFINE` *pvar*	Defines a private variable.
`DIM` *pvar*[*n*]	Defines a private variable array.

Refer to the ***ACL** Reference Guide* supplied with your controller for complete descriptions of the **ACL** commands and examples of use.

End Program

The following command line is used to mark the end of a program:

```
END
```

Saving a Program File

Save your program as an ASCII text file. You may save the file under the same name you used to open the file, or save as a different file.

For consistency, and to simplify technical support, it is recommended that you use the following file name extensions.

`.DNL`	File containing **ACL** source code (programs, positions, variables) which will be downloaded to controller. Also contains #INCLUDE directives for downloading definitions and macros.
`.DMC`	File containing #DEFINE and #MACRO directives.

When you have completed editing and saved your program, exit your editor. The **ACLoff-line** menu will reappear.

Sample Program

The file you will create in this example contains a program named MV5 for **Controller-A**. (For **Controller-B**, adjust the values.)

- Activate the **ACLoff-line** menu.
- Type and [Enter] LEARN.DNL on the file name option line to create a new file.
- Press F2 to activate the text editor.
- Type and [Enter] the following program lines. (The comments are optional.)
- Save the file.

```
DEFP PP           ; defines position PP.
DIMP S[2]         ; defines vector S: two positions: S[1] and S[2].

TEACH PP          ; records Cartesian values for position PP.
2000
0
500
-900
0

SETPV S[1]        ; records joint values for position S[1].
0
2500
0
0
0

SETPV S[2]        ; records joint values for position S[2]
1000
3000
-200
100
-100

GLOBAL IND        ; defines global variable IND.
SET IND=1         ; sets initial value of IND to 1.
```

```
PROGRAM MV5        ; MV5 moves the robot
DEFINE J           ; from PP to S[1] and S[2]
FOR J=IND TO 2
    MOVE PP
    OPEN
    MOVELD S[J]
    CLOSE
    ENDFOR
END
```

J is a private variable used as the loop counter. The program executes a repeating movement: the robot is sent to a constant position PP, where the gripper opens; then it is sent to one of two positions S[1], where the gripper closes. The sequence is repeated, and the robot is sent to the second position S[2].

Downloading a Program

Before you transfer a program file to the controller, make sure the name displayed in the **ACLoff-line** menu is the file you want to download.

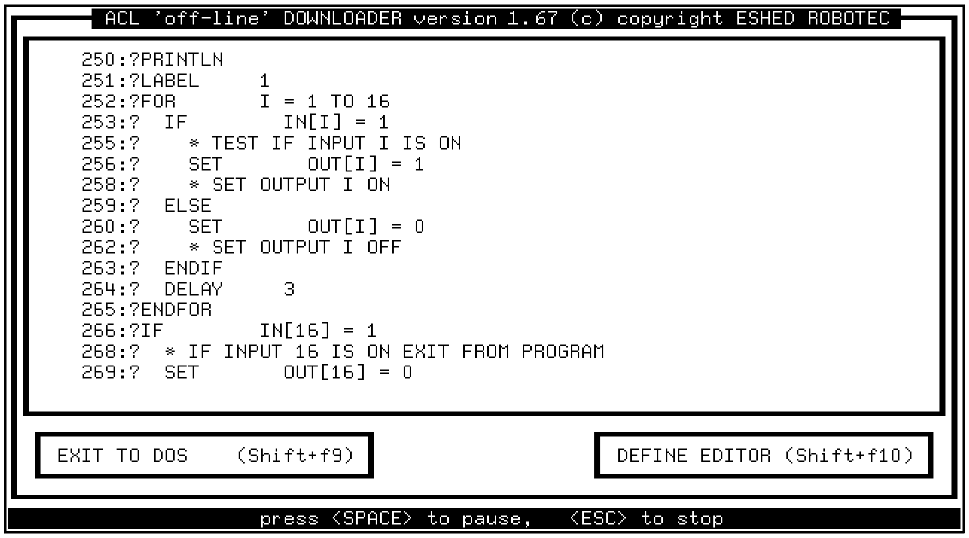
Also make sure the controller and computer are properly connected and the controller has been switched on.

In the following example, we will use the file DEMO.DNL. Type and enter DEMO.DNL on the program name option line.

To activate the **Downloader**, press: **`F1`**

Pressing F1 will abort any programs currently being executed by the controller.

The window on the screen expands. All program lines are displayed on the screen as they are sent to the controller.



```
ACL 'off-line' DOWNLOADER version 1.67 (c) copyright ESHED ROBOTEC

250:?PRINTLN
251:?LABEL     1
252:?FOR       I = 1 TO 16
253:?  IF        IN[I] = 1
255:?    * TEST IF INPUT I IS ON
256:?    SET       OUT[I] = 1
258:?    * SET OUTPUT I ON
259:?  ELSE
260:?    SET       OUT[I] = 0
262:?    * SET OUTPUT I OFF
263:?  ENDIF
264:?  DELAY     3
265:?ENDFOR
266:?IF        IN[16] = 1
268:?  * IF INPUT 16 IS ON EXIT FROM PROGRAM
269:?  SET       OUT[16] = 0

EXIT TO DOS    (Shift+f9)                DEFINE EDITOR (Shift+f10)

press <SPACE> to pause,   <ESC> to stop
```

At the bottom of the screen you see the prompt:

```
Press <SPACE> to pause. <ESC> to stop
```

Press the space bar to temporarily halt the downloading procedure. This will allow you to read the lines or messages displayed on the screen. Press any key to continue the downloading.

Pressing <Esc> will abort the downloading procedure.

When the downloading is completed, the **ACLoff-line** menu reappears.

Try to download the file called LEARN.DNL, which you previously created.

Downloading Messages

Following is a list of messages which may appear during downloading. When two messages are shown, the first is the one displayed by **Controller-A**, while the second is the one displayed by **Controller-B**.

- If the **Downloader** encounters a program name which already exists in the controller, it prompts you to confirm the overwrite:

  ```
  PROGRAM DEMO ALREADY EXISTS, OVERWRITE IT (Y/N)? N
  ```

- If the **Downloader** encounters a position or variable name which has already been defined, or if an axis number following the DEFPC command is incorrect, the following message appears:

  ```
  *** ERROR *** Name already in use or bad axis
  ```

  ```
  Name already in use or invalid axis.
  ```

- If too few lines are given after the TEACH or TEACHR commands, the missing lines are assigned a value of 0 and a message is displayed:

  ```
  <<MISSING NUMBER, USE ’0’>>
  ```

- If too many lines are given, the program ignores the extra ones, and displays:

  ```
  <<EXTRA NUMBER, IGNORE>>
  ```

- If the **Downloader** reaches the end of the file but has not encountered the END command, a message appears:

  ```
  <<MISSING END>>
  ```

- If the **Downloader** encounters incorrect **ACL** commands, such as command lines containing undefined variables, the downloading is stopped, and the **ACL** error message is displayed, followed by the message:

  ```
  FATAL ERROR DETECTED , DOWNLOAD STOPPED
  ```

- If the **Downloader** encounters unclosed loops (IF without ENDIF, for example), or a GOTO command whose LABEL has not been defined, the following is displayed:

  ```
  PROGR is not valid
  ```

  ```
  Invalid program.
  ```

- When a program has been downloaded successfully, the message is:

  ```
  PROGR is valid
  ```

Activating ATS

After you have downloaded a file, it is recommended that you check the controller contents.

To activate the Advanced Terminal Software (**ATS**), press: **F4**

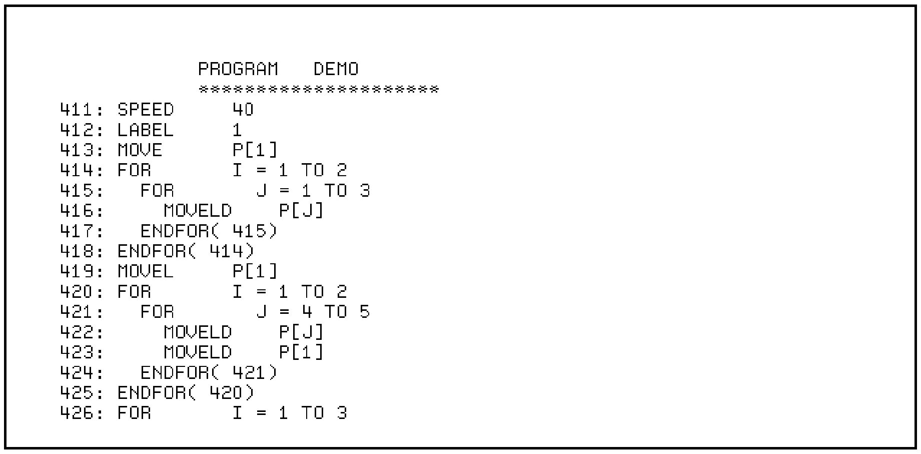
Pressing F4 activates the terminal emulation program defined in the CONFIG.DLD file. By default, it is **ATS**.

Once you have activated **ATS** you can then view, edit and run your program using the standard **ACL** commands.

To view the program DEMO, type the command:

```
LIST DEMO
```

The following will be displayed on your screen.



```
            PROGRAM   DEMO
            *********************
411: SPEED     40
412: LABEL     1
413: MOVE      P[1]
414: FOR       I = 1 TO 2
415:   FOR       J = 1 TO 3
416:     MOVELD    P[J]
417:   ENDFOR( 415)
418: ENDFOR( 414)
419: MOVEL     P[1]
420: FOR       I = 1 TO 2
421:   FOR       J = 4 TO 5
422:     MOVELD    P[J]
423:     MOVELD    P[1]
424:   ENDFOR( 421)
425: ENDFOR( 420)
426: FOR       I = 1 TO 3
```

Additional ACLoff-line Files

CONFIG.DLD

The CONFIG.DLD file is an optional file which the user may create using any text editor. It will contain two lines:

- The first line is the command which activates the text editor program. For example:

```
EDIT          or          C:\DOS\EDIT
```

- The second line is the command which activates the terminal emulation program (**ATS**). For example:

```
C:\ATS\ATS    or          C:\ATS\TERM_ACL /C2
```

When **ACLoff-line** is loaded, it searches for the CONFIG.DLD file in the current directory. If it exists, the editor and the terminal defined in the file are used by **ACLoff-line**.

If the /E switch is used in the command line, the editor defined in the CONFIG.DLD file will be ignored.

If the /T switch is used in the command line, the terminal defined in the CONFIG.DLD file will be ignored.

REPORT.DLD

A file named REPORT.DLD is created whenever an **ACLoff- line** file is downloaded. This file contains all the program lines which are sent to the controller, all the controller's responses and additional **ACLoff-line** information. This report file enables you to analyze your programs.

The existing contents of the REPORT.DLD file are erased the first time you activate the downloading procedure during a working session. Each subsequently downloaded file is added to the REPORT.DLD file, until you exit to DOS.

In the REPORT.DLD file, a certain character precedes some lines, indicating the type of directive which affected it.

+ Indicates a line which was added to the program when a #MACRO was called.

! Indicates a line in the program which was altered when a #DEFINE directive was expanded.

– Indicates a line in the program which will be skipped according to an #IF directive.

DEMO.DNL

DEMO.DNL is a demonstration file, for **Controller-A** only, which is similar to the DEMO.CBU file included in the **ATS** for **Controller-A** diskette. However, this file is intended for downloading through **ACLoff-Line**.

ACLoff-line Command Line Options

You can activate **ACLoffline** directly from the DOS command line or from the Windows icon with switches for a number of options. The format is as follows:

```
OFFLINE [name.ext [/R [/H] [/Y] [/N] [/A]]] [/D name value]
        [ /C] [ /E] [ /T]
```

Downloading Options

name.ext	The name and extension of the file to be used. Include the path, if necessary.
/R	Directly activates the downloading procedure without activating the **ACLoff-line** software. /R activates all other command line switches. If the /R switch is omitted, all other switches will be ignored. Use this option only with programs which have already been downloaded once and are known to be valid.
/H	During downloading, program lines are not displayed on screen. This shortens downloading time.
/Y	Programs which exist in the controller will be overwritten during downloading without a prompt for user confimation. This switch is ignored if the program name within the download file includes an overwrite switch.
/N	Programs which already exist in the controller will not be overwritten during loading. This switch is ignored if the program name within the download file includes an overwrite switch.
/A	During downloading, the contents of the file will be added to the end of the REPORT.DLD file. (The existing contents of the REPORT.DLD file will not be erased.)
/D *name value*	This switch defines a symbolic constant before downloading the file, as if a #DEFINE directive were included in the file. This switch can be used repeatedly in one command line.

Additional Options

`/C#`	Defines the computer's RS232C port number to be used for communication with the controller. For example: /C2
`/E`*`editor`*	The name and path of the text editor to be used. For example: /EC:\BRIEF\B
`/T`*`terminal`*	The name and path of the file which activates the terminal or the terminal emulation software: For example: /TC:\ATS\TERM_ACL