FANUC FLEXIBOWL PLUGIN

This Plugin was born with the idea of communicating quickly and safely with Flexibowl through FANUC robots.
The Plugin requires the "Fanuc User Socket Messaging" license for correct operation.

Karel
FANUC

STEP 1:

CONFIGURING THE SOCKET MESSAGING OPTION

Overview

In order to use Socket Messaging, you need to configure the following network hardware and software parameters:

• On the client,

— The IP address or name of your server

— The port on the server that you want to use for socket messaging.

Setting up a Client Tag

You need configure the client tags you want to use for socket messaging.

Note If the client tags you want to use are being used by a network protocol other than TCP/IP, you need to undefine the tags before they can be used for socket messaging.

Procedure To Setting up a ClientTag

Conditions

• The tag you want to set up is not configured to be used by another device on your network.

Steps

1. Cold start the controller.

a. **On the teach pendant**, press and hold the SHIFT and RESET keys. Or, **on the operator panel**, press and hold RESET.

b. While still pressing SHIFT and RESET on the teach pendant (or RESET on the operator panel), turn on the power disconnect circuit breaker.

c. Release all of the keys.

2. On the teach pendant, press MENUS.

3. Select SETUP.

4. Press F1, [TYPE].

5. Select Host Comm.

6. Press F4, [SHOW].

7. Choose Clients.

8. Move the cursor to the tag you want set up for Socket Messaging, and press F3, DETAIL.

You will see screen similar to the following.

```
SETUP Tags

  Tag C3:

  Comment:            ****************
  Protocol Name:      ********
  Current State:      UNDEFINED
  Startup State:
  Server IP/Hostname:****************
  Remote Path/Share: ****************
  Port:               *****
  Inactivity Timeout:  15 min
  Username:                  anonymous
  Password                  **********
```

9. Move the cursor to the Protocol Name item, and press F4, [CHOICE].
10. Select SM.
11. Move the cursor to the Startup State item, press F4, [CHOICE], and choose DEFINE.
12. Move the cursor to the Server IP/Hostname item, and press ENTER.
13. Type in hostname or IP address the of the remote host server you want to use for socket messaging.
Note If you are not using DNS, you must add the remote host and its IP address into the host entry table.
14. Press F2, [ACTION], and select DEFINE.
15. Set the system variable:
a. Press MENUS.
b. Select NEXT.
c. Select SYSTEM, and press F1, [TYPE].
d. Select Variables.
e. Move the cursor to $HOSTC_CFG, and press ENTER.
f. Move the cursor to the structure corresponding to the tag selected in Step 8 . For example, if you are setting up tag C3, move the cursor structure element [3], as shown in the following screen.

```
SYSTEM Variables
$HOSTC_CFG
  1    [1]            HOST_CFG_T
  2    [2]            HOST_CFG_T
  3    [3]            HOST_CFG_T
  4    [4]            HOST_CFG_T
  5    [5]            HOST_CFG_T
  6    [6]            HOST_CFG_T
  7    [7]            HOST_CFG_T
  8    [8]            HOST_CFG_T
```

g. Press ENTER. You will see a screen similar to the following.

```
SYSTEM Variables
$HOSTC_CFG[3]
  1 $COMMENT         *uninit*
  2 $PROTOCOL        'SM'
  3 $PORT            *uninit*
  4 $OPER            3
  5 $STATE           3
  6 $MODE            *uninit*
  7 $REMOTE          *uninit*
  8 $REPERRS         FALSE
  9 $TIMEOUT         15
 10 $PATH            *uninit*
 11 $STRT_PATH       *uninit*
 12 $STRT_REMOTE     *uninit*
 13 $USERNAME        *uninit*
 14 $PWRD_TIMOUT     0
 15 $SERVER_PORT     0
```

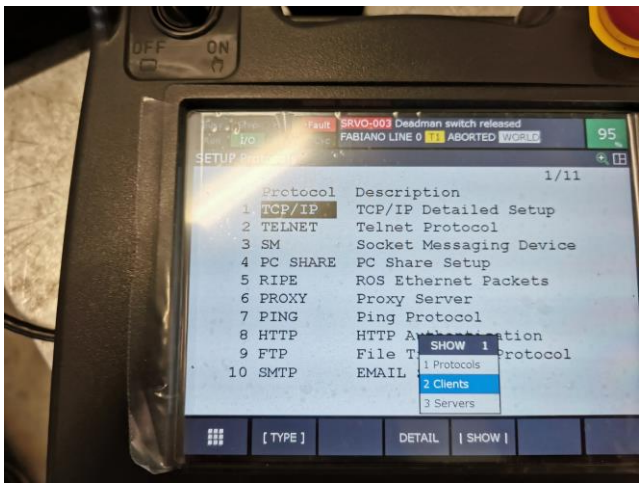
h. Move the cursor to $SERVER_PORT. Type in the name of the TCP/IP server port you want to use for socket messaging. The client tag is now ready to use from a KAREL program.

Karel
FANUC

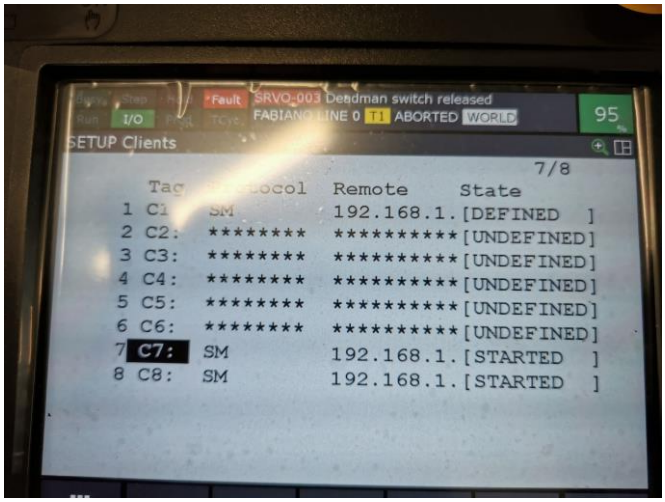
The following will indicate how to set the Client and the Karel code to communicate with the FlexiBowl. This script can be called to move the FlexiBowl or to put it in a parallel task, and to regulate the execution through a traffic light.
The code will receive a command to execute, and return a string with the response from the FlexiBowl.

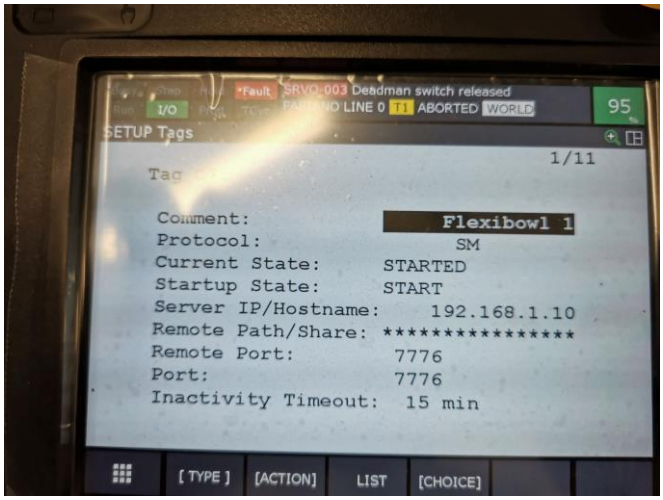
STEP 2:

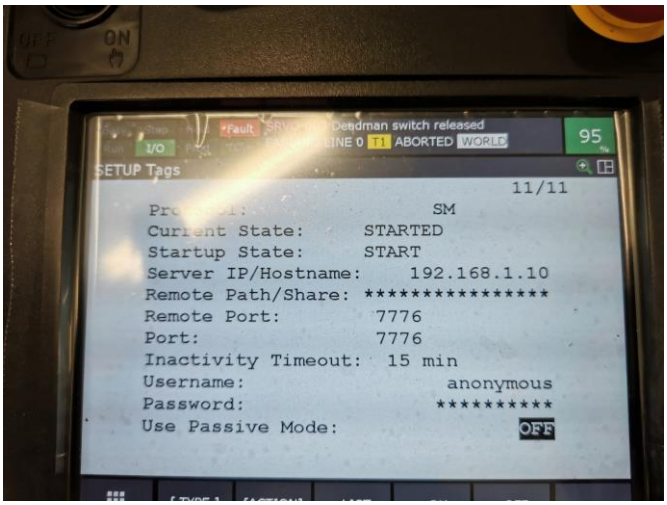
Host Client C7 configuration to communicate with the FlexiBowl address: 192.168.1.10



Karel
FANUC


SRVO-003 Deadman switch released
ABORTED
95
SETUP Clients
7/8
Tag Protocol Remote State
1 C1: SM 192.168.1.[DEFINED]
2 C2: ******** **********[UNDEFINED]
3 C3: ******** **********[UNDEFINED]
4 C4: ******** **********[UNDEFINED]
5 C5: ******** **********[UNDEFINED]
6 C6: ******** **********[UNDEFINED]
7 C7: SM 192.168.1.[STARTED]
8 C8: SM 192.168.1.[STARTED]

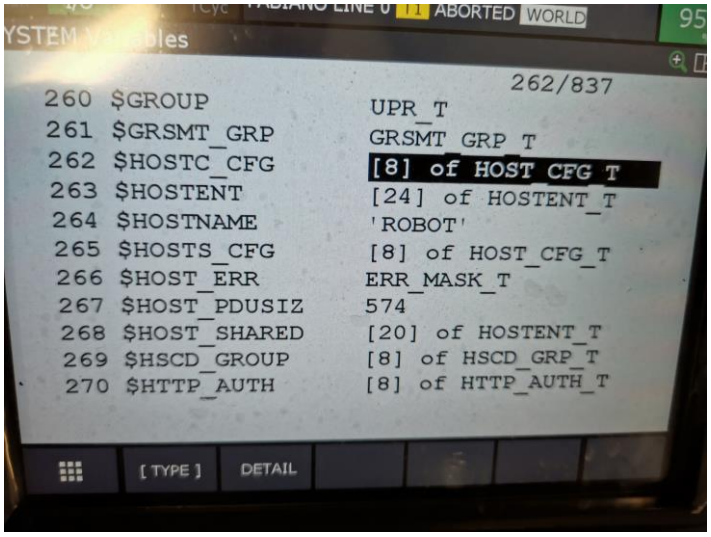

SRVO-003 Deadman switch released
ABORTED
95
SETUP Tags
1/11
Comment: Flexibowl 1
Protocol: SM
Current State: STARTED
Startup State: START
Server IP/Hostname: 192.168.1.10
Remote Path/Share: **********
Remote Port: 7776
Port: 7776
Inactivity Timeout: 15 min
[TYPE]
[ACTION]
LIST
[CHOICE]

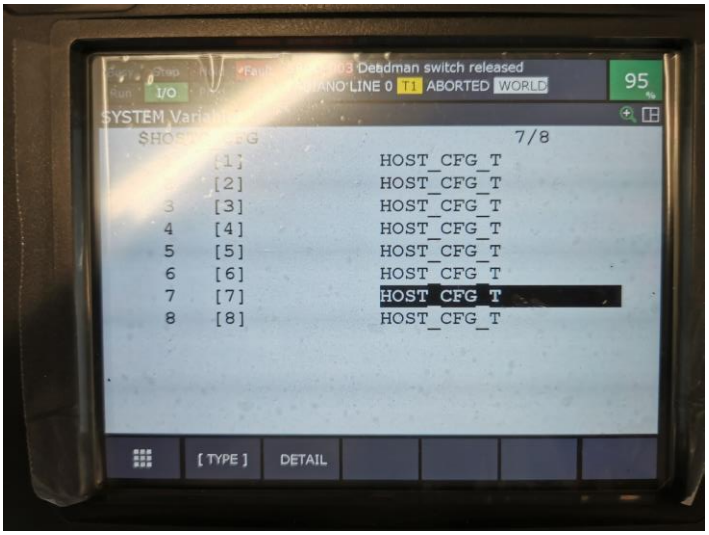

Deadman switch released
ABORTED
95
SETUP Tags
11/11
SM
Current State: STARTED
Startup State: START
Server IP/Hostname: 192.168.1.10
Remote Path/Share: **********
Remote Port: 7776
Port: 7776
Inactivity Timeout: 15 min
Username: anonymous
Password: **********
Use Passive Mode: OFF

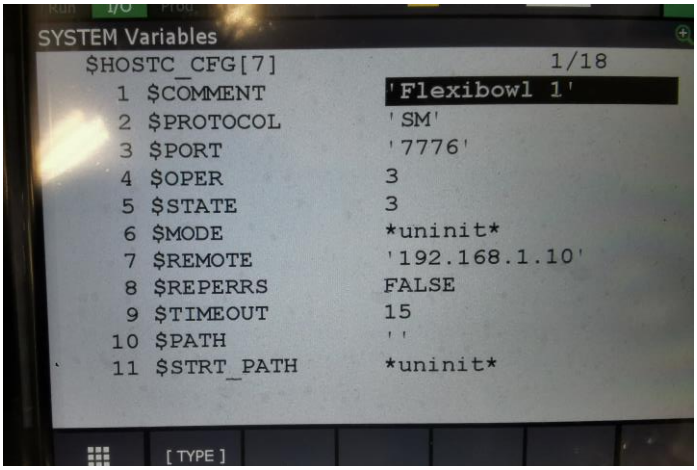
Karel
FANUC

STEP 3:

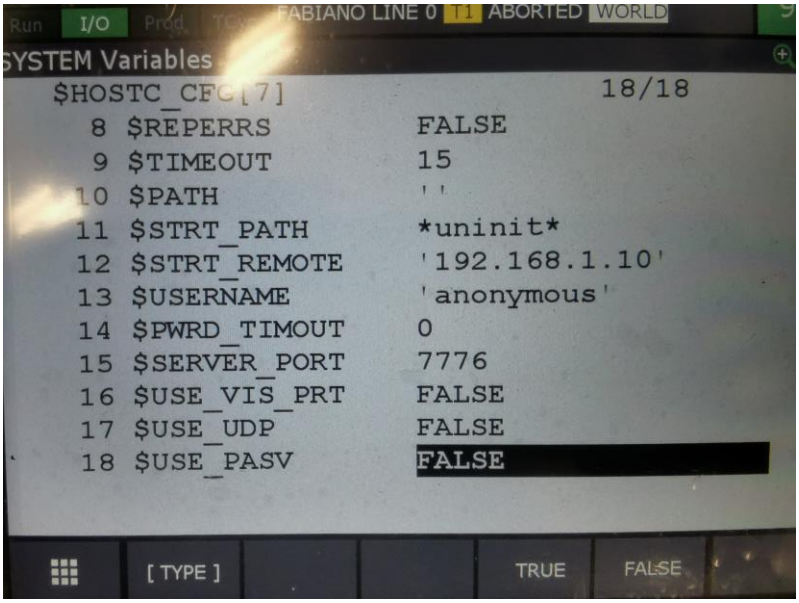
System Variable configuration







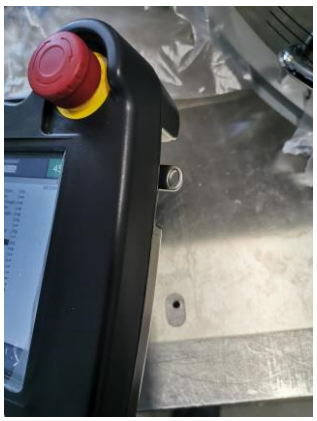
Karel
FANUC

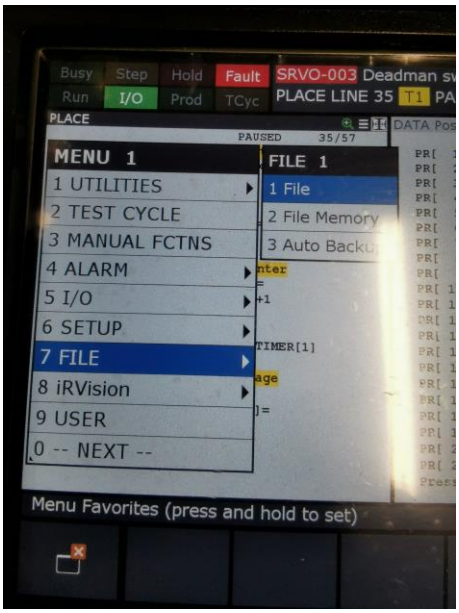


STEP 4:

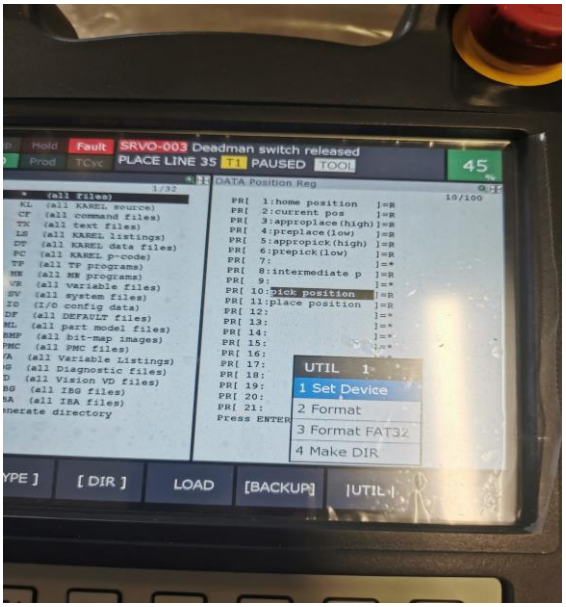
File Karel

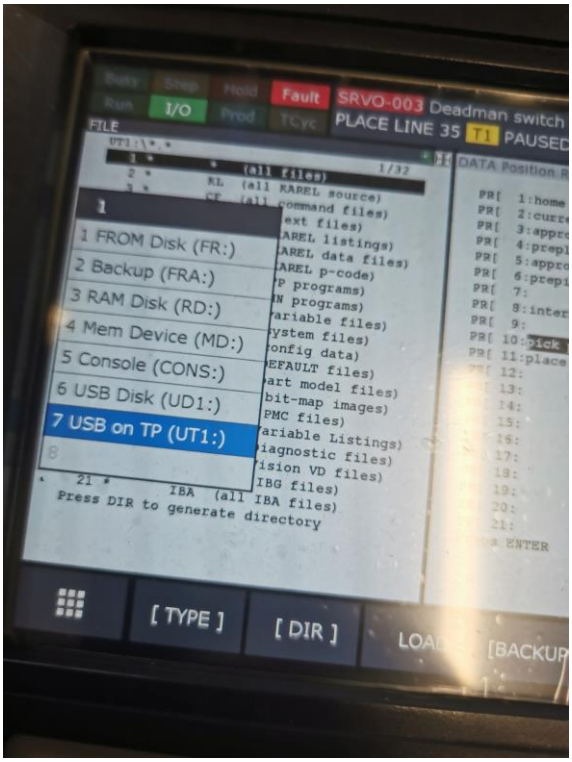
Once the client is configured we insert the FLB_PLUGIN.PC file in the controller.
To do this, copy the file into a USB stick and connect it to the pendant's USB port.



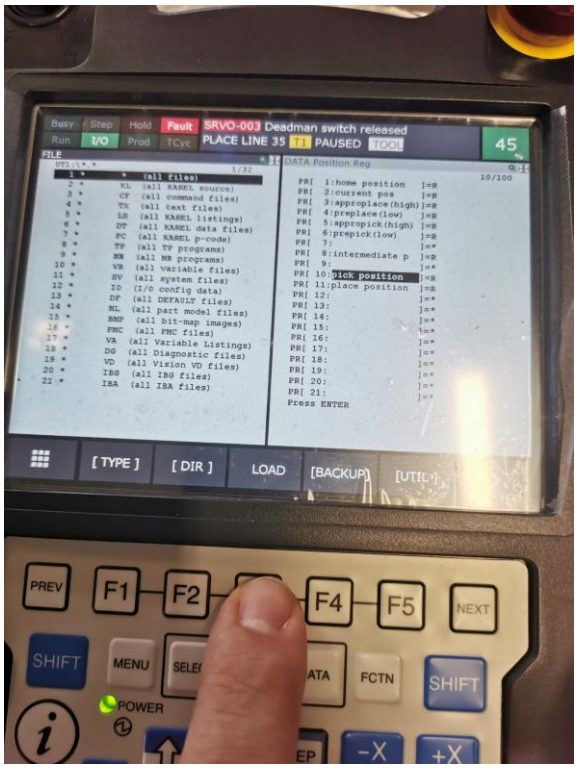


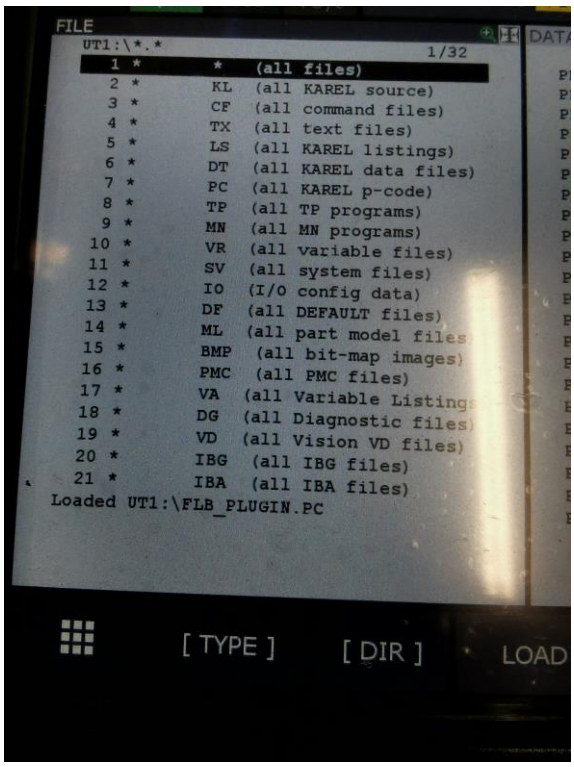
Karel
FANUC


Fault
SRVO-003 Deadman switch released
PLACE LINE 35 T1 PAUSED TOOL
45
DATA Position Reg
PR[1:home position]=R
PR[2:current pos]=R
PR[3:approplace(high)]=R
PR[4:preplace(low)]=R
PR[5:approprck(high)]=R
PR[6:prepick(low)]=R
PR[7:]=*
PR[8:intermediate p]=R
PR[9:]=*
PR[10:pick position]=R
PR[11:place position]=R
PR[12:]=*
PR[13:]=*
PR[14:]=*
PR[15:]=*
PR[16:]=*
PR[17:]=*
PR[18:]=*
PR[19:]=*
PR[20:]=*
PR[21:]=*
Press ENTER
UTIL 1
1 Set Device
2 Format
3 Format FAT32
4 Make DIR
[DIR]
LOAD
[BACKUP]
[UTIL]


Fault
SRVO-003 Deadman switch
PLACE LINE 35 T1 PAUSED
1
1 FROM Disk (FR:)
2 Backup (FRA:)
3 RAM Disk (RD:)
4 Mem Device (MD:)
5 Console (CONS:)
6 USB Disk (UD1:)
7 USB on TP (UT1:)
8
Press DIR to generate directory
[TYPE]
[DIR]

Karel
FANUC


SRVO-003 Deadman switch released
PLACE LINE 35 T1 PAUSED
FILE
DATA Position Reg
[TYPE]
[DIR]
LOAD
[BACKUP]
PREV
F1
F2
F4
F5
NEXT
SHIFT
MENU
FCTN
SHIFT
POWER
-X
+X

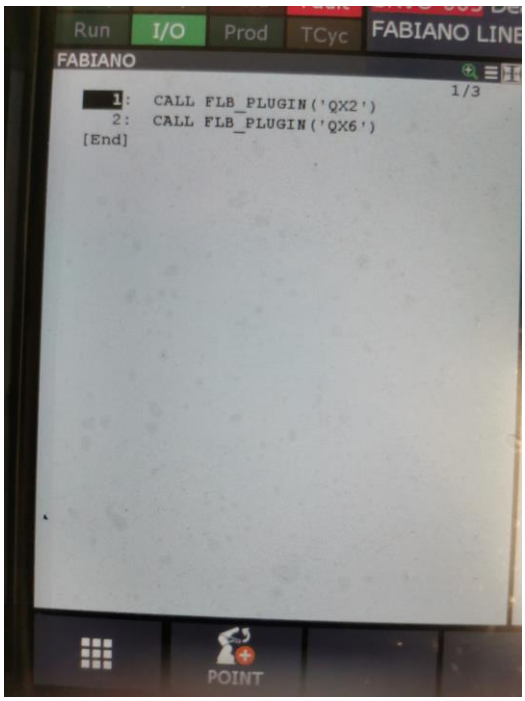

FILE
UT1:*.*
1/32
1 * * (all files)
2 * KL (all KAREL source)
3 * CF (all command files)
4 * TX (all text files)
5 * LS (all KAREL listings)
6 * DT (all KAREL data files)
7 * PC (all KAREL p-code)
8 * TP (all TP programs)
9 * MN (all MN programs)
10 * VR (all variable files)
11 * SV (all system files)
12 * IO (I/O config data)
13 * DF (all DEFAULT files)
14 * ML (all part model files)
15 * BMP (all bit-map images)
16 * PMC (all PMC files)
17 * VA (all Variable Listings)
18 * DG (all Diagnostic files)
19 * VD (all Vision VD files)
20 * IBG (all IBG files)
21 * IBA (all IBA files)
Loaded UT1:\FLB_PLUGIN.PC
[TYPE]
[DIR]
LOAD

Karel
FANUC

STEP 5:

Run the commands from the Tp program

Once the client is configured, after importing the Flb_Plugin.pc file, it will be possible to move the FlexiBowl or make diagnostics from the Tp program, the FlexiBowl response will be inserted in the String [10], editable by Karel.



Karel
FANUC

STEP 6:
Karel

Example of KAREL code:

```
PROGRAM Flb_Plugin

VAR
PingAddress:BOOLEAN
return_string : STRING[128]
status : INTEGER
moving:INTEGER
cmd_type: INTEGER
cmd_int_val:INTEGER
cmd_real_val:REAL
cmd_str_val:STRING[128]

command_Str:STRING[128]
file_var : FILE
tmp_str : STRING[128]
tmp_str1:STRING[128]
tmp_str2:STRING[128]
entry : INTEGER


ROUTINE Ping
BEGIN

        PingAddress=FALSE
        MSG_DISCO('C7:',status)
        MSG_CONNECT('C7:',status)
        IF(status=0) THEN
                PingAddress=TRUE
        else
                PingAddress=FALSE
        ENDIF
        MSG_DISCO('C7:',status)

END Ping

----////////////////////////////////////////////////////////////////////////////////////////////////////
---////////////////////////////////////////SENDCOMMAND////////////////////////////////////////////////
--////////////////////////////////////////////////////////////////////////////////////////////////////

ROUTINE SendCommand(CommandExecute:STRING):STRING


BEGIN
return_string='FAIL';
command_Str=CommandExecute
```

```
CLR_IO_STAT(file_var)
status = IO_STATUS(file_var)
IF status = 0 THEN
        --Write
        tmp_str=CHR(0)+CHR(7)+ command_Str + CHR(13)
        WRITE file_var (tmp_str)
        IF( IO_STATUS(file_var)<>0) THEN
                return_string='FAIL'
                WRITE ('Status var OUT fail',CR)
                go to end_it
        ENDIF

        --READ
        entry=0
        WHILE entry <1 DO
                BYTES_AHEAD(file_var, entry, status)
        ENDWHILE

        STATUS = IO_STATUS(file_var)
        IF(STATUS<>0)THEN
                return_string='FAIL'
                WRITE ('Status var IN fail',CR)
                go to end_it
        ENDIF
        IF (entry>0) THEN
                READ file_var (tmp_str1::entry)
                tmp_str2=SUB_STR(tmp_str1,2,entry-1)
                return_string=tmp_str2
        ELSE
                return_string='FAIL'
                go to end_it
        ENDIF
ELSE
        WRITE ('Status var fail',CR)
        return_string='FAIL'
        go to end_it
ENDIF

end_it::
RETURN(return_string)
END SendCommand
----//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
BEGIN
	GET_TPE_PRM(1,cmd_type,cmd_int_val,cmd_real_val,cmd_str_val,S
TATUS)
	IF(STR_LEN(cmd_str_val)>0) THEN
		command_Str=cmd_str_val
	else
		return_string='FAIL'
		go to end_it2

	ENDIF
	--setto i parametri di connessione
	SET_FILE_ATR(file_var, ATR_IA)
	MSG_DISCO('C7:', status)
	MSG_CONNECT('C7:',status)
	IF(status <> 0) THEN
		return_string='FAIL'
		go to end_it2
	ENDIF
	OPEN FILE file_var ('RW','C7:')
	return_string=SendCommand(command_Str)
	IF((INDEX(return_string,'%')<>0)
			AND (INDEX(command_Str,'Q')<>0)) THEN
		-----wait move
		moving=1
		WHILE (moving = 1) DO
			command_Str='RS'
			return_string=SendCommand(command_Str)
			IF(INDEX(return_string,'F')>0) THEN
				moving = 1
			else
				moving = 0
			ENDIF
			DELAY(50)
		 ENDWHILE
		 return_string='Done'
	else
		 return_string=return_string
	ENDIF
end_it2::
	CLOSE FILE file_var
	MSG_DISCO('C7:',status)

	--write the return on string 10
	SET_STR_REG(10,return_string,status)
END Flb_Plugin
```

Lista dei comandi:

Action	Description
MOVE	Moves the feeder the current parameters.
MOVE-FLIP	Moves the feeder and activates Flip simultaneously
MOVE-BLOW-FLIP	Moves the feeder and activates Flip and blow simultaneously
MOVE-BLOW	Moves the feeder and activates Flip simultaneously
SHAKE	Shakes the feeder with the current parameters
LIGHT ON	Light on
LIGHT OFF	Light off
FLIP	Flip
BLOW	Blow
QUICK_EMPTING	Quick Emptying Option
RESET_ALARM	Reset Alarm and enable the motor

Commands	Description
QX2	Move
QX3	Move-Flip
QX4	Move-Flip-Blow
QX5	Move-Blow
QX6	Shake
QX7	Light on
QX8	Light off
QX9	Blow
QX10	Flip
QX11	Quick Emptying Option
QX12	Reset Alarm