The heart of Robotics

SOFTWARE DOCUMENTATION

FORD HANDZHOU

Industrial Software Products

ABB

Overview of this manual

About this manual

This manual contains information about ford Handzhou kawasaki robot software .

Usage

This manual describe base application software which handle plc basic IO communication (job, fault, interlock, HMI ..). This manual also describe all the routines used for sealing, welding, handling , stud welding, fast welding, brushing.

Who should read this manual

This manual is intended for :

- Personnel in charge of robot programming
- Personnel in charge of installation
- Personnel in charge of commissioning
- Personnel in charge of maintenance

User qualification

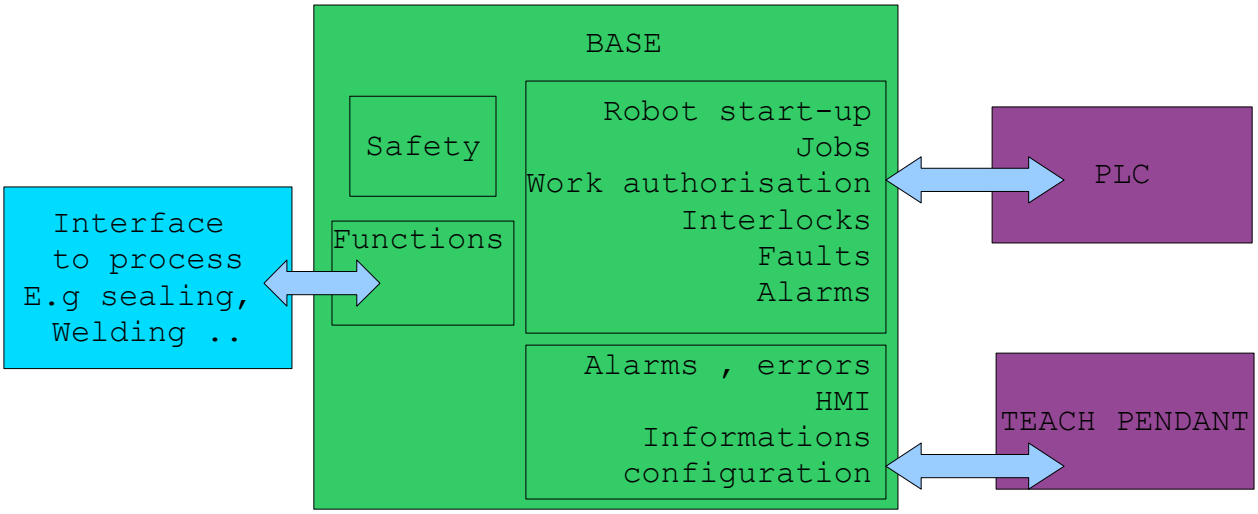
An operator only requires minimal information and instruction for operating the installation in order to be able to work with kawasaki robot.
A programmer should be familiar with kawasaki programming and be sufficiently trained for the kawasaki robot procedure.
In particular he should be familiar with the following items :

- Kawasaki robot programming language AS
- Kawasaki bloc teaching language

What is base application

Description

Base application is a software module that is present on every robots which handles plc communication, HMI and which give service to other application.



Base application handles :

- robot program safety (robot position at start-up , robot stop on hold request from plc ..)
- Jobs : in conjunction with plc base application start the requested job number
- Work authorisation : in conjunction with plc work authorisation are handled
- Interlocks : PLC manage interlocks and the case effective give a robot a privilege access .
- Error and alarm reporting : Base application reports error and alarm to PLC for a centralised viewing.
- A dedicated Human Machine Interface is displayed on the robot teach pendant to help users in case of troubleshooting. This HMI allows in some special case user to make some configuration (e.g: gripper)

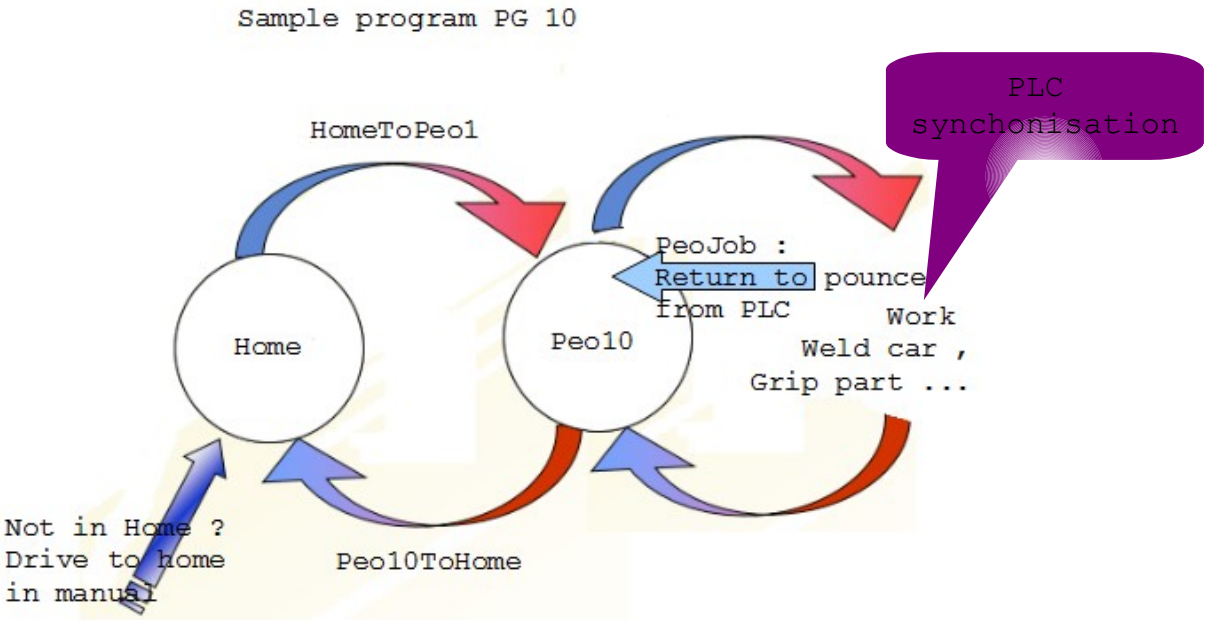
Terms and concepts

This chapter contains information relative to path organisation and controls user can achieve.
Production tasks are achieve through path chosen by a remote controller. The remote controller (eg PLC) send information (IO) through an industrial network (Profinet) the robot get this informations and select the path.
Communication length is 128 bits. Each bit has a signification and these bits are the same on every robot on the line (standard definition of IO exchange for PLC).
We define a cycle as a program number (eg a car type) , during the cycle program execution can be synchronised with PLC, such a way of programming avoid collision and allow PLC to track robot job on some specific positions.

Specific positions

HOME

Usually a cycle starts from a specific position (HOME position) and finish at the same position.
HOME position is a global and unique position , the robot internal software checks if each axis position are in a range of 2 degrees and set the case of the output robot in HOME. A new cycle is only taken in account if the robot is at HOME position. For robot safety at program start-up this position is checked and if the robot is any elsewhere the user is invited to drive the robot by hand to home .



Kawasaki user program

```
.PROGRAM pg10()#230
  HOME ; needed as standard command
  CALL hometopeo10; must provide a home to pounce program
  CALL peojob(10); must call peojob with pg number
  IF zendcyclereq GOTO endcycle ; needed by application
  ; synchronisation primitive
  CALL waitforsegment(16,"Spot Car Chb10GEO","¹Ì¶¨°¸½Ó")
  CALL spotchb10geo ; work program
  CALL endsegment(16) ; synchronisation primitive
  HOME ; needed as standard command
  endcycle: ; needed end of cycle management
.END
```

PEOX (E.G PounceX)

To avoid cycle time issue (we specified that HOME position is a global position) we need to have another specific position near the robot work. This position is to be taught out of collision and is specific to the job number (program sent by PLC indicating task to be done).
User must provide two program for safely drive to and from HOME position

- HometopeoX
- PeoXtoHome

Where X is the job number. User cannot change these names because they are automatically called by base application when PLC request to drive back to HOME (abort the task).

Nb : for code number 10 (PG10) these path are called

- Hometopeo10
- Peo10ToHome

Service position

A service position is needed for some process, it allow user to enter in the cell to do some maintenance operations (cleaning , check ..) .
Service position can be freely programmed user must only call a routine named " inservice". When this routine is called plc allow user to enter in the cell when all maintenance operations are finish the operator issue a cycle start and robot main cycle can restart.

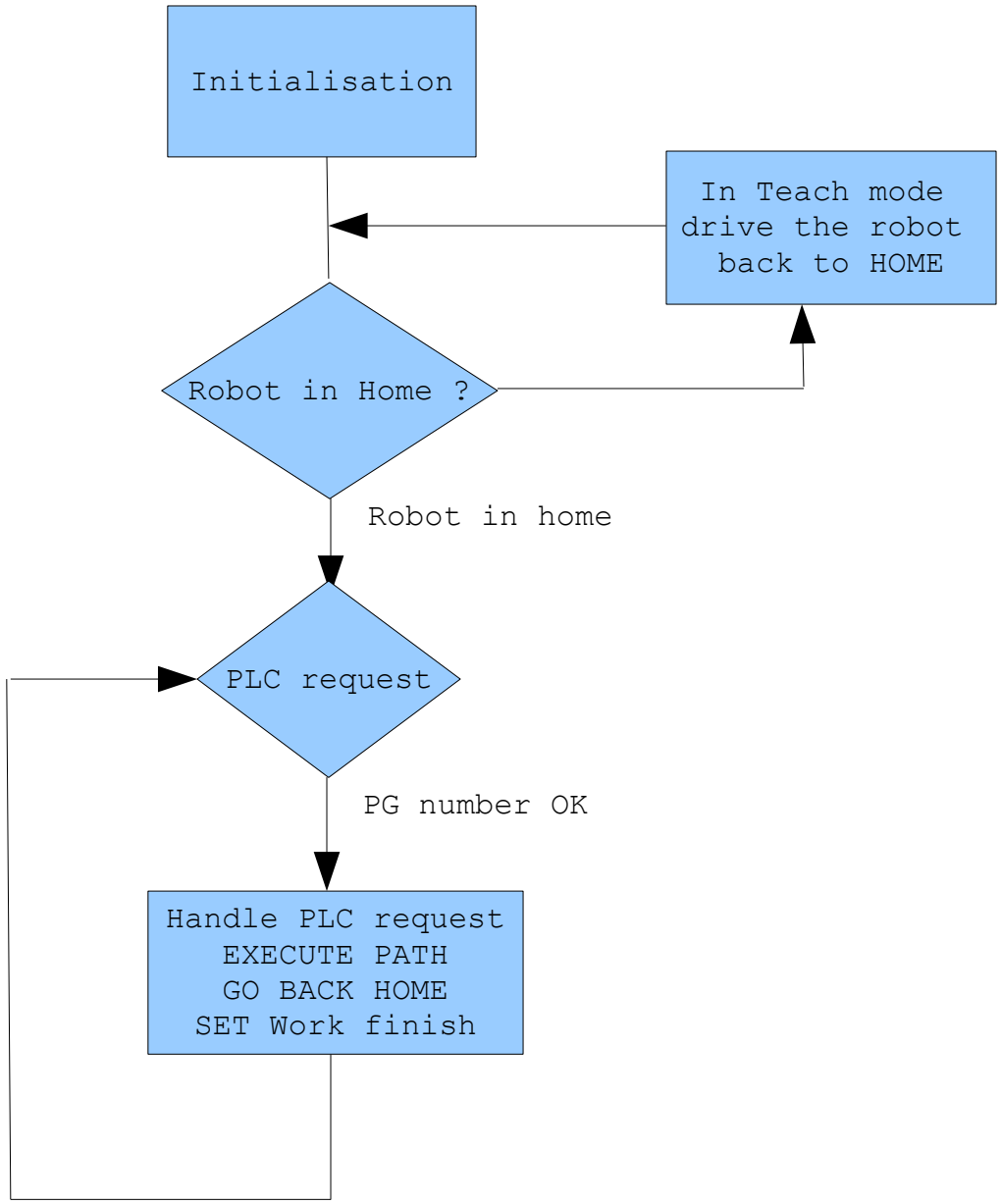
Tip change position

A service position is needed for some process, it allow user to enter in the cell to do some operations. For example in welding process , tips need to be changed so when the robot is in service position user can enter in the cell and change them.
Service position can be freely programmed user must only call a routine named "intipchg". When this routine is called plc allow user to enter in the cell when tip are changed the operator issue a cycle start and robot main cycle can restart.

Robot main program

Description

Robot main task is program “zmain” , this program is monitored by background task and selected automatically when the robot is in automatic mode (this to avoid that an unhallowed program run in automatic mode). The main task of this program is to check initialize some parameters, check if robot at start-up is at HOME position and then select the task to be done according to PLC request.



Synchronisation primitives

Description

In order to synchronise robot movement and plc task we define some functions to allow user to interact with PLC.
Through pre-defined IO robot communicate with PLC.
There is 3 kind of synchronisation objects :

- Area or segment
- Interlocks
- Service position

Area / segment synchronisation object

When performing job task the robot need to wait for some event to arise (like car in position , table rotation finished , …) to do this user need at some position to call a predefined routine and ask plc to grant him the access.
This action is performed by calling :

- WaitforSegment

User should call this function after a fine point (accuracy set to 0) and provide some parameters :

- area number : (integer from 1 to 16) given by PLC
- area description in english (string)
- area description in chinese (string)

```
; synchronisation primitive
CALL waitforsegment(16,"Spot Car Chb10GEO","¹Ì¶¨°¸½Ó")
```

When requested operation is finished and in order to stay synchronised with PLC user need to inform PLC. To do so user must call :

- EndSegment

And provide one argument :

- area number

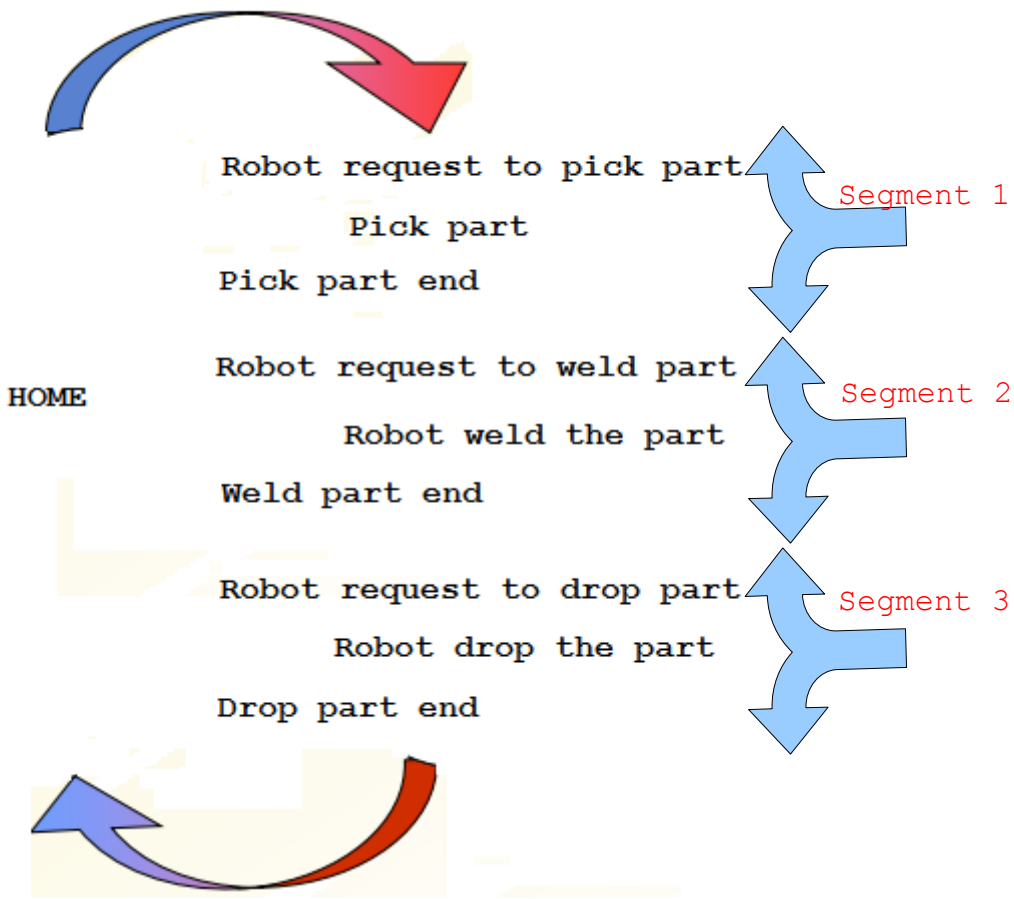
```
; synchronisation primitive
CALL endsegment(16)
```

Example :
Here in this sample the robot task is pick a part ,weld then drop it on the car. To perform such a task we will need at least 3 area :

- Area 1 – PLC grant access to pick the part
- Area 2 – PLC grant access to weld the part
- Area 2 – PLC grant access to drop the part

Then the robot program should be like

```
.PROGRAM pg10()
  HOME ; needed as standard command
  CALL hometopeo10; must provide a home to pounce program
  CALL peojob(10); must call peojob with pg number
  IF zendcyclereq GOTO endcycle ; needed by application
  ; synchronisation primitive
  CALL waitforsegment(1,"Pick part 10 ","Pick part 10 ")
  CALL pickchb10 ; work program
  CALL endsegment(1); synchronisation primitive
  CALL waitforsegment(2,"Spot Part 10","Spot Part 10")
  CALL spotchb10 ; work program
  CALL endsegment(2) ; synchronisation primitive
  CALL waitforsegment(3,"Drop part 10","Drop part 10")
  CALL dropchb10 ; work program
  CALL endsegment(3) ; synchronisation primitive
  HOME ; needed as standard command
  endcycle: ; needed end of cycle management
.END
```



NB : Areas must first be taken (waitforsegment) and then released (endsegment) , a call to endsegment in a cycle

without preceding it by waitforsegment will cause an error ("trying to free a segment never used"). Multiple areas can be used in a program up to 16. Areas can be released in the user convenience order.

Ex : releasing segment in user define order

```
.PROGRAM pg10()
  HOME ; needed as standard command
  CALL hometopeo10; must provide a home to pounce program
  CALL peojob(10); must call peojob with pg number
  IF zendcyclereq GOTO endcycle ; needed by application
  ; synchronisation primitive
  CALL waitforsegment(16,"Anticipation ","anticipation ")
  CALL waitforsegment(1,"Spot car model 10","Spot car ")
  CALL spotchb10 ; work program
  CALL endsegment(16) ; end of work open clamps
  CALL Moveback10 ; move back program
  CALL endsegment(1) ; release car
  HOME ; needed as standard command
  endcycle: ; needed end of cycle management
.END
```

In this program user request plc to open clamps on the last welding point, and then when the robot is out of car collision request plc to move the car.

Interlock synchronisation object

In a cell multiple robot cell sometimes robot can work in the same workspace. Robot programmer need to synchronise the execution of each robot path to avoid collisions. Base application defines 16 interlock zone to be freely programmed by robot programmer , therefore for each zone one interlock must be programmed in accordance with PLC.
To avoid master slave interlock programming in robot software , each interlock line is sent to plc and then it is up to plc to decide which robot he will grant the access.
To program an interlock user should use the routine called :

- Enterzone

User should call this function after a fine point (accuracy set to 1) and provide some parameters :

- zone number : integer from 1 to 16

When the robot is out of collision zone then it is time to free the zone . To do so user must call :

- ExitZone

User should call this function after a fine point (accuracy set to 0) and provide some parameters :

- zone number : integer from 1 to 16

NB : Some cells contains 6 robots and some robots can work in the same workspace than all other means that this robot may need 6 interlocks. As we said that interlocks are freely programmable in accordance with plc, we set some strings to define which zone is connected to which robot ($ildescrp[1] .. $ildescrp[16])

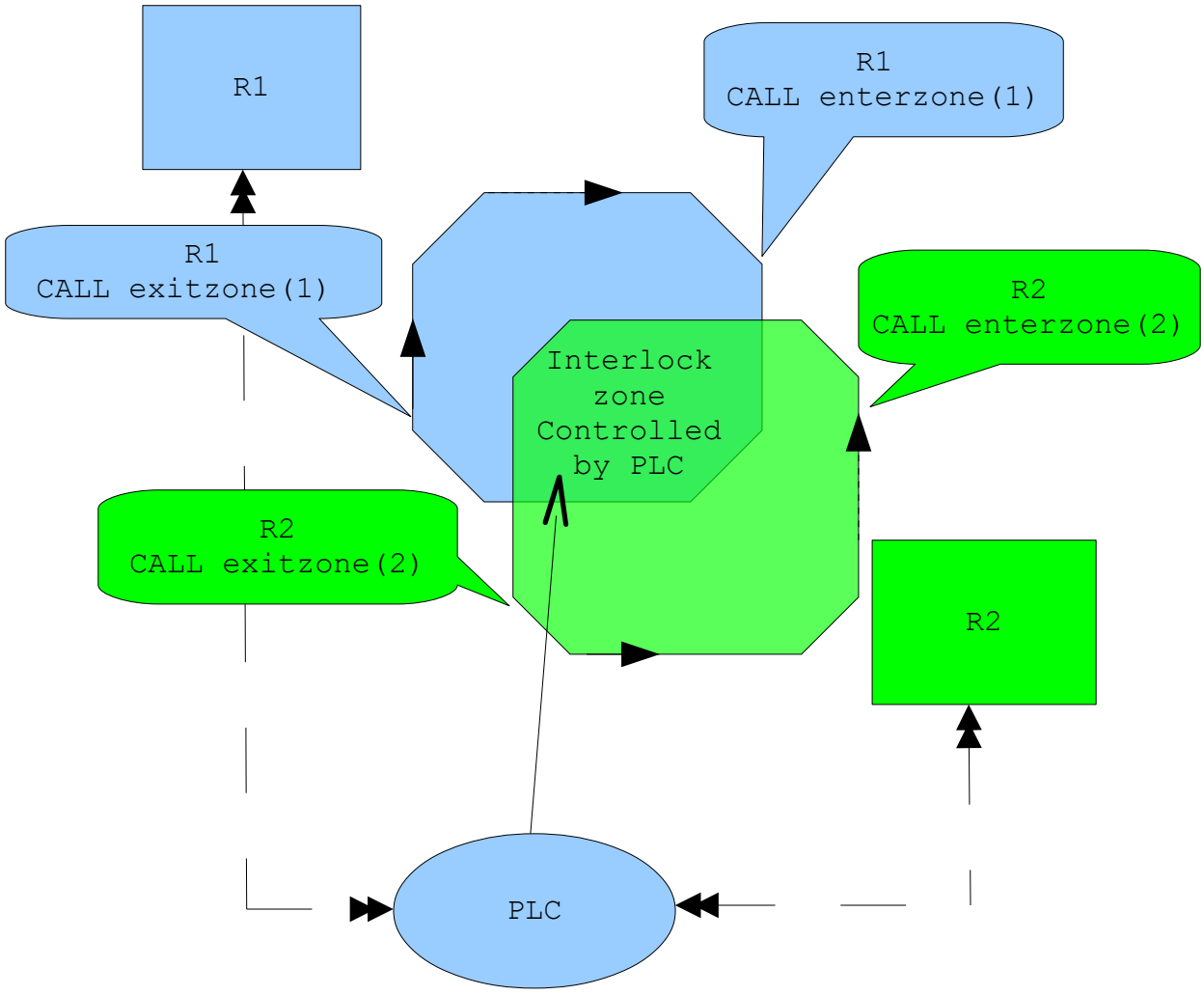
```
.PROGRAM pgdescription()#21
  ;Add a description to a pg programme
  ;These pg programme can be called by Prgmanualselect
  ;$pgname[X] ="description" X=PG CODE ACCORDING TO PLC
  ;$pgname[1]="Ford program model 1"
  ;$pgname[200]="Tip dress gun 1
  ;Interlock description english
   $pgname[10] = "Ford program model 10"
   $ildescrp[1] = "8x020-R02"
   $ildescrp[2] = "8x020-R02"
   $ildescrp[3] = "8x020-R02"
   $ildescrp[4] = "8x020-R03"
   $ildescrp[5] = "8x020-R04"
   $ildescrp[6] = "8x020-R05"
   $ildescrp[7] = "8x020-R08"
   $ildescrp[8] = "*********"
   $ildescrp[9] = "8x020-R12"
   $ildescrp[10] = "8x020-R32"
   $ildescrp[11] = "8x020-R42"
   $ildescrp[12] = "8x020-R52"
   $ildescrp[13] = "8x020-R62"
   $ildescrp[14] = "8x020-R72"
   $ildescrp[15] = "8x020-R82"
   $ildescrp[16] = "8x020-R92"
 .END
```

Block teaching program

In this program the user take interlock 5 which is with robot R1 and free it later. Please notice the ACCU0 before each call.

```
JOINT SPEED9 ACCU0 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,0,1,1) OX= WX=
CL1=0.000,10.0,10.0,0.0,0.0,0.0,0.0 #[-8.7104,-8.6297,-18.952,92.273,-
81.548,-22.625,245.02] ;

CALL enterzone(5);r1_enterzone

JOINT SPEED9 ACCU4 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,0,1,1) OX= WX=
CL1=0.000,10.0,10.0,0.0,0.0,0.0,0.0 #[-7.3042,14.211,-17.398,91.139,-
83.013,-20.949,245.02] ;
JOINT SPEED9 ACCU4 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,0,1,1) OX= WX=
CL1=0.000,10.0,10.0,0.0,0.0,0.0,0.0 #[-7.3827,12.704,-13.079,90.625,-
82.871,-16.599,259.3] ;
```

```
 …..........................
JOINT SPEED9 ACCU0 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,0,1,1) OX= WX=
CL1=0.000,10.0,10.0,0.0,0.0,0.0,0.0  #[-9.5646,18.259,-16.781,85.214,-
106.13,-16.167,222.32] ;

CALL exitzone(5);r1_exitzone

JOINT SPEED9 ACCU0 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,0,1,1) OX= WX=
CL1=0.000,10.0,10.0,0.0,0.0,0.0,0.0  #[-9.7441,17.679,-12.65,86.46,-
106.26,-11.863,29.999] ;
```



In the present scenario the first robot who request the zone will be granted access by PLC. In case of simultaneous access PLC will only give the privilege to one robot avoiding in that case any collision.

Human machine interface

Description

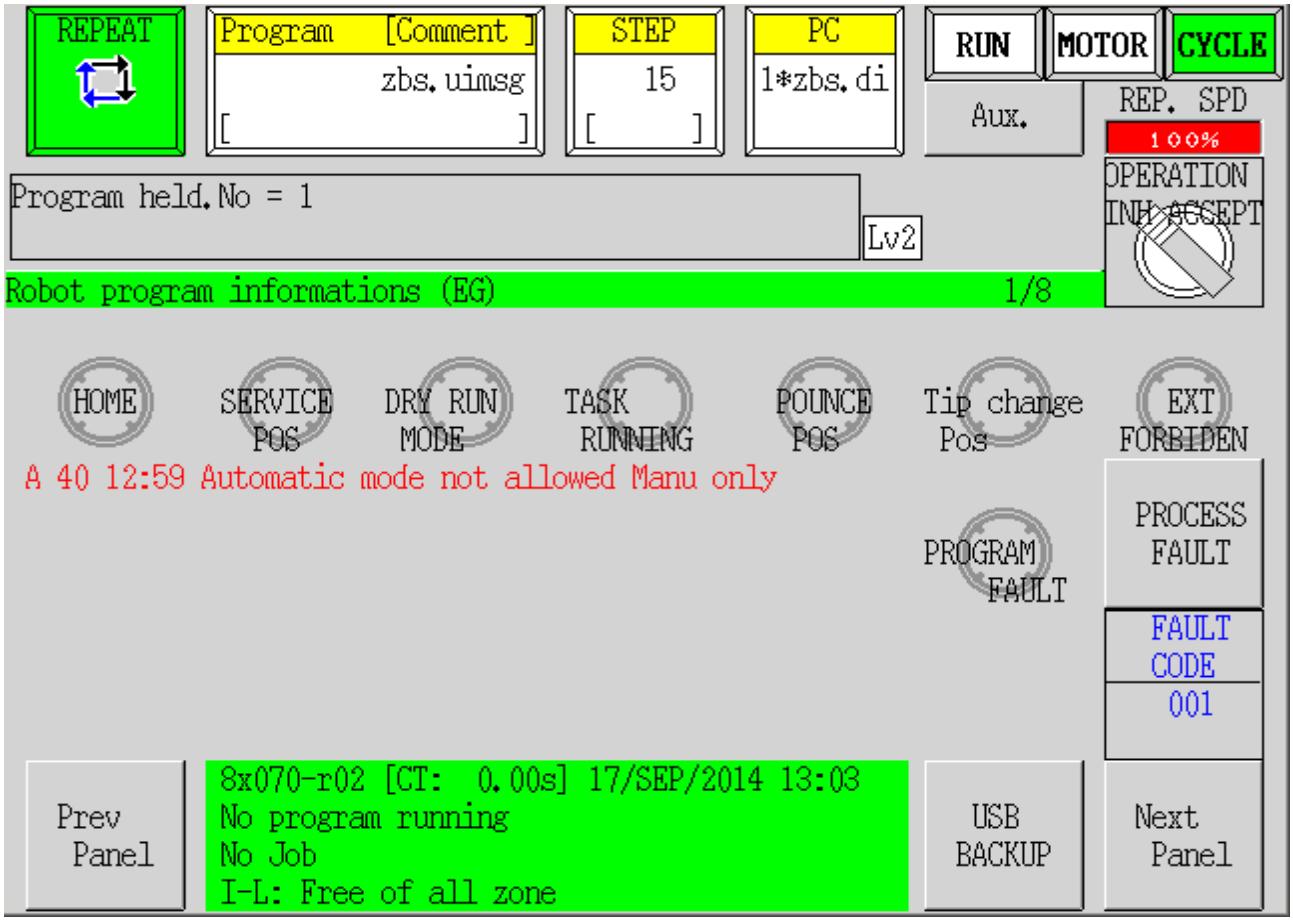
A dedicated interface is present on the pendant of the robot. This interface is composed of 6 different panels. Panel 1, 2 and 6 are for base application 3,4,5 are for process.

Panel description

Panel 1

This panel is a general panel that

- Display specific position HOME,SERVICE POS, POUNCE, Tip Change
- Display specific Input Dry run , Task running , Pg fault ,Process fault
- Display alarms
- Display robot number
- Display last Cycle time
- display program name running
- Display segment/Area used
- Display interlock used
- Make a backup on usb key
- Navigate through panels

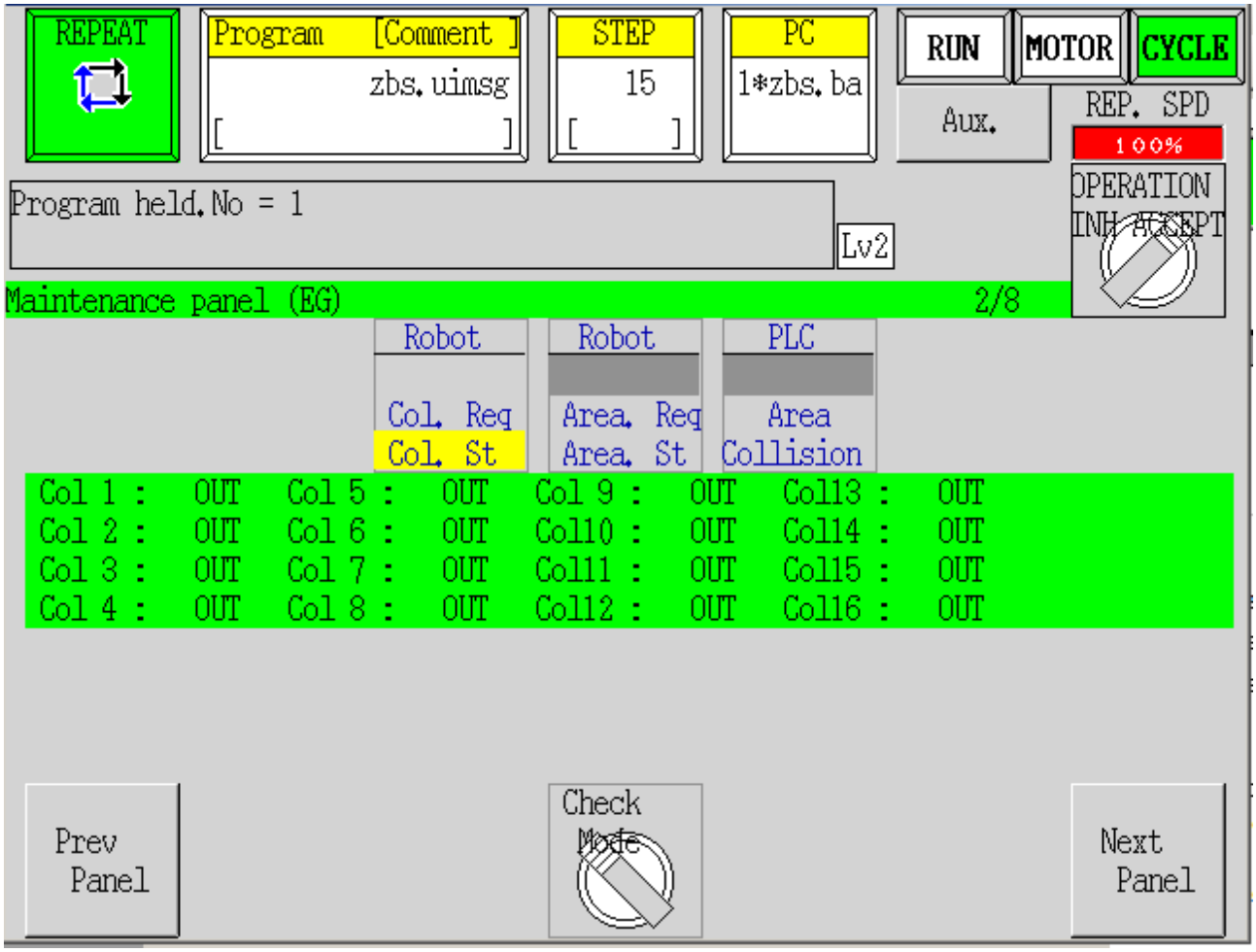


Panel 2

This panel is used to display

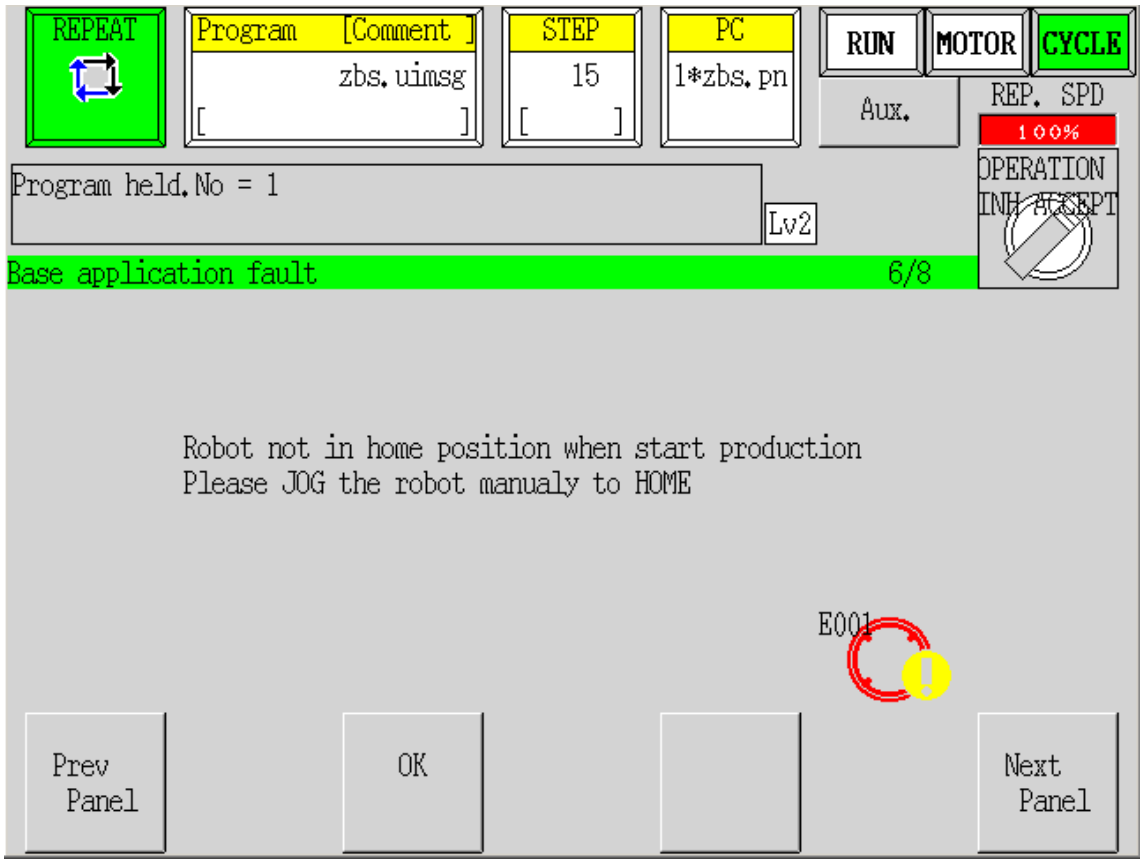
- Input and output used for segment (status and request)
- Input and output used Interlocks (status and request)

This panel also allow user to drive the robot in full automatic (normally not allowed) speed for maintenance purpose in a mode called “check mode”. This mode allow user to freely select a program for example “load determination”. When the robot in this mode PLC cannot request any program from robot.



Panel 6

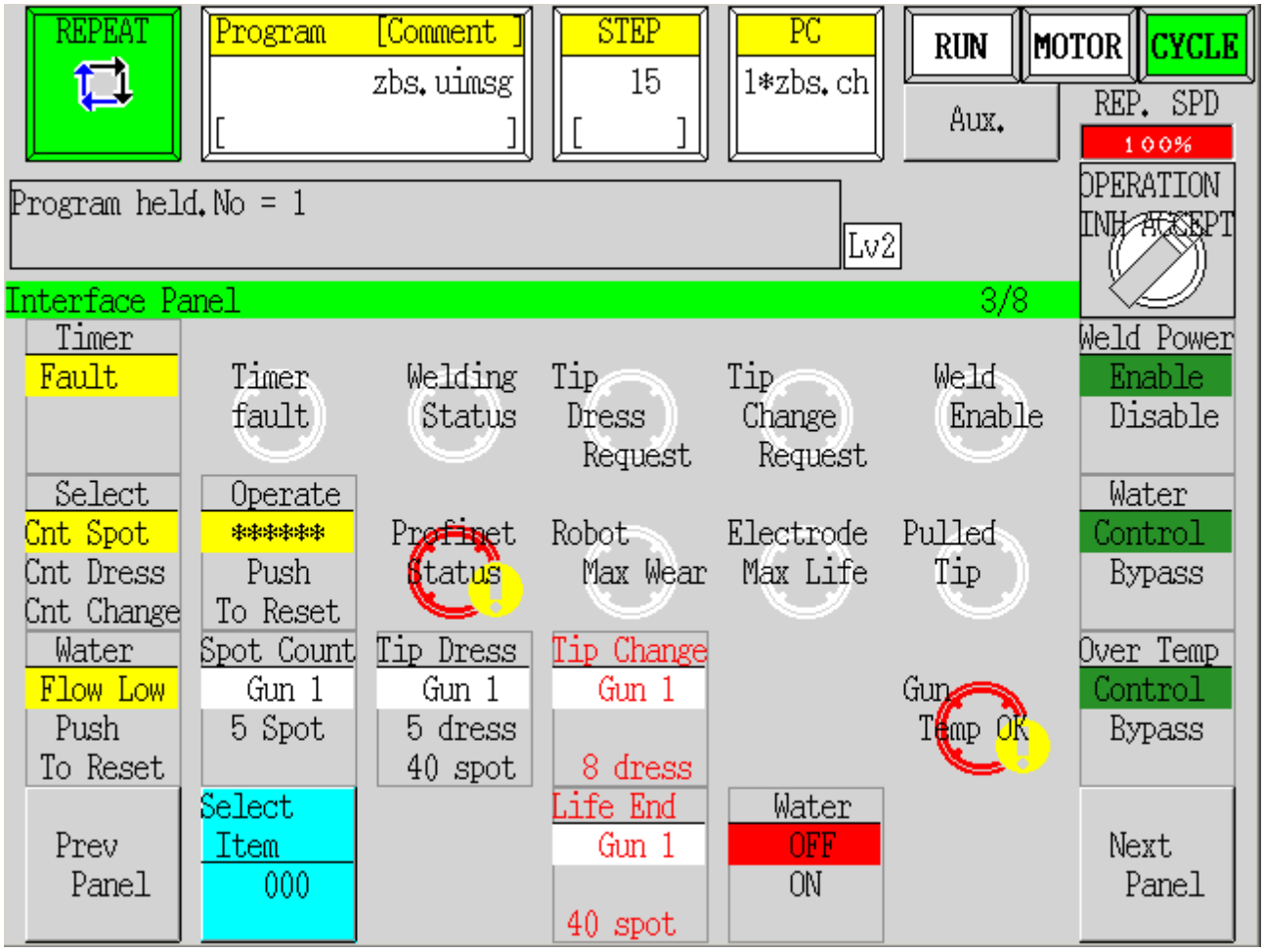
This panel is dedicated to display robot error that need validation from the operator.



This panel is automatically displayed when robot is in error , user can still navigate through all panels but need to acknowledge the error.

Panel 3,4,5

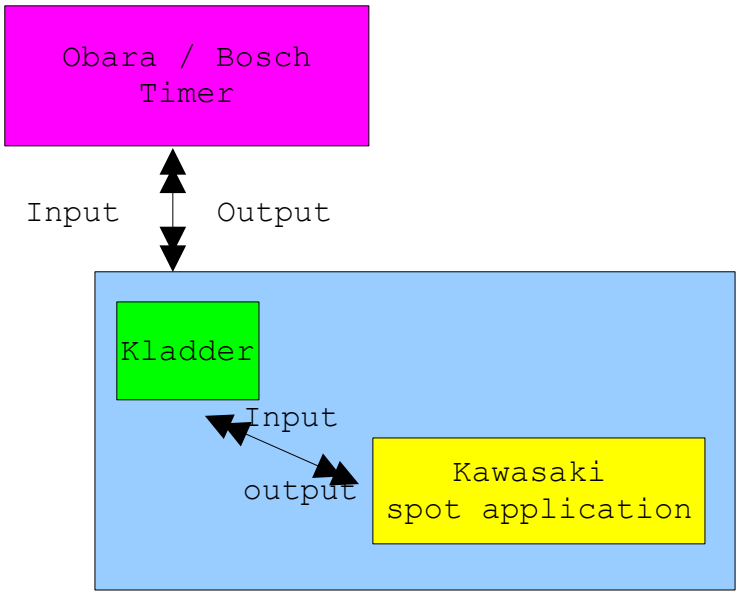
These panels are used by process application , to display specific informations. The picture above show a screen of welding application with obara timer.



Process

Spot welding application

This manual will not cover all spot application. Kawasaki provide bloc teaching instructions for teaching spot point. This manual will cover the adaptation made by abb to be able to use spot welding in conjunction with obara and bosch timer.
A Timer (bosch or obara) communicates with the robot through profinet , our application handles (in a plc background task “kladder”) all input and output coming from the timer and then set then in order to make kawasaki spot application work with these timers.



This application provide full functionality for spot welding as :

- Send timer programme number in advance
- Count the number of weld
- Handle tip dressing request
- Handle tip change request
- Handle for obara timer tip change (count number of spot , number of dress, tip end of life)
- Handle spot fault (retry abort …)
- Handle no weld in dry run
- Handle water server
- Handle pulled tip
- Handle clear identification of error for bosch timer
- Tip dress motor rotation

- Automatic Tip changer

Several programs are at user disposition and one process HMI.

Program list

Name	description
PG256	This program is automatically called by kawasaki software when the gun is closed.It contains spot welding action in case of error report
StartDresser	Start the tip dress motor
StopDresser	Stop the tip dress motor
ClearTipDress	Clear timer tip dress request
ClearTipChange	Clear timer tip change request
ManChgTip	Manual change tip. Request from user to change the tip and then allow program to continue

Description of a welding point

```
JOINT SPEED9 ACCU0 TIMER0 TOOL1 WORK0 CLAMP1 (ON,3,1,1) OX= WX=
CL1=0.000,10.0,10.0,0.0,0.0,0.0,0.0#[27.621,1.641,-14.289,
83.793 , 66.986, -164.42 , 2.7527 ] ;UB-81201-04-L
```

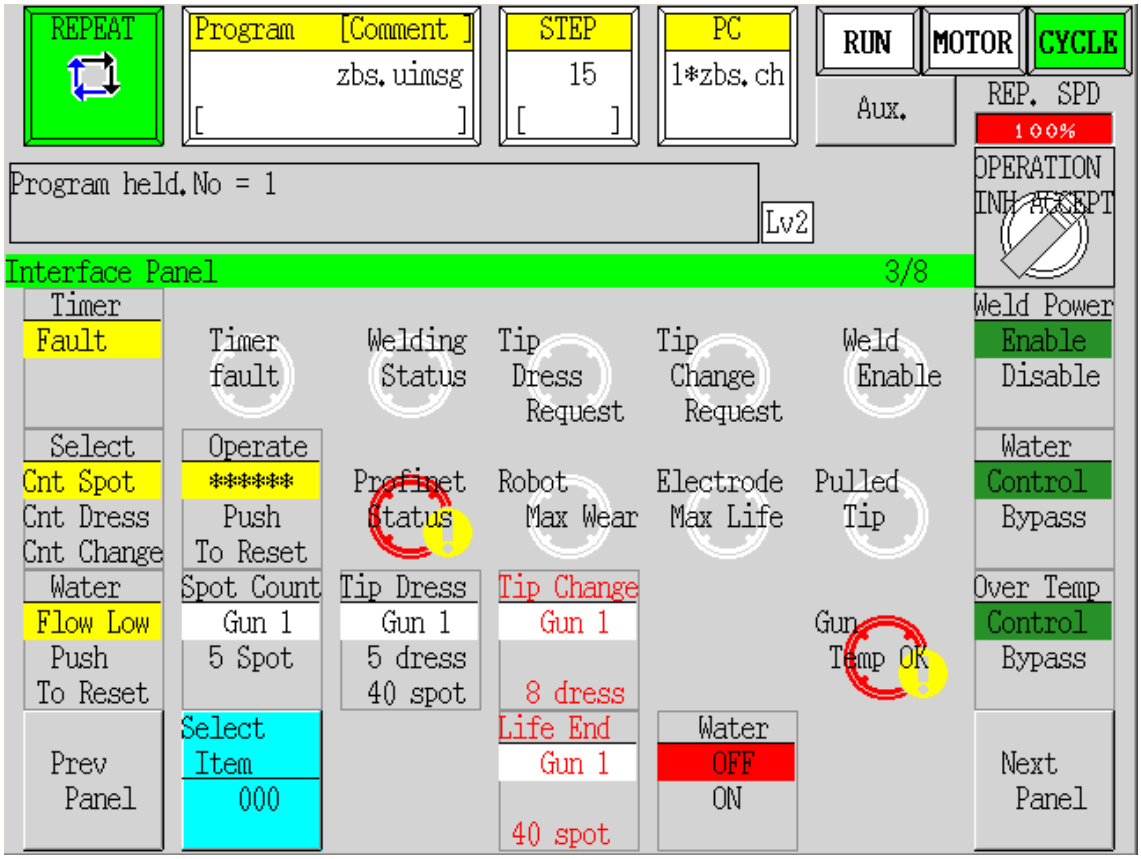
ACCU0 : Accuracy of the point should be 0
CLAMP1 (ON,3,1,1)

- Clamp 1 for gun 1
- ON means close the gun and apply force set by program
- 3 program number

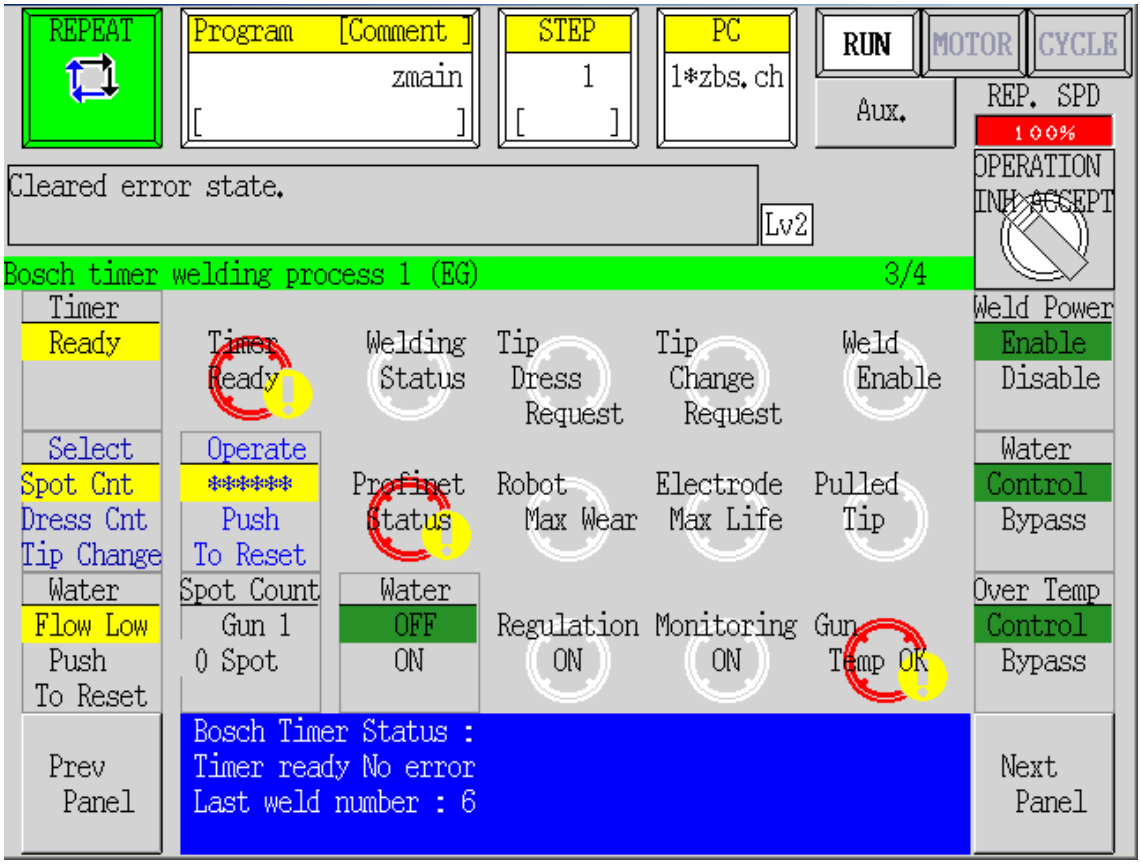
2.7527 : Thickness of the part
UB-81201-04-L : comment meaning point

NB : for more details please refer to kawasaki manual

Obara timer HMI



Bosch timer HMI



Handling application

Description

Handling application is a set of routine and two panel in robot HMI. Handling application allow user to

- configure a gripper
- simulate some inputs of a gripper
- open / close a gripper
- display the status of a gripper

In our terminology to perform an action we define two action close and open. These actions are activated by one or two outputs that can be

- clamp
- suction valve
- locating pin
- …

and the resulting state is given by one or two inputs. We usually also want to perform these actions in a certain order. Sometimes to close a gripper user must first close one set of clamp wait until these clamps are close then close some other. A gripper is defined as followed :

- 1 or 2 outputs per sequence
- Up to 6 associated states (1 or 2 inputs per state (open/closed))
- Up to 6 sequence A,B,C,D,E,F
- Up to 6 part presence

User routines

CloseSeq : close a group of sequence of a gripper and wait for a group of input.

CloseSeq(.Gripper,.Seqno,.nowait, .Timeout)

- Gripper : gripper number
- Seqno : Sequence number (pré-defined sequence are SeqA,SeqB,SeqC,SeqD,SeqE,SeqF)
- Nowait :
  - False close sequence and wait
  - True close sequence and don't wait
- Timeout : time before trigger a fault message in the case sequence is not closed

Ex :
CloseSeq (1 , SeqA+SeqB+SeqE,FALSE,2)
Close sequence A,B,E , check associated inputs. If one of the sequence is not close within 2 seconds a fault

message will be triggered and the gripper HMI will be displayed on the pendant screen.

OpenSeq : Open a group of sequence of a gripper and wait for a group of input.

OpenSeq(.Gripper,.Seqno,.nowait, .Timeout)

- Gripper : gripper number
- Seqno : Sequence number (pré-defined sequence are SeqA,SeqB,SeqC,SeqD,SeqE,SeqF)
- Nowait :
  - False close sequence and wait
  - True close sequence and don't wait
- Timeout : time before trigger a fault message in the case sequence is not open

Ex :
OpenSeq (1 , SeqA+SeqB+SeqE,FALSE,2)
Open sequence A,B,E , check associated inputs. If one of the sequence is not open within 2 seconds a fault message will be triggered and the gripper HMI will be displayed on the pendant screen.

OpenToolChanger : Open the tool changer

CloseToolChanger : Close the tool changer

ChkBeforePick : Perform all necessary check before pick a gripper. For the moment just check tool ID.

ChkAfterDrop : Perform all necessary check after drop a gripper. For the moment just check tool ID.

DropGripper : Drop the gripper on a tool stand. Deactivate bus supervision and open tool changer.

DropGripper (.Gripper)
- Gripper : gripper number

PickGripper :Pick the gripper on a tool stand. Activate bus supervision and close tool changer.

PickGripper (.Gripper,.Timeout)
- Gripper : gripper number
- Timeout : Timeout for bus detection.

CheckPpOn / CheckPpOFF : check the state of part presence

CheckPpOn (.Gripper,.ppart ,.Timeout)
CheckPpOff (.Gripper,.ppart ,.Timeout)
- Gripper : gripper number

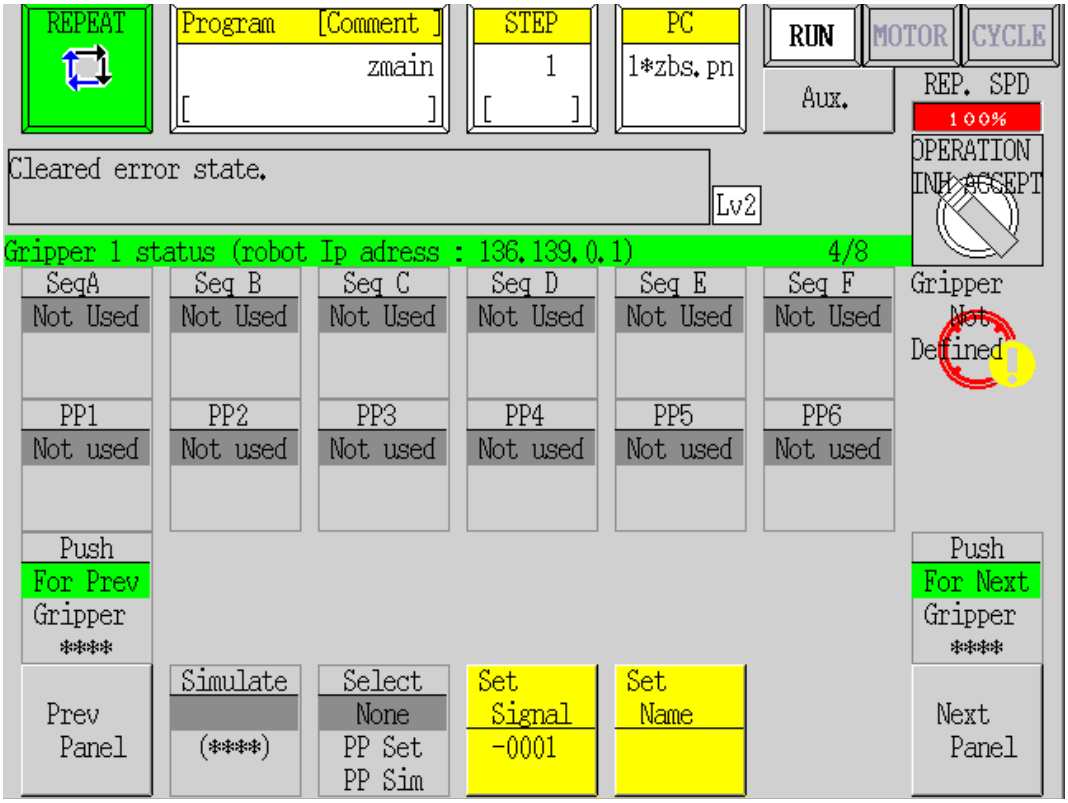
- .ppart : pre-defined PP1,PP2,PP3,PP4,PP5,PP6
- Timeout : Timeout for bus detection.

Human interface

Handling application is composed of two panels :

- Panel 1 : gripper general status
- Panel 2 : gripper associated sequence state and definition

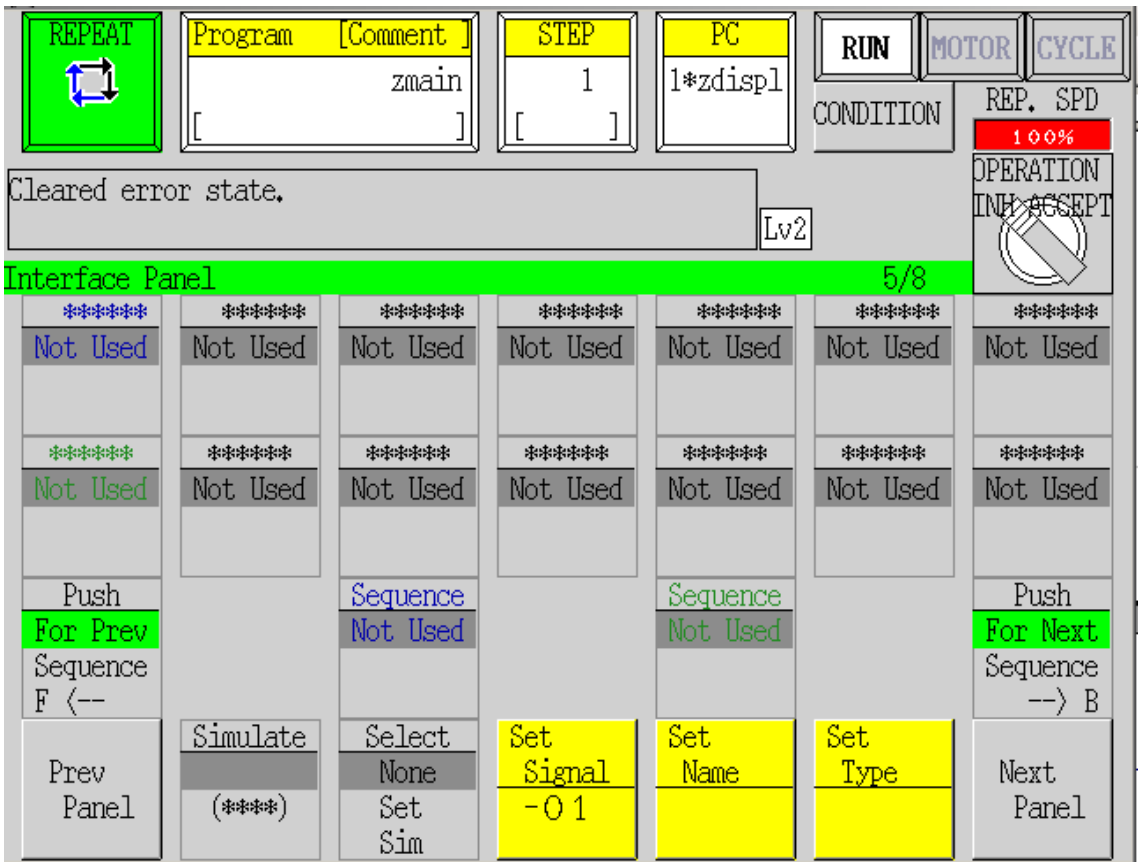
Panel 1



In this panel the operator can :

- Display the status of each sequence used by a gripper
- Set a signal number for a part presence
- Set a name for a part presence
- Simulate a part presence
- Change the gripper number

Panel 2



In this panel the operator can :

- Set a signal number
- Set a signal name
- Set a name
- Simulate an input
- Navigate through sequence
- Open/Close a sequence
- Display the state of a sequence

Sealing application

Description

Sealing application is a set of routines and one panel in robot HMI. This application cover sealing with one or two gun. Graco system is used to provide seal. Sealing start and sealing stop is achieve using clamp functions (clamp 1 seal 1, clamp 2 seal 2). Robot is configured has a sealing robot. Sealing application handle in background task the necessary operations for applying glue. Sealing application also check the amount of seal applied during the path and with the help of an handshake with graco system warm user if the volume of seal is not in the wanted tolerance.

User routines

SealingInit : Initialisation of the graco system

SealingInit (.Gun)

- Gun : sealing system number

SealingEnd : Check graco system and volume of seal , report any error during the path.

SealingEnd (.Gun)

- Gun : sealing system number

SetFlow : Set a pre-pressure before start sealing and set also the amount of seal dropped during sealing .

SetFlow (.Flow,.FlowPerc)

- Flow : amount of seal
- FlowPerc : (pre-pressure) percentage of maximum flow

Sample path

```
CALL SealingInit (1)

CALL SetFlow (12.5,50)

LINEAR SPEED3 ACCU2 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,0,0,O) 2
(OFF,0,0,0)   OX=   WX=      #[-60.623,53.431,-21.712,156.7,-
76.667,-174.33] ;

LINEAR SPEED3 ACCU2 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,0,0,O) 2
(OFF,0,0,0)   OX=   WX=      #[-60.654,55.055,-20.967,156.81,-
77.472,-174.69] ; sealing starts 100 mm after this point
```

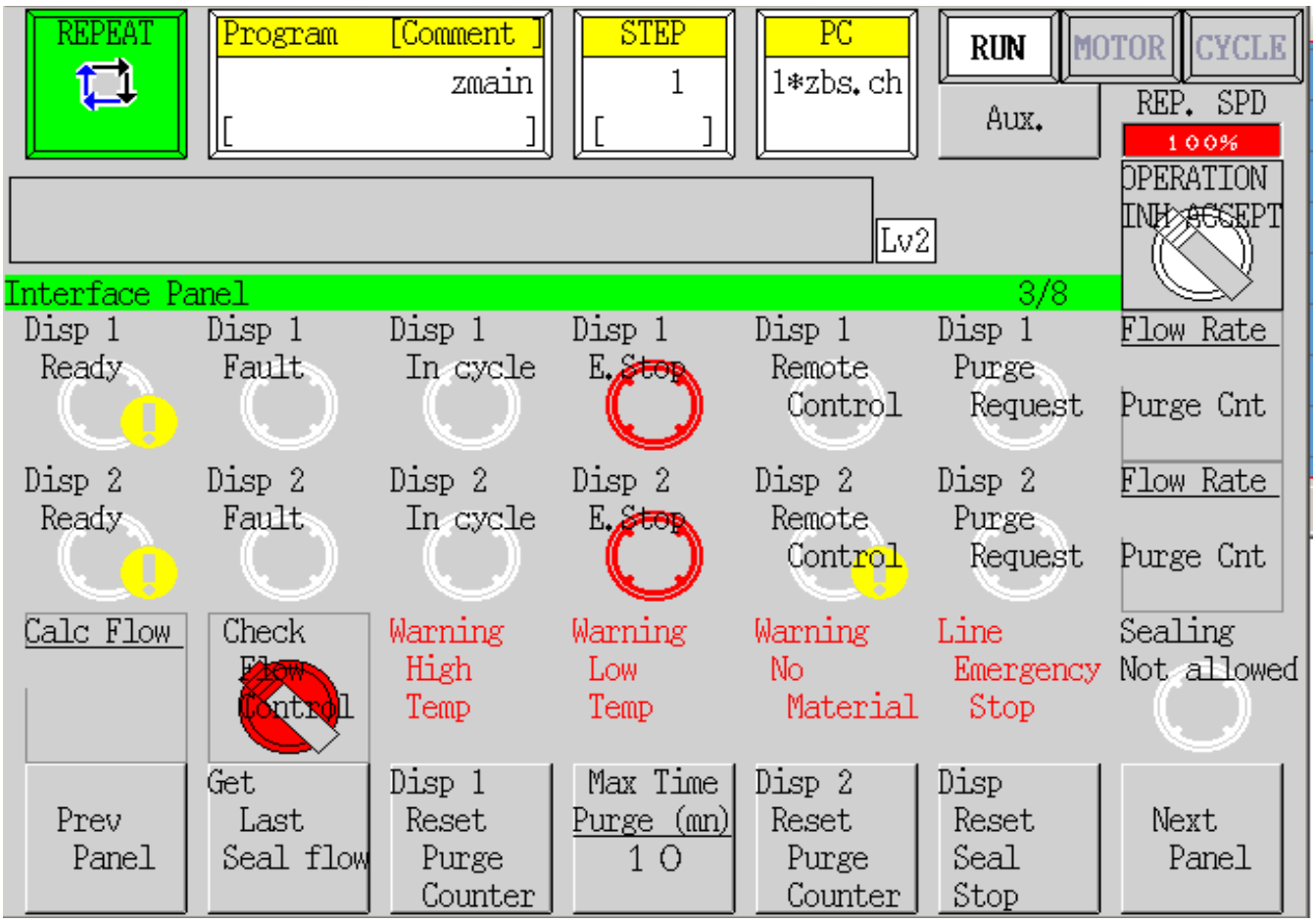
```
LINEAR SPEED3 ACCU2 TIMER0 TOOL1 WORK0 CLAMP1 (ON,100,1,0) 2
(OFF,0,0,0)   OX=   WX=      #[-60.191,54.595,-22.326,156.42,-
78.339,-174.95] ;

LINEAR SPEED3 ACCU2 TIMER0 TOOL1 WORK0 CLAMP1 (ON,0,1,0) 2
(OFF,0,0,0)   OX=   WX=      #[-56.191,54.595,-32.326,156.42,-
78.339,-174.95] ;

LINEAR SPEED3 ACCU2 TIMER0 TOOL1 WORK0 CLAMP1 (ON,0,0,O) 2
(OFF,0,0,0)   OX=   WX=      #[-60.623,53.431,-21.712,156.7,-
76.667,-174.33] ;

LINEAR SPEED3 ACCU2 TIMER0 TOOL1 WORK0 CLAMP1 (ON,0,0,O) 2
(OFF,0,0,0)   OX=   WX=      #[-60.654,55.055,-20.967,156.81,-
77.472,-174.69] ; Sealing stop 100 mm after this point

LINEAR SPEED3 ACCU2 TIMER0 TOOL1 WORK0 CLAMP1 (OFF,100,0,O)
2  (OFF,0,0,0)  OX=  WX=    #[-60.623,53.431,-21.712,156.7,-
76.667,-174.33] ;

CALL SealingStop(1)
```

NB : In case of a robot stop or fault issued by the graco system the robot will not sealing any more until the program reach the instruction sealing stop, an error will be reported , step number at start step number at stop amount of seal provided.

Human interface

Sealing application is composed of one panel, the operator can

- Display general information about graco system
- Read the amount of seal dropped
- Reset the purge time

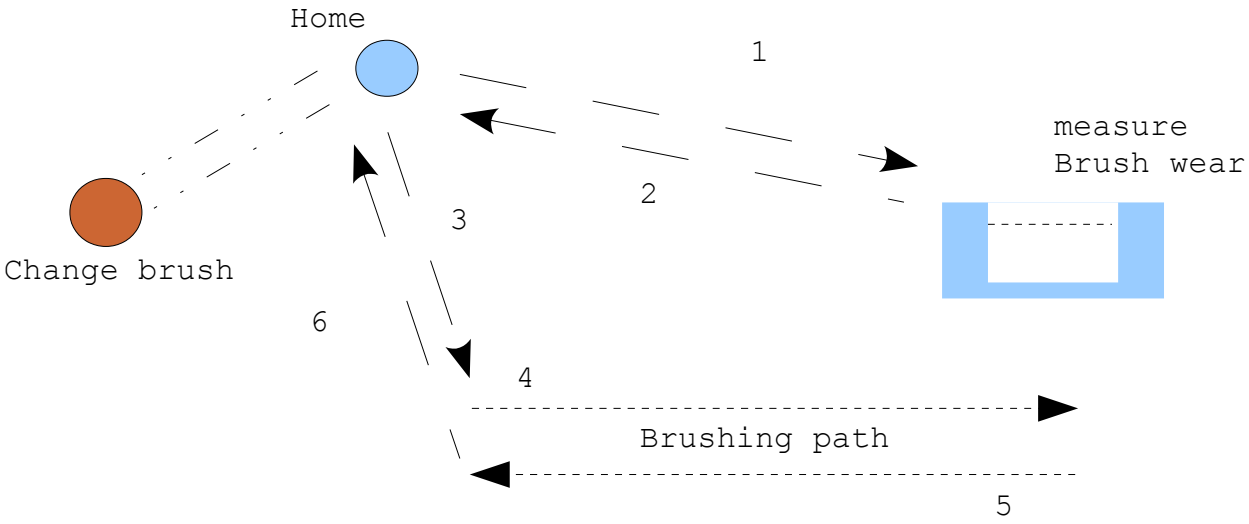

REPEAT
Program [Comment]
zmain
[ ]
STEP
1
[ ]
PC
1*zbs.ch
RUN
MOTOR
CYCLE
Aux.
REP. SPD
100%
OPERATION
INH ACCEPT
Lv2
Interface Panel
3/8
Disp 1 Ready
Disp 1 Fault
Disp 1 In cycle
Disp 1 E.Stop
Disp 1 Remote Control
Disp 1 Purge Request
Flow Rate
Purge Cnt
Disp 2 Ready
Disp 2 Fault
Disp 2 In cycle
Disp 2 E.Stop
Disp 2 Remote Control
Disp 2 Purge Request
Flow Rate
Purge Cnt
Calc Flow
Check Flow Control
Warning High Temp
Warning Low Temp
Warning No Material
Line Emergency Stop
Sealing Not allowed
Prev Panel
Get Last Seal flow
Disp 1 Reset Purge Counter
Max Time Purge (mn)
10
Disp 2 Reset Purge Counter
Disp Reset Seal Stop
Next Panel

Brushing application

Description

Brushing is the operation that consist in cleaning the car after laser welding. A rotating brush is carried by the robot . The brush rotate forward and backward a sensor detect the rotation of the brush if during the path the rotation stops then the robot move away from the car print a message and then wait for an acknowledge. When the the brush rotate again then robot path can be continued.
Brush is made of an abrasive metal when entering in contact with the welded area the diameter of the brush is reduce. To allow the brush to stay in contact with the car , tool tip must be recalculated. A sensor is used to perform this action.

Brushing path



Steps :

1. Robot move to measure position , check the distance between first measure and actual measure (brush wear) then recalculate new tool tip . If the calculation is outside tolerance then a error is displayed on teach pendant.
2. Robot drive back to Home position. If brush wear is greater than maximum wear allowed then the robot will move to change brush position and user will be prompted to change the brush.
3. Robot move to brushing path start position.
4. Head pressure is set.Brush motor start in forward or backward direction. Monitoring of motor rotation is started.
5. Motor stop, motor start forward/backward and monitoring is started.
6. Motor stop and robot drive back to HOME.

Brushing routines

BrushMotStart (On,1) : start motor forward and check after 1 second if motor turning

BrushMotStart (OFF,1): start motor backward and check 1 second if motor is turning

SetPressure (500,1,OFF) :

- OFF wait pressure achieve
- ON don't wait

BrushMotStop (1) : stop motor and check state

BrushcalcTcp (speed , init) :

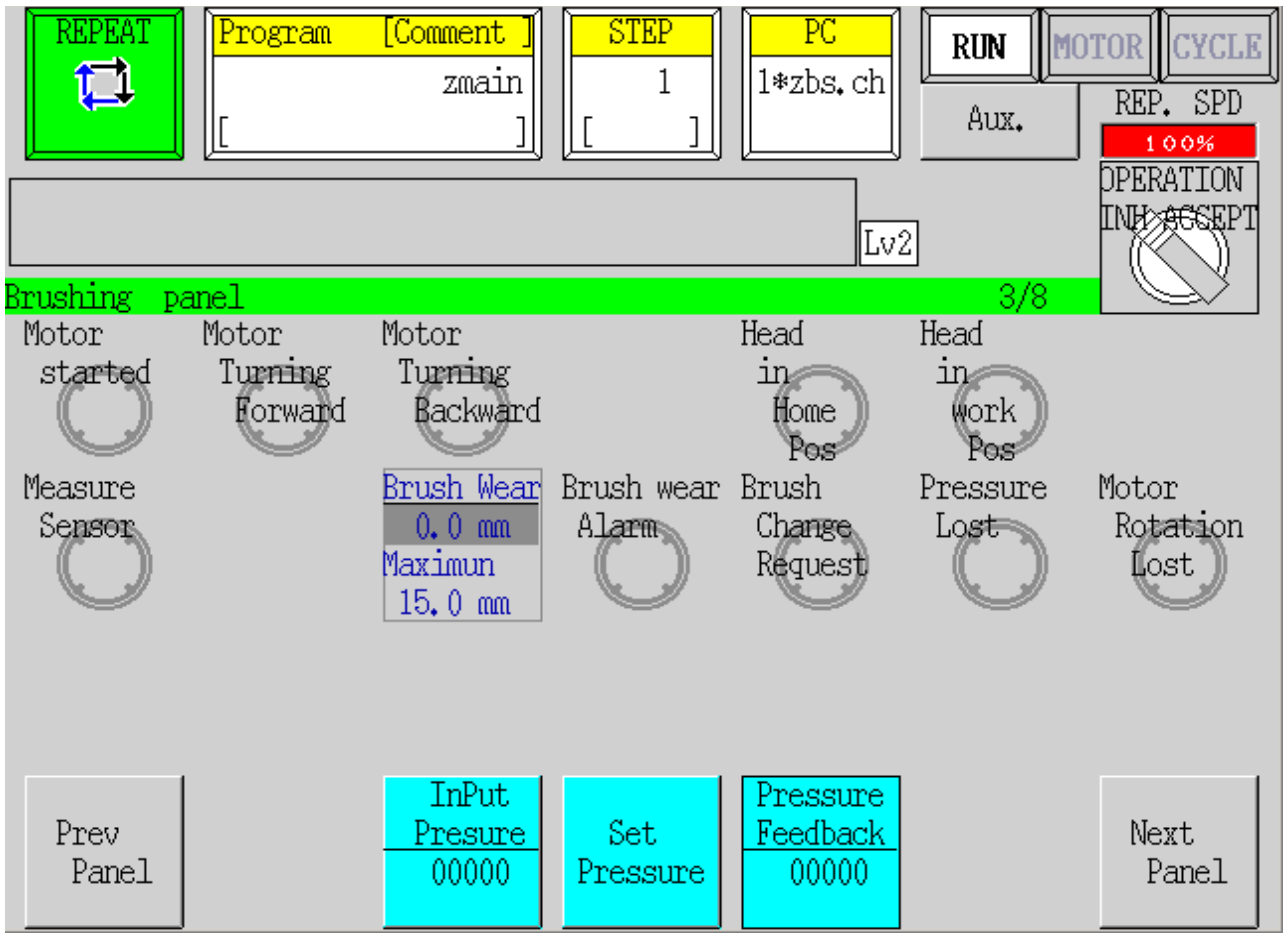
- speed: speed to reach the search point
- init :
  - TRUE indicate that the brush is new
  - FALSE indicate that software must recalculate brush wear

Human interface

brushing application is composed of one panel, the operator can :

- Display process information
- Monitor brush wear
- Set a pressure for the brush head

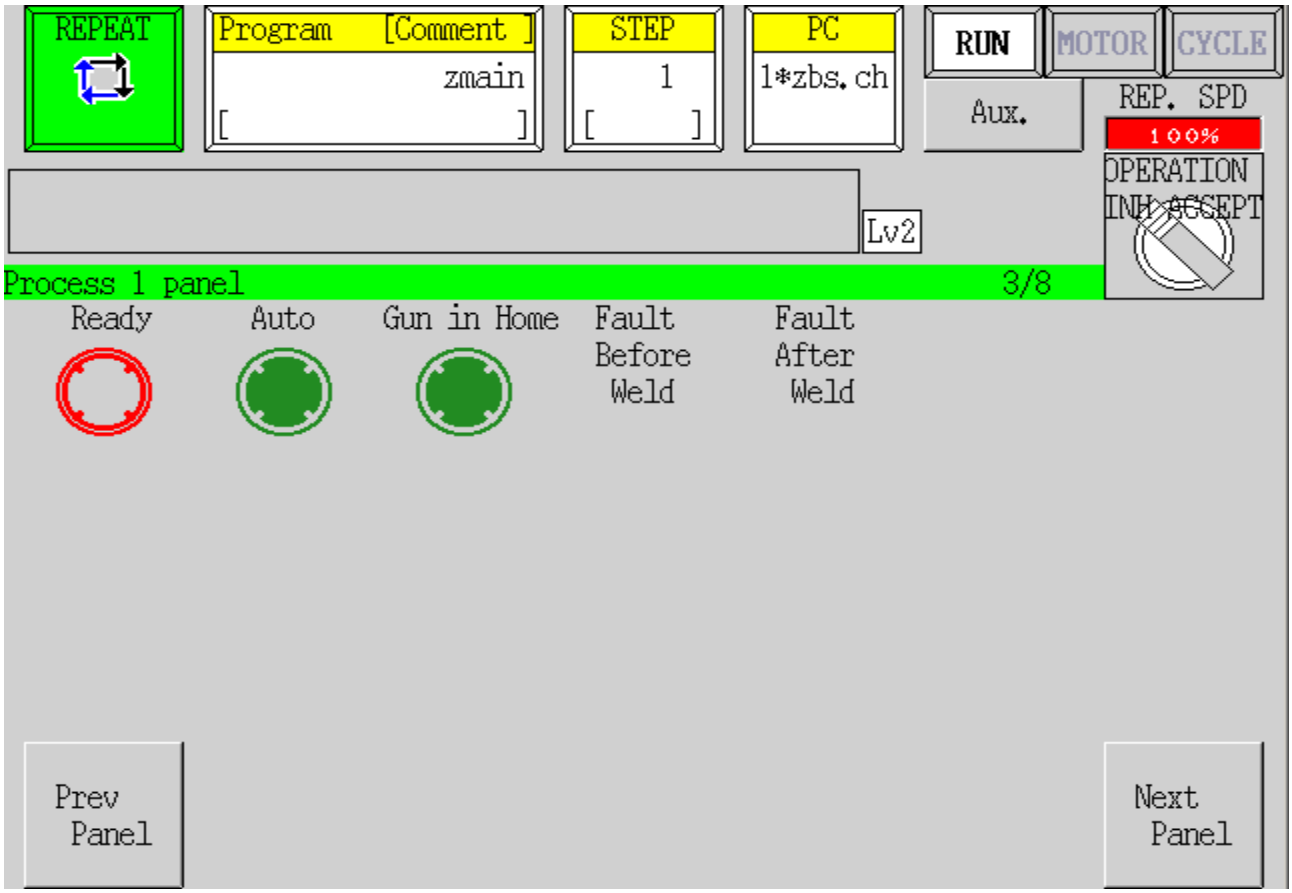
Brushing panel



Stud weld application

Description

Stud weld application expose only one routine. The purpose of this routine is to send a programme number to the welding machine and then wait until job is finished. Stud weld application monitor signals from the welding machine and reports any error message.
Human interface is very simple and display only relevant information from the welding machine.



Welding routine

StudWeld(.pgnumber) : allow user to weld a stud , program number is sent to the machine.

ARC welding

Description

Arc welding application intensively uses kawasaki arc welding software, please report to kawasaki documentation for more information about how to use instruction set.
Welding timer used for this application is a fronius generator , fronius uses jobs to describe a welding program (weld current, wire speed …) , our application sends the correct job number and remap all necessary information needed by kawasaki software (wire feed , wire stick..).

Routine exposed to user

CheckClean : check if torch cleanning has to be done
CleanTorch (.time) : clean torch for a variable duration
advancewire (.time) : feed the wire for a variable duration

Arc welding path

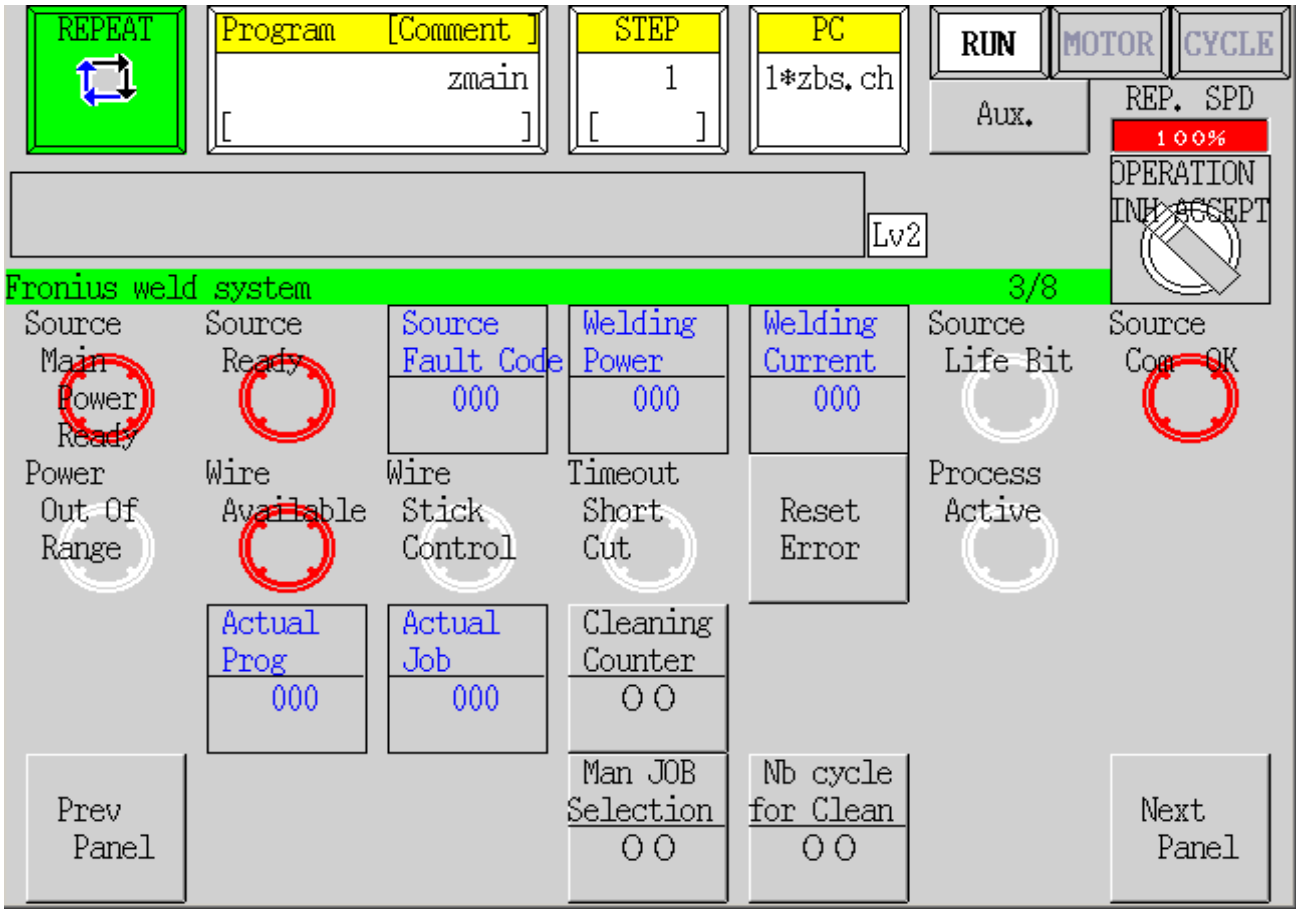
```
AC JOINT SPEED9 ACCU4 TIMER0 OX= WX= #[-83.206,-27.462,-104.6,4.8059,-89.519,13.792,1099.9] ;

AC JOINT SPEED9 ACCU4 TIMER0 OX= WX= #[-27.425,-27.462,-104.61,-8.0802,-68.18,13.791,1099.9] ;

WS JOINT SPEED9 TIMER0 OX= WX= #[-22.121,-11.615,-105.73,47.047,-48.129,-66.552,1099.9] ;

WS JOINT SPEED9 TIMER0 OX= WX= #[-21.551,-14.412,-117.17,60.637,-41.698,-81.254,1099.9] ;

WE LINEAR WELD_COND4 OX= WX= #[-19.88,-15.011,-118.35,61.672,-42.849,-83.162,1099.9] ;

AC JOINT SPEED9 ACCU4 TIMER0 OX= WX= #[-14.251,6.4301,-87.49,61.408,-64.883,-33.34,1099.9] ;

AC JOINT SPEED9 ACCU4 TIMER0 OX= WX= #[-11.241,16.698,-91.585,83.769,-70.616,-43.722,1099.9] ;

WS JOINT SPEED9 TIMER0 OX= WX= #[-11.88,20.393,-101.2,96.854,-73.816,-59.518,1099.9] ;

WC LINEAR WELD_COND2 OX= WX= #[-31.26,27.711,-60.074,36.52,-73.443,-0.59632,1034.9] ;

WE LINEAR WELD_COND4 OX= WX= #[-11.665,21.761,-99.479,96.633,-73.955,-59.126,1099.9] ;
```

AC : air cut , move without welding
WS : weld start point
WE : weld end point (weld condition)
WC : weld continue (weld condition)

HMI Panel

This panel allow user to :

- Monitor process informations
- Set cleaning counters
- Reset fronius generator



Body Side pickup

Description

Body sides are transferred to the station with an approximate position , this leads to an incapacity of the robot robot to pick a body side. A vision system from perceptron has been added to solve the problem. This system communicates with the robot by the help of dedicated Ethernet frames (refer to perceptron manuals) and some IO to the line PLC. Perceptron software has been packaged in base application as a module , and some specific routines added to simplify robot programmer job.
From the robot side the application is quite simple , programmer need to provide :

1. A good tool tip center (camera tcp). To achieve a good precision camera tcp is taught with the help of the vision system. (TOOL 9 for block teaching , tool camera for as langage)
2. Provide at least 3 robot position of 3 know areas on the part according to perceptron request. When measuring the part a number identifying the position has to be sent to perceptron system.
3. Teach a robot frame.
4. Provide a pickup path in the shifted frame (original frame + perceptron shift frame "userframe »)

Then after performing the 3 point measurement , the perceptron system is able to compute a new frame offset and provide it to the robot.

Routines exported to user :

MeasurePoint (.pointnum,.framenum)

- pointnum : point number indentifying the position
- framenum : user frame to be shifted

Getfrmeasured (.framenum,.result)

- framenum : user frame to be shifted
- result : result of operation

Be aware to never change the original frame (forig[X]) otherwise all positions will be lost.

Robot program

```
.PROGRAM pg10()#5640
;****************************************
; CD539_6X030_R01
;****************************************
  HOME
  CALL hometopeo10;
  CALL peojob(10);
  IF zendcyclereq GOTO endcycle
  CALL waitforsegment(1,"Measure body-side model
10","²âÁ¿10°Å³µÐÍ")
  CALL measchb010
  CALL endsegment(1)
  CALL waitforsegment(2,"GetFrame from model
10","»ñÈ¡10°Å³µÐÍµÄ×ø±êÏµ")
  CALL getframechb010
  CALL endsegment(2)
  CALL waitforsegment(3,"PickPart model
10","×¥10°Å³µÐÍÁã¼þ")
  CALL pickpartchb01
  CALL endsegment(4)
  CALL waitforsegment(5,"Drop Part moodel
10","·Å10°Å³µÐÍÁã¼þ")
  CALL droppartchb01
  CALL waitforsegment(6,"Leave Car moodel
10","Àë¿ª10°Å³µÉí")
  CALL leavecarchb010
  CALL endsegment(6)
  CALL waitforsegment(7,"Move Back To Home","move back to
home")
  HOME
  CALL endsegment(7)
endcycle:
.END
```

NC LOCATOR

Description

NC is used to hold the car during robot job operations. NC consist in a multi-axis robot controller. Each robot is composed of 3 axis , a NC can drive up to 8 robot (3 axis Cartesian robot). Kawasaki controller has one motion planner for every robot connected , it can drive each robot independently on in a synchronised way. We use this ability (synchronised motion planner) to drive a set of robot. NC locator program is the same has robot program , for PLC a NC or a welding robot is almost the same. The IO exchange between PLC and NC is the same has for robot. Production program will also be the same , we will use the same synchronisation primitives to synchronise NC program and plc program.

NC locator production program

```
.PROGRAM r1_pg10()#2

CALL waitsegment.1(1) ; plc synchronisation primitive
; Go to CAR Position
FN500[ 1 ]  ; robot motion sync start

JOINT SPEED9 ACCU1 TIMER0 TOOL1 WORK0 GROUP1  CLAMP1 (OFF,0,0,O)
2  (OFF,0,0,O)  OX=  WX=  #[99.67,9.337,3,205.43,150.46,3,-2000,-
2000] ;

JOINT SPEED9 ACCU1 TIMER0 TOOL1 WORK0 GROUP1  CLAMP1 (OFF,0,0,O)
2 (OFF,0,0,O) OX= WX= #[99.67,9.34,320.53,205.43,150.46,158.66,-
2000,-2000] ;

FN501[ 1 ]  ; robot motion sync stop

CALL endsegment.1(1)

CALL waitsegment.1(2)

CALL r1_closeclamps

CALL endsegment.1(2)

CALL waitsegment.1(3)

CALL r1_openclamps

CALL endsegment.1(3)

;Back to HOME

CALL waitsegment.1(4)
```

```
CALL r1_checkpos(10)

FN500[ 1 ]

JOINT SPEED9 ACCU4 TIMER0 TOOL1 WORK0 GROUP1  CLAMP1 (OFF,0,0,O)
2 (OFF,0,0,O) OX= WX= #[148.8,189.2,3,200,159.3,3,-2000,-2000] ;

JOINT SPEED9 ACCU1 TIMER0 TOOL1 WORK0 GROUP1  CLAMP1 (OFF,0,0,O)
2 (OFF,0,0,O) OX= WX= #[148.8,0,3,200,0,3,-2000,-2000] ;
FN501[ 1 ]

CALL endsegment.1(4)

.END
```

Waitsegment.X(Y) : motion planner X (1 to 4) , number of segment Y (1 to 16)

- Motion planner X wait a segment number Y

Endsegment.X : motion planner X (1 to 4) , number of segment Y (1 to 16)

- Motion planner X wait send end segment number Y

rX_openclamps : motion planner X (1 to 4)

- Motion planner X open the clamps and wait clamps to be opened

rX_closeclamps : X (1 to 4)

- Motion planner X close the clamps and wait clamps to be closed

rX_checkpos(Y) : motion planner X (1 to 4) , number of positions Y (1 to 16)

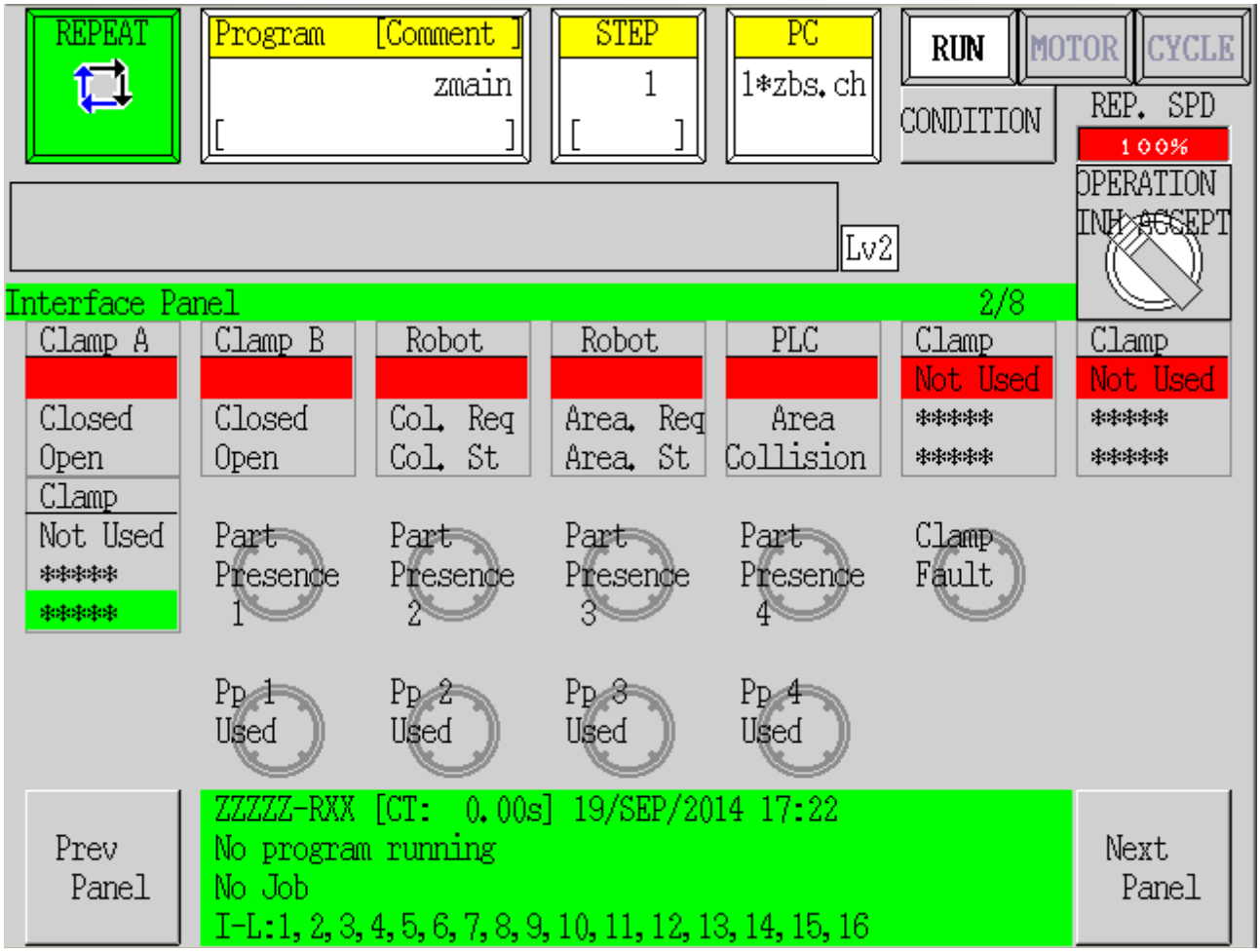
- Motion planner X check if associated robot are in position Y. 16 positions (16 cars type) can be achieved.

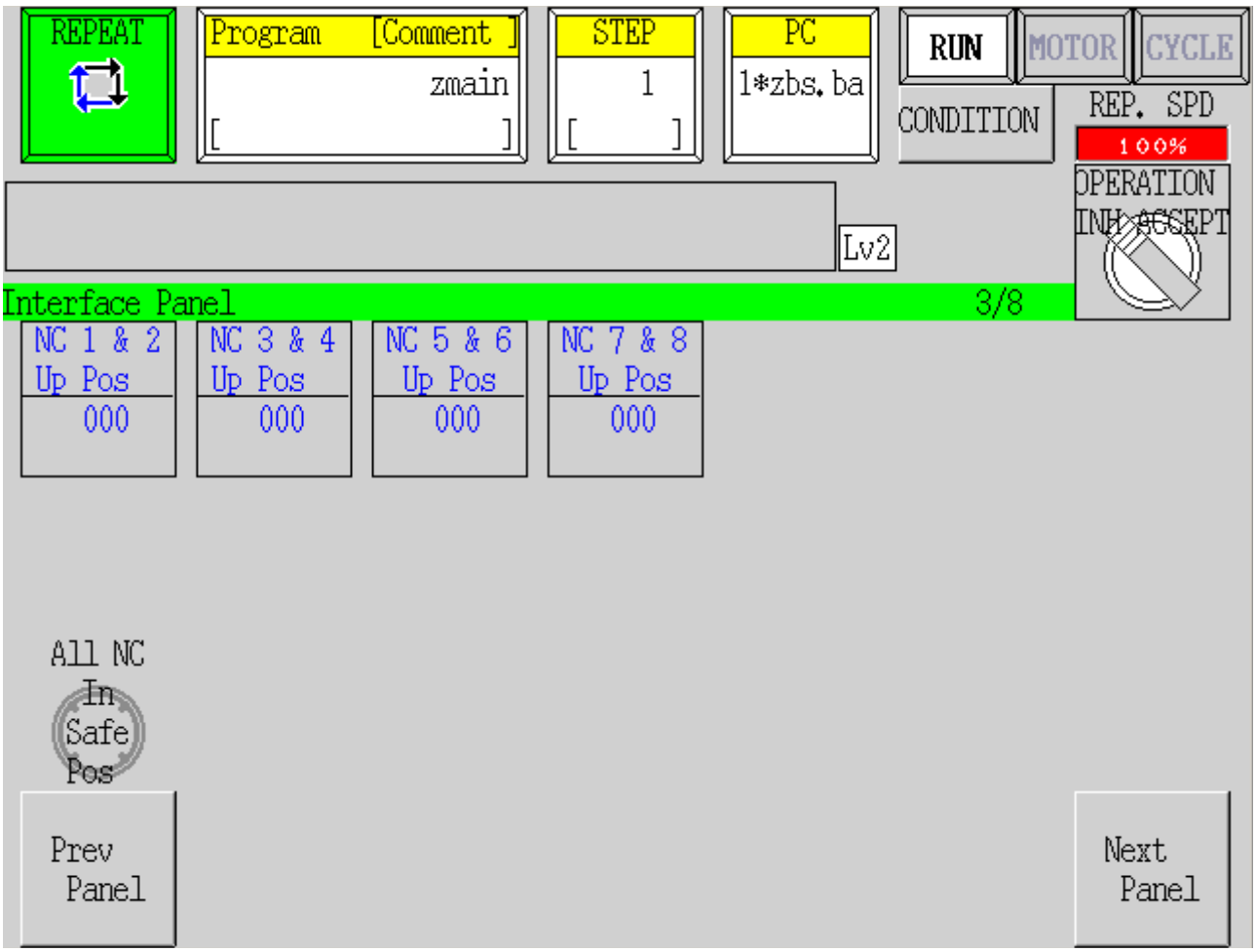
Human machine interface

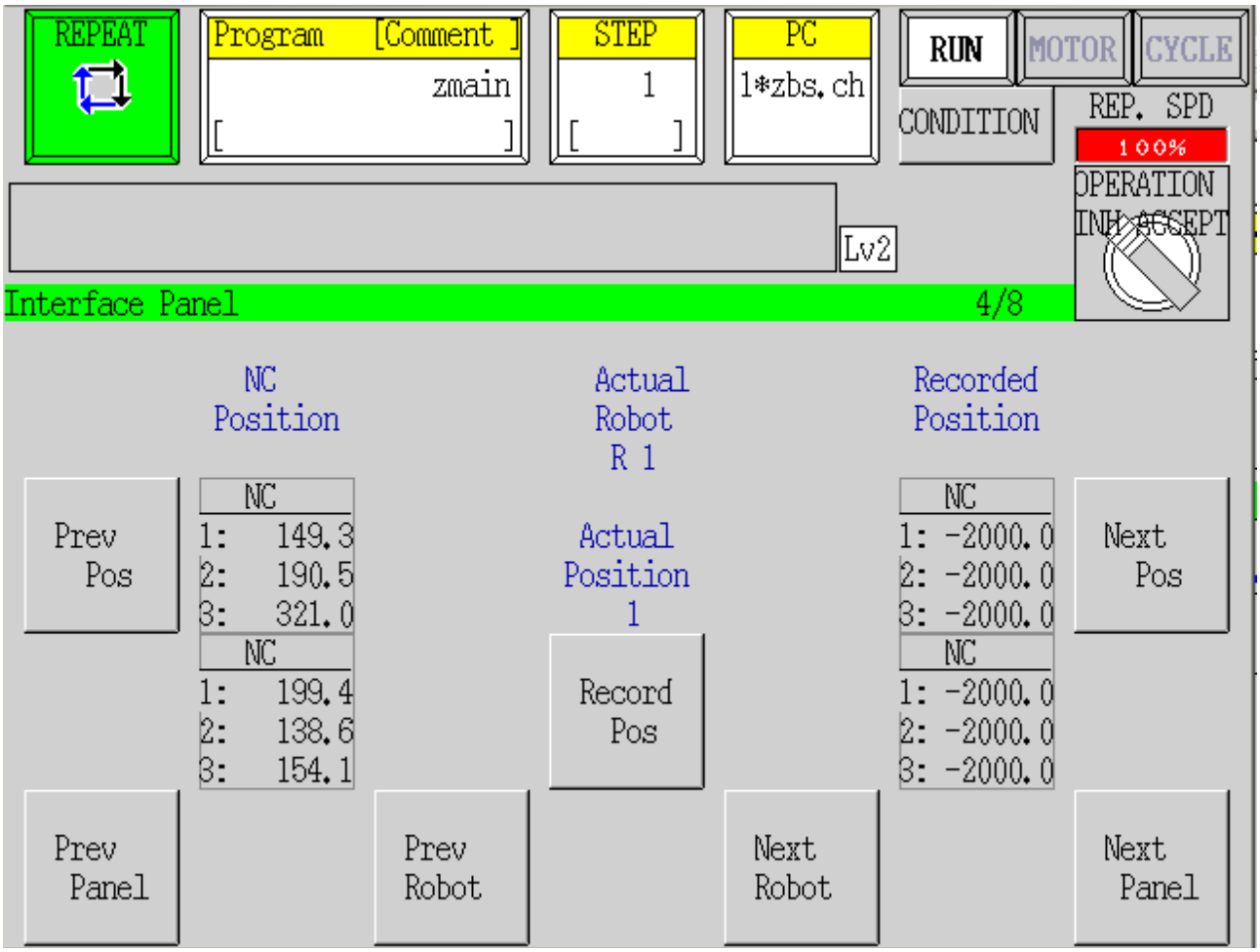
Description

NC locator HMI is composed of 3 panels :

- Panel 1 allow user to display clamp and part presence status
- Panel 2 allow user to display robot position (from 1 to 16)
- Panel 3 allow user to teach position 1 to 16 location




REPEAT
Program [Comment]
zmain
[ ]
STEP
1
[ ]
PC
1*zbs.ba
RUN
MOTOR
CYCLE
CONDITION
REP. SPD
100%
OPERATION
INH ACCEPT
Lv2
Interface Panel
3/8
NC 1 & 2 Up Pos
000
NC 3 & 4 Up Pos
000
NC 5 & 6 Up Pos
000
NC 7 & 8 Up Pos
000
All NC In Safe Pos
Prev Panel
Next Panel


REPEAT
Program [Comment]
zmain
[ ]
STEP
1
[ ]
PC
1*zbs.ch
RUN
MOTOR
CYCLE
CONDITION
REP. SPD
100%
OPERATION
INH ACCEPT
Lv2
Interface Panel
4/8
NC
Position
Actual
Robot
R 1
Recorded
Position
Prev
Pos
NC
1: 149.3
2: 190.5
3: 321.0
NC
1: 199.4
2: 138.6
3: 154.1
Actual
Position
1
Record
Pos
NC
1: -2000.0
2: -2000.0
3: -2000.0
NC
1: -2000.0
2: -2000.0
3: -2000.0
Next
Pos
Prev
Panel
Prev
Robot
Next
Robot
Next
Panel