IRBP Driver
Main Routine Example

Revision 1.1.2
Introduction
This document describes the main routine, which is shipped with the IRBP drivers. The document is written for people using the IRBP driver and wants to use the main routine as a base for development of a cell control program.

For a complete documentation of the IRBP driver functionality, please refer to the IRBP Driver Reference Manual.

What minimum modifications you need to do
The main routine in this example is the routine used when doing the long-run test before the positioner is delivered. Before the positioner can be used in your production you need to do some modifications:

1. Change the main() procedure to check for your Op Ready signal
2. Create user welding programs in the procedures ProgStn1 and ProgStn2, and remove the Stop; instructions
3. Create the tool service program (if used)
When you start a station for the first time, the INIT procedure starts. If the init procedure finishes, the bConfigDone flag will be set, and the INIT will not run the next time you start from MAIN.

The INIT procedure passes the following screens when executed:

**IRBP Safe position**

First, the Robot Safe Position is taught. The driver requires this position to safely perform a station interchange.
IRBP Interchange calibration

Then, the interchange should be calibrated. This sets the Home position (0 degrees) for the positioner. While the calibration is in progress, the positioner will move slightly around the estimated home position.

Positioner might move

This is a screen to make sure everything is out of the positioner’s working range, and that it is OK to start the calibration.
Calibrating...

The screen above is shown during calibration. It is normal that the calibration takes a few minutes. The counter on the screen will show the progress of the calibration process.

Calibration finished

When the calibration is finished, the above screen indicated that the positioner passed the calibration.
Fine Calibrate

After the calibration procedure is run, the positioner is resting in the Home position (0 degrees). You have now reached the point where you should do the Fine Calibrate, just as on the robot.

Process position, side 1

After the calibration, it is time to program the Process positions. A process position is the position the side towards the robot will be in after the interchange. Program it to be in a good starting position for the first weld to reduce cycle time.
Then it is time to program the Load positions. A load position is the position the side towards the operator will be in after the interchange. Program it in a good position for the operator to load/unload parts.

Positions - Station 2

After programming the process and load positions on side 1, the process is repeated on side 2.

Production mode

If the INIT procedure finishes, production mode will be entered.
%%
VERSION:1
LANGUAGE:ENGLISH
%%

MODULE IrbpMain
!
! This module is an example of a main routine
! which uses the positioner drivers. You can
! use it as a base for your program, or just use
! it to see how the drivers are used.
!
! Load data zero for simulation on the VC
LOCAL CONST loaddata zeroLoad:=[0.001,[0,0,0.001],[1,0,0,0],0,0,0];
! Counter for tool service interval
LOCAL PERS num nOptServ:=0;
! Number of cycles to run before tool service is performed.
! -1 = do not run tool service program
LOCAL CONST num nSERVICE_FREQ:=-1;
! Boolean to memorize if the setup has been run
LOCAL PERS bool bConfigDone:=FALSE;

LOCAL PROC CheckSafePos(num PosIndex)
!
! PROCEDURE CheckSafePos
! Checks if the robot is at a safe position.
! If it is at a safe position, do nothing.
! If the robot is not at a safe position, setup the safe position
! and activate the safety zone around the positioner.
!
VAR num nAnswer;
VAR jointtarget jSafe;
VAR string sPosname;
VAR speeddata sdM:=[1000,500,5000,90];

IF IrbpAtSafePos() RETURN;
jSafe:=IrbpGetSafePos(\PosIndex:=PosIndex\Status:=nAnswer);
IF nAnswer<>snOK THEN
  sPosname:=IrbpGetName(\PosIndex:=PosIndex\Status:=nAnswer);
  TPErase;
  TPWrite "IRBP Safe position ("+sPosname+")";
  TPWrite ".
  TPWrite "1. Press OK."
  TPWrite "2. Move the robot to a safe position"
  TPWrite "for the interchange."
  TPWrite "3. Restart program."
  TPReadFK nAnswer,"","","","",""OK"
  TPErase;
  Stop\NoRegain;
  ! Jog to a safe pos
  jSafe:=CJointT();
  ! Loadid needs a defined safepos
  IrbpSetSafePos PosIndex\NewSafePos:=jSafe;
ELSE
  MoveAbsJ jSafe, sdM, fine, tool0;
ENDIF

LOCAL PROC SetLoadAngle(
num stn)
!
! PROCEDURE SetLoadAngle
! This procedure defines the IRBP Load Position,
! i.e. the position of the station which is turned
! towards the operator after the station interchange.
!
VAR irbpstatus snStatus;
VAR jointtarget jLoad;
VAR num nAnswer;
VAR string sPosname;

sPosname:=IrbpGetName(
\AppStn:=stn\Status:=snStatus);
TPErase;
TPWrite "IRBP Load position ("+sPosname+");
TPWrite " 
; TPWrite "1. Press OK."
; TPWrite "2. Move IRBP to load position"
; TPWrite "3. Restart program.";
TPReadFK nAnswer, "", "", "", "", "", "OK";
TPErase;
Stop\NoRegain;
! Jog to load position
jLoad:=CJointT();
IrbpSetAppStnPos stn\LoadAngle, jLoad.extax\Status:=snStatus;
IF snStatus<>snOK THEN
  DisplayStatus snStatus\PosName:=sPosname;
EXIT;
ENDIF
ENDPROC

LOCAL PROC SetProcAngle(
num stn)
!
! PROCEDURE SetProcAngle
! This procedure defines the IRBP Process Position,
! i.e. the position of the station which is turned
! towards the robot after the station interchange.
!
VAR irbpstatus snStatus;
VAR jointtarget jProc;
VAR num nAnswer;
VAR string sPosname;

sPosname:=IrbpGetName(
\AppStn:=stn\Status:=snStatus);
TPErase;
TPWrite "IRBP Process position ("+sPosname+");
TPWrite " 
; TPWrite "1. Press OK."
; TPWrite "2. Move IRBP to service position"
; TPWrite "3. Restart program.";
TPReadFK nAnswer, "", "", "", "", "", "OK";
TPErase;
Stop\NoRegain;
! Jog to process position
jProc:=CJointT();
IrbpSetAppStnPos stn\ProcAngle, jProc.extax\Status:=snStatus;
IF snStatus<>snOK THEN
  DisplayStatus snStatus\PosName:=sPosname;
EXIT;
ENDIF
ENDPROC

LOCAL PROC TuneIt(
num PosIndex)
PROCEDURE TuneIt

This procedure is used for calibrating the positioner. It finds the 0-position of the interchange, and it must be run before using the positioner in production.

VAR irbpstatus Status;
VAR num nAnswer;
VAR string sPosname;

sPosname:=IrbpGetName(PosIndex:=PosIndex(Status:=nAnswer));
IrbpCalib(PosIndex:=PosIndex(Status:=Status);
IF Status<>snOK THEN
  DisplayStatus Status(PosName:=sPosname);
ENDIF
ENDPROC

LOCAL PROC Init()

PROCEDURE Init

This procedure initiates the positioner before first use after delivery. It shows what should be done before the positioner is used in production. Note that this initiation is usually performed very seldom – when the station is delivered, moved or something unusual has happened forcing a recalibration. A recalibration can be forced by setting the bConfigDone to FALSE.

VAR irbpstatus Status;
VAR num nAnswer;
VAR speeddata sdM:=[1000,500,5000,90];

IF NOT bConfigDone THEN
  ! Teach Safe Position
  CheckSafePos 1;
  ! Find station 1
  TuneIt 1;
  ! LoadId
  IF RobOS() THEN
    IrbpLoadId(PosIndex:=1);
  ELSE
    ! load valid loaddata
    IrbpSetLoadData 1,zeroLoad;
    IrbpSetLoadData 2,zeroLoad;
  ENDIF
  CheckSafePos 1;
  TPErase;
  TPWrite "IRBP Init";
  TPWrite " The station interchange might now ";
  TPWrite "move. Make sure that you keep your";
  TPWrite "safety distance";
  TPReadFK nAnswer,"","","","","","OK";
  IndexToStn 1(Speed:=sdM; IrbpActMechUnit(AppStn:=1(Status:=Status;
  IF Status<>snOK THEN
    DisplayStatus Status(AppStn:=1);
  EXIT;
  ENDIF
  ! Teach Process Position on side 1
  SetProcAngle 1;
  ! Teach Load Position on side 1
  SetLoadAngle 1;
  IrbpDeactMechU(AppStn:=1(Status:=Status;
  IF Status<>snOK THEN

DisplayStatus Status\AppStn:=1;
EXIT;
ENDIF
CheckSafePos 1;
TPErase;
TPWrite "IRBP Init";
TPWrite " The station interchange might now ";
TPWrite "move. Make sure that you keep your"
TPWrite "safety distance";
TPReadFK nAnswer,"","","","","","OK";
IndexToStn 2\Speed:=sdM;
IrbpActMechUnit\AppStn:=2\Status:=Status;
IF Status<>snOK THEN
    DisplayStatus Status\AppStn:=2;
    EXIT;
ENDIF
! Teach Process Position on side 2
SetProcAngle 2;
! Teach Load Position on side 2
SetLoadAngle 2;
IrbpDeactMechU\AppStn:=2\Status:=Status;
IF Status<>snOK THEN
    DisplayStatus Status\AppStn:=2;
    EXIT;
ENDIF
bConfigDone:=TRUE;
ENDIF
ENDPROC

PROC IndexToStn(
    num Stn
    \speeddata Speed)
    ! PROCEDURE IndexToStn
    ! This procedure performs the actual interchange.
    ! i.e. the position of the station which is turned
    ! towards the operator after the station interchange.
    !
    VAR string sPosName;
    VAR num nPosIndex;
    VAR irbpstatus snStatus;
    ! Get the name to use in error handling below
    sPosName:=IrbpGetName(\AppStn:=Stn\Status:=snStatus);
    IF snStatus<>snOK THEN
        DisplayStatus snStatus\AppStn:=Stn;
        EXIT;
    ENDIF
    ! Only increase the service counter if tool service is used to avoid
    ! overflow
    IF nSERVICE_FREQ>0 Incr nCptServ;
    ! Check if it is time to do tool service
    IF nCptServ<nSERVICE_FREQ AND nSERVICE_FREQ>0 THEN
        Clear nCptServ;
        ! Perform tool service by calling the service procedure
        MyServTraj;
   ENDIF
    ! Perform the interchange
    IrbpMoveToStn Stn\Speed?Speed\Status:=snStatus;
    IF snStatus<>snOK THEN
        DisplayStatus snStatus\PosName:=sPosName;
        EXIT;
    ENDIF
ERROR
    IF ERRNO<>snOK THEN

LOCAL PROC PreUserPrg(
  num AppStn)
!
! PROCEDURE PreUserPrg
! This procedure should be run before the user program.
! It activates the correct axis, and moves the station
! to the process angle.
!
VAR irbpstatus snStatus;
!
! Activate the unit
IrbpActMechUnit\AppStn:=AppStn\Status:=snStatus;
IF snStatus<>snOK THEN
  DisplayStatus snStatus\AppStn:=AppStn;
  EXIT;
ENDIF
!
! Move the station to proc angle
IrbpMoveAppStn AppStn\ProcAngle,v300\Status:=snStatus;
IF snStatus<>snOK THEN
  DisplayStatus snStatus\AppStn:=AppStn;
  EXIT;
ENDIF
ENDPROC

LOCAL PROC PostUserPrg(
  num AppStn)
!
! PROCEDURE PostUserPrg
! This procedure should be run after the user program.
! It deactivates the axis, and moves the station
! to the load angle to prepare for the user to load/unload the part.
!
VAR irbpstatus snStatus;
!
! Move the station to load angle
IrbpMoveAppStn AppStn\LoadAngle,v300\Status:=snStatus;
IF snStatus<>snOK THEN
  DisplayStatus snStatus\AppStn:=AppStn;
  EXIT;
ENDIF
!
! Deactivate the unit
IrbpDeactMechU\AppStn:=AppStn\Status:=snStatus;
IF snStatus<>snOK THEN
  DisplayStatus snStatus\AppStn:=AppStn;
  EXIT;
ENDIF
ENDPROC

PROC ProgStn1()
!
! This is the work program for side 1
! Put your process instructions here or
! put a procedure call for your program
!
! Remove stop instruction below before
! production.
!
Stop;
!
ENDPROC
PROC ProgStn2()
! This is the work program for side 2
! Put your process instructions here or
! put a procedure call for your program
! Remove stop instruction below before
! production.
! Stop;
! ENDPROC

PROC MyServTraj()
! This is the procedure to do tool service
! Put your instructions here or
! put a procedure call for your service program
! Remove below stop instruction before
! production.
Stop;
! ENDPROC

PROC main()
Init;
WHILE TRUE DO
  CheckSafePos 1;
  ! Check OP READY (here simulated by the wait below)
  WaitTime 5;
  ! Goto station 1
  IndexToStn 1;
  ! Activate stn1
  PreUserPrg 1;
  ! Run user prog stn1
  ProgStn1;
  ! Deactivate stn1
  PostUserPrg 1;
  CheckSafePos 1;
  ! Check OP READY (here simulated by the wait below)
  WaitTime 5;
  ! Goto station 2
  IndexToStn 2;
  ! Activate stn2
  PreUserPrg 2;
  ! Run user prog stn2
  ProgStn2;
  ! Deactivate stn2
  PostUserPrg 2;
ENDWHILE
! ENDPROC
ENDMODULE