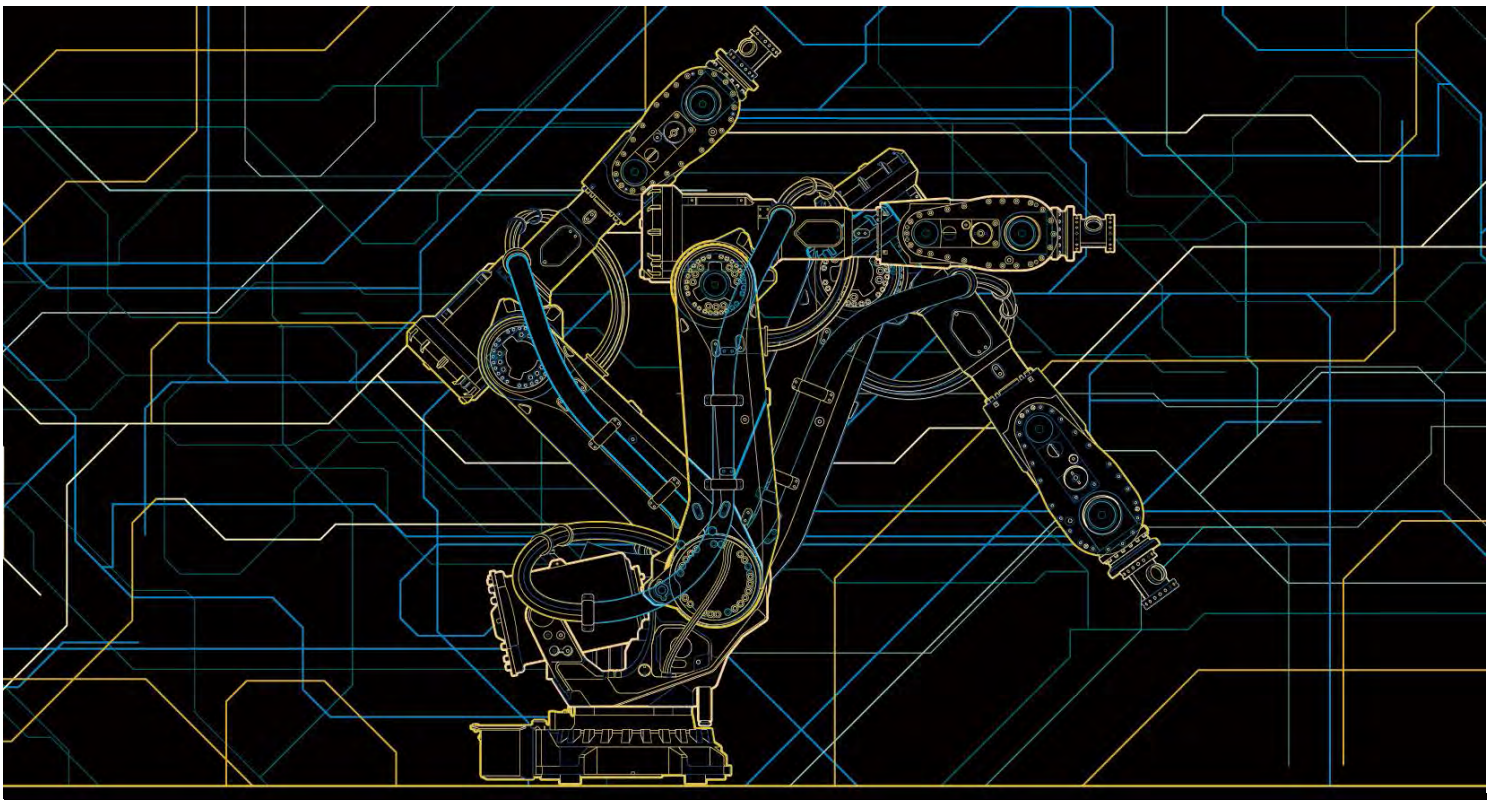


Application manual
PickMT with ABB robot

Power and productivity
for a better world™

Trace back information:
Workspace Main version a138
Checked in 2015-12-20
Skribenta version 4.6.209

Application manual
PickMT with ABB robot

Document ID: 3HAC051770-001
Revision: A

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damages to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

Additional copies of this manual may be obtained from ABB.

The original language for this publication is English. Any other languages that are supplied have been translated from English.

ABB AB
Robotics Products
Se-721 68 Västerås
Sweden

Table of contents

This page is intentionally left blank

Overview of this manual

About this manual

This manual describes the robot integration for PickMT.

Usage

User manuals are used to understand how to use the product, for example to install, configure, or operate.

Users

This manual is intended for:

- Personnel that are responsible for installation and configuration of robot systems
- Programmers
- Service engineers

Trademarks

FlexMT is a trademark of ABB.

PickMT is a trademark of SVIA, Svensk Industriautomation AB.

References

Reference	Document ID
Product specification - FlexMT	*3HAC049820-001*
Product manual - FlexMT	*xyz-1*
Product manual - IRB 2600	*3HAC035504-001*
Product manual - IRB 4600	*3HAC033453-001*
Application manual - PickMT	*3HAC051771-001*
Product manual - Safety center for FlexMT	*3HAC051769-001*
Application manual - FeedLine Light	*3HAC052311-001*

Revisions

Revision	Description
-	First edition.
A	Minor corrections.

This page is intentionally left blank

1 Introduction

PickMT is a vision system designed to guide industrial robots during materials handling. A camera is used to identify the location, position and orientation of a detail and this information is sent to the robot. The robot can then pick up or manipulate the detail in some other way without the need for any mechanical fixtures.

This manual describes how an ABB robot communicates and works together with PickMT. Read this manual together with the PickMT manual to integrate an ABB-robot with PickMT.

This page is intentionally left blank

2 Calibration

In order for PickMT to send the correct coordinates to the robot, the system must be calibrated together, e.g. PickMT's co-ordinate system and the robot's co-ordinate system must be made to correspond. The supplied calibration plate and calibration tool are used for this purpose.

The calibration process varies, depending on whether PickMT is used with a machine that has a Stop & Go conveyor or one that has Conveyor Tracking.

Section 6.1 describes the calibration together with the robot of a machine with stop and go conveyor.

Section 6.2 describes the calibration together with the robot of a machine with conveyor tracking.

See PickMT manual for camera calibration.

Continues on next page

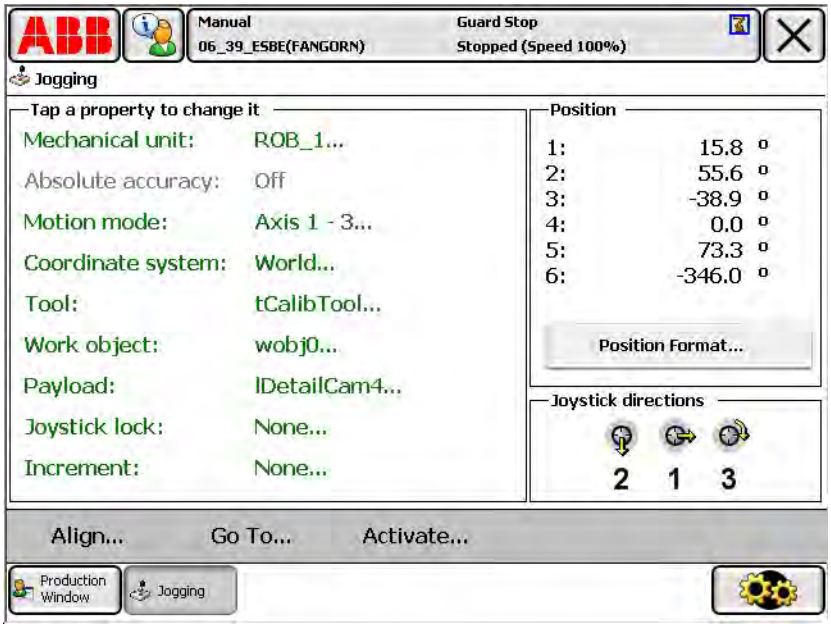
2.1 Machine with Stop & Go conveyor

2.1.1 Calibration ABB (IRC5)

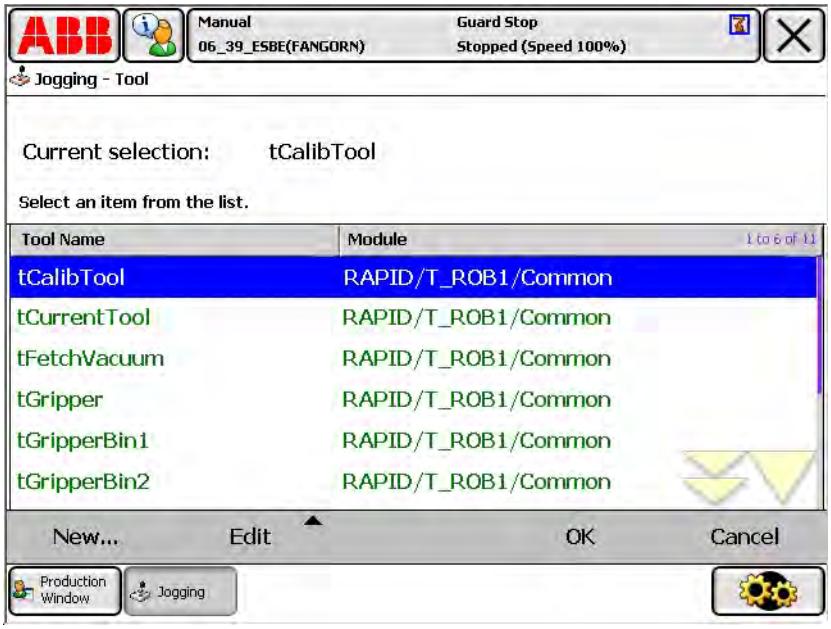
After calibration of PickMT is complete you will need to calibrate the robot. Note that the calibration plate must not be moved until calibration of the robot is complete. To be sure the calibration plate can be taped to the camera belt.

Continues on next page

Attach the supplied calibration tool to the robot. Note that the calibration tool may look different to that shown in image above, depending on the type of robot and which other gripper equipment is used in the specific application.



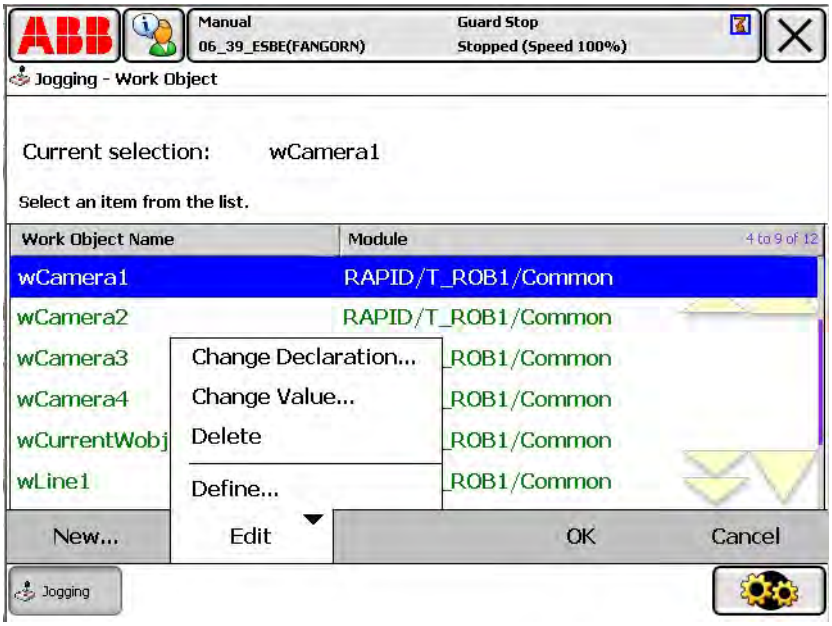
Open the jogging menu and select the tCalibTool.



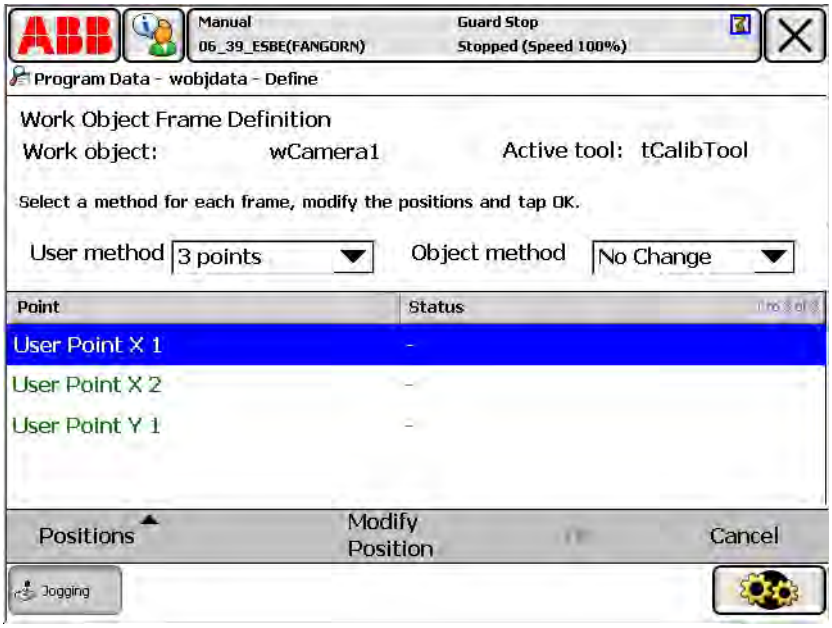
Continues on next page

2.1.1 Calibration ABB (IRC5)
Continued

Then open ”Programdata”, select ”wobjdata” and press ”Show data”.

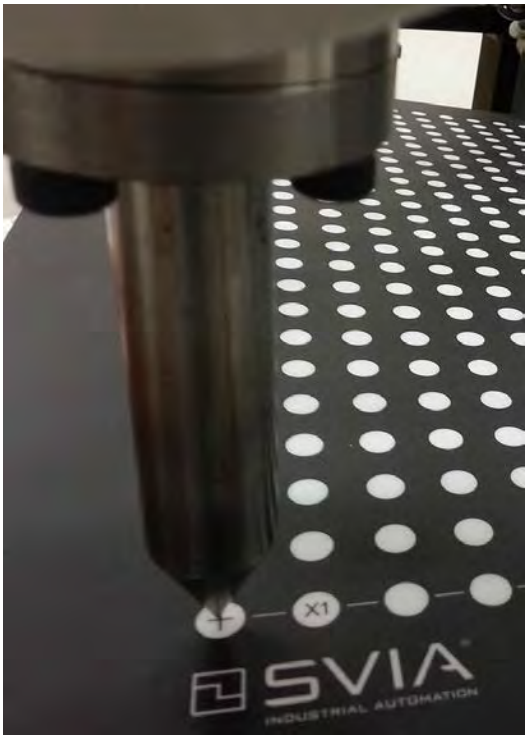


Mark the work object to be calibrated. wCamArea1 for camera 1, wCamArea2 for camera 2, etc. Then press ”Edit” and select ”Define”. Do NOT calibrate wCamera, it is only used temporarily in the robot.

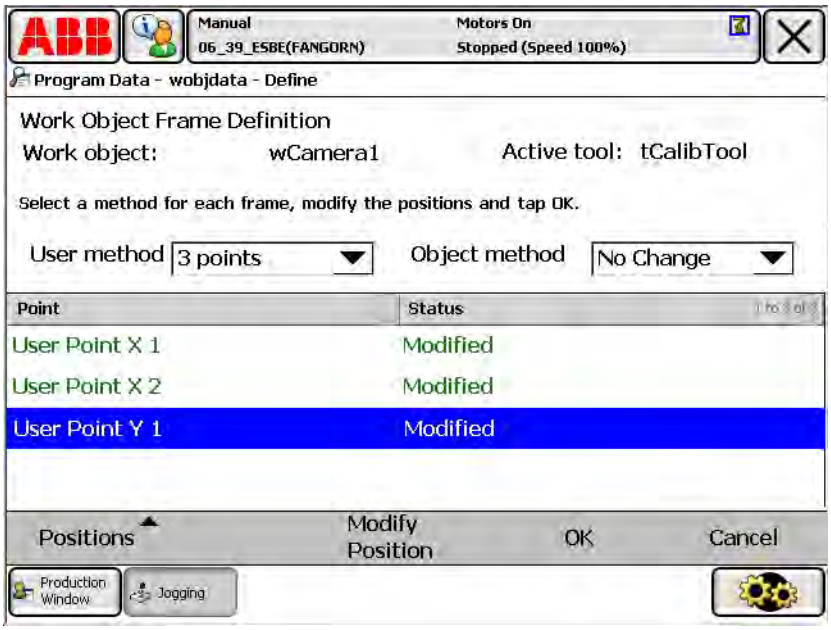


Continues on next page

Under ”Use.method” ”3 points”, must be selected. Under ”Object method” ”No change” must be selected.



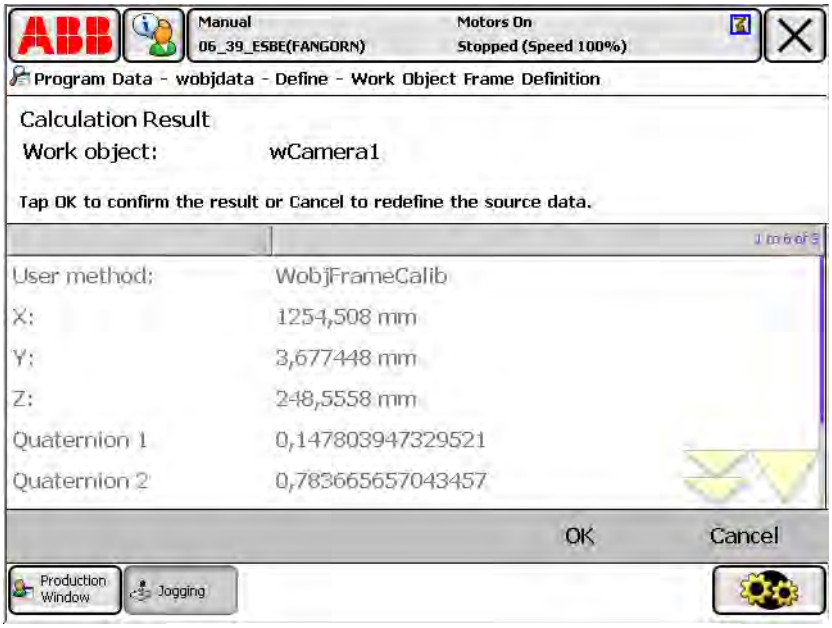
Then jog the robot so that the tip of the calibration tool is above point X1 on the calibration plate, see image above. Alternatively a point on the X-axis near X1 can be selected.



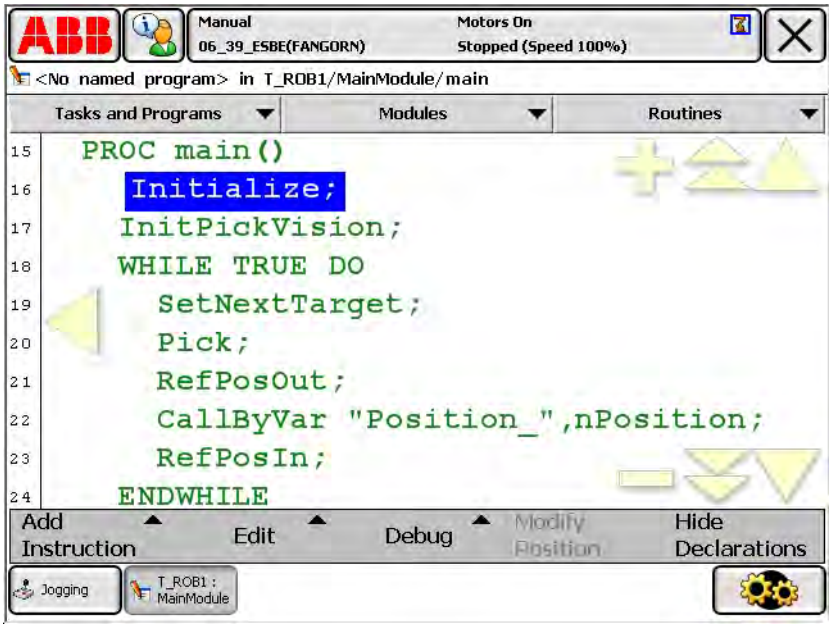
Continues on next page

2.1.1 Calibration ABB (IRC5)
Continued

Then mark ‘User point X1’ and press “Change mode”. Repeat the jogging and position modifying process for positions X2 and Y1. Then press OK.

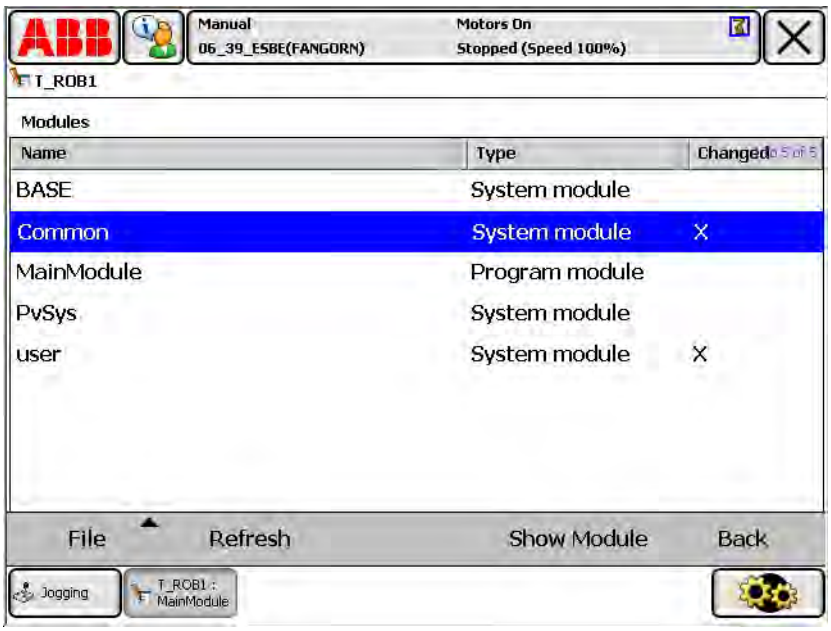


A summary of the calibration results will now be displayed. The system is now calibrated and ready, but the calibration must be saved to file to avoid the risk of losing it when the system is next started up.

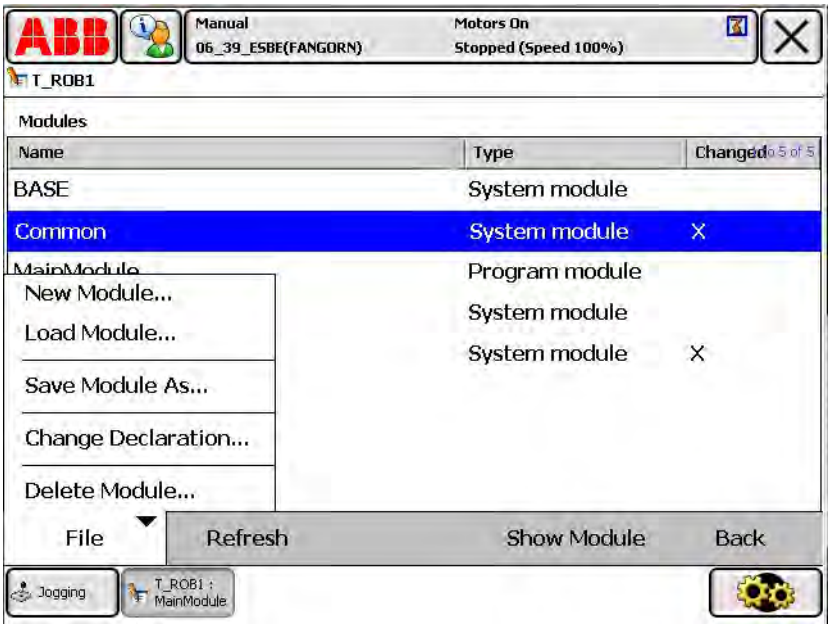


Continues on next page

Open ”Programeditor”.



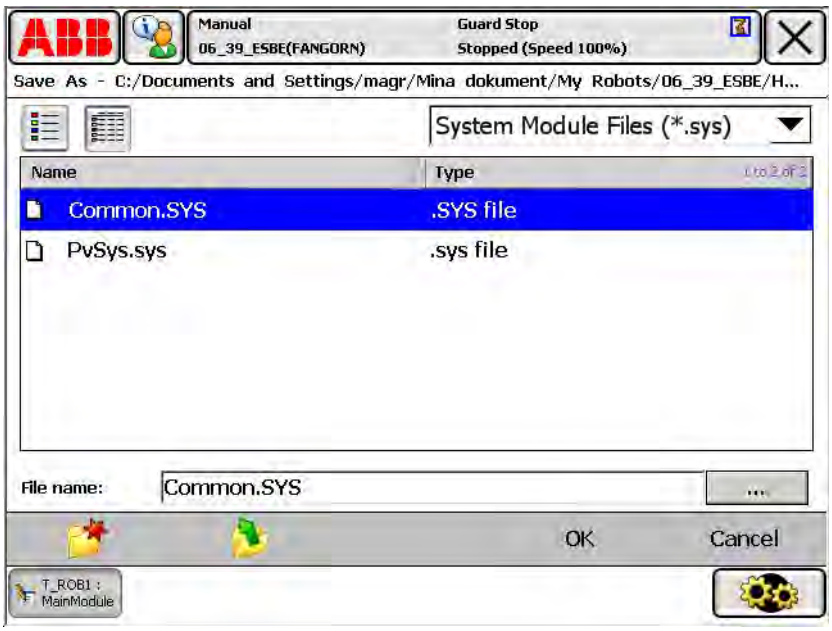
Choose ”Modules” and mark the module “Common”.



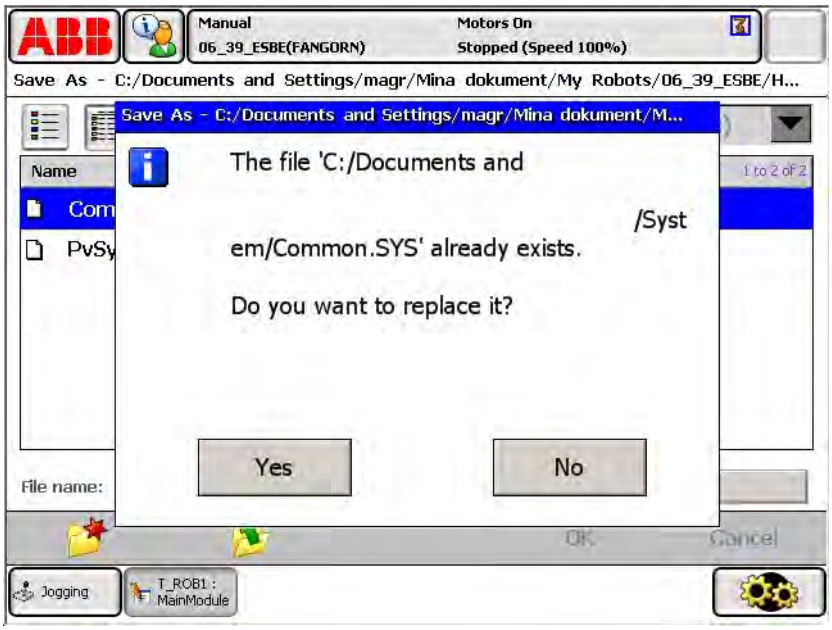
Continues on next page

2.1.1 Calibration ABB (IRC5)
Continued

Press ”File” and select ”Save module as...”.



Go to the folder ”HOME:/PickMT/System”. There should already be a file called ”COMMON.SYS” in this folder. Click on ‘OK’.



Continues on next page

Confirm the message to replace the file with ”Yes”.

Confirm the next message to replace the file with ”OK”. The calibration has now been saved and the system is ready to start working with the current camera.

2.1.2 Calibration 3D

Prior to calibration between robot and 3D scanner, wCamera1 must be defined (if 3D scanner is camera 1). In this case, no calibration plate is used, and it is entirely up to the robot operator to position the coordinate system appropriately.

Suitable positioning is, for example, on the pallet guides (in case a pallet is used). The Z axis should be pointing upward and the X axis should be oriented along the long side of the pallet. See the previous section for help with defining and saving the work object.

The RAPID module Calibration3D.sys includes robot demo code that supports the calibration process. 15 recommended positions are defined. Mount and activate the calibration cone as a tool with the TCP at the top of the cone. Check the predefined positions. Modify positions if necessary, it is possible to place them in many different ways.

For best results, it is preferable to include as much as possible of the 3D sensor measurement volume used for robot picking. Please note that it is possible to add or delete positions. Change Calibration3DRoutine accordingly so that the robot moves to the positions to be used.

When PickMT requests to start the robot, the routine Calibration3DRoutine shall be called. If all movements are tested, the robot can be switched to automatic mode to simplify the calibration process. The RAPID modules contains code that communicates with PickMT and sends all the information needed to PickMT.

Observe that the module name Calibration3D may not be changed. Neither the names of the communication variables nConeCalibrationAnswer, n3DCalibrateX, n3DCalibrateY, and n3DCalibrateZ.

2.2 Machine with Conveyor Tracking

Conveyor Tracking is a function that is used when the robot is working with a moving conveyor. This mode allows the robot's movements to be co-ordinated automatically with the conveyor.

The following sections only describe how to calibrate Conveyor Tracking with PickMT. For a complete description of Conveyor Tracking refer to the manuals for the relevant robot manufacturer.

Continues on next page

2.2.1 Calibration

These instructions are intended for experienced robot technicians. If these are not sufficient, see the product manual for the robot and PickMaster.

1. Add the calibration plate with the X axis in the belt's direction of movement. The plate must be adjusted so that the edge of the plate is parallel with the edge of the belt, i.e. it is not necessary for the plate to be straight in the image.

2. Remove the frame in front of the camera belt so that the calibration plate can run across the belt edge without hitting anything.

3. Tape the rear edge of the calibration plate securely against the belt.

4. Calibrate in PickMT as usual.

5. Open the ”ppacal.prg” program, which is in the Home folder on the robot. Note that it is in the old S4 format. Also note that when operating with a Basler camera, PylonViewer must be opened, set Trigger Mode to 'On' there and start taking images from Pylon Viewer. Then run ”ppacal.prg”. Do not forget to reset Trigger Mode when finished.

6. Start the program and select calibration of ”CNV” as well as ”CNV1”.

7. When it is stated that the robot is ready for calibration the program can be stopped.

8. The camera belt must now be run forwards manually until the 0 point (origin) on the calibration plate is inside the robot's range. For example, start/stop is set on one of the robot panel's rapid buttons.

9. Install the calibration device on the robot and ensure that the correct tools is selected in the robot.

10. Then go to Calibration -> CNV1 -> Baseframe in the robot. Select 4 point calibration.

11. Mark the origin on the calibration plate with the calibration device and update point 1. Then move the robot and run the belt forward slightly. Mark the origin again on the calibration plate with the calibration device and update point 2.

12. Perform the same for points 3 and 4.

13. Once calibration is complete results are given about how successful the calibration was. The average margin is the most interesting here and it should be below 0.5 mm for the machine to run optimally. If the average margin is greater than this, calibrate again until a better calibration is carried out.

14. Restart the controller.

15. Calibration complete!

2.2.2 Configuring

Not currently available.

This page is intentionally left blank

3 Robot Program

This chapter only gives the basic structure of the robot's program. More or less extensive additions and adjustments are required depending on the design of the actual robot cell. For full information on programming and program structure see relevant robot's product and programming manual.

Continues on next page

3.1 Standard IRC5

3.1.1 Main program

The program ‘PVMain.pgf’ is always loaded into the robot’s program memory. This program in turn loads ’MainModule.mod’, which is always the same for all details and all operating modes. The module is responsible for loading the modules that are linked to the details to be run. Below these are called ‘Modcam1’ for details to be run using camera 1, ‘Modcam2’ for details to be run using camera 2, and so on. So it is these that are specified during ‘Teachin’ in PickMT.

The program memory always contains two loaded system modules, ’PvMcSys’ for all communication with PickMT and ’Common’ for managing common program data and code.

Continues on next page

3.1.1.1 PvMain

‘PvMain’ is always supplied with PickMT and is built up as shown above.

’PvMain provides the following routines and data:

- Routines:
  - Main() is pre-programmed and should not be modified more than at the places where it is explicitly stated that conditions must or can be added.
  - CheckSystem() checks whether anyone has requested entry to the cell or has been inside the cell. The routine also ensures that new images are taken when the cell is started after someone has been inside. This happens as it can no longer be guaranteed that the detail remains under the camera.
  - InitializeMain(): This contains all the necessary initializations for the main program.
  - LoadCameraModules() loads relevant module files for each camera. The name of each module that is to be loaded is provided by PickMT.
  - PickCamera() is called up when the robot is to retrieve a new detail by the camera. This routine calls up several sub routines to request new co-ordinates and perform movements to and from the pick position.
  - Position_n() is called up after PickCamera(). This routine then calls up the correct position routine in the correct ModCamX depending on which camera was used at the last pick and in which position PickMT identified the detail.
  - StopRoutine() calls up the correct stop routine or stop routines in ModCamX. This happens when PickMT stops.

3.1.1.2 ModCam1

- Routines:
  - Cam1Position_n(): Once a detail has been gripped and ‘RefPosOut()’ has been run, various routines are called up depending on which position has been gripped. If position 1 is identified by camera 1 as having been gripped, the ’Cam1Position_1()’ routine is called up; If position 2 is identified by camera 1 as having been gripped, the ’Cam1Position_2()’ routine is called up. All programming that involves the handling of different positions by each camera should therefore be done using these routines. Each ’Cam1Position_n()’ routine must be carried out so that RefPosIn() can be called up directly afterwards.
  - InitializeCam1() contains camera specific initializations for camera 1. Called up automatically at program start.
  - LoadMachine() is filled by the robot programmer. This routine can be used when loading the machine and called up from Cam1Position_n.
  - PickCam_1() picks a detail from the conveyor.
  - RefPosInCam_1() is run when the robot has handled a detail and is heading towards the camera to pick a new detail.
  - ’RefPosOutCam_1()’ is run when the robot has picked a detail and is heading away from the camera to handle the detail.
  - StopRoutineCam_1() is called up when the robot receives a stop from PickMT. For example, can be used to move to a standby mode.

The same content, with changed digits for respective camera, is found in the modules ModCam2, ModCam3, etc. The robot technician makes most additions to the ModCam files during machine operation. All variables that are detail specific are declared in these modules.

3.1.1.3 PvMcSys

This module contains some routines needed for communication between the robot and PickMT and must not be changed.

PvMcSys provides the following routines and data:

Normal use:

- Routines
  - ConfirmPick1..4(). If SetNextTarget returns valid grip information, the ConfirmPick must be called up by the user to run the conveyor on and complete the started Grab-Pick sequence. This usually occurs by calling up RefPosOut. NOTE: Note that one GrabImage must always be confirmed with SetNextTarget and ConfirmPick before the next GrabImage is started.
  - DiscardAndTakeNewImage(nCamera) is called up by the user if a new image is to be taken in PickMT before all sent co-ordinates are used. NOTE: Routines such as DiscardAndTakeNewImage() are to be used with care. If the normal GrabImage-SetNextTarget-ConfirmPick chain is deviated from it is extremely important to have full control of all image taking to prevent taking double images (new image taken before previous image is managed).
  - GrabImage(nCamera) is called up by the user to take a new image with the selected camera. The routine waits until the accompanying signal InPosition is 1 and sends a command to PickMT to take a new image with nCamera. NOTE: If the camera is set to ALLOW_AUTO_GRAB = TRUE (set in IntitializeCamX in ModCamX) do not use GrabImage. New images will be requested automatically after ConfirmPick1..4().
  - InitPickMT() is called up at the beginning of the program to initialise the communication with PickMT.
  - PvInputBox() can be called up to write an input query on the PickMT user interface. The function returns the reply that the operator has entered.
  - PvMessageBox() can be called up to write a message on the PickMT user interface and read off the operator’s reply (Ok, Yes, No).
  - SendPvStatus() returns current operating status in PickMT (PV_IDLE, PV_OPERATION, PV_STARTING, PV_STOPPING)
  - SetNextTarget(nCamera) is called up to retrieve the next valid grip position from PickMT for camera nCamera. The function returns when the coordinates for a valid detail that can be gripped are available. It is possible to specify that only one selected teachin position in PickMT is to be returned.
  - StopPickMT() stops PickMT at the robot’s request.
  - WriteLog() writes a message directly to the PickMT log file.

Continues on next page

3.1.1.3 PvMcSys
Continued

- Important data:
  - BELT_ACTION{4} must be set after the desired system behaviour for each camera. (Default: RUN_ONE_DETAIL: Run the conveyor belt directly after the last detail has been picked, RUN_NO_DETAIL: Run the conveyor belt directly after the last detail has been found in the image, RUN_NEVER: Never run the conveyor belt, RUN_MANY_DETAILS: Run the conveyor belt after each pick). Normally set from InitializeCamX in ModCamX.
  - ALLOW_AUTO_GRAB{4} must be set after the desired system behaviour for each camera. (TRUE: When the belt has been run and the InPosition signal set a new image is taken by the automatic device. FALSE: Imaging must be explicitly initiated by the robot program by calling up GrabImage). Normally set from InitializeCamX in ModCamX.

Important to note when running with setting 'RUN_NEVER', e.g. when picking from pallets. If, in this instance, PickMT does not find anything after taking a new image, the robot must have some type of management for this. The user must check the 'Action' variable before calling up SetNextTarget. If it is equal to 1 ('NO_DETAIL') the appropriate action must be taken. This can be taking a new image several times before the cell is stopped, for example. If SetNextTarget is called up when 'Action' is 1 the robot program stops.

Misc.:

There are various other routines that can be useful. Most can be found below. Some routines are only help routines and are only called up by other routines in PvMcSys. These are not named here.

- Routines
  - CameraContrast(nCamera, nValue) is called up to change the camera contrast.
  - CameraExposureTime(nCamera, nValue) is called up to change the exposure time in the camera.
  - CameraGain(nCamera, nValue) is called up to change camera brightness.
  - ClearCoordinates() – for internal use only.
  - DisablePosition(nCamera, nPosition) is called up to deactivate a certain teachin position in PickMT. All positions can be locked at the same time by specifying a value nPosition > 999.
  - DiscardAndStartBelt(nCamera) is called up to discard previous co-ordinates and start the belt. Use with caution, see DiscardAndTakeNewImage.
  - EnablePosition(nCamera, nPosition) is called up to activate a certain teachin position in PickMT. All positions can be enabled at the same time by specifying a value nPosition > 999.
  - ForceInPosition() – for internal use only.
  - isPositionPresent(nCamera, nPosition) is called up to check whether a certain position is among the results from PickMT.

Continues on next page

- numPositionPresent(nCamera, nPosition) is called up to check how many times a certain position occurs in the results sent by PickMT.
- ResetAlarm(sMessage) is called up to reset a certain alarm that was set in PickMT from the robot.
- ResetInformation(sMessage) is called up to reset a specific information message that was set in PickMT from the robot.
- ResetWarning(sMessage) is called up to reset a certain warning that was set in PickMT from the robot.
- SendBlackRegion(sBlackRegion) sends a reconfiguration to PickMT to concentrate the camera's field of view on the selected areas. The user can either make a rectangular area invisible (=black image), or make the area outside a selected rectangle. The areas are specified in the format A-BBB-CCC-DDD-EEE-F. A indicates camera number (1-4), BBB indicates in percent where the rectangle starts in the x-axis (000-100), CCC indicates in percent where the rectangle ends in the x-axis (000-100), DDD indicates in percent where the rectangle starts in the Y-axis (000-100), EEE indicates in percent where the rectangle ends in the Y-axis(000-100),F indicates whether the actual rectangle becomes black (1) or if the surrounding area becomes black(0).
- SendFreeParameter1(nCamera) / SendFreeParameter2(nCamera) returns the values that the user has entered at detail level during teachin. Camera for applicable inquiry must be indicated.
- SendClearSend() - for internal use only.
- SendConfirmInterlayerPickCam1...4() is called up to confirm to PickMT that an interlayer has been picked. Used in certain instances when picking from pallets.
- SendConfirmPickCam1...4() - for internal use only.
- SelectDetailOnTheFly(nCamera, sGroupName, sDetailName) is used to change detail during operation. Use with caution.
- SendEdgeHeight(nCamera, nEdgeHeight) is called up to set new edge height in the selected camera. Usually used when running stacks on the belt.
- SendGrab(nCamera) - for internal use only.
- SendStop() - for internal use only.
- SetAlarm(sMessage) is called up to set an alarm in PickMT from the robot.
- SetInformation(sMessage) is called up to set an information message in PickMT from the robot.
- SetWarning(sMessage) is called up to set a warning in PickMT from the robot.
- StartBelt(nCamera) starts the belt under the selected camera. Usually only used from PvCom.
- WriteFilePickMT(sFileName, sMessageData) is called up to write a text to the selected file name on the PickMT computer.

3.1.1.4 Common

All the data for tools and work objects must be stored in the ‘Common’ system module. This system module is accessible by all programs, and by storing the data in this module all programs always have access to the right data. All other data that is common to all programs as well as program code that is common to all programs should be stored here.

The following routines are found in Common:

- DefWorldZones() is called up automatically when the robot is switched on. From this routine DefForbiddenZone, DefSafeZone and DefMachineZone are called up.
- DefForbiddenZone() activates a user defined area where the robot may never be. This must normally be positioned directly above the robot so that it cannot move ”overhead”.
- DefSafeZone() activates a user defined area where it is safe for the robot to start its program from the start.
- DefMachineZone() activates a user defined area for the machine that the robot manages. A DOF_LoaderOut signal connects to the zone. The signal is always high when the robot is outside this zone.
- isDistanceOK(rCheckPosition, wWorkObject, nMaxDistance) is called up to check whether the robot is within the maximum distance from a certain point.
- SafeRobotStart() is always called up first in the main program The routine warns the user that the robot is too far from its home position at start-up.
- Tool(...) is called up to open, close the robot’s gripper. If the robot uses a suction cup the routine can also be used to switch on or off the air to it.

Example program

The following example program only shows the principle for how a robot program should communicate with PickMT.

There is no guarantee that any of the examples work except that it has been adjusted and tested for the specific robot cell.

```
Copyright (c) 2011, Svensk Industriautomation AB (SVIA)
All rights reserved.
Redistribution and use in source and binary forms, with or without
     modification, are permitted provided that the following
     conditions are met:
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
     “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT
     NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
     FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
     SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY
     DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
     CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
     PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
     DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
     AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
     LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
```

Continues on next page

```
     IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
     OF THE POSSIBILITY OF SUCH DAMAGE.
```

Example 1

Running with up to 4 cameras (Stop&Go) where picking is to occur from the camera where PickMT identified the detail.

```
PERS num BELT_ACTION{4}:=[RUN_ONE_DETAIL, RUN_ONE_DETAIL,
     RUN_ONE_DETAIL, RUN_ONE_DETAIL];
PERS bool ALLOW_AUTO_GRAB{4}:=[TRUE, TRUE, TRUE, TRUE];
PROC Main()
InitializeMain;
InitPickMT;
WHILE TRUE DO
CheckSystem;
TEST sState
CASE ”Idle”:
! Condition to pick from camera (1..4) and call correct Position
     routine
IF bCoordReceived{CAMERA_NO_1}=TRUE OR
     bCoordReceived{CAMERA_NO_2}=TRUE OR
     bCoordReceived{CAMERA_NO_3}=TRUE OR
     bCoordReceived{CAMERA_NO_4}=TRUE THEN
sState:=”PickCameraAndPosition_n”;
PickCamera;
Position_n;
ENDIF
! Additional conditions could be added here...
CASE ”PickCameraAndPosition_n”:
sState:=”Idle”;
DEFAULT:
ENDTEST
WaitTime 0.05;
ENDWHILE
ENDPROC
```

Example 2

Running with two cameras, PalletPicker together with camera for post adjustment/re grip. Taking images manually.

```
PERS num BELT_ACTION{4}:=[RUN_NEVER, RUN_NEVER, RUN_NEVER,
     RUN_NEVER];
PERS bool ALLOW_AUTO_GRAB{4}:=[FALSE, FALSE, FALSE, FALSE];
!
PROC main()
InitializeMain;
InitPickMT;
WHILE TRUE DO
CheckSystem;
GrabImage CAMERA_NO_1;
RefPosIn_CAMERA_NO_1;
SetNextTarget CAMERA_NO_1;
WHILE nAction{CAMERA_NO_1} <= NO_DETAIL DO
```

Continues on next page

3.1.1.4 Common
Continued

```
!Possible to change camera settings here before new try to find
     detail
DiscardAndTakeNewImage CAMERA_NO_1;
WaitUntil bCoordReceived{CAMERA_NO_1}=TRUE;
SetNextTarget CAMERA_NO_1;
Incr nImageRetries{CAMERA_NO_1};
IF nImageRetries{CAMERA_NO_1} > MAX_IMAGE_RETRIES{CAMERA_NO_1} THEN
!Create alarm or inform user in some other way
EXIT;
ENDIF
ENDWHILE
IF (nAction=CHANGE_BIN) THEN
! make the operator change the bin
RefPosOut_CAMERA_NO_1;
ELSEIF (nAction=PICK_INTERLAYER) THEN
! make the robot pick the interlayer. pPick.pos.z contains the
     estimated pick height for interlayer
RefPosOut_CAMERA_NO_1;
ELSEIF (nAction=PICK_INTERLAYER_AND_CHANGE_BIN) THEN
! make the robot pick the interlayer. pPick.pos.z contains the
     estimated pick height for interlayer
! make the operator change the bin
RefPosOut_CAMERA_NO_1;
ELSE
PickCam_CAMERA_NO_1;
RefPosOut_CAMERA_NO_1;
ToRegripCamera;
RefPosIn_CAMERA_NO_2;
GrabImage CAMERA_NO_2;
SetNextTarget CAMERA_NO_2;
WHILE nAction{CAMERA_NO_2} <= NO_DETAIL AND
nImageRetries{CAMERA_NO_2} < MAX_IMAGE_RETRIES{CAMERA_NO_2} DO
!Possible to change camera settings here before new try to find
     detail
DiscardAndTakeNewImage CAMERA_NO_2;
WaitUntil bCoordReceived{CAMERA_NO_2}=TRUE;
SetNextTarget CAMERA_NO_2;
Incr nImageRetries{CAMERA_NO_2};
ENDWHILE
IF nAction{CAMERA_NO_2} THEN
!Still no detail found
!Create alarm and EXIT or
RefPosOut_CAMERA_NO_2;
!Move detail to trash
ELSE
RefPosOut_CAMERA_NO_2;
CallByVar "Cam2Position_",nPosition;
ENDIF
ENDIF
ENDWHILE
ENDPROC
```

Continues on next page

Example 3

Running with one camera that is used for both FeedLine and post adjustment.

```
PERS num BELT_ACTION{4}:=[RUN_NO_DETAIL, RUN_NEVER, RUN_NEVER,
     RUN_NEVER];
PERS bool ALLOW_AUTO_GRAB{4}:=[TRUE, FALSE, FALSE, FALSE];
!
PROC main()
InitializeMain;
InitPickMT;
WHILE TRUE DO
CheckSystem;
RefPosIn_CAMERA_NO_1;
SetNextTarget CAMERA_NO_1;
PickCam_CAMERA_NO_1;
!Now prepare for second image (in this example position 2 in
     PickMT);
BELT_ACTION{CAMERA_NO_1} := RUN_NEVER;
DisablePosition CAMERA_NO_1, 1000;
EnablePosition CAMERA_NO_1, 2;
MoveRobotToPositionForSecondImage;
ConfirmPick1; !Confirms first image and takes new
WaitTime 1;
SetNextTarget CAMERA_NO_1;
WHILE nAction{CAMERA_NO_1} <= NO_DETAIL AND
nImageRetries{CAMERA_NO_1} < MAX_IMAGE_RETRIES{CAMERA_NO_1} DO
!Possible to change camera settings here before new try to find
     detail
DiscardAndTakeNewImage CAMERA_NO_1;
WaitUntil bCoordReceived{CAMERA_NO_1} = TRUE;
SetNextTarget CAMERA_NO_1;
Incr nImageRetries{CAMERA_NO_1};
ENDWHILE
IF nAction{CAMERA_NO_1} THEN
!Still no detail found
!Create alarm and EXIT or
BELT_ACTION{CAMERA_NO_1} := RUN_NO_DETAIL;
DisablePosition CAMERA_NO_1, 1000;
EnablePosition CAMERA_NO_1, 1;
RefPosOut_CAMERA_NO_1;
!Move detail to trash
ELSE
!Prepare for new image from FeedLine (in this example position 1
     in PickMT)
BELT_ACTION{CAMERA_NO_1} := RUN_NO_DETAIL;
DisablePosition CAMERA_NO_1, 1000;
EnablePosition CAMERA_NO_1, 1;
RefPosOut_CAMERA_NO_1;
Cam1Position_nPosition;
ENDIF
ENDWHILE
ENDPROC
```

3.1.2 Background program

The standard also includes several background programs that monitor and control important functions that cannot be managed adequately from the main program. The following background programs are usually used:

Continues on next page

3.1.2.1 EntryControl

This program monitors the entry request to the cell. As soon as entry is requested, a signal is sent to the main program and the robot stops as soon as it has completed its cycle.

3.1.2.2 EntryControlLamp

This program ensures that the indication lamp for the entry request lights up/flashes correctly depending on the status of the machine. The lamp starts to flash when entry is requested and stays lit when entry is permitted. When the robot runs in normal operation the lamp goes out.

3.1.2.3 PvCom

PvCom primarily manages control of feed front to camera and reception of the new co-ordinates from PickMT.

If PickMT cannot identify anything in an image, PvCom ensures that the belt starts and runs until a new detail comes to the camera. The belt then stops and a new image is taken. When PickMT has identified something, PvCom receives the co-ordinates that are sent from PickMT. Then bCoordReceived{nCamera} is set for the relevant camera so that the main program knows that there are new co-ordinates.

This page is intentionally left blank

4 PickMT

4.1 Teachin Mechanics

When an ABB robot is used, two types of robot function are visible in PickMT's teachin tab "Mechanics". It is possible to send user-defined parameters from PickMT to the robot, as well as allow PickMT to select robot program and automatically start the program when PickMT starts.

The operator can specify two user defined parameters. The robot program can retrieve these parameters if necessary. This function can, for example, be used to transfer detail specific information to a general robot program.

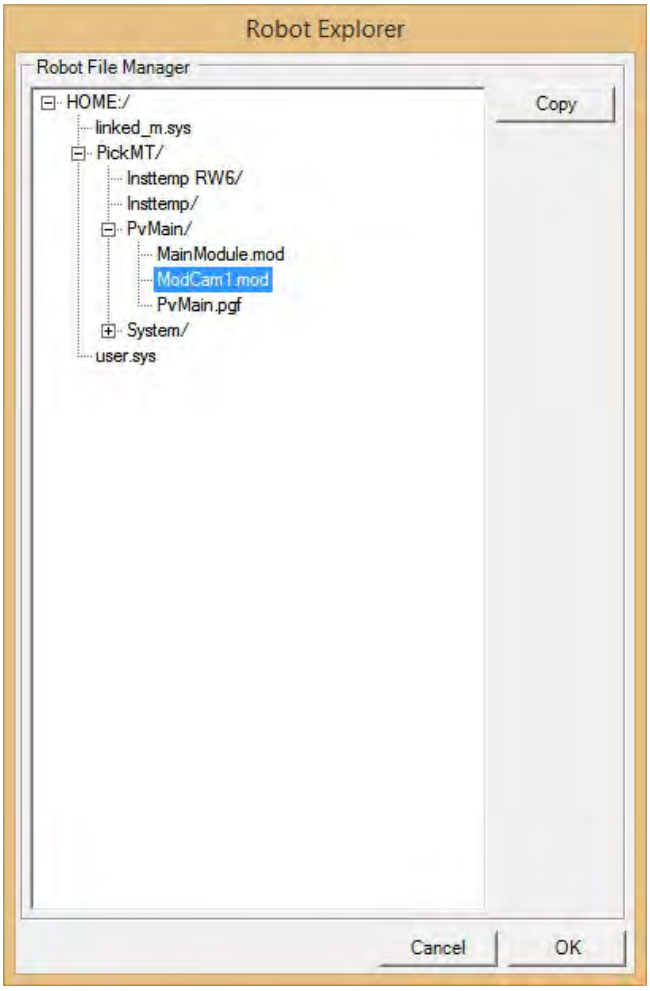
Robot program: IRC5

The following sections apply to ABB robots with IRC5 (PC-SDK) control systems.

When the system starts and stops the right program is automatically loaded by the robot. The robot can also start with the currently loaded program. If this is required you must tick the 'Run with currently loaded program' box. Normally the program should be loaded each time the system starts.

Continues on next page

The complete search path for the program must be entered in the robot program box. If you click on the ‘Browse’ button it will open a file manager window for the robot.



The robot’s HOME file system with available folders and files is shown here. Select the program to be used for the detail and click on ‘Ok’ to use this program. The search path will now be updated for the chosen program.

Program copying

Click on the ‘Copy’ button to copy a program folder. The program folder that is to be copied must be selected first.

Enter the desired name of the program folder and click on ‘OK’. The copy will be placed in the same folder as the original.

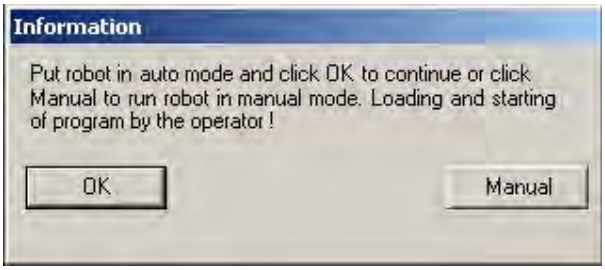
4.2 Operating from PickMT

When an ABB robot is used with PickMT, PickMT starts and stops the robot as standard. However, it can also operate without this function. Below is a function description when PickMT starts and stops the robot.

Continues on next page

4.2.1 Start

When a detail is started in PickMT, by clicking 'Start', the right program is downloaded to the robot, which then starts automatically. However, this requires that the robot is set to "Auto". If the robot is not in "Auto" mode the following dialogue box will appear.



To start the system and load the program, put the robot in "Auto" mode and then click on 'OK'. To start the system and run the robot at reduced speed, click on 'Manual'. Note that PickMT will not load the program that belongs to the detail when 'Manual' mode is selected. In this case the correct program must be loaded and started manually. Note that the program must be started from the beginning. In a Multi camera system the system must have been started at least once with robot in "Auto" mode in order to load all modules correctly.

4.2.2 Stop

To stop the system, click on ‘Stop’ in PickMT. If the robot is in ”Auto” mode, it will also be stopped. Pressing ‘Stop’ will not make the robot stop immediately; it will only stop at the end of its cycle or after a certain time. To stop the robot immediately you must use the stop button on the robot or, if an emergency situation has arisen, use the emergency stop.

If PickMT is configured so that the robot should control the stop situation, the stop button behaves differently. In this case PickMT does not actively stop the robot when the operator clicks stop. Instead, only DOF_EndCycle is sent to the robot, whereupon PickMT awaits the robot to call the procedure StopPickMT which in turn will stop everything as usual (as described above).

In case the user clicks stop in PickMT again while PickMT is awaiting the call to StopPickMT, PickMT will enforce a stop. This means that PickMT will no longer be waiting for the robot, but instead will stop as usual (as in the case where PickMT is configured to control the robot stop).

The function that the robot controls the stop situation is useful when there are special cycles that can not be stopped in the middle of operation. In this case, the robot normally knows when a stop is appropriate.

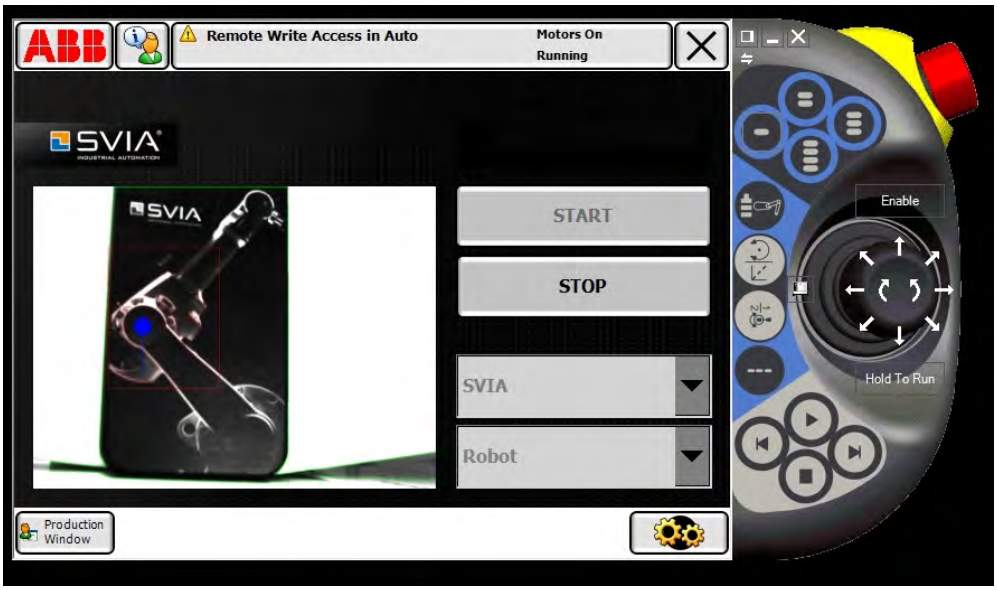
This page is intentionally left blank

5 Flexpendant Master (option)

It is possible to control PickMT via a graphic interface in the Flexpendant. For it to used, the robot option 'Flexpendant interface' is required together with IRC5 and robotware 5.14 or later.

Flexpendant Master makes it easy for the operator to manage PickMT from the FlexPendant. Only information that is required for operation of the robot cell during production appears.

FlexPendant Master can only be used when the robot is in "Auto". If the robot is in "Manual" mode, only the status display is updated.



The following function is found in the FlexPendant Master interface:

- Select group and detail (camera 1)
- Starting and stopping PickMT
- Display of latest image taken during operation
- Display of status for PickMT connection. Indicates any contact, or what PickMT is currently doing.
- Display of latest alarm from PickMT with possibility of resetting alarm (displayed at top of the window)
- Checkbox to select whether the latest camera image is displayed or not. If image display is selected, the cycle time for PickMT will be slightly longer!

FlexPendant Master is most suitable for use when there is a fixed mounted monitor with the PickMT computer. However, it is possible to use it if there is a proper monitor for PickMT as well.

In order to show Flexpendant Master on the robot, in addition to the Flexpendant interface robot option, certain files must be placed on the robot. The following files must be placed directly in the robot's HOME directory (X_YY = RobotWare version):

- TpsViewPickMTStartStopX_YY.dll
- TpsViewPickMTStartStopX_YY.gtpu.dll

Continues on next page

Continued

- PickMT.bmp
- Language.txt

Once this is done, the program can be opened by pressing the PickMT icon that is now visible in the list under the ABB menu.

6 Appendix A: Installation

6.1 Installation robot

6.1.1 ABB IRC5

Operating system and options

The robot operating system will only require re-installing in exceptional cases. If so, refer to the product manual for the robot. System requirement RobotWare 5.07 or higher The following options must be available for the robot:

- World Zones
- PC Interface
- Multitasking
- Flexpendant Interface (only if Flexpendant Master is used)

Following a base installation a recent valid backup should be used to restore the robot's settings. In special cases, the robot's settings can be restored in stages, using files from the most recent valid backup.

System program and configuration files

Make a backup of the robot before you carry out the installation!

The following files are used for installation of the system program:

PickMT\PvMain\PvMain.pgf PickMT\PvMain\MainModule.mod PickMT\System\PvMcSys.sys PickMT\System\ PvCom.mod PickMT\System\ EntryControl.mod PickMT\System\ EntryControlLamp.mod PickMT\System\ ManualModeSupervision.mod (ONLY Feedlines with manual belt control) PickMT\System\Common.sys

Insttemp\EIO.CFG Insttemp\SYS_ADD.CFG (With ManualModeSupervision.mod there is an extra SYS_ADD.CFG) Insttemp\MOC_ADD.CFG

The files must be copied to the HOME folder (named as the robot's serial number) on the robot, according to the folder structure displayed above.

EIO.CFG is the file that configures inputs and outputs in the robot. In the basic version, the signal name to PLC, to gripper and standard machine communication are configured. The physical location of the signals on the I/O block must be given. Edit the file before copying to robot if it is necessary for the relevant application.

Go to ”Control panel”, ”Configuration” and load the configuration files from HOME:/Insttemp according to the following It is important that the correct method is used to load the configuration file.

EIO.CFG must be loaded with ”File”, ”Enter parameters ...”, ”Delete existing parameters before entering” SYS_ADD.CFG and MOC_ADD.CFG must be loaded with ”File”, ”Enter parameters ...”, ”Enter parameters and replace duplicates”

Thereafter all system modules must be loaded in the robot's memory, which can be done using RobotStudio or via the Flexpendant. Using Flexpendant this is done as follows: Go to ProgramEditor. Load program ”PvMain.pgf”. You will receive an error message that different procedures and signal names are not defined,

Continues on next page

acknowledge. Load module ”PvMcSys.sys”. You will receive an error message that different procedures and signal names are not defined, acknowledge. Load the module ”Common.sys”.

After all configuration files have been loaded the robot must be restarted. Messages that the program pointer has changed are normal. No other significant faults have occurred.

6.2 Installation PickMT computer

6.2.1 Robot Communication Runtime

In order for an ABB robot to communicate with PickMT, Robot Communication Runtime must be installed on the computer.

Run the file '<CD-ROM:>\4 – Robot Communication Runtime\ABB RobotCommunicationRunTime \RCR X.YY\Setup.exe'. Then select the correct version for your robot. CD-ROM: should be changed to the device name for the CD-ROM drive in the relevant computer.

Note that a newer Robot Communication Runtime on the computer can usually communicate with an older version of RobotWare in the robot's control system.

6.2.2 PickMaster 3 - Conveyor Tracking

For operation with Conveyor Tracking, PickMaster3 is used with PickMT as external sensor. Adhere to PickMaster3 manual for detailed information on setup and configuration.

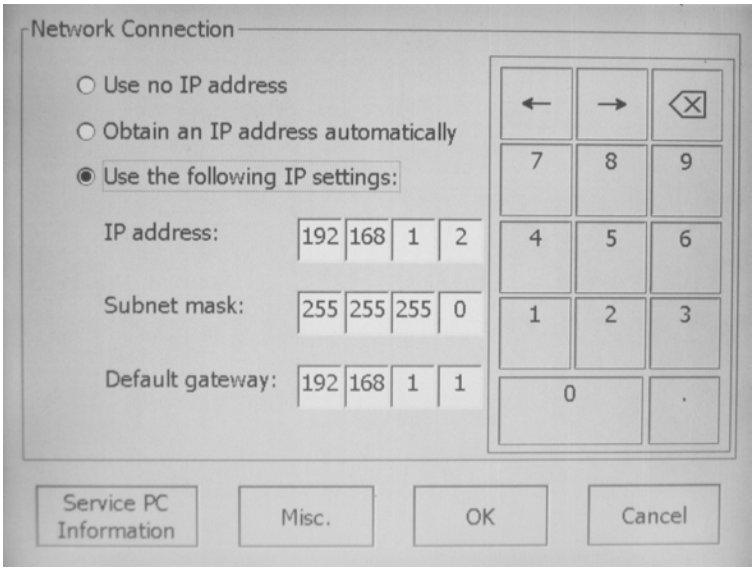
6.3 Network configuration

Before operation, an IP address must be set for the robot. Just like for other units in the installation, the robot has a fixed IP address. Following is used as standard:

IP address : 192.168.1.2
Subnet-mask: 255.255.255.0
Gateway: 192.168.1.1

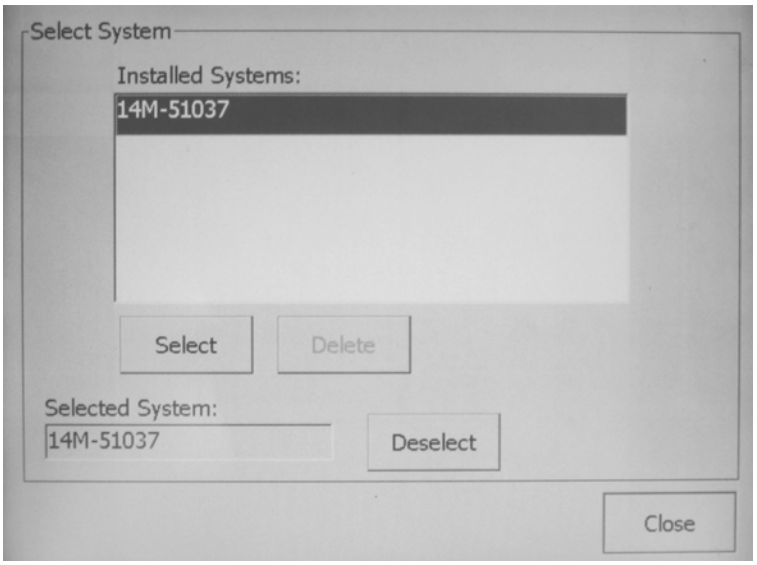
IP addresses for the robot are set via an X-Start. With X-Start the main menu in ”Boot Application” will come up.

”Settings” should be selected, then ”LAN Settings”.



Continues on next page

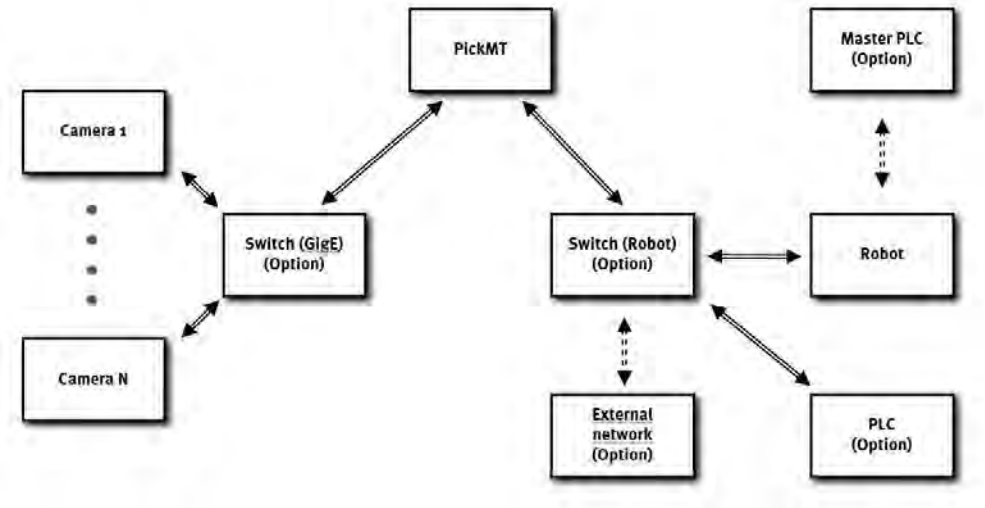
Information for each robot must be entered here. Press the ‘OK’ button to return to main menu. Choose ”Select system”.



Mark the relevant system, press ”Select”, ”Close”, and then ”Restart Controller” in the main menu.

7 Appendix B: Communication

The following image shows the basic communication between PickMT and surrounding equipment.



Continues on next page

7.1 Communication between PickMT – Robot

Communication between PickMT and ABB robot is via PC SDK. Information is transmitted between robot and PickMT through direct writing to variables and I/O signals.

See robot files for more information.

Continues on next page

7.1.1 Operation in a Standard system

PickMT		Robot	Regards	Formatting
		bGrab, bGrab1..4	Take new image, with camera 1..4	direct writing
X			X-value	direct writing
Y			Y-value	direct writing
Z			Z-value	direct writing
PZ			Z-value over picking position	direct writing
RX			Rotation around X-shaft	direct writing
RY			Rotation around Y-shaft	direct writing
RZ			Rotation around Z-shaft	direct writing
Position			Detail position	direct writing
Action			Volume information	direct writing
nAmountOfDetails			Detail quantity (option)	direct writing
nAmountOfDetails1			Detail amount 1 (option)	direct writing
nAmountOfDetails2			Detail amount 2 (option)	direct writing
nAmountOfDetails3			Detail amount 3 (option)	direct writing
DOF_Coord			Set to coordinates printing	direct writing
		bCoord	Coordinator received	direct writing
Cam			Camera number	direct writing

Volume information contains information about how many details that can be picked are in the image field: No details (Action=1), precisely one detail (Action=2), or more than one detail (Action=4).

When the option for multi coordinate gripping or PalletPicker2D is used, the following volume information is also available: More than one detail, but no new images must be taken after picking (Action=256), replace pallet (Action=8), pick interlayers (Action=16), pick interlayers and replace pallet immediately afterwards (action=32).

7.1.2 Other commands

At each start, a request to empty coordinates is transmitted to ensure that old coordinates are not transmitted. This should be used in all machine types.

PickMT		Robot	Regards	Formatting
		bClearSend	Relieve Coordinate queue.	direct writing
DOF_EndCycle			Stop program at suitable point	direct writing
		bStop	Stop PickMT	direct writing
		bPVStatus	Request for operating status PickMT	direct writing
nPVStatus			PV_IDLE=1, PV_OPERATION=2, PV_STARTING=4, PV_STOPPING=8	direct writing
		bConfirmPick	Confirm pick	direct writing
		bConfPickCam1 .. 4	Confirm pick for camera 1,2,3 or 4	direct writing
		bConfIntPick	Confirm pick of interlayers	direct writing
		bConfIntPickCam1 .. 4	Confirm pick of interlayers for camera 1,2,3 or 4	direct writing
		bFreeParam1, bFreeParam2	Request for user parameter 1 or 2	direct writing
		bFreeParam1Cam1 .. 4 bFreeParam2Cam1 .. 4	Request for user parameter 1 or 2 for camera 1,2,3 or 4	direct writing
sFreeParameter			the value of the parameter	direct writing
		sMessageBoxOk, sMessageBoxYesNo	Show message on PickMT interface	direct writing
sMsgResponse			RESPONSE_OK, RESPONSE_YES, RESPONSE_NO	direct writing
		sMsgAnswer, sInputBox	Show message (sInputBox) on PickMT interface and retrieve input from operator (sMsgAnswer as suggested answer).	direct writing
sMsgResponse			Text entered by operator	direct writing
		nDisablePos	Disables selected position. Disables all positions if the value is > 9999	direct writing

Continues on next page

		nEnablePos	Enables selected position. Enables all positions if the value is > 9999	direct writing
		nDisablePosCam1 .. 4	Disables selected position for camera 1,2,3 or 4. Disables all positions if the value is > 9999	direct writing
		nEnablePosCam1 .. 4	Enables selected position for camera 1,2,3 or 4. Enables all positions if the value is > 9999	direct writing
		sBlackRegion	Colours parts of the image field black. Format 'A-BBB-CCC-DDD-EEE-F'. A camera, BBB StartX (%), CCC StopX (%), DDD StartY (%), EEE StopY (%), F black on inside (1) or outside (0)	direct writing
sMsgResponse			Text entered by operator	direct writing
		nSlaveAutoStart	Starts PickMT.	direct writing
		nSlaveCycleStop	Stops PickMT.	direct writing
		nSlaveSelectId	Replaces detail in PickMT from selected ID number, i.e. value of nSlaveSelectId.	direct writing
		nSlaveSelectId-Cam1..4	Replaces detail in PickMT for stated camera from selected ID number, i.e. value of nSlaveSelectId-Cam1..4.	direct writing
		nExposureTime	Sets exposure time for actual camera within the area 0%-100%.	direct writing
		nExposureTimeCam1 .. 4	Sets exposure time for camera 1..4 within the area 0%-100%.	direct writing
		nContrast	Sets contrast for actual camera within the area 0%-100%.	direct writing
		nContrastCam1..4	Sets contrast for camera 1..4 within the area 0%-100%.	direct writing
		nGain	Sets gain for actual camera within the area 0%-100%.	direct writing
		nGainCam1..4	Sets gain for camera 1..4 within the area 0%-100%.	direct writing

Continues on next page

7.1.2 Other commands
Continued

The following status signals are used in communication. The signals are digital outputs and must be defined in the robot.

DOF_Coord: Used to signal to the robot program that new coordinates are available.

DOF_MotSupTrigg Cross connected to system output 'MotSupTrigg'. Used so that PickMT can receive the information about collision.

DOF_Error Cross connected to system output 'Error'. Used so that PickMT can receive the information about fault incidents.

DOF_EndCycle Used so that PickMT can stop the robot when a correct cycle is completed and not in the middle of a cycle.

DOF_CycleEnded Response signal from robot on DOF_EndCycle.

DOF_CycleOn Cross connected to system output 'CycleOn'. This signal is checked before the coordinates are sent. If the robot is not running, an alarm is left and PickMT waits until the robot program starts again before coordinates are sent.

DOF_AutoOn Cross connected to system output AutoOn'. This signal is checked upon startup and stop through PickMT.

DOF_ResetEStop Cross connected to system input 'ResetEStop'. Used so that PickMT can automatically start the robot motors after an emergency stop.

DOF_ MotorsOn Cross connected to system input 'MotorOn'. Used so that PickMT at the selected option can start the robot via system inputs.

DOF_ StartAtMain Cross connected to system input 'MotorOn'. Used so that PickMT at the selected option can start the robot via system inputs.

DOF_ StopEndInstr Cross connected to system input 'MotorOn'. Used so that PickMT at the selected option can stop the robot via system inputs.

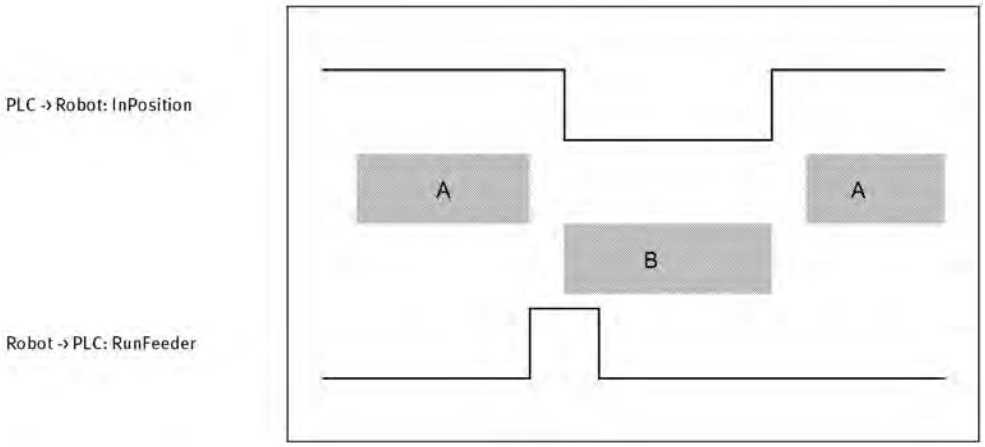
DOF_ MotOnStart Cross connected to system input 'MotorOn'. Used so that PickMT at the selected option can start the robot via system inputs.

DOF_RunFeeder1...4 Cross connected to 'RunFeeder1...4'. Used by PvCom to run supply equipment under camera1-4.

DOF_InPosition1...4 Cross connected to 'Feeder1(...4)InPosition'. Used by PvCom to check when a new detail has reached camera1-4.

7.2 Communication Robot – PLC

Communication between Robot and PLC is via digital signals.



A: Taking images and picking, B: Conveyor is running

7.3 Communication Master-Slave

It is possible to allow an external PLC to be master for PickMT via the robot. This means that the external PLC can stop, start and change detail in PickMT. The robot acts as an intermediary to get it to work. The following signals must occur between the robot and the PLC.

- AutomaticStart, CycleStop, SelectId
- AutomaticStrtOk, AutomaticStrtNok, CycleStopOk, CycleStopNok, SelectIdOk, SelectIdNok
- IdArticle (Group input 16 bits)

Note that both background modules for the slave function through the robot and program module must be called PVSLAVE for PickMT to communicate correctly. PVSLAVE is available as a ready standard module.

Read the PickMT manual regarding communication Master-Slave for further information on how the signals above are to be sent to achieve the desired function.

8 Appendix C: Own notes

Contact us

ABB AB
Discrete Automation and Motion
Robotics
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS, Robotics
Discrete Automation and Motion
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 51489000

ABB Engineering (Shanghai) Ltd.
No. 4528 Kangxin Hingway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

www.abb.com/robotics

3HAC051770-001, Rev A, en