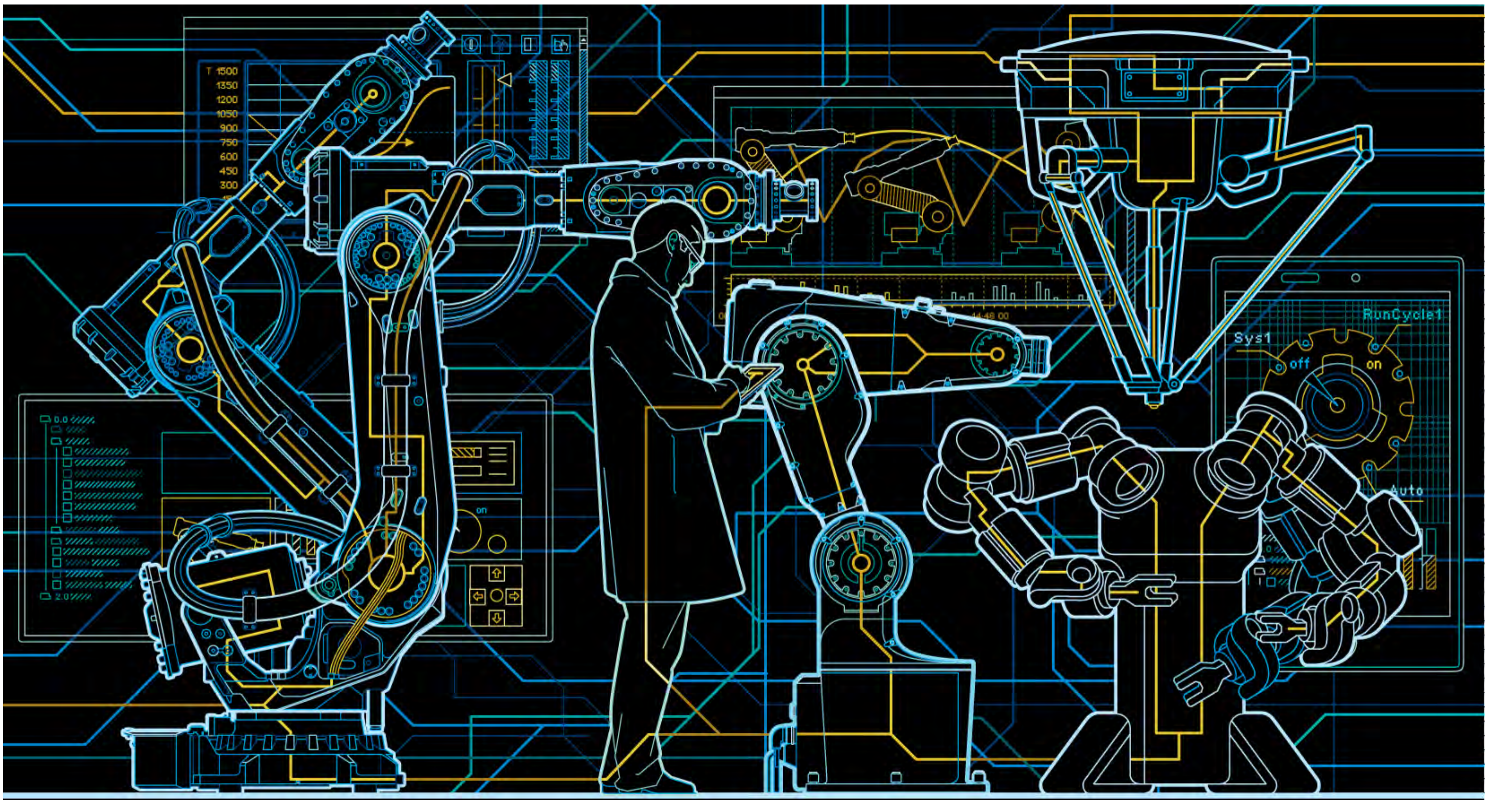


Application manual
BullsEye

Power and productivity
for a better world™

Trace back information:
Workspace R16-2 version a5
Checked in 2016-09-08
Skribenta version 4.6.318

Application manual

BullsEye

RobotWare 6.04

Document ID: 3HAC050989-001

Revision: A

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 explains the basics of when and how to use the option BullsEye®.

- Product overview
- Operation overview
- Requirements overview
- Software set-up
- Software reference, RAPID

Usage

This manual can be used either as a reference to find out if an option is the right choice for solving a problem, or as a description of how to use an option. Detailed information regarding syntax for RAPID routines, and similar, is not described here, but can be found in the respective reference manual.

Who should read this manual?

This manual is intended for:

- installation personnel
- maintenance personnel
- repair personnel.
- robot programmers

Prerequisites

Maintenance/repair/installation personnel working with an ABB Robot must:

- be trained by ABB and have the required knowledge of mechanical and electrical installation/repair/maintenance work.
- be familiar with industrial robots and their terminology
- be familiar with the RAPID programming language
- be familiar with system parameters and how to configure them.

Reference documents

References	Document ID
Operating manual - General safety information [i]	*3HAC031045-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC050917-001*
Technical reference manual - RAPID overview	*3HAC050947-001*
Operating manual - IRC5 with FlexPendant	*3HAC050941-001*
Technical reference manual - System parameters	*3HAC050948-001*
Operating manual - RobotStudio	*3HAC032104-001*
Application manual - Production Manager	*3HAC052855-001*

[i] This manual contains all safety instructions from the product manuals for the manipulators and the controllers.

Continues on next page

Revisions

Revision	Description
-	Released with RobotWare 6.0.
A	Released with RobotWare 6.04. • BullsEye is now a separate RobotWare option.

Product documentation, IRC5

Categories for user documentation from ABB Robotics

The user documentation from ABB Robotics is divided into a number of categories. This listing is based on the type of information in the documents, regardless of whether the products are standard or optional.

All documents listed can be ordered from ABB on a DVD. The documents listed are valid for IRC5 robot systems.

Product manuals

Manipulators, controllers, DressPack/SpotPack, and most other hardware is delivered with a **Product manual** that generally contains:

- Safety information.
- Installation and commissioning (descriptions of mechanical installation or electrical connections).
- Maintenance (descriptions of all required preventive maintenance procedures including intervals and expected life time of parts).
- Repair (descriptions of all recommended repair procedures including spare parts).
- Calibration.
- Decommissioning.
- Reference information (safety standards, unit conversions, screw joints, lists of tools).
- Spare parts list with exploded views (or references to separate spare parts lists).
- Circuit diagrams (or references to circuit diagrams).

Technical reference manuals

The technical reference manuals describe reference information for robotics products.

- *Technical reference manual - Lubrication in gearboxes*: Description of types and volumes of lubrication for the manipulator gearboxes.
- *Technical reference manual - RAPID overview*: An overview of the RAPID programming language.
- *Technical reference manual - RAPID Instructions, Functions and Data types*: Description and syntax for all RAPID instructions, functions, and data types.
- *Technical reference manual - RAPID kernel*: A formal description of the RAPID programming language.
- *Technical reference manual - System parameters*: Description of system parameters and configuration workflows.

Continues on next page

Application manuals

Specific applications (for example software or hardware options) are described in **Application manuals**. An application manual can describe one or several applications.

An application manual generally contains information about:

- The purpose of the application (what it does and when it is useful).
- What is included (for example cables, I/O boards, RAPID instructions, system parameters, DVD with PC software).
- How to install included or required hardware.
- How to use the application.
- Examples of how to use the application.

Operating manuals

The operating manuals describe hands-on handling of the products. The manuals are aimed at those having first-hand operational contact with the product, that is production cell operators, programmers, and trouble shooters.

The group of manuals includes (among others):

- *Operating manual - Emergency safety information*
- *Operating manual - General safety information*
- *Operating manual - Getting started, IRC5 and RobotStudio*
- *Operating manual - IRC5 Integrator's guide*
- *Operating manual - IRC5 with FlexPendant*
- *Operating manual - RobotStudio*
- *Operating manual - Trouble shooting IRC5*

1 Safety

1.1 Safety signals in the manual

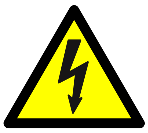
Introduction to safety signals

This section specifies all dangers that can arise when doing the work described in the user manuals. Each danger consists of:

- A caption specifying the danger level (DANGER, WARNING, or CAUTION) and the type of danger.
- A brief description of what will happen if the operator/service personnel **do not** eliminate the danger.
- Instruction about how to eliminate danger to simplify doing the work.

Danger levels

The table below defines the captions specifying the danger levels used throughout this manual.

Symbol	Designation	Significance
 xx0200000022	DANGER	Warns that an accident *will* occur if the instructions are not followed, resulting in a serious or fatal injury and/or severe damage to the product. It applies to warnings that apply to danger with, for example, contact with high voltage electrical units, explosion or fire risk, risk of poisonous gases, risk of crushing, impact, fall from height, and so on.
 xx0100000002	WARNING	Warns that an accident *may* occur if the instructions are not followed that can lead to serious injury, possibly fatal, and/or great damage to the product. It applies to warnings that apply to danger with, for example, contact with high voltage electrical units, explosion or fire risk, risk of poisonous gases, risk of crushing, impact, fall from height, etc.
 xx0200000024	ELECTRICAL SHOCK	Warns for electrical hazards which could result in severe personal injury or death.
 xx0100000003	CAUTION	Warns that an accident may occur if the instructions are not followed that can result in injury and/or damage to the product. It also applies to warnings of risks that include burns, eye injury, skin injury, hearing damage, crushing or slipping, tripping, impact, fall from height, etc. Furthermore, it applies to warnings that include function requirements when fitting and removing equipment where there is a risk of damaging the product or causing a breakdown.
 xx0200000023	ELECTROSTATIC DISCHARGE (ESD)	Warns for electrostatic hazards which could result in severe damage to the product.

Continues on next page

Symbol	Designation	Significance
 xx0100000004	NOTE	Describes important facts and conditions.
 xx0100000098	TIP	Describes where to find additional information or how to do an operation in an easier way.

1.2 Safety risks during installation and service work on robots

Overview

This section includes information on general safety risks to be considered when performing installation and service work on the robot.

These safety instructions have to be read and followed by any person who deals with the installation and maintenance of the robot. Only persons who know the robot and are trained in the operation and handling of the robot are allowed to maintain the robot. Persons who are under the influence of alcohol, drugs or any other intoxicating substances are not allowed to maintain, repair, or use the robot.

General risks during installation and service

- The instructions in the product manual in the chapters *Installation and commissioning*, and *Repair* must always be followed.
- Emergency stop buttons must be positioned in easily accessible places so that the robot can be stopped quickly.
- Those in charge of operations must make sure that safety instructions are available for the installation in question.
- Those who install or service/maintain the robot must have the appropriate training for the equipment in question and in any safety matters associated with it.

Spare parts and special equipment

ABB does not supply spare parts and special equipment which have not been tested and approved by ABB. The installation and/or use of such products could negatively affect the structural properties of the robot and as a result of that affect the active or passive safety operation. ABB is not liable for damages caused by the use of non-original spare parts and special equipment. ABB is not liable for damages or injuries caused by unauthorized modifications to the robot system.

Personal protective equipment

Always use suitable personal protective equipment, based on the risk assessment for the robot installation.

Nation/region specific regulations

To prevent injuries and damages during the installation of the robot, the regulations applicable in the country concerned and the instructions of ABB Robotics must be complied with.

Non-voltage related risks

- Make sure that no one else can turn on the power to the controller and robot while you are working with the system. A good method is to always lock the main switch on the controller cabinet with a safety lock.
- Safety zones, which must be crossed before admittance, must be set up in front of the robot's working space. Light beams or sensitive mats are suitable devices.

Continues on next page

- Turntables or the like should be used to keep the operator out of the robot's working space.
- If the robot is installed at a height, hanging, or other than standing directly on the floor, there may be additional risks than those for a robot standing directly on the floor.
- The axes are affected by the force of gravity when the brakes are released. In addition to the risk of being hit by moving robot parts, there is a risk of being crushed by the parallel arm (if there is one).
- Energy stored in the robot for the purpose of counterbalancing certain axes may be released if the robot, or parts thereof, are dismantled.
- When dismantling/assembling mechanical units, watch out for falling objects.
- Be aware of stored heat energy in the controller.
- Never use the robot as a ladder, which means, do not climb on the robot motors or other parts during service work. There is a serious risk of slipping because of the high temperature of the motors and oil spills that can occur on the robot.
- Never use the robot as a ladder, which means, do not climb on the manipulator motors or other parts during service work. There is a risk of the robot being damaged.

To be observed by the supplier of the complete system

When integrating the robot with external devices and machines:

- The supplier of the complete system must ensure that all circuits used in the safety function are interlocked in accordance with the applicable standards for that function.
- The supplier of the complete system must ensure that all circuits used in the emergency stop function are interlocked in a safe manner, in accordance with the applicable standards for the emergency stop function.

Complete robot

Safety risk	Description
Hot components!	**CAUTION** Motors and gearboxes are HOT after running the robot! Touching motors and gearboxes may result in burns! With a higher environment temperature, more surfaces on the manipulator will get HOT and may also result in burns.
Removed parts may result in collapse of the robot!	**WARNING** Take any necessary measures to ensure that the robot does not collapse as parts are removed. For example, secure the lower arm according to the repair instruction if removing the axis-2 motor.

Continues on next page

Safety risk	Description
Removed cables to the measurement system	**WARNING** If the internal cables for the measurement system have been disconnected during repair or maintenance, then the revolution counters must be updated.

Cabling

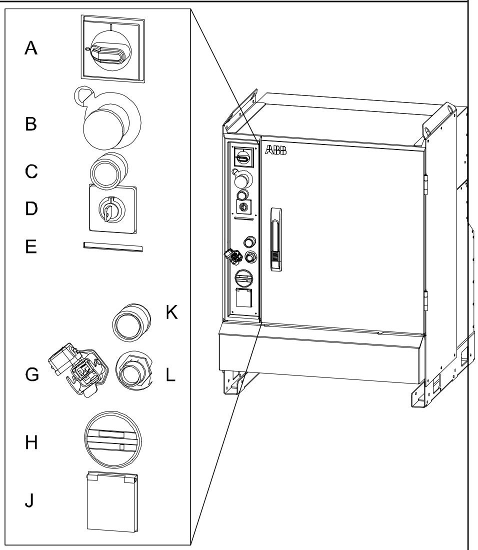
Safety risk	Description
Cable packages are sensitive to mechanical damage!	**CAUTION** The cable packages are sensitive to mechanical damage. Handle the cable packages and the connectors with care in order to avoid damage.

1.3 DANGER - Make sure that the main power has been switched off!

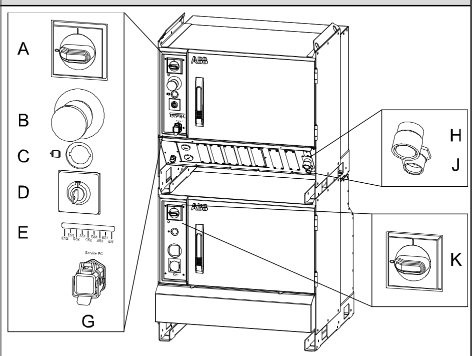
Description

Working with high voltage is potentially lethal. Persons subjected to high voltage may suffer cardiac arrest, burn injuries, or other severe injuries. To avoid these dangers, do not proceed working before eliminating the danger as detailed below.

Elimination, Single Cabinet Controller

	Action	Note/illustration
1	Switch off the main switch on the controller cabinet.	 xx0600002782 A: Main switch

Elimination, Dual Cabinet Controller

	Action	Note/illustration
1	Switch off the main switch on the Drive Module.	 xx0600002783 K: Main switch, Drive Module
2	Switch off the main switch on the Control Module.	A: Main switch, Control Module

1.4 Risks associated with live electric parts

Voltage related risks, general

Work on the electrical equipment of the robot must be performed by a qualified electrician in accordance with electrical regulations.

- Although troubleshooting may, on occasion, need to be carried out while the power supply is turned on, the robot must be turned off (by setting the main switch to OFF) when repairing faults, disconnecting electric leads and disconnecting or connecting units.
- The main supply to the robot must be connected in such a way that it can be turned off from outside the working space of the robot.
- Make sure that no one else can turn on the power to the controller and robot while you are working with the system. A good method is to always lock the main switch on the controller cabinet with a safety lock.

The necessary protection for the electrical equipment and robot system during construction, commissioning, and maintenance is guaranteed if the valid regulations are followed.

All work must be performed:

- by qualified personnel
- on machine/robot system in deadlock
- in an isolated state, disconnected from power supply, and protected against reconnection.

Voltage related risks, IRC5 controller

A danger of high voltage is associated with, for example, the following parts:

- Be aware of stored electrical energy (DC link, Ultracapacitor bank unit) in the controller.
- Units such as I/O modules, can be supplied with power from an external source.
- The main supply/main switch
- The transformers
- The power unit
- The control power supply (230 VAC)
- The rectifier unit (262/400-480 VAC and 400/700 VDC. Note: capacitors!)
- The drive unit (400/700 VDC)
- The drive system power supply (230 VAC)
- The service outlets (115/230 VAC)
- The customer power supply (230 VAC)
- The power supply unit for additional tools, or special power supply units for the machining process.
- The external voltage connected to the controller remains live even when the robot is disconnected from the mains.
- Additional connections.

Continues on next page

Voltage related risks, robot

A danger of high voltage is associated with the robot in:

- The power supply for the motors (up to 800 VDC).
- The user connections for tools or other parts of the installation (max. 230 VAC).

Voltage related risks, tools, material handling devices, etc.

Tools, material handling devices, etc., may be live even if the robot system is in the OFF position. Power supply cables which are in motion during the working process may be damaged.

1.5 WARNING - The unit is sensitive to ESD!

Description

ESD (electrostatic discharge) is the transfer of electrical static charge between two bodies at different potentials, either through direct contact or through an induced electrical field. When handling parts or their containers, personnel not grounded may potentially transfer high static charges. This discharge may destroy sensitive electronics.

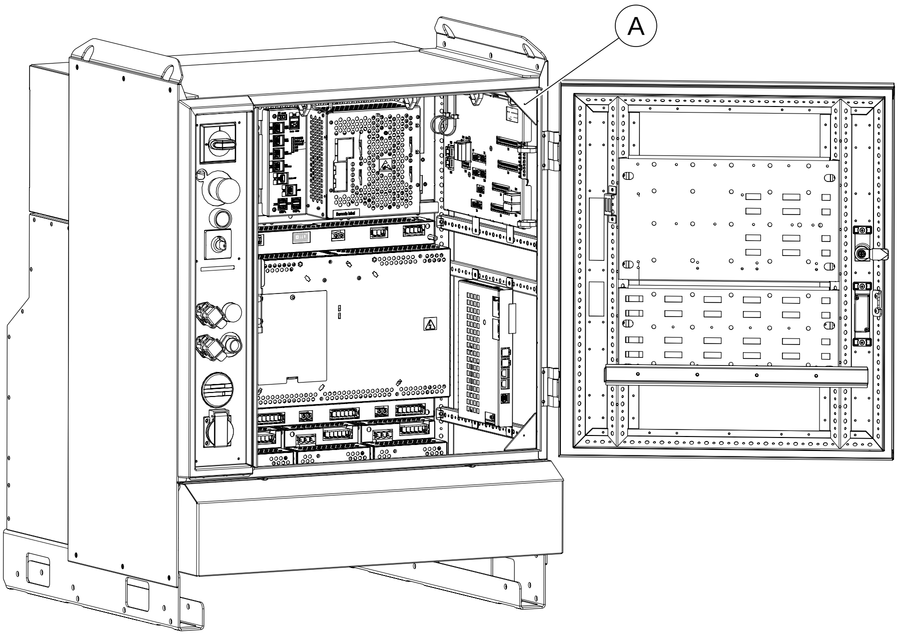
Elimination

	Action	Note
1	Use a wrist strap.	Wrist straps must be tested frequently to ensure that they are not damaged and are operating correctly.
2	Use an ESD protective floor mat.	The mat must be grounded through a current-limiting resistor.
3	Use a dissipative table mat.	The mat should provide a controlled discharge of static voltages and must be grounded.

Location of wrist strap button

The location of the wrist strap button is shown in the following illustration.

IRC5



xx1300000856

A	Wrist strap button

This page is intentionally left blank

2 Introduction to BullsEye®

2.1 Product overview

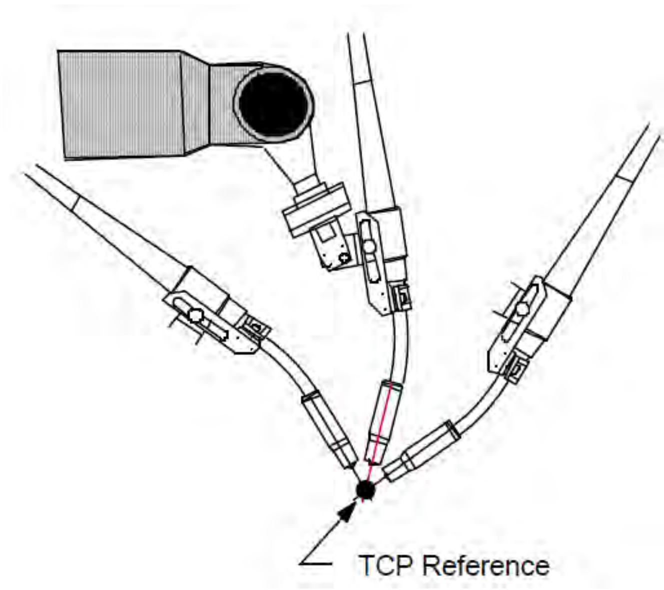
Introduction to BullsEye

BullsEye® 10 provides completely automated Tool Center Point (TCP) definition for the IRC5 robot controller and introduces support of new tools in addition to MIG welding torch configurations. Concentric cutting tools may also be used where the stick-out is defined as the distance from the cutting tip to the part surface.

TCP

TCP is defined as an invisible reference point in direct alignment and relationship to all axes of the robot arm and located at the precise point where the welding wire tip would touch the work piece using a pre-determined wire stick-out distance from the bottom of the gas nozzle.

Illustration: Welding torch revolving around a defined TCP



xx1400001210

BullsEye features

- Scanning behavior that can be configured for:
  - Scan lengths
  - Scan speeds
  - Tool dimensions
- Historical log file.
- Fully compatible with MultiMove systems.
- Accommodates RobotStudio.
- Simultaneous support for up to five unique tools per robot task.
- Integrated error handling.

Continues on next page

- Optimized update times.

2.2 Theory of operation

Example of operation

When the robot is programmed to revolve around the TCP all robot axes will move accordingly to keep the TCP stationary (see the following figures). If the torch is damaged and the program is run again, the robot repeats the same movements, but the TCP will no longer follow the same path due to the misalignment. You now have two choices:

1. Physically move the torch back into alignment (a task that could be difficult if not impossible) or
2. Adjust for the misalignment automatically by redefining the TCP to the new torch position using the BullsEye. After the BullsEye system updates the current TCP definition, the torch will rotate around the TCP as before because the robot arm has adjusted its path to compensate for the torch misalignment.

Once a point has been programmed, the robot remembers the tool center point location, not what the angles of the robot joints are. When the robot replays the programmed path, it calculates what the joint angles should be to get the TCP back to where it was when the path was programmed initially. As long as the robot controller is kept informed about where the tool center point is, it will always keep the paths properly adjusted.

Robot arm and torch movement with correct TCP

xx1400001211

Continues on next page

Robot arm follows same path but torch path has changed

xx1400001212

2.3 Limitations

System complexity

At the time of this printing, version 10.0, build 2, is the released build. It has not been tested in implementations that incorporate complex multi-axis robot carriers. For this reason, version 10 will not be supported on these applications until further notice.

Limitations for calibration

BullsEye 10 can be used to calibrate tools of a variety of shapes. While earlier versions of BullsEye were restricted to welding MIG tool designs, BullsEye 10 is also suited to cutting tools that do not have a consumable wire electrode like a MIG tool.

Here is a list of limitations:

1. The tool must be concentric along its centerline. Cylindrical and conical tools meet this criterion.
2. There may not be any obstructions on the scanned portion of the tool. Typically, the BullsEye is set up to make scans along the last several inches of the tool body. There can be no fittings, clamps, set screws, wires, hoses, or other features extending from the tool body in this section.
3. If the tool does not have a consumable wire electrode, or a wire-like extension, it must be assumed that the TCP will be inline with the centerline of the tool body.
4. The tool must have adequate clearance to allow the program to complete all moves without colliding with the BullsEye scanning device.

Typical tool designs

Here are some typical tool designs suited to BullsEye®:

Welding MIG tool

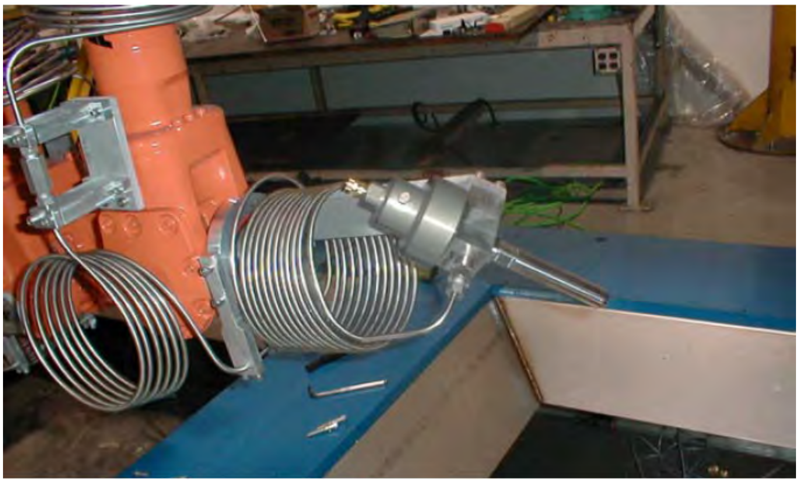
xx1400001214

Continues on next page

Hypothetical laser cutting tool

xx1400001215

Water-jet cutting tool



xx1400001216

TCP z-axis inline with mounting surface z-axis not supported

BullsEye is incapable of defining a tool that has the TCP centered along the z-axis of the robot 6th axis mounting surface, and the z-axis of the tool perpendicular to the mounting surface. Said another way, you cannot have the tool pointing straight out from the center of the mounting plate.

BE_Data.sys is a reserved module name

BullsEye uses a temporary system module called `BE_Data` to store and recover setup information. For this reason, it is not permitted to have another module loaded in the robot motion task called `BE_Data`, or BullsEye will be unable to save and retrieve data.

2.4 Safety information

WARNING

The power supply must always be switched off whenever work is carried out in the control cabinet.

WARNING

Even though the power is switched off at the robot controller, there may be energized cables connected to external equipment that are consequently not affected by the mains switch on the controller.

ELECTROSTATIC DISCHARGE (ESD)

ESD (electrostatic discharge) is the transfer of electrical static charge between two bodies at different potentials, either through direct contact or through an induced electrical field. When handling parts or their containers, personnel not grounded may potentially transfer high static charges. This discharge may destroy sensitive electronics.

	Action	Note
1	Use a wrist strap	Wrist straps must be tested frequently to ensure that they are not damaged and are operating correctly.
2	Use an ESD protective floor mat.	The mat must be grounded through a current-limiting resistor.
3	Use a dissipative table mat.	The mat should provide a controlled discharge of static voltages and must be grounded.

WARNING

Before beginning work with the robot, make sure you are familiar with the safety regulations described in the manual *Operating manual - General safety information*.

This page is intentionally left blank

3 Installation

Component list

BullsEye consists of the following components:

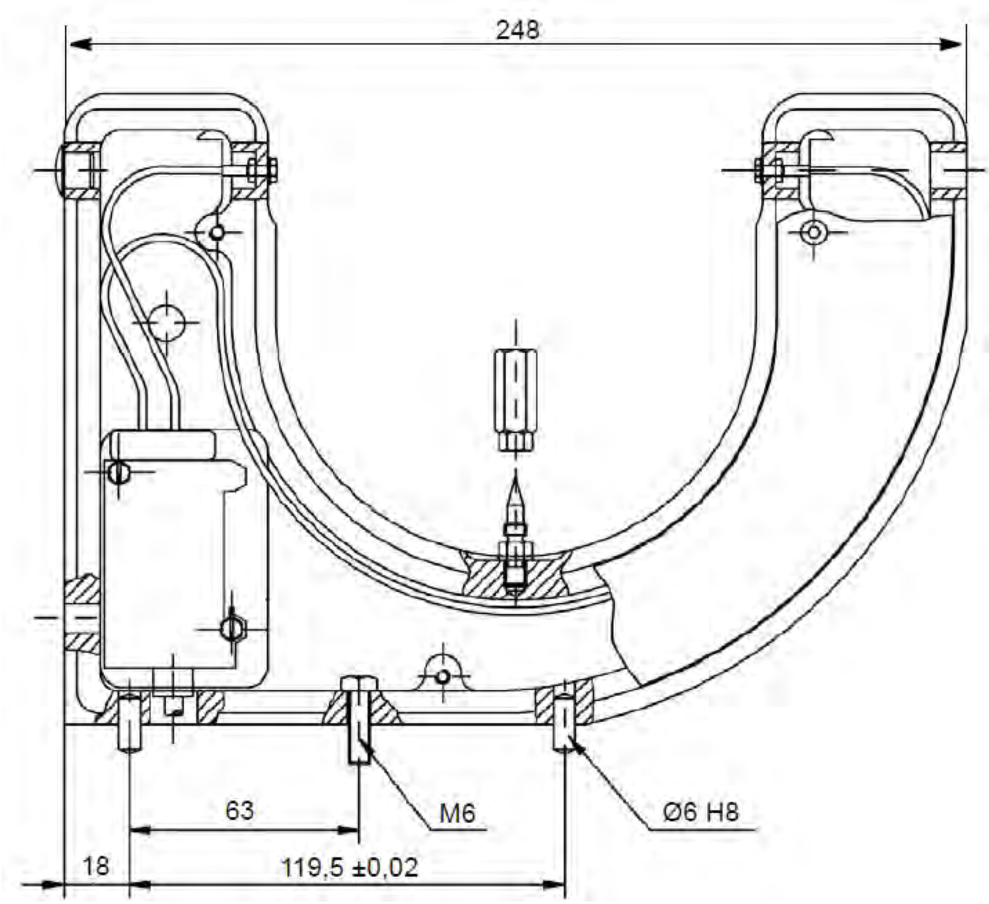
- BullsEye application manual (this manual). The manual is distributed in electronic format.
- BullsEye scanning device. Typically this will be the standard BullsEye yoke described below.
- BullsEye robot software. Software can be delivered as a separate product, or as part of cell management software like GAP and EasyArc.

BullsEye yoke specification

Electrical	40 ma, 24 VDC
Robot connections	One digital input, 24 VDC, and 0 VDC
Repeatability	± 0.006" (0.163 mm)

Dimensions

Variant 0503060880:



xx1400002302

Continues on next page

Requirements for placing the scanning device

The BullsEye scanning device must be placed in a location that allows the robot to move freely about the TCP without reaching its joint limits and without causing undesirable cable tension.

xx1400002303

Orientation of the scanning device

Although BullsEye can be configured to handle any scan device orientation, it is easiest to setup BullsEye when the beam of the scanning device is in a plane parallel to the plane of the robot base.

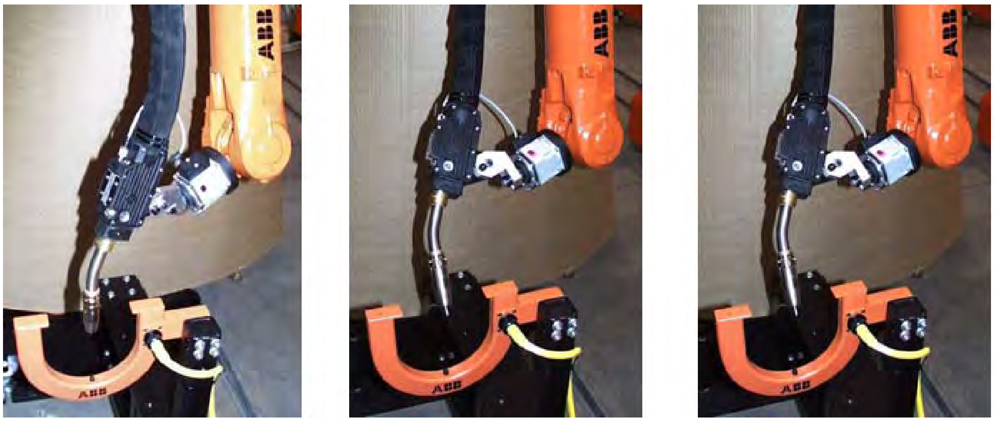
Requirements for placing the BullsEye

The BullsEye should be bolted securely in a position where the robot can reach it and where it is not in the way of personnel working around the robot.

Continues on next page

Illustration: Alignment angle

An alignment angle of 45° works best.



xx1400002304

Installing the BullsEye

1. Place the BullsEye in a desired position without securing it permanently.
2. Load the software, see *Loading BullsEye software on page 31*.
3. Complete the electrical installation, see *Electrical installation on page 31*.
4. Do the start-up test, see *Start-up test on page 31*.
5. Tighten the bolts holding the the BullsEye in position.

Loading BullsEye software

BullsEye software is loaded by selecting the BullsEye option in Installation Manager. The BullsEye option is available for the robot controller only if the BullsEye option is purchased.

If BullsEye is installed in a system with the *Arc* option, it will only be installed on the robots that installs *Arc*. If installed in a system without the *Arc* option it will be installed in all robots.

Electrical installation

The BullsEye is pre-wired at the factory for easy assembly. Connect the cable provided from the robot controller to the connector on the BullsEye unit.

The installation of the BullsEye is described in *Circuit diagram - Process Options Torch Equipment*, *3HEA802382-001*.

When the BullsEye is correctly wired, the LED on the I/O board corresponding to the input should be illuminated only when the beam is broken.

Start-up test

Do a start-up test before running BullsEye.

	Action
1	Make sure that the digital input connected to the scanning device is responding correctly, by verifying that the signal is defined as an input on an I/O board.

Continues on next page

	Action
2	Pass your hand through the BullsEye yoke beam to break the beam. The LED on the I/O board corresponding to the input should turn on when the beam is broken. If it does not, verify that the I/O board is configured properly and that the wiring is correct.

4 Maintenance

Overview

The BullsEye is shipped complete and requires very little maintenance aside from keeping the unit clean. For wiring information, see *Electrical installation on page 31*.

This page is intentionally left blank

5 User guide

WARNING

Failure to follow safety guidelines presented throughout this manual can result in property damage, serious injury, or death!

WARNING

The power supply must always be switched off whenever work is carried out in the control cabinet.

WARNING

Even though the power is switched off at the robot controller, there may be energized cables connected to external equipment and are consequently not affected by the mains switch on the controller.

Continues on next page

5.1 Overview

Initialization and define a tool

The first step in using BullsEye® is to define a tool. This is done using the `BESetupToolJ` instruction. This instruction adds a `tooldata` instance to the BullsEye collection of tools, defines the starting position, and lets BullsEye know how it should behave when other global methods are called. This information is passed to the instruction through several required and optional arguments.

```
BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scanBullsMig,
     devYokeUp, v100, fine, tWeldGun;
```

QuickCheck

To evaluate the TCP, use the QuickCheck functionality:

```
BECheckTcp tTestTemp\XYZOnly;
```

If the QuickCheck fails, a more involved search pattern will automatically be made. If successful, the tool may be updated. The optional argument `XYZOnly` indicates that the orientation of the tool should not be checked or updated. Using this will greatly decrease the time it takes to update the tool.

Update TCP (optional)

The instruction `BEUpdateTcp` will run a full scan sequence and update the tool regardless of how far off it is. This routine is generally used for evaluation purposes only.

5.2 Data storage

Storage

The data is stored in a text file on the robot controller. The format of the file represents a RAPID module allowing BullsEye to read the data into the controller when it needs to access the saved data.

The file is stored in the following directory, with a name like, `$HOME\BullsEye®\BE_Data_T_ROB1.sys`, where `T_ROB1` is the name of the task. Each robot task that is using BullsEye will have its own data file. The directory path may not be changed.

Automatic save

The data file is automatically saved after each BullsEye update action. It is automatically read before each BullsEye check action. If the file is missing, BullsEye assumes that no saved data is available and will force the user to execute a BullsEye setup routine.

Backup

The data file will be included in the backup when a system backup is ordered. A system restored from a backup will retain the stored data.

WARNING

BullsEye uses a temporary system module called `BE_Data` to store and recover setup information. For this reason, it is not permitted to have another module loaded in the robot motion task called `BE_Data`, or BullsEye will be unable to save and retrieve data.

5.3 Using BullsEye

Introduction

The user module in your system may look different than the basic example used in this procedure, however, all user modules will make calls to BullsEye methods like `BECheckTcp` and `BESetupToolJ`. This section focuses solely on the flexibility of these global methods themselves.

This section will focus on a discussion of `BESetupToolJ`, followed by an overview of `BECheckTcp`. More detailed, technical descriptions of any of these global methods may be found in section *Instructions on page 71*.

After reading this section you will know how to:

1. Reference appropriate scan data, device data, and tool design data when calling the setup routine, `BESetupToolJ`.
2. Create copies of default scan data, device data, and tool design data, make changes to those copies, and ultimately reference these new instances.
3. Use the optional arguments in all the global methods to tailor the behavior to your needs.

Continues on next page

5.3.1 The global methods of BullsEye

The term global method

BullsEye has several global methods used to access BullsEye features. The term, *global methods*, refers to RAPID instructions that are *visible* from your RAPID program. That is to say that the instructions may be *called* from your RAPID program in the same way you might make a *call* to the `MoveJ` instruction.

BullsEye global metohods

The BullsEye global methods are:

`BECheckTcp`	Check the TCP.
`BEDebugState`	Turn on/off debug logging.
`BERefPointer`	Move to the reference pointer.
`BESetupToolJ`	Setup the tool by making an initial measurement.
`BETcpExtend`	Change the TCP extension without re-measuring the tool.
`BEUpdateTcp`	Measure the tool and update regardless of the measured error.

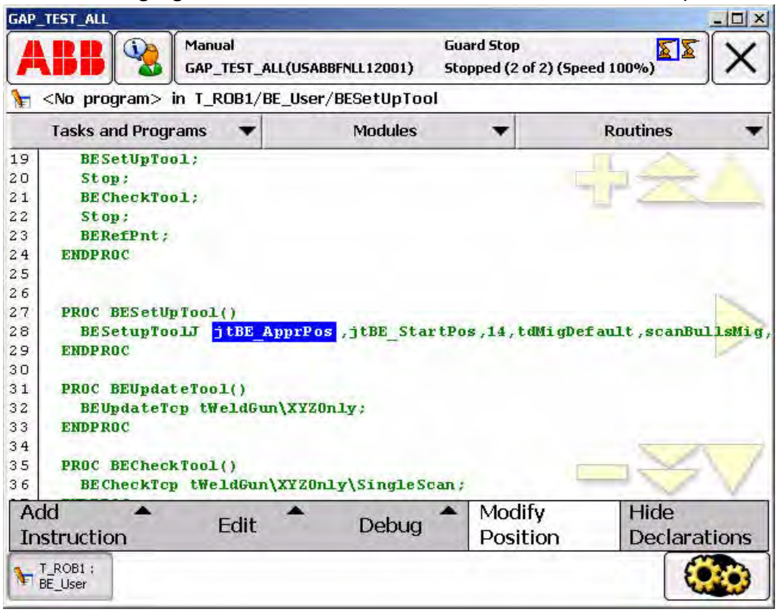
5.3.2 Defining a tool

Defining a tool

<table>
<tr><th></th><th>Action</th></tr>
<tr><td>1</td><td>Use the `BESetupToolJ` instruction to define a tool. This instruction adds a `tooldata` instance to the BullsEye collection of tools, defines the starting position, and lets BullsEye know how it should behave when other global methods are called. This information is passed to the instruction through several required and optional arguments.

```
BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scan-
    BullsMig, devYokeUp, v100, fine, tWeldGun;
```

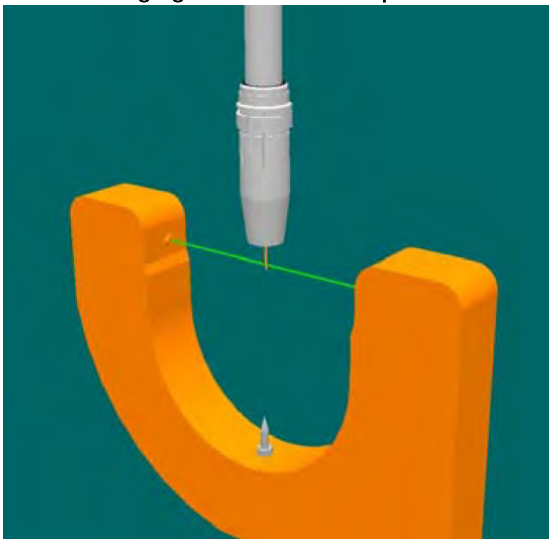
The following figure shows ModPos of the instruction `BESetupToolJ`.



xx1400001217

The instruction contains two `jointtarget` arguments, and one `tooldata` argument. As a result, the `jointtarget` may be modified using `ModPos`.</td></tr>
</table>

Continues on next page

<table>
<tr><th></th><th>Action</th></tr>
<tr><td>2</td><td>The approach position, in this example, <code>jtApprPos</code>, is an intermediate point that should be defined near the BullsEye sensor to allow unobstructed access to the sensor.
The start position, in this example, <code>jtStartPos</code>, defines the starting point for the measurement scans. The movements made by the global method <code>BESetupToolJ</code> are dictated by this starting position. This position must be chosen so that the robot will not reach its joint limits or pass too close to singularity. This takes practice and patience. Try to choose a position that does not put the robot near its joint limits to start. The start position should have the actual TCP near the center of the beam.
The following figure shows a start position.

xx1400001218</td></tr>
<tr><td>3</td><td>After the start position comes the TCP extension. This is the length of the TCP extension in millimeters. On a MIG welding torch this corresponds to wire stick-out as measured from the end of the gas cup.
<code>BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scanBullsMig, devYokeUp, v100, fine, tWeldGun;</code></td></tr>
<tr><td>4</td><td>After the TCP extension comes three BullsEye specific data types called <code>Tool Design Data</code>, <code>Scan Data</code>, and <code>Device Data</code>.
These three data types provide configurable parameters used to influence the behavior of BullsEye for the newly added tool. The names of the data type are <code>be_tooldesign</code>, <code>be_scan</code>, and <code>be_device</code>, respectively. This section will cover some of the basic parameters. For more detailed information refer to the section Data types on page 61.
<code>BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scanBullsMig, devYokeUp, v100, fine, tWeldGun;</code></td></tr>
<tr><td>5</td><td>The next argument in the <code>BESetupToolJ</code> instruction is the <code>speeddata</code> argument. The robot will move to the approach position with this TCP speed.
<code>BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scanBullsMig, devYokeUp, v100, fine, tWeldGun;</code></td></tr>
<tr><td>6</td><td>The <code>BESetupToolJ</code> instruction contains a <code>zonedata</code> argument. This zone will affect the behavior of the path as the robot moves past the approach position.
<code>BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scanBullsMig, devYokeUp, v100, fine, tWeldGun;</code></td></tr>
</table>

Continues on next page

	Action
7	**The next argument is the tool. All information passed to BullsEye with the** `BESetupToolJ` **instruction will be associated by the tool name.** `BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scanBullsMig, devYokeUp, v100, fine, `**`tWeldGun;`**

5.3.3 Default BullsEye data

Introduction

The BullsEye installation includes default data instances `be_tooldesign`, `be_scan`, and `be_device` that may be used directly, or copied for use in, the `BESetupToolJ` instruction.

These defaults include:

`tdMigDefault`	Default tool design parameters for a typical MIG welding torch.
`tdCutTool`	Default tool design parameters for a typical plasma or laser cutting head used with the standard BullsEye yoke scanning device.
`tdArtificialExt`	Some tools are best defined by adding a hardware extension probe to the end of the tool. This example contains data for a typical probe.
`tdCalibBall`	Calibration tooling balls are sometimes used for calibrating the robot cell. When a small tooling ball is mounted on the robot as a tool, this data instance will provide data that allows BullsEye to find the center of the ball.
`devYokeUp`	Default device data for a standard BullsEye yoke scanning device positioned with the yoke facing up relative to the robot base.
`devYokeDown`	Default device data for a standard BullsEye yoke scanning device positioned with the yoke facing down relative to the robot base.
`scanBullsMig`	Default scan data for a standard MIG torch with wire extension.
`scanCutTool`	Default scan data for a typical cutting head used with the standard BullsEye yoke scanning device.

Usage

Any of these default data instances may be used in the `BESetupToolJ` instruction. In the example used in this section, the defaults `tdMigDefault`, `scanBullsMig`, and `devYokeUp`, are used. These are good parameters for a standard MIG torch like the one shown in *Defining a tool on page 40*, used with the standard BullsEye yoke-style scanning device.

5.3.4 Selecting different BullsEye data

Introduction

Sometimes it is necessary to choose a different data instance. Consider a system where the BullsEye yoke is mounted upside down.

Illustration: scan device orientations

xx1400001219

xx1400001220

The image on the left shows the yoke mounted right side up.The figure on the right shows the yoke mounted upside down. If the yoke is mounted upside down, we can not use the default device data, `devYokeUp`, because its parameters will be incorrect.

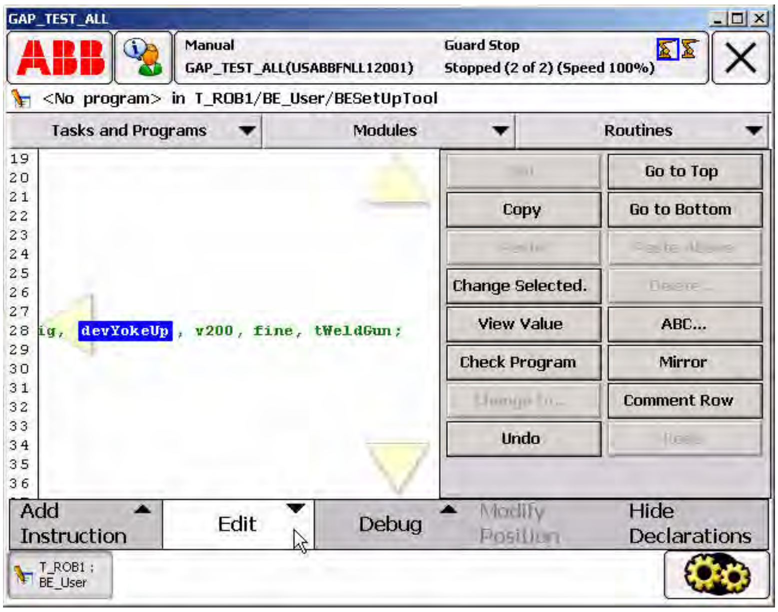
Continues on next page

Selecting different BullsEye data

<table>
<tr><th></th><th>Action</th></tr>
<tr><td>1</td><td>

To select a different data instance:

Select the device data argument in the `BESetupToolJ` instruction. Then tap **Change Selected** in the **Edit** menu.

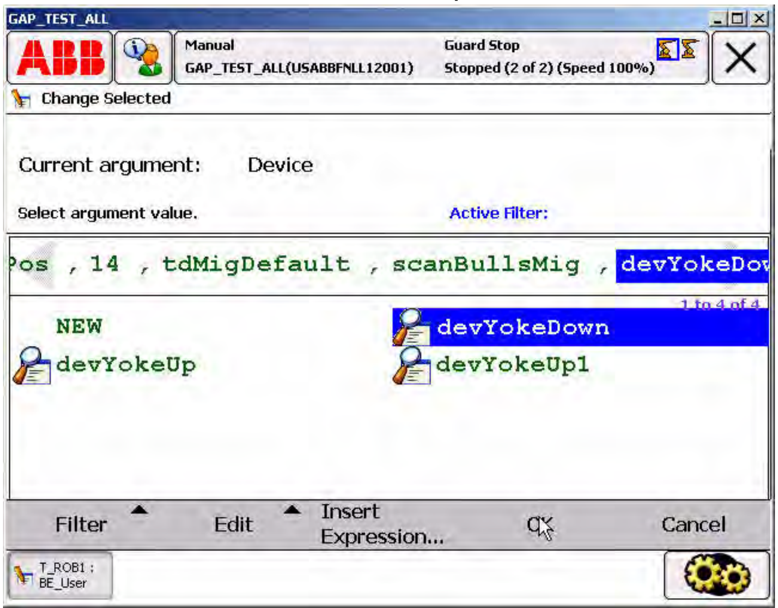


xx1400001221

</td></tr>
<tr><td>2</td><td>

A list of all available device data will be presented.

Select the `devYokeDown` instance and tap **OK**.



xx1400001222

</td></tr>
<tr><td>3</td><td>

The new device data is now added to the `BESetupToolJ` instruction. When the instruction is run, the parameters included in `devYokeDown` will be associated with `tWeldGun`.

```
BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scanBullsMig, devYokeDown, v100, fine, tWeldGun;
```

</td></tr>
</table>

Continues on next page

Note

This general procedure is used for choosing new `be_scan` and `be_tooldesign` data, also.

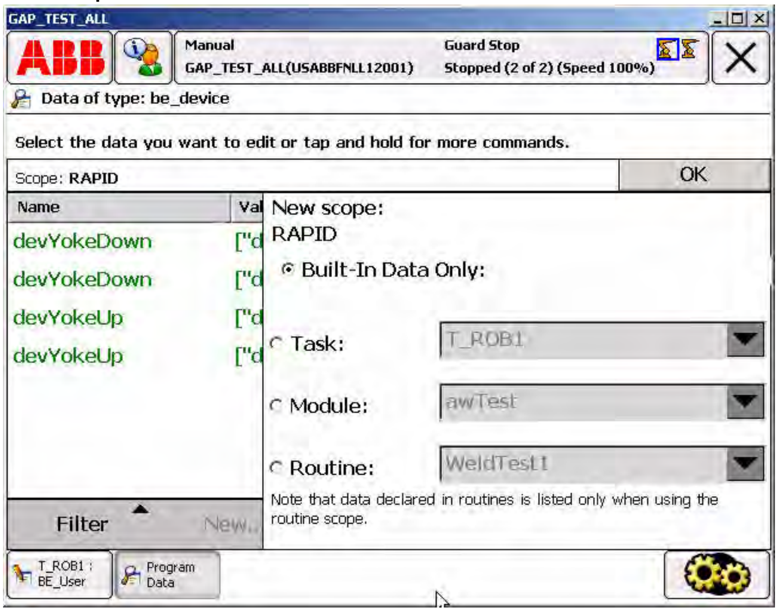
5.3.5 Creating new BullsEye data instances

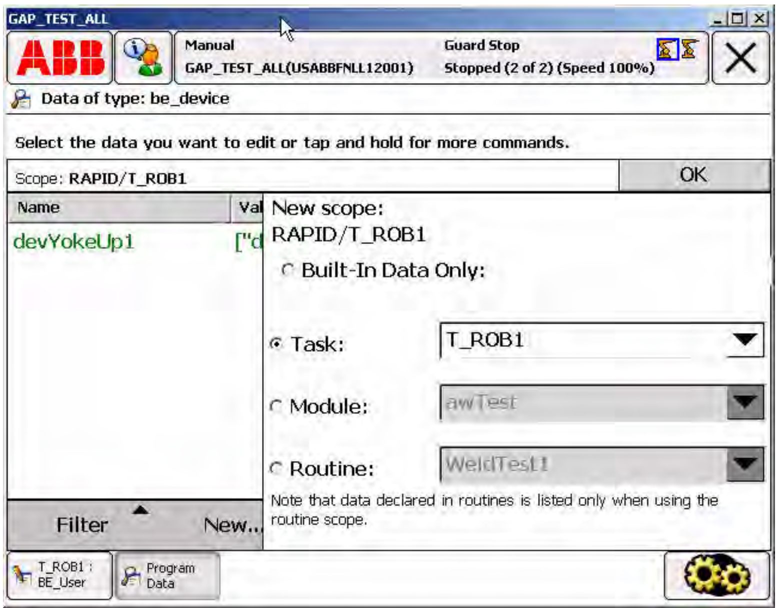
Introduction

The default `be_device`, `be_tooldesign`, and `be_scan` data instances provided with BullsEye cannot be changed because the module is declared as a read-only resource. Suppose the default parameters provided do not support the BullsEye setup in your system. A common parameter that sometimes requires a change is the *Signal Name*. The BullsEye scanning device is wired to a digital input in the controller. The signal name used in BullsEye must match the signal name defined in system parameters. Creating a new `be_device` data instance allows us to make that change.

Continues on next page

Creating new BullsEye data instances

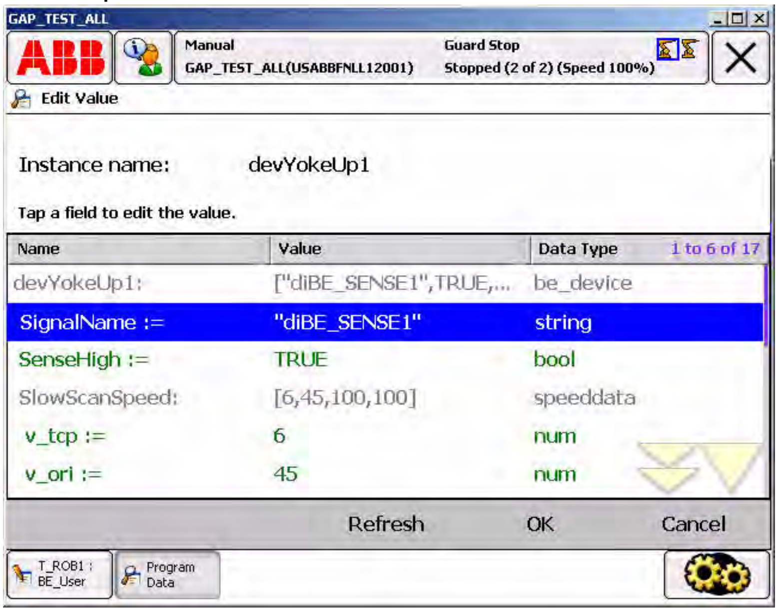
<table>
<tr><th></th><th>Action</th></tr>
<tr><td>1</td><td>From the Program Data window, view the <code>be_device</code> data in the system.
The following figures shows viewing the be_device data with built-in scope and with task scope.


xx1400001223


xx1400001224</td></tr>
</table>

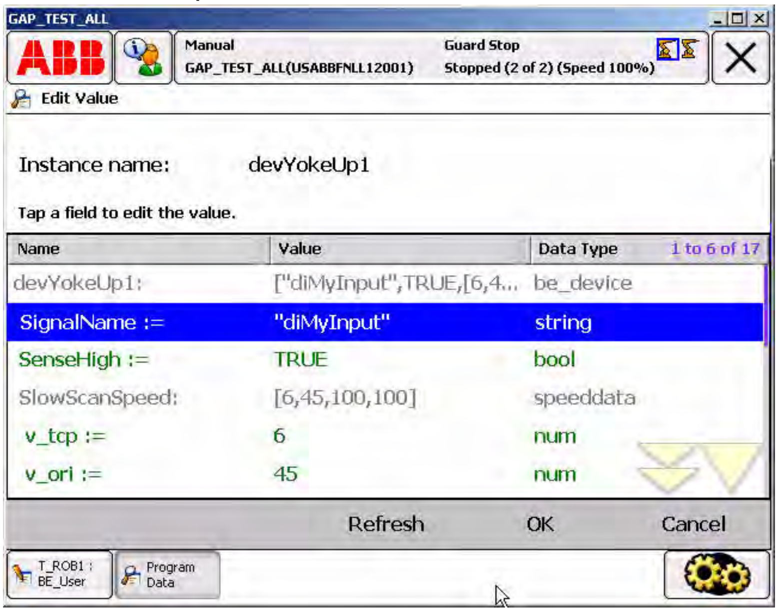
Continues on next page

	Action
2	The new data instance may be modified because it was declared in an *open* module, meaning it is not *read-only*. We need to modify the *Signal Name*. • Tap **Enter** to view the data instance fields.



xx1400001225

	Action
3	Tap **Text** to modify the name. When finished, tap **OK** to return to the list of `be_device` data instances.



xx1400001226

This new data instance can be used in the your `BESetupToolJ` instruction, see *Selecting different BullsEye data on page 44*.

```
BESetupToolJ jtApprPos, jtStartPos, 15, tdMigDefault, scan-
    BullsMig, devYokeUp1, v100, fine, tWeldGun;
```

Continues on next page

Note

This general procedure is used for choosing new `be_scan` and `be_tooldesign` data, also.

5.3.6 BullsEye data parameters

Introduction

The parameter fields in `be_device`, `be_scan`, and `be_tooldesign` data are described in their entirety in section *Data types on page 61*. If the default data instances provided by BullsEye cannot solve your particular BullsEye implementation problem, review the detailed analysis of each BullsEye data type before attempting to create your own versions.

Execution

When `BESetupToolJ` is executed, the robot will make a move to the start position, via the approach position, that is defined in the instruction. It will begin searching for the scanning device beam. If it can locate it, the robot will begin executing a series of scans to measure the TCP of the tool.

BullsEye measures the TCP several times to verify that the measurements have converged to a common solution. A typical setup should take about 10 minutes to complete. If there is a problem with robot calibration, the tool mounting hardware, or other factors not compensated for by BullsEye, the setup routine will fail and report a status message indicating the problem. In this case BullsEye may attempt to find a solution for up to 20 minutes before reporting a convergence error and halting execution.

The most common problem encountered while running the setup is a joint limit error. Joint limit errors occur when the robot tries to move to a position that is outside the working range of the robot. When this occurs, a new start position must be chosen and the `BESetupToolJ` instruction re-executed. It takes some practice to be able to run the setup on the first try. It is best to try running the BullsEye *before* permanently mounting the sensor, in case you find that it must be moved to complete the setup.

5.3.7 QuickCheck

About QuickCheck

QuickCheck is the trade name for the TCP evaluation features offered by the global method `BECheckTcp`. `BECheckTcp` may be called for any tool that has been initialized and set up.

```
BECheckTcp tWeldGun\XYZOnly\Status:=beStatus;
```

Function

When called, the robot makes a move to the start position via the approach position. Two complete scans are made. If the preliminary measurement shows a deviation, the robot will continue to make a complete measurement of the tool. Otherwise, the robot returns to the calling routine and no change is made to the TCP. If the tool is measured and found to have changed, then the tool is updated before returning to the calling routine.

Automatic mode

When running in automatic operating mode the update happens automatically without a prompt.

Manual mode

When running in manual operating mode, the user will be prompted for action before updating the tool.

Note

It is common practice to call `BECheckTcp` after a certain time or after a certain number of parts has been processed to ensure that the TCP is always correct.

Optional arguments

Like the `BESetupToolJ` instruction, `BECheckTcp` has several optional arguments.

`XYZOnly`

One commonly used optional switch is `XYZOnly`. If selected, the instruction will only update the translation portion of the `tooldata` when it is required to update the TCP definition. In this case the orientation of the tool is unaffected. Using this switch decreases the update time by about half. Keep in mind that large changes in TCP translation without updating orientation can eventually lead to problems in cases where tool orientation is critical as in a torch cleaning routine.

`Status`

Another commonly used optional argument is the `Status` argument. The `Status` argument returns an integer that may be evaluated in the calling RAPID code. Each error condition returns a unique error number.

5.4 BullsEye status codes

About status codes

BullsEye uses status codes to report errors from the user instructions. The error code may be captured using the `INOUT Status` parameter in `BEUpdateTcp`, `BERefPointer`, and `BECheckTcp`.

List of error codes

The following is a list of the error codes and a brief description for each. These error codes are global constants of the alias `num` type, `be_status`.

Error name	Error code	Description
BESuccess	1	If the instruction is executed in its entirety with no errors, status will be set to `BESuccess`.
BENoOverwrite	2	If the `OverWrite` flag was not set and the tool is already included in the BullsEye Collection, this code will be raised by `BESetupToolJ`. Add the optional switch, `OverWrite`, to the instruction to over write the existing data.
BENoNameMatch	3	No data could be located for the tool selected. Re-initialize the tool with `BESetupToolJ` to correct the problem.
BENoBEDataMod	4	The system module, `BE_Data`, appears to be missing. Load the module before continuing.
BEArrayFull	5	BullsEye will accept up to 5 tools.
BEToolNotFound	6	No data could be located for the tool selected. Re-initialize the tool with `BESetupToolJ` to correct the problem.
BEInvalidSignal	7	This digital input name used in the `be_device` data is invalid. Verify that the signal exists.
BEAliasSet	8	The connection to the digital input specified in the `be_device` data could not be made. Verify that the signal exists and is accessible.
BERangeLimFail	9	A joint limit will be exceeded if BullsEye attempts to run the scanning process. Try reinitializing the tool with a new start position using `BESetupToolJ`, or try moving the scanning device to a new location and re-initializing.
BERangeSingFail	10	The robot will run close to singularity if BullsEye attempts to run the scanning process. Try re-initializing the tool with a new start position using `BESetupToolJ`, or try moving the scanning device to a new location and re-initializing.
BERangeTiltFail	11	No acceptable tilt direction could be found for the scanning process. Try re-initializing the tool with a new start position using `BESetupToolJ`, or try moving the scanning device to a new location and re-initializing.
BEScanPlaneErr	12	BullsEye could not determine the scan plane of the device. Report this error to ABB.
BEBFrameNotRead	13	The base frame definition of the robot could not be found. Please verify that the robot is referred to as the master in system parameters. Report this error to ABB if the problem cannot be determined.

Continues on next page

Error name	Error code	Description
BEScanRadZero	14	The parameter `InitPatternRad`, in `be_scan` data is negative or zero. For a standard yoke-style beam-type scanning device, this value should be about 25 mm. Correct the data problem before retrying.
BEHeightSrchErr	15	The height search failed. Check that the proximity sensor in the tool is working properly and check that the height search instruction is named correctly in `be_scan` data. The height search instruction is tool-dependent and is not a part of the BullsEye software.
BEBeamNotFound	16	The robot could not locate the sensing beam of the scan device. Check to see that the tool is passing through the beam and that the sensor is triggering the digital input associated with it.
BEBeamSpinErr	17	Although the beam was located, its orientation could not be determined.
BESrchErrInBeam	18	BullsEye attempted to make a scan, but the start position of the scan broke the beam. Check that the tool dimensions are correct in `be_tooldesign`. Check that the scan margins are sufficient in `be_scan`. Check that the scanning device is triggering properly. Check that the robot is calibrated.
BESrchErrNoDet	19	BullsEye attempted to make a scan, but the scanning device never detected the tool. Check that the tool dimensions are correct in `be_tooldesign`. Check that the scanning device is triggering properly. Check that the robot is calibrated.
BENumOfScansErr	20	The number of redundant scans requested in the `be_scan` data, is less-than or equal to zero, or is not an integer.
BEDiaZeroOrLess	21	While scanning to find the center of the tool, the diameter of the tool was found to be less-than or equal to zero. Check that the tool dimensions are correct in `be_tooldesign`. Check that the scanning device is triggering properly. Check that the robot is calibrated.
BESliceCountErr	22	BullsEye will take "slices" of the tool to find the end of the tool. If it cannot find the end of the tool in a reasonable number of scans, the instruction will be aborted and this message will be raised. Verify that the flag, `Inverted`, is set properly in `be_device` data. Verify that the slice thickness specified in `<be_tooldesign>.SliceGap` is appropriate. Verify that the start position is defined correctly.
BEGetNewTcpMax	23	BullsEye will iterate until it converges to a TCP definition that is within the requested repeatability. If it cannot arrive at a good TCP after a reasonable number of iterations, the process will be aborted and this error code will be raised. This error can result if the repeatability, specified in the `be_device` data, is unreasonably small, or if the robot has an accuracy problem. Robot accuracy problems can be caused by incorrect calibration or damaged robot arm components.
BEBeamOriFail	24	The beam orientation could not be fine-tuned correctly. Check that the tool is perpendicular to the scan plane when at the start position.
BEGetTcpDelErr	25	BullsEye failed to determine the change in the TCP for the current iteration. This problem typically arises when the robot calibration is wrong, or when tool dimensions specified in `be_tooldesign` are incorrect.

Continues on next page

Error name	Error code	Description
BERefPosSetErr	26	Reference position data could not be written to `BE_Data`.
BERefToolSetErr	27	Reference tool data could not be written to `BE_Data`.
BERefBeamSetErr	28	Reference beam data could not be written to `BE_Data`.
BEBFrameDefErr	29	BullsEye does not understand the base frame definition of the robot. Verify that the manipulator parameters are correct (MOC.cfg).
BESetupAlready	30	This tool is already set-up. Use the optional argument `n` with `BESetupToolJ` to redo the setup.
BERefResetErr	31	The reference data could not be reset. This indicates that BullsEye could not write to the `BE_Data` module.
BESetupFailed	32	The instruction `BESetupToolJ` failed for some unknown reason.
BE Start Not Set	33	The start position is not set for this tool. Run `BESetupToolJ` to correct the problem.
BEToolNotSet	34	The tool is not set up. Run `BESetupToolJ` to correct the problem.
BEStartChanged	35	The start position has changed. This can only occur by manually changing data in the `BE_Data` module, loading a `BE_Data` module from a different robot, or by loading the wrong version of the `BE_Data` module. Load the correct `BE_Data` module, or reinitialize and run the setup instruction.
BEBeamMoveErr	36	BullsEye has detected that the beam has moved. Re-run the setup.
BECheckErr	37	There was a problem in the `BECheckTcp` instruction. The cause is unknown.
BESkipUpdate	38	The TCP has moved, but the operator did not accept the change.
BEStrtningErr	39	An error occurred while straightening the tool. The tool may be very bent, the tool dimensions may be wrong in `be_tooldesign`, or the scan margins may be too small in `be_scan`.
BEAllNotSet	40	The tool is not completely set-up. Redo the setup by running `BESetupToolJ`. If the same error occurs, re-initialize the tool with `BESetupToolJ` before running `BESetupToolJ`.
BEQuikRefNotDef	41	The `QuickCheck` functionality in `BECheckTcp` could not run because the quick reference position was not saved during the setup. Redo the setup with `BESetupToolJ`.
BEConvergErr	42	BullsEye will iterate until it converges to a TCP definition that is within the requested repeatability. If it cannot arrive at a good TCP after a reasonable number of iterations, the process will be aborted and this error code will be raised. This error can result if the repeatability, specified in the `be_device` data, is unreasonably small, or if the robot has an accuracy problem. Robot accuracy problems can be caused by incorrect calibration or damaged robot arm components.
BEInstFwdErr	43	`BESetupToolJ` cannot be run in step-forward mode. Execute in continuous mode to setup the tool.

Continues on next page

Error name	Error code	Description
BEGetGantryErr	44	This tool has been initialized with the optional `UserFramePos`. The optional functionality is not working correctly and the execution has been aborted.
BEUnknownErr	300	An unknown error has occurred.

5.5 Frequently asked questions

How do I configure the digital input signal?

BullsEye scanning devices use a single digital input signal. The digital input must be defined on an I/O board. The signal is commonly given the name `diBE_SENSE1`.

```
CONST be_device devYokeUp:=["diBE_SENSE1",TRUE,…
CONST be_device devYokeUp:=["diMyNewSense",TRUE,…
```

BullsEye must be informed of the name of the digital input. The name of the signal is defined in the `be_device` data instance that is passed into the `BESetupToolJ` instruction. See `be_device` in *Data types on page 61*, and `BESetupToolJ` in *Instructions on page 71*

If the signal name is different from the default names provided, new BullsEye device data must be created. For more information about this, see section Selecting different BullsEye data.

How do I implement multiple tools?

BullsEye can handle up to five different tools at a time by simply calling `BESetupToolJ` with five different tools.

How should robot carriers be configured?

Robots moved by carriers, such as tracks, must have the user frame coordination defined for the carrier.

Example, the following definition will not work with BullsEye:

```
MECHANICAL_UNIT:
#
-name "TRACK" -use_run_enable "" -use_activation_relay "" \
-use_brake_relay "" -use_single_0 "TRACK" \
-stand_by_state -activate_at_start_up
-deactivation_forbidden
```

It should look like this:

```
MECHANICAL_UNIT:
#
-name "TRACK" -use_run_enable "" -use_activation_relay "" \
-use_brake_relay "" -use_single_0 "TRACK"
-allow_move_of_user_frame \
-stand_by_state -activate_at_start_up
-deactivation_forbidden
```

This is addition is needed to support coordinated work objects that have the user frame moved by the track. It is always recommended to define tracks and other robot carriers this way. Doing so also improves the usability of the system for other reasons beyond the BullsEye requirements.

In addition to these mechanical unit settings, we also recommend that the BullsEye sensor yoke be mounted to move with the robot. Doing so ensures that vibrations in the robot carrier do not affect the relationship between the BullsEye sensor yoke and the robot arm. Vibrations can yield poor TCP quality. Mounting the sensor with

Continues on next page

the robot also allows the possibility of executing TCP checks anywhere in the working range of the robot carrier. This can cut TCP checking time tremendously.

How do I set up BullsEye when the robot is moved by a track?

If the BullsEye scanning device is mounted on the carrier with the robot, no changes are needed. This is the preferred method since it negates the positional inaccuracy of the robot carrier. If the BullsEye scanning device is fixed in the world, then a flag must be set in the `be_device` data to inform BullsEye.

```
CONST be_device devYokeUpTrack:=["diBE_SENSE",
     TRUE,[6,45,100,100],[40,45,100,100],0.10,FALSE, FALSE,TRUE];
```

The flag in the device data is called `MovedWithRobot`. For more information see *be_device - Device data on page 61*.

Can I change my TCP extension without rerunning the initialization?

Yes. Use the `BETcpExtend` instruction, see *BETcpExtend - BullsEye extend TCP on page 83*.

Can the BullsEye yoke be mounted in any orientation?

Yes, but the BullsEye scanning device must be mounted so that the scan plane is parallel with the robot's physical base surface.

How do I set up a non-ABB supplied I/O device?

Only ABB I/O devices are guaranteed to work with BullsEye. Many I/O devices from other vendors are too slow or too unrepeatable to allow BullsEye to work correctly When using non-ABB devices, you may need to slow the scan speeds substantially to improve accuracy.

A WAGO I/O device, for example, may be used in the COS (*Change of State*) mode, but the PIT (*Production Inhibit Time*) should be reduced as much as possible, preferably to zero. This is done in the system parameter *Production inhibit time*, in the topic *I/O*, the type *Unit Type*.

What is a convergence error?

BullsEye measures the TCP more than once during the setup. It converges on a solution that is within limits influenced by the `be_device` data, `Repeatability`. If the deviation between two TCP measurements cannot be reduced to a level specified by the `Repeatability` value, BullsEye eventually times-out and reports a *convergence error*.

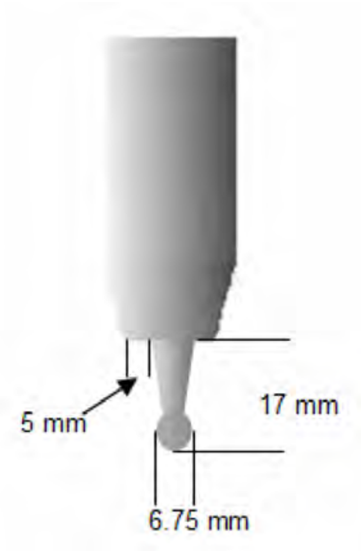
Convergence errors can occur for a variety of reasons:

Problem	Solution
Incorrect parameters are used in the setup.	This can be corrected by fixing the parameter values to match the tool and scanning equipment.
The tool is not mounted securely or tool mount bracket is too flexible.	This can be corrected by improving the tool mount.
The relationship between the BullsEye sensor and the robot base is not solid.	This can be corrected by improving the mounting structures.

Continues on next page

Problem	Solution
The I/O system is not responsive enough.	This can be corrected by reducing the search speeds.
The I/O not repeatable enough.	Non-ABB I/O equipment could be improved by changing the configuration. See *How do I set up a non-ABB supplied I/O device? on page 58*.
Motor calibration wrong.	Check calibration.
Inaccurate robot due to bearing imperfections.	Increasing the `Repeatability` value can work.
The BullsEye sensor is faulty.	Occasionally there are problems with the optical sensor. These must be replaced.

How do I setup BullsEye to calibrate a tool like this?



xx1400001227

There is a `be_tooldesign` instance provided as a default constant for a similar tool called `tdCalibBall`:

```
CONST be_tooldesign tdCalibBall:=
      [TRUE,30,1,55,12,4,FALSE,FALSE,1.2,[130,100,100,100],
      [220,130,100,100]];
```

The tool pictured above, is very similar. Assuming you want the TCP in the center of the sphere, you would create a similar `be_tooldesign` instance like this:

```
CONST be_tooldesign tdMyProbe:=
      [TRUE,30,1,50,3.5,4,FALSE,FALSE,1.2,[130,100,100,100],
      [220,130,100,100]];
```

When a tool with welding wire is measured, BullsEye cannot actually measure the real location of the end of the wire. The wire location is measured close to the end of the gas cup, and the TCP is mathematically extended down from the end of the gas cup based on the `TCP Extension` parameter passed into the `BESetupToolJ` instruction.

This approach works well for welding torches because the wire is often bent in an unpredictable direction and the length will vary. However, for a tool like the probe

Continues on next page

pictured, it is more accurate to measure the end of the tool where the TCP actually is, because we do not have to worry about variation in location.

Here is an explanation of the `be_tooldesign` parameters with comments:

Parameter	Description
OrientBody	This we want `TRUE` so we define orientation also.
MaxBodyDia	Set to a value at least as large as the largest section diameter. 30 mm, in this case.
MinBodyDia	We want to put a very small number here so that BullsEye will not think it has reached the end of the tool until it makes slices all the way past the end of the ball. We will use 1 mm.
ScanRange	Searchable portion of tool. 50 mm, in this case.
RangeShift	The ball is almost 7 mm in diameter. Putting 3.5 mm here will force the final measurement to be near the middle of the ball. If BullsEye misses the end of the ball during the setup process, this number could be increased.
SliceGap	6 mm is a good number. Small numbers are important when there are features that you don't want to miss when BullsEye is taking slices. Big numbers are good when you want the setup process to take less time.
ScanWire	This should be `FALSE`. The tool does not have a wire that we will mathematically extend out from the gas cup. Instead we will measure all the way to the end of the tool.
OffsEndSearch	We want the final z-axis search to be inline with the ball. So, this parameter should be `FALSE`. In contrast, a welding gun has a wire that is too narrow to search and the wire is always a different length. For this reason, a welding tool definition would have this parameter set to `TRUE` so that the z-axis search occurs next to the wire and searches for the end of the gas cup.
WireDia	This parameter has no affect when `OffsEndSearch` and `ScanWire` are `FALSE`.
SlowMoveSpeed	Movement speed. This is not the search speed.
FastMoveSpeed	Movement speed. This is not the search speed.

Last, the TCP extension passed into the `BESetupToolJ` instruction, must be fixed.

```
BESetupToolJ jtBEApprPos,jtBEStartPos,-3.375 , tdMyProbe...
```

A negative number will move the TCP from the end of the ball to the center of the ball. The default settings for `be_scan` and `be_device` will work fine for a standard ABB I/O board.

How do I proceed when BullsEye gives large deviations?

If BullsEye gives large deviations during reorientation, try rotating the BullsEye sensor 90 degrees in order to reduce the influences from mechanical tolerances in the robot arm.

6 RAPID reference

6.1 Data types

6.1.1 be_device - Device data

Usage

`be_device` contains parameters that are used to describe the scanning device's properties.

Components

`SignalName`

Data type: `string`

Digital input name used by the scanning device.

`SenseHigh`

Data type: `bool`

Set to true if signal is high when the detecting the tool.

`SlowScanSpeed`

Data type: `speeddata`

Slow scans will be executed with this speed setting.

See *Technical reference manual - RAPID Instructions, Functions and Data types* for an explanation of `speeddata`.

`FastScanSpeed`

Data type: `speeddata`

Fast scans will be executed with this speed setting.

See *Technical reference manual - RAPID Instructions, Functions and Data types* for an explanation of `speeddata`.

`Repeatability`

Data type: `num`

The expected repeatability for TCP measurements. This number should be about twice that of the published repeatability for the robot arm. This equates to about +/- 0.12 mm for an IRB 1400. Other factors, such as torch leads exerting undue force on the tool mount bracket, may have an adverse affect on the repeatability. In such cases it may be necessary to increase Repeatability in order for the robot to find an acceptable solution. A convergence error is reported via the BullsEye error code argument when the system cannot reach the desired repeatability within a reasonable time.

Units: mm

Continues on next page

`Inverted`

Data type: `bool`

If TRUE invert the scan plane relative to robot base.

xx1400001219

Device upright

xx1400001220

Device inverted

`MovedWithRobot`

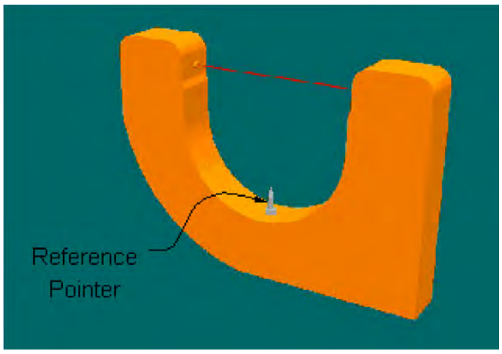
Data type: `bool`

If the robot baseframe is moved by a mechanism, does the BullsEye move with it? If not, set this to FALSE.

`RefPoint`

Data type: `bool`

If there is a reference pointer to define, set this parameter to TRUE.



xx1400001228

Structure

```
<dataobject of be_device>
  <SignalName of string>
  <SenseHigh of bool>
  <SlowScanSpeed of speeddata>
  <FastScanSpeed of speeddata>
  <Repeatability of num>
  <Inverted of bool>
```

Continues on next page

```
<MovedWithRobot of bool>
<RefPoint of bool>
```

Related information

	Described in:
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*
be_scan	*be_scan - Scan data on page 64*
be_tooldesign	*be_tooldesign - Tool design on page 67*

6.1.2 be_scan - Scan data

Usage

`be_scan` describes how BullsEye® should behave during the scanning process.

Components

`NumOfScans`

Data type: `num`

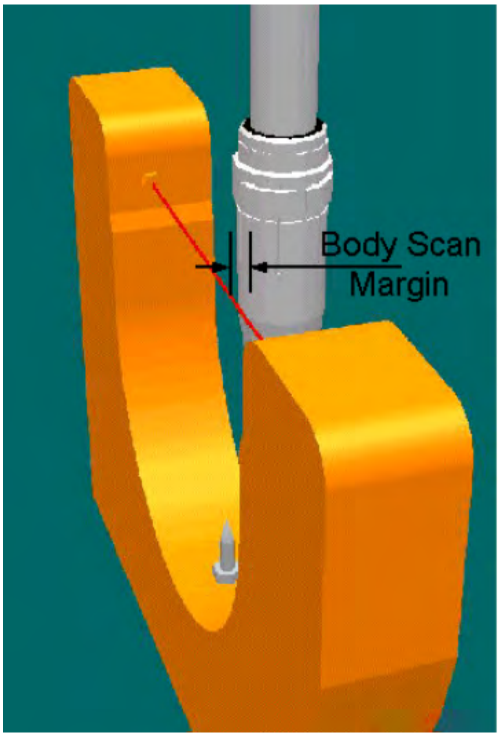
The number of redundant scans is defined here. Redundant scanning will give better repeatability and accuracy.

`BodyScanMargin`

Data type: `num`

This distance (mm) plus half the `MaxBodyDia` from `be_tooldesign` gives the start offset of the body scan.

Units: mm



xx1400001229

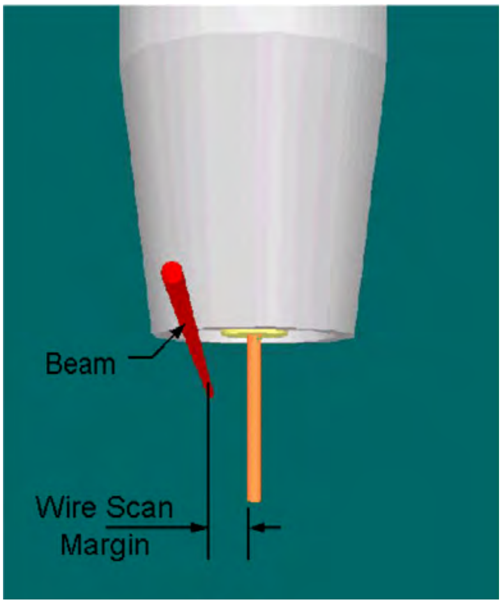
`WireScanMargin`

Data type: `num`

This distance (mm) plus half the `WireDia` from `be_tooldesign` gives the start offset of the wire scan.

Continues on next page

Units: mm



xx1400001230

`TwistAngle`

Data type: `num`

Plus and minus `TwistAngle` gives overall twist envelope for scans.

Units: degrees

`TiltAngle`

Data type: `num`

From no-Tilt to `TiltAngle` gives overall Tilt envelope for scans.

Units: degrees

`InitPatternRad`

Data type: `num`

Initial pattern radius when scanning for beam orientation. Use 25 mm for standard MIG torch and standard yoke-type scanning device.

Units: mm

Structure

```
<dataobject of be_scan>
  <NumOfScans of num>
  <BodyScanMargin of num>
  <WireScanMargin of num>
  <TwistAngle of num>
  <TiltAngle of num>
  <InitPatternRad of num>
```

Continues on next page

Related information

	Described in:
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*
be_device	*be_device - Device data on page 61*
be_tooldesign	*be_tooldesign - Tool design on page 67*

6.1.3 be_tooldesign - Tool design

Usage

The `be_tooldesign` data type describes the tool dimensions and other related physical properties.

Components

`OrientBody`

Data type: `bool`

If selected, the orientation of the tool will be found by scanning the tool body.

`MaxBodyDia`

Data type: `num`

The maximum tool body diameter within the scan range.

Units: mm

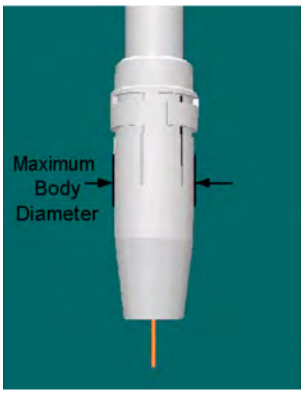
xx1400001231

`MinBodyDia`

Data type: `num`

The minimum tool body diameter within the scan range. This is typically the diameter at the "end" of the tool.

Units: mm



xx1400001232

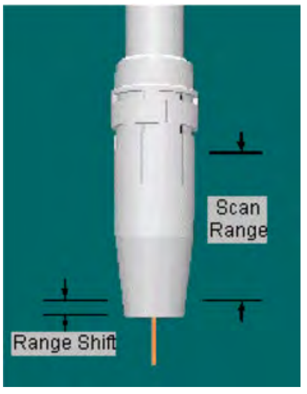
Continues on next page

`ScanRange`

Data type: `num`

The length of cylindrical tool section used during tool straightening. This portion is measure from the end of the RangeShift.

Units: mm



xx1400001233

`RangeShift`

Data type: `num`

Length of lower tool body section to ignore. This is measured from the "end" of the tool. The `RangeShift` is useful in ignoring weld spatter on a MIG welding torch.

Units: mm

`SliceGap`

Data type: `num`

When scanning to find the end of the tool BullsEye® takes "slices" of the tool until the end is found. The SliceGap is the thickness of each slice.

Units: mm

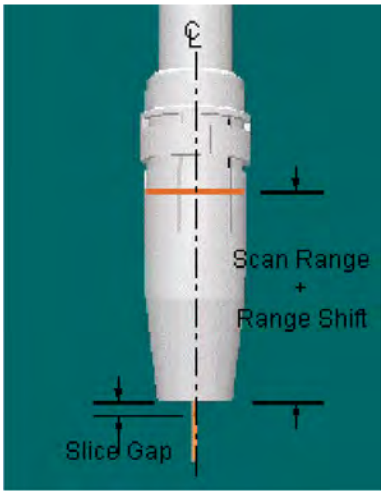
`ScanWire`

Data type: `bool`

If ScanWire is TRUE, then BullsEye® will look for a wire or similar narrow extension at the end of the tool. Otherwise the TCP will be determined by measuring the end of the tool body. When ScanWire is true, the tool centerline is measured by scanning the wire a distance of one SliceGap from the end of the tool body. When ScanWire

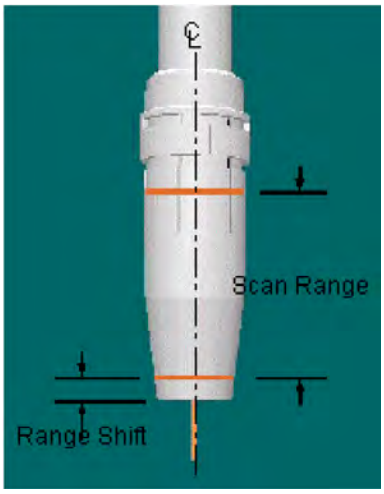
Continues on next page

is FALSE, the tool centerline is measured on the tool body a distance of one RangeShift up from the end of the tool body.



xx1400001234

Figure 6.1: ScanWire: TRUE



xx1400001235

Figure 6.2: ScanWire: FALSE

`OffsEndSearch`

Data type: `bool`

If selected, the z-search will be offset from the tool centerline. This is used to ignore a narrow TCP extension, like a welding wire. When ScanWire is TRUE, this parameter has no effect as the z-search will be offset automatically.

`WireDia`

Data type: `num`

The WireDia defines the approximate diameter of the wire or similar TCP extension. This parameter has no effect when ScanWire is FALSE.

Units: mm

`SlowMoveSpeed`

Data type: `speeddata`

Slow movements will be executed with this speed setting. See the RAPID Reference Manual for an explanation of speeddata.

> **CAUTION**
>
> Setting this parameter too high may cause damage to the work tool or may introduce resonance into large gantry-style robot applications.

`FastMoveSpeed`

Data type: `num`

Fast movements will be executed with this speed setting. See the RAPID Reference Manual for an explanation of speeddata. Caution: Setting this parameter too high may cause damage to the work tool or may introduce resonance into large gantry-style robot applications.

Continues on next page

Structure

```
<dataobject of be_tooldesign>
  <OrientBody of bool>
  <MaxBodyDia of num>
  <MinBodyDia of num>
  <ScanRange of num>
  <RangeShift of num>
  <SliceGap of num>
  <ScanWire of bool>
  <OffsEndSearch of bool>
  <WireDia of num>
  <SlowMoveSpeed of speeddata>
  <FastMoveSpeed of speeddata>
```

Related information

	Described in:
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*
be_device	*be_device - Device data on page 61*
be_scan	*be_scan - Scan data on page 64*

6.2 Instructions

6.2.1 BECheckTcp - BullsEye check TCP

Usage

`BECheckTcp` is used to measure deviation in a tool that has been previously initialized and set up with `BESetupToolJ`.

Basic examples

```
BECheckTcp tTestTemp;
```

The tool, `tTestTemp`, will be measured by making two scans. This is known as the QuickCheck. If the measurement indicates that the tool TCP has moved, BullsEye will do a complete evaluation to get the new TCP. If the change is found to be less than the maximum allowed change, the TCP will be updated.

```
BECheckTcp tTestTemp\XYZOnly\Status:=beStatus;
```

As in the previous example, the tool will be updated if necessary. However, only the translation properties of the TCP will be changed. The orientation of the TCP will not be scanned and will not be updated. This option is used to decrease the time it takes to update the TCP.

Arguments

```
BECheckTcp Tool [\UserInterface] [\XYZOnly] | [\XYOnly]
     [\SingleScan] [\ElapsedTime] [\Status] [\TLoad]
```

`Tool`

Data type: `tooldata`

Tool is the tooldata instance that will be evaluated. The tool must be initialized and setup using the instruction, `BESetupToolJ`, before `BECheckTcp` can be used.

`[\UserInterface]`

Data type: `string`

An optional user interface may be specified here. Indicate the name of the procedure and the module name.

Example: `"MyUseInt:MyBEUserInter"`. Although the name of the procedure may be altered, the structure of the arguments must follow this model:

```
PROC MyBEUserInter(
  VAR num Response,
  string Header,
  string FkKey{*},
  string TextList{*},
  num DimSet,
  be_status Condition)
  <body of procedure>
ENDPROC
```

`[\XYZOnly]`

Data type: `switch`

Continues on next page

If selected, the orientation of the tool will not be measured and will not be updated. Use this switch when it is undesirable to update the orientation, when the tool design makes tool straightening impossible, or when update time must be shortened. Update time may be reduced by as much as 50% when using this optional switch.

`[\SingleScan]`

Data type: `switch`

If selected, the initial QuickCheck will use single scans, even if the `NumOfScans` in `be_scan` data is set to a number higher than one. This override may be used to shorten the QuickCheck time. Using this switch sometimes causes the robot to run a full measurement sequence due to the limited accuracy of single scans.

`[\XYOnly]`

Data type: `switch`

If selected, the TCP may be updated based on the result of the QuickCheck only. With this option, the update time is greatly reduced, but the resulting accuracy may not be ideal. With this option, neither the z-dimension of the tool, nor the orientation of the tool, is updated.

CAUTION

This is not a recommended BullsEye method.

`[\ElapsedTime]`

Data type: `num`

This parameter will return the overall time required to complete the QuickCheck plus any TCP updating time.

Units: seconds

`[\Status]`

Data type: `be_status`

This optional parameter returns the status code. A status code other than 1 indicates a problem in execution. For a list of possible status codes, see *BullsEye status codes on page 53*.

`[\TLoad]`

Data type: `loaddata`

The `\TLoad` argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the `\TLoad` argument is used, then the `loaddata` in the current `tooldata` is not considered.

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Continues on next page

Program execution

The robot will move to the initial position for the tool. A QuickCheck will be made consisting of two scans. If the TCP has not changed appreciatively, the robot will return to production. If the change is found to be greater than the minimum threshold defined during the tool initialization, a full measurement will be made. The change will be evaluated again. In rare cases, the change may appear to be smaller after this step and no update will be made. This is due to the fact that the QuickCheck does not gather enough information to measure the tool very accurately. In this case the robot will return to production. If a robot continues to exhibit this behavior, run the setup again by calling `BESetupToolJ` or update the tool with `BEUpdateTcp`. This should correct the problem.

In most cases, the re-evaluated TCP change will require the tool to be updated. In automatic mode, this will be done automatically before returning to production. In manual mode, the operator will be prompted for a response before the tool is updated.

Execution in stepwise mode

Execution in stepwise mode is not supported.

Error handling

Known errors are raised as BullsEye error codes in the optional argument `Status`. These codes can be handled outside the instruction with standard conditional statements. BullsEye error codes are not n constants handled in a RAPID error handler.

Syntax

```
BEUpdateTcp
  [ Tool ':='] < expression (PERS) of tooldata >
  [ '\' UserInterface ':=' < expression (IN) of string > ]
  [ '\' XYZOnly ] < switch >
  [ '|' XYOnly ] < switch >
  [ '\' SingleScan ] < switch >
  [ '\' ElapsedTime ':=' < expression (INOUT) of num > ]
  [ '\' Status ':=' < expression (INOUT) of be_status > ]
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

	Described in:
be_device	*be_device - Device data on page 61*
be_scan	*be_scan - Scan data on page 64*
be_tooldesign	*be_tooldesign - Tool design on page 67*
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*
Definition of `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*

6.2.2 BEDebugState - Debug state control

Usage

`BEDebugState` is used to control the debug log detail level. Normally only limited information in stored in the BullsEye log files. With this instruction, more detailed information is recorded to help advanced users determine the cause of an error.

This instruction is hidden from the IPL.

Basic examples

```
BEDebugState\On;
```

Turns on the debugging flag.

```
BEDebugState\Off;
```

Turns off the debugging flag.

Arguments

```
BEDebugState [\On] [\Off]
```

`[\On]`

Data type: `switch`

Used to turn on debugging.

`[\Off]`

Data type: `switch`

Used to turn off debugging.

Program execution

The instruction should be placed before BullsEye instructions. The log files affected are called `BE_Oper.log` and `BE_Init.log` and are found in the folder `TEMP`.

Syntax

```
BEDebugState
  [ '\' On ] < switch >
  [ '|' Off ] < switch > ';'
```

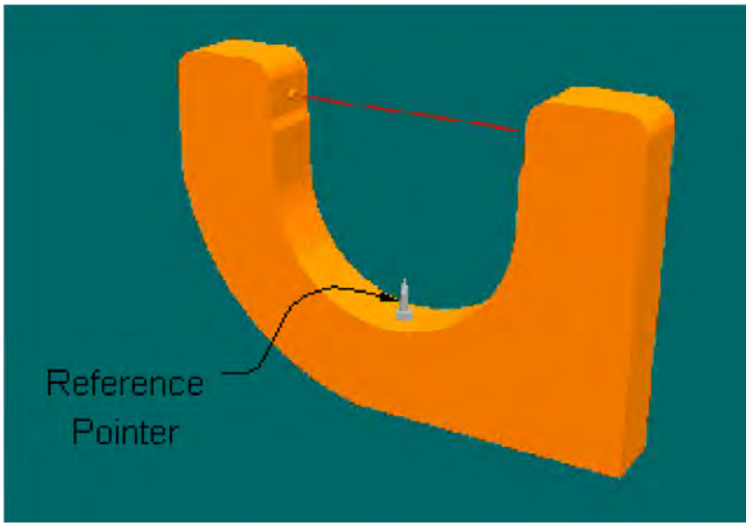
Related information

	Described in:
BECheckTcp	*BECheckTcp - BullsEye check TCP on page 71*
BEUpdateTcp	*BEUpdateTcp - BullsEye update TCP on page 85*
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*

6.2.3 BERefPointer - BullsEye reference pointer

Usage

`BERefPointer` is used to view the deviation in a tool that has been previously initialized and setup with `BESetupToolJ`.



xx1400001228

Basic examples

```
BERefPointer tTestTemp;
```

The robot will move to the scanning device and prompt the user with a choice to move to the reference pointer with the Day1 TCP definition, or with the current TCP definition. No changes will be made to the TCP.

Arguments

```
BERefPointer Tool [\UserInterface] [\Status] [\TLoad]
```

`Tool`

Data type: `tooldata`

Tool is the tooldata instance that will be evaluated. The tool must be initialized in the BullsEye Collection with the instruction `BESetupToolJ` before `BERefPointer` can be used.

`[\UserInterface]`

Data type: `string`

An optional user interface may be specified here. Indicate the name of the procedure and the module name.

Example: `"MyUseInt:MyBEUserInter"`. Although the name of the procedure may be altered, the structure of the arguments must follow this model:

```
PROC MyBEUserInter(
  VAR num Response,
  string Header,
  string FkKey{*},
  string TextList{*},
```

Continues on next page

```
        num DimSet,
        be_status Condition)

        <body of procedure>

      ENDPROC
```

`[\Status], <INOUT>`

Data type: `be_status`

This optional parameter returns the status code. A status code other than 1 indicates a problem in execution. For more information on status codes, see *BullsEye status codes on page 53*.

`[\TLoad]`

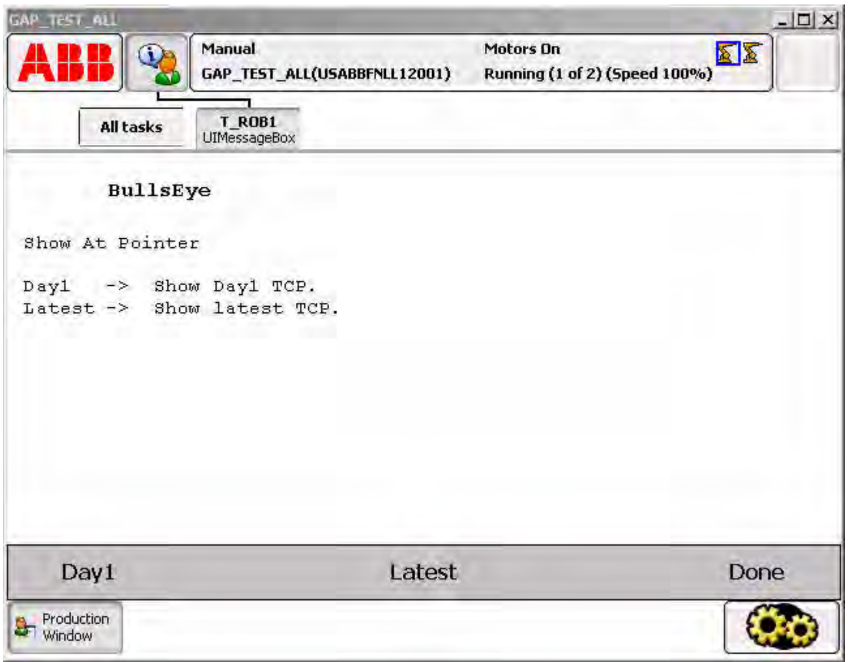
Data type: `loaddata`

The `\TLoad` argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the `\TLoad` argument is used, then the `loaddata` in the current `tooldata` is not considered.

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Program execution

The robot moves to the scanning device. No warning is given. Once the tool is positioned at the scanning device, a prompt will be presented on the FlexPendant:



xx1400001236

Pressing **Day1** or **Latest** will cause the robot to move to the pointer with each of the TCP definitions. When finished, press **Done** to return to the program.

Execution in stepwise mode

Execution in stepwise mode is not supported.

Error handling

Known errors are raised as BullsEye error codes in the optional argument `Status`. These codes can be handled outside the instruction with standard conditional statements. BullsEye error codes are not n constants handled in a RAPID error handler.

Syntax

```
BERefPointer
  [ Tool ':='] < expression (PERS) of tooldata >
  [ '\' UserInterface ':=' < expression (IN) of string > ]
  [ '\' Status ':=' < expression (INOUT) of be_status > ]
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

	Described in:
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*
Definition of loaddata	*Technical reference manual - RAPID Instructions, Functions and Data types*

6.2.4 BESetupToolJ - BullsEye setup tool joint move

Usage

`BESetupToolJ` is used to define a TCP and add the tool to the BullsEye collection. The scanning behavior is dictated by the parameters passed into the instruction.

Basic examples

```
BESetupToolJ jtApprPoint, jtStartPos,15,tdMigDefault, scanBullsMig,
     devYokeUp,v200,fine,tTestTemp;
```

The tool, `tTestTemp`, will be added to the BullsEye collection with a TCP extension of 15 mm and BullsEye parameters defined by `tdMigDefault`, `scanBullsMig`, and `devYokeUp`. BullsEye will execute a scan routine to determine the TCP, storing the results in `tTestTemp` and storing setup information in the BullsEye collection.

Arguments

```
BESetupToolJ ApprPoint StartPoint TcpExtens ToolDesign Scan Device
     Speed Zone Tool [\FixedAxes] [\ElapsedTime] [\MaxError]
     [\MaxFromDay1] [\MeanDev] [\MaxDev] [\CheckRange]
     [\CheckBeamAngle] [\TLoad]
```

`ApprPoint`

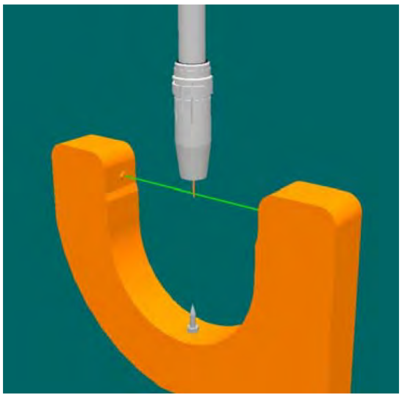
Data type: `jointtarget`

This is the approach position for the BullsEye scanning process. The tool should be defined in a position that allows free movement to the StartPoint.

`StartPoint`

Data type: `jointtarget`

This is the start position for the BullsEye scanning process. The tool should be positioned so that the tool center pointer (TCP) is located on the scan beam near its center. The tool should be oriented so that the tool is perpendicular to the scanning device's scan plane.



xx1400001218

`TcpExtens`

Data type: `num`

The length of the TCP extension, as measured from the end of the tool body, is defined here in millimeters.

Continues on next page

Units: mm

xx1400001237

`ToolDesign`

Data type: `be_tooldesign`

The ToolDesign data type describes the tool dimensions and other physical properties.

`Scan`

Data type: `be_scan`

Scan data describes how BullsEye should behave during the scanning process.

`Device`

Data type: `be_device`

This data structure contains parameters that are used to describe the scanning device's properties.

`Speed`

Data type: `speeddata`

The speed the TCP will move to the `ApprPoint`. For more information on `speeddata`, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

`Zone`

Data type: `zonedata`

The zone applied to the movement to `ApprPoint`. For more information on `zonedata`, see *Technical reference manual - RAPID Instructions, Functions and Data types*.

`Tool`

Data type: `tooldata`

Tool is the `tooldata` instance that is to be added to the BullsEye collection.

Continues on next page

[\FixedAxes]

Data type: be_fixedaxes

If the robot is moved by a multi-axis mechanical unit and the scanning device is mounted on one of links of this mechanical unit, other than the final link, this argument must be used. The structure consists of six boolean flags representing each of the six possible external axes. If an axis must be in a certain position to maintain the robot-to-scan-device relationship, then the flag for that axis should be set to TRUE. For example, if the robot is mounted on a rotating tower with linear carriage movement on the boom, then it is possible that the BullsEye scanning device could be mounted to the first link, and the robot mounted to the second link. In this case, it is necessary to set the FixedAxes flag corresponding to the linear axis to TRUE, because this axis must be driven to a designated position to fix the relationship between the scanning device and the robot.

[\ElapsedTime]

Data type: num

This parameter will return the overall time required to complete the setup.

Units: seconds

[\MaxError]

Data type: num

MaxError is the distance in millimeters that the TCP is allowed to deviate before QuickCheck will indicate the change. When not selected, MaxError will be set to four times the value of Repeatability found in the be_device data.

Units: mm

[\MaxFromDay1]

Data type: num

If the TCP is found to be more than the distance, MaxFromDay1, the tool will need to be set up again. The default is 5 mm when not selected.

Units: mm

[\MeanDev]

Data type: num

BullsEye uses four scan orientations to determine the TCP. Some deviation between measurements is normal, but excessive deviation suggests that the robot may be calibrated incorrectly, or the tool or TCP extension may be loose. This parameter may be queried to evaluate the accuracy of the TCP after the setup is complete.

Units: mm

[\MaxDev]

Data type: num

This parameter may be used together with MeanDev to evaluate the accuracy of the TCP after the setup is complete.

Units: mm

Continues on next page

`[\CheckRange]`

Data type: `switch`

If selected, the robot will make a series of moves to approximate the motion of the robot arm during the scan sequence. This argument may only be used when the supplied tool includes values that are approximately correct. This setting can be useful in determining where to mount the BullsEye sensor. This argument is used together with `CheckBeamAngle`.

`[\CheckBeamAngle]`

Data type: `num`

This argument is used to provide the orientation of the BullsEye beam relative to the base of the robot. BullsEye assumes that the sensing beam is parallel to the plane of the robot base. This value determines how the beam is oriented in that plane. The `CheckRange` argument must be used together with this argument.

`[\TLoad]`

Data type: `loaddata`

The `\TLoad` argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the `\TLoad` argument is used, then the `loaddata` in the current `tooldata` is not considered.

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Program execution

The tool is added to the BullsEye collection along with all of the data that is passed into the instruction. BullsEye will then perform a scan sequence to determine the TCP of the tool.

Execution in stepwise mode

Forward

In forward step mode, the robot will stop at the approach point. Pressing forward step again will advance the robot to the start point and start the scanning routine.

Backward

Not supported.

Error handling

Known errors are raised as BullsEye error codes in the optional argument `Status`. These codes may be handled outside the instruction with standard conditional statements. BullsEye error codes are not ERRNO constants handled in a RAPID error handler.

Syntax

```
BESetupToolJ
  [ ApprPoint ':='] < expression (IN) of jointtarget > ','
```

Continues on next page

```
[ StartPoint ':='] < expression (IN) of jointtarget > ','
[ TcpExtens ':='] < expression (IN) of num >
[ ToolDesign ':=' ] < expression (IN) of be_tooldesign > ','
[ Scan ':=' ] < expression (IN) of be_scan > ','
[ Device ':=' ] < expression (IN) of be_device >
[ Speed ':='] < expression (IN) of speeddata > ','
[ Zone ':='] < expression (IN) of zonedata > ','
[ Tool ':='] < expression (PERS) of tooldata > ','
[ '\' FixedAxes ':=' < expression (IN) of be_fixedaxes > ]
[ '\' MaxError ':=' < expression (IN) of num > ]
[ '\' MaxFromDay1 ':=' < expression (IN) of num > ]
[ '\' ElapsedTime ':=' < expression (INOUT) of num > ]
[ '\' MeanDev ':=' < expression (INOUT) of num > ]
[ '\' MaxDev ':=' < expression (INOUT) of num > ]
[ '\' CheckRange ] < switch >
[ '\' CheckBeamAngle ':=' <expression (IN) of num > ]
[ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Related information

	Described in:
be_device	*be_device - Device data on page 61*
be_scan	*be_scan - Scan data on page 64*
be_tooldesign	*be_tooldesign - Tool design on page 67*
Definition of loaddata	*Technical reference manual - RAPID Instructions, Functions and Data types*

6.2.5 BETcpExtend - BullsEye extend TCP

Usage

`BETcpExtend` is used to vary the TCP along its z-axis. The instruction may be used to modify electrode stick-out for a tool that has already been set up in BullsEye. There is no need to re-run the BullsEye initialization and setup routines after making a change with `BETcpExtend`.

xx1400001237

Basic examples

```
BETcpExtend tWeldGun\Change:=4;
```

The tool, `tWeldGun`, will be altered so that the TCP definition is now 4 mm longer. All setup information is automatically updated so that `BECheckTcp` and other methods may still be called.

Arguments

```
BETcpExtend Tool [\Change] | [\Absolute] [\NewExtens] [\Status]
```

`Tool`

Data type: `tooldata`

`Tool` is the `tooldata` instance that will be modified. The tool must be set-up using the instruction, `BESetupToolJ`, before `BETcpExtend` can be used.

`[\Change]`

Data type: `num`

This is the amount that the TCP will be extended along its z-axis.

`[\Absolute]`

Data type: `num`

This is the absolute TCP extension that is requested.

`[\NewExtens]`

Data type: `num`

Continues on next page

Returns the value of the new TCP extension. This is useful when using the `Change` argument to get the resulting TCP extension.

`[\Status]`

Data type: `be_status`

This optional parameter returns the status code. A status code other than 1 indicates a problem in execution. For a list of possible status codes, see *BullsEye status codes on page 53*.

Program execution

This instruction does not cause robot motion. All data is converted if successful. Otherwise, no data is converted.

Execution in stepwise mode

Forward

Execution when stepping forward is the same as in continuous execution.

Backward

Not supported.

Error handling

Known errors are raised as BullsEye error codes in the optional argument `Status`. These codes may be handled outside the instruction with standard conditional statements. BullsEye error codes are not ERRNO constants handled in a RAPID error handler.

Syntax

```
BETcpExtend
  [ Tool ':='] < expression (PERS) of tooldata >
  [ '\' Change ':=' < expression (IN) of num > ]
  | [ '\' Absolute ':=' < expression (IN) of num > ]
  [ '\' NewExtens ':=' < expression (INOUT) of num > ]
  [ '\' Status ':=' < expression (INOUT) of be_status > ] ';'
```

Related information

	Described in:
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*

6.2.6 BEUpdateTcp - BullsEye update TCP

Usage

`BEUpdateTcp` is used to measure and update the TCP of a tool that has been previously initialized and setup with `BESetupToolJ`.

Basic examples

```
BEUpdateTcp tTestTemp;
```

The tool, `tTestTemp`, will be measured by making a full set of scans, including scans to update the tool orientation.

```
BEUpdateTcp tTestTemp\XYZOnly\Status:=beStatus;
```

As in the previous example, the translational dimensions of the TCP will be updated. The orientation of the TCP, however, will not be scanned and will not be updated. This option is used to decrease the time it takes to update the TCP. The optional argument `Status` provides status codes after the instruction is run.

Arguments

```
BEUpdateTcp Tool [\UserInterface] [\XYZOnly] [\ElapsedTime]
     [\Status] [\TLoad]
```

`Tool`

Data type: `tooldata`

`Tool` is the `tooldata` instance that will be modified. The tool must be set-up using the instruction, `BESetupToolJ`, before `BETcpExtend` can be used.

`[\UserInterface]`

Data type: `string`

An optional user interface may be specified here. Indicate the name of the procedure and the module name.

Example: `"MyUseInt:MyBEUserInter"`. Although the name of the procedure may be altered, the structure of the arguments must follow this model:

```
PROC MyBEUserInter(
  VAR num Response,
  string Header,
  string FkKey{*},
  string TextList{*},
  num DimSet,
  be_status Condition)

  <body of procedure>
ENDPROC
```

`[\XYZOnly]`

Data type: `switch`

If selected, the orientation of the tool will not be measured and will not be updated. Use this switch when it is undesirable to update the orientation, when the tool design makes tool straightening impossible, or when update time must be

Continues on next page

shortened. Update time may be reduced by as much as 50% when using this optional switch.

`[\ElapsedTime]`

Data type: `num`

This parameter will return the overall time required to complete the QuickCheck plus any TCP updating time.

Units: seconds

`[\Status]`

Data type: `be_status`

This optional parameter returns the status code. A status code other than 1 indicates a problem in execution. For a list of possible status codes, see *BullsEye status codes on page 53*.

`[\TLoad]`

Data type: `loaddata`

The `\TLoad` argument describes the total load used in the movement. The total load is the tool load together with the payload that the tool is carrying. If the `\TLoad` argument is used, then the `loaddata` in the current `tooldata` is not considered.

If the `\TLoad` argument is set to `load0`, then the `\TLoad` argument is not considered and the `loaddata` in the current `tooldata` is used instead. For a complete description of the `TLoad` argument, see `MoveL` in *Technical reference manual - RAPID Instructions, Functions and Data types*.

Program execution

The robot will move to the initial position for the tool. A full measurement will be made and the tool will be updated.

Execution in stepwise mode

Execution in stepwise mode is not supported.

Error handling

Known errors are raised as BullsEye error codes in the optional argument `Status`. These codes may be handled outside the instruction with standard conditional statements. BullsEye error codes are not ERRNO constants handled in a RAPID error handler.

Syntax

```
BEUpdateTcp
  [ Tool ':='] < expression (PERS) of tooldata >
  [ '\' UserInterface ':=' < expression (IN) of string > ]
  [ '\' XYZOnly ] < switch >
  [ '\' ElapsedTime ':=' < expression (INOUT) of num > ]
  [ '\' Status ':=' < expression (INOUT) of be_status > ]
  [ '\' TLoad':=' ] < persistent (PERS) of loaddata > ] ';'
```

Continues on next page

Related information

	Described in:
be_device	*be_device - Device data on page 61*
be_scan	*be_scan - Scan data on page 64*
be_tooldesign	*be_tooldesign - Tool design on page 67*
BESetupToolJ	*BESetupToolJ - BullsEye setup tool joint move on page 78*

6.3 Functions

6.3.1 OffsToolXYZ - Offsets tool cartesian

Usage

`OffsToolXYZ` is a function that requires an instance of `tooldata` and an offset as `pos data`. The function will return a new `tooldata` value offset in tool coordinates by the amount specified by the `pos offset`.

Basic examples

```
CONST pos psMyOffset := [1,2,3];
tMyOffsetTool:=OffsToolXYZ (tMyOriginalTool,psMyOffset);
```

The tool is offset 1 mm in X, 2 mm in Y, and 3 mm in Z, relative to the tool coordinates.

Return value

Data type: `tooldata`

The new TCP data.

Arguments

`OffsToolXYZ (Tool Offset)`

`Tool`

Data type: `tooldata`

Original tool.

`[Offset]`

Data type: `pos`

Offset in mm.

Syntax

```
OffsToolXYZ '('
  [ Tool ':=' ] < expression (IN) of tooldata > ','
  [ Offset ':=' ] < expression (IN) of pos > ')'
```

Related information

	Described in:
OffsToolPolar	*OffsToolPolar - Offsets tool cartesian on page 89*
Definition of `pos`	*Technical reference manual - RAPID Instructions, Functions and Data types*

6.3.2 OffsToolPolar - Offsets tool cartesian

Usage

`OffsToolPolar` is a function that requires an instance of `tooldata`, an offset radius as `num` data, and an angle as `num`. The function will return a new `tooldata` value offset in tool coordinates by the amount specified by the offset in the direction specified in the angle.

Basic examples

```
CONST num MyRadius := 3;
CONST num MyAngle := 35;

tMyOffsetTool:=OffsToolPolar (tMyOriginalTool, MyRadius, MyAngle);
```

The tool is offset 3 mm in the X-Y plane. The direction is specified by `MyAngle`.

Return value

Data type: `tooldata`

The new TCP data.

Arguments

`OffsToolPolar (Tool Radius Angle)`

`Tool`

Data type: `tooldata`

Original tool.

`[Radius]`

Data type: `num`

Offset in mm.

`[Angle]`

Data type: `num`

Direction of offset in X-Y plane in degrees.

Syntax

```
OffsToolPolar '('
  [ Tool ':=' ] < expression (IN) of tooldata > ','
  [ Radius ':=' ] < expression (IN) of num > ','
  [ Angle ':=' ] < expression (IN) of num > ')'
```

Related information

	Described in:
OffsToolXYZ	*OffsToolXYZ - Offsets tool cartesian on page 88*

This page is intentionally left blank

7 Spare parts

Introduction

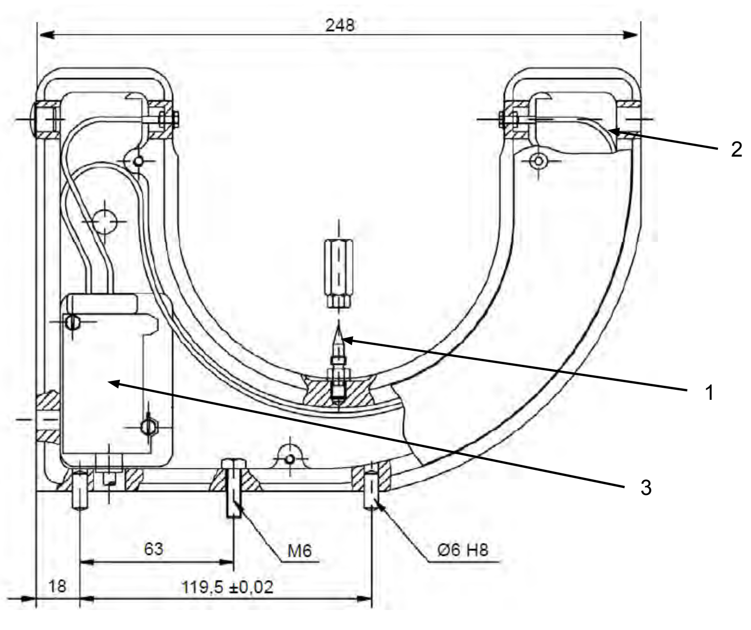
The spare parts list contains all information required for ordering special parts of the TCP gauging unit. Make sure that you give us the precise description of the part which you require.

Required equipment

A pair of special-purpose pliers is essential for fitting the fiber-optic cable for the TCP gauging unit.

TCP gauging unit

Item	Quantity	Article number	Description
		0503060880	Complete for TC-96 BullsEye
1	1	0746335025	Measuring pin
2	1	0746346011	BullsEye fiber-optic, including special tool
3	1	0746346012	Opto-electronic sensor



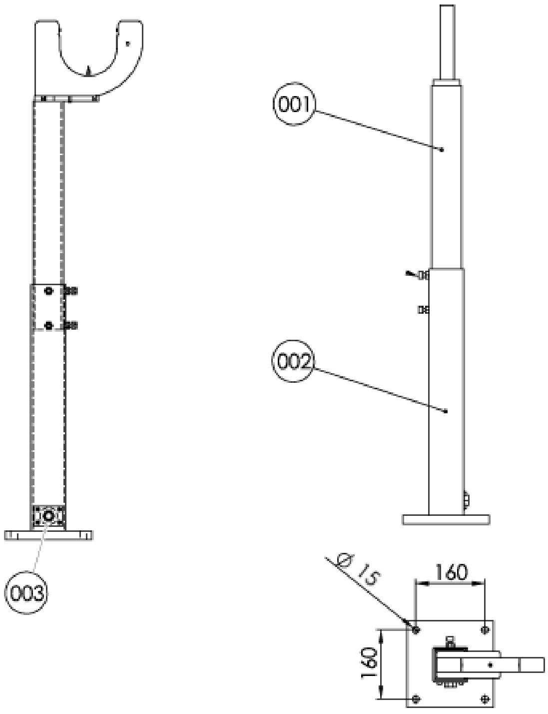
xx1400002302

BullsEye complete

Pos	Article number	Description	Note
-	0506310880	BullsEye stand alone, complete	
001	0505004880	BullsEye upper pole	
002	0505003880	BullsEye pole foot	

Continues on next page

Pos	Article number	Description	Note
003	0503293880	Ext. cable	7 m
003	0503293881	Ext. cable	10 m
003	0503293883	Ext. cable	15 m



xx1400002305

Index

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 Highway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

www.abb.com/robotics

3HAC050989-001, Rev A, en